Teorija informacija i kodova

Neki uvod

Sadržaj

Pojmovi

Informacija – Informacija u savremenom svijetu ima mnogo značenja ali nema preciznu definiciju. Najpreciznije informacija je poruka koju primamo odnosno dio koji iz te poruke razumijemo.
Signal – Signal je fizički koncept – veličina koja nosi informaciju (npr. namagnetisanje, napon, struja, optički signal, zvuk, grafički simboli, slike itd.).
Deterministički sistemi – Sistemi u kojima su sve veličine koje se pojavljuju precizno definisane odnosno kod kojih za poznati ulaz dobijamo precizan izlaz.
Stohastički sistemi – Sistemi u kojima postoji nepoznanica o nekom konceptu odnosno sistemi kod kojih zbog nepotpunog poznavanja procesa saznanje o nekom elementu procesa smanjuje neznanje o njemu odnosno donosi informaciju.
Vjerovatnoća i estimacija vjerovatnoće –Vjerovatnoća je način na koji izražavamo šansu da se neki događaj ili događaji dogode. Procjena (estimacija) vjerovatnoće je postupak kojim se procijenjuju vjerovatnoće nekih događaja, ishoda ili simbola.
Kontinualni (analogni) sistemi (promjenljive) – kontinualni sistemi ili kontinualne promjenljive su oni sistemi/promjenljive koji mogu uzeti beskonačno mnogo vrijednosti iz nekog domena odnosno bilo koju vrijednosti iz predmetnog domena.
Diskretni sistemi (promjenljive) – diskretni sistemi ili diskretne promjenljive su oni sistemi/promjenljive koji mogu da uzmu vrijednosti iz konačnog skupa vrijednosti (diskretnog skupa).
Šum – Slučajni događaj koji se ne može ili veoma teško može opisati determinističkim zakonom.
Uslovna vjerovatnoća – Vjerovatnoća nekog događaja pod uslovom da znamo da se neki događaj već dogodio.
Nezavisni događaji – Ako vjerovatnoća nekog događaja ne zavisi od vjerovatnoće odnosno pojave drugog događaja mi ih nazivamo nezavisnim.
Slučajni proces – predstavlja veličinu čije izvršavanje odnosno vjerovatnoća odvijanja zavisi i od vremena, dakle jedan ishod slučajnog događaja se može pretpostaviti kao jedna funkcija (ne jedna vrijednost).
Ansambl – skup svih mogućih vremenskih funkcija koje odgovaraju nekom slučajnom procesu.
Ergodičnost – ergodični slučajni procesi – slučajni proces je ergodičan ako njegove slučajne vrijednosti su iste računate po vremenu i za čitav ansambl (statističke karakteristike se mogu odrediti iz jednog dovoljno dugog posmatranja).
Stacionarni slučajni proces – podrazumijeva da se funkcija gustine vjerovatnoće ne mijenja u toku trajanja procesa. Ovo se ponekad naziva i stacionarnošću u užem smislu.
Stacionarnost u širem smislu – podrazumijeva da se tokom trajanja procesa ne mijenjaju srednja vrijednost i varijansa.
Kontinualni signal sa diskretnim skupom vrijednosti – signal koji je definisan u svakom trenutku posmatranja ali može da uzme vrijednost koja je iz nekog konačnog skupa.
Digitalni signal – signal koji je definisan samo u određenim vremenskim trenucima a ujedno uzima vrijednosti iz konačnog skupa mogućih.
Redudancija – dupliranje, ponavljanje elemenata radi povećanja pouzdanosti u kontekstu kodne teorije dodavanje bitova poruci kako bi se omogućila detekcija i korekcija pogreške.
Koder izvora – element sistema za prenos informacija koji se koristi za kompresiju poruke u cilju njenog kompaktnijeg zapisa (radi pojeftinjenja prenosnih i memorijskih resursa ...).
Koder kanala – element sistema za prenos informacija koji je zadužen da omogući kasnije dekodiranje poruke na osnovu redudancije koju ovaj blok dodaje u sistem (dodaje se dio bitova tako da se omogući dekodiranje poruke čak i u slučaju grešaka u kanalu).
Kanal – medij za prenos informacija (spojni put, fizički medij koji povezuje izvor i prijemnik informacije, može da bude žica, optički kabal, bežični link, podvodna ultrazvučna komunikacija, ali može biti i papir, CD, magnetna traka, DVD, hard disk).
Dekoder kanala – element sistema za prenos informacija namjenjen za detekciju i ispravljanje grešaka koje su se eventualno dogodile prilikom prenosa poruke u kanalu.
Dekoder izvora – vrši dekodiranje poruke iz koje je koder izvora uklonio redudanciju u oblik koji je pogodan krajnjem korisniku odnosno jednak originalu sa redundancijom.
Huffmanov kod – tip koda koji je razvio David A. Huffman (Dejvid Hafmen) 1952. u sklopu rada na doktoratu. Za poznate vjerovatnoće simbola alfabeta daje najbolju moguću kompresiju (najmanju prosječnu dužinu kodne riječi). Kod posjeduje i odgovarajući postupak dekodiranja.
Kodiranje entropije – postupak u kojem se teži postignuti kodiranje takvo da je dostignuta prosječna dužina kodne riječi što je moguće bliža entropiji sistema

Kodiranje zasnovano na rječniku ili kodnoj knjizi – tokom procesa kodiranja formira se rječnik simbola (kodna knjiga) tokom procesa kodiranja na sličan način bi trebalo da se radi i u postupku dekodiranja.
Lempel Ziv kod – (sa proširenjem Lempel Ziv Welch kod) je kod zasnovan na rječniku (dictionary based). Dobra strana ovog koda je činjenica da se rječnik formira na isti način i na predaji i na prijemu. Kod je optimalan u smislu optimalnog kodiranja Markovljevog sistema nepoznate dužine i nepoznatih vjerovatnoća.
LZW kod – Lempel-Ziv-Welch kod za kodiranje izvora (kompresiju bez gubitaka) zasnovan na rječniku.
RLE kod – Run length encodding kod kojim se kodira uzastopno pojavljivanje simbola u poruci parom simbol - broj uzastopnih pojavljivanja simbola.
Diferencijalni kod – Kodiranje kod kojega se kodira razlika uzastopnih simbola a ne sami simboli (primjenjuje se kod sporopromjenljivih izvora).
Grayov kod – Pomoćni kod koji se formira tako da dva simbola koji imaju susjedne numeričke ekvivalente (recimo 01 i 10) razlikuju se na samo jednoj poziciji (samo jednom bitu).
Kodovi za detekciju greške – kodovi koji prijavljuju primaocu da u poruci postoji greška ali ne ukazuju na prirodu greške.
Kodovi za korekciju greške – kodovi koji su u stanju da bez intervencije primaoca izvrše ispravljanje primljene poruke.
ASCII kod – American Standard Code for Information Interchange (Američki standardni kod za razmjenu informacija) propisuje 8 bita za svaki karakter iz osnovnog skupa karaktera na računaru jedan bit se može koristiti kao bit parnosti radi dobijanja informacije o grešci.
Blok kodovi – kodovi za kodiranje kanala namjenjeni za slučaj kada nam je potrebno ispravljanje greške, osnovna karakteristika im je konstanta dužina kodne riječi.
Pravougaoni kodovi – primitivna grupa kodova za kodiranje kanala; sastoje se iz više poruka "naslaganih" u tabelu gdje krajnji red i krajnja kolona predstavljaju bite parnosti, očekivana jedna greška se nalazi na presjeku indicirane kolone i indicirane vrste.
Trougaoni kodovi – nešto naprednija grupa kodova za kodiranje kanala kod kojih se može zamisliti da je poruka postavljena trougaono, ovdje su biti parnosti na hipotenuzi i svaki od njih kontroliše sopstvenu vrstu i kolonu a greška se ponovo nalazi u presjeku dva indikovana bita parnosti.
Hammingovi kodovi – kao i prethodne grupe kodova pripadaju grupi blok kodova i predstavljaju optimalan sistem za ispravljanje jedne pogreške.
BCH kodovi – Bose-Chaudri-Hocquenghem kod predstavlja sistematski postupak za kodiranje kanala uz sposobnost za ispravljanje više od jedne greške; pripadaju grupi blok kodova.
Konvolucioni kodovi – grupa kodova za korekciju greške (kodiranje kanala) kod kojih se ulazni biti transformišu u izlazne tako da je trenutni bit nastao kao funkcija nekoliko ulaznih bita; generalizacija blok kodova.
Turbo kodovi – grupa kodova za korekciju greške (kodiranje kanala) koji predstavljaju generalizaciju konvolucionih kodova kod kojih se intenzivno koristi interliver prilikom kodiranja.
Interliver – blok koji permutuje ulaznu poruku odnosno svi biti ulazne poruke se nalaze u izlaznoj ali je njihov redosljed izmjenjen (često se koristi za poboljšanje performansi kodiranja).
Deinterliver – vrši suprotnu operaciju od interlivera odnosno vraća simbole poruke u raspored simbola koji je postojao prije interlivera.
Korelacija – statistička veza između pojedinih događaja može se koristiti kao sinonim za zavisnost.
Kriptografija – (grčke riječi krypto – tajno, graphos – pisati) podoblast teorije kodova koja se bavi prenosom tajnih informacija koje su nedostupne neovlašćenim korisnicima.
Kompresija – postupak kojim se originalna informacija sabija tako da produkuje najmanju moguću količinu podataka na nekom meorijskom mediju ili da zauzima komunikacioni kanal najkraće moguće vrijeme, suštinski originalnu poruku zapisujemo sa što je moguće manje simbola (bita). Dvije osnovne tehnike kompresije su kompresija bez gubitaka (originalna informacija se može rekonstruisati bez izmjena – gubitaka); kompresija sa gubicima (originalna informacija se rekonstruiše sa malim ponekad jedva observabilnim izmjenama).
ARQ – Automatic Repeat reQuest sistem prenosa kod kojega je na osnovu koda za detekciju pogreške prijemnik u stanju da detektuje postojanje greške u prenosu i da zahtjeva automatski (bez kontrole operatera) od predajnika ponovno slanje.
Entropija – mjera neodređenosti sistema kod nas mjera neodređenosti slučajne promjenljive odnosno mjera količine informacije (ponekad se naziva samoinformacija ili prosto neodređenost).
Entropija združenih događaja – združena entropija – je ukupna entropija dva ili više događaja.
Uslovna entropija – Entropija događaja ako je poznat ishod nekog drugog događaja koji možda utiče na posmatrani događaj.
Bit, Dit, Nat – jedinice mjere informacije dobijene kada se logaritmi u definicijama entropije evaluiraju sa osnovama 2, 10 i *e* respektivno.
Marginalna entropija – u sistemima koji imaju više od jednog alfabeta (skupa simbola) marginalnom entropijom se naziva entropija pojedinačnog događaja (alfabeta).
Relativna entropija – (naziva se i Kullback-Leiblerova distanca ili diskriminaciona funkcija) mjera razlike između dvije funkcije vjerovatnoće – dvije slučajne promjenljive.
Međusobna informacija – mjera međusobne veze dva događaja – alfabeta – slučajne promjenljive.

Chain rule – pravilo lanca ima mnogo značenja ali ga mi koristimo da izrazimo vezu združenih entropija sa entropijama događaja nižeg reda pored ovoga koristi se kao termin kod određivanje vjerovatnoća uslovnih ili združenih događaja njihovim svođenjem na vjerovatnoće združenih ili uslovnih događaja manjeg reda (sa manje promjenljivih).
Konveksnost/Konkavnost – funkcija je konveksna (udubljena) u nekom intervalu ako je moguće povući pravu koja spaja dvije tačke te funkcije u datom intervalu tako da je uvijek sama funkcija ispod te prave, dok je konkavna (ispupčena) ako prava koja spaja tačke na intervalu funkcije uvijek leži ispod same funkcije. U prvom slučaju drugi izvod funkcije je u posmatranom intervalu je nenegativan dok je u drugom slučaju drugi izvod nepozitivna veličina.
Markovljevi slučajni procesi – su slučajni procesi kod kojih pojava simbola u posmatranom trenutku zavisi od pojave simbola u prethodnim trenucima, ako zavisnost ne postoji riječ je o Markovljevom sistemu nultog reda, ako zavisnost postoji u odnosu na događaj u jednom trenutku (obično neposrednom prethodnom) riječ je o Markovljevom sistemu prvog reda, kod Markovljevog sistema *n*-tog reda vjerovatnoća pojave simbola u posmatranom trenutku zavisi od *n* prethodnih simbola. Ovi sistemi se često nazivaju sistemima sa memorijom tako je Markovljev sistem *n*-tog reda sa memorijom dužine *n*.
Ergodični/neergodični Markovljevi sistemi – Markovljev sistem je ergodičan ako se iz svakog stanja može preći u svako stanje u konačnom broju koraka, dok je neergodičan ako postoji stanje iz kojega se ne može preći u drugo stanje. Ergodičnost procesa se može utvrditi i putem tranzicione matrice, odnosno ako postoji cijeli broj k takav da $\mathbf{P}^k$ gdje je $\mathbf{P}$ tranziciona matrica ima makar jednu nenultu kolonu proces je ergodičan. Proces je regularan ako postoji k takvo da $\mathbf{P}^k$ nema nijedan nulti elemenat. Svaki regularan proces je ujedno ergodičan dok regularnost ergodičnih procesa ne mora da važi.
Proces bez memorije - Markovljev sistem nultog reda gdje vjerovatnoća pojave simbola, odnosno, stanja ne zavisi od prethodnih stanja.
Stanja Markovljevog procesa – Markovljev proces se može nalaziti u većem broju stanja (odnosno simboli alfabeta koji se odnosi na Markovljev proces). Stanja procesa su tranzijentno (napuštanje navedenog stanja ne mora rezultovati povratkom u to stanje u narednom koraku), rekurentno-esencijalno (postoji mogućnost povratka u dato stanje iz svakog stanja u koje se dolazi u dato stanje); periodično (ako postoji cijeli broj *d* koraka nakon kojih se može vratiti u dato stanje, *d* se naziva periodom); ako je d=1 stanje se naziva aperiodičnim dok ako se stanje ne može napustiti naziva se apsorbujuće.
Matrica tranzicije – Matrica koja se koristi za opis Markovljevih sistema (obično prvog reda) odnosno kod opisa komunikacionog kanala. Sastoji se od uslovnih vjerovatnoća da nakon nekog simbola se pojavi neki drugi simbol odnosno da se simbol ulaznog alfabeta u kanal primi kao neki simbol izlaznog alfabeta.
Graf tranzicije – grafička shema koja se koristi za prikaz Markovljevih sistema (sa memorijom) u čvorovima grafa dok su veze između njih usmjerene i predstavljaju vjerovatnoće prelaska između stanja. Alternativan naziv je dijagram stanja.
Uslovna međusobna informacija – međusobna informacija dva ili više događaja u uslovima da je poznat ishod nekog ili nekih događaja.
Trelis - dinamički dijagram stanja Markovljevog sistema (premda se mnogo koristi kod digitalnih komunikacija za različite namjene) kod kojega su prikazani čvorovi i njihova pozicija u vremenu odnosno po taktnim impulsima.
Fanoova nejednakost – Fanoova nejednakost iz ranih 50tih godina dvadesetog vijeka uspostavlja vezu između prosječnog gubitka informacija u kanalu sa šumom i vjerovatnoće greške.
Nejednakost procesiranja podataka – Data processing inequality – Nejednakost (teorema) koja ukazuje da ako je Z nastalo procesiranjem Y gdje je $Z=F(Y)$ a funkcija $F()$ bilo deterministička bilo slučajna funkcija koja ne zavisi od X ne dolazi do uvećanja znanja koje nam Y može saopštiti o X (grubo govoreći ako ne znate neke bliže informacije o događaju X ne procesirajte Y u kojem je sakriveno X jer ne možete dobiti povećanje informacije).
Binarni simetrični kanal – BSC-kanal za prenos binarnih informacija (i na prijemu i na predaji se nalaze biti 0 i 1) sa jednakim vjerovatnoćama pogreške da je poslata i nula i jedinica.
Asimptotska ekviparticiona osobina – (često se označava i kao AEP – P = property) je osobina stohastičkih sistema koja kaže da od svih mogućih realizacija stohastičkog skupa sekvence sa nekom zajedničkom karakteristikom se najčešće dešavaju tzv. tipične sekvence gdje se svaki simbol pojavljuje "očekivani" broj puta.
Tipična sekvenca – skup sekvenci čija je ukupna vjerovatnoća pojavljivanja bliska jedinici. Postojanje ovakve sekvence je posljedica zakona velikih brojeva i asimptotske ekviparticione osobine.
Konvergencija po vjerovatnoći – za neku sekvencu (red) kažemo da konvergira po vjerovatnoći kada je moguće za svako ε koje predstavlja apsolutnu razliku između vrijednosti reda i vrijednosti ka kojoj konvergira odrediti dužinu reda n za koju za svaki član reda koji ima indeks veći od n vjerovatnoća razlike konvergira ka nuli.
Visokovjerovatni skup – čine sekvence iz skupa svih mogućih sekvenci čija ukupna vjerovatnoća je bliska jedinici.
Entropijski odnosi – za sisteme koji nisu sa nezavisnim i na isti način distribuiranim simbolima ponekad se ne može odrediti entropija već se određuje tzv. entropijski odnos kao limes odnosa mjerene združene entropije primljene sekvence sa dužinom sekvence.
Singularni kod - kod kod kojega se dva simbola ulaznog alfabeta preslikavaju u istu kodnu riječ izlaznog alfabeta, ovo je dozvoljeno samo kod kodova sa gubicima, suprotno je nesingularni kod.
Jednoznačno dekodabilni kodovi – kodovi kod kojih svaka kombinacija ulaznih simbola produkuje kod koji se može jednoznačno dekodirati.
Prefiksni (trenutni) kodovi – kodovi koji se mogu jednoznačno dekodirati na osnovu posmatranog i prethodnih simbola.

Koma kodovi – specijalan slučaj trenutnih kodova kod kojih je posljednji simbol ili kombinacija simbola oznaka kraja kodne riječ npr. 00 znači da se završava kodna riječ bez obzira što je prethodilo ovoj kombinaciji.
Optimalni (kompaktni) kod – je kod ili postupak kodiranja koji daje najmanju moguću prosječnu dužinu kodne riječi od svih ostalih jednoznačno dekodabilnih kodova.
Kraftova nejednakost – predstavlja nejednakost koju moraju zadovoljiti prosječne dužine kodnih riječi svakog jednoznačno dekodabilnog koda.
Drvo, stablo, kodno stablo – (algoritamska) struktura koja se koristi za vizuelizaciju koda a često se na osnovu ove strukture kreiraju i algoritmi za kodiranje/dekodiranje.
Šenon-Fano postupak kodiranja – algoritam za kodiranje izvora predložen od strane Šenona i Fanoa u cilju minimizacije prosječne dužine kodne riječi. Kod funkcioniše dijeljenjem kodnih riječi u dva skupa sa približno jednakim vjerovatnoćama i rekurzivno ponavlja prvi korak dok se ne stigne do jednog simbola. Kod se pokazao neuspješnim zbog nejasnog i nepreciznog postupka dijeljenja u podskupove.
RLE kod – (Run Length Encodding) kodiranje pokretne dužine – kod koji mijenja sekvencu istih simbola sa dva podatka dužinom sekvence i simbolom. Pokazuje dobre rezultate kod kodiranja nekih tipova slika ali se obično koristi kao pomoćni.
Diferencijalni kodovi – široka grupa kodova kod kojih se ne kodira direktno neka informacija već se zapravo kodira razlika informacija. Ovo je pogodno kod informacija koje se u nekom domenu sporo mijenjaju. Npr. informacija o temperaturi se sporo mijenja, vrijednost osvjetljaja neke tačke u video formatu se mijenja sporo od jednog do drugog frejma itd.
Aritmetički kodovi – isti imenitelj se koristi za veću grupu kodova kod kojih se bez poznavanja vjerovatnoća simbola i uslovnih vjerovatnoća pojedinih kombinacija simbola kodira poruku na osnovu odgovarajućeg aritmetičkog proračuna. Postoje kodovi zasnovani na određivanju pseudovjerovatnoće (demonstrirani u knjizi) kao i oni koji vrše kodiranje putem određivanja autokorelacionih koeficijenata.
Informacioni kanal – fizički medij preko kojega se prenosi informacija.
Matrica prenosa (tranzitivnosti) kanala – matrica uslovnih vjerovatnoća da se određeni simbol pojavi na izlazu ako je poslat neki simbol na ulazu u kanal. Matrica apstrahuje fizičke osobine kanala u obliku uslovnih vjerovatnoća.
Kapacitet kanala – najveća moguća količina informacije koja se može prenijeti u najpovoljnijim uslovima kroz kanal. Ograničena i zavisi samo od fizičkih karakteristika kanala.
Kodni odnos (code rate, brzina prenosa) – odnos logaritma broja simbola koda sa dužinom kodne riječi.
Gausovski kanal – kanal u kojem je smetnja koja uzrokuje greške u kanalu Gausovski slučajni šum (dobar model praktičnih komunikacionih sistema).
BCD kod – binarno kodirani dekadni brojevi, svaki dekadni broj se kodira četvorobitno sekvencom koja predstavlja binarni ekvivalent datog broja.
JMBG – Jedinstveni matični broj građana. Prevaziđeni sistem kodiranja građana koji postoji u Crnoj Gori i nekim zemljama nastalim na prostoru bivše Jugoslavije. Jedinstveno određuje građanina sa podacima o datumu, mjestu rođenja, polu i redosljedu rođenja. Trinaesta cifra koda je kontrolna i služi za detekciju eventualnih pogreški.
ISBN – International Standard Book Number, broj koji na jedinstven način identifikuje knjigu. Ranije verzije su imale deset simbola ali danas verzije ISBN (ponekad se nazivaju ISBN-13) imaju 13 simbola. Posljednji simbol je kontrolni.
Sindrom – funkcija koja služi da indicira da li je neka primljena poruka neispravna pa čak i gdje se greška dogodila.
Hammingova distanca – broj pozicija na kojima se dvije kodne riječi razlikuju.
Golayev kodovi – blok kodovi specijalne strukture koji su poznati kao jedini nebinarni kompaktni kodovi.
Reed Solomonovi kodovi – varijanta BCH koda koja se uobičajeno koristi kod nebinarnih alfabeta specijalne strukture.
Reed Mullerovi kodovi – varijanta Hammingovih kodova, veoma se brzo realizuju i mogu jednostavno povezati sa digitalnim transformacijama.
Likelihood ratio – odnos vjerodostojnosti – sličnosti – mjera koja se sa svojom logaritamskom varijantom veoma često koristi prilikom dizajna tehnika za procjenu parametara nekog procesa kojega možemo opisati probabilističkim tehnikama.
SISO – Soft-In-Soft-Out dekoder je tip dekodera koji se koristi kod turbo kodova ali i kod nekih drugih tehnika dekodiranja kao što su na primjer kod konvolucionih kodova;
CRC kodovi – Cyclic Redundancy Check su klasa kodova namijenjeni za detekciju ali ne i korekciju većeg broja grešaka. Cilj je onemogućiti pojavljivanje obrazaca nekoliko grešaka koje se ne bi mogle detektovati na alternativne načine.
Granica pakovanja sfera – (poneka se naziva i Hammingovom granicom) je granica koja svjedoči o mogućnosti da se napravi kod sa datim parametrima (date dužine, broja informacioonih bita za dati alfabet) i koja garantuje postojanje navedenog koda (ne ukazuje na konstruktivnost navedenog pristupa niti na to da su granice relativno uske – dobre; kodovi koji ovu granicu zadovoljavaju sa jednakošću nazivaju se perfektnim.
Perfektni kodovi – kodovi koji dostižu granicu pakovanja sfera (Hammingovu granicu) sa jednakošću.
Gillbertov-Varshamova granica – slična Hammingovom dokazuje postojanje koda sa navedenim parametrima (granica nije uska ni konstruktivna); primjenjuje se isključivo kod linearnih kodova; Postoje i linearni kodovi (npr. Goppa) koji prevazilazi granice ove granice.
Singleton granica – primjenjuje se na bilo koji kod ali se koristi da dokaže nepostojanje kodova sa nekim parametrima a ne postojanje kodova.
Katastrofalni kodovi – Kodovi koji se u postupku dekodiranja ne mogu oporaviti.

Sekvencijalno dekodiranje – jedan od postupaka dekodiranja složenijih konvolucionih kodova.
VLSI – Very large scale integration – tip integralnih kola sa veoma visokim stepenom integracije.
Dekodiranje putem majoritetne logike – Tip dekodiranja konvolucionih i sličnih kodova kod kojega se odluka od stanju na jednom primljenom bitu donosi na osnovu više logičkih funkcija nad bitovima a odluka se donosi većinom glasova u odnosu na logičke funkcije.
Viterbijev algoritam – algoritam za otkrivanje skrivenih stanja u sistemu koji se često koristi u različitim oblicima za dekodiranje konvolucionih, turbo i sličnih kodova. Kod je prikazan u obliku trelisa a dekodiranje se vrši na osnovu određivanja optimalne putanje kroz trelis rekurzivnim algoritmom.
Parcijalni najbolji putevi – Kod Viterbijevom algoritma za prethodni bit ne pamti se samo najbolja putanja već sve najbolje putanje do pojedinih stanja u prethodnom koraku pošto u novom bitu može da dođe do ažuriranja putanje i da se najbolja putanja sastoji od bilo kojeg parcijalnog najboljeg puta do stanja u prethodnom bitu i dodatka novog stanja.
Vandermondova matrica – Kvadratna matrica specijalne strukture koja se često pojavljuje u teoriji kodova. Jedna varijanta ove matrice je da su elementi neke vrste različiti a da su u svim ostalim vrstama odgovarajući stepeni te vrste. Dobre osobine ove matricu su jednostavno računanje inverzne matrice i determinante.
WLAN – bežična lokalna mreža – wireless local array network – mreža u kojoj je neophodno implementirati sofisticirane algoritme korekcije pogreški koji uključuju konvolucione i turbo kodove.
NASA – Nacionalna vazduhoplovna i svemirska administracija SAD – agencija koja je bila motor razvoja brojnih kodnih standarda.
DVB-T – digitalni prenos videa preko zemaljskih prenosnih puteva, standard komunicanje koji uključuje turbo kodove.
CDMA – dode division multiple access – tip digitalne modulacije koji koristi turbo kodove.
UMTS – univerzalni mobili telekomunikacioni sistemi evropsko rješenje u telekomunikacijama, standard koji koristi turbo kodove.
Bipolarni signal – binarne poruke gdje je jedno od stanja predstavljeno negativnim nivoom a drugo pozitivnim.
Unipolarni signal – binarne poruke gdje je jedno od stanja predstavljeno nulom a drugo pozitivnim nivoom.
Burst greške – situacija kada se u poruci generiše veliki broj uzastopnih grešaka.
MDS – minimum distance separating grupa kodova koji dostiži Singletonovu granicu sa jednakošću.

Oznake

$p(A)$	Za diskretne događaje/simbole vjerovatnoća događaja A, simbola A, a za kontinualne funkcija gustine promjenljive.
$p_i=p(x_i)$	Vjerovatnoća pojave događaja x_i.
p	Vjerovatnoća binarnog događaja.
$\mathbf{X}$	Alfabet (skup) objedinjuje sve moguće simbole koji se mogu pojaviti.
μ_X	Srednja vrijednost alfabeta (podrazumijeva se da su simbolima alfabeta pridružene numeričke vrijednosti x_i).
$\mathrm{E}\{\}$	Srednja vrijednost – matematičko očekivanje.
μ	Srednja vrijednost (oznaka uvedena kod Gausovske slučajne promjenljive).
$\hat{p}_i$	Procjena (estimacija) vjerovatnoće slučajne promjenljive (simbol ^ se obično koristi da naglasi procjenu neke veličine).
k_i	broj pojavljivanja simbola x_i u sekvenci.
$P(A\|B)$	Uslovna vjerovatnoća da se dogodio događaj A pod uslovom da se dogodio događaj B.
$P(X=x,Y=y)$	Združena vjerovatnoća da je događaj X uzeo vrijednost i da je događaj Y uzeo vrijednost y.
$P(A,B)$	Združena vjerovatnoća događaja A i B (označavamo i kao P(dogodilo se A i B)).
$\sigma_{\mathbf{x}}^2$	Varijansa simbola alfabeta $\mathbf{X}$.
σ_X	Standarna devijacija simbola alfabeta $\mathbf{X}$.
K	Ukupan broj realizacija nekog slučajnog procesa kojem odgovara neki alfabet.
$\subset$	Podskup.
$p(x)$	Funkcija gustine raspodjele (ponekad se naziva marginalna raspodjela u sistemima sa više promjenljivih).
$P(x)$	Funkcija raspodjele.
$p(y\|x)$	Uslovna vjerovatnoća događaja y u zavisnosti (pod uslovom da se dogodio) događaj x.
$p(x,y)$	Vjerovatnoća združenog događaja.
$H(X)$	Entropija.
$H(X,Y)$	Združena entropija dva skupa (događaja) X i Y.
$H(Y\|X)$	Uslovna entropija.
$H(Y\|X=x)$	Uslovna entropija za fiksnu vrijednost uslovnog događaja.
$S_{xx}(\omega)$	Spektralna gustina snage slučajnog procesa.
$R_{\xi\xi}(t_1,t_2)$	Autokorelaciona funkcija koja može biti definisana kao $R_{\xi\xi}(t_1,t_2)=\mathrm{E}\{x(t_1)x^*(t_2)\}$.
$S_{\xi\xi}(\omega)$	Spektralna gustina snage slučajnog događaja ξ.

$P(x)$	Označava funkciju raspodjele slučajne promjenljive odnosno vjerovatnoću da slučajna promjenjljiva uzme vrijednosti koja je manja ili jednaka od x.
$p(x)$	Funkcija gustine vjerovatnoće nekog slučajnog događaja (izvod funkcije raspodjele $p(x)=P'(x)$), premda se može definisati i za diskretne slučajne događaje obično ima smisla koristiti je samo za kontinualne slučajne promjenljive.
$\mathbf{N}(\mu,\sigma^2)$	Označava Gausovsku raspodjelu slučajne promjenljive sa srednjom vrijednošću μ i varijansom σ^2 (funkcija gustine raspodjele ove slučajne je $p(x)=\exp(-(x-\mu)^2/2\sigma^2)/\sqrt{2\pi}\sigma$).
ξ,ζ	Korišćeno za označavanje kontinualnih slučajnih promjenljivih sa jednom i sa dvije promjenljive.
$P_{\xi\zeta}(x,y)$	Funkcija raspodjele dvije slučajne promjenljive.
$p_{\xi\zeta}(x,y)$	Funkcija gustine raspodjele dvije slučajne promjenljive.
$\xi(t,s)$	Slučajni proces, stohastička veličina koja zavisi i od vremena.
$P_{\xi}(x,t)$	Funkcija raspodjele slučajnog procesa.
$A\{x(t)\}$, $\overline{x(t)}^T$	Srednja vrijednost slučajnog procesa unutar trajanja za datu realizaciju.
$\lim_{x\to b}$	Limes granična vrijednost koju neka funkcija uzima u okolini tačke $x=b$ u kojoj funkcija može ali i ne mora biti definisana.
$X_T(\omega)$	Fourierova transformacija slučajnog događaja posmatranog u limitiranom vremenskom intervalu širine T.
$\xi(s,t)$	Slučajni proces čiji je ishod jedna moguća vremenska funkcija.
P_{xx}	Snaga slučajnog procesa.
$u(x)$	Heavisajdova odskočna funkcija koja ima vrijednost 1 za $x\geq 0$ i 0 drugdje.
p,q	Često vjerovatnoće kod binarnih događaja označavamo sa p i q gdje je p vjerovatnoća jednog ishoda a q vjerovatnoća drugog ishoda (često su ti ishodi uspješan prenos i greška u prenosu).
$p(\xi_1,...,\xi_n\|\zeta_1,\ldots,\zeta_m)$	vjerovatnoća združenog događaja $\xi_1,...,\xi_n$ pod uslovom da se združeni događaj $\zeta_1,\ldots,\zeta_m$ dogodio.
e_i	Greška (u nekim primjerima u pitanju je greška na bitu a u nekima je vjerovatnoća na datom bitu što se lako može shvatiti iz konteksta).
$E_p[g(X)]$	Matematičko očekivanje slučajne promjenljive $g(X)$ gdje su simbolima iz osnovnog alfabeta X pridružene vjerovatnoće $p(X)$.
$H(X,Y)$	Entropija združenog događaja (skupova, alfabetova) X i Y.
$H(X_1, X_2, ..., X_n)$	Združena entropija više događaja.
$H(X)$	Entropija događaja.
$H(p)$	Entropija binarnog događaja kod kojega se jedan od simbola (ishoda) pojavljuje sa vjerovatnoćom p.
$p_X(x)=\Pr\{X=x\}$	Vjerovatnoća simbola x iz alfabeta X.
$H(Y\|X)$	Entropija događaja Y pod uslovom da se dogodio (da je poznat) ishod događaja X.
$H(Y\|X=x)$	Entropija događaja Y pod uslovom da se zna da je ishod događaja X bio $X=x$.
$D(p\,\|\|\,q)$	Relativna entropija, Kullback Leiblerova distanca ili diskriminaciona promjenljiva koja mjeri razliku između raspodjela p i q odnosno između slučajnih promenljivih koje predstavljaju te raspodjele.
$I(X;Y)$	Međusobna informacija (zavisnost) dvije slučajne promjenljive X i Y.
$f'(x)$, $f''(x)$	Prvi i drugi izvod funkcije
$f'''(x)$, $f^{(p)}(x)$	Treći i opšti p-ti izvod funkcije.
$p(s_{j+1}\|s_1,s_2,...,s_j)$	Markovljevi sistemi opisani su uslovnim vjerovatnoćama pojave simbola u j+1 trenutku u zavisnosti od pojave simbola u prethodnih j trenutaka (ovo je Markovljev sistem j-tog reda). Ponekad se kaže i proces sa memorijom j-tog reda.
$\mathbf{p}=[p_1,p_2,...,p_N]^T$	vektor vjerovatnoća pojave simbola alfabeta u nekom trenutku, koristi se kod Markovljevih sistema.
$\mathbf{P}$	Matrica tranzicije koja se koristi kod opisa kanala odnosno kod Markovljevih sistema (posebno onih prvog reda), sastoji se od uslovnih vjerovatnoća da nakon jednog simbola se pojavi drugi odnosno da se simbol koji je poslat na prijemnoj strani primi kao neki simbol izlaznog alfabeta.
$I(X;Y\|Z)$	Uslovna međusobna informacija događaja X i Y pod uslovom da je poznat ishod događaja Z.
P_{ij}	Elementi matrice tranzicije – uslovne vjerovatnoće i je indeks ulaznog simbola a j je indeks izlaznog simbola.
$X\to Y\to Z$	Označava Markovljev lanac odnosno niz slučajnih procesa koji redom zavise jedan od drugog.
$\hat{X}$	Procjenjena (estimirana) vrijednost.
P_e	Česta oznaka za vjerovatnoću greške
ε, δ	Male vrijednosti koje teže nuli.
$P[\|X_n-X\|<\varepsilon]>1-\delta$ za $n>n_0$	Formalni iskaz konvergencije po vjerovatnoći.
S_n	Oznaka srednje vrijednosti niza.
$\partial/\partial x$	Parcijalni izvod – izvod funkcije više promjenljivih po promjenljivoj x.

$p(x_1,...,x_n)$, $p(X_1,...,X_n)$	Vjerovatnoća sekvence od n simbola.
$A_{\varepsilon}^{(n)}$	Oznaka tipičnog skupa.
$B_{\delta}^{(n)}$	Visokovjerovatni skup.
χ^n	Za alfabet označen sa χ predstavlja skup svih mogućih sekvenci dužine n.
$H(\chi), H'(\chi)$	Nazivaju se entropijskim odnosima predstavljaju limese entropija slučajnih procesa gdje pojedini simboli nisu nezavisni i na isti način distribuirani.
λ	Lagranžev množilac, koristi se prilikom određivanja minimuma i maksimuma funkcije uz neko ograničenje.
$l(x)$	Dužina kodne sekvence.
$\square$	Označava da funkcije (redovi) na lijevom i na desnoj strani znaka jednakosti imaju isti red konvergencije.
C^*	Skup svih mogućih nadovezanih sekvenci kodnih riječi na izlazu iz kodera. Ako je ulazna sekvenca $X_1X_2X_3...X_n$ elementi skupa svih mogućih sekvenci na izlazu se označavaju kao $C(X_1,X_2,...,X_n)$ gdje su nadovezane kodne riječi označene kao $C(X_1)C(X_2)...C(X_n)$.
$\|A\|$, D	Broj članova skupa – broj kodnih riječi u skupu.
$H_D(X)$	Entropija računata sa logaritmom za osnovu D.
L^*	Optimalna dužina koda.
$\lceil\ \rceil$	Najmanji cijeli broj veći ili jednak argumentu.
R	Kodni odnos (kodna brzina).
C	Kapacitet kanala.
α	Vjerovatnoća brisanja, odnosno vjerovatnoća da prijemnik ne može da donese odluku da li je signal sa predaje uopšte prisutan.
P	Matrica prenosa – tranzicije kanala.
λ_i, $\lambda^{(n)}$	Vjerovatnoće greške, oznaka se koristi kod izvođenja druge Šenonove teoreme (teoreme o kapacitetu kanala). $\lambda^{(n)}$ je maksimalna vjerovatnoća greške po simbolu.
$Q(x)$, $\Phi(x)$	Vjerovatnoća greške i komplementarna vjerovatnoća greške kod Gausovog šuma.
$\cup$	Unija predstavlja elemente skupa koji pripadaju bilo kojem od skupova koji su operandi u ovoj operaciji.
r, c, X, **Y**	Oznake vektora koji predstavljaju kodne riječi u dijelu teorije kodova
H	Kontrolna matrica koda
S	Sindrom kodne riječi (ukazuje na činjenicu da li je kodna riječ pravilno prenesena i ako nije gdje se greška dogodila).
e	Riječ u kojoj jedinice označavaju poziciju greške.
G	Generatorska matrica koda.
i, $i(x)$	Informaciona riječ – može biti prikazana u obliku vektora ili polinoma.
$c(x)$	Kodna riječ zadata u obliku polinoma.
$e(x)$	Riječ koja predstavlja grešku zadatu u obliku polinoma.
P	Matrični minor koji se pojavljuje u kontrolnoj i generatorskoj matrici koda.
$\mathbf{I}_k$	Jedinična matrica dimenzija kxk.
$g(x)$	Generatorski polinom koda (kod sa kojim množimo informacioni polinom da bi dobili kodni polinom).
a, α	Korjeni prostog polinoma.
$S(x)$, S_j	Sindromski polinom i koeficijenti sindromskog polinoma kod BCH koda.
$l(x)$, $u(x)$	Polinomi lokator pogreške i polinom vrednovanja pogreške kod BCH kodiranja.
gcd	Najveći zajednički djelilac (greatest common divisor) dva broja.
$L()$	Oznaka za log-likelihood odnos.
EL_m	Generatorska matrica Reed Mullerovog koda.
$\mathbf{H}_k$	Hadamardova transformaciona matrica.
$\mathbf{F}_q^n$	Polje sa q simbola alfabeta kojeg čine vektori dužine n (ili polinomi n-1 stepena).
KK	Konvolucioni kod.
deg()	degree – stepen polinoma kod blok i konvolucionih kodova.
G	Generatorska matrica konvolucionog koda
M	Memorija konvolucionog koda.
N	Ograničena dužina konvolucionog koda.
G_L	Generatorska matrica konvolucionog koda sa ograničenjem L.
L	Oznaka ograničenja konvolucionog koda.
$\tilde{\mathbf{d}}(x)$	Oznaka primljenog koda preko polinoma
$L()$	Logaritamski odnos sličnosti (vjerodostojnosti) – log likelihood odnos.

d_{free}	Slobodna distanca – mininalno moguće rastojanje između dvije sekvence na izlazu iz konvolucionog kodera.
sign	Funkcija znaka, za pozitivne vrijednosti jednaka 1, negativne vrijednosti argumenta -1, i nula za nulti argument.
min, max	Funkcije koje daju minimum odnosno maksimum argumenata.
$\oplus$	Ekskluzivno ili – sabiranje po modulu 2.

Poglavlje I
PREGLED ELEMENATA TEORIJE VJEROVATNOĆE

1.1. Osnovni elementi teorije vjerovatnoće

Kratko ćemo proći kroz osnovne elemente teorije vjerovatnoće koji će biti korišćeni u knjizi. Napomenimo da ćemo ovdje već pomenuti neke pojmove kao što su **signal** i **informacija** koji će biti definisani u narednoj lekciji. Pretpostavimo da u električnom kolu jedino stanje koje se može pojaviti u nekom čvoru je 5V. To stanje onda nije neodređeno i ako apriori znamo da je u toj tački napon 5V mjerenje ne moramo ni obaviti. U rječniku teorije informacija kaže se da ovo stanje ne nosi informaciju.

Ako postoji mogućnost da u pomenutoj tački postoji vrijednost 0V i 5V postoji potreba da se izvrši mjerenje i da se "ukine neodređenost" o vrijednosti stanja. Sada vrijednosti 0V možemo pridružiti vrijednost 0 i vrijednost 5V pridružiti vrijednost 1 u binarnoj aritmetici. Za ovu operaciju se kaže da smo kodirali stanje (poruku).

Posmatrajmo događaj bacanja kocke. Ishod toga događaja nama je unaprijed nepoznat i na osnovu dobijenog rezultata mi dobijamo određenu informaciju. Ta informacija je zasigurno veća od one koju smo dobili nakon što smo se upoznali sa ishodom događaja bacanja novčića jer kod novčića postoji neizvjesnost u izboru samo dvije moguće pojave dok je kod kocke ta neizvjesnost veća i odnosi se na 6 mogućih ishoda.

Dalje, posmatrajmo mogućnost da imamo trideset mogućih naponskih stanja 0V, 1V, 2V, ... 29V kojima možemo pridružiti slova naše abecede A, B, C, ... Ž. Sada zamislimo da nam neko mijenjajući stanje u tački kola prenosi poruku. Mi tu poruku možemo tumačiti mjerenjem napona u pojedinim trenucima. Ova operacija se uslovno može zvati dekodiranje.

Da bi se način "prenosa" optimizovao (poboljšao) neophodno je odrediti prirodu poruke koja se prenosi. Znači mi očekujemo neku poruku nepoznate sadržine iz skupa svih poruka koje se na našem jeziku mogu izgovoriti. Da bi tačno kvantifikovali mogućnost pojavljivanja nekog slova koristimo pojam vjerovatnoće. Tako možemo reći da je vjerovatnoća pojavljivanja slova A u ukupnom skupu slova u našem jeziku neka vrijednost. Očigledno je ta vrijednost veća nego za pojavljivanje slova Š. Da bi način mjerenja (odnosno prenosa informacija) podesili pravilno trebalo bi da nam mjerna skala oko vrijednosti napona koja daje slovo A bude preciznija nego ona oko slova Š. Isto se dešava kod prenosa informacija. Dakle, poznavanje vjerovatnoće pojavljivanja nekog slova u poruci opredjeljuje performanse sistema. Sistem koji govori o porukama koje su unaprijed nepoznate sadržine je "stohastički" ili slučajan. Suprotno od ovoga su deterministički sistemi. Deterministički sistemi ne nose poruke i informacije ali su lakši za analizu pa se stoga i u mnogim oblastima koje se bave porukama koriste kao model. Teorija informacija i kodova međutim polazi od slučajnih poruka sa određenom vjerovatnoćom pojavljivanja određenog simbola.

Drugi, veoma značajan, razlog za uvođenje pojmova teorije vjerovatnoće u teoriju informacija je postojanje određenih vrsta smetnji koje se pri slanju, prenosu, prijemu i razumjevanju poruke mogu dogoditi. Sve ove smetnje se na nekom nivou mogu opisati determinističkim zakonima. Međutim, za relativno preciznu analizu dovoljne su procjene procesa koji dovode do smetnji. Takve procjene ćemo nazivati šumovima. Važno je napomenuti da neke vrste šumova potiču od termičkog kretanja elektrona, druge su izazvane atmosferskim procesima ili ljudskim aktivnostima. Međutim, mi u ovom kursu nećemo razmatrati šum na fizičkom nivou već samo kao model odnosno taj model će biti simplifikovan do nivoa da ćemo razmatrati samo na nivou vjerovatnoća sa kojima neke za nas apstraktne fizičke pojave utiču na sistem za prenos informacija.

1.2. Procjena vjerovatnoće

Posmatrajmo sada sljedeći slučajni događaj izbor jednog elementa iz konačnog skupa **X** koji se sastoji od elemenata $x_i \in \mathbf{X}$, $i=1,2,...,N$. Ovaj skup ćemo zvati alfabetom. Ponavlja se veliki broj "izbora" iz skupa **X**. Neka je svaki x_i element "izvučen" k_i puta. Tada se može reći da je procjena vjerovatnoće pojavljivanja nekog elementa iz skupa **X** jednaka:

$$\hat{p}_i = \frac{k_i}{k_1 + k_2 + ... + k_N} = \frac{k_i}{\sum_{i=1}^{N} k_i} = \frac{k_i}{K} \qquad \sum_{i=1}^{N} \hat{p}_i = 1$$

gdje je: $k_1+k_2+...+k_N=K$. Ako je K veoma veliki broj (praktično beskonačan) nije više riječ o procjeni vjerovatnoće pojavljivanja nekog elementa već o vjerovatnoći. Uvijek je sporno koliko je dobra procjena vjerovatnoće događaja. Naime, za skup slova našeg jezika bi potpuno valjana procjena obuhvatila sve što je ikada rečeno i napisano na našem jeziku. Čak ni takva procjena ne može nam govoriti o kvalitetu mjerenja (određivanja neke poruke) jer npr. u poruci može biti izostavljen neki često ponavljani karakter. U daljem tekstu mi ćemo podrazumjevati da su nam vjerovatnoće (a ne procjene) poznate. U nekim slučajevima ćemo se vratiti na problem procjenjivanja vjerovatnoća.

Zapamtite. Suma vjerovatnoća svih događaja je 1. Vjerovatnoća nemogućeg događaja je 0.

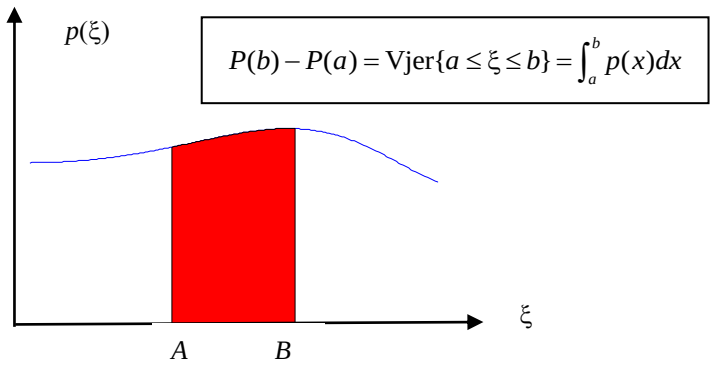
Ako su elementi skupa **X** zapravo numerički kvantifikovani (brojevi) tada se srednja vrijednost definiše kao:

$$\mu_{\mathbf{X}} = \sum_{i=1}^{N} x_i p_i$$

Druga važna karakteristika koja se koristi zapisivanje slučajnih procesa je varijansa:

$$\sigma_{\mathbf{X}}^2 = \sum_{i=1}^{N} p_i (x_i - \mu_x)^2 = \sum_{i=1}^{N} p_i x_i^2 - \mu_x^2$$

gdje je $\sum_{i=1}^{N} p_i x_i^2$ srednja kvadratna vrijednost. Srednja vrijednost predstavlja mjeru najvjerovatnijeg događaja dok varijansa predstavlja mjeru odstupanja od tog događaja.



Vjerovatnoća da slučajna promjenljiva uzme neke vrijednosti u skupu **Y** koji je podskup skupa **X** je jednaka:

$$p_{\mathbf{Y} \subset \mathbf{X}} = \sum_{i=1, x_i \in \mathbf{Y}}^{N} p_i$$

Mi ćemo informacije posmatrati skoro isključivo kao diskretne slučajne procese (slučajne procese koji mogu uzeti vrijednost iz konačnog skupa vrijednosti). Pored ovakvih postoje i kontinualne slučajne promjenljive koje sa određenom vjerovatnoćom mogu uzeti neku vrijednost iz kontinualnog intervala. U tom slučaju nema smisla govoriti o vjerovatnoći da događaj uzme vrijednost 2, jer događaj može uzeti i vrijednost 2.01, 2.0000001, itd, odnosno vjerovatnoća "uzimanja" jedne diskretne vrijednosti iz posmatranog skupa može se smatrati jednakom nula. Stoga se uvodi pojam funkcije raspodjele. Ako je ξ slučajna promjenjiva koja može uzeti bilo koju vrijednost na intervalu od $-\infty$ do ∞ tada se funkcija raspodjele definiše kao vjerovatnoća da događaj uzme neku vrijednosti koja je manja od vrijednosti x:

$$P(x) = \text{Vjer}\{\xi \le x\}$$

Očigledno važe slijedeće relacije $P(-\infty) = 0$ i $P(\infty) = 1$ kao i činjenica da je funkcija raspodjele neopadajuća funkcija (kako vrijednost argumenta x raste tako ova funkcija raste ili uzima prethodnu vrijednost). Ova činjenica se može matematički zapisati kao $x_1 \le x_2 \Rightarrow P(x_1) \le P(x_2)$. Druga izuzetno značajna funkcija koja se definiše kod kontinualnih slučajnih promjenljivih je funkcija gustine vjerovatnoće:

$$p(x) = \frac{dP(x)}{dx} = P'(x)$$

Ova funkcija ima sljedeće značajne osobine:

$$p(x) \ge 0 \qquad \int_{-\infty}^{\infty} p(x)dx = 1$$

Vjerovatnoća da kontinulna slučajna promjenljiva ξ uzme vrijednost iz intervala od a do b je jednaka:

$$P(b) - P(a) = \text{Vjerovatnoca}\{a \le \xi \le b\} = \int_a^b p(x)dx$$

Srednja vrijednost i varijansa se kod kontinualnih slučajnih promjenljivih definišu kao:

$$\mu_x = \int_{-\infty}^{\infty} xp(x)dx \qquad \sigma_x^2 = \int_a^b x^2 p(x)dx - \mu_x^2$$

Već smo napomenuli da praktično u okviru našeg kursa nećemo razmatrati informaciju koja može uzeti vrijednosti unutar kontinualnog skupa podataka. Međutim, smetnje koje se neumitno pojavljuju unutar prenosnog puta (kanala) ćemo smatrati da mogu uzeti bilo koju vrijednost iz nekog intervala pa ćemo takve smetnje modelovati kontinualnim slučajnim promjenljivim. Smetnje ćemo nazivati u daljem tekstu - šumovi. Najpoznatiji tip šuma je Gausovski koji se modeluje funkcijom gustine raspodjele:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}$$

gdje je μ srednja vrijednost procesa a σ^2 varijansa šuma. Gausov šum se često označava kao $\mathbf{N}(\mu,\sigma^2)$. Napomenimo da procese odnosno šumove koji imaju srednju vrijednost nula nazivamo bijelim (Gausov šum je bijeli za μ=0). Gausov šum je posljedica djelovanja više nezavisnih izvora smetnji (ti izvori su uglavnom determinističke prirode npr. kretanje elektrona, ili atmosferska dešavanja, varničenja ali sa stanovišta nemogućnosti razmatranja svakog elektrona u okruženju mi te promjene modelujemo šumovima određene gustine raspodjele). Poznata relacija iz vjerovatnoće i statistike je centralna granična teorema po kojoj suma velikog broja nezavisnih slučajnih procesa teži procesu sa Gausovskom raspodjelom čije su srednja vrijednost i varijansa jednake sumi srednjih vrijednosti i varijansi pojedinačnih procesa.

Drugi relativno često korišćeni model šuma je sa uniformnom raspodjelom na intervalu od a do b:

$$p(x) = \frac{1}{b-a} \quad \text{za} \quad x \in [a,b] \quad \text{i} \quad p(x) = 0 \quad \text{drugdje}$$

U teoriji vjerovatnoće postoje i kombinovane slučajne promjenljive koje mogu uzeti osobine i kontinualnih i diskretnih slučajnih promjenljivih. Postoji još jedna značajna tema iz vjerovatnoće koju ćemo ovdje pomenuti. To su uslovni događaji. Posmatrajmo pojavu slova S u našem jeziku. Možemo je opisati vjerovatnoćom pojavljivanja koju ćemo procjeniti na osnovu pojavljivanja ovog slova u ukupnom broju slova u jeziku. Međutim, uočimo da se poslije ovog slova veoma rijetko (praktično nikad u istoj riječi) ne pojavljuje slovo Š dok se veoma često pojavljuje slovo T. To motiviše opisivanje slučajnih događaja sa uslovnim vjerovatnoćama: kolika je vjerovatnoća nekog događaja pod uslovom da se neki drugi događaj već javio. Kod nas će to uglavnom biti kolika je vjerovatnoća neke informacije (odnosno informacionog simbola pod uslovom da je u prethodnom trenutku bio neki simbol). Alternativno ovo može biti vjerovatnoća da ako pošaljemo neki simbol alfabeta on bude zbog greške u sistemu prepoznat kao neki drugi simbol ili možemo da primimo simbol na izlazu iz komunikacionog sistema i da se pitamo kolika je vjerovatnoća da je ovaj simbol vjerodostojno primljen. Dakle, moramo konstatovati značaj pojma uslovne vjerovatnoće. Ako dva događaja (dvije informacije, dva simbola) ne utiču jedan na drugi riječ je o nezavisnim događajima. Vjerovatnoća događaja A ako je nastupio događaj B se označava kao $P(A|B)$ i čita se kao vjerovatnoća A u zavisnosti od B. Dakle, vjerovatnoća ovog događaja je jednaka vjerovatnoći da su se dogodili i događaj A i događaj B kroz vjerovatnoća da se dogodio događaj B:

$$P(A|B) = \frac{P(\text{dogodili se } A \text{ i } B)}{P(B)}$$

Ako su A i B nezavisni tada važi $P(\text{dogodili se } A \text{ i } B) = P(A)P(B)$ odnosno $P(A|B) = P(A)$. Događaj dogodili se i A i B se često obilježava kao AB. Uočimo dalje da važi:

$$P(B \mid A) = \frac{P(AB)}{P(A)} \qquad \text{odnosno} \qquad P(B \mid A) = \frac{P(A \mid B)P(B)}{P(A)}$$

Ako se u nekom trenutku može prenositi set simbola B_i, i=1,...,N tada je vjerovatnoća događaja A jednaka:

$$P(A) = \sum_{i=1}^{N} P(B_i)P(A|B_i)$$

Pa se Bayesovo pravilo može svesti na:

$$P(B_j \mid A) = \frac{P(A \mid B_j)P(B_j)}{\sum_{i=1}^{N} P(B_i)P(A \mid B_i)}$$

i opisuje aposteriori vjerovatnoću (vjerovatnoću ako već znamo da se dogodio događaj A kolika je vjerovatnoća da ga je uzrokovao događaj B_j).

Primjer. Pokušaćemo da objasnimo neke od uvedenih pojmova kroz jedan primjer. Pretpostavimo da raspolažemo sa dva novčića od po 1E. Jedan novčić je vaš i predstavlja fer novčić gdje sa vjerovatnoćom 0.5 može da se desi da prilikom bacanja dobije se pismo (ishod 1) i glava (ishod 0). Pored vas u igri učestvuje još jedan igrač sa novčićem od 1E. Njegov novčić nije fer već sa vjerovatnoćom 0.7 pada na pismo (ishod 1), a sa vjerovatnoćom 0.3 pada na glavu (ishod 0). Formirati vjerovatnoće združenog događaja prilikom bacanja ova dva novčića.

Neka je A ishod bacanja prvog novčića sa vrijednostima u alfabetu A={0, 1} sa asociranim vjerovatnoćama {0.5, 0.5} , to jest $P(A=0)$=0.5 i $P(A=1)$=0.5. Slično kod drugog događaja B koji je definisan preko istog binarnog alfabeta: $P(B=0)$=0.3 i $P(B=1)$=0.7. Pošto su događaji bacanja dva novčića nezavisni to važi da je vjerovatnoća združenog događaja jednaka proizvodu vjerovatnoća $P(A=0,B=0)=P(A=0)P(B=0)$=0.15, $P(A=0,B=1)=P(A=0)P(B=1)$=0.35, $P(A=1,B=0)=P(A=1)P(B=0)$ =0.15, $P(A=1,B=1)=P(A=1)P(B=1)$=0.35. Uvedimo sada događaj C koji predstavlja sumu ishoda dobijenih bacanjem dva novčića. Alfabet koji odgovara ovom slučaju je C={0, 1, 2} vjerovatnoće ova dva događaja su: {0.15, 0.5, 0,35}. Sada smo u potpunosti u stanju da uvedemo koncept uslovne vjerovatnoće. Pretpostavimo da smo bacili prvi novčić i da znamo njegov ishod interesuje nas kolika je vjerovatnoća da dobijemo pojedini ishod događaja C. Recimo $P(C=2|A=1)$ je jednako $P(C=2,A=1)/P(A=1)$ odnosno ovo je identično $P(B=1)$ da na drugom novčiću padne vrijednost 1 dok $P(C=2|A=0)=0$ jer je nemoguć događaj da se samo sa jednim novčićem dobije ishod 2. Pregled odgovarajućih uslovnih vjerovatnoća: $P(C=2|A=1)=P(C=2,A=1)/P(A=1)=P(B=1)=0.7$, $P(C=2|A=0)$ $=P(C=2,A=0)/P(A=0)=0$, $P(C=1|A=1)=P(C=1,A=1)/\ P(A=1)=P(B=0)=0.3$, $P(C=1|A=0)=$ $P(C=1,A=0)/P(A=0)=P(B=1)=0.7$, $P(C=0|A=1)=P(C=0,A=1)/P(A=1)=0$, $P(C=0|A=0)=$ $P(C=0,A=0)/$ $P(A=0)=P(B=0)=0.3$. Predmetne rezultate je zgodno prikazati tabelarno.

$P(C,A)$	A=0	A=1	$P(C)$
C=0	0.15	0.00	0.15
C=1	0.35	0.15	0.50
C=2	0.00	0.35	0.35
$P(A)$	0.50	0.5	1.00

Iz tabele združenih vjerovatnoća vidimo važnu činjenicu da ako sumiramo združenu vjerovatnoću $P(C,A)$ po jednoj od dvije veličine (recimo po A) u tom redu ili koloni dobijamo vjerovatnoću druge slučajne promjenljive (u ovom slučaju C). Zbog toga što se ove vjerovatnoće pišu na margini tabele nazivaju se ponekad marginalnim.

Tabelu uslovnih vjerovatnoća $P(C|A)$ možemo dobiti tako što podijelimo svaku vrijednosti iz tabele združenih vjerovatnoća sa marginalnom vjerovatnoćom koja je upisana u koloni sa marginalnom $P(A)$:

$P(C\|A)$	A=0	A=1
C=0	0.3	0
C=1	0.7	0.3
C=2	0	0.7

Potpuno analogno prethodnom možemo da sračunamo uslovnu vjerovatnoću $P(A|C)$ koja nam kaže kolika je vjerovatnoća događaja A ako nam je poznat ishod događaja C. Ovo se zove aposteriori vjerovatnoća jer znamo ishod kompletnog događaja i želimo da odredimo vjerovatnoću polaznog. To je često slučaj kojega imamo u komunikacijama. Primili smo neki podatak i želimo da odredimo da li je poslat taj ili neki drugi podatak (simbol).

Provjerimo sada tačnost sljedeće relacije:

$$P(A)=\sum_{i=1}^{N}P(C_i)P(A\,|\,C_i)$$

U našem slučaju ovo iznosi:

$$P(A=0)=P(C=0)P(A=0\,|\,C=0)+P(C=1)P(A=0\,|\,C=1)+P(C=2)P(A=0\,|\,C=2)=$$
$$=0.15\cdot 1+0.5\cdot 0.7+0.35\cdot 0=0.5$$

$$P(A=1)=P(C=0)P(A=1\,|\,C=0)+P(C=1)P(A=1\,|\,C=1)+P(C=2)P(A=1\,|\,C=2)=$$
$$=0.15\cdot 0+0.5\cdot 0.3+0.35\cdot 1=0.5$$

Uočite da su članovi $P(C=i)P(A=j\,|\,C=i)$ zapravo združene vjerovatnoće $P(A=j,C=i)$. Ujedno smo posredno dokazali i Bayesovo pravilo:

$$P(C=j\,|\,A=i)=\frac{P(A=i\,|\,C=j)P(C=j)}{\sum_{l=1}^{N}P(C=l)P(A=i\,|\,C=l)}$$

Za ovu priliku ćemo ga provjeriti samo za A=1 i C=1:

$$P(C=1\,|\,A=1)=\frac{P(A=1\,|\,C=1)P(C=1)}{P(C=0)P(A=1\,|\,C=0)+P(C=1)P(A=1\,|\,C=1)+P(C=2)P(A=1\,|\,C=2)}=$$
$$=\frac{0.3\cdot 0.5}{0.15\cdot 0+0.3\cdot 0.5+0.35\cdot 1}=\frac{0.15}{0.5}=0.3$$

Funkcija združene gustine raspodjele (združena raspodjele) slučajnih promjenljivih ξ i η se definiše kao vjerovatnoća događaja $\{\xi\le x,\eta\le y\}$ i označava se kao $P_{\xi\eta}(x;y)$ $P_{\xi\eta}(x,y)=\text{Vjer}\{\xi\le x,\eta\le y\}$. Marginalne funkcije se definišu kao $P_{\xi\eta}(x,\infty)=P_{\xi}(x)$ i $P_{\xi\eta}(\infty,y)=P_{\eta}(y)$. Funkcija gustine raspodjele se definiše kao:

$$p_{\xi\eta}(x,y)=\frac{\partial^2 P_{\xi\eta}(x,y)}{\partial x\partial y}$$

Za diskretnu dvodimenzionu slučajnu promjenljivu funkcija gustine raspodjele se definiše kao: $p_{\xi\eta}(x,y)=\text{Vjer}\{\xi=x,\eta=y\}$. Marginalne funkcije gustine raspodjele se dobijaju kao:

$$p_{\xi}(x)=\int_{-\infty}^{+\infty}p_{\xi\eta}(x,y)dy \qquad\qquad p_{\eta}(y)=\int_{-\infty}^{+\infty}p_{\xi\eta}(x,y)dx$$

1.3. Slučajni procesi

Slučajni proces $\xi(t,s)$ je funkcija koja preslikava prostor događaja u familiju vremenskih funkcija. Ovo je proširenje pojma slučajne promjenljive jer svakom od mogućih događaja (stanja ili ishoda) umjesto broja pridružuje odgovarajuća vremenska funkcija. Skup svih mogućih vremenskih funkcija se naziva **ansambl** a pojedine funkcije su realizacije ili članovi ansambla.

Funkcija raspodjele slučajnog procesa $s(t_1)=s_1$ se označava sa $P_{\xi_1}(x_1;t_1) = \text{Vjerovatnoca}\{\xi(t_1) \le x_1\}$. Slučajan proces je stacionaran ako njegova raspodjela ne zavisi od pomjeraja duž vremenske ose:

$$P_{\xi}(x,t) = P_{\xi}(x,t+\Delta t)$$

Slučajni proces je stacionaran u širem smislu ako srednja vrijednost i varijansa procesa ne zavise od vremenskog trenutka:

$$E\{\xi(t)\} = \text{const} \qquad R_{\xi\xi}(\tau) = E\{\xi(t)\xi(t+\tau)\}$$

gdje je $R_{\xi\xi}(\tau)$ autokorelaciona funkcija.

Očekivana vrijednost procesa predstavlja vremensku funkciju koja kaže koliko pojedine realizacije iz ansambla imaju srednju vrijednost u ovom trenutku. Autokorelaciona funkcija predstavlja srednju vrijednost proizvoda signala u dva različita vremenska trenutka.

Pored ovih veličina može se uvesti i srednja vrijednost slučajnog procesa unutar njegovog trajanja za datu realizaciju:

$$A\{x(t)\} = \lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{T/2} x(t)dt = \overline{x(t)}^{t}$$

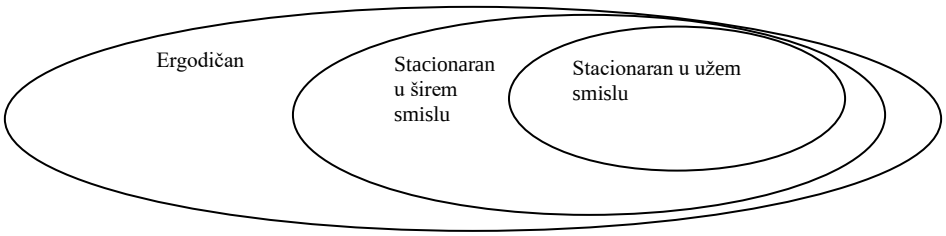
Vremenska autokorelaciona funkcija $R_{xx}(\tau)$ se definiše kao:

$$R_{xx}(\tau) = A\{x(t)x(t+\tau)\} = \overline{x(t)x(t+\tau)} = \lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{T/2} x(t)x(t+\tau)dt$$

Srednja vrijednost po vremenu za generalizovanu autokorelaciju se označava kao:

$$A\{F[x(t)x(t+\tau_1)x(t+\tau_1+\tau_2)...x(t+\tau_1+...+\tau_{n-1})]\}$$

Ako su sve srednje vrijednosti računate po ansamblu jednake svim srednjim vrijednostima računatim po vremenu proces se naziva ergodičnim. Svaki ergodični proces je stacionaran. Naime, srednja vrijednost po vremenu ne zavisi od posmatranog vremenskog trenutka. Ako je proces ergodičan to znači da ni srednja vrijednost po ansamblu (koja je jednaka srednjoj vrijednosti po vremenu) ne zavisi od vremena. Ovo je osobina koja važi za stacionarne procese. Međutim, stacionarnost ne implicira ergodičnost.



Odnosi između pojedinih familija slučajnih procesa

1.4. Spektralne karakteristike slučajnih procesa

Spektralna gustina snage slučajnog procesa $x(t)$ se definiše kao:

$$S_{xx}(\omega) = \lim_{T\to\infty}\frac{|X_T(\omega)|^2}{T}$$

gdje je:

$$X_T(\omega) = \int_{-T/2}^{T/2} x(t)e^{-j\omega t}dt$$

Snaga slučajnog signala je jednaka:

$$P_{xx} = \lim_{T\to\infty}\frac{1}{T}\frac{1}{2\pi}\left[\int_{-\infty}^{\infty}|X_T(\omega)|^2\,d\omega\right] \qquad P_{xx} = \frac{1}{2\pi}\int_{-\infty}^{\infty} S_{xx}(\omega)d\omega$$

Kako je spektralna gustina parna realna funkcija važi:

$$P_{xx} = \frac{1}{\pi}\int_{0}^{\infty} S_{xx}(\omega)d\omega$$

Vrijednosti koje smo do sada izveli su dobijene na osnovu jednog posmatranja slučajnog procesa. Da bi se odredila spektralna gustina snage čitavog procesa treba posmatrati spektralnu gustine snage slučajnog procesa odnosno srednju vrijednost po ansamblu spektralnih gustina slučajnog procesa pojedinih realizacija $S_{\xi\xi}(\omega) = E\{S_{xx}(\omega)\}$.

1.5. Wiener-Khinchine-ova teorema

Izražavanje spektralne gustine snage slučajnog procesa na osnovu prethodne formule je nepraktično pošto nam nije poznata $X_T(\omega)$ odnosno bilo bi je potrebno odrediti za svaku realizaciju. Stoga se ova veličina mora sračunati ili procjeniti na neki drugi način. Možemo zapisati:

$$S_{\xi\xi}(\omega) = E\left\{\lim_{T\to\infty}\frac{1}{T}|X_T(\omega)|^2\right\} = \lim_{T\to\infty}\frac{1}{T}E\left\{|X_T(\omega)|^2\right\} =$$

$$= \lim_{T\to\infty}\frac{1}{T}E\left\{X_T(\omega)X_T^*(\omega)\right\} = \lim_{T\to\infty}\frac{1}{T}E\left\{\int_{-T/2}^{T/2}\int_{-T/2}^{T/2}x(t_1)x^*(t_2)e^{-j\omega t_1+j\omega t_2}dt_1dt_2\right\} =$$

$$= \lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{T/2}\int_{-T/2}^{T/2}E\{x(t_1)x^*(t_2)\}e^{-j\omega t_1+j\omega t_2}dt_1dt_2$$

Sada možemo uočiti da je: $E\{x(t_1)x^*(t_2)\} = R_{\xi\xi}(t_1,t_2)$ čime se prethodna relacija svodi na:

$$S_{\xi\xi}(\omega) = \lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{T/2}\int_{-T/2}^{T/2}R_{\xi\xi}(t_1,t_2)e^{-j\omega t_1+j\omega t_2}dt_1dt_2$$

Uvodeći smjenu $t=t_1$ i $\tau=t_2-t_1$ dobijamo:

$$S_{\xi\xi}(\omega) = \lim_{T\to\infty}\frac{1}{T}\int_{-T/2-t}^{T/2-t}\int_{-T/2}^{T/2}R_{\xi\xi}(t,t+\tau)e^{-j\omega\tau}dtd\tau =$$

$$= \int_{-\infty}^{\infty}\left\{\lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{T/2}R_{\xi\xi}(t,t+\tau)dt\right\}e^{-j\omega\tau}d\tau$$

Izraz:

$$\left\{\lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{T/2}R_{\xi\xi}(t,t+\tau)dt\right\}$$

je srednja vrijednost po vremenu. Pod pretpostavkom da ova veličina ne zavisi od vremena slijedi:

$$\left\{\lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{T/2}R_{\xi\xi}(t,t+\tau)dt\right\} = R_{\xi\xi}(\tau)$$

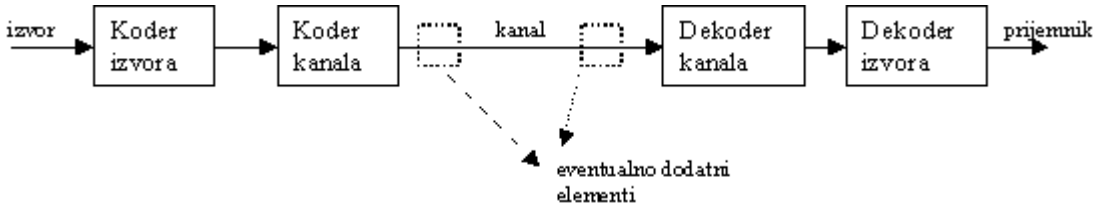
Odavde slijedi:

$$S_{\xi\xi}(\omega) = \int_{-\infty}^{\infty}R_{\xi\xi}(\tau)e^{-j\omega\tau}d\tau$$

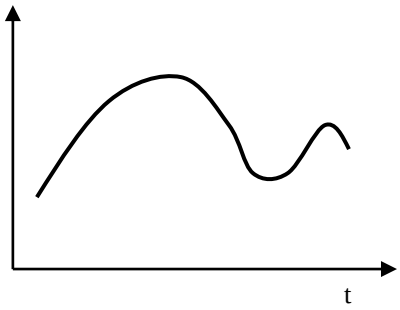
Posljednji izraz predstavlja Wiener-Khinchine-ovu teoremu koja kaže da su spektralna gustina snage slučajnog signala i autokorelaciona funkcija Fourierov transformacioni par. Ovo naravno važi samo za procese koji su stacionarni u širem smislu.

1.6. Model sistema za prenos informacija

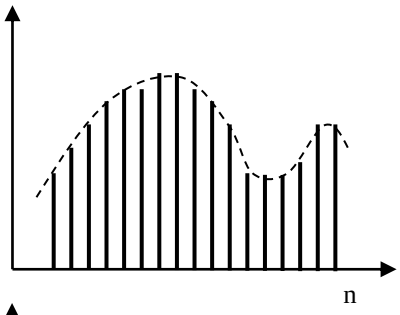
Osnovni komunikacioni model za prenos informacija se može opisati sljedećim dijagramom.



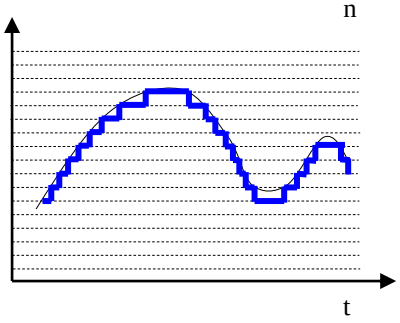
U opštem slučaju izvor poruka može da proizvodi četiri tipa poruka: Analogne; Diskretne; Kontinualne sa diskretnim vrijednostima amplitude; Digitalne. Na sljedećoj slici smo ilustrovali veze koje postoje između ovih tipova poruka.



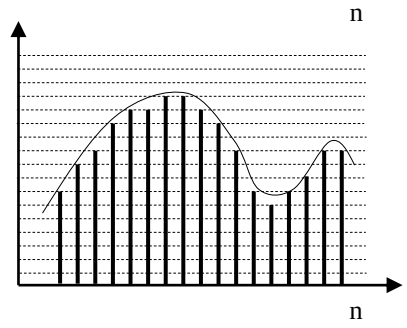
Analogna poruka (signal) postoji u svakom vremenskom trenutku datog intervala i može uzimati bilo koju vrijednost.



Diskretna poruka uzima vrijednosti samo u nekim diskretnim pozicijama (odbirci) u vremenu.



Analogna poruka (signal) postoji u svakom vremenskom trenutku datog intervala i može uzimati diskretan set vrijednosti.



Digitalna poruka odnosno poruka koja je diskretna i po vremenu i po vrijednostima. Naziva se digitalnom pošto se može jednostavno prikazati u našim računarskim sistemima

Osnovni problem prilikom prenosa informacija je da se na nekoj tački tačno ili približno tačno reproducira poruka odabrana nakoj drugoj tački (C. Shannon 1948.god).

Mi ćemo posmatrati izvor digitalnih ili diskretnih podataka. Koder izvora teži da eliminiše koliko je god moguće redudanciju između podataka. Redudancija je nepotrebno ponavljanje informacija koja je česta i u svakodnevnom životu. Npr. danas, 15.X.2003 srijeda pada kiša. Ako je ova rečenica izgovorena na dati datum onda je 15.X.2003 i srijeda redudantno sa riječju danas. Uklanjanje redundancije se vrši u cilju kompresije podataka (smanjivanja broja podataka koji je neophodno prenijeti kanalom). Koder kanala dodaje malu količinu redudancije u podatke da bi obezbjedio mogućnost detekcije i korekcije grešaka iz podataka. Kanal je medijum preko kojeg se prenose poruke. To može biti bilo koji put kojim poruke se mogu transportovati uključujući i magnetne medije za smještaj poruka, generičke kanale, radio veze, radio relejne linkove, kompjutersku mrežu, satelitsku komunikacije, itd. Dekoder kanala vrši korekciju i detekciju grešaka, dekoder izvora vrši operacije koje su neophodne da se dobija poruka u obliku prihvatljivom za korisnika.

Od kodova za kodiranje kanala mi ćemo učiti Huffmanov kod kao primjer kodiranja entropije i LZW kod kao primjer kodiranja zasnovanog na rječniku. Pored toga pomenućemo neke pomoćne kodove koji se koriste u postupku kodiranja izvora kao što su RLE kod, diferencijalni i Grayov kod. Što se tiče kodiranja kanala tu ćemo se upoznati sa kodovima koji se grubo mogu podijeliti u dvije grupe: kodove za detekciju i kodove za korekciju greške. Kodovi za detekciju greške su jednostavniji i već znamo ASCII kod a upoznaćemo još neke. Što se tiče kodova za korekciju greške upoznaćemo se sa blok kodovima. To je relativno široka grupa koja ne obuhvata sve kodove koji se koriste a obično se izučavaju zbog svoje jednostavnosti. Mi ćemo se susresti sa: pravougaonim, trougaonim, klasom Hammingovih kodova kao i sa BCH kodovima. BCH kodovi su veoma složeni i sofisticirani ali predstavljaju jedan od osnovnih alata za kodiranje kanala iz više razloga. Vrijedi napomenuti da su druge varijante kodova namjenjenih za kodiranje kanala kao što su konvolucioni i turbo kodovi još složeniji sa stanovišta dekodiranja ali da se sa njima postižu nešto bolji rezultati. Nakon kodera kanala često se dodaju drugi blokovi koji na neki način poboljšavaju performanse koda. Ovdje ćemo pomenuti samo dva tipa takvih blokova. Jedna grupa blokova odnosi se na kriptografiju. Ovi kodovi imaju namjenu da zaštite komunikaciju od neovlašćenog pristupa ili zloupotrebe. Ova oblast prevazilazi nivo našeg kursa. Drugi blok koga vrijedi pomenuti u ovom dijelu je interliver. Ovaj blok se može smatrati sastavnim dijelom uređaja za kodiranje kanala. Njegova uobičajena namjena je da smanji korelaciju između grešaka koje se mogu dogoditi u kanalu. Naime, standardni kodovi kanala teže da imaju sposobnost da isprave određeni broj nekoreliranih grešaka u kanalu. Međutim, u praksi česte su situacija kada imamo korelirane greške. Ako recimo u nekom kanalu (recimo radio kanalu) greška se pojavi usljed grmljavine velika je vjerovatnoća da će se ponovo pojaviti u nekom narednom trenutku ili u narednom intervalu. Otklanjanje ovakvih grešaka kod standardnih kodera zahtjeva znatno slabljenje performansi. Umjesto da posegnemo za otklanjanje ovih grešaka kroz spuštanje performansi kodera kanala mi dodajemo jedan blok koji se naziva interliver (na strani dekodiranja tu je deinterliver)

Model sistema za prenos informacija može da uključi i neke druge blokova. Neki sistemi sadrže blok koji predstavlja ograničenja koja su nametnuta tom kanalu, blok koji omogućuje kriptografiju i dekripciju, blok koji vrši kompresiju sa gubicima. Ove teme se mogu ubrojati u one koje pokriva teorija informacija i kodova ali na žalost to ne može da bude obuhvaćeno na našem nivou razmatranja. Od narednog poglavlja ćemo izučavati jedno od najvažnijih pitanja teorije informacija a to je kako mjeriti količinu informacije. Naredno pitanje je koliko i kako možemo vršiti kompresiju. Dalje kako možemo izbjeći greške koje se dešavaju u prenosu? Koliko brzo možemo prenositi informacije u kanalu? itd. Ovo su samo neka od pitanja u teoriji informacija i kodova kojih ćemo se držati. Naravno neka od ovih pitanja nijesu do kraja riješena i predstavljaju i dalje važan teorijski problem ali mi ćemo se zadržati na osnovnim rješenjima i dati vam neophodno predznanje da možete sami da idete dalje kroz ovu ponekad komplikovanu oblast.

Teorija informacija se može posmatrati kao fundament informacione revolucije koja je počela krajem prošlog vijeka. Ako je kraj devetnaestog vijeka obilježila industrijska revolucija zasnovna na zakonima statističke mehanike postojeća informatička revolucija je zasnovana na zakonima teorije informacija. Fundamenti ove teorije su postavljeni od strane jednog čovjeka Clauda Shannona koji je definisao sve fundamentalne zakone u ovoj teoriji počevši od 1948. godine. Neki od fundamentalnih principa i teorijskih limita koje je on postavio osnova su ove naučne discipline. Teorija je prednjačila za informatičkom revolucijom više desetina godina što nije bio slučaj sa industrijskom revolucijom. Učinak Shannona se stoga može mjeriti samo sa učinkom Ajnštajna u fizici. Drugo važno ime u ovoj oblasti je ruski matematičar Kolmogorov koji je dao značajnu matematičku podlogu. Nažalost rezultati Kolmogorova su poznati na zapadu tek početkom sedamdesetih godina prošlog vijeka. Drugi razlog zbog čega ovi rezultati nijesu direktno primjenjeni u teoriji informacija su činjenica da ih je Kolmogorov primjenjivao u teoriji skupova. Važno je zapamtiti da se mi nećemo obazirati na konkretan problem prenosa informacija a posebno nećemo razmatrati konkretne "modulacije" koje se koriste u prenosu informacije već ćemo to

sagledavati matematizirano. Ovo je predmet proučavanja drugih disciplina i na ovom nivou studentima ne bi moglo biti jasno.

Sistemi postaju sve složeniji i ne mogu biti razmatrani na intuitivan - empirijski način već moraju biti posmatrano egzaktno matematički i algoritamski.

Zadaci za vježbu

1.1. Slučajna promjenljiva ξ ima samo dvije moguće vrijednosti 0 i 1. Vjerovatnoća pojave 0 je jednaka q dok je vjerovatnoća pojave 1 jednaka p. Odrediti funkciju raspodjele ove funkcije. Odrediti srednju vrijednost i varijansu.

Rješenje: Za $x \geq 1$ događaj $\{\xi \leq x\}$ je siguran pa je: $P\{\xi \leq x\} = 1$ Za $0 \leq x < 1$ vjerovatnoća događaja $\{\xi \leq x\}$ je jednaka vjerovatnoći $\{\xi = 0\}$: $P\{\xi \leq x\} = P\{\xi = 0\} = q$. Događaj $x<0$ je nemoguć $\{\xi \leq 0\}$ pa važi: $P\{\xi \leq x\} = 0$. Ovo se ukratko može napisati kao:

$$P_{\xi}(x) = \begin{cases} 1 & x \geq 1 \\ q & 0 \leq x < 1 \\ 0 & x \leq 0 \end{cases}$$

Srednja vrijednost ovog događaja je jednaka $E\{\xi\} = 0 \cdot q + 1 \cdot p = p$ dok je varijansa jednaka: $E\{(\xi - E\{\xi\})^2\} = E\{\xi^2\} - E^2\{\xi\} = p - p^2$. Imajte na umu da je $q+p=1$.

1.2. U binarnom kanalu se prenose cifre 0 i 1. Ove dvije cifre se pojavljuje sa vjerovatnoćom ½. Zbog prisustva smetnji moguće je da ne bude primljena cifra koja je poslata. Vjerovatnoća da je primljena ispravna cifra 0 ako je poslata 0 je jednaka $P(0|0)=p$ dok je vjerovatnoća da se primi 1 kada je poslato 1 jednaka: $P(1|1)=q$. Odrediti funkciju raspodjele primljenog signala.

Rješenje: Na mjestu prijema vrijednost nula se pojavljuje u dva slučaja: poslata je nula koja je uspješno prenesena; poslata je jedinica i došlo je do greške. Ovo se može zapisati kao:

$$P\{\xi = 0\} = P(0)P(0|0) + P(1)P(0|1) = \frac{1}{2}p + \frac{1}{2}(1-q) = \frac{1}{2}(p - q + 1)$$

Vjerovatnoća događaja da je primljena jedinica je:

$$P\{\xi = 1\} = P(1)P(1|1) + P(0)P(1|0) = \frac{1}{2}q + \frac{1}{2}(1-p) = \frac{1}{2}(q - p + 1)$$

Funkcija gustine raspodjele se može zapisati kao:

$$P_{\xi}(x) = \frac{1}{2}(q - p + 1)u(x) + \frac{1}{2}(p - q + 1)u(x-1)$$

gdje je $u(x)$ jedinična diskretna Heavisajdeova odskočna funkcija.

1.3. Signal ξ koji je slučajan uniformno raspoređen na intervalu $[0,a]$ ulazi u kvantizator koji ima karakteristiku: $y=ns$ za $ns<x\leq(n+1)s$ gdje je n cijeli broj dok je s data konstanta $s=a/5$. Odrediti očekivanu vrijednost slučajne promjenljive na izlazu iz kvantizatora.

Rješenje: Slučajna promjenljiva na izlazu iz kvantizatora je diskretna sa mogućim vrijednostima: 0, s, $2s$, $3s$ i $4s$. Odgovarajuće vjerovatnoće su:

$$P\{\eta = 0\} = P\{0 < \xi \le s\} = \frac{1}{5}$$

$$P\{\eta=1\}=1/5 \quad P\{\eta=2\}=1/5 \quad P\{\eta=3\}=1/5 \quad P\{\eta=4\}=1/5$$

Dakle, očekivana vrijednost je:

$$E\{\eta\} = \sum_{i=1}^{5} P(y_i) y_i = 0\frac{1}{5} + s\frac{1}{5} + 2s\frac{1}{5} + 3s\frac{1}{5} + 4s\frac{1}{5} = 2s$$

1.4. Diskretna slučajna promjenljiva uzima vrijednosti: x_1, x_2, x_3, x_4 i x_5 sa vjerovatnoćama: 0.1, 0.2, 0.3, 0.3, 0.1. Odrediti funkciju raspodjele, srednju vrijednost i varijansu.

Rješenje. Funkcija raspodjele se može zapisati kao (pod pretpostavkom $x_i<x_{i+1}$):

$$P(\xi) = \begin{cases} 0 & \xi < x_1 \\ 0.1 & x_1 \le \xi < x_2 \\ 0.3 & x_2 \le \xi < x_3 \\ 0.6 & x_3 \le \xi < x_4 \\ 0.9 & x_4 \le \xi < x_5 \\ 1 & x_5 \le \xi \end{cases}$$

Srednja vrijednost je:

$$\mu_x = 0.1x_1 + 0.2x_2 + 0.3x_3 + 0.3x_4 + 0.1x_5$$

Varijansa je jednaka:

$$\sigma_x^2 = E\{(x-\mu_x)^2\} = E\{x^2\} - \mu_x^2 =$$
$$= 0.1x_2^2 + 0.2x_2^2 + 0.3x_3^2 + 0.3x_4^2 + 0.1x_5^2 - (0.1x_1 + 0.2x_2 + 0.3x_3 + 0.3x_4 + 0.1x_5)^2$$

1.5. Diskretna slučajna promjenljiva ξ uzima vrijednosti: 2, 4, 6, 8 i 10 sa vjerovatnoćama 0.1, 0.3, 0.4, 0.1, 0.1. Odrediti raspodjelu i očekivanu vrijednost za slučajnu promjenljivu η=max{ξ,3}. Odrediti varijansu slučajne promjenljive ρ=ξ+1.

Rješenje. Slučajna promjenljiva η ima raspodjelu datu na sledeći način vrijednosti 3, 4, 6, 8 i 10 se pojavljuju sa vjerovatnoćama 0.1, 0.3, 0.4, 0.1, 0.1. Srednja vrijednost je jednaka:

$$\mu_\eta = 0.1\cdot 3 + 0.3\cdot 4 + 0.4\cdot 6 + 0.1\cdot 8 + 0.1\cdot 10 = 5.7$$

Varijansa slučajne promjenljive ρ je:

$$E\{(\rho-\mu_\rho)^2\} = E\{\rho^2\} - \mu_\rho^2 =$$
$$= E\{\xi^2 + 2\xi + 1\} - \mu_\rho^2 =$$
$$= E\{\xi^2\} + 2E\{\xi\} + 1 - \mu_\rho^2 =$$
$$= E\{\xi^2\} + 2\mu_\xi + 1 - \mu_\rho^2$$

Srednja vrijednost slučajne promjenljive ρ je:

$$E\{\rho\} = E\{\xi + 1\} = E\{\xi\} + 1 = \mu_\xi + 1$$

pa odavde slijedi:

$$E\{(\rho - \mu_\rho)^2\} = E\{\xi^2\} + 2\mu_\xi + 1 - \mu_\xi^2 - 2\mu_x - 1 =$$
$$= E\{\xi^2\} - \mu_\xi^2 = \sigma_\xi^2$$

Dakle, slučajne promjenljive ξ i ρ imaju istu varijansu. Srednja vrijednosti promjenljive ξ je jednaka:

$$\mu_\xi = 0.1 \cdot 2 + 0.3 \cdot 4 + 0.4 \cdot 6 + 0.1 \cdot 8 + 0.1 \cdot 10 = 5.6$$

dok je srednja kvadranta vrijednost:

$$E\{\xi^2\} = 0.1 \cdot 4 + 0.3 \cdot 16 + 0.4 \cdot 36 + 0.1 \cdot 64 + 0.1 \cdot 100 = 36$$

Sada dobijamo da je varijansa slučajnih promjenljivih ξ i ρ jednaka:

$$\sigma_\xi^2 = 36 - 5.6^2 = 4.64$$

Za vježbu odrediti zavisnost srednje vrijednosti i varijanse slučajne promjenjive ρ od ovih parametara za slučajnu promjenljivu ξ ako je veza između ovih veličina linearna:

$$\rho = a\xi + b$$

1.6. Slučajna promjenljiva sa jednakim vjerovatnoćama uzima vrijednosti –1, 0, 1. Odrediti srednju vrijednost i varijansu slučajnih promjenljivih: $\eta_1 = 1 - 2\xi^2$ i $\eta_2 = a + \xi^2$.

Rješenje. Slučajne promjenljive η_1 i η_2 uzimaju sa vjerovatnoćama navedenim u tabeli:

ξ	-1	0	-1
η_1	-1	1	-1
η_2	a+1	a	a+1
p	1/3	1/3	1/3

Odnosno ovo se može sada svesti na:

η_1	-1	1
η_2	a+1	a
p	2/3	1/3

Srednja vrijednosti navedenih slučajnih promjenljivih je:

$$\mu_{\eta_1} = -\frac{2}{3} + \frac{1}{3} = -\frac{1}{3} \qquad \mu_{\eta_2} = \frac{2(a+1)}{3} + \frac{a}{3} = a + \frac{2}{3}$$

Srednje kvadratne vrijednosti ovih promjenljivih su:

$$E\{\eta_1^2\} = \frac{2}{3} + \frac{1}{3} = 1 \qquad E\{\eta_1^2\} = \frac{2(a+1)^2}{3} + \frac{a^2}{3} = a^2 + \frac{4a}{3} + \frac{2}{3}$$

Sada su varijanse jednake:

$$\sigma_{h_1}^2 = E\{\eta_1^2\} - \mu_{\eta_1}^2 = 1 - \frac{1}{9} = \frac{8}{9} \qquad \sigma_{h_2}^2 = E\{\eta_2^2\} - \mu_{\eta_2}^2 = \frac{2}{3} - \frac{4}{9} = \frac{2}{9}$$

1.7. Prenose se dvije poruke. Slučajna promjenljiva ξ uzima vrijednost 1 ako je prva poruka uspješno prenesena i 0 ako nije. Slučajna promjenljiva η se na isti način odnosi prema drugoj poruci. Ako je vjerovatnoća uspješnog prenosa prve poruke p_1 a druge poruke p_2 odrediti združenu raspodjelu slučajnih promjenljivih ξ i η.

Rješenje.

$$P\{\xi = 1, \eta = 1\} = p_1 p_2 \qquad P\{\xi = 1, \eta = 0\} = p_1(1 - p_2)$$
$$P\{\xi = 0, \eta = 1\} = (1 - p_1) p_2 \qquad P\{\xi = 0, \eta = 0\} = (1 - p_1)(1 - p_2)$$

1.8. Uslovna gustina raspodjele: $p_{\xi_1\xi_2|\xi_3\xi_4}(x_1, x_2 \mid x_3, x_4)$. Na osnovu nje treba odrediti $p_{\xi_1|\xi_3\xi_4}(x_1 \mid x_3, x_4)$. Ovdje važi pravilo koje treba dokazati:

$$p_{\xi_1|\xi_3\xi_4}(x_1 \mid x_3, x_4) = \int_{-\infty}^{\infty} p_{\xi_1\xi_2|\xi_3\xi_4}(x_1, x_2 \mid x_3, x_4) dx_2$$

Rješenje. Događaj $\xi_1\xi_2 \mid \xi_3\xi_4$ predstavlja uniju događaja da se dogodilo ξ_1 pod uslovom da se dogodilo $\xi_3\xi_4$ i događaja ξ_2 pod uslovom da se dogodilo $\xi_3\xi_4$. Stoga važi odgovarajuća marginalna:

$$p(x) = \int_{-\infty}^{\infty} p(x, y) dy$$

gdje je x u ovom slučaju događaj $\xi_1 \mid \xi_3\xi_4$ dok je događaj y zapravo $\xi_2 \mid \xi_3\xi_4$ čime smo dokazali predmetnu tvrdnju.

1.9. Zadata je uslova gustina raspodjele: $p_{\xi_1\xi_2\xi_3|\xi_4\xi_5\xi_6}$. Traži se uslovna gustina $p_{\xi_1\xi_2\xi_3|\xi_4\xi_6}$. Ovdje važi pravilo koje treba dokazati:

$$p_{\xi_1\xi_2\xi_3|\xi_4\xi_6} = \int_{-\infty}^{\infty} p_{\xi_1\xi_2\xi_3|\xi_4\xi_5\xi_6} p_{\xi_5|\xi_4\xi_6} dx_5$$

Rješenje. Predmetni izraz možemo preglednije prikazati kao:

$$p(a \mid c) = \int_{-\infty}^{\infty} p(a \mid bc) p(b \mid c) db$$

gdje je $a = \xi_1\xi_2\xi_3$, $c = \xi_4\xi_6$ i $b = \xi_5$. Ako sada uzmemo da se dogodio događaj c ovo se svodi na Bayesovo pravilo za kontinualne slučajne promjenljive.

1.10. Bacaju se dvije kocke. Događaj X je suma vrijednosti dobijena bacanjem dok je događaj Y maksimalna vrijednost dobijena bacanjem. Sračunati uslovnu vjerovatnoću $p(x|y)$.

$p(x,y)$	Y=1	Y=2	Y=3	Y=4	Y=5	Y=6	$p(x)$
X=2	1/36	0	0	0	0	0	1/36
X=3	0	2/36	0	0	0	0	2/36
X=4	0	1/36	2/36	0	0	0	3/36
X=5	0	0	2/36	2/36	0	0	4/36
X=6	0	0	1/36	2/36	2/36	0	5/36
X=7	0	0	0	2/36	2/36	2/36	6/36
X=8	0	0	0	1/36	2/36	2/36	5/36
X=9	0	0	0	0	2/36	2/36	4/36
X=10	0	0	0	0	1/36	2/36	3/36
X=11	0	0	0	0	0	2/36	2/36
X=12	0	0	0	0	0	1/36	1/36
$p(y)$	1/36	3/36	5/36	7/36	9/36	11/36	

Relativno jednostavno se može sračunati $p(x|y)=p(x,y)/p(y)$:

$p(x\|y)$	Y=1	Y=2	Y=3	Y=4	Y=5	Y=6
X=2	1	0	0	0	0	0
X=3	0	2/3	0	0	0	0
X=4	0	1/3	2/5	0	0	0
X=5	0	0	2/5	2/7	0	0
X=6	0	0	1/5	2/7	2/9	0
X=7	0	0	0	2/7	2/9	2/11
X=8	0	0	0	1/7	2/9	2/11
X=9	0	0	0	0	2/9	2/11
X=10	0	0	0	0	1/9	2/11
X=11	0	0	0	0	0	2/11
X=12	0	0	0	0	0	1/11

1.11. Sistem prenosi tri poruke A, B i C koje se mogu izabrati sa vjerovatnoćama p_A, p_B, p_C respektivno. Važi $p_A+p_B+p_C=1$. Vjerovatnoća da je primljena poruka Q ako je poslata poruka R je $P_{Q|R}$ (npr. ako je poslato A a primljeno B to je $P_{B|A}$). Odrediti vjerovatnoće primljenih simbola. Odrediti vjerovatnoće da ako je primljen neki simbol da je taj simbol i poslat.

1.12. Posmatrajte problem ARQ pomorskog saobraćaja. Prenosi se paket od 3 bloka po 7 bita. U jednom bloku uvijek se generiše 4 jedinice i 3 nule. Vjerovatnoća da je jedan bit prenesen ispravno je jednaka p. Prijemnik broji jedinice i nule u primljenom bloku. Kolika je vjerovatnoća da će koder detektovati grešku koja se dogodila u paketu. Kolika je vjerovatnoća da dekoder neće moći detektovati grešku u prenosu koja se dogodila (jedan slučaj kada se greška ne može otkriti je kada se jedna jedinica prebaci na nulu i jedna nula na jedinicu u istom bloku).

Rješenje: U slučaju da se ne dogodi pogreška sistem radi korektno. U slučaju da se greška dogodi na jednom bitu sistem primjećuje pogrešku pa i dalje radi korektno. U slučaju da se dogode dvije greške sistem radi pogrešno ako jedna jedinica postane nula i jedna nula postane jedinica jer u tom slučaju nije narušen broj jedinica i nula. Slično prethodnom problem ne postoji u bilo kojoj kombinaciji sa neparnim brojem pogreški (3, 5 i 7) dok kod parnog broja pogreški (četiri i šest) sistem ne radi korektno ako je primljen isti broj grešaka na bitima koji su jedinice i na bitima koji su nule. Vjerovatnoća pojedinačnog događaja sa dvije pogreške je:

$$p^2q^5$$

Ukupan broj ovakvih ishoda je 12 (4x3). Vjerovatnoća pojedinačnog događaja sa 4 pogreške je

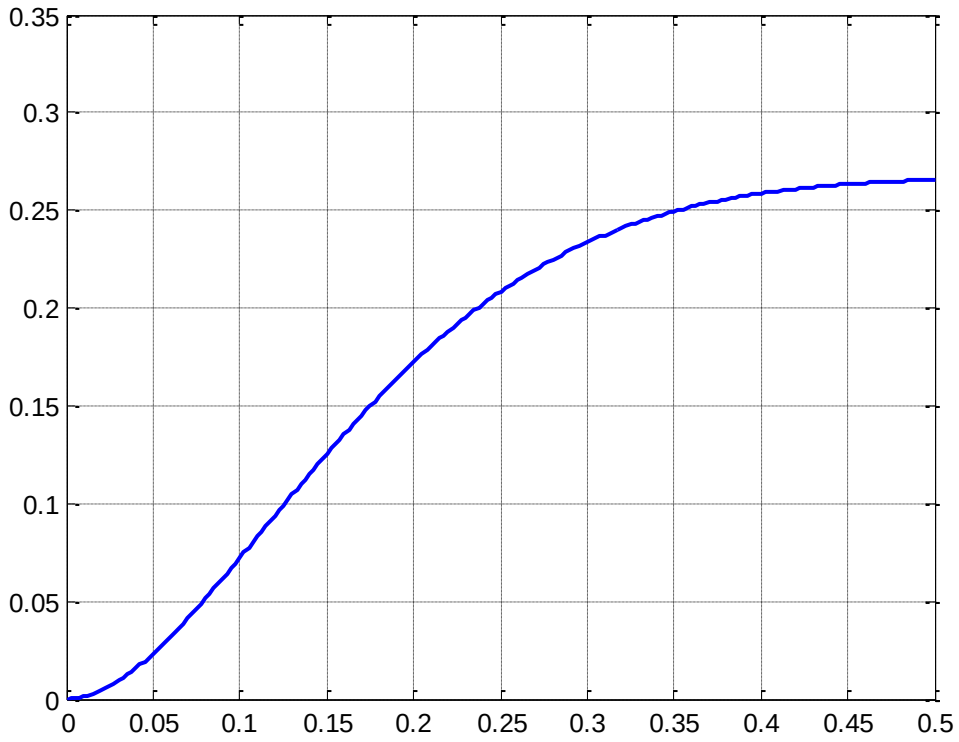
$$p^4q^3$$

Ukupan broj ishoda gdje se po dva puta griješi na jedinicama i na nulama je 18 (napominjemo da je ukupan broj ishoda sa 4 pogreške $\binom{7}{4}=35$). Konačno vjerovatnoća pojedinačnog skora sa 6 pogreški je:

$$p^6q$$

Broj situacija kada imamo 6 pogreški je 7 (greška se ne mora javiti na jednom od 7 bitova). Četiri situacije ovdje nisu povoljne odnosno situacija da se greška ne dogodi na bitu koji se pojavljuje više puta. Dakle ukupna vjerovatnoća greške (da sistem kaže poruka je OK a ona nije) je:

$$P_E=12p^2(1-p)^5+18p^4(1-p)^3+4p^6(1-p)$$



1.13. Često se u vojnim komunikacijama kao i u nekim drugim aplikacijama koristi sistem prenosa podataka većinom glasova. Isti simbol (nula ili jedan) se šalje neparan broj puta i odluka koji je simbol primljen se donosi na osnovu onog simbola koji je unutar jednog signalizacionog intervala primljen veći broj puta. Neka je vjerovatnoća greške na jednom bitu 0.1 kolika je vjerovatnoća greške ako se ovo radi na osnovu 3 i na osnovu 5 ponavljanja. Kolike su vjerovatnoće greške kada je pojedinačni bit primljen pogrešno sa vjerovatnoćom 0.3.

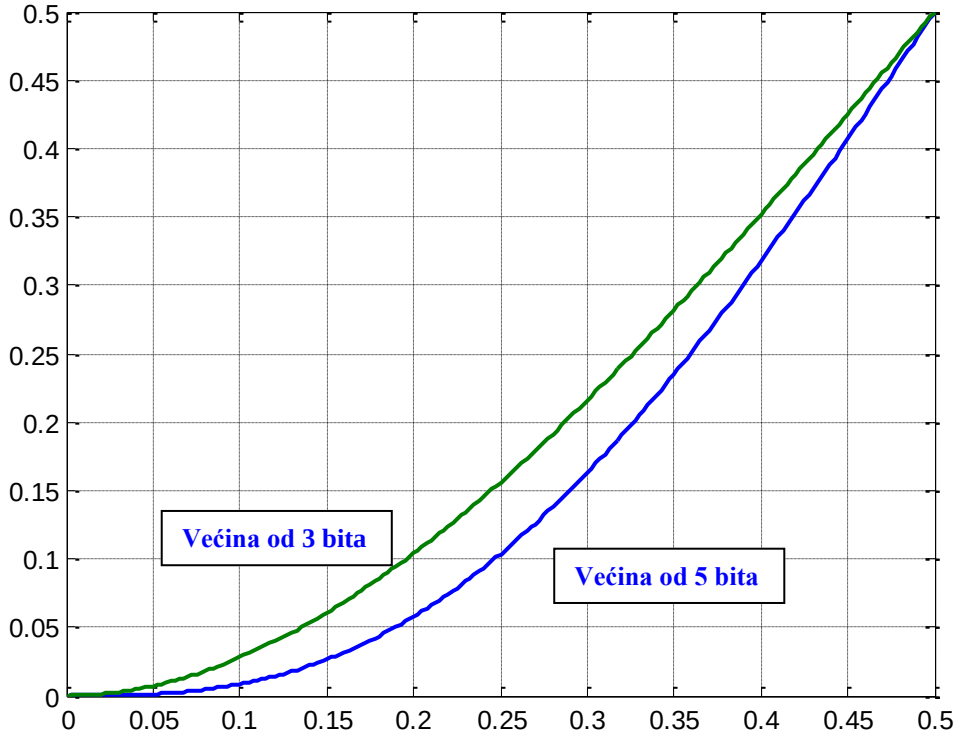
Rješenje: Ako imamo poruku od tri bita grešku ćemo detektovati kada dva bita primimo sa greškom odnosno kada primimo grešku na sva tri bita:

$$P_E=3p^2(1-p)+p^3$$

Ovdje broj 3 koji množi prvi izraz podrazumijeva da se ispravan bit mogao desiti na prvoj, drugoj ili trećoj lokaciji (ukupno tri kombinacije). Dakle vjerovatnoća greške za p=0.1 je jednaka P_E=0.028 dok je za p=0.3 P_E=0.216. U slučaju donošenja odluke većinom od 5 bita greška će se javiti ako se greška ostvari redom 3 ili više bita. Vjerovatnoća greške onda iznosi:

$$P_E=10p^3(1-p)^2+5p^4(1-p)+p^5$$

Za vjerovatnoću greške p=0.1 vjerovatnoća P_E je jednaka P_E=0.0086 dok je za p=0.3 vjerovatnoća nedetekcije pogreške P_E=0.1631.



1.14. Odredili ste srednju vrijednost i varijansu neke slučajne poruke na osnovu procjenjenih vjerovatnoća pojedinih događaja: p_i, i=1,...,N. Međutim, stvarne vrijednosti vjerovatnoća pojedinih događaja su p_i+e_i, i=1,...,N. Koliko je srednje odstupanje i srednje kvadratno odstupanje određene veličine u odnosu na tačnu veličinu. Pretpostaviti da je srednja kvadratna vrijednost $E\{e_i^2\}=\Delta^2$.

Rješenje: Srednja odstupanje vjerovatnoće je jednako:

$$E\{p_i+e_i\}=E\{p_i\}+\mathrm{E}\{e_i\}=p_i$$

pod pretpostavkom da je $E\{e_i\}$=0 što je realno jer suma vjerovatnoća mora biti jednaka 1 tako da i suma p_i+e_i mora biti jednako 1 ostaje da je suma grešaka jednaka 0 odnosno da je odgovarajuće matematičko očekivanje jednako 0. Ovo kažemo da je procjena vjerovatnoće nekog elementa iz alfabeta nepristrasna veličina. Što se tiče varijanse lako se može pokazati da važi da je:

$$\sigma^2 = E\{(p_i + e_i)^2\} - E^2\{p_i^2\} = E\{p_i^2\} - 2E\{p_i e_i\} + E\{e_i^2\} - E\{p_i^2\} =$$
$$= -2E\{p_i e_i\} + \Delta^2$$

Pod pretpostavkom da je vjerovatnoća događaja p_i i greške u procjenu vjerovatnoće toga događa nezavisni događaji onda slijedi $E\{p_i e_i\} = E\{p_i\}E\{e_i\} = p_i E\{e_i\} = p_i \cdot 0 = 0$ pa dobijamo:

$$\sigma^2 = \Delta^2$$

Posmatrajmo sada jedan jednostavan primjer koji demonstrira kako se mogu simulirati neki karakteristični slučajni događaji. Sekvenca od recimo 10 simbola iz binarnog alfabeta se u MATLAB-u može simulirati na sledeći način:

```
rand(1,10)>0.5
```

Naime ova funkcija generiše vektor od 10 simbola sa jedinicama na pozicijama gdje je rand(1,10) veće od 0.5. Kako rand uzima slučajne vrijednosti u intervalu [0,1] sa uniformnom vjerovatnoćom ovo recimo simulira slanje binarne poruke kod koje se sa istom vjerovatnoćom šalju obije vrijednosti bita. Pogledajmo sledeći primjer.

```
R=[];
for k=1:1000,R(k)=sum(rand(1,10)>0.5);end
```

Što ovdje radimo? Generišemo oko 1000 sekvenci po 10 bita i brojimo koliko jedinica imamo u tih deset bita. Očekujemo da ćemo dobiti 5 jedinica. Međutim u kratkim sekvencama ne mora to biti tako pa sa sledećim naredbama (moguća su odstupanja u zavisnosti od generisanja slučajnih brojeva) dobijamo:

```
for r=0:10,length(find(R==r)),end
1   6   27   114   213   237   223   118   46  14  1
```

Vidimo da se u ovoj kratkoj sekvenci dešava da dobijemo slučaj sa deset nula ili deset jedinica! U 237 slučajeva (od 1000) dobili smo 5 nula i 5 jedinica a u još 436 slučajeva 4 ili 6 jedinica. Ponovimo eksperiment sa sekvencom koja ima 1000 nula ili jedinica:

```
for k=1:1000,R(k)=sum(rand(1,1000)>0.5);end
```

Pogledajmo sada:

```
find(R<450)
ans =[]
find(R>550)
ans =[]
```

Dakle, sada smo dobili znatno bolju pravilnost odnosno situaciju da u hiljadu slučajnih događaja sa istom vjerovatnoćom svih hiljadu se u našoj realizaciji dogodilo takvih da je sekvenca imala između 450 i 550 jedinica. U velikom broju ponavljanja dobijamo bolju pravilnost nego u malom broju ponavljanja.

Pretpostavimo i sledeću situaciju. Kako kreirati situaciju kad se dešava 60% jedinica. Odgovor je prilično prost i takvu sekvencu možemo kreirati kao:

```
R=rand(1,10)>0.4;
```

Ovom slučaju je sa vjerovatnoćom 1-0.4=0.6 (60%) dobijamo jedinice u našoj sekvenci. Konačno pretpostavimo sledeću situaciju da je na prvom bitu jedinica ili nula sa jednakom vjerovatnoćom a zatim prethodno stanje se zadržava sa vjerovatnoćom 0.7 a prelazi se u drugo stanje sa vjerovatnoćom 0.3. Ovo se može programirati na različite načine a dolje je demonstrirano kreiranje sekvence od 10000 bita koji su dobijeni na prethodno opisani način.

```
N=10000; R=zeros(1,N);
R(1)=rand>0.5;
for k=2:N,R(k)=rem(R(k-1)+(rand>0.7),2);end
```

Ovdje koristimo zgodnu naredbu `rem` koju ćemo često upotrebljavati a koja nam daje ostatak pri dijeljenu sa dva ili ekskluzivno ili operaciju. Ako je (`rand>0.7`) jednak nuli a to će se dogoditi u 70% situacija (sa vjerovatnoćom 0.7) ostatak pri dijeljenju će biti jednak vrijednosti prvog argumenta `R(k-1)` odnosno ostajemo u stanju `R(k-1)` (prethodnom stanju), bilo da je nula ili jedan. Ako je (`rand>0.7`) jednak 1 dolazi do promjene stanja jer je `rem(0+1,2)=1` odnosno `rem(1+1,2)=0` znači bilo da je `R(k-1)` nula ili jedinica ide na suprotno stanje jedan ili nula.

Posmatrajmo još jednu jednostavnu simulaciju. Želim da dobijemo slučajni proces koji sa jednakom vjerovatnoćom daje vrijednosti iz skupa (alfabeta) {0, 1, 2}. To se može postići na sledeći način:

```
R=floor(3*rand(1,1000))
```

Proizvod `3*rand(1,1000)` daje uniformno raspoređene slučajne brojeve na intervalu [0,3). Naredba floor ih zaokružuje naniže tako da dobijamo samo 0, 1 i 2 i to sa jednakim vjerovatnoćama.

Poglavlje II
MJERA KOLIČINE INFORMACIJA

2.1. Entropija

Već na ovom času ćemo se upoznati sa jednim najvažnijih pojmova u teoriji informacija. To je pojam entropije. Strogo matematički, entropija je mjera neodređenosti slučajne promjenljive. Mi ćemo je definisati na sledeći način.

Definicija: Posmatrajmo skup X čiji elementi mogu uzimati diskretne vrijednosti koje pripadaju skupu A (ponekad se kaže alfabetu A) i neka je poznata funkcija vjerovatnoće pojavljivanja pojedinog elementa iz skupa X: $p(x)=p_X(x)=\Pr\{X=x\}$, $x\in A$. Tada je entropija skupa X (ili neodređenost skupa X) $H(x)$ data kao:

$$H(X)=-\sum_{x\in A} p(x)\log p(x)$$

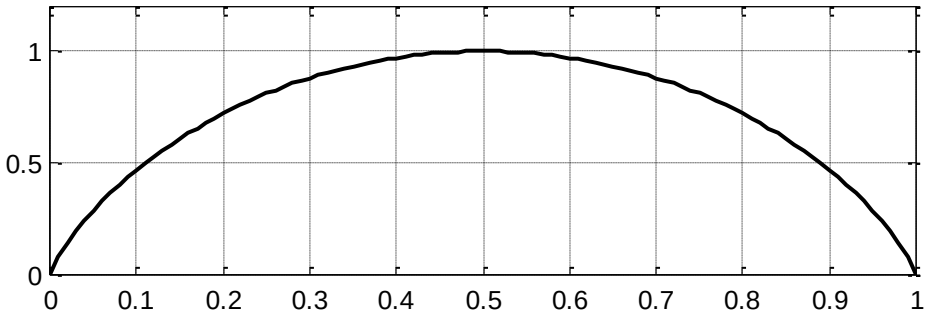
Po konvenciji se usvaja da je 0log0=0. Uvijek je osnova logaritma veća od 1, a po pravilu se uzima dva i tada se entropija mjeri u bitovima.[1] Entropija je nenegativna veličina.

Kako entropija mjeri neodređenost sistema. Nije teško naći za ovo primjer. Pretpostavimo događaj da kiša pada u Sahari. To se dešava sa jako malom vjerovatnoćom, recimo 0.01 (0.99 je da kiša ne pada). Tada je ovaj događaj sasvim malo neodređen i njegova entropija je $H(X)=0.0808$. Vjerovatnoća događaja da kiša pada u Britaniji je recimo 0.5. To daje maksimalnu neodređenost i entropija ovog sistema $H(X)=1$. Dakle, što je entropija veća, to je veća i neodređenost u sistemu, odnosno, veća je količina informacije koja se dobije na osnovu opisa vremenskog stanja. Podsjetimo se da mi u principu ne znamo kolika je vjerovatnoća nekog događaja, ali da na osnovu snimanja velikog broja slučajeva vršimo procjenu.

Ako imamo binarni slučajni proces, koji uzma jednu vrijednost sa vjerovatnoćom p, a drugu sa vjerovatnoćom 1-p, njegova entropija je jednaka:

$$H(X)=-p\log p-(1-p)\log(1-p)$$

Zbog česte primjene entropija ovakvog događaja se označava sa $H(p)$ i naziva se često binarnom entropijom.



Sa slike je jasno (a može se lako i dokazati) da je entropija najveća za p=0.5 (jednakovjerovatne događaje) kada iznosi $H(p)$=1bit.

Posmatrajmo ponovo primjer kiše u Sahari. Ako je vjerovatnoća događaja p=0.01 ovaj događaj nosi veliku informaciju kada se dogodi -log2(0.01)=6.64 međutim njegov doprinos je entropiji dosta mali pošto je ovaj događaj sa veoma malom vjerovatnoćom. Slična situacija je kod glasova u našem

[1] Ako je osnova algoritma 10 entropija se mjeri u dit- ovima, dok u slučaju primjene prirodnog logaritma e entropija se mjeri u nat-ovima. Dugo vremena je postojala dilema oko usvajanja osnove algoritma u SI sistemu mjera ali se i tamo koristi bit odnosno osnova logaritma 2 sa time da se šesto ova veličina naziva Shannon i označava Sh.

jeziku. Samoglasnici se prilično rijetko javljaju sa približno 0.45 totalnom vjerovatnoćom (približno 0.09 je vjerovatnoća svakog od njih u prosjeku, dosta više za A od ostalih) dok suglasnici imaju ukupnu vjerovatnoću 0.55 (nešto ispod 0.02 u prosjeku što je znatno manje od samoglasnika, i ovdje se znatno češće pojavljuju neka slova od drugih). Dakle, doprinos svakog suglasnika entropiji je manji od doprinosa samoglasnika ali kada se pojavi suglasnik on mnogo više nosi informacija od samoglasnika. Posmatrajmo sledeća dva imena gdje su u na jednu stranu izdvojeni suglasnici a na drugu samoglasnici. Pogledajte iz koje grupe je lakše doći do pogotka o kojoj je imenu riječ (moje je mišljenje da se iz samoglasnika može naslutiti ime ili riječ dok je to nemoguće uraditi iz suglasnika):

MRK	AO
MLJN	IIAA

Definicija: Ako je $X \sim p(x)$ i ako je $g(X)$ funkcija slučajne promjenljive X, tada je matematičko očekivanje, slučajne promjenljive $g(X)$: $E_p[g(X)] = E[g(X)] = \sum_{x \in A} g(x)p(x)$. Kao posljedica ovoga entropija se može zapisati kao:

$$H(X) = -E[\log p(X)]$$

Definicija: Združena entropija događaja (X,Y) koji se mogu opisati združenom funkcijom vjerovatnoće $p(x,y)$ (ovo se ponekad obilježava $(X,Y) \sim p(x,y)$) je jednaka:

$$H(X,Y) = -\sum_{x \in A}\sum_{y \in B} p(x,y)\log p(x,y) = -E[\log p(X,Y)]$$

Definicija: Uslovna entropija $H(Y|X)$ se definiše kao:

$$H(Y \mid X) = -\sum_{x \in A}\sum_{y \in B} p(x,y)\log p(y \mid x) = -E[\log p(Y \mid X)]$$

Definicija: Uslovna entropija $H(Y|X=x)$ se definiše kao:

$$H(Y \mid X = x) = -\sum_{y \in B} p(y \mid x)\log p(y \mid x) = -E[\log p(Y \mid X) \mid X = x]$$

Uslovna i združena entropija su nenegativne veličine. **Dokazati!** Maksimalna vrijednost uslovne entropije se postiže kada je $H(Y \mid X) = H(Y)$. **Dokazati!**

Dokaz 1. Prvi dokaz je izuzetno jednostavan. Ponovo imamo slučaj da je logaritam veličine koja je manja ili jednaka 1 negativan broj odnosno nula kada je argument jednak 1. Ovo se množi sa pozitivnom veličinom koja pa imamo zbir nepozitivnih veličina pomnožene sa -1.

Dokaz 2. Pođimo od:

$$H(Y \mid X) = -\sum_{x \in A}\sum_{y \in B} p(x,y)\log p(y \mid x)$$

Posmatrajmo sada razliku $H(Y|X)$-$H(Y)$:

$$H(Y \mid X) - H(Y) = -\sum_{x \in A}\sum_{y \in B} p(x,y)\log p(y \mid x) + \sum_{y \in B} p(y)\log p(y)$$

Koristivši činjenicu da je:

$$\sum_{x \in A} p(x,y) = p(y)$$

drugi izraz možemo svesti na dvostruku sumu:

$$H(Y \mid X) - H(Y) = -\sum_{x\in A}\sum_{y\in B} p(x,y)\log p(y \mid x) + \sum_{x\in A}\sum_{y\in B} p(x,y)\log p(y) =$$

$$= \sum_{x\in A}\sum_{y\in B} p(x,y)\log\frac{p(y)}{p(y \mid x)}$$

Od ove tačke na dalje dokaz može da ide u raznim smjerovima ali je najjednostavnije koristiti sledeću osobinu logaritamske funkcije $\log a \le (a-1)\log e$. Provjera ove nejednakosti je relativno jednostavna jer se može pokazati da funkcija $f(a)=\log a-(a-1)\log e$ dostiže maksimum za $a=1$ koji iznosi 0. Odavde slijedi:

$$H(Y \mid X) - H(Y) \le \sum_{x\in A}\sum_{y\in B} p(x,y)\left[\frac{p(y)}{p(y \mid x)} - 1\right]\log e = \sum_{x\in A}\sum_{y\in B} p(x,y)\frac{p(y)}{p(y \mid x)}\log e - \sum_{x\in A}\sum_{y\in B} p(x,y)\log e$$

$$= \sum_{x\in A}\sum_{y\in B} p(x)p(y \mid x)\frac{p(y)}{p(y \mid x)}\log e - \log e = \sum_{x\in A}\sum_{y\in B} p(x)p(y)\log e - \log e = 0$$

Ovim je dokazano da važi $H(Y|X)\le H(Y)$. Dakle, uslovna entropija je manja od marginalne! Ovo je logično ali prije nego to malo detaljnije diskutujemo razmotrimo situaciju kada se dostiže jednakost. Jednakost se očigledno dostiže kada važi $\log a=(a-1)\log e$ a to dalje znači kada je $a=1$ odnosno kada je: $p(y)/p(y|x)=1$ odnosno kada je $p(y)=p(y|x)$ a ovo važi kada y je nezavisno do x odnosno ako y ne zavisi od x važi: $H(Y|X)=H(Y)$. Dakle, ako x (odnosno događaji iz skupa X) ukazuju nešto o y (odnosno o događajima iz skupa Y) dolazi do umanjivanja entropije (neznanja – neodređenosti) o događaju y. Jedini način da nam događaji iz skupa X ne umanje entropiju o događajima iz skupa Y je da između njih nema veze odnosno da su događaji iz skupa Y nezavisni od događaja iz skupa X.

2.2. Relativna entropija i međusobna informacija

Definicija: Relativna entropija (ili Kullback-Leiblerova distanca ili diskriminaciona funkcija) između dvije funkcije $p(x)$ i $q(x)$ se definiše kao (obično se primjenjuje na vjerovatnoće iz diskretnog skupa ili funkcije gustina raspodjele):

$$D(p \,\|\, q) = \sum_{x\in A} p(x)\log\frac{p(x)}{q(x)} = E_{p(x)}\left[\frac{p(X)}{q(X)}\right]$$

Ovdje važe konvencije da je $0\log(0/q)=0$ i da $p\log(p/0)=\infty$.

Definicija: Međusobna informacija $I(X;Y)$ između dvije diskretne slučajne promjenljive X i Y sa združenom gustinom raspodjele $p(x,y)$ i marginalnim raspodjelama $p(x)$ i $p(y)$ se definiše kao:

$$I(X;Y) = \sum_{x\in A}\sum_{y\in B} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} = D(p(x,y) \,\|\, p(x)p(y)) = E_{p(x,y)}\left[\log\frac{p(X,Y)}{p(X)p(Y)}\right]$$

Međusobna informacija se može prikazati kao razlika entropija. Međusobna informacija predstavlja dio informacije koju dijele (koja je zajednička) događaji $\boldsymbol{X}$ i $\boldsymbol{Y}$.

Lemma: Međusobna informacija zadovoljava:

$$I(X;Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X) = H(X) + H(Y) - H(X,Y)$$

Dokaz:

$$I(X;Y)=\sum_{x\in A}\sum_{y\in B}p(x,y)\log\frac{p(x,y)}{p(x)p(y)}=\sum_{x\in A}\sum_{y\in B}p(x,y)\log\frac{p(x)p(y\mid x)}{p(x)p(y)}=$$
$$=\sum_{x\in A}\sum_{y\in B}p(x,y)\log\frac{p(y\mid x)}{p(y)}=-\sum_{x\in A}\sum_{y\in B}p(x,y)\log p(y)+\sum_{x\in A}\sum_{y\in B}p(x,y)\log p(y\mid x)$$

Kako važi:

$$\sum_{x\in A}p(x,y)=p(y)$$

Dobija se:

$$I(X;Y)=-\sum_{y\in B}p(y)\log p(y)+\sum_{x\in A}\sum_{y\in B}p(x,y)\log p(y\mid x)=H(Y)-H(Y\mid X)$$

Sada možemo definisati i sledeće pravilo (u engleskoj literaturi poznato kao chain rule – pravilo lanca) za entropije:

$$H(X,Y)=H(X)+H(Y\mid X)=H(Y)+H(X\mid Y)$$

Ovo se može intepretirati kao: informacija o sadržaju *X* i *Y* je jednaka informaciji o *X* plus informaciji o *Y* za dato *X*.

Međusobna informacija je jednaka zbiru entropija (samoinformacija) o dvijema slučajnim promjenljivim, minus združena informacija o ova dva događaja.

Provjera svih ovih relacija može se obaviti na jednom od prethodnih zadataka.

$$H(Y)=-\frac{1}{36}\log_2\frac{1}{36}-\frac{3}{36}\log_2\frac{3}{36}-\frac{5}{36}\log_2\frac{5}{36}-\frac{9}{36}\log_2\frac{9}{36}-\frac{11}{36}\log_2\frac{11}{36}\approx 2.32$$

$$H(X)=-2\cdot\frac{1}{36}\log_2\frac{1}{36}-2\cdot\frac{2}{36}\log_2\frac{2}{36}-2\cdot\frac{3}{36}\log_2\frac{3}{36}-$$
$$-2\cdot\frac{4}{36}\log_2\frac{4}{36}-2\cdot\frac{5}{36}\log_2\frac{5}{36}-\frac{6}{36}\log_2\frac{6}{36}\approx 3.27$$

Ako pogledamo tabelu ispod uočavamo da se u tabeli združenih vjerovatnoća 6 puta pojavljuje vjerovatnoća 1/36 i 15 puta vjerovatnoća 2/36 pa možemo da zapišemo:

$$H(X,Y)=-6\cdot\frac{1}{36}\log_2\frac{1}{36}-15\cdot\frac{2}{36}\log_2\frac{2}{36}\approx 4.33$$

Sada po definiciji sračunajmo $H(X|Y)$. Da bi se to uradilo moraju se kombinovati rezultati iz tabele združenih vjerovatnoća i tabele uslovnih vjerovatnoća:

$$H(X\mid Y)=-\frac{1}{36}\log_2 1-\frac{2}{36}\log_2\frac{2}{3}-\frac{1}{36}\log_2\frac{1}{3}-\frac{2}{36}\log_2\frac{2}{5}-\frac{2}{36}\log_2\frac{2}{5}-\frac{1}{36}\log_2\frac{1}{5}-$$
$$-\frac{2}{36}\log_2\frac{2}{7}-\frac{2}{36}\log_2\frac{2}{7}-\frac{2}{36}\log_2\frac{2}{7}-\frac{1}{36}\log_2\frac{1}{7}-\frac{2}{36}\log_2\frac{2}{9}-\frac{2}{36}\log_2\frac{2}{9}-$$
$$-\frac{2}{36}\log_2\frac{2}{9}-\frac{1}{36}\log_2\frac{1}{9}-\frac{2}{36}\log_2\frac{2}{11}-\frac{2}{36}\log_2\frac{2}{11}-\frac{2}{36}\log_2\frac{2}{11}-\frac{2}{36}\log_2\frac{2}{11}-\frac{1}{36}\log_2\frac{1}{11}$$

$$H(X \mid Y) = 0 - \frac{2}{36}\log_2\frac{2}{3} - \frac{1}{36}\log_2\frac{1}{3} - \frac{4}{36}\log_2\frac{2}{5} - \frac{1}{36}\log_2\frac{1}{5} -$$
$$- \frac{6}{36}\log_2\frac{2}{7} - \frac{1}{36}\log_2\frac{1}{7} - \frac{8}{36}\log_2\frac{2}{9}$$
$$- \frac{1}{36}\log_2\frac{1}{9} - \frac{10}{36}\log_2\frac{2}{11} - \frac{1}{36}\log_2\frac{1}{11} \approx 2.01$$

Sada vidimo da je neodređenost (entropija) događaja Y 2.32 bita. Združena neodređenost X i Y je jednaka 4.33 bita. Ona se može dobiti kada znamo događaj Y i kada na neodređenost o njemu dodamo neodređenost o X kada znamo Y. Dakle, eksperimentalno smo potvrdili da je $H(X,Y)=H(Y)+H(X|Y)$. Iz ovih veličina praktično možemo sračunati sve ostale koje su nam potrebne recimo $H(Y|X)$ je jednaka $H(Y|X)=H(X,Y)-H(X)=1.06$ bita. Posmatrajmo sada međusobnu informaciju:

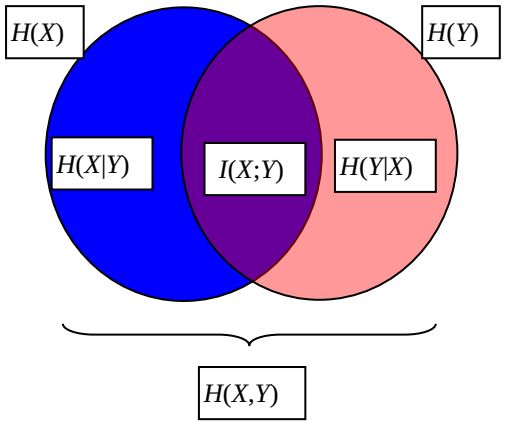
$$I(X;Y)=H(X)+H(Y)-H(X,Y) =3.27+2.32-4.33=1.26$$

Pokušajmo da na jedan plastičan način objasnimo što je to međusobna informacija. Pođimo od ove definicije $I(X;Y)=H(X)-H(X|Y)$. Vidimo da uslovna entropija ovdje umanjuje entropiju događaja pa međusobnu informaciju možemo shvatiti kao onaj dio entropije koji preostaje u nekom događaju kada znamo ishod nekog drugog događaja i znamo zavisnost posmatranog događaja od tog drugog uslovnog događaja.

$p(x,y)$	Y=1	Y=2	Y=3	Y=4	Y=5	Y=6	$p(x)$
X=2	1/36	0	0	0	0	0	1/36
X=3	0	2/36	0	0	0	0	2/36
X=4	0	1/36	2/36	0	0	0	3/36
X=5	0	0	2/36	2/36	0	0	4/36
X=6	0	0	1/36	2/36	2/36	0	5/36
X=7	0	0	0	2/36	2/36	2/36	6/36
X=8	0	0	0	1/36	2/36	2/36	5/36
X=9	0	0	0	0	2/36	2/36	4/36
X=10	0	0	0	0	1/36	2/36	3/36
X=11	0	0	0	0	0	2/36	2/36
X=12	0	0	0	0	0	1/36	1/36
$p(y)$	1/36	3/36	5/36	7/36	9/36	11/36	

$p(x\|y)$	Y=1	Y=2	Y=3	Y=4	Y=5	Y=6
X=2	1	0	0	0	0	0
X=3	0	2/3	0	0	0	0
X=4	0	1/3	2/5	0	0	0
X=5	0	0	2/5	2/7	0	0
X=6	0	0	1/5	2/7	2/9	0
X=7	0	0	0	2/7	2/9	2/11
X=8	0	0	0	1/7	2/9	2/11
X=9	0	0	0	0	2/9	2/11
X=10	0	0	0	0	1/9	2/11
X=11	0	0	0	0	0	2/11
X=12	0	0	0	0	0	1/11

Ovdje možemo da napravimo interesantnu ilustraciju koje dobro osvjetljava odnose entropija i međusobne informacije. Posmatrajmo sledeći dijagram.



Dijagram predstavlja ilustraciju veza između entropija i međusobne informacije. Lako je uočiti zašto u združenoj entropiji moramo od zbira $H(X)$ i $H(Y)$ da oduzmemo međusobnu informaciju. Nadalje, ovaj dijagam se može koristiti kao memotehnika za pamćenje osnovnih činjenica kao što je npr. $H(X|Y) \leq H(X)$. Ujedno vidimo i što se dešava kada su događaji nezavisni. Kružnice se ne presjecaju $H(X|Y)=H(X)$ i $H(X,Y)= H(X)+H(Y)$ i $I(X;Y)=0$.

2.3. Važne teoreme vezane za međusobnu informaciju

Uvedimo na početku nekoliko matematičkih preliminarnih stavova, koji uključuju Jensenovu nejednakost, da bi kasnije iz ovoga izveli neke od najvažnijih stavova teorije informacija.

Definicija: Funkcija $f(x)$ je konveksna na intervalu (a,b), ako za svako x_1, $x_2 \in (a,b)$ i $0 \leq p \leq 1$ važi $f(px_1 + (1-p)x_2) \leq pf(x_1) + (1-p)f(x_2)$. Funkcija je striktno konveksna, ako jednakost važi samo za $p=0$ i $p=1$.

Na sličan način se može definisati i konkavna funkcija. Međutim, mi ćemo se zadovoljiti sa definicijom da je funkcija konkavna ako je $-f(x)$ konveksna.

Teorema: Ako $f(x)$ ima drugi izvod $f''(x)=d^2f(x)/dx^2$ koji je nenegativan tada je $f(x)$ konveksna. Striktna konveksnost važi ako je $f''(x)>0$.

Teorema (Jansenova nejednakost): Ako je $f(x)$ konveksna funkcija i X slučajna promjenljiva tada

$$E[f(X)] \geq f(E[X])$$

Posljedica: Ako je funkcija striktno konveksna tada $X=E[X]$ je zadovljeno sa vjerovatnoćom 1 odnosno X je konstantno.

Dokaz. Jensenova nejednakost nema jednostavan dokaz. Lijeva strana nejednakosti je jednaka (za diskretne slučajne promjenljive):

$$E[f(X)] = \sum_{i=1}^{k} p_i f(x_i)$$

gdje slučajna promjenljiva uzima k diskretnih vrijednosti x_i, i=1,...,k sa vjerovatnoćama p_i, i=1,...,k. Dalje, možemo pisati:

$$E[f(X)] = \sum_{i=1}^{k} p_i f(x_i) = \sum_{i=1}^{k-1} p_i f(x_i) + p_k f(x_k) = (1-p_k)\sum_{i=1}^{k-1} \frac{p_i}{1-p_k} f(x_i) + p_k f(x_k) =$$

$$= p_k f(x_k) + (1-p_k)\sum_{i=1}^{N-1} p_i' f(x_i)$$

gdje je $p_i' = p_i/(1-p_k)$. Druga suma se može zapisati korišćenjem osobine konveksnosti važi:

$$\sum_{i=1}^{N-1} p_i' f(x_i) \geq f\left(\sum_{i=1}^{N-1} p_i' x_i\right)$$

Dalje se dobija da je:

$$E[f(X)] \geq p_k f(x_k) + (1-p_k) f\left(\sum_{i=1}^{k-1} p_i' x_i\right) \geq f\left(p_k f(x_k) + (1-p_k)\sum_{i=1}^{k-1} p_i' x_i\right) = f\left(\sum_{i=1}^{k} p_i x_i\right) = f\left(E[X]\right)$$

čime je teorema dokazana.

Teorema (Informaciona nejednakost). Sledeća nejednakost važi:

$$D(p\,||\,q) = \sum_{x\in A} p(x)\log\frac{p(x)}{q(x)} \geq 0$$

dok jednakost važi ako i samo ako $p(x)=q(x)$ za svako x.

Dokaz. Dokaz ove važne teoreme vam prepuštam.

Posljedica: Međusobna informacija je nenegativna veličina: $I(X;Y)\geq 0$.

Dokaz. $I(X;Y)=D(p(x,y)||p(x)p(y))\geq 0$.

Posljedica: Za bilo koje dvije uslovne slučajne promjenljive važi: $D(p(y|x)||q(y|x))\geq 0$ dok jednakost važi ako i samo ako $p(y|x)=q(y|x)$ za svako (x,y) i $p(x)>0$.

Teorema. Neka je $X\sim p(x)$ diskretna slučajna promjenljiva definisana na alfabetu veličine $||A||$. Tada važi $H(X)\leq \log||A||$, a jednakost je zadovoljena ako i samo ako $p(x)=1/||A||$, to jest, ako je X uniformno distribuirana na alfabetu A.

Dokaz. Neka je $u(x)=1/||A||$ i neka je $p(x)\sim X$. Tada:

$$D(p\,||\,u) = \sum_{x\in A} p(x)\log\frac{p(x)}{u(x)} = -\sum_{x\in A} p(x)\log u(x) + \sum_{x\in A} p(x)\log p(x) = \log||A|| - H(X) \geq 0$$

Jednakost se dokazuje na sličan način.

Dokaz: Prethodni dokaz je veoma jednostavan ali nažalost počiva na netrivijalnom dokazu nenegativnosti relativne entropije. Srećom može se dokazati na nešto duži način koji ne uključuje dokazivanje nenegativnosti relativne entropije. Posmatrajmo entropiju:

$$H(X) = -\sum_{i=1}^{N} p_i \log p_i = H(p_1, p_2, ..., p_N)$$

Pošto postoji ograničenje da je suma vjerovatnoća jednaka 1:

$$\sum_{i=1}^{N} p_i = 1$$

Entropija se ne mijenja kada na njenu vrijednosti dodamo tzv. Lagranžev množioc: $\lambda\left(\sum_{i=1}^{N} p_i - 1\right)$ odnosno:

$$H(p_1, p_2, ..., p_N) = -\sum_{i=1}^{N} p_i \log p_i + \lambda\left(\sum_{i=1}^{N} p_i - 1\right)$$

Računajući parcijalni izvod $H(p_1, p_2, ..., p_N)$ po bilo kojoj promjenljivoj p_j (uzećemo radi jednostavnosti izraza da je logaritam evaluiran sa osnovom e odnosno da je u pitanju prirodan logaritam):

$$\frac{\partial H(p_1, p_2, ..., p_N)}{\partial p_j} = -\log p_j - 1 + \lambda = 0$$

Dobijamo da je

$$p_j = e^{\lambda - 1}$$

Kako je svako p_j isto i jednako $p_j = e^{\lambda-1}$ zaključujemo da je $p_1=p_2=...=p_N=1/N$ i da je maksimalna entropija onda:

$$H(X) = -\sum_{i=1}^{N} \frac{1}{N} \log(1/N) = \log N$$

Istini za volju vrijedi napomenuti da bi ovdje trebalo pokazati da je u pitanju maksimum a ne prevojna tačka ili minimum međutim taj dio dokaza vam prepuštamo a ujedno je i mnogo jednostavniji nego ovo do sada.

Teorema. O uslovnoj redukciji entropije. $H(X|Y) \leq H(X)$ sa jednakošću koja važi ako su X i Y, nezavisne promjenljive.

Dokaz. $I(X;Y)=H(X)-H(X|Y) \geq 0$ a jednakost važi ako i samo ako je $p(x,y)=p(x)p(y)$.

Teorema. Neka je $(X_1,X_2,...,X_m)$ sa $p(x_1,x_2,...,x_m)$. Tada važi:

$$H(X_1, X_2, ..., X_m) \leq \sum_{i=1}^{m} H(X_i)$$

sa jednakošću, ako su X_i statistički nezavisni.

Dokaz. Koristivši "chain rule" i činjenicu da uslovni dogadjaj ne može da uveća entropiju slijedi:

$$H(X_1, X_2, ..., X_m) \leq \sum_{i=1}^{m} H(X_i \mid X_1, X_2, ..., X_{i-1}) \leq \sum_{i=1}^{m} H(X_i)$$

Jasno je da svaka uslovna informacija zapravo doprinosi smanjenju entropije kompletnog događaja. Jednakost se postiže jedino ako je svaki događaj iz niza nezavisan od svih ostalih tj. ako informacije o njemu nisu sadržane ni u jednom drugom događaju.

2.4. Uslovna međusobna informacija

Definicija: Uslovna međusobna informacija slučajnih promjenljivih X i Y za dato Z je:

$$I(X;Y \mid Z) = H(X \mid Z) - H(X \mid Y,Z) = E\left[\log \frac{p(X,Y \mid Z)}{p(X \mid Z)p(Y \mid Z)}\right]$$

Teorema: Uslovna međusobna informacija je nenegativna veličina: $I(X;Y|Z) \geq 0$ gdje jednakost važi ako i samo ako su X i Y uslovno nezavisni od Z.

Dokaz. $I(X;Y|Z)=D(p(x,y|z)||p(x|z)p(y|z)) \geq 0$.

Teorema: Sledeći izraz za međusobnu informaciju važi ("chain rule" za međusobnu informaciju):

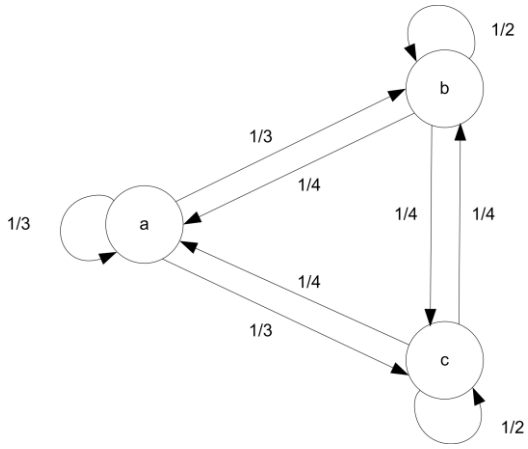
$$I(X_1,X_2,...,X_n|Y)=H(X_1,...,X_n)-H(X_1,...,X_n|Y)= \sum_{i=1}^{n} H(X_i \mid X_1,....,X_{i-1}) - \sum_{i=1}^{n} H(X_i \mid X_1,....,X_{i-1},Y) =$$

$$= \sum_{i=1}^{n} I(X_i;Y \mid X_1,...,X_{i-1})$$

2.5. Markovljeve slučajne promjenljive

Kod Markovljevih procesa podrazumjeva se (što je i prirodno) da vjerovatnoća pojavljivanja nekog simbola u velikoj mjeri zavisi od prethodnih (ili budućih stanja). Markovljev proces j-tog reda se opisuje uslovnom vjerovatnoćom: $p(s_{j+1} \mid s_1, s_2, ..., s_j)$ koja predstavlja vjerovatnoću pojavljivanja simbola s_j, pod uslovom da su se prethodno pojavili simboli $s_1, s_2, ..., s_j$ u datom redosljedu. Terminološki: proces bez memorije je onaj kod kojeg je pojavljivanje simbola nezavisno od onoga što je prethodilo tom simbolu. Markovljev proces j-tog reda predstavlja opis procesa sa memorijom j. Kod Markovljevog procesa posmatraju se sve kombinacije kako se može doći do određenog simbola. Za alfabet sa q kodnih riječi Markovljev proces j-tog reda ima ukupno q^j ovakvih stanja. Najčešće se posmatra Markovljev proces 1 reda. Markovljev sistem nultog reda se naziva sistemom bez memorije. Markovljevi procesi se najčešće prikazuju grafom tranzicije. Na slici je dat graf tranzicije za ternarni sistem gdje su vjerovatnoće da se dođe do nekog stanja iz datog stanja date pored strelica za ovaj sistem koji je Markovljev prvog reda. Markovljev proces se naziva **ergodičnim**, ako se iz bilo kog stanja može preći u bilo koje stanje. Ako se iz nekog stanja ne može preći u neko drugo stanje proces je **neergodičan**.

Stanje S_i kod Markovljevih sistema je **tranzijentno**, ako iz bar jednog stanja u koja se može iz tog stanja (direktno ili indirektno) više ne može vratiti u stanje S_i. Stanje je **rekurentno (esencijalno)**, ako se iz svakog stanja u koja se može doći iz tog stanja može vratiti u dato stanje. Stanje je **periodično**, ako postoji cio broj d (d>1) takav da se u isto stanje može vratiti samo poslije kd koraka (k pozitivan cio broj). Ako je d=1 u pitanju je **aperiodično** stanje. Stanje je **apsorbujuće**, ako se ne može napustiti.



Vjerovatnoće pojedinih stanja kod Markovljevog procesa se definišu preko vektora. Pretpostavimo da imamo N mogućih stanja. Tada se vektorom $\mathbf{p}^0 = [p_1, p_2, ..., p_N]^T$ mogu opisati vjerovatnoće pojavljivanja pojedinih simbola u početnom trenutku. Vjerovatnoće prelaza iz stanja i u stanje j

neka su date kao P_{ij}. Pretpostavimo da se one ne mijenjaju tokom vremena (to je jedan od uslova da imamo ergodičan ili stacionaran proces). Ova matrica zvana matricom tranzicije se može napisati kao:

$$\mathbf{P}=\begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1N} \\ p_{21} & p_{22} & \cdots & p_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ p_{N1} & p_{N2} & \cdots & p_{NN} \end{bmatrix}$$

Suma svih redova u transformacionoj matrici mora biti jednaka 1 (zašto?). Vjerovatnoća pojavljivanja pojedinog simbola poslije M trenutaka se može zapisati kao:

$$\mathbf{p}^M=\mathbf{P}\,\mathbf{p}^{M-1}=\mathbf{P}^M\,\mathbf{p}^0$$

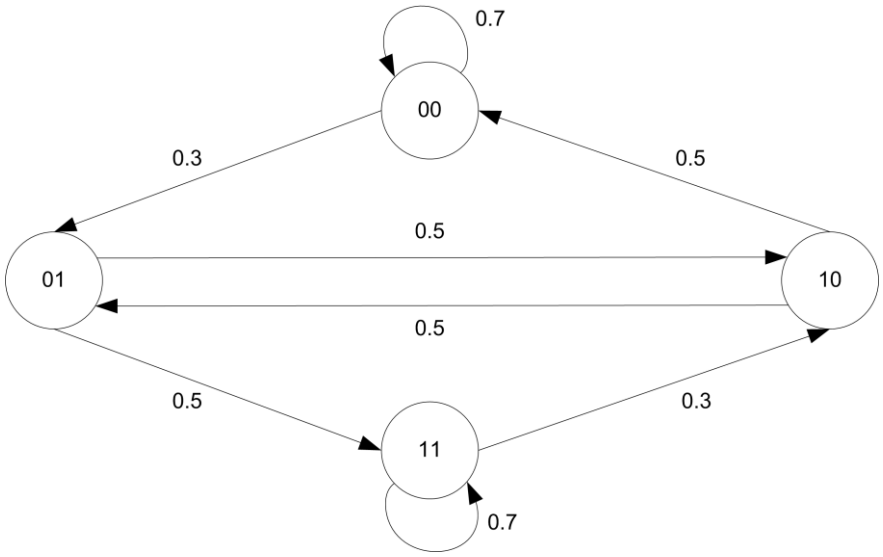
Stacionarno stanje vjerovatnoća se može odrediti na osnovu sljedeće logike. Ako je stacionarno stanje postignuto u novom trenutku (nakon nove transformacije) neće biti promjenjeno. Odnosno u stacionarnom režimu važi:

$$\mathbf{p}=\mathbf{P}\,\mathbf{p}$$

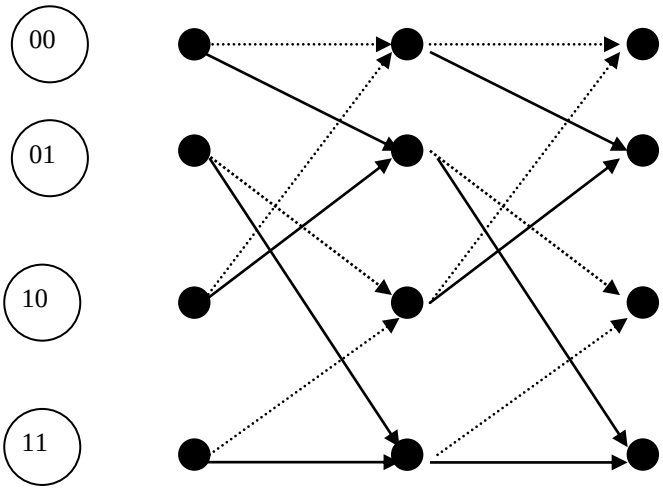
Ovo je sistem sa N nepoznatih i N-1 nezavisnom jednačinom (u najboljem slučaju) ali imamo dodatni uslov da je $p_1+p_2+...+p_N=1$ na osnovu kojeg možemo odrediti vjerovatnoće pojavljivanja pojedinih simbola u stacionarnom stanju.

Ako imamo tranzicionu matricu **P**, ona je regularna ako postoji cijeli broj k_0 takav da matrica $\mathbf{P}^{k_0}$ nema nijedan nulti element (tada i stepeni $\mathbf{P}^k$ neće imati nultih elemenata). Ergodičnost procesa se može opisati i preko matrice **P**. Proces je ergodičan, ako se iz svakog stanja u svako stanje može preći u konačnom broju koraka. Proces je ergodičan ako postoji k takvo da $\mathbf{P}^k$ ima barem jednu nenultu kolonu. Svaki regularan proces je ujedno i ergodičan, dok obrnuto ne mora da važi.

Dijagram stanja (graf tranzicije) se može koristiti i u slučaju da imamo Markovljeve izvore višeg reda. Npr. ako je u pitanju binarni izvor (q=2) drugog reda (m=2) sa uslovim vjerovatnoćama $P(0|00)=P(1|11)=0.7$, $P(1|00)=P(0|11)=0.3$, $P(0|01)=P(0|10)=P(1|01)=P(1|10)=0.5$ dijagram se može prikazati kao na slici.



Umjesto dijagrama stanja koji su pogodni za generalne opservacije, često se koristi dinamički dijagram stanja (tzv. **trelis**) koji je pogodan za praćenje konkretnog stanja.



Isprekidanim linijama su označena stanja koja prouzrokuje nula dok su punim označena ona stanja koje prouzrokuje jedinica.

Definicija: Slučajne promjenljive formiraju Markovljev lanac označen sa $X\rightarrow Y\rightarrow Z$ ako:

$$p(z|x,y)=p(z|y)$$

to jest Z je uslovno nezavisna od X za dato Y ili preciznije rečeno, ako imamo informaciju vezanu za Y, informacija o X nam ne daje ništa novo o slučajnoj promjenljivoj Z. Dalje, $X\rightarrow Y\rightarrow Z$ znači: $p(x,y|,z)=p(x)p(y|x)p(z|y)$.

Posljedica: $X\rightarrow Y\rightarrow Z$ ako i samo ako su X i Z uslovno nezavisni za dato Y. Ovo slijedi iz:

$$p(x,z\mid y)=\frac{p(x,y,z)}{p(y)}=\frac{p(x,y)p(z\mid x,y)}{p(z)}=\frac{p(x,y)p(z\mid y)}{p(z)}=p(x\mid y)p(z\mid y)$$

Očigledno $X\rightarrow Y\rightarrow Z$ implicira $Z\rightarrow Y\rightarrow X$.

Teorema (Nejednakost procesiranja informacija - Data processing inequality): Ako $X\rightarrow Y\rightarrow Z$ tada: $I(X;Y)\geq I(X;Z)$.

Drugim riječima, ova teorema kaže da ako je Z nastalo procesiranjem podataka nad Y, $Z=F(Y)$, bilo da je $F()$ deterministička ili slučajna funkcija nezavisna od X ne dolazi do uvećanja znanja koje nam Y može saopštiti o X.

Dokaz: Međusobna informacija se može zapisati kao:

$$I(X;Y,Z)=I(X;Z)+I(X;Y|Z)=I(X;Y)+I(X;Z|Y)$$

Kako su X i Z nezavisni za dato Y tada slijedi da je: $I(X;Z|Y)=0$. Kako je $I(X;Y|Z)\geq 0$ tada važi:

$$I(X;Y)\geq I(X;Z)$$

Ova nejednakost ima daleko veći značaj koji prevazilazi teoriju informacija. Na primjer u telekomunikacijama mi vršimo filtriranje korisnog signala zato što znamo da se on nalazi na nekoj frekvenciji ili kanalu uklanjajući šum koji se ne nalazi na predmetnoj frekvenciji. Međutim, ako ne znamo gdje se korisni signal nalazi niti imamo način da odredimo njegovu poziciju najbolje je ne raditi ništa odnosno ne vršiti nikakvu operaciju bez obezbijeđenog predznanja. U mnogim oblastima inžinjerskih nauka se često pravi greška procesiranjem signala o kome ne postoji predznanje (ili postoji u nedovoljnom obimu) čime se informacije pohranjene u signalu mogu samo izgubiti a nikako uvećati.

2.6. Fanoova nejednakost

Fundamentalni problem u nauci, a posebno u komunikacijama je procjena (estimacija) neke vrijednosti na osnovu odgovarajućeg mjerenja. Pretpostavimo da je vrijednost X poslata u komunikacioni kanal, gdje je promjenjena i da je primljena vrijednost Y. Naša je želja da na osnovu primljene vrijednosti procjenimo vrijednost X. Procjenjene vrijednosti ćemo označavati kao $\hat{X}$. Dakle, procjena se može opisati sljedećom funkcijom: $\hat{X}=g(Y)$.

Naravno, postoji potreba da se odredi koja je vjerovatnoća da je naša procjena jednaka pravoj vrijednosti. Jedan način da se ovo opiše je poznata Fanoova nejedenakost, koja povezuje vjerovatnoću greške sa uslovnom entropijom $H(X|Y)$. Intuitivno govoreći, ako je mala neodređenost o vrijednosti X kada imamo Y, onda je vjerovatnoća greške mala i obrnuto. Entropija $H(X|Y)=0$ kaže da bi vjerovatnoća greške u tom slučaju trebala biti 0, odnosno, nakon observacije Y mi nemamo neodređenosti oko vrijednosti X. Fanoova nejednakost vrši kvantifikovanje ovih tvrdnji. Uvedimo prije teoreme sljedeću oznaku:

$$P_e = \text{Vjerov.Gr.}=\Pr\{\hat{X} \neq X\}$$

Teorema. (Fanoova nejednakost):

$$H(P_e)+P_e\log(\|A\|-1)\geq H(X|Y).$$

Teorema se može redukovati na: $1+P_e\log(\|A\|)\geq H(X|Y)$ gdje je $\|A\|$ kardinalnost posmatranog alfabeta.

Napomena. Za $P_e=0$ ova se teorema svodi na $H(X|Y)=0$.

Dokaz: Definišimo slučajnu promjenljivu E kao:

$$E = \begin{cases} 1 & X \neq \hat{X} \\ 0 & X = \hat{X} \end{cases}$$

Sada se $H(E,X|Y)$ može razviti na dva načina korišćenjem lančanog pravila:

$$H(E,X|Y)=H(X|Y)+H(E|X,Y)=H(E|Y)+H(X|E,Y)$$

Dalje, $H(E|X,Y)=0$, jer ako je poznato i X i Y mi ne pravimo grešku. Takođe, na osnovu pravila o redukovanju entropije slijedi: $H(E|Y)\leq H(E)=H(P_e)$.

$$H(X|E,Y)=\Pr(E=0)H(X|Y,E=0)+\Pr(E=1)H(X|Y,E=1)\leq(1-P_e)0+P_e\log(\|A\|-1)$$

Odavde slijedi:

$$H(X|Y)+H(E|X,Y)=H(E|Y)+H(X|E,Y)\leq H(P_e)+P_e\log(\|A\|-1)$$

čime je dokaz obavljen. Fanoova nejednakost se može tumačiti na sledeći način. Prilikom prijema na izlazu potrebno je $H(P_e)$, da bi se provjerilo da li je došlo do greške kao i još $\log(\|A\|-1)$ da bi se u slučaju da je do greške došlo provjerilo koji je simbol zapravo primljen.

Zadaci za vježbu

2.1. Uspostaviti vezu između $H(Y|X)$ i $H(Y|X=x)$:

Rješenje:

$$H(Y \mid X) = -E[E[\log p(Y \mid X) \mid X = x]] = -\sum_{x\in A} p(x)E[\log p(Y \mid X) \mid X = x] = -\sum_{x\in A} p(x)H(Y \mid X = x)$$

2.2. Na osnovu Jensenove nejednakosti dokazati da za sve nenegativne brojeve $(a_1,...,a_n)$ i $(b_1,...,b_n)$ važi:

$$\sum_{i=1}^{n}\left(a_i \log\frac{a_i}{b_i}\right) \geq \sum_{i=1}^{n} a_i \log\left(\sum_{i=1}^{n} a_i \Big/ \sum_{i=1}^{n} b_i\right)$$

dok jednakost važi za $a_i=b_i$. Po konvenciji se usvaja da je $0\log0=0$, $a\log(a/0)=\infty$ i $0\log(0/0)=0$.

Rješenje: Funkcija $f(x)=x\log x$ je striktno konveksna za svako $0\leq x<\infty$ jer je:

$$f'(x) = \log x + x\frac{1}{x}\log e = \log x + \log e \qquad f''(x) = \frac{1}{x}\log e > 0$$

$$\sum_{i=1}^{n} p_i x_i \log x_i = E[X \log X] \geq E[X]\log(E[X]) = \left(\sum_{i=1}^{n} p_i x_i\right)\log\left(\sum_{i=1}^{n} p_i x_i\right)$$

gdje je p_i, neka funkcija koja ispunjava uslove funkcije gustine raspodjele. Uzimajući da je:

$$p_i = b_i / \sum_{i=1}^{n} b_i \quad x_i = a_i / \sum_{i=1}^{n} a_i$$

može se pisati:

$$\sum_{i=1}^{n} \frac{b_i}{\sum_{i=1}^{n} b_i}\frac{a_i}{b_i}\log\left(\frac{a_i}{b_i}\right) \geq \left(\sum_{i=1}^{n} \frac{b_i}{\sum_{i=1}^{n} b_i}\frac{a_i}{b_i}\right)\log\left(\sum_{i=1}^{n} \frac{b_i}{\sum_{i=1}^{n} b_i}\frac{a_i}{b_i}\right)$$

$$\sum_{i=1}^{n}\left(\frac{a_i}{\sum_{i=1}^{n} b_i}\log\frac{a_i}{b_i}\right) \geq \left(\sum_{i=1}^{n} \frac{a_i}{\sum_{i=1}^{n} b_i}\right)\log\left(\sum_{i=1}^{n} \frac{a_i}{\sum_{i=1}^{n} b_i}\right)$$

$$\frac{\sum_{i=1}^{n} a_i \log\frac{a_i}{b_i}}{\sum_{i=1}^{n} b_i} \geq \left(\frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i}\right)\log\left(\frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i}\right)$$

Na kraju treba pomnožiti obije strane nejednačine sa $\sum_{i=1}^{n} b_i$ čime je dokaz završen.

2.3. Dokazati lemu 2.2.3.

2.4. Dokazati teoremu 2.3.3.

Rješenje: Na osnovu rješenja zadatka 2.2 možemo pisati (ovo je samo jedan od mogućih dokaza):

$$D(p \,\|\, q) = \sum_x p(x)\log\frac{p(x)}{q(x)} \geq \sum_x p(x)\log\frac{\sum_x p(x)}{\sum_x q(x)} = 1\log 1 = 0$$

2.5. Izvršiti alternativni dokaz teoreme 2.3.4 koristivši nejednakost $\log x \leq (x-1)\log e$.

Rješenje: Za realno x može se pisati:

$$H(X) - \log\|A\| = -\sum_{x\in A} p(x)\log p(x) - \log\|A\| = \sum_{x\in A} p(x)\log\frac{1}{\|A\|\,p(x)} \leq$$

$$\sum_{x\in A} p(x)\left(\frac{1}{\|A\|\,p(x)} - 1\right)\log e = \sum_{x\in A}\left(\frac{1}{\|A\|} - p(x)\right)\log e = 0$$

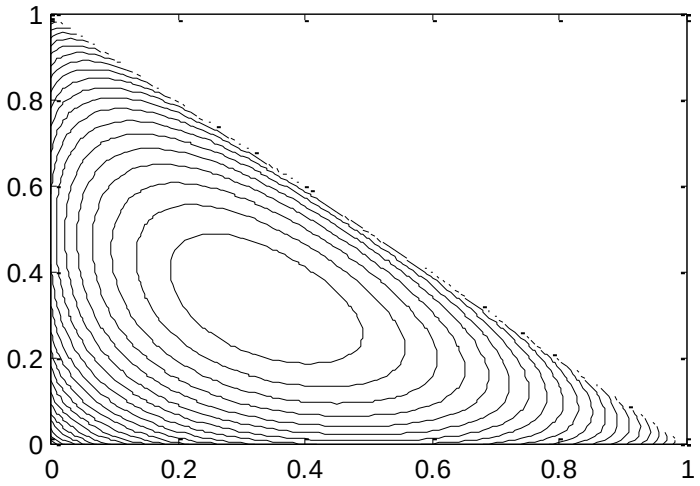
2.6. Skicirati entropiju ternarnog događaja. Uzeti da su vjerovatnoće pojedinih događaja p, q, i $1-p-q$ pa entropiju crtati u zavisnosti od p i q. Kada je ova entropija maksimalna i koliko iznosi.

Rješenje: Entropija iznosi:

$$H(p,q) = -p\log p - q\log q - (1-p-q)\log(1-p-q)$$

Entropija se može sračunati samo pod ograničenjem da je $p+q \leq 1$. Ilustracija entropije je data na slici u konturnom crtežu sa izolinijama. I u ovom slučaju se maksimalna entropija dobija kada je $p=q=1-p-q=1/3$. Dokazuje se određivanjem izvoda funkcije $H(p,q)$ po p i po q i njihovim

izjednačavanjem sa nulom. Ova veličina iznosi $H(1/3,1/3)=1.585$bita. Pogledajte zadatak 7 za domaći.



2.7. Entropija skupa X, je $H(X)=8$bita a entropija skupa Y je $H(Y)=12$bita. U kojim se granicama nalazi $H(Y|X)$, ako se $H(X|Y)$ mijenja od minimalne do maksimalne vrijednosti.

Rješenje: Međusobna informacije se može zapisati kao:

$$I(X;Y)=H(X)-H(X\mid Y) \qquad I(X;Y)=H(Y)-H(Y\mid X)$$

Lako je pokazati da važi:

$$H(Y\mid X)=H(Y)-H(X)+H(X\mid Y)$$

Dalje važi:

$$H(Y\mid X)_{\max}=H(Y)-H(X)+H(X\mid Y)_{\max} \qquad H(Y\mid X)_{\min}=H(Y)-H(X)+H(X\mid Y)_{\min}$$

Maksimalno $H(X|Y)$ je jednako $H(X)$ pa je $H(Y\mid X)_{\max}=H(Y)=12$bita dok je minimalno $H(X|Y)$ jednako 0 pa se dobija: $H(Y\mid X)_{\max}=H(Y)-H(X)=4$bita.

2.8. Posmatrajte binarni simetrični kanal, gdje je vjerovatnoća pojavljivanja 1 jednaka p i vjerovatnoća greške pri prenosu simbola kroz kanal jednaka P. Kolika je entropija sistema na ulazu a kolika na izlazu? Kolika je međusobna informacija stanja na ulazi i izlazu i kolike su odgovarajuće uslovne entropije? Kakve zaključke iz ovoga možete da izvedete?

Rješenje: Na ulazu u sistem imamo binarni događaj kod kojega se jedinica i nula pojavljuju respektivno sa vjerovatnoćama p i $1-p$. Dakle, entropija na ulazu je $H(X)=H(p)=-p\log_2 p-(1-p)\log_2(1-p)$. Na izlazu jedinica se pojavljuje sa vjerovatnoćom $P_1=p(1-P)+(1-p)P$ dok se nula pojavljuje sa vjerovatnoćom $P_0=(1-p)P+pP=1-P_1$. Grubo govoreći jedinica se pojavljuje na izlazu kada je jedinica sa ulaza prenesena bez greške i kada je nula sa ulaza prenesena sa greškom. Na osnovu navedenog entropija na izlazu je:

$$H(Y)=H(P_1)=-P_1\log_2 P_1-P_0\log_2 P_0=-P_1\log_2 P_1-(1-P_1)\log_2(1-P_1)$$

Uslovna entropija ima formalnu definiciju:

$$H(Y\mid X)=-\sum_{x\in A}\sum_{y\in B}p(x,y)\log p(y\mid x)$$

Zapišimo sada vjerovatnoće mogućih događaja:

$p(x,y)$ X→Y↓	0	1
0	$(1-p)(1-P)$	pP
1	$(1-p)P$	$p(1-P)$

$p(y\|x)$ X→Y↓	0	1
0	$(1-P)$	P
1	P	$(1-P)$

Odavde slijedi da je uslovna entropija:

$$H(Y \mid X) = -(1-p)(1-P)\log_2(1-P) - pP\log_2 P - (1-p)P\log_2 P - p(1-P)\log_2(1-P) =$$
$$= -[(1-p)+p](1-P)\log_2(1-P) - [p+(1-p)]P\log_2 P =$$
$$= -(1-P)\log_2(1-P) - P\log_2 P = H(P)$$

$$I(X;Y)=H(Y)\text{-}H(Y|X)=H(P_1)\text{-}H(P)=H(p(1\text{-}P)+(1\text{-}p)P)\text{-}H(P)$$

Združena entropija sistema je jednaka:

$$H(X,Y) = -\sum_{x\in A}\sum_{y\in B} p(x,y)\log p(x,y) = -(1-p)(1-P)\log(1-p)(1-P) -$$
$$- pP\log pP - (1-p)P\log(1-p)P - p(1-P)\log p(1-P) =$$
$$= -(1-p)(1-P)\log(1-p) - (1-p)(1-P)\log(1-P) - pP\log p -$$
$$- pP\log P - (1-p)P\log(1-p) - (1-p)P\log P - p(1-P)\log p -$$
$$- p(1-P)\log(1-P) =$$
$$= -(1-p)\log(1-p) - (1-P)\log(1-P) - p\log p - P\log P =$$
$$= H(p) + H(P)$$

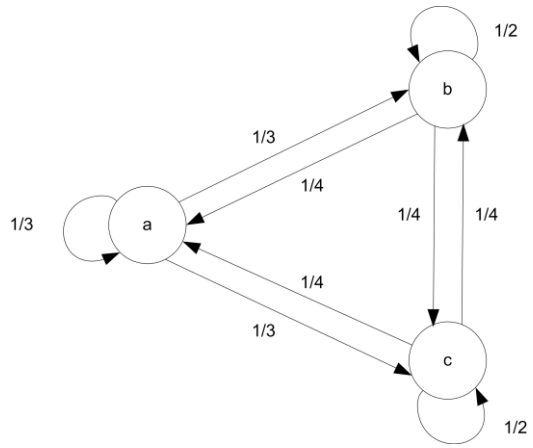
Do ovoga se moglo lakše doći na osnovu proste relacije da je:

$$H(X,Y)=H(X)+H(Y|X)$$

Slično možemo odrediti entropiju $H(X|Y)$ kao:

$$H(X|Y)=H(X,Y)\text{-}H(Y)=H(p)+H(P)\text{-}H(P_1)=$$
$$=H(p)+H(P)\text{-}H(p(1\text{-}P)+(1\text{-}p)P)$$

2.9. Pogledajte graf tranzicije kojega smo već koristili u našim primjerima.



Ako su vjerovatnoće početnih stanja $p(a)=1/2$, $p(b)=1/4$ i $p(c)=1/4$ odrediti vjerovatnoće u stacionarnom stanju.

Rješenje: Zapišimo vektor polaznih vjerovatnoća kao:

$$\mathbf{p}^0 = \begin{bmatrix} p(a) \\ p(b) \\ p(c) \end{bmatrix}$$

Vektor vjerovatnoća za naredni trenutak iznosi:

$$\begin{bmatrix} p^{(1)}(a) \\ p^{(1)}(b) \\ p^{(1)}(c) \end{bmatrix} = \begin{bmatrix} p(a \mid a) & p(a \mid b) & p(a \mid c) \\ p(b \mid a) & p(b \mid b) & p(b \mid c) \\ p(c \mid a) & p(c \mid b) & p(c \mid c) \end{bmatrix} \begin{bmatrix} p(a) \\ p(b) \\ p(c) \end{bmatrix}$$

Dok poslije proizvoljnog broja koraka vjerovatnoća iznosi:

$$\mathbf{p}^{(i)} = P\mathbf{p}^{(i-)}$$

U stacionarnom stanju će važiti:

$$\mathbf{p}^{(i)} = \mathbf{p}^{(i-)}$$

a to dalje znači da tražimo takav set vjerovatnoća $\mathbf{p}^{(i)} = [p'(a) \;\; p'(b) \;\; p'(c)]^T$ za koji važi:

$$\begin{bmatrix} p'(a) \\ p'(b) \\ p'(c) \end{bmatrix} = \begin{bmatrix} p(a \mid a) & p(a \mid b) & p(a \mid c) \\ p(b \mid a) & p(b \mid b) & p(b \mid c) \\ p(c \mid a) & p(c \mid b) & p(c \mid c) \end{bmatrix} \begin{bmatrix} p'(a) \\ p'(b) \\ p'(c) \end{bmatrix}$$

Lijevu stranu izraza uvijek možemo napisati kao proizvod sa jediničnom matricom:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p'(a) \\ p'(b) \\ p'(c) \end{bmatrix} = \begin{bmatrix} p(a \mid a) & p(a \mid b) & p(a \mid c) \\ p(b \mid a) & p(b \mid b) & p(b \mid c) \\ p(c \mid a) & p(c \mid b) & p(c \mid c) \end{bmatrix} \begin{bmatrix} p'(a) \\ p'(b) \\ p'(c) \end{bmatrix}$$

pa svođenjem na istu stranu izraza dobijamo:

$$\begin{bmatrix} p(a\mid a)-1 & p(a\mid b) & p(a\mid c) \\ p(b\mid a) & p(b\mid b)-1 & p(b\mid c) \\ p(c\mid a) & p(c\mid b) & p(c\mid c)-1 \end{bmatrix}\begin{bmatrix} p'(a) \\ p'(b) \\ p'(c) \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Lako se može pokazati da ovaj homogeni sistem od 3 jednačine ne može da da jedinstveno rješenje pa moramo da uvedemo neku dodatnu jednačinu to je u ovom slučaju poznato ograničenje da je suma vjerovatnoća jednaka 1: $p'(a)+p'(b)+p'(c)=1$

$$\begin{bmatrix} p(a\mid a)-1 & p(a\mid b) & p(a\mid c) \\ p(b\mid a) & p(b\mid b)-1 & p(b\mid c) \\ 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} p'(a) \\ p'(b) \\ p'(c) \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

U ovom slučaju smo posljednju jednačinu zamijenili sa ograničenjem $p'(a)+p'(b)+p'(c)=1$. Uvrstimo sada vjerovatnoće:

$$\begin{bmatrix} -2/3 & 1/4 & 1/4 \\ 1/3 & -1/2 & 1/4 \\ 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} p'(a) \\ p'(b) \\ p'(c) \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Najjednostavije je ovaj sistem riješiti preko računanja inverzne matrice ili putem MATLAB naredbe:

$$\begin{bmatrix} p'(a) \\ p'(b) \\ p'(c) \end{bmatrix}=\begin{bmatrix} -2/3 & 1/4 & 1/4 \\ 1/3 & -1/2 & 1/4 \\ 1 & 1 & 1 \end{bmatrix}^{-1}\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

```
p=inv([-2/3 ¼ ¼; 1/3 -1/2 ¼;1 1 1])*[0 0 1]'
```

Ako ne znamo da odredimo inverznu matricu ili nemamo pri ruci MATLAB moramo se poslužiti standardnom tehnikom eleminacije promjenljivih:

$$\begin{aligned} -\frac{2}{3}p'(a)+\frac{1}{4}p'(b)+\frac{1}{4}p'(c)&=0 \\ \frac{1}{3}p'(a)-\frac{1}{2}p'(b)+\frac{1}{4}p'(c)&=0 \\ p'(a)+p'(b)+p'(c)&=1 \end{aligned}$$

Iz prve jednačine odredimo $p'(a)$ izrazimo ga preko $p'(b)$ i $p'(c)$ kao:

$$p'(a)=\frac{3}{8}p'(b)+\frac{3}{8}p'(c)$$

Zatim ovo uvrstimo u preostale dvije jednačine:

$$-\frac{3}{8}p'(b)+\frac{3}{8}p'(c)=0$$
$$\frac{11}{8}p'(b)+\frac{11}{8}p'(c)=1$$

Iz prve jednačine se sada vidi da je $p'(b)=p'(c)$ (ovo smo mogli i na početku da zaključimo jer za ova dva simbola je sve simetrično) pa se lako dobija da je:

$$p'(b)=p'(c)=\frac{4}{11}$$

Dalje, slijedi da je:

$$p'(a)=\frac{3}{11}$$

Uočimo da ovdje polazne vjerovatnoće nisu bitne i da se zadatak mogao riješiti i bez poznavanja polaznih vjerovatnoća. Odnosno bilo koji polazni set vjerovatnoća u ovom slučaju, da kažemo, "dobro povezanog grafa" (provjerite koji su uslovi potrebni da bi ovo važilo) daje iste vjerovatnoće simbola u stacionarnom stanju nakon dovoljnog broja koraka.

2.10. Posmatrajte dijagram stanja Markovljevog procesa drugog reda koji je dat prethodno. Da li je moguće odrediti stacionarne vjerovatnoće, kao i stacionarne uslovne vjerovatnoće prvog reda za ovaj sistem. Ako je moguće odredite ih, a ako nije obrazložite zbog čega to nije moguće. Koliko stanja ima ukupno ovaj dijagram?

Rješenje: Ovdje se radi o zasigurno složenijem problemu nego što je slučaj bio u prethodnom zadatku. Uzmimo da je vjerovatnoća nule u početnom trenutku $p'(0)$ i $p'(1)$. Pretpostavimo da znamo uslovne vjerovatnoće u početnom trenutku $p'(0|0)$, $p'(1|0)$, $p'(0|1)$, $p'(1|1)$. Združene vjerovatnoće pojedinih parova početnih simbola su:

$$p'(00)=p'(0)p'(0|0) \qquad p'(10)=p'(0)p'(1|0) \qquad p'(01)=p'(1)p'(0|1) \qquad p'(11)=p'(1)p'(1|1)$$

Sada možemo da odredimo kolike su združene vjerovatnoće u narednom trenutku:

$$\begin{bmatrix} p''(00) \\ p''(01) \\ p''(10) \\ p''(11) \end{bmatrix} = \begin{bmatrix} 0.7 & 0 & 0.5 & 0 \\ 0.3 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0 & 0.3 \\ 0 & 0.5 & 0 & 0.7 \end{bmatrix} \begin{bmatrix} p'(00) \\ p'(01) \\ p'(10) \\ p'(11) \end{bmatrix}$$

Za stacionarno stanje važi jednačina:

$$\begin{bmatrix} p(00) \\ p(01) \\ p(10) \\ p(11) \end{bmatrix} = \begin{bmatrix} 0.7 & 0 & 0.5 & 0 \\ 0.3 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0 & 0.3 \\ 0 & 0.5 & 0 & 0.7 \end{bmatrix} \begin{bmatrix} p(00) \\ p(01) \\ p(10) \\ p(11) \end{bmatrix}$$

Odnosno:

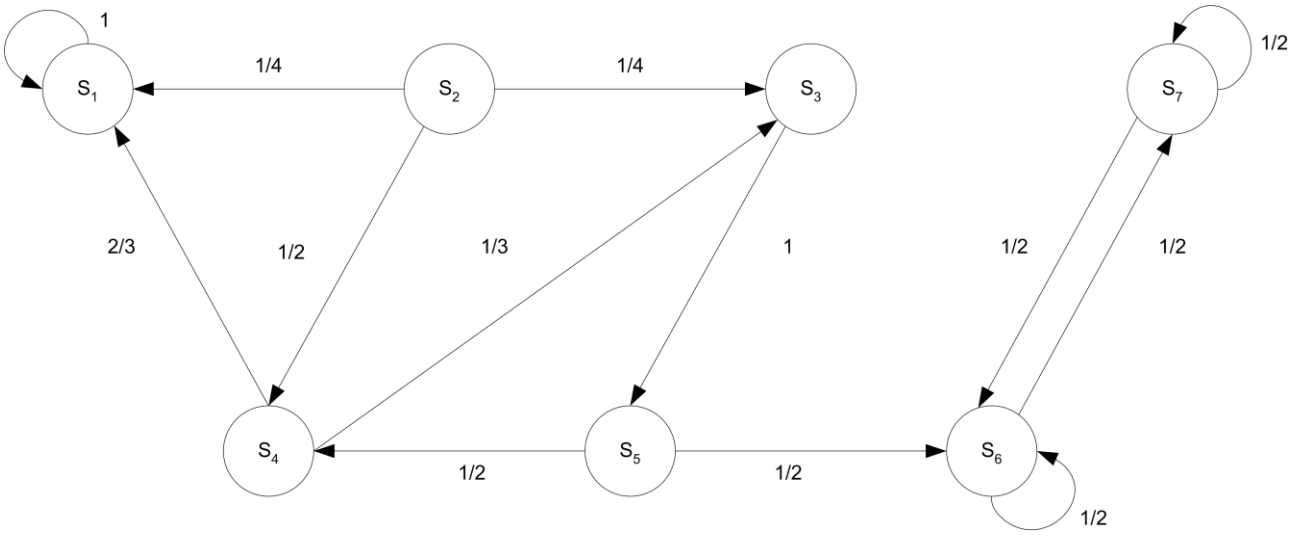
$$\begin{bmatrix} -0.3 & 0 & 0.5 & 0 \\ 0.3 & -1 & 0.5 & 0 \\ 0 & 0.5 & -1 & 0.3 \\ 0 & 0.5 & 0 & -0.3 \end{bmatrix} \begin{bmatrix} p(00) \\ p(01) \\ p(10) \\ p(11) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Ponovo jednu jednačinu (recimo posljednju) možemo zamijeniti sa uslovom $p(00)+p(01)+p(10)+p(11)=1$:

$$\begin{bmatrix} -0.3 & 0 & 0.5 & 0 \\ 0.3 & -1 & 0.5 & 0 \\ 0 & 0.5 & -1 & 0.3 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} p(00) \\ p(01) \\ p(10) \\ p(11) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Rješavanjem ovog sistema dobijamo da je $p(00)=p(11)=1/3.2=5/16$ dok je $p(10)=p(10)=3/16$. Postoji više načina da se sada odredi stacionarno stanje za uslovne vjerovatnoće prvog reda te za vjerovatnoće događaja $p(0)$ i $p(1)$. Međutim, mišljenja smo da umjesto bilo kakve matematike možemo se pozvati na rezon odnosno na činjenicu da je u predmetnom slučaju sve simetrično što vodi zaključku da je u stacionarnom režimu $p(0)=p(1)=0.5$. Sada se jednostavno mogu odrediti uslovne vjerovatnoće prvog reda.

2.11. Dat je dijagram stanja Markovljevog procesa. Da li je proces ergodičan? Odrediti tranzijentna, rekurentna, periodična, aperiodična i apsorbujuća stanja u dijagramu. Prikazati matricu odgovarajuće sistema.



Rješenje: Stanja S_2, S_3, S_4 i S_5 su **tranzijentna** jer iz barem jednog stanja u koje se može iz njih doći ne može da se vratimo u to stanje. Rekurentna (esencijalna) stanja su u našem slučaju S_1, S_6 i S_7 jer se iz svakog stanja u koje se iz njih može doći može izvršiti povratak. Stanja S_3, S_4, S_5 su periodična jer se možemo kretati između njih periodičnom putanjom dužine $d=3$ ($d>1$). Stanje S_1 je apsorbujuće jer kada se stigne u ovo stanje više se ne može napustiti. Predmetni proces nije ergodičan jer se iz stanja S_1 ne može u konačnom broju koraka stići u sva druga stanja. Kao što smo rekli u pitanju je apsorbujuće stanje.

2.12. Bacaju se dvije kocke. X je binarni događaj koji daje 1 ako je vrijednost bačena sa prvom kockom veća od vrijednosti bačene drugom kockom dok je događaj Y binarni događaj koji je jednak

1 ako je suma na dvije kocke parna. Odrediti pojedine vjerovatnoće događaja X, Y i združenog događaji X i Y kao i uslovne vjerovatnoće. Odrediti entropije i međusobnu informaciju.

Rješenje: Uradite sami ovaj zadatak. Za provjeru međusobna informacija je mala i iznosi $I(X;Y)=0.0207$bita/pikselu.

2.13. Posmatrajte binarni simetrični kanal kojim se prenose poruke: 1 sa vjerovatnoćom p i 0 sa vjerovatnoćom 1-p. Vjerovatnoća da je prenesen isti simbol koji je poslat je jednak P. Odrediti vjerovatnoću da je 1 koje je primljeno na izlazu ujedno i poruka koja je poslata.

Rješenje: Premda je ovaj primjer više nego jednostavan vrijedi ga objasniti. Na prijemniku imamo primljeno 1 a to se dešava sa vjerovatnoćom pP+(1-p)(1-P). Ovo je za nas događaj koji se dogodio. Postavlja se pitanje kolika je vjerovatnoća da je 1 primjemljeno i 1 poslato. Ta vjerovatnoća je pP pa je vjerovatnoća uslovnog događaja da je 1 poslato ako je 1 primljeno jednaka:

$$\frac{pP}{pP+(1-p)(1-P)}$$

2.14. Diskretna slučajna promjenljiva ξ uzima vrijednosti: 1, 2, 3, 4 i 5 sa vjerovatnoćama 0.1, 0.3, 0.4, 0.1, 0.1. Odrediti raspodjelu i očekivanu vrijednost za slučajnu promjenljivu $\eta=(\xi-3)^2$. Odrediti združene raspodjele kao i odgovarajuće uslovne vjerovatnoće ove dvije slučajne promjenljive.

Rješenje. Formirajmo tabelu sa vrijednostima promjenljivih ξ i η i sa odgovarajućim vjerovatnoćama:

ξ	1	2	3	4	5
η	4	1	0	1	4
P	0.1	0.3	0.4	0.1	0.1

Sada možemo pojednostaviti prikaz za slučajnu promjenljivu η kao:

η	0	1	4
p	0.4	0.4	0.2

Združena raspodjela ove dvije slučajne promjenljive se može prikazati sledećom tabelom.

$p(\xi,\eta)$	ξ=1	ξ=2	ξ=3	ξ=4	ξ=5	$p(\eta)$
η=0	0	0	0.4	0	0	0.4
η=1	0	0.3	0	0.1	0	0.4
η=4	0.1	0	0	0	0.1	0.2
$p(\xi)$	0.1	0.3	0.4	0.1	0.1	

$p(\xi\|\eta)$	ξ=1	ξ=2	ξ=3	ξ=4	ξ=5
η=0	0	0	1	0	0
η=1	0	0.75	0	0.25	0
η=4	0.5	0	0	0	0.5

$p(\eta\|\xi)$	ξ=1	ξ=2	ξ=3	ξ=4	ξ=5
η=0	0	0	1	0	0

η=1	0	1	0	1	0
η=4	1	0	0	0	1

2.15. Pronađite više podataka o spektralnim karakteristikama slučajnih procesa i napišite esej/seminarski rad o tome.

2.16. Generiše se vrijednost iz skupa od N slučajnih događaja. Neka su numeričke karakteristike ovih događaja koje proučavamo a_i, i=1,...,N. Neka su događaji sa vjerovatnoćama p_i, i=1,...,N. Neka je p_1+...+p_N=1. Informacija o događaju se prenosi preko posrednika koji sa vjerovatnoćom P_{ii} prenosi pravilno poruku da se dogodio događaj a_i dok sa vjerovatnoćama P_{ij} generiše poruke da se dogodio događaj a_j a zapravo se dogodio događaj a_i. Odrediti vjerovatnoće informacije o događajima koja je primljena ako je između generisanja ovog događaja (izvor informacija) i vas bilo M prenosioca informacija sa istim karakteristika vezanim za tačnost prenosa informacije. **Napomena:** U rješavanju ovog problema pogodno je vjerovatnoće događaja zapisati u matričnoj formi.

Rješenje. Vjerovatnoća da je događaj a_l primljen je jednaka vjerovatnoći da je i=a_1 poslato i da je izmjenjeno u a_l, plus da je poslato a_i=2 i izmjenjeno u a_l, poslato a_i=3 i izmjenjeno u a_l, ..., poslato a_i=a_l i ostalo neizmjenjeno itd. Ovo se može zapisati kao:

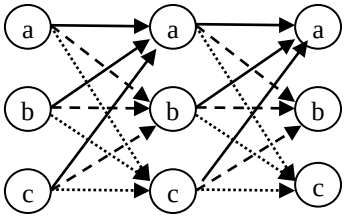
$$P(y=a_l)=\sum_{i=1}^{N}P(x=a_i)P(y=a_l\mid x=a_i)=\sum_{i=1}^{N}p_iP_{il}$$

Ovo se može u matričnoj formi zapisati kao:

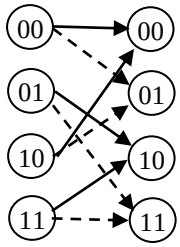
$$\begin{bmatrix}P(y=a_1)\\P(y=a_2)\\\dots\\P(y=a_N)\end{bmatrix}=\begin{bmatrix}P_{11}&P_{12}&\dots&P_{1N}\\P_{21}&P_{22}&\dots&P_{2N}\\\vdots&\vdots&\ddots&\vdots\\P_{N1}&P_{N2}&\dots&P_{NN}\end{bmatrix}\begin{bmatrix}p_1\\p_2\\\dots\\p_N\end{bmatrix}$$

2.17. Prikažite trelise za procese opisane u zadacima 2.9 i 2.10.

Rješenje. Za prvi sistem u pitanju je ternarni kod pa imao trelis u sledećem obliku.



Ovdje smo punom linijom označavali prelaze forsirane sa simbolom a, dok su prelazi forsirani sa b prikazani sa isprekidanim linijama, konačno prelazi forsirani sa simbolom c su prikazani sa tačkastim linijama. Za sledeći sistem prikazaćemo samo jedan blok u trelisu a to je bio Markovljev sistem drugog reda.



2.18. Posmatrajte izvor koji proizvodi alfabet veličine N sa vjerovatnoćama simbola p_i, i=1,...,N. Odrediti vjerovatnoće koje maksimizuju entropiju. **Napomena 1:** Koristiti ograničenje da je suma vjerovatnoća jednaka 1. **Napomena 2:** Kada odredite vjerovatnoće koje daju nulu prvog izvoda morate dokazati da se doista radi o maksimumima.

Rješenje: Entropiju možemo zapisati kao:

$$H(X) = H(p_1, p_2, ..., p_N) = -\sum_{i=1}^{N} p_i \log p_i$$

Odredimo prvi izvod ove funkcije po vjerovatnoći p_j:

$$\frac{\partial H(p_1, p_2, ..., p_N)}{\partial p_j} = -\log p_j - 1 = 0$$

Radi jednostavnosti podrazumijevamo da je logaritam sa osnovnom e (probajte sa drugim osnovama). Sada se dobija da je "maksimum" za $p_j=e^{-1}$. Međutim, ovo nije tačno rješenje jer očigledno $p_1=p_2=...=p_N=e^{-1}$ pa slijedi $p_1+p_2+...+p_N=Ne^{-1}\neq 1$ što je nemoguće. Naime, mi moramo da u proces izvođenja i određivanje minimuma ugradimo ograničenje da mora da važi $p_1+p_2...+p_N=1$. Ovo se radi na jednostavan način tako što na $H(p_1,p_2,...,p_N)$ dodamo neki umnožak $p_1+p_2...+p_N$ -1. Pošto ovaj uslov zapravo treba da da nulu neće se promjeniti vrijednost entropije. Dakle, modifikovana entropija je:

$$\begin{aligned} H'(p_1, p_2, ..., p_N) &= H(p_1, p_2, ..., p_N) + \lambda(p_1 + p_2 + p_3 + ... + p_N - 1) \\ &= -\sum_{i=1}^{N} p_i \log p_i + \lambda(p_1 + p_2 + p_3 + ... + p_N - 1) \end{aligned}$$

Koeficijenat λ se naziva Langraževim množiocem. Sada izvršimo diferenciranje funkcije $H'(p_1,p_2,...,p_N)$ po bilo kojoj vjerovatnoći (trebalo bi po svima ali uvijek dobijemo isti rezultat):

$$\frac{\partial H(p_1, p_2, ..., p_N)}{\partial p_j} = -\log p_j - 1 + \lambda = 0$$

Dakle sada dobijemo da je $p_j=e^{1-\lambda}$ a kako su sve vjerovatnoće iste to zaključujemo da mora da važi $p_j=1/N$ a ovo se može postići pogodnim izborom λ odnosno λ je jednako: $\lambda=1-\log N$. Dakle, maksimalna entropija se dobija kada je vjerovatnoća svakog simbola ista i iznosi $H(X)=\log N$. Kako dokazati da je ovo zaista maksimum. Taj dokaz se vrši na osnovu drugog izvoda koji bi morao u datoj tački biti negativan (prvi izvod nula i drugi izvod negativan sugeriše maksimum, prvi izvod nula i drugi izvod pozitivan je dokaz minimuma a oba izvoda 0 označavaju tzv. prevojnu tačku). Drugi izvod iznosi (prvi izvod za $p_j=1/N$):

$$\frac{\partial^2 H(p_1, p_2, ..., p_N)}{\partial p_j^2} = -\frac{1}{p_j} = -N < 0$$

2.19. Neka su vjerovatnoće simbola ¼ i ¾. Kolika je entropija ovog sistema. Kolika je entropija sistema kod kojeg se šalju kombinacije od po dva simbola sa datim vjerovatnoćama ako su simboli međusobno nezavisni. Ponoviti proceduru za tri nezavisna simbola. Koliku entropiju očekujete kod *n* nezavisnih ponavljanja simbola po istim pravilima.

Rješenje: Entropija događaja je jednaka:

$$H(p) = -p\log_2 p - (1-p)\log_2(1-p) = 0.8113$$

Združeni događaj pojavljivanja dva simbola ima ukupno 4 moguće pojave sa vjerovatnoćama: 1/16 (recimo 00), 3/16 (01), 3/16 (10) i 9/16 (11). Entropija ovog alfabeta (sa simbolima {00, 01, 10, 11}) je:

$$H_2=1.6226$$

Za tri simbola alphabet ima vjerovatnoće: {1/64, 3/64, 3/64, 3/64, 9/64, 9/64, 9/64, 27/64} pa je entropija jednaka:

$$p=[1/64\ 3/64\ 3/64\ 3/64\ 9/64\ 9/64\ 9/64\ 27/64]$$
$$H_3=\text{-sum}(p.*\log2(p))= 2.4338$$

Kada malo bolje sagledamo situaciju dolazimo do činjenice da je $H_2=2H(p)$, $H_n=3H(p)$ dok bi H_n bilo jednako $nH(p)$. Dakle, nismo od početka morali da računamo H_2, H_3 itd već smo mogli da uočimo ovu zavisnost i da je iskoristimo radi lakšeg određivanja entropije združenih događaja. Riječ je o jednostavnoj činjenici da se entropija združenog događaja može sračunati kao:

$$H(X,Y)=H(X)+H(Y|X)$$

U ovom slučaju entropija dva nezavisna događaja je jednaka $H(Y|X)=H(Y)$ jer događaj od kojega Y ne zavisi ne nosi nikakvu informaciju o samom Y odnosno ne umanjuju neodređenost o njemu. Dakle $H(X,Y)=H(X)+H(Y)$. Kako su ovdje događaji X i Y sa istom entropijom slijedi da je $H(X)=H(Y)=H(p)$ tako da je $H(X,Y)=2H(p)$. Na sličan način se može pokazati i za više uzastopnih simbola nezavisnih i na isti način distribuiranih (ovaki procesi se u engleskoj literaturi označavaju kao i.i.d. – independent and identically distributed).

2.20. **Extra zadatak za naprednije studente:** $D(p\|q)$ je konveksna za par funkcija gustine raspodjele (p,q) ako su (p_1,q_1) i (p_2,q_2) funkcije gustine raspodjele:

$$D(\mu p_1 + (1-\mu)p_2) \| \mu q_1 + (1-\mu)q_2) \le \mu D(p_1 \| q_1) + (1-\mu) D(p_2 \| q_2)$$

za svako $0 \le \mu \le 1$.

Rješenje: Pođimo od definicije međusobne informacije:

$$D(p \| q) = \sum_{x \in A} p(x) \log \frac{p(x)}{q(x)}$$

Lijeva strana izraza u razvijenoj formi je:

$$D(\mu p_1 + (1-\mu)p_2) \,\|\, \mu q_1 + (1-\mu)q_2) = \sum_{x\in A} [\mu p_1(x) + (1-\mu)p_2(x)] \log \frac{[\mu p_1(x) + (1-\mu)p_2(x)]}{[\mu q_1(x) + (1-\mu)q_2(x)]}$$

dok je desna strana:

$$\mu D(p_1 \,\|\, q_1) + (1-\mu) D(p_2 \,\|\, q_2) = \mu \sum_{x\in A} p_1(x) \log \frac{p_1(x)}{q_1(x)} + (1-\mu) \sum_{x\in A} p_2(x) \log \frac{p_2(x)}{q_2(x)}$$

Posmatrajmo sledeću nejednakost koja je dokazana u okviru primjera 2.2

$$\sum_{i=1}^{n} \left(a_i \log \frac{a_i}{b_i} \right) \geq \sum_{i=1}^{n} a_i \log \left(\sum_{i=1}^{n} a_i \Big/ \sum_{i=1}^{n} b_i \right)$$

Ako je $a_i=\lambda_i p_i$, $b_i=\lambda_i p_i$ i $\lambda_1=\mu$ i $\lambda_2=1-\mu$ dobijamo:

$$\sum_{i=1}^{2} (\lambda_i p_i(x)) \log \frac{\sum_{i=1}^{2} (\lambda_i p_i(x))}{\sum_{i=1}^{2} (\lambda_i q_i(x))} \leq \sum_{i=1}^{2} \lambda_i p_i(x) \log \frac{\lambda_i p_i(x)}{\lambda_i q_i(x)}$$

$$[\mu p_1(x) + (1-\mu) p_2(x)] \log \frac{[\mu p_1(x) + (1-\mu) p_2(x)]}{[\mu q_1(x) + (1-\mu) q_2(x)]} \leq \mu p_1(x) \log \frac{p_1(x)}{q_1(x)} + (1-\mu) p_2(x) \log \frac{p_2(x)}{q_2(x)}$$

Ako sada obije strane sumiramo po elementima skupa A dobijamo:

$$\sum_{x\in A} [\mu p_1(x) + (1-\mu) p_2(x)] \log \frac{[\mu p_1(x) + (1-\mu) p_2(x)]}{[\mu q_1(x) + (1-\mu) q_2(x)]} \leq \mu \sum_{x\in A} p_1(x) \log \frac{p_1(x)}{q_1(x)} + (1-\mu) \sum_{x\in A} p_2(x) \log \frac{p_2(x)}{q_2(x)}$$

Čime smo zapravo kompletirali dokaz:

$$D(\mu p_1 + (1-\mu) p_2) \,\|\, \mu q_1 + (1-\mu) q_2) \leq \mu D(p_1 \,\|\, q_1) + (1-\mu) D(p_2 \,\|\, q_2)$$

Poglavlje III
UVOD U KODIRANJE IZVORA

3.1. Konvergencije po vjerovatnoći

Dajemo nekoliko osnovnih matematičkih pojmova vezanih za konvergenciju po vjerovatnoći kako bi zainteresovani studenti mogli da dokažu važnu teoremu o asimptotskoj ekviparticionoj osobini koju ćemo formulisati i objasniti na primjeru a nećemo dokazivati. Ova teorema je osnova na kojoj se može temeljiti kodiranje izvora.

Definicija Niz $X_1, X_2, ..., X_n$ konvergira u vjerovatnoći, ako $P[|X_n - X| < \varepsilon] > 1 - \delta$ za svako $n > n_0$. Postoje i drugi oblici konvergencije (za nas manje bitni).
Primjer. Neka je X_i niz slučajnih promjenljivih i neka:

$$S_n = \frac{1}{n}\sum_{i=1}^{n} X_i$$

je srednja vrijednost ovog niza. Neka je tačna srednja vrijednost ovog niza S. Tada:

$$P\left[|S_n - S| > \varepsilon\right] \to 0 \text{ kada } n \to \infty$$

Jednostavnije, ovo se može reći da $S_n \to S$ u vjerovatnoći.

Jedan od načina da kvantifikujemo kako stižemo do konvergencije (brzinu konvergencije) je u smislu Markovljeve relacije, koja za pozitivnu slučajnu promjenljivu i $\delta > 0$

$$P\left[X \geq \varepsilon\right] \leq E\left[X\right]/\delta$$

Odavde se može zapisati Chebyshev-ljeva nejednakost za slučajnu promjenljivu Y sa srednjom vrijednošću μ i varijansom σ^2

$$P\left[|Y - \mu| > \varepsilon\right] \leq \sigma^2 / \varepsilon^2$$

Sada možemo da dokažemo sljedeću teoremu.

Teorema. Neka su $X_1, X_2, ..., X_n$ nezavisne slučajne promjenljive sa istom funkcijom gustine raspodjele $p(x)$. Tada:

$$-\frac{1}{n}\log p\left(X_1, X_2, ..., X_n\right) \to H(X)$$

po vjerovatnoći.

Dokaz. Kako su X_i nezavisne slučajne promjenljive sa istom gustinom raspodjele, isto važi i za $\log p(X_i)$. Odavde slijedi:

$$-\frac{1}{n}\log p\left(X_1, X_2, ..., X_n\right) = -\frac{1}{n}\sum_{i=1}^{n}\log p\left(X_i\right)$$

Sada možemo primijeniti osobinu da srednja vrijednost odbiraka konvergira u vjerovatnoći ka srednjoj vrijednosti ansambla:

$$-\frac{1}{n}\sum_{i-1}^{n}\log p\left(X_i\right) \to -E\left[\log p(X)\right] = H(X)$$

čime je dokaz završen. Odavde dalje slijedi: $p(x_1, x_2, ..., x_n) \approx 2^{-nH(X)}$ što se naziva asimptotskom ekviparticionom osobinom.

3.2. Asimptotska ekviparticiona osobina

Ono što smo zaključili iz prethodnog izlaganja je činjenica da se entropija događaja (alfabeta, skupa) X može procijeniti na osnovu vjerovatnoće sekvence događaja $X_1, X_2, ..., X_n$ koji su nezavisni i na isti način distribuirani (i.i.d.) kod kojih se realizacija događaja X ponavlja:

$$H(X) \approx -\frac{1}{n}\log p(X_1, X_2, ..., X_n)$$

Ova relacija važi za relativno veliko *n*. Dalje možemo napisati da važi:

$$p(X_1, X_2, ..., X_n) \approx 2^{-nH(X)}$$

Posmatrajmo sledeći primjer.

Primjer. Posmatrajmo $X \in \{0,1\}$ sa $p(1) = p$ i $p(0) = q$. Vjerovatnoća sekvence $X_1, X_2, ..., X_n$ je: $p^{\sum_{i=1}^{n} X_i} \; q^{\; n-\sum_{i=1}^{n} X_i}$. Posmatrajmo sledeći slučaj *q*=0.7 i *p*=0.3 posmatrajmo $n = 10$ i vjerovatnoće sljedećih događaja:

Niz	Vjerovatnoća	Broj kombinacija	Ukupna vjerovatnoća
0000000000	0.0282	1	0.0282
0000000001	0.0121	10	0.121
0000000011	0.005	45	0.225
0000000111	0.0022	120	0.264
0000001111	0.00095	210	0.1995
0000011111	0.0004	252	0.1008
0000111111	0.00017	210	0.036
0001111111	0.000075	120	0.009
0011111111	0.000032	45	0.00144
0111111111	0.0000138	10	0.000138
1111111111	0.0000059	1	0.0000059

Jasno je da svih 2^n poruka nema istu vjerovatnoću. Uočljivo je da one sekvence koje imaju približno *np* jedinica imaju najveću totalnu vjerovatnoću. Ovdje se vidi da je to za onu sekvencu koja sadrži 3 jedinice. To se može smatrati tipičnom sekvencom. Kolika je vjerovatnoća događaja $p(X_1, X_2, ..., X_n)$? Iz prethodne priče slijedi da je to blisko $2^{-nH()}$ sa velikom vjerovatnoćom. U prethodnom primjeru može se reći da su najvjerovatnije one sekvence koje imaju *np* jedinica i sve su takve sekvence jednakovjerovatne. To je prosto zakon velikih brojeva. Skup sekvenci koje se pojavljuju najčešće nazivaju se **tipične sekvence**.

Posmatrajmo primjer 3.1.2. $H(X) = 0.8819$ i $nH(X) = 8.819$. Tada je $2^{-nH(X)} = 0.00223$. Ovo je očigledno veoma blisko vjerovatnoći pojedinačnog događaja da se pojave tri jedinice. Tipičnim skupom se definiše onaj skup sekvenci kod kojega je vjerovatnoća pojavljivanja sekvence $p(X_1, X_2, ..., X_n)$ u granicama koje su bliske $2^{-nH(X)}$.

$$2^{-n(H(X)+\varepsilon)} \le p(x_1, x_2, ..., x_n) \le 2^{-n(H(X)-\varepsilon)}$$

Tipični skup definiše sledeća teorema.

Teorema. 1. Ako je $(x_1, x_2, ..., x_n) \in A_\varepsilon^{(n)}$ tada $H(X) - \varepsilon \le -\log p(x_1, x_2, ..., x_n)/n \le H(X) + \varepsilon$

2. $\Pr\{A_\varepsilon^{(n)}\} > 1 - \varepsilon$ za n dovoljno veliko.

3. $\|A_\varepsilon^{(n)}\| \le 2^{n(H(X)+\varepsilon)}$, gdje $\|A\|$ označava broj elemenata u skupu.

4. $\|A_\varepsilon^{(n)}\| \ge (1-\varepsilon)2^{n(H(X)+\varepsilon)}$ za n dovoljno veliko.

Tumačenje teoreme. Po (1) skoro svi elementi u tipičnom skupu su skoro jednako vjerovatni. Po (2) tipični set se pojavljuje sa vjerovatnoćom koja je bliska 1. Po (3) i (4) broj elemenata u tipičnom

skupu je blizu 2^{nH}. Dakle, na ovaj način smo došli do podataka o vjerovatnoći sekvenci u tipičnom skupu i do broja članova tipičnog skupa. Za veliko n vjerovatnoća pojave tipične sekvence je približno 1 ($2^{-nH(X)}2^{nH(X)}$). Podsjetimo se primjera sekvence sa 1000 nula i jedinica i vjerovatnoćom pojave jedinice od 0.5 kada su sve kreirane sekvence imale od 450 do 550 jedinica.

Primjer: Posmatrajmo primjer kada imamo 100 ponavljanja a da je vjerovatnoća pojave jedinice p=0.2 dok je vjerovatnoća pojave nule q=0.8.

Vjerovatnoća pojedinačnog događaja sa 20-jedinica i 80 nula je: $p^{20}q^{80}\approx 1.853\cdot 10^{-22}$. Broj ovakvih kombinacija je:

$$\binom{100}{20} \approx 5.36\cdot 10^{20}$$

Totalna vjerovatnoća ovih događaja je $5.36\cdot 10^{20}\cdot 1.853\cdot 10^{-22} \approx 0.0993$.

Entropija $H(X)$=0.7219, $2^{nH(x)}$=5.38 10^{21} a $2^{-nH(X)}$ =$1.856\cdot 10^{-22}$. Dakle, ovdje je uočljivo nekoliko stvari:

$$p^{20}q^{80}\approx 2^{-nH(X)}$$

Dalje uočavamo da je broj elemenata u tipičnom skupu na ovaj način definisanom $\binom{100}{20} \approx 5.36\cdot 10^{20}$ manji od $2^{nH(x)}$. Ovdje je taj odnos gotovo deset puta. Međutim, treba da imamo na umu da u tipičnu sekvencu ne ulaze isključivo vrijednosti gdje se simboli pojavljuju np i nq puta već da je dozvoljeno da elementi budu i iz sekvenci gdje se npr. jedinice pojavljuju približno np puta. Tako na primjer broj sekvenci sa 19-jedinica je $1.32\cdot 10^{20}$ dok je broj sekvenci sa 21-jedinicom $2.04\cdot 10^{21}$. Totalna vjerovatnoća sekvenci sa 19 jedinica je oko 0.0978 dok je totalna vjerovatnoća sekvenci sa 21-jedinicom 0.0945. Dakle, $2.72\cdot 10^{21}$ sekvenci uzimaju ukupnu vjerovatnoću od 0.2916 u odnosu na ostale sekvence koje uzimaju preostalu vjerovatnoću 0.7084. Ovih $2.72\cdot 10^{21}$ čini oko 2 milijardita dijela ukupnog broja svih sekvenci. Pogledajmo sada situaciju

n=20	0.0993	0.0993
n=19	0.0981	0.1974
n=21	0.0946	0.2920
n=18	0.0909	0.3829
n=22	0.0849	0.4678
n=17	0.0789	0.5467
n=23	0.0720	0.6187
n=16	0.0638	0.6825
n=24	0.0577	0.7402
n=15	0.0481	0.7883
n=25	0.0439	0.8322
n=14	0.0335	0.8657
n=26	0.0316	0.8973
n=27	0.0217	0.9190

Ukupan broj sekvenci koje daju vjerovatnoću kumulativnu preko 90% je $2.98\cdot 10^{24}$ što je oko 2 milionita dijela ukupnog broja sekvenci. Za kodiranje ovog broja sekvenci potrebno je oko $\log_2(2.98\cdot 10^{24})$ što iznosi nešto preko 81 odnosno 82 bita. Naravno, nismo rekli na koji način da

obezbjedimo ovakvo kodiranje te kako da kodiramo preostale sekvence. Konačno, uočimo da su sve sekvence koje pridodajemo visokovjerovatnom skupu se nalaze oko n=20 odnosno da smo ovdje uključili sve sekvence od n=14 do n=27.

Definicija Oznaka $a_n \doteq b_n$ znači:

$$\lim_{n\to\infty} \frac{1}{n} \log \frac{a_n}{b_n} = 0$$

Ova notacija se može primijeniti za sekvence $a_n = e^{3n+1}$ i $b_n = e^{3n}$, kao i $a_n = e^{3n+\sqrt{n}}$ i $b_n = e^{3n}$, ali ne može na $a_n = e^{4n}$ i $b_n = e^{3n}$.

Ovu ćemo oznaku koristiti da bi razmatrali jednu važnu karakteristiku tipične sekvence. Dakle, uočili smo da je tipična sekvenca mali skup koji daje veliku vjerovatnoću, ali nije jasno da li je to najmanji takav skup. Sada ćemo pokazati da bilo koji skup sa velikom vjerovatnoćom mora imati značajno preklapanje sa tipičnom sekvencom.

Definicija. Neka je skup $B_\delta^{(n)} \in \chi^n$ zadovoljava $\Pr\left(B_\delta^{(n)}\right) < 1-\delta$. Ovakav se skup naziva visokovjerovatnim.

Teorema . Neka su X_i slučajne promjenljive sa funkcijom raspodjele $p(x)$. Za $\delta < 1/2$ i neko $\delta' > 0$ ako je $\Pr\left(B_\delta^{(n)}\right) \geq 1-\delta'$ tada

$$\frac{1}{n} \log \left|B_\delta^{(n)}\right| > H - \delta'$$

za n dovoljno veliko. Dakle, $B_\delta^{(n)}$ mora sadržati najmanje 2^{nH} elemenata do prvog reda u eksponentu. Vidjeli smo da $A_\varepsilon^{(n)}$ sadrži oko $2^{nH\pm\varepsilon}$ elemenata. Ovaj skup je reda veličine istog broja članova kao i $B_\delta^{(n)}$, $\left|B_\delta^{(n)}\right| \doteq \left|A_\delta^{(n)}\right| \doteq 2^{nH}$.

Primjer. Pretpostavimo da formiramo visokovjerovatni skup koji obuhvata 120 sekvenci sa tri jedinice i sedam nula iz prethodnog primjera. Pored toga u ovaj skup uključimo sekvencu sa svim nulama te sedam sekvenci sa jednom jedinicom i 9 nula. Ukupna vjerovatnoća ovog visokovjerovatnog skupa je:

$$0.2664+7\cdot 0.0121+0.0282=0.3793$$

Pretpostavimo da ove sekvence kodiramo sa 7 bita (2^7=128) i da im uvodimo jedan bit (npr. 1) koji im prethodi kako bi naglasili da je u pitanju visokovjerovatni skup koji se kodira sa 7 bita. Ostalih 1024-128=896 sekvenci kodirajmo sa 10 bita i prefiksom (prethodnikom) 0 koji naglašava da je ovo sekvenca van tipičnog skupa. Prosječna dužina kodne riječi sada je:

$$0.3793\cdot(7+1)+0.6207\cdot(10+1)=9.8621\text{bita}$$

Dakle, na ovaj mora se priznati veoma primitivan način izvršili smo uštedu od 0.1379bita na poruku od 10 bita (ušteda oko 1.4%). Ovim smo dokazali da asimptotska ekviparticiona osobina može voditi ka kompresiji podataka. Formirajmo sada visokovjerovatni skup tako da obuhvati 128 najvjerovatnijih sekvenci: jednu sa svim nulama, 10 sa jednom jedinicom, 45 sa dvije jedinice i 72 iz tipičnog skupa sa tri jedinice (dakle ovaj visokovjerovatni skup čak i predmetnom slučaju sa malom dužinom riječi ima značajno preklapanje sa tipičnim skupom). Ukupna vjerovatnoća ovog skupa je:

$$0.0282+0.121+0.225+72\cdot 0.0022=0.5326$$

Riječ je o skupu sa znatno većom vjerovatnoćom nego onaj koji obuhvata kompletan tipični skup. Kodiranjem kako je prethodno objašenjeno ovog skupa dobijamo kodnu riječ dužine:

$$0.5326\cdot(7+1)+0.4674\cdot(10+1)=9.4022\text{bita}$$

Ušteda je u ovom slučaju ponovo veoma primitivnog kodiranja veća i iznosi oko 6%.

Uočimo da u primjeru najvjerovatniji pojedinačni događaj a to je slučaj svih 10 nula se ne pojavljuje u tipičnoj sekvenci. Stoga je uveden pojam visokovjerovatnog skupa kojega čine proizvoljne sekvence čija je ukupna vjerovatnoća bliska jedan (ili barem veća od neke velike zadate vjerovatnoće). Može se pokazati da svaki visokovjerovatni skup mora da ima veliko preklapanje sa tipičnim skupom. Tipična sekvenca ili visokovjerovatni skupovi su osnova koja se koristi za kompresiju podataka. Naime, ideja je da onaj broj događaja (obično relativno mali skup) sa velikim vjerovatnoćama kodiramo sa malim brojem bita a da veliki broj događaja sa malim vjerovatnoćama kodiramo sa manjim brojem bita kako bi prosječni broj bita sa kojim se kodira sekvenca učinili što je manjim mogućim i na taj način štedjeli na memorijskom prostoru za smještaj podataka ili na korišćenju komunikacionog kanala za prenos informacija. Nažalost od osnovne ideje do realizacije je dosta dugačak put.

Na primjer, posmatrajmo skup sa binarnim događajima, gdje je vjerovatnoća jedinice 0.9 sa $n=10$ ponavljanja. Skup $A_\varepsilon^{(n)}$ sadrži one događaje, gdje se 1 pojavljuje 9 puta, ali ne sadrži pojedinačno najvjerovatniji događaj da skup sadrži samo jedinice. Skup $B_\delta^{(n)}$ će uključiti sve najvjerovatnije sekvence, uključujući i onu koja sadrži sve jedinice.

3.2 Entropijski odnosi

Vidjeli smo da entropija *n* uzastopnih događaja sa istom raspodjelom teži $nH(X)$. Nezavisni slučajni događaji sa istom raspodjelom se često označavaju i.i.d.. Entropijski odnosi nam omogućavaju razmatranje entropije za niz slučajnih promjenljivih koje nijesu nezavisne. Naime, nezavisni i na isti način distribuirani procesi nijesu dobar model za realne poruke.

Definicija. Entropijski odnos za stohastički proces $\{X_i\}$ definiše se kao:

$$H(\chi)=\lim_{n\to\infty}\frac{1}{n}H(X_1,X_2,...,X_n)$$

pod pretpostavkom da postoji granična vrijednost.

Slična veličina se može usvojiti za uslovnu entropiju:

$$H'(\chi)=\lim_{n\to\infty}\frac{1}{n}H(X_n\mid X_{n-1},X_{n-2},...,X_1)$$

Ovo su suštinski dva različita koncepta koji za i.i.d. sekvence vode ka istom.

Teorema. Za stacionarni proces $H(X_n\mid X_{n-1},X_{n-2},...,X_1)$ je opadajuća funkcija od n i ima graničnu vrijednost $H'(\chi)$.

Teorema. Za neki stacionarni proces postoje dvije definicije entropije (konvergiraju) i međusobno su jednake:

$$H'(\chi)=H(\chi)$$

Na osnovu ovoga se može (ovdje bez dokaza) zapisati generalizacija AEP teoreme da za niz slučajnih promjenljivih sa identičnom funkcijom raspodjele (ali ne i nužno nezavisnih) važi:

$$-\frac{1}{n}\log p(X_1,X_2,...,X_n)=H(\chi)$$

Na osnovu ove generalizacije moguće je definisati tipičnu sekvencu i odrediti broj tipičnih sekvenci (približno $2^{nH(\chi)}$ svaka sa vjerovatnoćom $2^{-nH(\chi)}$). Za predstavljanje ove sekvence je neophodno približno $nH(\chi)$.

Alternativni način da se razmatra entropija za proces, koji nije i.i.d. je preko odgovarajućeg Markovljevog modela i uslovnih vjerovatnoća $p(s_i\mid s_{i_1},s_{i_2},....,s_{i_m})$. Neka je: $I(s_i\mid s_{i_1},s_{i_2},....,s_{i_m})=-\log p(s_i\mid s_{i_1},s_{i_2},....,s_{i_m})$. Uslovna entropija računata po alfabetu S preko simbola s_i se definiše kao:

$$H(S\mid s_{i_1},s_{i_2},....,s_{i_m})=-\sum_S p(s_i\mid s_{i_1},s_{i_2},....,s_{i_m})\log_2 p(s_i\mid s_{i_1},s_{i_2},....,s_{i_m}).$$

Uvodeći vjerovatnoću stanja $s_{i_1},s_{i_2},....,s_{i_m}$, kao $p(s_{i_1},s_{i_2},....,s_{i_m})$ entropija Markovljevog procesa se može definisati kao:

$$H(S)=\sum_S p(s_{i_1},s_{i_2},....,s_{i_m})H(s_i\mid s_{i_1},s_{i_2},....,s_{i_m})$$

$$H(S) = -\sum_{S^m}\sum_{S} p(s_{i_1}, s_{i_2}, ..., s_{i_m}) p(s_i \mid s_{i_1}, s_{i_2}, ..., s_{i_m}) \log p(s_i \mid s_{i_1}, s_{i_2}, ..., s_{i_m})$$

Kako važi: $p(s_{i_1}, s_{i_2}, ..., s_{i_m}) p(s_i \mid s_{i_1}, s_{i_2}, ..., s_{i_m}) = p(s_{i_1}, s_{i_2}, ..., s_{i_m})$ entropija se može izraziti kao:

$$H(S) = -\sum_{S^{m+1}} p(s_{i_1}, s_{i_2}, ..., s_{i_m}) \log p(s_i \mid s_{i_1}, s_{i_2}, ..., s_{i_m})$$

Da bi se odredila entropija Markovljevog izvora potrebno je naći stacionarne vjerovatnoće za svako stanje Markovljevog procesa. Ovo je teško poznavati unaprijed. Postoji koncept pridruženih sistema kojim se može izvršiti procjena Markovljevog izvora. Da bi upotrebu ovih sistema ilustrovali posmatrajmo Markovljevov proces prvog reda (kod procesa višeg reda samo se mijenja notacija). U ovom slučaju suštinski imamo rad sa dvije slučajne promjenljive pa formiramo vjerovatnoće $p(s_i), p(s_j)$ i združenu $p(s_j, s_i)$. Važe relacije $\sum_j p(s_j, s_i) = p(s_i) = p_i$ i $\sum_i p(s_j, s_i) = p(s_j) = p_j$. Pođimo od relacije (ovo slijedi na osnovu osobine međusobne informacije):

$$\sum_{S^2} p(s_j, s_i) \log\left[\frac{p(s_i)\, p(s_j)}{p(s_j, s_i)}\right] \le 0$$

Jednakost važi samo kada je $p(s_j, s_i) = p(s_j) p(s_i)$ (s_i i s_j su nezavisne slučajne promjenljive). Koristeći izraz za uslovnu vjerovatnoću $p(s_j, s_i) = p(s_i \mid s_j) p(s_j)$ dobija se:

$$\sum_{S^2} p(s_j, s_i) \log\left[p(s_i) / p(s_i \mid s_j)\right] \le 0$$

Dalje važi:

$$-\sum_{S^2} p(s_j, s_i) \log\left[p(s_i) / p(s_i \mid s_j)\right] \le \sum_{S^2} p(s_j, s_i) \log p(s_i)$$

Korišćenjem uslovnih raspodjela slijedi:

$$-\sum_j p(s_j) \sum_i p(s_i \mid s_j) \log p(s_i \mid s_j) \le \sum_i p(s_i) \log p(s_i) \le H(\bar{S})$$

gdje je $H(\bar{S})$ entropija sistema originalnih simbola koju ćemo zvati pridruženi sistem. Dakle, može se pisati da je:

$$\sum_j p(s_j) H(S \mid s_j) \le H(\bar{S})$$

pa je entropija Markovljevog sistema ograničena entropijom pridruženog sistema, koja se dobija za sisteme sa nultom memorijom i izvornim alfabetom $p(s_i) = p_i$. Jednakost važi za $p(s_j, s_i) = p_j p_i$. Dakle, ograničenja mogu samo da smanje entropiju. Nije teško pokazati da za n simbola Markovljevog koda važi ograničenje da je njegova entropija manja od $nH(\bar{S})$.

Primjer: Posmatran je fajl koji se sastoji od 270899 karaktera koji predstavljaju skraćen opis književnog djela "Doživljaji Toma Sojera" autora Marka Tvena i engleski prevod "Gorskog vijenca" autora Petra II Petrovića Njegoša. Nakon eliminacije bjelina i nealfabetskih karaktera fajl je sveden na 204596 karaktera. Broj pojavljivanja pojedinih karaktera kao i vjerovatnoća pojavljivanja dati su tabelarno. Smatramo da je relativno velika dužina ove sekvence pa da možemo smatrati da se ne radi o procjeni već tačnoj vrijedosti, međutim na oprez poziva neobičnost tekstova koji su upotrebljeni. Npr. za očekivati je da se riječ Tom neuporedivo češće javlja nego u standardnom tekstu što bi se pojavljivala.

a	15798	7.72%	b	3601	1.76%
c	4669	2.28%	d	7775	3.80%
e	25322	12.38%	f	4269	2.09%
g	4393	2.15%	h	12836	6.27%
i	13291	6.50%	j	736	0.36%
k	2625	1.28%	l	8235	4.03%
m	5014	2.45%	n	13849	6.77%
o	16863	8.24%	p	3092	1.51%
q	119	0.06%	r	12186	5.96%
s	13556	6.63%	t	18776	9.18%
u	6201	3.03%	v	2542	1.24%
w	4318	2.11%	x	184	0.09%
y	4054	1.98%	z	292	0.14%

Entropija ovog skupa je približno H=4.19bita po simbolu (`H=- sum(-pest.*log2(pest)`) gdje su `pest` estimacija vjerovatnoće pojavljivanja pojedinog simbola koja je data u tabeli. Dobijena entropija je tek nešto veća od 4.03 bita po simbolu koliko se pretpostavlja da iznosi entropija engleskog jezika dobijena na osnovu Markovljevog sistema nultog reda. Napominjemo da je naš tekst po svim standardima malo čudan ali opet nismo dobili značajno odstupanje od poznate vrijednosti entropije.

Sada dajemo dio koda za određivanje entropije dva simbola. Matrica Y broji koliko se puta pojavio neki simbol iza drugog simbola. Npr. `Y(1,1)` predstavlja koliko se puta pojavio simbol 'a' iza simbola 'a' dok `Y(1,2)` označava koliko puta se pojavio simbol 'b' nakon simbola 'a'. Ovo bi trebalo u konkretnom primjeru da bude proporcionalno uslovnoj vjerovatnoći p('b'|'a').

```
Y=zeros(26,26);
for k=1:length(B)-1
    Y(B(k)-'a'+1,B(k+1)-'a'+1)=Y(B(k)-'a'+1,B(k+1)-'a'+1)+1;
end
pest2=Y/sum(sum(Y));
H1=sum(-(pest2(:)+eps).*log2((pest2(:)+eps)))/2
```

Malu vrijednost eps=2^{-52} smo dodavali na vjerovatnoće da bi izbjegli log(0) jer se dešava da se neke kombinacije ne dogode nijednom. Dobijena entropija je 3.9bita po simbolu i manja je od entropije pojedinačnog događaja. Ovo je takođe približno očekivanom. Poznato je da entropija Engleskog jezik za Markovljev četvrtog reda iznosi nešto ispod 2.9 bita po simbolu. Međutim, postoje eksperimenti koji ukazuju da je moguće mnogo bolje kodiranje. Naime, ljudima su stavljani karakteri i oni su morali da pogode naredni karakter kada pogode naredni išli su dalje. Procjenjena entropija ovakvog događaja sa ljudskom inteligencijom je reda 1 bit po simbolu (ljudi pogađaju iz 2 puta koje je sljedeće slovo) ili maksimalno do 1.5 bita po simbolu. Dakle, u smislu predikcije ljudska inteligencija daleko prevazilazi prostu inteligenciju vezanu sa Markovljeve sistema sa uslovnim vjerovatnoćama.

3.3. Kompresija podataka – uvodna razmatranja

Osnovni smisao asimptotske ekviparticione osobine je da pokaže mogućnost kompresije podataka. Vidjeli smo da se tipična sekvenca pojavljuje većinom vremena za veliko n. Znamo da ASCII kod daje istu dužinu podatka za svaki od elemenata skupa. Ako bi elemente (sekvence) koje se često pojavljuju zamijenili sa manjim brojem bita, a one koje se pojavljuju veoma rijetko sa velikim brojem bita, ostvarili bi kompresiju. Ovdje se može donekle iskoristiti princip tipične sekvence. Naime, vidjeli smo da elemenata u tipičnoj sekvenci ima manje ili jednako $2^{-n(H(X)+\varepsilon)}$. Svaka od

ovih sekvenci se može kodirati sa $n(H+\varepsilon)+1$ bita. Pretpostavimo da svakoj tipičnoj sekvenci dodamo po jedan bit kao prefiks da bi se znalo da je to tipična sekvenca (npr. 0), to znači da se svaki element u tipičnoj sekvenci kodira sa $n(H+\varepsilon)+2$ bita. Svaka sekvenca koja nije u tipičnom skupu se može kodirati sa ne više od $n\log\|\chi\|+1$ bita i ako joj dodamo prefiks 1, dobili smo koder svih sekvenci u skupi χ^n. Naravno, ovo je samo gruba shema, ali može da posluži kao dobar primjer kako se može komprimovati sistem na osnovu tipične sekvence. Oznaka x^n podrazumijeva sekvencu (niz) $(x_1, x_2, ..., x_n)$.

Teorema 3.3.1. Neka je X^n sekvenca sa nezavisnim promjenljivima sa istom raspodjelom i neka je $\varepsilon > 0$. Tada postoji kodiranje koje preslikava sekvencu x^n u binarni niz tako da je preslikavanje 1 na 1 (kodiranje bez gubitaka) i da prosječna dužina koda zadovoljava:

$$\left[\frac{1}{n} l(X^n)\right] \le H(X)+\varepsilon$$

na n dovoljno veliko.

Tumačenje: Teorema kaže da je za dato *n* moguće kreirati kod koji će produkovati prosječnu dužinu kodne riječi koja je manja ili jednaka *H*(*X*)+ε gdje je ε neki mali broj (koji zavisi od *n* i opada sa *n*). U suštini kako ćemo kasnije pokazati za *n* konačno nije moguće proizvesti kod čija je prosječna dužina kodne riječi manja od *H*(*X*) ali se možemo primaći *H*(*X*). Teorema, nažalost, nije konstruktivna jer ne ukazuje na način na koji se kod koji daje predmetne performanse može postići. O tome ćemo nešto više riječi reći kasnije nakon što uvedemo potrebne pojmove. Ovo se ponekad naziva prvom Šenonovom teoremom (Claude Shannon). Klod Šenon otac teorije informacija je u dvije teoreme sažeo dva najvažnija ograničenja ove discipline (uskoro ćemo se upoznati i sa drugom Šenonovom teoremom) ali vrijedi napomenuti da nije bio uspješan u predlaganju kodova koji su u stanju da se primaknu granicama prognoziranim teoremom. Sam dokaz (ili skeč dokaza) ove teoreme dajemo nešto kasnije.

Kompresijom podataka u cilju umanjivanja redundancije će se u našim razmatranjima baviti **koder** ili **koder izvora**. Koder je uređaj koji vrši preslikavijanje **ulaznog alfabeta** u skup kodnih simbola. Podrazumjevaćemo da su ulazni alfabet i skup kodnih simbola konačni (ponekad se kaže diskretni) odnosno, da imaju konačan broj elemenata. Kod je preslikavanje niza simbola ulaznog alfabeta u niz kodnih simbola.

Definicija. Izvorni kod za slučajnu promjenljivu X je preslikavanje iz χ^* u D^* skup stringova sa konačnom dužinom za *D*-arni alfabet. Neka $C(x)$ predstavlja kodnu riječ koja odgovara x i neka $l(x)$ označava dužinu kodne sekvence kojom je kodirana riječ x.

Definicija. Kod se naziva **singularnim** ako se dva simbola ulaznog alfabeta preslikavaju u istu kodnu riječ. Ovakvi kodovi se ne mogu jednoznačno dekodirati (preslikati iz skupa kodnih simbola u polazni alfabet).

Primjer. Kod (kodno pravilo) dat tabelom je singularan

S	s_1	s_2	s_3	s_4
X	0	01	11	01

Primjer. Neka je D, $\mathfrak{D}=\{0,1\}$ i neka je $\chi=\{$crvena, plava, zelena, crna i ljubičasta$\}$. Neka je kod *C*(crvena)=1, *C*(plava)=11, *C*(zelena)=0, *C*(crna)=01, *C*(ljubičasta)=10. Pretpostavimo da imamo sekvencu simbola " zelena crvena crna ljubičasta " koja je kodirana sa 010110. Da li se ova sekvenca može jedinstveno dekodirati na prijemniku?

Prethodni kod predstavlja primjer nesingularnog koda koji nije jednoznačno dekodabilan. Znači nesingularnost je potreban ali ne i dovoljan uslov za jednoznačno dekodiranje.

Primjer. Neka je X slučajna promjenljiva sa sljedećom distribucijom i kodom $P(X=1)=1/2$, $C(1)=0$, $P(X=2)=1/4$, $C(2)=10$, $P(X=3)=1/8$, $C(3)=110$ i $P(X=4)=1/8$, $C(4)=111$. Entropija ove sekvence je 1.75. Toliko je jednaka i prosječna dužina kodne riječi. Ova se sekvenca može na jedinstven način dekodirati.

Primjer. $P(X=1)=1/2$ kodna riječ $C(1)=0$, $P(X=2)=1/4$ kodna riječ $C(2)=0$, $P(X=3)=1/8$ kodna riječ $C(3)=11$, $P(X=4)=1/8$ kodna riječ $C(4)=111$. Ne može se razlikovati kod $X=1$ i $X=2$.

Definicaja. Za kodnu riječ se kaže je nesingularna, ako se svaka riječ u X preslikava u različito $\mathfrak{D}^*$, tj. ako je $x_i \neq x_j$ da je $C(x_i) \neq C(x_j)$.

Definicija. Proširenje koda $C*$ koda C je preslikavanje stringa konačne dužine iz χ u string konačne dužine iz $\mathfrak{D}$, $C(x_1x_2...x_n)=C(x_1)C(x_2)...C(x_n)$ gdje je na desnoj strani izraza označeno nadovezivanje stringova.

Primjer. Ako je $C(x_1)=00$ i $C(x_2)=11$ tada je $C(x_1x_2)=0011$.

Definicija. Za kod se kaže da se može na jedinstveni način dekodirati, ako mu se na jedinstveni način može dekodirati proširenje $C*$.

Grubo rečeno, ako imamo string koji sadrži kodne riječi mi bi morali da znamo gdje koja kodna riječ počinje i završava. Recimo, sekvenca iz prvog primjera ne zadovoljava ovaj uslov. Na primjer 0110 može biti tumačena kao {crna, ljubičasta}, {zelena, plava, zelena}. Postoje i takvi kodovi koji se mogu na jedinstveni način dekodirati, ali da bi se izvršilo dekodiranje dekoder mora da gleda i unaprijed i unazad da bi odredio jedinstvenu sekvencu. U cilju pojednostavljenja hardvera dekodera treba ovakve situacije izbjegavati koliko je god to moguće.

Definicija. Kod se naziva **prefiksim** ili **trenutnim** ako nijedna kodna riječ nije prefiks nijedne druge kodne riječi. Ovakvi kodovi se mogu dekodirati bez "gledanja unazad".

Primjer. Naredna tabela prikazuje tri različita koda za slučajnu promjenljivu X:

X	**Kod 1**	**Kod 2**	**Kod3**
1	0	10	0
2	010	00	10
3	01	11	110
4	10	110	111

Pogledajmo primjer koda 2: Ako su prva dva bita 11 trebamo pogledati naredni bit i ako je 1 onda je to početak nove kodne riječi, a ako je 0 onda je to kraj tekuće kodne riječi ili početak kodne riječi 00. Suprotno tome kod 3 nema ovaj problem. Ovaj kod se naziva i **koma kod** jer 0 predstavlja kraj kodne riječi ili se do kraja kodne riječi stiže nakon tri bita. Šta se dešava kod koda 1?

Teorema. Trenutni kod definisan na alfabetu veličine D sa kodnim riječima $l_1, l_2, ..., l_m$ mora zadovoljavati:

$$\sum_{i=1}^{m} D^{-l_i} \leq 1$$

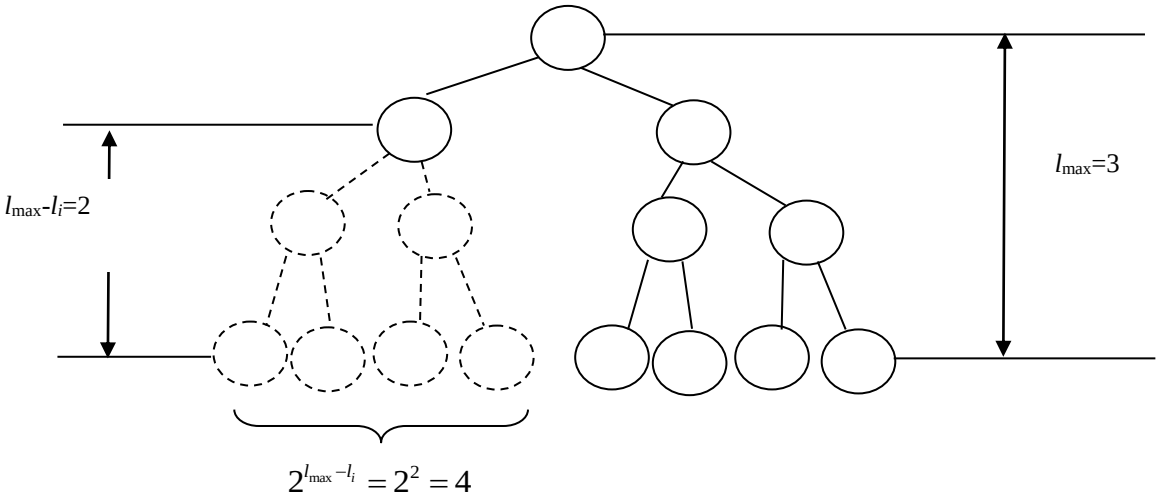
Dokaz. Pretpostavimo da je kod kreiran kao kodno stablo, gdje je put od korjena do listova kodna sekvenca i neka svaki list eliminiše prethodne čvorove kao moguće čvorove (to znači da prefiks posmatranom kodu ne može biti kodni simbol). Neka je $l_{\max}$ dužina najduže kodne riječi. Kodne riječi na nivou l_i ima $D^{l_{\max}-l_i}$ potomaka na nivou $l_{\max}$. Ovo znači da kodna riječ na nivou l_i eliminiše

$D^{l_{\max}-l_i}$ kodnih riječi na nižim nivoima. Ukupan broj listova na nivou $l_{\max}$ je $D^{l_{\max}}$. Sumirajući sve "eliminisane" kodne riječi dobijamo:

$$\sum_{i=1}^{m} D^{l_{\max}-l_i} \leq D^{l_{\max}}$$

Odavde slijedi da:

$$\sum_{i=1}^{m} D^{-l_i} \leq 1$$



3.4. Grayov kod

Napravimo malu digresiju u odnosu na ostatak priče i uvedimo prvi kod koji nema neke velike veze sa ostalim kodovima koji će biti uvedeni u ovom kursu, ali zbog njegove upotrebljivosti i mogućnosti da se kombinuje sa drugim kodovima biće ovdje definisan. To je Grayov kod, koji pripada klasi specijalnih kodova. Polazi od posmatranja nekog stanja reprezentovanog binarnim brojnim sistemom. Svaka promjena datog stanja treba da se reprezentuje sa jednom promjenom bita u binarnom stringu. Ovo je veoma pogodno za praćenje promjena koje se događaju kod kontinualnih varijabli, koje su diskretizovane i pretvorene u binarni zapis. Ne postoji jedinstvena strategija za realizaciju ovog koda (osim kod dužina kodne riječi 2 i 3) i ovdje ćemo demonstrirati jednu moguću metodu.

Ako sa a_i označimo bite polazne sekvence koja predstavlja binarni kod za dati broj. Najčešće se prvi bit u Grayovom kodu ostavi isti:

$$b_1=a_1$$

Zatim se svaki sljedeći kodira kao $b_i = a_i \oplus a_{i-1}$ gdje je $\oplus$ operator ekskluzivno ili. Tako se sada 7 i 8 kodiraju umjesto 0111 i 1000 kao 0100 i 1100, odnosno razlikuju se za samo jedan bit. 4-bitni kod je sada

Standardni	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010
Grayov	0000	0001	0011	0010	0110	0111	0101	0100	1100	1101	1111
Standardni	1011	1100	1101	1110	1111						
Grayov	1110	1010	1011	1001	1010						

Operacija ekskluzivno ili nad bitovima će se u narednim koracima često koristiti, pa ćemo je od sada na dalje često označavati prosto sa + vodeći računa da nije u pitanju ili već ekskluzivno ili operacija.

Dekodiranje Grayovog koda se provodi na veoma sličan način:

$$a_1=b_1$$

Za ostale simbole možemo da dodamo a_{i-1} lijevo i desnoj strani izraza $b_i = a_i \oplus a_{i-1}$. Dodavanje ćemo izvršiti pomoću operacije ekskluzivno ili koja je aditivna:

$$b_i \oplus a_{i-1} = a_i \oplus a_{i-1} \oplus a_{i-1}$$

Kako je $a_{i-1} \oplus a_{i-1} = 0$ i $a_i \oplus 0 = a_i$ (provjerite ove osobine, provjera se može izvršiti putem tablice istinitosti operacije) dobijamo:

$$a_i = b_i \oplus a_{i-1}$$

Dakle, operacija dekodiranja se obavlja kao:

$$a_1 = b_1$$
$$a_2 = b_2 \oplus a_1$$
$$a_3 = b_3 \oplus a_2$$
$$\ldots$$

3.5. Optimalni kodovi

Pod optimalnim kodom podrazumijevamo jednoznačno dekodabilan kod sa najmanjom prosječnom dužinom kodne riječi za dati alfabet. To se tretira kao sljedeći optimizacioni problem: pronaći kod sa dužinama kodne riječi l_1, l_2, .., l_m takav da je prosječna dužina kodne riječi:

$$L = \sum_i l_i p_i$$

što je moguće manja. Cilj je minimizacija prethodnog izraza pod uslovom da je zadovoljena Kraftova nejednakost:

$$\sum_{i=1}^{m} D^{-l_i} \leq 1$$

Da bi se podjednostavilo razmatranje pretpostavimo sledeće: (a) dužine kodnih riječi ne moraju biti cijeli brojevi (ovo je radi simplifikacije, a nije stvarna situacija); (b) Izraz na lijevoj strani Kraftove nejednakosti je jednak jedan (važi nejednakost). To dalje možemo da posmatramo kao Lagrange-ve množioce:

$$J = \sum_i p_i l + \lambda\left(\sum_i D^{-l_i} - 1\right)$$

Uzmimo izvode po dužinama koda i izjednačimo sa nulom:

$$\frac{\partial J}{\partial l_j} = p_j - \lambda D^{-l_j} \log D = 0$$

$$D^{-l_j} = \frac{p_j}{\lambda \log D}$$

Zamjenom izraza u ograničenje dobijamo:

$$\sum_i \frac{p_i}{\lambda \log D} = 1 \rightarrow \lambda = \frac{1}{\log D} \Rightarrow p_i = D^{-l_i}$$

Dalje se dobija da je optimalna dužina koda $l_i^* = -\log_D p_i$, pa je minimalna prosječna dužina kodne riječi:

$$L^* = \sum_i l_i^* p_i = H_D(X)$$

gdje je $H_D(X)$ entropija kod koje je uzet logaritam sa osnovom *D*. U praksi dva uvedena ograničenja ne važe pa smo na ovaj način identifikovali donju granicu.

Zadaci za vježbu

3.1. Dokaži teoremu 3.1.2.

Rješenje: 1. Uzmimo $-\frac{1}{n}\log_2$ od definicije tipičnog skupa:

$$-\frac{1}{n}\left(-n\left(H(X)+\varepsilon\right)\right) \le \frac{1}{n}\log p(x_1, x_2, ..., x_n) \le \frac{1}{n}\left(-n\left(H(X)-\varepsilon\right)\right)$$

2. Po definiciji tipičnog skupa, po asimptotskoj ekviparticionoj osobini i definiciji konvergencije po vjerovatnoćama slijedi:

$$\Pr((X_1, X_2, ..., X_n) \left| (X_1, X_2, ..., X_n) \in A_\varepsilon^{(n)}\right.) \rightarrow 1 \text{ kada } n \rightarrow \infty$$

Tada za bilo koje $\delta > 0$ postoji, takvo da za svako $n \ge n_0$ važi:

$$\Pr\left\{\left| -\frac{1}{n}\log p(X_1, ..., X_n) - H(X)\right| < \varepsilon\right\} > 1 - \delta$$

Postavimo $\delta = \varepsilon$ i dobili smo (2) teoreme.

3. Jasno je da važi:

$$1 \ge \sum_{x \in \chi^{(n)}} p(x) \ge \sum_{x \in A_\varepsilon^{(n)}} p(x) \ge \sum_{x \in A_\varepsilon^{(n)}} 2^{-n(H(X)+\varepsilon)} = 2^{-n(H(X)+\varepsilon)} \left| A_\varepsilon^{(n)} \right|$$

4. Za dovoljno veliko n, $\Pr\left\{A_\varepsilon^{(n)}\right\} > 1 - \varepsilon$ tako:

$$1 - \varepsilon \le \Pr\left\{A_\varepsilon^{(n)}\right\} \le \sum_{x \in A_\varepsilon^{(n)}} 2^{-n(H(X)-\varepsilon)} = 2^{-n(H(X)-\varepsilon)} \left| A_\varepsilon^{(n)} \right|$$

3.2. Dokažite teoremu 3.2.2.

Dokaz. Da bi dokazali predmetnu teoremu prvo dokažimo teoremu o "carskoj sredini", koja kaže ako $a_n \rightarrow a$ i $b_n = \sum_{i=1}^n a_i$ tada $b_n \rightarrow a$. Ideja je da ako je a_n blisko a to znači da srednja vrijednost prvih n brojeva u nizu teži takođe a stoga što su prvi elementi sve manje i manje važni u srednjoj vrijednosti velikog broja elemenata.

Dokaz. Pošto $a_n \rightarrow a$ to znači da za neko $\varepsilon > 0$ postoji $N(\varepsilon)$ tako da za $n \ge N(\varepsilon) \left| a_n - a \right| \le \varepsilon$. Ovo je definicija konvergencije. Dakle:

$$|b_n - a| = \left|\frac{1}{n}\sum_{i=1}^{n} a_i - a\right| \le \frac{1}{n}\left|\sum_{i=1}^{n} a_i - a\right|$$

Posljednja relacija slijedi na osnovu poznate nejednakosti trougla:

$$\frac{1}{n}\left|\sum_{i=1}^{n} a_i - a\right| \le \frac{1}{n}\left|\sum_{i=1}^{N_\varepsilon} a_i - a\right| + \frac{(n+N)}{n}\varepsilon \le \frac{1}{n}\left|\sum_{i=1}^{N_\varepsilon} a_i - a\right| + \varepsilon$$

Prvi član u ovom izrazu teži nuli ako n teži beskonačnosti. Naime, broj članova u sumi je ograničen, dok n može da raste. Dakle, iz ovog se može lako zaključiti da se razlika između b_n i a može učiniti proizvoljno malom. Sada se može dokazati i jednakost $H(X) = H'(X)$.

Dokaz. Po lančanom pravilu slijedi

$$\frac{H(X_1, X_2, ..., X_n)}{n} \le \frac{1}{n} H(X_n | X_1, X_2, ..., X_n)$$

Poznato je da entropije imaju granične vrijednosti. Po carskoj sredini srednja vrijednost ima istu tu graničnu vrijednost, a koja je u ovom slučaju jednaka granici kondicionalne entropije:

$$H(\chi) = \lim_{n\to\infty} \frac{H(X_1, X_2, ..., X_n)}{n} = \lim_{n\to\infty} H(X_n | X_1, X_2, ..., X_n) = H'(\chi)$$

3.3. Dokazati teoremu 3.3.1.

Rješenje: Očekivana dužina kodne riječi je:

$$\begin{aligned} E\left[l(X^n)\right] &= \sum_{x^n} p(x^n) l(x^n) = \sum_{x\in A_\varepsilon^n} p(x^n) l(x^n) + \sum_{x\in A_\varepsilon^n} p(x^n) l(x^n) \le \\ &\le \sum_{x\in A_\varepsilon^n} p(x^n)\left[n(H(X)+\varepsilon)+2\right] + \sum_{x\in A_\varepsilon^n} p(x^n)\left[n\log|\chi|+2\right] \\ &= \Pr(A_\varepsilon^{(n)})\left[n(H(X)+\varepsilon)+2\right] + \Pr(A_\varepsilon^{(n)})\left[n\log|\chi|+2\right] \le \\ &\le n(H(X)+\varepsilon) + 2 + \varepsilon n \log|\chi| = n(H(X)+\varepsilon') \end{aligned}$$

gdje je

$$\varepsilon' = \varepsilon + \varepsilon\log|\chi| + \frac{2}{n}$$

koje se može učiniti proizvoljno malim na osnovu izbora ε i izbora n.

3.4. Dokazati sledeću teoremu: Očekivana vrijednost dužine L za svaki trenutni D-arni kod za slučajnu promjenljivu X zadovoljava: $L \ge H_D(X)$.

Dokaz.

$$L - H_D(X) = \sum_i p_i l_i - \sum_i p_i \log_D \frac{1}{p_i} = -\sum_i p_i \log_D D^{-l_i} + \sum_i p_i \log_D p_i =$$

$$\sum_i p_i \log_D \frac{p_i}{D^{-l_i}} = \sum_i p_i \log_D \frac{p_i}{D^{-l_i}(\sum_j D^{-l_j})/(\sum_j D^{-l_j})}$$

$$r_i = D^{-l_i}/(\sum_j D^{-l_j}),\ c = \sum_i D^{-l_i}$$

$$L - H_D(x) = \sum_i p_i \log_D \frac{p_i}{r_i} - \log_D c = D(p\,||\,r) + \log_D \frac{1}{c} \ge 0$$

jer je relativna entropija nenegativna i $c \le 1$ po Kraftovoj nejednakosti.

3.5. Posmatran je fajl sa podacima na engleskom jeziku. Broj pojavljivanja određenih simbola je:

a	4186	b	832	c	1589	d	2045	e	6103	f	1050
g	947	h	2232	i	3785	j	134	k	456	l	2018
m	1223	n	3702	o	3697	p	1079	q	51	r	3124
s	3420	t	4450	u	1333	v	555	w	952	x	85
y	855	z	64								

Odrediti tipičnu sekvencu za predmetni model, ako se posmatra 100 karaktera teksta.

Rješenje: Procentualno pojedini karakteri se pojavljuju sa sledećim vjerovatnoćama:

a	8.38%	b	1.67%	c	3.18%	d	4.09%	e	12.21%	f	2.10%
g	1.90%	h	4.47%	i	7.57%	j	0.27%	k	0.91%	l	4.04%
m	2.45%	n	7.41%	o	7.40%	p	2.16%	q	0.10%	r	6.25%
s	6.84%	t	8.91%	u	2.67%	v	1.11%	w	1.91%	x	0.17%
y	1.71%	z	0.13%								

U tipičnoj sekvenci A se pojavljuje barem 8 puta, B jednom, C tri puta, D četiri puta, E dvanaest puta, F 2 puta, G jednom, H četiri puta, I sedam puta, L četiri puta, M dva puta, N sedam puta, O sedam puta, P dva puta, R šest puta, S šest puta, T osam puta, U dva puta, W jednom i Y jednom. To je ukupno 88 karaktera. Možemo dodati za još 12 slova po jedno pojavljivanje i odlučili smo se da to bude 12 onih kojima najmanje nedostaje do cijelog procenta:

a	8	b	2	c	3	d	4	e	12	f	2
g	2	h	5	i	8	j	0	k	1	l	4
m	3	n	8	o	7	p	2	q	0	r	6
s	7	t	9	u	3	v	1	w	2	x	0
y	2	z	0								

3.6. Posmatrajte model opisan prethodnim zadatkom. Pretpostaviti da je K najčešće pojavljivanih simbola kodirano sa $\log_2 K$ bita i prefiksom 1, dok je ostatak simbola kodiran sa ASCII kodom sa prefiksnim bitom 0. Koliko je bita potrebno za zapis originalne sekvence u ASCII kodu, kao i za zapis komprimovane sekvence, ako je $K = 2,4,8,16$. Ne treba voditi računa o mogućnosti dekodiranja predmetne poruke.

Rješenje: Dva najčešće pojavljivana simbola su **e** sa vjerovatnoćom 12.21% i **t** sa vjerovatnoćom 8.91%. Ako njih kodiramo sa 1 bitom i dodatnim jednim prefiksnim bitom 1 a ostalih 24 simbola kodiramo sa 5 bita i jednim prefiksnim dobijamo prosječnu dužinu kodne riječi dobijamo prosječnu dužinu kodne riječi:

$$2x0.2112+6x0.7888=5.1552\text{bita/simbolu}$$

Naredna dva najvjerovatnija simbola su: **a** sa 8.38% i **i** sa vjerovatnoćom 7.57%. Dakle četiri najvjerovatnija u tom slučaju kodiramo sa 2 bita i jednim za prefiks do ostalih 22 moramo kodirati sa 5 plus 1 za prefiks. Prosječna dužina kodne riječi je u ovom slučaju:

$$3x0.3807+6x0.6193=4.8579\text{bita/simbolu}$$

Naredna 4 najvjerovatnija simbola imaju vjerovatnoće: 7.41%, 7.40%, 6.84% i 6.25%. Tada ukupna vjerovatnoća 8 najvjerovatnijih simbola je: 56.97% i moraju se kodirati sa 3+1 bit dok preostalih 18 imaju vjerovatnoću 43.03% i kodiraju se sa 5+1 bit. Prosječna dužina kodne riječi je:

$$4x0.5697+6x0.4303=4.8606\text{bita/simbolu}$$

Konačno situacija sa pridavanjem narednih 8 simbola je trivijalna jer 16 simbola u glavnoj grupi moramo kodirati sa 4+1 bit baš kao i 12 preostalih simbola u grupi sa malim vjerovatnoćama. Kao što vidimo najbolji rezultat se postiže ako je odabrano kodiranje 4 najvjerovatnija simbola odvojeno od kodiranja preostalih 22 simbola. Vidimo da je ušteda simbolična ali postoji čak i kod ovog krajnje primitivnog načina kodiranja koji je upotrijebljen samo da ilustruje AEP.

3.7. Provjerite asimptotsku ekviparticionu osobinu za $n = 100$, $p = 0.2$, $p = 0.4$.

Rješenje: U skladu sa uputstvima koja su data prethodno uz eventualnu programsku realizaciju ponovite postupak koji je demonstriran u lekciji.

3.8. Pretpostaviti da je poznata entropija izvora $H(X)$. Neka je nad alfabetom izvora primjenjena linearna transformacija i neka je na osnovu kodnog simbola x dobijen kodni simbol $y=ax+b$ gdje su a i b konstante. Odrediti entropiju izvora y u zavisnosti od entropije izvora x.

Rješenje: Entropija se definiše kao:

$$H(Y) = -\sum_{y} p(y) \log p(y)$$

Međutim, kako je predmetno preslikavanje 1:1 to znači da jednoj vrijednosti x odgovara jedna vrijednost y tako da je jasno da je $H(Y)=H(X)$.

3.9. Razmotriti jednoznačnu dekodabilnost kodova kao i njihovu optimalnost ako su vjerovatnoća pojavljivanja pojedinih simbola $P(s_1)=1/2$, $P(s_2)=1/4$, $P(s_3)=1/8$, $P(s_4)=1/8$.

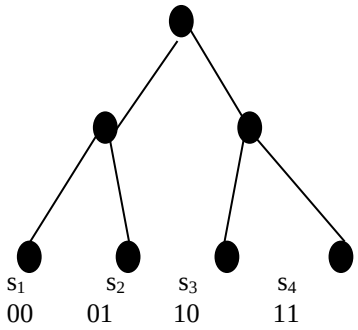
S	(a)	(b)	(c)
s_1	00	0	0
s_2	01	10	01
s_3	10	110	011
s_4	11	1110	0111

Prvi kod je jednoznačno dekodabilan ali nije optimalan, drugi kod je jednoznačno dekodabilan (koma kod) ali nije optimalan. Treći kod je takođe jednoznačno dekodabilan i ponovo nije optimalan. Ujedno ovaj kod nije trenutan jer moramo da gledamo u "budućnost" da bi se izvršilo dekodiranje. Optimalni kod se može dobiti na jednostavan način iz drugog koda tako što se eliminiše posljednja cifra iz koda za s_4. Napominjemo da je mnogo pogodnije da koristimo kod (b) jer je on trenutan odnosno dekodiranje simbola se obavlja na osnovu tekućeg i prethodnih simbola bez gledanja u budućnost.

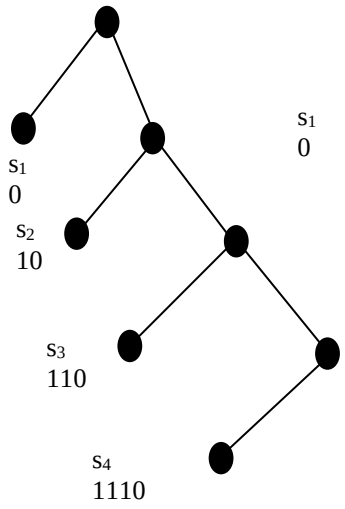
Poglavlje IV
HUFFMANOV, LZW I ARITMETIČKI KOD

4.1. Vizuelizacija kodova preko kodnog stabla

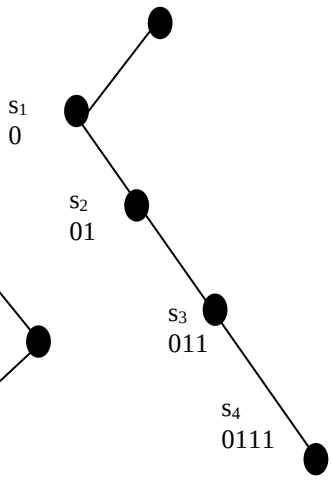
Kodovi koji se koriste za kodiranje izvora često se vizuelizuju preko drveta (podsjetite se ove strukture podataka iz programiranja). U korjenu drveta se ne upisuje nijedan simbol. Lijevi sinovima se dodjeljuje kodni simbol 0, dok se desnim dodjeljuje kodni simbol 1. Treba napomenuti da samo diskretan skup čvorova drveta predstavlja moguće kodne riječi. Po pravilu, svi listovi su kodne riječi. Naredna tri koda su redom, kod sa konstantom dužinom od 2 bita, gdje svi mogući simboli su kodne riječi (potpuno binarno drvo visina 2), zatim je prikazan kod, kod kojeg je posljednji simbol kraj kodne riječi. Oba ova koda su trenutni jer se njihovo jednoznačno dekodiranje može izvršiti odmah nakon prijema posljednjeg bita iz poruke. To se ne može reći za posljednji kod, koji, iako je jednoznačno dekodabilan, mora da pogleda naredne bitove da bi izvršio dekodiranje. Dakle, ovaj kod nije trenutan, iako je jednoznačno dekodabilan. Pomoću kodnog stabla mogu se vizuelizovati i *D*-arni kodovi, kako je to urađeno na slici dolje. Kao što ćemo kasnije vidjeti, kodno stablo ima značajnu ulogu u kodiranju i dekodiranju kod Huffmanove šeme.



Jednoznačno dekodabilan trenutan kod sa riječima iste dužine.

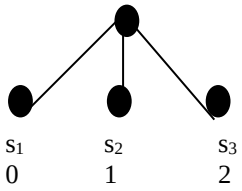


Jednoznačno dekodabilan trenutan kod sa riječima različite dužine, u pitanju je koma kod a dekodabilnost koda neće biti narušena ako se simbol s_4 "skrati" za jedan bit.

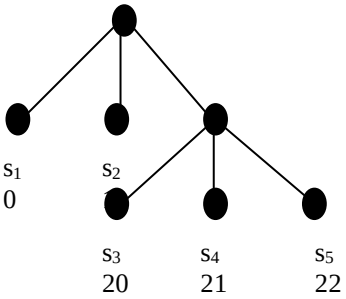


Jednoznačno dekodabilan ali ne i trenutan kod.

D-arni kod:



Ternarni kod.



Ternarni kod sa kodnim riječima različite dužine.

4.2. Kodiranje izvora - nastavak

Na ovaj način smo odredili donju granicu za prosječnu dužinu koda. Može se pokazati da je gornja granica za trenutni kod koji se može fizički realizovati $H(X)+1$. Dužina kodne riječi se može zapisati kao:

$$l_i = \lceil \log_D (1/p_i) \rceil$$

gdje je $\lceil x \rceil$ najmanji cijeli broj veći od x. Ove kodne riječi zadovoljavaju Kraftovu nejednakost:

$$\sum_i D^{-l_i} = \sum_i D^{-\left\lceil \log \frac{1}{p_i} \right\rceil} \le \sum_i D^{-\log \frac{1}{p_i}} = \sum_i p_i = 1$$

Trenutni kod koji daje najmanju prosječnu dužinu kodne riječi naziva se se **kompaktnim** ili **optimalnim**. Kodne dužine zadovoljavaju:

$$\log_D \frac{1}{p_i} \le l_i < \log_D \frac{1}{p_i} + 1$$

uzimajući operator očekivane vrijednosti dobijamo:

$$H_D(X) \le L < H_D(X) + 1$$

Sledeći problem je smanjiti prekoračenje u odnosu na $H_D(X)$ distribuirajući pretek entropsije preko više simbola. Pretpostavimo da šaljemo sekvencu dužine n simbola. Njena dužina je:

$$L_n = \sum_{\mathbf{x} \in \chi^n} p(x_1, ..., x_n) l(x_1, ..., x_n)$$

Provodeći dokaze kao u prethodnom slučaju slijedi:

$$H(X_1, ..., X_n) \le L_n \le H(X_1, ..., X_n) + 1$$

Ako su simboli međusobno nezavisni i sa istom raspodjelom važi $H(X_1, ..., X_n) = nH(X)$ pa odavde slijedi:

$$H(X) \le L \le H(X) + \frac{1}{n}$$

gdje je L prosječna dužina potrebna za kodiranje jednog simbola. Nažalost, simboli u porukama su rijetko kada međusobno nezavisni, ali se i dalje može pisati:

$$\frac{H(X_1, ..., X_n)}{n} \le L \le \frac{H(X_1, ..., X_n)}{n} + \frac{1}{n}$$

Iako smo do sada izvodili operacije nad trenutnim kodovima, Kraftova nejednakost i njene posljedice važe kod svih kodova koji se na jedinstven način mogu dekodirati, a ne samo kod trenutnih kodova.

Ilustrujmo na jednom primjeru činjenicu da sa proširivanjem alfabeta nad kojim se vrši kodiranje smo u stanju da ostvarimo bolje rezultate. Neka je dat sistem sa dva kodna simbola s_1 i s_2 koji se pojavljuju sa vjerovatnoćama 7/8 i 1/8. Simbole možemo kodirati sa 0 i 1 respektivno i prosječna dužina kodne riječi je 1, iako je entropija ovog sistema: 0.5436. Pod pretpostavkom da su simboli izvora međusobno nezavisni (izvor bez memorije) i da je izvršeno kodiranje dva bita ulazne poruke kao:

S^2 P_i X_i

s_1s_1	49/64	0
s_1s_2	7/64	10
s_2s_1	7/64	110
s_2s_2	1/64	111

Entropija S^2 skupa je 1.0871, dok je prosječna dužina kodne riječi L_2=1.3594bita/simbolu skupa S^2, odnosno po simbolu iz skupa S to je $L_2/2$=0.68bit/simbolu što je znatno bolje nego 1bit/simbolu. Ako bi se n povećavalo, dobili bi smo sve bolje rezultate, po cijenu složenosti koda.

Posmatrajmo još jedno proširenje ovog koda sa tri simbola:

S^3	P_i	X_i
$s_1s_1s_1$	343/512	1
$s_1s_1s_2$	49/512	011
$s_1s_2s_1$	49/512	010
$s_2s_1s_1$	49/512	001
$s_1s_2s_2$	7/512	00011
$s_2s_1s_2$	7/512	00010
$s_2s_2s_1$	7/512	00001
$s_2s_2s_2$	1/512	00000

U ovom slučaju prosječna dužina kodne riječi je:

$$1\cdot\frac{343}{512}+3\cdot\frac{147}{512}+5\cdot\frac{22}{512}=\frac{343+441+110}{512}=\frac{894}{512}=1.746$$

Dužina kodne riječi po jednom simbolu je 1.746/3=0.582bita/simbolu. Dakle, značajno smo se primakli entropiji sistema.

4.3. Šenon-Fanov postupak kodiranja

Osnovna ideja, kako doći do optimalnog (kompaktnog) koda za poznate vjerovatnoće greške pojedinih simbola, je uzeti dužinu kodne riječi koja je proporcionalna sa $-\log(p_i)$ gdje je p_i vjerovatnoća simbola. Kako se vjerovatnoće rijetko mogu pisati u formi D^k, gdje je k negativni cijeli broj, a D osnova logaritma, odnosno broj simbola sa kojim se vrši kodiranje izvora, ovo se može uraditi izuzetno rijetko. Dva primjera za ovakav tip kodiranja za binarni alfabet su:

	Kod	**(a)**		**Kod**	**(b)**	
S	P_i	$-\log_2(P_i)$	X_i	P_i	$-\log_2(P_i)$	X_i
s_1	¼	2	00	½	1	0
s_2	¼	2	01	¼	2	10
s_3	¼	2	10	1/8	3	110
s_4	¼	2	11	1/8	3	111

Prosječna dužina kodne riječi u oba slučaja dostiže optimalnu vrijednost odnosno entropiju izvora:

$H_a(S)=\log_2 4=2$ $L_a=4\times 0.25\times 2=2$ $H_b(S)=1.75$ $L_b=1.75$

Ideja koja se prirodno nameće je da se za necjelobrojno D^k usvajaju prve veće cijelobrojne vrijednosti. U nekim slučajevima to će dati kompaktan kod, ali u mnogim neće. Na primjer kod (a) koji bi bio kreiran na način koji je prethodno opisan za dati kod nije kompaktan, dok kod (b) jeste.

S	P_i	$-\log(P_i)$	$l_i=\lceil\log(1/P_i)+1\rceil$	(a)	(b)
s_1	2/3	0.58	1	0	0
s_2	2/9	2.17	3	100	10
s_3	1/9	3.17	4	1010	11

Uočite da su oba koda trenutni, ali da je prosječna dužina kodne riječi koda (b) 1.33 odnosno, znatno manja od prosječne dužine drugog koda. Postavlja se pitanje, kako odrediti optimalni kod pod datim pretpostavkama (poznate vjerovatnoće grešaka pojavljivanja pojedinih simbola). Postoje dva pristupa. Prvi je Šeno-Fanoovo kodiranje koje nije ni jednoznačno ni u pojedinim situacijama optimalno. Ovdje ga dajemo više iz istorijskih razloga. Ideja je prosta. Sve kodne riječi treba podijeliti u dvije grupe tako da ukupna vjerovatnoća obije grupe bude približno 0.5. Jednoj grupi se dodjeljuje kodni simbol 0, a drugoj 1. Zatim se obije grupe dijele u dvije grupe, tako da podgrupe imaju približno jednake vjerovatnoće. Postupak se ponavlja rekurzivno dok u svakoj od podgrupa ne ostane jedan simbol. Ilustrujmo to na jednom primjeru.

S	P_i	I podjela	II podjela	III podjela
s_1	0.5	0	0	0

s_2	0.25	1	10	10	
s_3	0.125	1	11	110	
s_4	0.125	1	11	111	

U ovom slučaju je dobijen kompaktni kod i prosječna dužina kodne riječi jednaka entropiji. Pogledajmo sledeća tri primjera:

S	P_i	I podjela	II podjela	III podjela	IV podjela
s_1	0.6	0	0	0	0
s_2	0.2	1	10	10	10
s_3	0.1	1	11	110	110
s_4	0.07	1	11	111	1110
s_5	0.03	1	11	111	1111

S	P_i	I podjela	II podjela	III podjela
s_1	0.3	0	00	00
s_2	0.2	0	01	01
s_3	0.2	1	10	10
s_4	0.2	1	11	110
s_5	0.1	1	11	111

S	P_i	I podjela	II podjela	III podjela	IV podjela
s_1	0.4	0	00	00	00
s_2	0.2	0	01	01	01
s_3	0.12	1	10	100	100
s_4	0.08	1	10	101	101
s_5	0.08	1	11	110	110
s_6	0.08	1	11	111	1110
s_7	0.04	1	11	111	1111

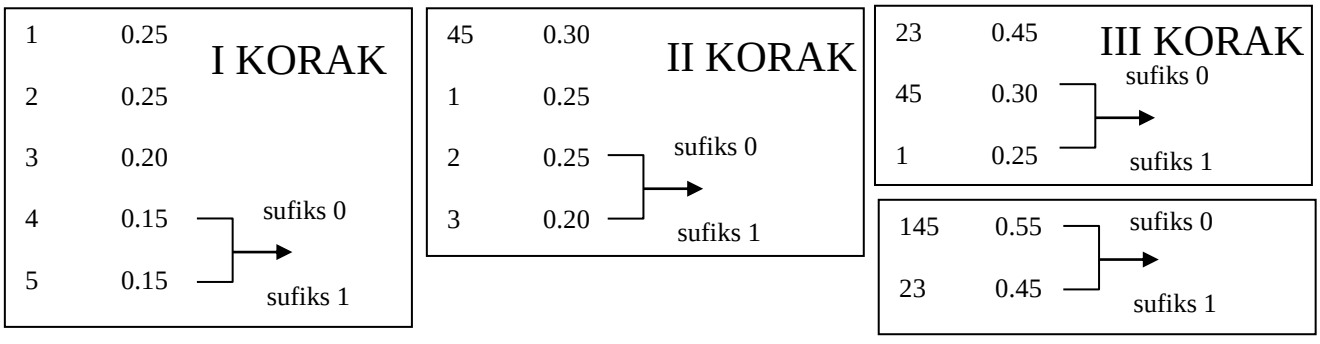
Posljednji kod nije najbolji, koji bi se mogao konstruisati (nije kompaktan). Pošto Šenon-Fanoov postupak ima niz proizvoljnih tumačenja, to se traženje najboljeg koda često provodi između nekoliko "dobrih". Suprotno ovom postupku Huffmanov postupak daje optimalan kod.

4.4. Huffmanov kod

Huffmanov (Hafmen) kod je optimalni kod za datu distribuciju kodnih simbola.

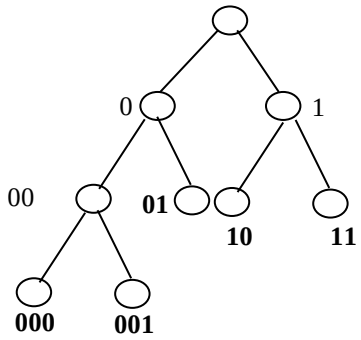
Primjer. Posmatrajmo distribuciju X koja uzima vrijednosti u skupu $\chi = \{1,2,3,4,5\}$ sa vjerovatnoćama 0.25, 0.25, 0.2, 0.15 i 0.15 respektivno.

U svakom koraku kombinujemo dva najmanje vjerovatna simbola u jedan simbol. U prvom koraku 0.15 i 0.15 su kombinovani u simbol sa 0.30 i nova sekvenca je 0.30, 0.25, 0.25 i 0.2. Sada se kombinuju dvije najmanje vjerovatnoće koje daju 0.45 (0.25 i 0.2) i sekvenca je 0.45, 0.3, 0.25. Treći korak je 0.3 i 0.25 koji daju 0.55 da sekvencom 0.55, 0.45. Konačno saberemo ove dvije vjerovatnoće i dobijamo 1. Sada na drvetu dodjelimo kodove. Prosječna dužina koda je 2.3. Ovo je tek nešto više od entropije skupa koja iznosi 2.285bita.



Kodiranje 1 01; 2 10; 3 11; 4 000; 5 001

Dokaz optimalnosti Huffmanovog koda se može postići na osnovu sledeće jednostavne leme.



Boldirano su prikazani listovi koji predstavljaju kodne riječi.

Lema. Za svaku raspodjelu postoji optimalni trenutni kod sa najmanjom prosječnom dužinom koji zadovoljava sledeće osobine:

1. Ako je $p_j > p_k$ onda $l_j \leq l_k$.
2. Dvije najduže kodne riječi imaju istu dužinu.
3. Ove dvije riječi odgovaraju najmanje vjerovatnim simbolima i razlikuju se samo u posljednjem bitu.

Dokaz leme: 1. Ako je suprotno odnosno ako je $p_j > p_k$ može da znači $l_j > l_k$ onda kod nije optimalan jer se prostom zamjenom kodnih riječi za simbole sa vjerovatnoćama p_j i p_k dobija kraći kod.
2. Ako ovo nije zadovoljeno odnosno ako je jedna kodna riječ duža za 1 bit onda bi najdužu kodnu riječ mogli da skratimo za jedan bit jer skraćivanjem za jedan bit dobijamo simbol koji je i dalje list pošto ako je roditelj najduže kodne riječi roditelj još nekom simbolu a onda dolazimo do suprotnosti da je naša kodna riječ najduža. Dakle ako je optimalni trenutni kod moguće na jednom simbolu dalje skratiti za jedan bit dolazimo do koda koji je "optimalniji" što je u suprotnosti sa postavkom leme. 3. Isto kao prethodno najduže koda riječ mora da ima barem jednu partnersku kodnu riječ sa kojom dijeli roditelja inače će ponovo biti moguće skratiti najdužu riječ za jedan odnosno pronaći optimalniji kod od optimalnog.

Ako pogledate postupak dobijanja Huffmanovog koda očigledno je da taj kod ispunjava uslove koji su navedeni u lemi.

Modifikacija Huffmanovog koda je moguća i za slučaj nebinarnih alfabeta. Pravilno se procedura u ovom slučaju provodi na sledeći način. Treba prvo odrediti koliki broj kodnih simbola sa najmanjom vjerovatnoćom treba kodirati u prvom koraku. To se provodi tako što se od ukupnog broja kodnih poruka oduzima po D-1 u svakom koraku (ovo je ekvivalentno tome što se u Huffmanovom kodu razmjenjuje D kodnih poruka sa jednom). Broj kodnih poruka koji ostane u proceduri a manji je ili jednak D sa najmanjim vjerovatnoća,a se kodira u prvom koraku.

Dakle da sublimiramo. Pretpostavimo da imamo N simbola u alfabetu kojega želimo da kodiramo sa D simbola izlaznog alfabeta. Algoritam za određivanje broja simbola u početnom koraku se može opisati sledećim pseudokodom:

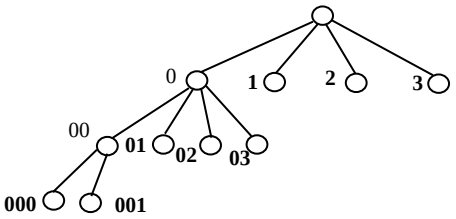
```
N'=N
WHILE N'>D
      N'=N'-(D-1)
ENDWHILE
```

Dobijeno N' je broj simbola koji se kodiraju u početnom koraku.

Primjer: N=8, D=4. N'=8, u sljedećem koraku umanjujemo za D-1 na 5 a u narednom koraku na 2. Pošto je 2 manje ili jednako od D dobijamo da u prvom koraku možemo uzeti dva simbola da kodiramo. Pogledajte primjer dolje.

Kodne poruke	Kod	Vjero.	I korak	Vj.I	II kor.	Vj.II	III.Kor	Ukupno
s_1	1	0.22	1	0.22	1	0.22	1	1.00
s_2	2	0.20	2	0.20	2	0.20	2	
s_3	3	0.18	3	0.18	3	0.18	3	
s_4	00	0.15	00	0.15	00	0.40	0	
s_5	01	0.10	01	0.10	01			
s_6	02	0.08	02	0.08	02			
s_7	030	0.05	030	0.07	03			
s_8	031	0.02	031					

Odredite prosječnu dužinu kodne riječi u ovom slučaju.



Studenti često pitaju što ako imamo dvije kodne riječi sa istom vjerovatnoćom. U tom slučaju za bilo koju kodnu riječ se odlučili dobijeni kod će biti kompaktan. Postoje neki praktični razlozi zbog kojih se proizvoljnost u ovom slučaju redukuje, ako iste vjerovatnoće imaju u nekoj od iteracija algoritma kodni simboli koji su već iskodirani sa određenim brojem bita. Tada se visočije u hijerarhiji kodnih riječi postavljaju one, koje imaju veću dužinu već iskodiranih kodnih simbola (kasnije ulaze u postupak grupisanja i kodiranja). Ovo ne utiče na optimalnost (kompaktnost) koda već samo na varijansu dužine kodne riječi koja igra ulogu u nekim specifičnim situacijama.

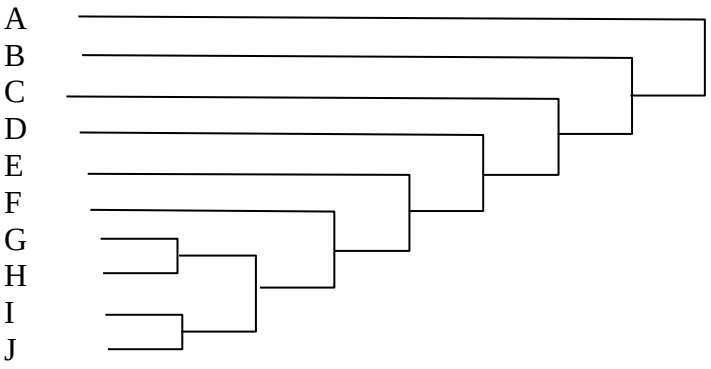
Napomenimo da je kompjutersko programiranje Huffmanovog koda rekurzivno, zasnovano na šemi drveta i kao takvo veoma jednostavno. Dekodiranje može sa programerske tačke gledišta predstavljati veći problem.

Često se u praksi Huffmanov kod ne primjenjuje za sve kodne riječi već za određeni broj kodnih riječi koje se relativno često pojavljuju, dok se one kodne riječi koje se pojavljuju ekstremno rijetko kodiraju kodom fiksne dužine. Ovo stoga da se pojava nekih kodnih riječi koje su inače ekstremno rijetke, ne bi odrazila i pokvarila kvalitet kompresije aktuelne poruke.

Posmatrajmo sledeći primjer. Kod ima 10 simbola od kojih se posljednja 4 javljaju sa jako malim vjerovatnoćama:

A	0.4	B	0.25	C	0.15	D	0.10	E	0.05	F	0.04
G	0.004	H	0.003	I	0.002	J	0.001				

Ovaj kod možemo kodirati standardnom Huffmanovom tehnikom:



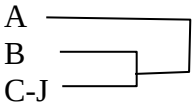
Ako postavimo da je u svakom koraku gornja grana kodirana sa simbolom 1 a donja grana sa simbolom 0 dobijamo sljedeće kodne riječi:

A	1	B	01
C	001	D	0001
E	00001	F	000001
G	00000011	H	00000010
I	00000001	J	00000000

Prosječna dužina kodne riječi je:

$$L = 1\cdot 0.40 + 2\cdot 0.25 + 3\cdot 0.15 + 4\cdot 0.10 + 5\cdot 0.05 + 6\cdot 0.04 + 8\cdot 0.01 =$$
$$= 2.32\text{bita/simbolu}$$

U pitanju je kompaktni kod sa manom da postoje veoma dugačke kodne riječi sa po 8 bita. Uzmimo sada da simbole C-J kodiramo uniformnom dužinom kodne riječi. Ima ih ukupno 8 i nakon prefiksa kojega ćemo odrediti kodiraćemo ostatak sa 3 bita. Ovaj generički simbol ima vjerovatnoću 0.35:



Po istom principu kao ranije A se kodira kao 1, B kao 01, dok se generički simboli kodiraju sa prefiksom 00:

C	00000
D	00001
E	00010
F	00011
G	00100
H	00101
I	00110
J	00111

Prosječna dužina kodne riječi u ovom slučaju je

$$L = 1\cdot 0.40 + 2\cdot 0.25 + 5\cdot 0.35 = 2.65\text{bita/simbolu}$$

Dakle ostvarili smo gubitak od 0.33bita/simbolu ali očekujemo da se ovo barem djelimično isplati kroz povećanje sposobnosti ispravljanja pogreški u kodu nakon prenosa kodne riječi komunikacionim kanalom.

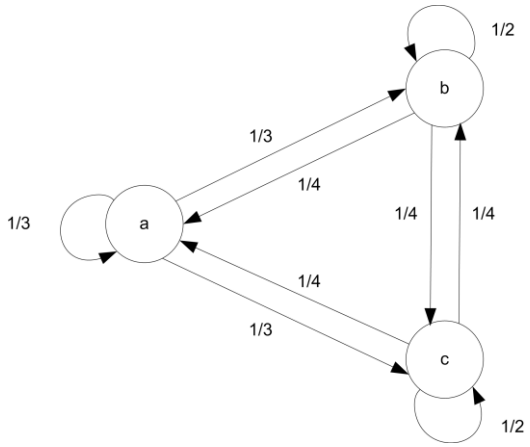
Huffmanov kod radi dobro kod stacionarnih izvora i kod poznatih vjerovatnoća pojedinih simbola. Međutim, kod nestacionarnih izvora ovo nije slučaj. Tu se može koristiti adaptivni Huffmanov kod koji mjeri vjerovatnoću pojavljivanja simbola i kada usvojeni Huffmanov kod prestane da daje optimalnu dužinu koda vrši se ponovno izračuvanja (procjenu vjerovatnoća) kako bi formirali novi Huffmanovog kod koji bolje korispodnira poruci koju prenosi.

Huffmanovo kodiranje se može koristiti i kod Markovljevih procesa sa memorijom. Posmatrajmo primjer koji je već više puta korišćen:

$p(a|a)=1/3$ $p(b|a)=1/3$ $p(c|a)=1/3$ $p(a|b)=1/4$ $p(b|b)=1/2$ $p(c|b)=1/4$

$p(a|c)=1/4$ $p(b|c)=1/4$ $p(c|c)=1/2$

Graf tranzicije je:



Podsjetimo se da se u stacionarnom stanju dobija $p_a=3/11$ i $p_b=p_c=4/11$. Ako bi izvršilo Huffmanovo kodiranje u stacionarnom stanju simbol *b* (ili *c*) bi kodirali sa jednim bitom, dok bi simbol *a* i *c* (ili *b*) kodirali sa dva bita pa bi prosječna dužina kodne riječi bila 18/11=1.6364. Međutim, mi možemo kodirati prelaske pošto kod trenutnih kodova znamo prethodni simbol. Kod prelaska iz *a*, opet dva simbola imaju dužinu 2, a jedan 1 i pošto su svi jednakovjerovatni to je prosječna dužina kodne riječi: $L_a=5/3$. Na sličan način za ostale prelaske slijedi: $L_b=3/2$ i $L_c=3/2$. Ukupna prosječna dužina kodne riječi je:

$$L = p_a L_a + p_b L_b + p_c L_c = 17/11 = 1.5455$$

čime smo na neki način pokazali da Markovljev proces ima manju entropiju od procesa bez memorije i da se može bolje komprimovati.

Sada će biti dato detaljno funkcionisanje ovog koda. Tri kodna simbola se pojavljuju sa vjerovatnoćama:

simbol	vjerovatnoća	kod
a	3/11	00
b	4/11	01
c	4/11	1

Ako je prethodno bio simbol *a* vjerovatnoća pojavljivanja narednog simbola je:

simbol	vjerovatnoća	kod
a|*a*	1/3	00
b|*a*	1/3	01
c|*a*	1/3	1

Ako je prethodno bio simbol *b* dobijamo sljedeću tabelu (kod je jedna moguća realizacija):

simbol	vjerovatnoća	kod
a|*b*	¼	00
b|*b*	½	1
c|*b*	¼	01

Ako je prethodno bio simbol *c* kodiranje se može obaviti sledećom tabelom:

simbol	vjerovatnoća	kod
a|*c*	¼	00
b|*c*	¼	01
c|*c*	½	1

Ako je poslata sekvenca
aabcacabca
prvim kodom će biti kodirana svaki simbol pojedinačno:
00 00 01 1 00 1 00 01 1 00
Dekodiranje se obavlja na osnovu istog kodnog stabla koje je bilo u prethodnom slučaju.

Međutim ako koristimo matrice uslovnih vjerovatnoća onda prvi simbol kodiramo po početnoj tabeli:
00
Naredni simbol pošto je poslato *a* kodiramo po tabeli uslovnih vjerovatnoća kada je poslato *a*
00
Trećem simbolu ponovo primjenjujemo istu tabelu
01
Narednom simbolu primjenjujemo tabelu za uslovno *b*
01
Naredni kod je za uslovno *c*
00
pa za uslovno *a*
1
ponovo za uslovno *c*
00
b|*a* se kodira kao
01
c|*b* je
01
i konačno
a|*c* je
00

Ukupna sekvenca u slučaju kodiranja drugom tehnikom je:
00 00 01 01 00 1 00 01 01 00

Dekodiranje (ovdje ćemo demonstrirati samo drugi slučaj) se obavlja jednostavno. Za prvi simbol se može koristiti prva tabela i zaključiti da je 00->*a* za sledeći simbol gleda se tabela u zavisnosti od *a*. U toj tabeli 00 ponovo predstavlja simbol *a*. Naredni simbol se dekodira ponovo na osnovu tabele u zavisnosti od *a*. Poruci 01 odgovara simbol *b*. Nakon toga koristimo tabelu u zavisnosti od *b* gdje 01 odgovara simbolu *c*. Nastavite sami sa dekodiranjem poruke.

Prvim kodom dobili smo da je sekvenca kodirana sa 17 bita dok drugim kodom imamo sekvencu sa 19 bita. Zbog čega je "optimalniji" kod u ovom slučaju radio lošije?

Realizacija Huffmanovog koda se može naći na Internetu. Između mnoštva realizacija vrijedi istaći da je Huffmanovo kodiranje veoma pogodno za strukturu stabla (nadam se da ovo nije potrebno objašnjavati) ali je ujedno i veoma pogodno za aplikaciju objektno orijentisanih metoda. Naime, kodno stablo se može realizovati kao struktura drveta sa dva pokazivača koji pokazuju na lijevog i na desnog sina. Međutim, pošto se naše stablo mora kreirati unazad od čvorova ka korjenu nije zgoreg imati još jedan pokazivač koji pamti roditelja. Pošto se dekodiranje obavlja od glave ka

listovima zgodno je da svaki čvor pamti i glavu kao statički podatak član zajednički za sve čvorove u sistemu. Dobra strana objektno orjentisanog pristupa je i činjenica da se kreiranje novih čvorova može obaviti u konstruktoru te da se uništavanje čvorova može obaviti u destruktoru. Ovim se ne opterećuje ostatak koda provjerama postojanja čvora, odnosno uništavanjem čvorova i njihovim kreiranjem.

Pored korišćenja objektno orjentisanog programiranja i MATLAB posjeduje već gotove funkcije za Huffmanovo kodiranje i dekodiranje. Posmatrajmo sledeći primjer:

```
symbols = [1:6];
p = [.5 .125 .125 .125 .0625 .0625];
```

Promjenljiva symbols predstavlja pojedine simbole koda (u ovom slučaju 1, 2, 3, 4, 5, 6) dok su u vektoru p postavljene vjerovatnoće pojedinih simbola. Naredba huffmandict formira kodno stablo i rječnik podataka

```
[dict,avglen] = huffmandict(symbols,p);
```

Rječnik je smješten u promjenljivu dict (u pitanju je matrica sa ćelijama). Npr. `dict{3,2}` daje kodnu riječ za treći simbol. Prosječna dužina kodne riječi se nalazi u promjenljivoj `avglen` i za dati primjer iznosi 2.125. Inače naredba `huffmandict` može imati treći argument koji predstavlja broj simbola alfabeta (kod može biti nebinaran) dok četvrti argument može biti `'min'` odnosno `'max'` što determiniše ponašanje koda kod riječi sa istim vjerovatnoćama (`'min'` uvijek bira riječ tako da minimizuje varijansu dužina riječi dok `'max'` maksimizuje).

Naredbom

```
actualsig = randsrc(1,100,[symbols; p]);
```

generiše se niz od 100 simbola ulaznog alfabeta sa datim karakteristikama. Naredba `huffmanenco` vrši kodiranje niza sa datim rječnikom

```
comp = huffmanenco(actualsig,dict);
```

dok naredba `huffmandeco` vrši dekodiranje.

```
dsig = huffmandeco(comp,dict);
```

Provjera uspješnosti se obavlja poređenjem dekodirane poruke sa originalnom

```
isequal(actualsig,dsig)
```

i ako je rezultat 1 (u predmetnom slučaju nema razloga da tako ne bude) dobili smo dekodiranu poruku koja je identična originalnoj.

4.5. RLE i diferencijalni kod

RLE (Run Length Encodding) je veoma jednostavna, ali u brojnim praktičnim aplikacijama izuzetno korisna kodna šema. Koristi se kod podataka koji se pojavljuju veliki broj puta uzastopno. Npr. fax dokument se dijeli u tačkice, koje nazivamo pikseli. Svaka od tačaka je jedno crno ili bijelo polje. Ako se crno polje ponavlja veliki broj puta uzastopno uputno je zapamtiti samo da je polje crno i broj piksela koji se ponavljaju. Na sličan način se kodira i bijelo polje. Ovakav kod se zatim može primjeniti i kod slika sa velikim uniformnim pozadinama, kao i kod video sekvenci gdje se

osvjetaljaj djelova slike u susjednim frejmovima ne mijenja. RLE kodiranje se rijetko sprovodi samostalno, a često u kombinaciji sa nekim drugim kodom kao što je npr. Huffmanov kod.

Primjer RLE kodiranja. Posmatrajmo sekvencu:

15	15	15	17	17	17	17	17	21	21	21	21	21
21	21	6	23	23	34	34	34	34	11	11	11	11

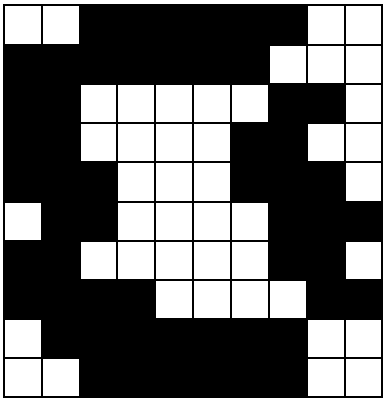
Možemo je kodirati sa parovima:
(15,3) (17,5) (21,7) (6,1) (23,2) (34,4) (11,4)

Dakle polazna sekvenca ima 26 simbola a kodirana sekvenca ima 14 simbola čime je ostvarane kompresija na 14/26=53.8% originalne informacije. Pogledajmo sada drugi primjer:

15	16	15	17	17	17	16	17	21	21	21	22	21
21	21	6	23	23	34	33	34	34	11	12	11	11

Sekvenca je samo malo izmjenjena u odnosu na polaznu a kodira se kao:
(15,1) (16,1) (15,2) (17,3) (16,1) (17,1) (21,3) (22,1) (21,3) (6,1) (23,2) (34,1) (33,1) (34,2) (11,1) (12,1) (11,2)

Dobili smo sekvencu sa 34 simbola odnosno ostvarili smo tzv. negativnu kompresiju. Ovo ukazuje na činjenicu da se RLE kod koristi samo kod specifičnih signala ili u kombinaciji sa drugim kodovima. Jedan takav tip signala su faksovi kod kojih postoje samo pikseli koji su crni ili bijeli (jedan ili nula). Posmatrajmo sledeću sliku.



Originalni kod zahtjeva 100 bita za kodiranje ove površi dimenzija 10x10. Pošto ovdje nema drugih simbola osim jedinica i nula mi ćemo kodiranje obaviti tako što ćemo kodirati prvi simbol i zatim samo pisati koliko puta se taj simbol pojavi zaredom (nastavljajući na naredni red):
1 2 6 2 7 3 2 5 2 1 2 4 2 2 3 3 3 2 2 4 5 5 2 1 4 4 2 1 7 4 6 2.
Kao što vidimo imamo 32 simbola umjesto 100 bita i na prvi pogled smo napravili značajnu uštedu. Međutim sada umjesto bita imamo nebinarne simbole. Pretpostavimo da prvi simbol kodiramo sa 1 bitom a ostale da kodiramo sa 3 dobijamo da je potrebno za predmetni kod 1+31x3=94 bita a to znači da je ušteda samo 6% uz dodatno procesiranje. Stoga kodiranje fax dokumenata na primjer podrazumjeva i gledanje korelacije prethodnih kodnih riječi. Npr. redovi od drugog do osmog se malo razlikuju kao što se međusobno malo razlikuju deveti i deseti red. Međutim, ovo izlazi van okvira kursa.

Naredni kod koji se rijetko provodi samostalno je diferencijalni kod. Kod sporopromjenljivih podataka se često pamti prva vrijednost i zatim umjesto originalnih vrijednosti razlike između pojedinih susjednih vrijednosti. Često se izlaz iz diferencijalnog kodera dovodi do sistema za Huffmanovo kodiranje.

Pretpostavimo na primjer da imamo kod koji prati temperaturu u jednočasovnim intervalima.

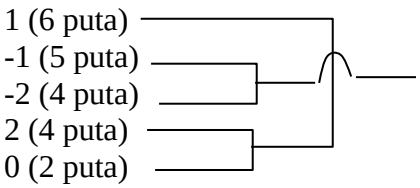
6 5 4 2 0 -2 -1 -1 0 1 2 3 5 7 9 11 12 10 9 8 7 7

Domen vrijednosti je [-2,12] i obuhvata 15 različitih vrijednosti za čije kodiranje je potrebno 22x4=88 bita.

Diferencijalnim kodom ovo kodiramo na taj način što ponovim prvi simbol a zatim pišemo razlike drugo i prvog, trećeg i drugog itd:

6 -1 -1 -2 -2 -2 1 0 1 1 1 1 2 2 2 2 1 -2 -1 -1 -1 0

U ovom slučaju prvi simbol ponovo kodiramo sa 4 bita dok narednih 14 simbola možemo kodirati sa samo 3 bita jer predstavljaju 5 mogućih vrijednosti razlika [-2,2] tako da dobijamo 4+3x21=70 bita u poruci. Pogledajmo još jednu mogućnost a to je da kodiramo simbole [-2,2] sa Huffmanovim kodom. Simboli ovog skupa se pojavljuju redom [4 puta -2, 5 puta -1, 2 puta 0, 6 puta 1 i 4 puta 2]



Simboli 1, -1 i 2 su kodirani sa 2 bita dok su simboli 2 i 0 kodirani sa 3 bita. Za kodiranje nam je dakle potrebno: 4+15x2+6x3=42 bita što predstavlja veoma značajnu uštedu od (88-42)/88=52.2% bita.

Ovdje dajemo jednostavnu MATLAB-funkciju za RLE kodiranje:

```
function y=rle_kod(x)
N=length(x);
y(1)=x(1);
y(2)=1;
l=1;
for k=2:N
   k
   if(x(k)==x(k-1)),y(l+1)=y(l+1)+1;
else,l=l+2;y(l)=x(k);y(l+1)=1;end
end
```

Uzimamo prvi odbirak da je element koda a da je drugi 1. Ovaj drugi predstavlja broj pojavljivanja prvog elementa ako je naredni element isti kao prethodni `x(k)==x(k-1)` uvećavamo `y(l+1)=y(l+1)+1` ako ovo nije zadovoljeno onda idemo na novi par, pa prvi element postavljamo da je `x(k)` a drugi da je 1 i drugi element se uvećava za svako ponavljanje elementa u polaznom nizu. Dekodiranje se može obaviti na još jednostavniji način, na primjer:

```
function x=rle_dec(y)
N=length(y);
x=zeros(1,length(sum(y(2:2:N))));
ind=0;
for k=1:2:N
x(ind+1:ind+y(k+1))=y(k)*ones(1,y(k+1));
ind=ind+y(k+1);
end
```

Realizacija je jednostavna uzimamo parove elemenata iz niza y i zatim prebacujemo u niz x podnizove sa odgovarajućim elementom odgovarajuće dužine `y(k)*ones(1,y(k+1))`.

4.6. Lempel-Ziv kod

Huffmanov kod je veoma efikasan kod kodiranja, kada su poznate makar procjene vjerovatnoća pojavljivanja pojedinih simbola. Nažalost, vjerovatnoće u mnogim situacijama nije moguće odrediti unaprijed. Lempel i Ziv su 1977-predložili najpopularniji kod za kompresiju bez gubitaka. Posebno je poznata njegova ekstenzija LZW kod (W potiče od Weltch koji je predložio određena poboljšanja 1984-te godine). Ovaj kod pripada grupi kodova koji su zasnovani na kodnoj knjizi (code book) ili kako se često kaže rječniku (dictionary based) u kojem se pamte dosadašnji stringovi (ponavljanja kodnih simbola). Posmatrajmo primjer polazeći od praznog rječnika i pod pretpostavkom da naš rječnik nema više od 8 simbola kada je pun. Neka nam je dat string:

1011010100010...

Ovaj stringa se odvaja najkraći string koji prethodno nije postojao u rječniku. String se upisuje kao novi elemenat u rječnik, koji se sastoji od prethodnog prefiksa i novog bita. Npr. bit 1 nije bio viđen prethodno i zato mi šaljemo indeks njegovog prefiksa (postavljen na 000) i zatim broj 1. Dodamo 1 u riječnik. Zatim nula nije postojala prethodno i mi postavljamo indeks njenog prefiksa i broj 0. Dodamo 0 u rječnik. Niz 11 ima prefiksni string 1 i zato se šalje prefiks string od 1 i pa šaljemo njegov indeks i broj 1. Nakon ove operacije riječnik će izgledati:

LZ indeks	sadržaj
000	null
001	1
010	0
011	11
100	01
101	010
110	00
111	10

Pa kodirana sekvenca ima oblik:

(000,1),(000,0),(001,1),(010,10),(100,0),(010,0),(001,0)

Naravno u predmetnom primjeru podaci nijesu komprimovani. Umjesto 13 bita dobili smo 28. Međutim, za duži niz podataka i veću kodnu knjigu dobili bi smo relativno visok stepen kompresije.

Lempel-Ziv su dokazali da se asimptotski izlazni odnos približava odnosu entropija za stacionarne izvore. Dokazi ove činjenice su veoma komplikovani, međutim, osnovu predstavlja sledeća teorema.

Teorema: Neka $\{X_i\}_{-\infty}^{\infty}$ bude ergodični stohastički proces. Neka $l(X_1, X_2, ..., X_n)$ bude dužina kodne riječi za Lempel-Ziv kod pridružen X_1, X_2, ..., X_n. Tada:

$$\limsup_{n\to\infty} \frac{1}{n} l(X_1, X_2, ..., X_n) \le H(\chi)$$

Dokaz ove na prvi pogled jednostavne teoreme je izuzetno složen i prevazilazi nivo ove knjige.

Očigledne popravke osnovnog primjera su da se u početku tabele kodova za indeksiranje tabele kodova koristi manji broj bita nego kasnije. Druga implementaciona pitanja koja treba riješiti su:

- performanse kompresije u odnosu na veličinu tabele;
- inicijalizacija rječnika (startovati sa praznim riječnikom ili startovati sa nekim alfabetom);
- organizacija riječnika i strategija pretraživanja;
- adaptacija (što raditi sa sekvencama koje su rijetko korišćene).

Primjer. Posmatrajmo alfabet sa tri simbola $\{a, b, c\}$. U ovom slučaju je pretpostavljeno da rječnik posjeduje alfabet izvora. Sekvenca je:

ababcbababaaaaa...

Izlaz je (1,*b*),(4,*c*),(2,*a*),(6,*b*),(1,*a*),(8,*a*)...

Dok je formirani riječnik:

LZ indeks	1	2	3	4	5	6	7	8	9
sadržaj	*a*	*b*	*c*	*ab*	*abc*	*ba*	*bab*	*aa*	*aaa*

Dekodiranje LZW koda u svim varijantama se obavlja na isti način kako se vrši kodiranje. Naime, u ovom slučaju polazimo od rječnika koji se sastoji od koda izvora:

LZ indeks	1	2	3

sadržaj *a* *b* *c*

Primljeni par je sada (1,*b*) kojega dekodiramo kao *ab* (sadržaj rječnika sa lokacije 1 i sufiks *b*). U rječnik upisujemo na lokaciji 4 poruku *ab*. Zatim imamo (4,*c*) koje dekodiramo kao *abc* i tu poruku upisujemo na lokaciju 5. Zatim stiže (2,*a*) što se dekodira kao *ba* i smiješta na lokaciju 6. Par (6,*b*) dekodiramo kao *bab* koje upisujemo na lokaciju 7, (1,*a*) se dekodira kao *aa* što se upisuje na lokaciju 8 dok se (8,*a*) dekodira kao *aaa* koje se upisuje na narednu lokaciju. I za ostale varijante LZW koda obaviti dekodiranje na odgovarajući način.

Napomena: Treba reći da je veća zasluga Ziva predmetnom kodu nego Lempela. Svi ovi detalji su obrađeni u [2].

Posmatrajmo moguću praktičniju realizaciju LZW koda na primjeru nabrajalice iz našeg djetinjstva:

Eci peci pec ti si mali zec a ja mala prepelica eci peci pec.

Kodiraćemo 30 karaktera (lj, nj i dž ćemo brojati kao 1) jezika binarnim ekvivalentom pozicije u azbuci datog slova (A=1=00001, M=15=01111, Š=30=11110). Kraj sekvence ćemo kodirati kao 0=00000 dok ćemo bjelinu kodirati kao 11111. Te vrijednosti ćemo i upisati u rječnik. Dodatna dva simbola ź i ś nećemo kodirati zbog toga što se relativno rijetko pojavljuju u riječima. U riječi ima 107 karaktera, i 15 bjelina. Znakove interpunkcije nećemo računati.

Prvo šaljemo E (kodiramo kao 00111) a u rječnik smiještamo
EC i kodiramo kao 100000 (vrijednost 32 od sada pa nadalje sve poruke koje šaljemo moraju da imaju 6 bita)
Zatim šaljemo C (kodiramo kao 011011) čime gubimo jedan bit a u rječnik smiještamo
CI i kodiramo kao 100001
Zatim šaljemo I (kodiramo kao 001010) čime gubimo drugi bit a u rječnik smiještamo
I_ i kodiramo kao 100010
Šaljemo _ (000000) čime gubimo i treći bit a u rječnik smiještamo
_P i kodiramo kao 100011
Šaljemo P (010011) čime gubimo i četvrti bit a u rječnik smiještamo
PE i kodiramo kao 100100
Zatim iz rječnika šaljemo EC (100000) a time gubitak poništavamo
ECI i kodiramo kao 100101
Šaljemo iz rječnika I_ (100010) čime štedimo 4 bita a u rječnik smiještamo
I_P i kodiramo kao 100110
Šaljemo PE (100100) štedimo dodatna 4 bita (dobit je sada 8 bita) u rječnik smiještamo
PEC i kodiramo kao 100111
Šaljemo C (011011) gubimo jedan bit (smanjujemo dobit na 7 bita) a u rječnik smiještamo
C_ i kodiramo kao 101000
Šaljemo _ (000000) čime gubimo jedan bit (dobit je 6 bita) a u rječnik smiještamo
_T i kodiramo kao 101001
Šaljemo T (010110) čime gubimo jedan bit (dobit je 5 bita) a u rječnik smiještamo
TI i kodiramo kao 101010
Šaljemo I_ (100010) čime dobijamo 4 bita (dobit je 9 bita) a u rječnik smiještamo
I_S i kodiramo kao 101011
Šaljemo S (010101) čime gubimo 1 bit (dobit je 8 bita) a u rječnik smiještamo
SI i kodiramo kao 101100
Šaljemo I_ (100010) dobijamo 4 bita (dobit je 12 bita) a u rječnik smiještamo
I_M i kodiramo kao 101101
Šaljemo M (001111) gubimo jedan bit (dobit je 11 bita) a u rječnik smiještamo

MA i kodiramo kao 101110
Šaljemo A (000001) gubimo jedan bit (dobit je 10 bita) a u rječnik smiještamo
AL i kodiramo kao 101111
Šaljemo L (001101) gubimo jedan bit (dobit je 9 bita) a u rječnik smiještamo
LI i kodiramo kao 110000
Šaljemo I_ (100010) dobijamo 4 bita (dobit je 13 bita) a u rječnik smiještamo
I_Z i kodiramo kao 110001
Šaljemo Z (001001) gubimo 1 bit (dobit je 12 bita) a u rječnik smiještamo
ZE i kodiramo kao 110010
Šaljemo EC (100000) dobijamo 4 bita (dobit je 16 bita) a u rječnik smiještamo
EC_ i kodiramo kao 110011
Šaljemo zatim _ (011111) gubimo jedan bit (dobit je 15 bita) a u rječnik smiještamo
_A i kodiramo kao 110100
Šaljemo A (000001) gubimo jedan bit (dobit je 14 bita) a u rječnik smiještamo
A_ i kodiramo kao 110101
Šaljemo _ (011111) gubimo jedan bit (dobit je 13 bita) a u rječnik smiještamo
_J i kodiramo kao 110110
Šaljemo zatim J (001011) gubimo jedan bit (dobit je 12 bita) a u rječnik smiještamo
JA i kodiramo kao 110111
Šaljemo A_ (110101) dobijamo 4 bita (dobit je 16 bita) a u rječnik smiještamo
A_M i kodiramo kao 111000
Šaljemo MA (101110) dobijamo 4 bita (dobit je 20 bita) a u rječnik smiještamo
MAL i kodiramo kao 111001
Šaljemo L (001101) gubimo 1 bit (dobit je 19 bita) a u rječnik smiještamo
LA i kodiramo kao 111010
Šaljemo A_ (110101) dobijamo 4 bita (dobit je 23 bita) a u rječnik smiještamo
A_P i kodiramo kao 111011
Šaljemo P (010011) gubimo 1 bit (dobit je 22 bita) a u rječnik smiještamo
PR i kodiramo kao 111100
Šaljemo R (010100) gubimo 1 bit (dobit je 21 bit) a u rječnik smiještamo
RE i kodiramo kao 111101
Šaljemo E (000111) gubimo 1 bit (dobit je 20 bita) a u rječnik smiještamo
EP i kodiramo kao 111110
Šaljemo PE (100100) dobijamo 4 bita (dobit je 24 bita) a u rječnik smiještamo
PEL i kodiramo kao 111111
Šaljemo LI (110000) dobijamo 4 bita (dobit je 28 bita) a u rječnik smiještamo
LIC i kodiramo sada sedmobitnim kodom 1000000 (od sada će svi zapisi morati biti sedmobitni)
Šaljemo C (0011011) gubimo 2 bita (dobit je 26 bita) a u rječnik smiještamo
CA i kodiramo sa 1000001
Šaljemo A_ (0110101) dobijamo 3 bita (dobit je 29 bita) a u rječnik smiještamo
A_E i kodiramo 1000010
Šaljemo ECI (0100101) dobijamo 8 bita (dobit 37 bita) a u rječnik smiještamo
ECI_ i kodiramo 1000011
Šaljemo _P (0100011) dobijamo 3 bita (dobit je 40 bita) a u rječnik smiještamo
_PE i kodiramo kao 1000100
Šaljemo ECI_ (1000011) dobijamo 13 bita (dobit je 53 bita) a u rječnik smiještamo
ECI_P i kodiramo kao 1000101
Šaljemo konačno PEC (0100111) čime ostvarujemo dobit od 8 bita tako da je ukupna dobit ostvarena primjenom LZW koda na ovu dječiju nabrajalicu 61 bit.

Ostvarena dobit je dosta značajna jer u navedenoj riječi ima 60 karaktera (uključujući bjeline) tako da bi kodiranjem kodom fiksne dužine bilo potrebno 60x5=300 karaktera dok je u navedenom

primjeru potrebno 300-61=239 bita što je značajno smanjenje od nešto više od 20%. Uporedite što bi se moglo dobiti u kodiranju ove dosta kratke sekvence varijantama Huffmanovog kodiranja.

U trenutku pisanja ovog materijala Matlab nije imao funkciju niti toolbox koji je realizovao LZW kodiranje. Neke verzije Communications toolbox-a su posjedovale funkcije koje su se mogle koristiti za LZW kodiranje i dekodiranje. Umjesto toga možete koristiti funkcije norm2lzw i lzw2norm koje su preko MATLAB-ovom sajta za razmjenu fajlova dostupne (autor je Giuseppe Ridino. Postoji demo program koji se može iskoristiti za testiranje ovih funkcija (napominjem da su riječi razdvojene specijalnim simbolom \, te da je pretpostavljeno kodiranje sa predefinisanom ASCII tabelom koja je smještena u rječnik). U svakom slučaju za programera ne bi trebalo da predstavlja nepremostiv problem da na osnovu ovog ili nekog drugog koda ili čak i bez njega napravi sopstvenu verziju funkcija za LZW kodiranje i dekodiranje.

4.7. Aritmetički kodovi

Aritmetički kodovi su alternativa Huffmanovim kodovima. Mišljenje je da je to skupa alternativa. Naime, Huffmanov kod nije zaštićen autorskim pravima i može se slobodno koristiti. Vlasnik većine patenata iz oblasti aritmetičkog kodiranja je IBM. Kod koji će biti demonstriran je vezan za pseudovjerovatnoću a postoje alternative vezane za broj pojavljivanja. Ušteda kod aritmetičkog kodiranja u odnosu na Huffmanovo je 5-10%. Upotreba standardnih aritmetičkih kodova je zaštićena autorskim pravima Kod aritmetičkih kodova postoje dvije faze u kodiranju: (a) Prediktivna faza; (b) Faza kodiranja. Aritmetički kodovi vrše preslikavanje realne linije na interval 0 do 1. Kod 01 se može interpretirati kao 0.01 što predstavlja binarni interval [0.01,0.10) (značenje srednje zagrade je da je uključena vrijednost dok mala zagrada označava da vrijednost ne pripada intervalu) odnosno [0.25,0.5) dekadno. Npr. kod 0.01101 označava interval [0.01101,0.01110). Jasno je da duži niz odgovara užem intervalu. Pretpostavimo da radimo sa alfabetom $A=\{a_1,a_2,...,a_I\}$, gdje je a_I specijalni simbol, koji znači kraj prenosa. Izvor produkuje sekvencu x_1, x_2, ..., x_n,..., gdje simboli ne moraju biti nezavisni, niti sa istom distribucijom. Modelujmo prediktor kao:

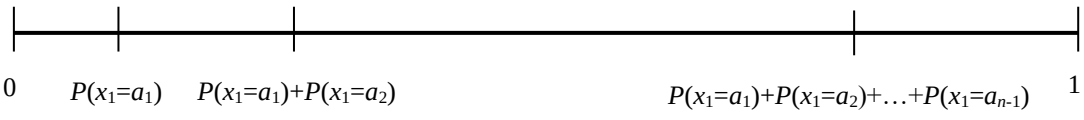
$$P(x_n = a_i \mid x_1, x_2, ..., x_{n-1})$$

Dijelimo segment [0,1) u I intervala čija je dužina jednaka vjerovatnoćama $P(x_1 = a_i), i = 1,2,...,I$. Prvi interval je:

$$[0, P(x_1 = a_1))$$

Drugi interval je:

$$[P(x_1 = a_1), P(x_1 = a_1) + P(x_1 = a_2))$$



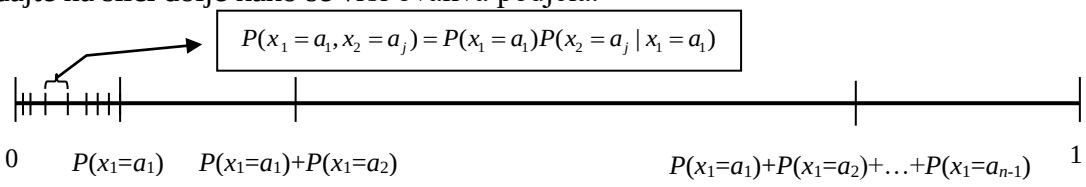
Ako posmatramo i druge simbole osim x_1, možemo definisati gornju i donju kumulativnu vjerovatnoću:

$$Q_n(a_i \mid x_1,...,x_{n-1}) = \sum_{j=1}^{i-1} P(x_n = a_j \mid x_1,...,x_{n-1}) \qquad R_n(a_i \mid x_1,...,x_{n-1}) = \sum_{j=1}^{i} P(x_n = a \mid x_1,...,x_{n-1})$$

Tada npr. za a_2 odgovara interval $[Q(a_2), R_1(a_2))$. Sada odredimo vjerovatnoću za naredni simbol. Uzmimo interval a_1 i podijelimo ga u intervale a_1a_1, a_1a_2, ..., a_1a_I na taj način da je dužina podintervala a_1a_j proporcionalna $P(a_j|a_1)$. U stvari, uzimamo da su dužine podintervala za a_1a_j

$$P(x_1 = a_1, x_2 = a_j) = P(x_1 = a_1)P(x_2 = a_j \mid x_1 = a_1)$$

Pogledajte na slici dolje kako se vrši ovakva podjela.



Za sumu dužina podintervala važi:

$$\sum_j P(x_1 = a_1, x_2 = a_j) = P(x_1 = a_1)$$

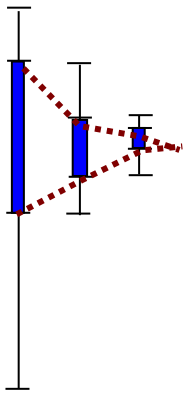
Na sličan način se svaki podinterval za $a_i a_j$ dijeli tako da važi:

$$P(x_1 = a_i, x_2 = a_j) = P(x_1 = a_1)P(x_2 = a_j \mid a_1 = a_i)$$

Zatim nastavljamo sa dijeljenjem svakog podintervala u stringove dužine N. Posmatrajmo algoritam kako da računamo interval $[u,v)$ za string $x_1x_2...x_N$.

Algoritam 1.

u=0.0 //donja granica intervala
v=1.0 //gornja granica intervala
$p=v-u$ //dužina tekućeg intervala (dužina sekvence)
for n=1 do N {
    računanje gornje i donje kumulativne vjerovatnoće
    izračunaj $R_n(i, x_1, ..., x_{n-1})$ za i=1,2,...,I
    izračunaj $Q_n(i, x_1, ..., x_{n-1})$ za i=1,2,...,I
    $u = u + pQ_n(x_n \mid x_1, x_2, ..., x_{n-1})$ //ažuriraj donju granicu sa združenom dužinom
    $v = u + pR_n(x_n \mid x_1, x_2, ..., x_{n-1})$ //ažuriraj gornju granicu sa združenom dužinom
    $p=u-v$ //računanje nove dužine (nove vjerovatnoće)
}



Na dijagramu se vidi kako se sa svakim novim simbolom širina moguće pseudovjerovatnoće sekvence postepeno smanjuje.

Prije nego pokažemo kodiranje aritmetičkim kodom usvojimo sledeće pravilo, koje se u literaturi naziva Laplasovim, za proračun uslovnih vjerovatnoća potrebnih za rad sa ovim kodom. Ovo je pravilo jedno od mogućih za ove namjene, ali ne i jedino. Kod brojnih sistema gdje se uslovne vjerovatnoće mogu procijeniti van algoritma se ne koristi. Po Laplasovom pravilu se usvaja alfabet $A = \{a_1, a_2, ..., a_I\}$. Pretpostavi se da su vjerovatnoće pojave pojedinih simbola na početku jednake:

$$P(x_1 = a_i) = 1/I$$

Za *n*-ti ulazni simbol uslovna vjerovatnoća se računa kao:

$$P(x_n = a_i \mid x_1,...,x_{n-1}) = \frac{F_i + 1}{\sum_{i=1}^{I}(F_i + 1)}$$

gdje je F_i broj pojavljivanja simbola F_i u sekvenci koja je prethodila datom simbolu (u sekvenci $x_1,...,x_{n-1}$). Postoje brojne modifikacije ovog pravila od kojih je najpoznatija Dirichletova koja vrši ažuriranje kao:

$$P(x_n = a_i \mid x_1,...,x_{n-1}) = \frac{F_i + \alpha}{\sum_{i=1}^{I}(F_i + \alpha)}$$

gdje je α neki mali pozitivni broj (npr. α=0.01). Čest je dalje slučaj kod Laplasovog pravila da neki od simbola ima posebna pravila za ažuriranje. Npr. uslovna vjerovatnoća za terminacioni simbol se ažurira po nekom specifičnom pravilu $P(x_n = a_I \mid x_1,...,x_{n-1})$. Tada se ostali simboli ažuriraju (ako je u pitanju Laplasova varijanta) kao:

$$P(x_n = a_i \mid x_1,...,x_{n-1}) = \frac{F_i + 1}{\sum_{i=1}^{I-1}(F_i + 1)}[1 - P(x_n = a_I \mid x_1,...,x_{n-1})],\ i \neq I$$

Nekoliko napomena o razlozima za izbor Laplasovog pravila kod aritmetičkog kodiranja će biti dato u sekciji 4.8. Ostalim tipovima ažuriranja uslovnih vjerovatnoća neće biti razmatrani mada su neki od njih prirodno razumljivi (npr. kod govora veća je vjerovatnoća pojave slova "t" nakon slova "s" nego slova "š" nakon slova "s"). Napomenimo dalje, da se u praksi koriste i drugi tipovi aritmetičkog kodiranja koji se u mnogo čemu razilikuju, osim po nazivu, od prezentiranog tipa. Kod tih drugih aritmetičkih kodova vrši se kodiranje na osnovu međusobne korelacije simbola koji se kodiraju. Ovo kodiranje, umjesto modela zasnovanog na uslovnim vjerovatnoćama, koristi korelaciju signala.

Primjer 4.7.1: Posmatrajmo prenos poruka *a* i *b*. Neka oznaka & predstavlja kraj prenosa. Pretpostavimo da želimo kodirati *bbba*&. Dobijamo sledeću tabelu (određene vjerovatnoće će biti diskutovane kasnije):

Sadržaj (sekvenca do sada)	Vjerovatnoća sledećeg simbola
	$P(a)$=0.425 $P(b)$=0.425 $P(\&)$=0.15
b	$P(a\|b)$=0.28 $P(b\|b)$=0.57 $P(\&\|b)$=0.15
bb	$P(a\|bb)$=0.21 $P(b\|b)$=0.64 $P(\&\|bb)$=0.15
bbb	$P(a\|bbb)$=0.17 $P(b\|bbb)$=0.68 $P(\&\|bbb)$=0.15
bbba	$P(a\|bbba)$=0.28 $P(b\|bbba)$=0.57 $P(\&\|bbba)$=0.15

Neka je dato $P(\&)$=0.15 (uzećemo da je ova vjerovatnoća konstanta i da ne zavisi od broja primljenih simbola kao ni od prethodno primljenog dijela poruke) kao i da je vjerovatnoća pojavljivanja dva simbola $P(a)$=$P(b)$=0.425. Nakon svakog primljenog simbola vršićemo ažuriranje vjerovatnoće tako što je:

$$P(a \mid \text{Prethodno}) = \frac{F_a + 1}{F_a + F_b + 2}[P(a) + P(b)] \qquad P(b \mid \text{Prethodno}) = \frac{F_b + 1}{F_a + F_b + 2}[P(a) + P(b)]$$

gdje je F_a broj *a*-ova koji je primljen do sada u sekvenci, dok je F_b broj simbola *b* koji je primljen do sada u sekvenci. Množenje sa $[P(a)+P(b)]$ se obavlja stoga što pretpostavljamo da terminacioni simbol ne učestvuje u operacijama. Prođimo sada korak po korak kroz proceduru.

u=0.0 *v*=1.0 *p*=*v*-*u*=1.0. Intervali prije početnog simbola su: $[0, P(a)) = [0, 0.425)$, $[P(a), P(a)+P(b)) = [0.425, 0.85)$ i $[P(a)+P(b), P(a)+P(b)+P(\&)) = [0.85, 1)$.

Dobijen je simbol *b* to znači da se naše rješenje nalazi u drugom intervalu $[P(a), P(a)+P(b)) = [0.425, 0.85)$. Binarno zapisan ovaj interval je [0.0...., 0.1...) odnosno nije određena ni prva cifra iza decimalnog zareza. Izvršimo ažuriranje vjerovatnoća.

Sada je *u*=0.425 *v*=0.85 *p*=*v*-*u*=0.425.
$P(a \mid \text{Prethodno}) \approx 0.28$, $P(b \mid \text{Prethodno}) \approx 0.57$, $P(\& \mid \text{Prethodno}) = 0.15$. Sada su podintervali: [0,0.28), [0.28,0.85) i [0.85,1). Primljen je ponovo simbol *b* tako da je odabran drugi interval i ažuriranje *u* i *v* se može obaviti na sledeći način:

u=*u*+*p* 0.28=0.544 *v*= *u*+*p* 0.85=0.7862 Binarno zapisane granice intervala su: [0.100010...,0.11001...). Odnosno, zasigurno je određena prva cifra i to je 1. Novo *p*=0.2422.

Sada je $P(a \mid \text{Prethodno}) \approx 0.21$, $P(b \mid \text{Prethodno}) \approx 0.64$, $P(\& \mid \text{Prethodno}) = 0.15$ (primljeno dva puta *b* i nijednom *a*). Opet je primljeno *b*, što znači da je selektovan interval [0.21,0.85). Nakon ažuriranja se dobija:
u=*u*+*p* 0.21= 0.5953 *v*= *u*+*p* 0.85=0.75158 *p*=0.1563. Novi binarni interval je: [0.100110...,0.110000) odnosno ponovo se nije precizirala naredna cifra.

Sada je $P(a \mid \text{Prethodno}) = 0.17$, $P(b \mid \text{Prethodno}) = 0.68$, $P(\& \mid \text{Prethodno}) = 0.15$ (primljeno tri puta *b* i nijednom *a*). Pošto je primljeno *a* odabran je uski interval [0,0.17), pa se nakon ažuriranja dobija:
u=*u*+*p* 0.0= 0.5953 *v*=*u*+*p* 0.17=0.6219 *p*=0.02657. Novi binarni interval je [0.100110...,0.100111), koji je potpuno određen na pet decimala.

Sada je $P(a \mid \text{Prethodno}) \approx 0.28$, $P(b \mid \text{Prethodno}) \approx 0.57$, $P(\& \mid \text{Prethodno}) = 0.15$ (primljeno tri puta *b* i jednom *a*). Pošto se predaje terminacioni karakter onda je odabran interval [0.85,1). Nakon ažuriranja se dobija interval:
u=*u*+*p* 0.85=0.61787 *v*=*u*+*p* 0.17=0.6219. To je binarno [0.10011110...,0.10011111...). Sada se očigledno vrijednost 1001111 može poslati kanalom, jer dovoljno precizno determiniše poslatu poruku.

Dekoder prima ovu poruku i obrađuje je bit po bit. Vjerovatnoće se određuju kao i prethodno. Kada se primi 10 poznato je da string počinje sa *b* jer 10 leži unutar intervala *b*. Nakon ovoga dekoder računa vjerovatnoće *P*(*a*|*b*), *P*(*b*|*b*), *P*(&|*b*) i na osnovu toga određuje granice intervala za *ba*, *bb* i *b*&. Procedura se nastavlja i dalje, dok se ne primi & što je znak za kraj poruke.

4.8. Model vjerovatnoće kod aritmetičkog kodiranja

Od dobrog modela vjerovatnoće izvora zavisi i kvalitet aritmetičkog koda. Posmatrajmo prethodni slučaj da se šalju simboli *a*, *b* i $. Neka su vjerovatnoće ovih događaja p_a, p_b i $p_\$$. Neka je *l* broj izlaza odnosno broj poslatih simbola *a* i *b*. Najčešće nemamo nikakvu informaciju o p_a i to može uzimati vrijednosti u intervalu [0,1]. To se može modelovati uniformnom distribucijom:

$$P(p_a)=1 \qquad \text{za} \qquad p_a \in [0,1]$$

Ako postoji neka informacija o prirodi *a* može se usvojiti neki drugi apriori model vjerovatnoće. Bayesova estimacija, koja se može primjeniti u ove svrhe polazi od apriori modela kojeg nadovezuje na dobijenu opservaciju. To se može opisati događajem: Posmatrana je sekvenca od *F* bita gdje je F_a broj *a* simbola i F_b broj *b* simbola. Vjerovatnoća da se *a* pojavljuje sa vjerovatnoćom p_a nakon što je mjerenje *s* obavljeno se može opisati kao:

$$P(p_a \mid s,F) = \frac{P(s \mid p_a,F)P(p_a)}{P(s \mid F)}$$

Uslovna vjerovatnoća u brojiocu se može zapisati kao:

$$P(s \mid p_a, F) = p_a^{F_a}(1-p_a)^{F_b}$$

Dalje se može odrediti da je:

$$P(s \mid F) = \int P(s \mid p_a, F)P(p_a)dp_a = \int p_a^{F_a}(1-p_a)^{F_b}P(p_a)dp_a = \frac{\Gamma(F_a+1)\Gamma(F_b+1)}{\Gamma(F_a+F_b+2)} = \frac{F_a!F_b!}{(F_a+F_b+1)!}$$

Sada se može odrediti vjerovatnoća p_a koja maksimizuje vjerovatnoću $P(p_a \mid s, F)$. Na primjer za $P(p_a \mid s = aba, F = 3) \propto p_a^2(1-p_a)$ što se maksimizuje za p_a=2/3. Takođe, željeli bi da smo u stanju da vršimo predikciju. Ako je data sekvenca s dužine F pronađimo predikciju a:

$$P(a \mid s, F) = \int P(a \mid p_a tt)P(p_a \mid s, F)dp_a$$

U ovom slučaju mi koristimo svu posteriori vjerovatnoću odnosno mi unosimo kompletnu neodređenost o p_a. Na osnovu ovoga se može odrediti prediktor:

$$P(a \mid s, F) = \int p_a \frac{p_a^{F_a}(1-p_a)^{F_a}}{P(s \mid F)}dp_a = \frac{F_a+1}{F_a+F_b+2}$$

Generalizacijom za višesimbolski alfabet se dobija Laplasovo pravilo. Vrijedi napomenuti da je ovo izvedeno pod apriornom pretpostavkom o uniformnoj raspodjeli simbola što u praksi često nije slučaj.

4.9. Alternativa aritmetičkom kodiranje

Prethodni model aritmetičkog koda uspostavlja vezu između sekvence simbola i kvazi-vjerovatnoće da se takva sekvenca pojavi. Kao što smo rekli riječ je o kvazi-vjerovatnoći odnosno o procjeni vjerovatnoće predmetnog ishoda. Statističke karakteristike ovakve procjene vjerovatnoće mogu da se pronađu u odgovarajućoj literaturi. Postoji niz problema kod ovog tipa estimatora (procjenjivača) koji je dobrim dijelom prevaziđen njegovom efikasnošću. Ovdje ćemo se osvrnuti kratko na drugi tip estimatora. Ti estimatori polaze od mogućnosti da se numerički ekvivalent nekog simbola poruka u trenutku n+1 izrazi preko numeričkih ekvivalenata poruke u prethodnim trenucima:

$$x(n+1)=a_0x(n)+a_1x(n-1)+...+a_Qx(n-Q)+\nu(n+1)$$

Ovakav sistem se u praksi naziva autoregresivnim modelom (AR) i za njegovo određivanje je neophodno da poznajemo prethodne simbole i set koeficijenata $\{a_0, a_1, \ldots, a_Q\}$. Dakle, ako znamo početna stanja za kodiranje poruke proizvoljne dužine dovoljno nam je poznavati set koeficijenata $\{a_0, a_1, \ldots, a_Q\}$. Naravno, postavlja se pitanje kako se određuje predmetni set koeficijenata jer postupak očigledno nije jednostavan. Pored toga preostaje problem da ovakva relacija nije precizna do kraja već uvijek kod prirodnih signala – poruka dolazi do odstupanja koje je modelovano šumom $\nu(n+1)$. Dakle, nakon određivanja koeficijenata $\{a_0, a_1, \ldots, a_Q\}$ potrebno je kodirati $\nu(n+1)$ i to na takav način da je broj bita za kodiranje ovih informacija manji od broja bita za kodiranje polaznog signala. Na primjer, ono što želimo da se dogodi je recimo da nakon određivanja seta koeficijenata razlika maksimuma i minimuma signala $\nu(n+1)$ bude mnogo manja od razlike maksimuma i minimuma originalnog signala $x(n)$ što bi sugerisalo da se relativno jednostavnom tehnikom kodiranja signala $\nu(n+1)$ možemo postići znatne uštede. Opet ukazujemo na činjenicu da su predmetne tehnike veoma složene a posebno računski. Ujedno studenti kojima je ovaj materijal namjenjen ne raspolažu ni aparatom koji bi im omogućio razumijevanje algoritama za određivanje ovih koeficijenata.

Zadaci za vježbu

4.1. Dokazati 4.4.1 lemu.

Rješenje: Ovdje ćemo dati dokaz u kratkim crtama. 1. Ako zamjenimo dužine kodne riječi u uslovu dobijamo kontradikciju odnosno kod nije optimalan. 2. Ako dvije najduže kodne riječi nemaju istu dužinu mi možemo tu najdužu kodnu riječ skratiti za jedan, jer će to i dalje ostati jedinstven trenutni kod (imajte na umu stablo kodiranja). 3. Pogledati prethodni stav i detaljniji dokaz ove leme u materijalu.

4.2. Pretpostavimo da nijesu poznate tačne vrijednosti vjerovatnoća pojavljivanja pojedinih kodnih simbola, već da se prilikom njihove procjene pravi greška e_i. Odrediti razliku u prosječnoj dužini kodne riječi u ovom slučaju kao i varijansu greške koja se pravi.

Rješenje: Neka su p_i tačne vrijednosti vjerovatnoća u procesu dizajna Huffmanovog koda. Neka su:

$$p_i' = p_i + e_i$$

vjerovatnoće koje su korišćene u dizajnu koda (e_i je greška u procjeni pojavljivanja pojedinih simbola). Očigledno važi: $\frac{1}{q}\sum_{i=1}^{q} e_i = 0$. Neka je:

$$\frac{1}{q}\sum_{i=1}^{q} e_i^2 = \sigma^2$$

mjera greške u procjeni vjerovatnoće kodnih simbola. Odredimo prosječnu dužinu koda:

$$L' = \frac{1}{q}\sum l_i p_i' = \frac{1}{q}\sum l_i p_i + \frac{1}{q}\sum l_i e_i = L + \frac{1}{q}\sum l_i e_i$$

Poslednji član je promjena u dužini prosječne kodne riječi. Da bi odredili ekstremne vrijednosti ovog izraza koristimo metod Lagranžovih množilaca i prethodna dva ograničenja:

$$\Lambda = \frac{1}{q}\sum l_i e_i - \lambda\frac{1}{q}\sum e_i - \mu\left(\frac{1}{q}\sum e_i^2 - \sigma^2\right)$$

Odredimo parcijalne izvode po e_i i izjednačimo sa nulom:

$$\frac{\partial\Lambda}{\partial \mathrm{e_i}} = \frac{1}{q}(l_i - \lambda - 2\mu e_i) = 0$$

Kada se ove jednačine posumiraju za svako *i* dobija se: $\lambda = \frac{1}{q}\sum_i l_i$. Ako se jednačine pomnože sa e_i i posumiraju dobija se:

$$\frac{1}{q}\sum_i l_i e_i - 2\mu\frac{1}{q}\sum_i e_i^2 = 0$$

Na osnovu ove jednakosti dobijamo izraz za μ. Množeći prethodnu jednačinu sa l_i i sumirajući po *i* uz uvrštavanje vrijednosti za λ i μ iz prethodnih jednačina dobijamo:

$$\frac{1}{q}\sum_i l_i^2 - \left(\frac{1}{q}\sum_i l_i\right)^2 - \frac{\left(\frac{1}{q}\sum_i e_i l_i\right)^2}{\frac{1}{q}\sum_i e_i^2} = 0$$

$$\left(\frac{1}{q}\sum_i e_i l_i\right)^2 = \left[\frac{1}{q}\sum_i l_i^2 - \left(\frac{1}{q}\sum_i l_i\right)^2\right]\sigma^2 = \mathrm{var}[l_i]\,\mathrm{var}[e_i]$$

Na osnovu ovoga možemo zaključiti da mala greška u procjeni vjerovatnoće pojavljivanja simbola neće značajno promjeniti prosječnu dužinu kodne riječi u odnosu na optimalnu sve dok varijansa l_i nije ekstremno velika. Ovo ukazuje na potrebu da u slučaju da u procesu Huffmanovog kodiranje imamo mogućnost da biramo simbole sa više istih vjerovatnoća selektujemo one koji su se u dosadašnjem postupku kodirali sa manjim brojem bita kako bi smanjili najduže kodne riječi odnosno smanjili varijansu l_i.

4.3. Posmatrajte trivijalan slučaj koda kod koga je p=0.9999 i q=0.0001. Koliku će prosječnu dužinu dati kodiranje sa $\lceil -\log_2 p_i \rceil$ bita, a koliku Huffmanovo kodiranje. Što na osnovu ovoga možete da zaključite?

Rješenje: $-\log_2(0.9999)\approx 0.00014$ dok je $-\log_2(0.0001)\approx 13.28$. Dakle, simbol sa velikom vjerovatnoćom bi se kodirao sa 1 bitom dok bi se simbol sa malom vjerovatnoćom kodirao sa 14bita. Prosječna dužina kodne riječi u ovom slučaju je 1.0013bita po simbolu. Huffmanov kod je očigledno efikasniji pošto je za kodiranje potrebno po jedan bit po simbolu (jedan se simbol kodira

sa nulom a jedan sa jedinicom). Ovo je još jedan, istina posredan, dokaz optimalnosti Huffmanovog kodiranja.

4.4. Izvršiti aritmetičko kodiranje ternarnog koda sa simbolima (*a*,*b*,*c*). Terminacioni simbol je $. Vjerovatnoća terminacionog simbola za prvi karakter je 0.1 a za svaki naredni raste za po 0.05. Za prvi karakter se može pretpostaviti da su vjerovatnoće karaktera (*a*,*b*,*c*) iste. Za svaki naredni se vrši ažuriranje na osnovu broja karaktera koji su se do sada pojavili u sekvenci (koristiti Laplasovo pravilo za ažuriranje uvedeno na predavanjima). Kodirati poruku **abbbbacc$**.

Rješenje: Kod prvog simbola vjerovatnoće su *a* u granicama [0,0.3), *b* u granicama [0.3,0.6), *c* u granicama [0.6,0.9) a $ [0.9,1). Pošto je poslato a onda imamo granice [0,0.3).

Prilikom kodiranja narednog simbola vjerovatnoća simbola $ je 0.15 ($P(\$|a)$=0.15) dok se ostali raspoređuju $P(a|a)$=0.5, $P(b|a)=P(c|a)$=0.25. Dakle, sada su granice: za *a* je granice [0,0.3x0.5x0.85)=[0,0.1275), dok je simbol *b* u granicama [0.1275,0.1275+0.3x0.25x0.85)= [0.1275,0.19125), simbol *c* je u granicama [0.19125, 0.19125+03x0.25x0.85)=[0.19125,0.255), konačno terminacioni simbol je u granicama [0.255,0.3). Kako smo poslali simbol *b* nakon a to znači da smo izabrali interval [0.1275,0.19125).

Prilikom kodiranja narednog simbola vjerovatnoća terminacionog simbola je 0.2 dok su vjerovatnoće simbola *a* i *b* $P(a|ab)=P(b|ab)$=2/5=0.4 dok je vjerovatnoća $P(c|ab)$=0.2. Razmak između gornje i donje granice intervala je 0.06375. Kako je ponovo izabran simbol *b* dobili smo da su nove granice u intervalu: [0.1275+0.06375x0.8x0.4,0.1275+0.06375x0.8x0.8)=[0.1479,0.1683). Sveli smo granice na interval širine 0.0204.

Vjerovatnoća terminacionog simbola je sada 0.25 dok su vjerovatnoće simbola *a*, *b* i *c*: redom 1/3, 1/2 i 1/6. Ponovo se šalje *b* pa je selektovan interval [0.1479+0.0204x0.75/3, 0.1479+0.0204x0.75x5/6)=[0.153,0.16065). Širina intervala je 0.00765.

Sada vjerovatnoća terminacionog simbola raste do 0.3. Vjerovatnoće simbola alfabeta su sada redom 2/7, 4/7 i 1/7. Ponovo se šalje *b* simbol pa su granice intervala sada:
[0.153+0.00765x0.7x2/7, 0.153+0.00765x0.7x6/7)=[0.15453,0.15759). Širina intervala je 0.00306.

Vjerovatnoća terminacionog simbola sada raste do 0.35. Vjerovatnoće simbola alfabeta su 1/4, 5/8 i 1/8. Pošto je poslato *a* dobijamo interval:
[0.15453, 0.15453x0.65x0.00306/4)=[0.15453,0.15502725). Širina intervala je svedena na 0.00049725.

Prilikom kodiranja narednog simbola vjerovatnoća terminacionog karaktera je 0.4, dok su vjerovatnoće pojedinih simbola 3/9, 5/9 i 1/9. Pošto je selektovan *c* granice intervala su:
[0.15453+0.00049725x0.6x8/9, 0.15453+0.00049725x0.6)=[0.1547952, 0.15482835). Širina intervala je sada: 0.00003315.

Prilikom kodiranja narednog simbola vjerovatnoća terminacionog karaktera je 0.45 dok su vjerovatnoće ostalih simbola 0.3, 0.5 i 0.2. Pošto je poslato *c* onda su granice intervala:
[0.1547952+0.00003315x0.55x0.8, 0.1547952+0.00003315x0.55)=[0.154809786, 0.1548134325). Širina intervala je svedena na: 0.0000036465.

Konačno u poslednjem koraku kodira se sam terminacioni karakter koji se pojavljuje sa vjerovatnoćom 0.5 pa su granice intervala (gornja se ne mijenja):
[0.154809786+0.0000036465/2, 0.1548134325)=[0.15481160925, 0.1548134325). Širina intervala je svedena na: 0.00000182325.

Obije granice se pretvaraju u binarne brojeve čijih je prvih 17 decimala 00100111101000011. Treba uočiti da naredne tri cifre 100 sigurno pripadaju ovom intervalu pa je dovoljno priključiti prvu cifru. 001001111010000111

4.5. Izvršiti aritmetičko kodiranje binarnog koda sa simbolima (0,1). Terminacioni simbol je $. Vjerovatnoća terminacionog simbola za prvi karakter je 0.1 a za svaki naredni raste za po 0.05. Za prvi karakter se može pretpostaviti da su vjerovatnoće karaktera (0,1) iste. Za svaki naredni se vrši ažuriranje na osnovu broja karaktera koji su se do sada pojavili u sekvenci (koristiti Laplasovo pravilo za ažuriranje uvedeno na predavanjima). Kodirati poruku **011110001$**.

Rješenje: Prvi simbol koji je poslat je 0 pretpostavimo da to redukuje interval na zonu [0,0.45). Vjerovatnoća terminacionog karaktera se sada penje na 0.15 a vjerovatnoće simbola '0' i '1' su respektivno: (1-0.15)x2/3=0.5667 i (1-0.15)/3=0.2833. Kako je poslata jedinica tako su se granice intervala suzile na:

[0+0.5667x0.45,0+0.85x0.45)=[0.2550,0.3825) širina intervala je 0.1275

Vjerovatnoća terminacionog karaktera je sada 0.2, a po 0.4 su vjerovatnoće '0' i '1' simbola:

[0.2550+0.4x0.1275,0.2550+0.8x0.1275)=[0.3060,0.3570) a širina intervala je redukovana na 0.051.

Za naredni simbol vjerovatnoća terminacionog karaktera raste na 0.25 a vjerovatnoće simbola '0' i '1' su: 0.25 i 0.50 respektivno. Pošto šaljemo simbol '1' odabrali smo interval:
[0.3060+0.0510x0.25,0.3060+0.0510x0.75)=[0.31875,0.34425) širina intervala je 0.0255.

Za naredni simbol vjerovatnoća pojavljivanja terminacionog karaktera raste na 0.30 a vjerovatnoće simbola '0' i '1' su respektivno 0.175 i 0.525 a pošto šaljemo '1' dobijamo:
[0.3232125, 0.3366) dok se širina intervala smanjuje na 0.0133875.
U ovom koraku je vjerovatnoća terminacionog simbola porasla na 0.35 dok su vjerovatnoće simbola '0' i '1' respektivno 0.13 i 0.52. Pošto šaljemo 0 imamo interval sveden na:
[0.3232125, 0.324952875) a interval je sužen na 0.001740375.
Sada je vjerovatnoća terminacionog simbola porasla na 0.4 a vjerovatnoće '0' i '1' su redom 0.2 i 0.4. Pošto se šalje simbol '0' selektovali smo prvi interval pa dobijamo interval:
[0.3232125, 0.323560575) a interval se skuplja na 3.48075×10^{-4}.
U narednom koraku vjerovatnoća terminacionog simbola raste na 0.45. Vjerovatnoće simbola '0' i '1' su respektivno su 33/140 i 44/140. Šalje se 0 a to znači da smo selektovali interval
[0.3232125, 0.32329454625) koji je sada sužen na $8.2046\ 10^{-5}$.
U narednom koraku vjerovatnoća terminacionog simbola raste na 0.5. Vjerovatnoće simbola '0' i '1' su iste i iznose 0.25. Pošto se šalje simbol '1' to znači da je selektovan interval
[0.3232330115625, 0.323253523125) dok je širina intervala $2.05116\ 10^{-5}$.
Konačno u poslednjem intervalu šaljemo terminacioni simbol koji se pojavljuje sa vjerovatnoćom 0.55 tako da smo selektovali interval:
[0.32324429292188, 0.323253523125)

Ovim je interval sužen na $1.1281 \cdot 10^{-5}$.

Što se tiče binarne poruke koja se treba poslati uočljivo je da je 16 prvih bita koji kodiraju početak intervala istih kao 16 bita koji kodiraju kraj intervala pa stoga se sigurno šalje tih šesnaest bita. Nakon toga imamo 0 i 1 za početak i kraj što nije dovoljno da donesemo odluku o konkretnom slanju ali nakon dva bita 00 i 10 možemo zaključiti da je kombinacija 01 sigurno u intervalu tako da nju možemo poslati da zaključimo kodnu sekvencu.

Početak intervala	[0101001011000000 0010]
Kraj intervala	[0101001011000000 1011]
Poslata sekvenca	[0101001011000000 01]

MATLAB posjeduje funkcije za aritmetičko kodiranje koje nažalost rade uglavnom samo za slučaj fiksnih vjerovatnoća. Primjer formirajmo sekvencu sa 50 jedinica, 50 dvojki i 400 trojki

```
seq = repmat([3 3 1 3 3 3 3 3 2 3],1,50);
```

Dajemo procente pojavljivanja pojedinih simbola

```
counts = [10 10 80];
```

vršimo formiranje koda na osnovu sekvence i procenata formiranja koda

```
code = arithenco(seq,counts);
```

a zatim dekodiramo (potreban nam je kod, procenti i dužina sekvence)

```
dseq = arithdeco(code,counts,length(seq));
```

Ako provjerimo dužinu kodne riječi

```
length(code)
```

i dužinu koda 470 bita

```
length(dseq)
```

dobijamo da je ušteda oko 6% (jer je `length(dseq)=500`).

Ako imamo pogrešnu procjenu vjerovatnoća:

```
counts = [10 15 75];
code = arithenco(seq,counts);
dseq = arithdeco(code,counts,length(seq));
length(code)
477
length(dseq)
500
```

Dobićemo da je ušteda u kodiranju 23/500=4.6%.

4.6. Dati su kodovi (a)-(e). Da li su ovo jednoznačno dekodabilni kodovi i da li su u pitanju trenutni kodovi.

S	(a)	(b)	(c)	(d)	(e)
s_1	00	0	0	0	0
s_2	01	100	10	100	10
s_3	10	110	110	110	110
s_4	11	111	111	11	11

Za date kodove provjeriti važenje Kraftove nejednakosti.

Rješenje: Kraftova nejednakost glasi:

$\sum_{i=1}^{m} D^{-l_i} \le 1$ gdje je D broj simbola izlaznog alfabeta dok je l_i dužina kodnih riječi sa kojima se kodiraju simboli alfabeta.

Svih pet kodova su binarni sa D=2.
(a) Za prvi kod
$4 \cdot \frac{1}{2^2} = 1 \le 1$ Dakle kod bi mogao biti jednoznačno dekodabilan (i jeste).
(b)
$\frac{1}{2} + 3 \cdot \frac{1}{2^3} = \frac{7}{8} \le 1$ Dakle kod bi mogao biti jednoznačno dekodabilan (i jeste).
(c)
$\frac{1}{2} + \frac{1}{4} + 2 \cdot \frac{1}{2^3} = 1 \le 1$ Dakle kod bi mogao biti jednoznačno dekodabilan (i jeste).
(d)
$\frac{1}{2} + \frac{1}{4} + 2 \cdot \frac{1}{2^3} = 1 \le 1$ Dakle kod bi mogao biti jednoznačno dekodabilan (i jeste premda nije trenutan i veoma je komplikovan za dekodiranje).
(e)
$\frac{1}{2} + 2 \cdot \frac{1}{4} + \frac{1}{2^3} = \frac{9}{8} > 1$ Čime dokazujemo da kod zasigurno nije jednoznačno dekodabilan.

4.7. Na prethodnom času smo dokazali Kraftovu nejednakost za trenutne jednoznačno dekodabilne kodove. Ova nejednakost važi kod svih jednoznačno dekodabilnih kodova. Dokazati!!! **Napomena.** Ovaj dokaz je originalno dao Makmilan.

Rješenje: Pretpostavimo da imamo D-arni alfabet. Pođimo od relacije:

$$K^n = \left(\sum_{i=1}^{q} D^{-l_i} \right)^n = (D^{-l_1} + D^{-l_2} + ... + D^{-l_q})^n$$

U sumi postoji q^n članova koji su oblika D^{-k} gdje je:

$$k = l_{i_1} + l_{i_2} + ... + l_{i_n}$$

k je iz skupa prirodnih brojeva u intervalu od n do $nl_{\max}$. Neka je broj ovih sabiraka koji imaju oblik D^{-k} N_k tada se K^n može zapisati:

$$K^n = \sum_{k=n}^{nl_{\max}} N_k D^{-k}$$

Za jednoznačno dekodabilan kod N_k ne može da bude veće od broja različitih sekvenci dužine n od D različitih kodnih simbola to jest od D^k . Odavde slijedi:

$$K^n \le \sum_{k=n}^{nl_{\max}} D^k D^{-k} = nl_{\max} - n + 1 < nl_{\max}$$

Data relacija mora da važi za svako n (npr. za n koje teži beskonačnosti) to znači da K mora da bude manje od 1:

$$K = \sum_{i=1}^{q} D^{-l_i} \le 1$$

4.8. Posmatrajmo zadatak 3.4 i pojavu slova u engleskoj abecedi. Izvršite Huffmanovo kodiranje i odrediti prosječnu dužinu kodne riječi.

A	8.38%	B	1.67%	C	3.18%	D	4.09%	E	12.21%	F	2.10%
G	1.90%	H	4.47%	I	7.57%	J	0.27%	K	0.91%	L	4.04%
M	2.45%	N	7.41%	O	7.40%	P	2.16%	Q	0.10%	R	6.25%
S	6.84%	T	8.91%	U	2.67%	V	1.11%	W	1.91%	X	0.17%
Y	1.71%	Z	0.13%								

Dva najređa simbola u engleskoj abecedu su Q i Z. Q ćemo kodirati sa 0 a Z sa 1 i to će biti novi generički simbol za vjerovatnoćom 0.23%.

A	8.38%	B	1.67%	C	3.18%	D	4.09%	E	12.21%	F	2.10%
G	1.90%	H	4.47%	I	7.57%	J	0.27%	K	0.91%	L	4.04%
M	2.45%	N	7.41%	O	7.40%	P	2.16%	QZ(0,1)	0.23%	R	6.25%
S	6.84%	T	8.91%	U	2.67%	V	1.11%	W	1.91%	X	0.17%
Y	1.71%										

U narednom koraku najmanje vjerovatni simboli su QZ i X. Ponovo ćemo prvi kodirati sa 0 a drugi sa 1.

A	8.38%	B	1.67%	C	3.18%	D	4.09%	E	12.21%	F	2.10%
G	1.90%	H	4.47%	I	7.57%	J	0.27%	K	0.91%	L	4.04%
M	2.45%	N	7.41%	O	7.40%	P	2.16%	QZX(00,01,1)	0.40%	R	6.25%
S	6.84%	T	8.91%	U	2.67%	V	1.11%	W	1.91%	Y	1.71%

Sada obuhvatamo QZX i J:

A	8.38%	B	1.67%	C	3.18%	D	4.09%	E	12.21%	F	2.10%
G	1.90%	H	4.47%	I	7.57%	QZXJ(000,001,01,1)	0.67%	K	0.91%	L	4.04%
M	2.45%	N	7.41%	O	7.40%	P	2.16%	Y	1.71%	R	6.25%
S	6.84%	T	8.91%	U	2.67%	V	1.11%	W	1.91%		

Sada dodajemo K prethodnom skupu (K se kodira sa 1 a ostatak sa "prednjačećom" nulom):

A	8.38%	B	1.67%	C	3.18%	D	4.09%	E	12.21%	F	2.10%
G	1.90%	H	4.47%	I	7.57%	QZXJK(0000,0001,001,01,1)	1.58%	W	1.91%	L	4.04%
M	2.45%	N	7.41%	O	7.40%	P	2.16%	Y	1.71%	R	6.25%
S	6.84%	T	8.91%	U	2.67%	V	1.11%				

Ponovo na generički simbol QZXJK dodajemo V:

A	8.38%	B	1.67%	C	3.18%	D	4.09%
G	1.90%	H	4.47%	I	7.57%	QZXJKV(00000,00001,0001,001,01,1)	2.69%
M	2.45%	N	7.41%	O	7.40%	P	2.16%
S	6.84%	T	8.91%	U	2.67%	R	6.25%

E	12.21%	F	2.10%
W	1.91%	L	4.04%
Y	1.71%		

Sada su dva najmanje vjerovatna simbola B i Y:

A	8.38%	BY(0,1)	3.38%	C	3.18%	D	4.09%
G	1.90%	H	4.47%	I	7.57%	QZXJKV(00000,00001,0001,001,01,1)	2.69%
M	2.45%	N	7.41%	O	7.40%	P	2.16%
S	6.84%	T	8.91%	U	2.67%	R	6.25%
E	12.21%	F	2.10%				
W	1.91%	L	4.04%				

G i W su sada dva najmanje vjerovatna simbola:

A	8.38%	BY(0,1)	3.38%	C	3.18%	D	4.09%
H	4.47%	GW(0,1)	3.81 %	I	7.57%	QZXJKV(00000,00001,0001,001,01,1)	2.69%
M	2.45%	N	7.41%	O	7.40%	P	2.16%
S	6.84%	T	8.91%	U	2.67%	R	6.25%
E	12.21%	F	2.10%				
L	4.04%						

Sledeći par je P i F:

A	8.38%	BY(0,1)	3.38%	C	3.18%	D	4.09%
H	4.47%	GW(0,1)	3.81 %	I	7.57%	QZXJKV(00000,00001,0001,001,01,1)	2.69%
M	2.45%	N	7.41%	O	7.40%	PF (0,1)	4.26%
S	6.84%	T	8.91%	U	2.67%	R	6.25%
E	12.21%	L	4.04%				

Sledeća kombinacija je M i U:

A	8.38%	BY(0,1)	3.38%	C	3.18%	D	4.09%
H	4.47%	GW(0,1)	3.81 %	I	7.57%	QZXJKV(00000,00001,0001,001,01,1)	2.69%
R	6.25%	N	7.41%	O	7.40%	PF (0,1)	4.26%
S	6.84%	T	8.91%	L	4.04%	MU(0,1)	5.12%
E	12.21%						

Sledeći simboli koji se obuhvata su generički QZXJKV i C:

QZXJKVC(000000,000001,00001,0001,001,01,1)						5.87%	
A	8.38%	BY(0,1)	3.38%	E	12.21%	D	4.09%
H	4.47%	GW(0,1)	3.81 %	I	7.57%	MU(0,1)	5.12%
R	6.25%	N	7.41%	O	7.40%	PF (0,1)	4.26%
S	6.84%	T	8.91%	L	4.04%		

Naredna kombinacija uključuje parove generičkih simbola: BY i GW:

QZXJKVC(000000,000001,00001,0001,001,01,1)						5.87%	
BYGW (00, 01, 10, 11)						7.19%	
A	8.38%	L	4.04%	E	12.21%	D	4.09%
H	4.47%	T	8.91%	I	7.57%	MU(0,1)	5.12%
R	6.25%	N	7.41%	O	7.40%	PF (0,1)	4.26%
S	6.84%						

Naredna dva simbola su L i D:

QZXJKVC(000000,000001,00001,0001,001,01,1)							
BYGW (00, 01, 10, 11)						7.19%	
A	8.38%	LD (0,1)	8.13%	E	12.21%	S	6.84%
H	4.47%	T	8.91%	I	7.57%	MU(0,1)	5.12%
R	6.25%	N	7.41%	O	7.40%	PF (0,1)	4.26%

Naredna kombinacija je H i generički simbol PF:

QZXJKVC(000000,000001,00001,0001,001,01,1)							
BYGW (00, 01, 10, 11)						7.19%	
HPF (0, 10, 11)						8.73%	
A	8.38%	LD (0,1)	8.13%	E	12.21%	S	6.84%
O	7.40%	T	8.91%	I	7.57%	MU(0,1)	5.12%
R	6.25%	N	7.41%				

Naredna kombinacija je QZXJKV i MU:

QZXJKVCMU(0000000,0000001,000001,00001,0001,001,01,10,11)						10.99%	
BYGW (00, 01, 10, 11)						7.19%	
HPF (0, 10, 11)						8.73%	
A	8.38%	LD (0,1)	8.13%	E	12.21%	S	6.84%
O	7.40%	T	8.91%	I	7.57%	N	7.41%
R	6.25%						

Kombinaciji BYGW dodajemo R:

QZXJKVCMU(0000000,0000001,000001,00001,0001,001,01,10,11)						10.99%	
BYGWR (000, 001, 010, 011, 1)						13.44%	
HPF (0, 10, 11)						8.73%	
A	8.38%	LD (0,1)	8.13%	E	12.21%	S	6.84%
O	7.40%	T	8.91%	I	7.57%	N	7.41%

Sada idu dva simbola S i O:

QZXJKVCMU(0000000,0000001,000001,00001,0001,001,01,10,11)						10.99%	
BYGWR (000, 001, 010, 011, 1)						13.44%	
HPF (0, 10, 11)						8.73%	
A	8.38%	LD (0,1)	8.13%	E	12.21%	SO (0,1)	14.24%
N	7.41%	T	8.91%	I	7.57%		

Ponovo obuhvatamo dva simbola N i I:

QZXJKVCMU(0000000,0000001,000001,00001,0001,001,01,10,11)						10.99%	
BYGWR (000, 001, 010, 011, 1)						13.44%	
HPF (0, 10, 11)						8.73%	
A	8.38%	LD (0,1)	8.13%	E	12.21%	SO (0,1)	14.24%
T	8.91%	NI (0,1)	14.98%				

Naredna kombinacija je A i LD:

QZXJKVCMU(0000000,0000001,000001,00001,0001,001,01,10,11)							10.99%	
BYGWR (000, 001, 010, 011, 1)							13.44%	
HPF (0, 10, 11)							8.73%	
ALD (0, 10, 11)							16.51%	
T	8.91%	NI (0,1)	14.98%	E	12.21%		SO (0,1)	14.24%

Ponavljamo proceduru sa simbol T i generički simbol HPF:

QZXJKVCMU(0000000,0000001,000001,00001,0001,001,01,10,11)						10.99%	
BYGWR (000, 001, 010, 011, 1)						13.44%	
THPF (0, 10, 110, 111)						17.64%	
ALD (0, 10, 11)						16.51%	
SO (0,1)	14.24%	NI (0,1)	14.98%	E	12.21%		

Sledeća kombinacija je generički simbol QZXJKVCMU i najvjerovatnije slovo engleske abecede E:

QZXJKVCMUE(00000000,00000001,0000001, 000001,00001,0001,001,010,011,1)						23.20%	
BYGWR (000, 001, 010, 011, 1)						13.44%	
THPF (0, 10, 110, 111)						17.64%	
ALD (0, 10, 11)						16.51%	
SO (0,1)	14.24%	NI (0,1)	14.98%				

Sada su svi preostali simboli generički. Najmanje vjerovatni su BYGWR i SO:

QZXJKVCMUE(00000000,00000001,0000001, 000001,00001,0001,001,010,011,1)	23.20%
BYGWRSO (0000, 0001, 0010, 0011, 01, 10, 11)	27.68%
THPF (0, 10, 110, 111)	17.64%
ALD (0, 10, 11)	16.51%
NI (0,1)	14.98%

Nova kombinacija je ALD sa NI:

QZXJKVCMUE(00000000,00000001,0000001, 000001,00001,0001,001,010,011,1)	23.20%
BYGWRSO (0000, 0001, 0010, 0011, 01, 10, 11)	27.68%
THPF (0, 10, 110, 111)	17.64%
ALDNI (00, 010, 011, 10, 11)	31.47%

Ostalo je još samo četiri kombinacije odnosno 3 koraka u algoritmu. Sada ćemo obuhvatiti QZXJKVCMUE sa THPF. Nakon toga ćemo obuhvatiti BYGWRSO sa ALDNI i dobijamo (dva koraka algoritma su obuhvaćena odjednom:

QZXJKVCMUETHPF (0000000000, 0000000001,00000001, 0000001,000001,00001,0001,0010,0011,01, 10, 110, 1110, 1111)	40.84%
BYGWRSOALDNI (00000, 00001, 00010, 00011, 001, 010, 011, 100, 1010, 1011, 110, 111)	59.16%

Konačno dobijamo sledeću kodnu tabelu:

Q	0000000000(10)	Z	0000000001(10)	X	000000001(9)
J	00000001(8)	K	0000001(7)	V	000001(6)
C	00001(5)	M	00010(5)	U	00011(5)
E	001(3)	T	010(3)	H	0110(4)
P	01110(5)	F	01111(5)	B	100000(6)
Y	100001(6)	G	100010(6)	W	100011(6)
R	1001(4)	S	1010(4)	O	1011(4)
A	1100(4)	L	11010(5)	D	11011(5)
N	1110(4)	I	1111(4)		

Prosječna dužina kodne riječi je:

$10 \cdot 0.0023+9 \cdot 0.0017+8 \cdot 0.0027+7 \cdot 0.0097+6 \cdot 0.0830+5 \cdot 0.2069+4 \cdot 0.4832+3 \cdot 0.2112=$
4.1588bita/simbolu

Vidimo da smo ostvarili unapređenje od 0.7bita/simbolu u odnosu na poluintuitivnu shemu koja je razmatrana u zadatku 3.7.

4.9. Posmatrajte tabelu datu na drugoj stranici dodatka na trećem času. Pokazati važenje asimptotske ekviparticione osobine za p=0.1, p=0.2 i p=0.4 i n=10.

Rješenje: Za p=0.1

Niz	Vjerovatnoća	Broj kombinacija	Ukupna vjerovatnoća
0000000000	$1x10^{-10}$	1	$1x10^{-10}$
0000000001	$9x10^{-10}$	10	$9x10^{-9}$
0000000011	$8.1\ x10^{-9}$	45	$3.645x10^{-7}$
0000000111	$7.29x10^{-8}$	120	$8.75x10^{-6}$
0000001111	$6.561x10^{-7}$	210	$1.378\ x10^{-4}$
0000011111	$5.9x10^{-6}$	252	0.00149
0000111111	$5.314x10^{-5}$	210	0.01116
0001111111	$4.783x10^{-4}$	120	0.0574
0011111111	0.0043	45	0.1937
0111111111	0.0387	10	0.3874
1111111111	0.3487	1	0.3487

Vidimo da je tipična sekvenca ona sa 9 jedinica i da se pojavljuje sa ukupnom vjerovatnoćom 0.3874. Entropija ovog događaja je $H(X)$=0.459, $nH(X)$=4.59 pa je $2^{-nH(X)}$= 0.03874 što je praktično isto kao ono što piše u tabeli.

Za dodatnu provjeru posmatrajmo da li je moguće ostvariti neku kompresiju korišćenjem najprimitivnijih oblika kodiranja na ovom kodu. Sekvenci u tipičnom skupu je 10, ovaj skup možemo proširiti sa najvjerovatnijim događajem sa svim jedinicama koji ima vjerovatnoću 0.3487 i pet događaja iz skupa sa osam jedinica sa ukupnom vjerovatnoćom 5x0.0043=0.0215. Ukupna vjerovatnoća ovih 16 događaja je 0.3874+0.3487+0.0215=0.7576. Ovih 16 simbola se mogu kodirati sa 4 bita (plus jedan bit prefiksa) dok se preostalih 1008 simbola mogu kodirati sa 10 bita (plus jedan bit prefiksa) tako da je prosječna dužina kodne riječi:

$$0.7576x5+0.2424x11=6.4544 \text{ bita}$$

Dakle, u stanju smo da na ovom veoma primitivnom principu prištedimo 35.45% u kodiranju.

Ponovimo proceduru za p=0.2.

Niz	Vjerovatnoća	Broj kombinacija	Ukupna vjerovatnoća
0000000000	$1.024x10^{-7}$	1	$1.024x10^{-7}$
0000000001	$4.096x10^{-7}$	10	$4.096x10^{-6}$
0000000011	$1.6384\ x10^{-6}$	45	$7.373x10^{-5}$
0000000111	$6.554x10^{-6}$	120	$7.864x10^{-4}$
0000001111	$2.6214x10^{-5}$	210	0.0055
0000011111	$1.0486x10^{-4}$	252	0.0264
0000111111	$4.1943x10^{-4}$	210	0.0881
0001111111	0.0017	120	0.2013
0011111111	0.0067	45	0.3020
0111111111	0.0268	10	0.2684
1111111111	0.1074	1	0.1074

Ponovo imamo da je sekvenca sa 2 nule i 8 jedinica tipična. Entropija ovog skupa je $H(X)$=0.7219 dok je $nH(X)$=7.219. Tada je $2^{-nH(X)}$=0.0067 i potpuno tačno odgovara vjerovatnoći pojavljivanja pojedinačne poruke iz tipičnog skupa! Analizirajmo mogućnost kompresije. Poruka u tipičnom skupu je 45 do najbližeg stepena dvojke je 19 (2^6=64). Dodajmo svih 10 sekvenci sa jednom nulom i jednu sekvencu sa svim jedinicama kao i 8 sekvenci sa 3 nule. Ukupna vjerovatnoća ovog visokovjerovatnog skupa je sada:

$$0.3020+0.2684+0.1074+8x0.0017=0.6912$$

Za kodiranje ove sekvence je potrebno 6 bita plus jedan bit kao prefiks dok je za ostale simbole sa ukupnom vjerovatnoćom 0.3088 potrebno 10 bita i jedan kao prefiks:

$$0.6912x7+0.3088=8.2354 \text{ bita}$$

Ušteda je sada manja nego u slučaju kada je tipična sekvenca imala manji broj članova i kada je bila karakterističnija po svom obliku. Vidjećemo da u narednom slučaju kada p raste i kada se primiče p=0.5 dolazi do smanjivanja uštede koja se dobija putem asimptotske ekviparticione osobine odnosno u opštem slučaju putem kodiranja kada su simboli sa uniformnijim raspodjelama dobijamo manju dobit.

Posmatrajmo tabelu za p=0.4.

Niz	Vjerovatnoća	Broj kombinacija	Ukupna vjerovatnoća
0000000000	$1.0486x10^{-4}$	1	$1.0486x10^{-4}$
0000000001	$1.5927x10^{-4}$	10	0.001573
0000000011	$2.359\ x10^{-4}$	45	0.010617
0000000111	$3.5389x10^{-4}$	120	0.042467
0000001111	$5.3084x10^{-4}$	210	0.111476
0000011111	$7.9626x10^{-4}$	252	0.200658
0000111111	0.00119439	210	0.250823
0001111111	0.0017916	120	0.214991
0011111111	0.0026874	45	0.120932
0111111111	0.0040311	10	0.040311
1111111111	0.0060466	1	0.0060466

Imamo da je tipična sekvenca sa 4 nule i 6 jedinica koja se pojavljuje sa ukupnom vjerovatnoćom 0.250823. Entropija ovog skupa je $H(X)=H(0.4)$=0.971 a dalje imamo $nH(X)$=9.71 pa je $2^{-nH(X)}=0.00119$ a ovo se tek na osmu cifru razlikuje od vjerovatnoće pojedinačnog člana tipičnog skupa. Dopunimo ovaj skup sa još četiri najvjerovatnija elementa (sve jedinice i 3 sekvence sa 9 jedinica) da bi skupili so 256=2^8 sekvenci. Vjerovatnoća ove visokovjerovatne sekvence je

$$0.250823+0.0060466+3x0.0040311=0.2689629$$

Ako kodiramo ove sekvence sa 8+1 bit (1 bit za predznak) a ostale sa 10+1 bit dobijamo da je prosječna dužina kodne riječi 10.462 bita po sekvenci od 10 simbola čime ne ostvarujemo nikakav dobitak već zapravo gubitak. Premda ćemo sofisticiranijim tehnikama kodiranja biti u stanju da ostvarimo dobitak ovo nam pokazuje da kod alfabetova sa ujednačenijim vjerovatnoćama simbola ili kombinacija simbola ne možemo da ostvarimo veliki dobitak kod kodiranja.

4.10. Posmatrati Markovljev sistem dat grafom tranzicije na drugom času. Odrediti entropiju sistema i uporediti je sa entropijom sistema bez memorije.

Rješenje: Kako su vjerovatnoće simbola u stacionarnom stanju 3/11, 4/11 i 4/11 to je entropija simbola (sistema):

```
p=[3/11 4/11 4/11];
Hx=-sum(p.*log2(p))
Hx =1.5726 bita/simbolu
```

Entropiju združenog događaja pojavljivanja dva uzastopna simbola možemo sračunati kao:

$$H(X,Y)=H(X)+H(Y|X)$$

$$H(Y|X)=p(a)H(Y|X=a)+p(b)H(Y|X=b)+p(\mathrm{c})H(Y|X=c)=3\cdot 1.585/11+4\cdot 1.5/11+4\cdot 1.5/11=1.5232$$

Dakle, $H(X,Y)=3.0958$ pa je entropija po jednom simbolu jednaka 1.5479bita/simbolu odnosno nešto je manja od $H(X)$.

4.11. Posmatran je dokumenat na našem jeziku koji sadrži sledeću raspodjelu simbola:

a	4694	b	589	v	1320	g	704	d	1463
đ	94	e	3409	ž	179	z	671	i	3734
j	1433	k	1505	l	1151	lj	166	m	1179
n	2324	nj	249	o	3517	p	1118	r	2143
s	1806	t	1729	ć	354	u	1582	f	159
h	210	c	439	č	390	dž	15	š	383

Odraditi što bi to mogla biti tipična sekvenca u slučaju da je n=100. Neka je kodiranje datog fajla obavljeno na sledeći način. Izdvojeno je K simbola koji se najčešće pojavljuju i kodirani su sa po $\log_2 K$ bita plus 0 kao prefiks, dok je ostatak kodiran po ASCII kodu i 1 kao prefiks. Odrediti koliko bita je potrebno za kodiranje čitavog fajla ako je K=2, 4, 8, 16.

4.12. Posmatrajte kodove iz tabele dolje. Osmislite jednoznačan postupak dekodiranja ovih poruka.

X	**Kod 2**	**Kod3**
1	10	0
2	00	10
3	11	110
4	110	111

Rješenje: Započećemo sa kodom 3 koji je koma kod i koji se može dekodirati putem stabla bez problema. Naime, odluka se odnosi na osnovu pojave nule ili trećeg uzastopnog bita bez nule.

Kod 2 koji nije trenutan pa je stoga znatno složeniji za dekodiranje. Ako se prvo primi 1 pa 0 donosimo odluku da je u pitanju simbol 1, ako se primi 0 očekuje se još jedna nula koja se zatim

dekodira kao simbol 2. Ako se primi 11 dolazi do problema jer ako se nakon toga primi 0 riječ je o simbolu 4 dok ako se ponovo primi 1 moramo čekati do narednog simbola. Ako primimo neparan broj jedinica prije prve nule recimo 9, zaključujemo da 4 puta primamo simbol 3 (kodiran sa 11) a da zatim primamo 1 (simbol 1). Ako međutim primimo paran broj jedinica prije prve nule npr. 10 znamo da je četiri puta poslat simbol 2 (kodiran kao 00) zatim dolazimo do simbola sa kojim imamo dilemu ali nastavljamo proceduru kao da smo na početku primili kombinaciju 110.

4.13. Odrediti način jednoznačnog dekodiranja Grayovog koda.

Rješenje: Grayov kod se može zapisati kao $b_1=a_1$ i $b_i=a_i \oplus a_{i-1}$ za $i>1$. Dok je suprotna operacija za prvi simbol trivijalna $a_1=b_1$. Sada treba odrediti b_i za $i>1$. To se može odrediti na sljedeći način odredimo rezultat operacije $b_i \oplus a_{i-1}$

$$b_i \oplus a_{i-1}=a_i \oplus a_{i-1} \oplus a_{i-1}$$

$a_{i-1} \oplus a_{i-1}$ je uvijek jednako 0 jer je $0 \oplus 0=0$ i $1 \oplus 1=0$ onda se predmetna relacija svodi na:

$$b_i \oplus a_{i-1}=a_i \oplus 0$$

$$a_i \oplus 0=a_i$$

Dakle, dobijamo $a_i=b_i \oplus a_{i-1}$ za $i>1$ jer smo rekurzivno prethodno već odredili a_{i-1} počevši od a_1.

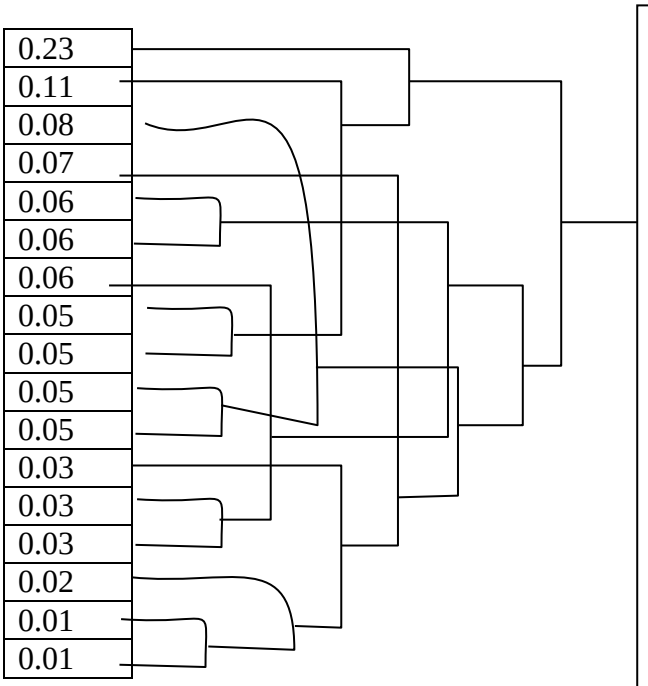
4.14. Na osnovu leme 4.4.1 dokazati optimalnost postupka kodiranja kod Huffmanovog koda. **Napomena:** Operacija se može dokazati indukcijom polazeći od binarnog seta simbola.

Dokaz leme: 1. Ako je suprotno odnosno ako je $p_j>p_k$ može da znači $l_j>l_k$ onda kod nije optimalan jer se prostom zamjenom kodnih riječi za simbole sa vjerovatnoćama p_j i p_k dobija kraći kod.
2. Ako ovo nije zadovoljeno odnosno ako je jedna kodna riječ duža za 1 bit onda bi najdužu kodnu riječ mogli da skratimo za jedan bit jer skraćivanjem za jedan bit dobijamo simbol koji je i dalje list pošto ako je roditelj najduže kodne riječi roditelj još nekom simbolu a onda dolazimo do suprotnosti da je naša kodna riječ najduža. Dakle, ako je optimalni trenutni kod moguće na jednom simbolu dalje skratiti za jedan bit dolazimo do koda koji je "optimalniji" što je u suprotnosti sa postavkom leme.
3. Isto kao prethodno najduže koda riječ mora da ima barem jednu partnersku kodnu riječ sa kojom dijeli roditelja inače će ponovo biti moguće skratiti najdužu riječ za jedan odnosno pronaći optimalniji kod od optimalnog.

Ako pogledate postupak dobijanja Huffmanovog koda očigledno je da taj kod ispunjava uslove koji su navedeni u lemi.

4.15. Odrediti Huffmanov kod i prosječnu dužinu kodne riječi ako su vjerovatnoće pojedinih simbola 0.23, 0.11, 0.08, 0.07, 0.06, 0.06, 0.06, 0.05, 0.05, 0.05, 0.05, 0.03, 0.03, 0.03, 0.02, 0.01, 0.01. Ponoviti proceduru ako se Huffmanovo kodiranje provodi samo na 9 najpojavljivanijih simbola dok se ostali kodiraju kodom fiksne dužine.

Rješenje:

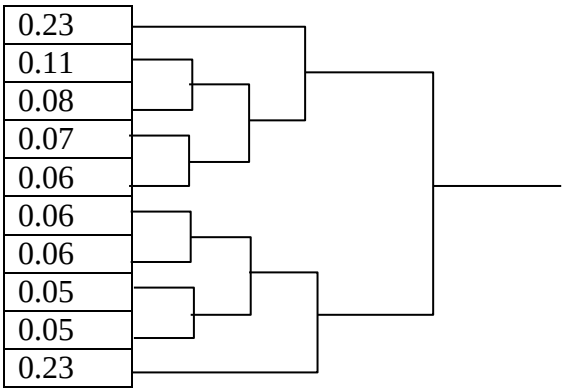


Ako smo svaku donju liniju kodirali sa 0 a gornju sa 1 dobijamo sljedeće kodove za pojedine simbole:

0.23	11
0.11	101
0.08	0011
0.07	0001
0.06	0111
0.06	0110
0.06	0101
0.05	1001
0.05	1000
0.05	00101
0.05	00100
0.03	00001
0.03	01001
0.03	01000
0.02	000001
0.01	0000001
0.01	0000000

Prosječna dužina kodne riječi je:
$0.23 \cdot 2+0.11 \cdot 3+0.43 \cdot 4+$
$0.19 \cdot 5+0.02 \cdot 6+0.02 \cdot 7=$
3.72bita/simbolu

8 najređe pojavljivanih simbola ima ukupnu vjerovatnoću 0.23. Izvršimo kodiranje ovog sistema:



Ako smo svaku donju liniju kodirali sa 0 a gornju sa 1 dobijamo sledeće kodove za pojedine simbole:

0.23	11
0.11	1011
0.08	1010
0.07	1001
0.06	1000
0.06	0111
0.06	0110
0.05	0101
0.05	0100
0.23	00

Posljednji generički simbol predstavlja zapravo 8 simbola ukupne vjerovatnoće 0.23 koje ćemo kodirati kodom fiksne dužine a 00 je prefiks ovog koda:

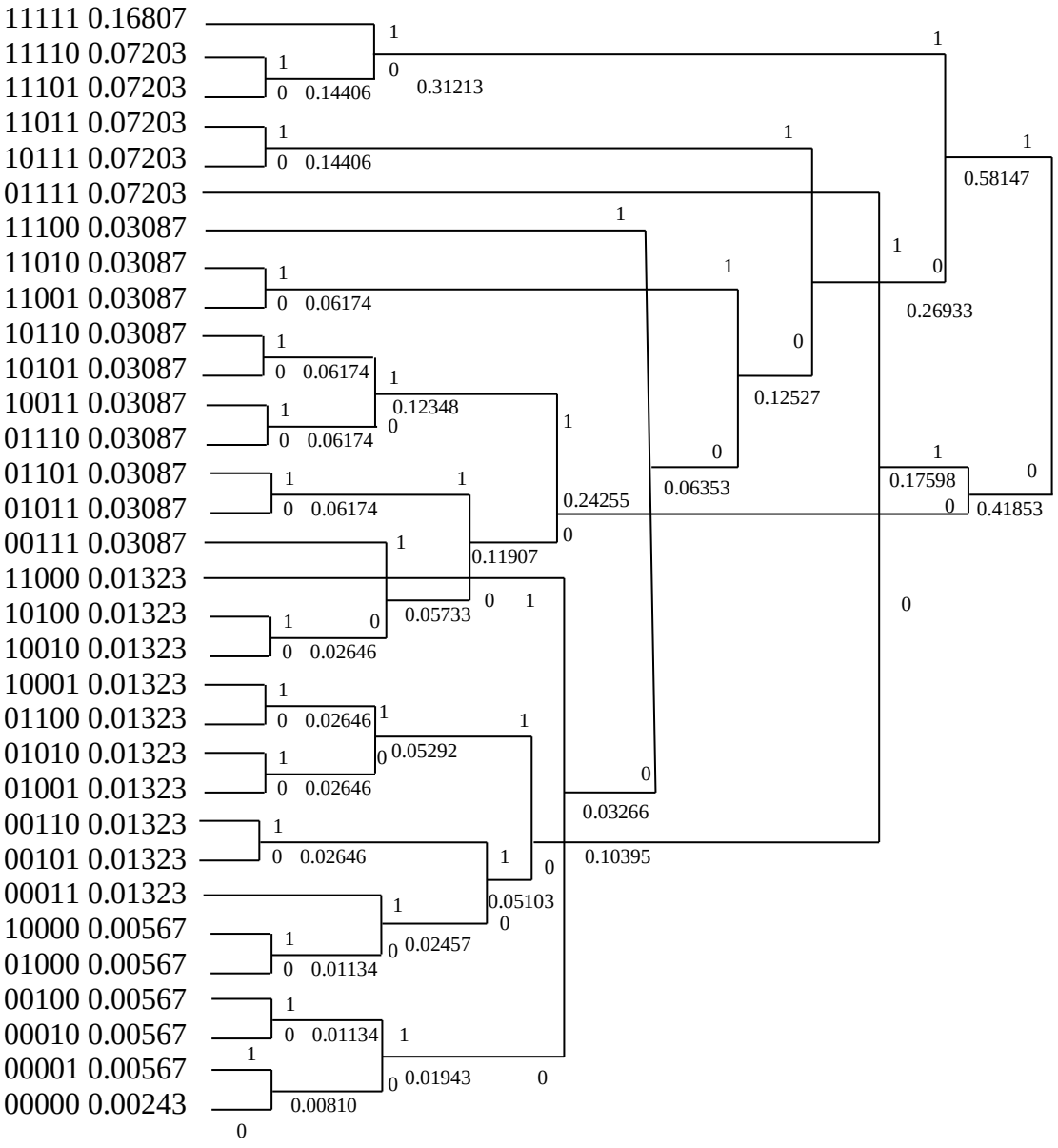
0.05	00 000	0.05	00 001
0.03	00 010	0.03	00 011
0.03	00 100	0.02	00 101
0.01	00 110	0.01	00 111

Prosječna dužina kodne riječi je: $0.23 \cdot 2+0.23 \cdot 5+0.54 \cdot 4=3.77$bita/simbolu. Dakle, izgubili smo efikasnost u smislu 0.05 bita/simbolu ali smo dobili skraćivanje najdužih kodnih riječi. Naime, ako dolazi do greške na nekom bitu dugačke kodne riječi praksa je pokazala da se ovakve greške mnogo teže ispravljaju nego one kraće.

4.16. Šalju se n=5 i.i.d. simbol sa vjerovatnoćom jedinice p=0.7 i nule q=0.3. Kreirati Huffmanov kod za dati skup riječi izvornog alfabeta i odrediti prosječnu dužinu kodne riječi. Ponoviti proceduru ako se Huffmanovo kodiranje obavlja na samo 8 najčešće pojavljivanih kodnih riječi.

Rješenje: Prvo objasnim ključne elemente zadatka. Kombinacija sa 5 bita ima ukupno 2^5=32. Od njih sa svim jedinicama 11111 je jedna i ima vjerovatnoću 0.7^5=0.16807. Ukupno 5 kombinacija ima sa jednom nulom i to su 11110, 11101, 11011, 10111, 01111. Vjerovatnoća svih ovih kombinacija je p^4q=0.07203. Ukupno deset kombinacija ima sa tri jedinice i dvije nule sa vjerovatnoćom svake od njih od p^3q^2=0.03087, takođe deset je kombinacija sa dvije jedinice i tri

nule sa vjerovatnoćom svake pojedinačne od njih p^2q^3=0.01323. Kombinacija sa jednom jedinicom ima pet i one imaju vjerovatnoću pq^4=0.00567 i konačno samo je jedna kombinacija sa 5 nula i njena vjerovatnoća je 0.00243. Sada se postupak provodi na ovih 32 generička simbola sa ciljem da se dobije smanjenje prosječne dužine kodne riječi sa početnih 5 bita po simbolu (odnosno 1 bit po bitu) na manji iznos.



Prikažimo sada pojedine kodne riječi:

Riječ	Kod
11111	111
11110	1101
11101	1100
11011	1011
10111	1010
01111	011
11100	10001
11010	10011
11001	00111
10110	00110
10101	00101

10011	00100
01110	00011
01101	00010
01011	00001
00111	100001
11000	000001
10100	000000
10010	010111
10001	010110
01100	010101
01010	010100
01001	010011
00110	010010
00101	010001
00011	01001
10000	0100001
01000	0100000
00100	10000011
00010	10000010
00001	10000001
00000	10000000

Dvije kodne riječi su kodirane sa 3 bita, 4 sa četiri i tako dalje. Prosječna dužina kodne riječi koja je potrebna za kodiranje pet bita je jednaka:

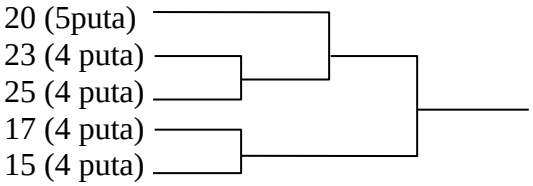
3x(0.16897+0.07203)+4x4x0.07203+ 5x (9x0.03087+0.01323)+6x(0.03087+9x0.01323)+
7x2x0.00567+8x(3x0.00567+0.00243)=
=0.72300+1.15248+1.45530+0.89964+0.07938+0.15552=4.4653. Ovo daje prosječnu dužinu kodne riječi po jednom simbolu od 0.8931bita/simbolu.

4.17. (a) Dati proceduru za RLE kodiranje i dekodiranje. (b) Posmatrajte funkciju $y(n)$=round(20+5sin(0.2πn)) gdje funkcija round znači zaokruživanje na najbliži cijeli broj. Vrijednosti n su u granicama od 0 do 20. Kreirati Huffmanov kod za datu sekvencu i kombinaciju diferencijalni kod pa zatim Huffmanov kod. Odrediti prosječne dužine kodne riječi u oba slučaja.

Rješenje:
a) Procedura kodiranja i dekodiranja je data MATLAB programom u Glavi 4.5.

b) Sekvenca od 21 element je:

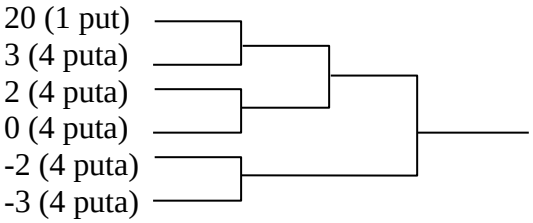
20	23	25	25	23	20	17
15	15	17	20	23	25	25
23	20	17	15	15	17	20



14, 15 i 20 se kodira sa 2 bita a 23 i 25 sa 3 bita. Znači ukupno nam je potrebno 50 bita za kodiranje ove sekvence. Provjerimo da li se dobija nešto bolje u predmetnom slučaju primjenom prvo diferencijalnog koda.

Diferencijalni kod je:

20	3	2	0	-2	-3	-3
-2	0	2	3	3	2	0
-2	-3	-3	-2	0	2	3



-2 i -3 se kodiraju sa 2 bita dok se ostali simboli kodiraju sa 3 bita. Ukupno je potrebno 55 bita za kodiranje poruke odnosno primjena diferencijalnog kodiranja u ovom slučaju pogoršava kvalitet kodiranja. Zbog čega je to slučaj?

4.18. Kodiranje se obavlja u ternarnom kodu *a, b, c*. Odrediti LZ kod sekvence:

ab aba ba cb abc bb cc aa ab bc cb ab ac

a) Pretpostaviti da je rječnik na početku prazan i da se nakon 8 upisa rječnik resetuje odnosno prazni.

b) Postoji rječnik koji posjeduje simbole *a, b, c*. Rječnik se ne resetuje u okviru ove sekvence. Zadatak odraditi korak po korak sa detaljnim opisom rječnika.

Rješenje:

a) Pretpostavimo da je rječnik na početku prazan. Kodiramo simbol *a* a u rječnik upisujemo na poziciju

001 *a* Do sada kodirani dio sekvence je 000*a*

Zatim kodiramo *b* koje do sada ne postoji u rječniku i u rječnik upisujemo:

010 *b* Ovaj simbol je 000*b*

Zatim kodiramo *ab* (*a* već postoji u rječniku dok je *b* sufiks). Upis u rječnik je:

011 *ab* a kod je 001*b*

Zatim se kodira *aba* (*ab* se prefiks dok je *a* sufiks):

100 *aba* kod je 011*a*

Zatim se kodira *c* koje se do sada nije pojavljivalo u sekvenci:

101 *c* kod je 000*c*

Naredna kombinacija je *ba* koja su upisuje u rječnik na poziciji 6:

110 *ba* i kodira kao 010*a*

Sledeća kombinacija je *bc* koje se upisuje u rječnik na poziciji 7:

111 *bc* i kodira kao 010*c*

Naredna kombinacija *bb* koje se upisuju u prazan rječnik na poziciji 001 a sastoji se od prethodnog *b* i novododatog *b*:

001 *bb* kodira se 010*b* (nakon ovoga je prazno ono što je bilo upisano u rječniku, može se eventualno početi i od kodiranja samog *b* ali smo na ovaj način uštedili nekoliko bitova u kodiranju)

Sledeća kombinacija je *c* i upisuje se u rječnik na poziciju 010

010 *c* 000*c*

Nakon nje slijedi *cc*

011 *cc* 010*c*

Zatim je kombinacija *a* pa za njom slijedi *ab*:

100 *a* 000*a*

101 *ab* 100*b*

Nakon ove kombinacije ide *b* (uočite da je *bb* već u rječniku) pa zatim *ccb*

110 *b* 000*b*

111 *ccb* 011*b*

U rječniku kojeg treba isprazniti imamo već kombinaciju koja slijedi *ab* pa je glupo da je ne iskoristimo. Stoga kodiramo *aba* pa tek onda zajedno sa upisivanjem ovog elementa u rječnik (upisivanje nije obavezno) vršimo pražnjenje rječnika i kodiranje preostalog simbola *c*:

000 *aba* 101*a*
001 *c* 000*c*

Situacija kada je u rječniku već zastupljen kod izvora je da imamo na početku na lokacijama 000, 001 i 010 redom upisane *a, b* i *c*. Nakon toga kodiramo redom: *ab, aba, ba, cb* i *abc*:
011 *ab* 000*b*
100 *aba* 011*a*
101 *ba* 001*a*
110 *cb* 010*b*
111 *abc* 011

Sada možemo isprazniti ostatak rječnika i kodirati redom: *bb, cc, aa, ab* i *bc*:
011 *bb* 001*b*
100 *cc* 010*c*
101 *aa* 000*a*
110 *ab* 000*b*
111 *bc* 001*c*

Konačno preostaje nam kodiranje *cb, ab, ac*:
011 *cb* 010*b*
100 *ab* 000*b*
101 *ac* 000*c*

U našoj realizaciji drugi algoritam je nešto jednostavniji pošto nam je bilo potrebno samo 13 upisa dok je prva primjena zahtjevala 15 upisa. U realnim aplikacijama prednost jednog ili drugog postupka će se osjetiti tek nakon dužeg stringa a ujedno bitno zavisi i složenosti punjenja i pražnjenja rječnika jer ako startujemo sa rječnikom u kojem ima nešto (recimo kod izvoda) brže ćemo rječnik puniti i morati da ga praznimo češće.

Provjera dosadašnjeg gradiva

Za svaku od tema treba sebi postaviti i riješiti barem jedan zadatak.

1. Da li znate definicije: entropije, uslovne entropije, entropije združenog događaja, relativne entropije i međusobne informacije. Veze između ovih veličina i ograničenja koja postoje. Zadati konkretni model sistema za prenos informacija i odrediti entropije ulaza, izlaza, uslovne entropije, združenu entropiju i međusobnu informaciju.

2. Da li znate osnovnu ideju asimptotske ekviparticione osobine; definiciju trenutnog koda; Kraftovu nejednakost; način predstavljanja trenutnih kodova; Grayov kod.

3. Huffmanovo kodiranje (zadajte sebi zadatak vezan za bilo koju varijantu Huffmanovog koda sa mogućim vezama sa diferencijalnim i RLE kodom).

4. LZ kodiranje (zadajte sebi zadatak sa ovim kodom u varijantama koje su učene).

5. Naučiti osnovne pojmove o aritmetičkom kodiranju i uraditi jedan od zadataka, koji su vam ostavljeni za samostalan rad.

Poglavlje V
KANAL ZA PRENOS INFORMACIJA

5.1. Informacioni kanal - uvod

Informacioni kanal je statistički model medija preko kojeg prolaze signali nosioci informacija. U praksi postoje značajna fizička ograničenja vezana za kanal koja definišu koliko se informacija može prenositi preko kanala. U teoriji informacija kanal se modeluje kao skup uslovnih vjerovatnoća $P(b_j \mid a_i)$ koje kažu da se sa ulaza simbol a_i pojavio na izlazu kao simbol b_j. Veličina seta simbola s i q ne mora biti ista. Obično je set simbola koji se mogu primiti mnogo veći od onog koji su poslati, jer se prilikom prenosa u kanalu mogu dešavati greške. Detekcija i ispravljanje ovih grešaka su omogućene kodovima za korekciju i detekciju greške. Postoji mogućnost i da je set primljenih simbola manji od seta poslatih simbola, ali to nije za detaljnije razmatranje. Obično se uvodi oznaka $P(b_j \mid a_i) = P_{i,j}$. Uočite promjenu redosljeda indeksa. Red matrice uslovnih vjerovatnoća sadrži vjerovatnoće da je ulazni simbol a_i postao izlazni simbol b_j. Matrica tranzicije stanja ima sledeće osobine: 1) i-ti red odgovora i-tom ulaznom simbolu a_i; 2) j-ta kolona odgovara j-tom izlaznom simbolu b_j; 3) Suma elemenata u redu uvijek je jednaka 1:

$$\sum_{j=1}^{S} P_{i,j} = \sum_{j=1}^{S} P(b_j \mid a_i) = 1$$

(ovo znači da će za svaki ulaz nešto izaći iz sistema i da $P(b_j \mid a_i)$); 4. Ako je $p(a_i)$ vjerovatnoća simbola a_i važi:

$$\sum_{i=1}^{q} \sum_{j=1}^{s} P(b_j \mid a_i)\, p(a_i) = 1$$

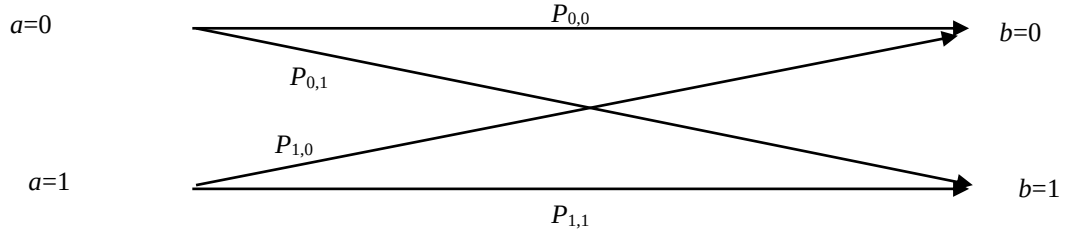
Vjerovatnoće simbola a_i i b_j se odnose na sledeći način:

$$p(a_1)P_{1,1} + p(a_2)P_{2,1} + ... + p(a_q)P_{q,1} = p(b_1)$$
$$p(a_1)P_{1,2} + p(a_2)P_{2,2} + ... + p(a_q)P_{q,2} = p(b_2)$$
$$...$$
$$p(a_1)P_{1,s} + p(a_2)P_{2,s} + ... + p(a_q)P_{q,s} = p(b_s)$$

ili u simplifikovanoj notaciji:

$$\sum_{i=1}^{q} p(a_i)P(b_j \mid a_i) = p(b_j) \quad j = 1, 2, ..., q$$

Posmatrajmo binarni simetrični kanal. Kanal je simetričana ako važi $P_{0,0}=P_{1,1}$ i $P_{1,0}=P_{0,1}$.



Neka su vjerovatnoće ulaznih simbola $p(a=0)=p$ i $p(a=1)=1-p$. Neka dalje važi $P_{0,0}=P_{1,1}=P$ i $P_{1,0}=P_{0,1}=Q$. Matrica kanala je:

$$\begin{bmatrix} P & Q \\ Q & P \end{bmatrix}$$

Odgovarajuća jednakost kanala postaje:

$$pP+(1-p)Q=p(b=0)$$
$$pQ+(1-p)P=p(b=1)$$

Kanal se analizira pomoću entropije i međusobne informacije koje su kako smo prethodno izveli vezane kao:

$$H(A,B)=H(B)+H(A\mid B)$$
$$I(A;B)=I(B;A)$$

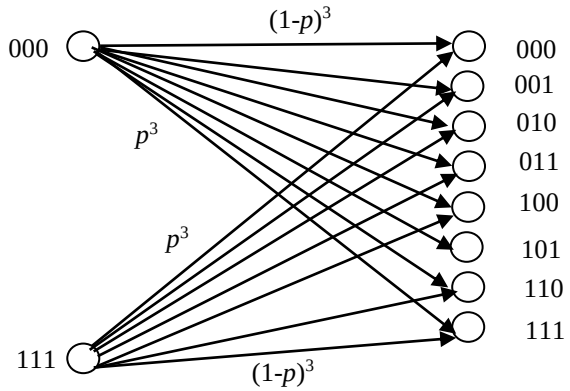
$$I(A;B)=H(A)+H(B)-H(A,B)=H(A)-H(A\mid B)=H(B)-H(B\mid A)\geq 0$$

Međusobna informacija se kada je u pitanju kanal naziva i prenesenom informacijom.

Takođe važi: $0\leq H(A\mid B)\leq H(A)$ i $0\leq H(B\mid A)\leq H(A)$ i $H(A,B)\leq H(A)+H(B)$.

Protumačimo fizički izlaganje u okviru ove sekcije. Mi pretpostavljamo da imamo izvor informacija koje treba prenijeti preko kanala. U idealizovanom modelu koji predstavljamo, mi kanal modelujemo preko uslovnih vjerovatnoća i kažemo da se kodni simboli koje prenosimo mogu promijeniti tokom prenosa. Dakle, prije kanala a nakon kodiranja izvora, morali smo postaviti blok koji se naziva koder kanala koji unosi malu redudanciju u naše poruke (dodatne bitove) tako da nakon promjena u kanalu i dalje možemo izvršiti rekonstrukciju poslate poruke. Dakle, odredištu informacije treba da prethodi dekoder kanala. Sam prenos i fizički detalji sistema, kao i signali nosioci informaciju nijesu tema teorije informacija i kodova. Očigledno se kod kanala koriste pojmovi koje smo ranije uveli (entropije, međusobne informacije itd. pa ih se obavezno podsjetite).

Primjer: Posmatrajmo slučaj kada želimo putem kanala za prenos informacija prenijeti simbole {0, 1}. U saznanju smo da se u kanalu mogu dogoditi greške (neka je vjerovatnoća greške po bitu p i neka je prenos bitova međusobno nezavisan). Veoma primitivan način kodiranja kanala je da umjesto jednog simbola šaljemo triput isti simbol čime omogućujemo dekodiranje na osnovu većinske – majoritetne logike. Dakle u ovom slučaju u kanal šaljemo dvije moguće sekvence {000, 111} a mogući su prijemi svih kombinacija od tri bita {000, 001, 010, 011, 100, 101, 110, 111}. Prikažimo ovaj model putem dijagrama



U nemogućnosti da prikažemo preglednije pojedine vjerovatnoće to ćemo uraditi putem matrice tranzicije:

$$\begin{array}{l} 0\rightarrow \\ 1\rightarrow \end{array}\left[\begin{array}{cccccccc} (1-p)^3 & (1-p)^2 p & (1-p)^2 p & (1-p)p^2 & (1-p)^2 p & (1-p)p^2 & (1-p)p^2 & p^3 \\ p^3 & (1-p)p^2 & (1-p)p^2 & (1-p)^2 p & (1-p)p^2 & (1-p)^2 p & (1-p)^2 p & (1-p)^3 \end{array}\right]$$
$$\begin{array}{cccccccc} \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \end{array}$$

5.2. Kapacitet kanala - Definicija i primjeri

Kapacitet kanala je mjera koja kaže koliko se informacija može dobiti korišćenjem kanala po nekoj jedinici korišćenja.

Definicija. Kapacitet kanala se definiše kao maksimalna međusobna informacija: $C = \max_{p(x)} I(X;Y)$

gdje je maksimum tražen preko svih mogućih ulaznih distribucija.

Primjer. Posmatrajmo kanal bez smetnji (bez šuma). Za svako x na ulazu dobijamo y na izlazu bez ikakve sumnje C=1bit, se dobija kada je $p(x)$=(1/2,1/2).

Primjer. Zašumljeni kanal sa ne-preklapajućim izlazima C=1 bit (kada ima smetnji nedovoljnih da promjene poslanu informaciju) za $p(x)$=(1/2,1/2).

Primjer. Prenos 26 štampanih karaktera sa vjerovatnoćom greške od 1/2. Postoji više načina da se ovo razmatra, ali najednostavniji je:

$$I(X;Y)=\max[H(Y)-H(Y|X)]=\max(H(Y))-1=\log 26-1=\log 13$$

Primjer. Kanal za prenos binarnih poruka sa vjerovatnoćom greške p:

$$I(X;Y) = H(Y) - H(Y \mid X) = H(Y) - \sum_x p(x) H(Y \mid X = x) =$$
$$= H(Y) - \sum_x p(x) H(p) = H(Y) - H(p) \le 1 - H(p)$$

Dakle, kapacitet kanala je: C=1-$H(p)$.

Pretpostavimo da je niz kojega šaljemo: X^n=10110100011011100101111001011000 i neka je primljena poruka: Y^n=10110000011011100111111001111000. Pa je sekvenca sa greškom:

$$E^n=0000010000000000000100000000100000$$

Ako znamo kod greške, možemo tačno dekodirati originalnu poruku X^n, jer u binarnom kanalu jedino se mogu pojavljivati jedinice i nule. Koliku informaciju nosi sekvenca E^n? Ako podrazumjevamo da je kreirana Bernulijevim izvorom sa vjerovatnoćom p, informacija koju nosi ova sekvenca je $H(p)$. Ova vrijednost ujedno predstavlja i informaciju koju smo mi izgubili u prenosu zašumljenim kanalom, tako da je kapacitet kanala: 1-$H(p)$ po svakom bitu informacije koja je primljena.

Primjer. Binarni kanal sa brisanjem. Vjerovatnoća da je prenos korektan 1-α, dok je vjerovatnoća da je poruka izbrisana i nije primljena α.

$$I(X;Y) = H(Y) - H(Y \mid X) = H(Y) - H(\alpha)$$

Da bi se maksimizovala ova veličina moramo naći max$H(Y)$. Postoje tri moguća ishoda ali se ne može postići $H(Y)$=log3 za bilo koju ulaznu distribuciju. Da bi se kapacitet odredio mora se odraditi malo složeniji posao. Neka je $P(X=1)=\pi$:

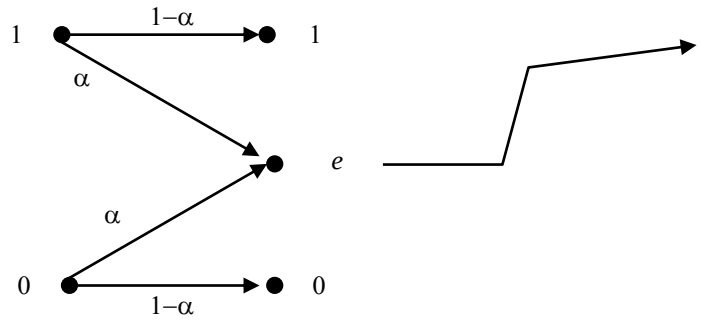
$$P(Y=0)=P(X=0)(1-\alpha)=(1-\pi)(1-\alpha) \qquad P(Y=e)=P(X=0)\alpha+P(X=1)\alpha=\alpha$$
$$P(Y=1)=P(X=1)(1-\alpha)=\pi(1-\alpha)$$

Tada važi:

$$H(Y) = -[(1-\pi)(1-\alpha)\log(1-\pi) + \alpha\log\alpha + \pi(1-\alpha)\log\pi(1-\alpha)] = H(\alpha) + (1-\alpha)H(\pi)$$

Sada važi: $I(X;Y) = H(\alpha) + (1-\alpha)H(\pi) - H(\alpha) = (1-\alpha)H(\pi)$. Ovaj izraz se maksimizuje za π=0.5 kada iznosi C=1-α.

Ovaj tip kanala se može ilustrovati kao:



Novouvedeno generičko stanje (stanje brisanja) na osnovu koga ne možemo donijeti odluku što je poslato.

Transmisiona matrica ovakvog kanala je:

$$P = \begin{bmatrix} 1-\alpha & \alpha & 0 \\ 0 & \alpha & 1-\alpha \end{bmatrix}$$

Kapacitet kanala ima sledeće osobine:

1. $C \geq 0$ (zašto?).
2. $C \leq \log||X||$ (zašto?).
3. $C \leq \log||Y||$ (zašto?).
4. $I(X;Y)$ je kontinualna funkcija od $p(x)$.
5. $I(X;Y)$ je konkavna funkcija od $p(x)$ (posjeduje maksimum).

1. Kapacitet je vezan za međusobnu informaciju koja je nenegativna veličina.
2. Međusobna informacija je jednaka $I(X;Y)=H(X)-H(X|Y)$. Kako je $H(X;Y)$ nenegativna veličina to je $I(X;Y)$ zasigurno veće ili jednako $H(X)$. Kako se C maksimalna vrijednost međusobne informacije to je kapacitet limitiran na maksimalnu vrijednost $H(X)$. Kao što smo već vidjeli maksimum entropije se postiže kada su događaji jednakovjerovatni i iznosi $\log||X||$ (logaritam broja elemenata u skupu).
3. Slično prethodnom $I(X;Y)=H(Y)-H(Y|X)$. Dalji dokaz u potpunosti prati prethodni postupak.
4. i 5. su tvrdnje koje nije baš jednostavni dokazati ali se može poći od definicije međusobne informacije i provjeriti zavisnost prvog i drugog izvoda od $p(x)$. U opštem slučaju ponovo ističemo da je ovo veoma teško pa se radi pojednostavljivanja može posmatrati binarni skup X sa vjerovatnoćama $\{p,1-p\}$ gdje postoji zavisnost samo od p i dokazati kontinualnost od p. Pored toga parametre kanala (u ovom slučaju uslovne vjerovatnoća) treba uzeti konstantnim jer one predstavljaju apstrakciju realnih procesa koji se dešavaju u kanalu. Međusobna informacija je dakle:

$$I(X;Y)=H(Y)-H(Y|X)$$

Ako uzmemo da je kanal simetričan (ovo će se za nesimetričan kanal vidjeti nešto kasnije) dobijamo da je međusobna informacija:

$$I(X;Y)=H(Y)-H(P)$$

tako da se problem dokaza kontinualnosti i postojanja maksimuma međusobne informacije svodi na dokaz kontinualnosti i postojanja maksimuma $H(Y)$. Y je binarni skup u kojem se nula i jedinica pojavljuju sa respektivnim vjerovatnoćama $pP+(1-p)(1-P)$ i $p(1-P)+(1-p)P$. Entropija $H(Y)$ je dakle:

$$H(Y) = -[pP + (1-p)(1-P)]\log_2\left[pP + (1-p)(1-P)\right] - \left[p(1-P) + (1-p)P\right]\log_2\left[p(1-P) + (1-p)P\right]$$

Izvod ove funkcije po p je

$$\frac{dH(Y)}{dp} = (1-2P)\log_2\frac{(1-P)(1-p)}{1-(1-P)(1-p)}$$

Dakle, pošto je u pitanju konačan izvod funkcije (osim za p=1, odnosno u granici intervala, ali tada je $I(X;Y)$=0) zaključujemo da je u pitanju $I(X;Y)$ kontinualna funkcija u ovom slučaju. Pored toga kandidat za maksimum je relativno jednostavan odnosno maksimum se postiže kada dostignemo nulu prvog izvoda a to je slučaj kada je argument logaritma jednak jedinici a to je za:

$$\frac{(1-P)(1-p)}{1-(1-P)(1-p)} = 1$$

Odavde slijedi da je:

$$2(1-P)(1-p) = 1 \rightarrow p = 1 - \frac{1}{2(1-P)}$$

Drugi izvod $H(Y)$ po p je

$$\frac{d^2H(Y)}{dp^2} = \frac{(1-2P)^2}{(1-p)(1-P)\left[(1-p)(1-P)-1\right]\ln(2)}$$

Brojilac ovog izraza je sigurno pozitivan dok je imenilac $(1-p)(1-P)\left[(1-p)(1-P)-1\right]$ sigurno negativan pošto je prvi član ispred srednje zagrade pozitivan dok drugi predstavlja istu vjerovatnoću umanjenu za 1 odnosno negativan broj pa se na osnovu toga da zaključiti da je drugi izvod u nuli prvog izvoda sigurno negativan što dalje znači da je u pitanju maksimum.

Definicija. Kanal se naziva simetričnim ako su redovi transmisione matrice kanala $p(y|x)$ permutacije istih vrijednosti i ako su kolone permutacije istih vrijednosti. Kanal je slabo simetričan ako je svaki red transmisione matrice $p(\cdot|x)$ permutacija i ako su sume svih kolona $\sum_x p(y \mid x)$ jednake.

Primjer transmisione matrice:

$$p(y \mid x) = \begin{bmatrix} 0.3 & 0.2 & 0.5 \\ 0.5 & 0.3 & 0.2 \\ 0.2 & 0.5 & 0.3 \end{bmatrix} = P$$

Indeksiranje se obavlja sa x po redovima i y po kolonama $P_{x,y}=p(y|x)$.

Da li je kanal sa brisanjem slabosimetričan (on očigledno nije simetričan a pitanje je može li da ispuni rasterećujuće uslove slabosimetričnosti i kada)?

Teorema. Za slabo simetrične kanale važi

$$C=\log||Y||-H(\text{redova transmisione matrice})$$

Ovo se postiže za (diskretnu) uniformnu distribuciju ulaznog alfabeta.

Dokaz. Neka je **r** red transmisione matrice. Tada:

$$I(X;Y)=H(Y)-H(Y\mid X)=H(Y)-H(\mathbf{r})\le \log||y||-H(\mathbf{r})$$

Jednakost važi za uniformnu distribuciju ulaznog alfabeta.

Definicija. Diskretni kanal se može označiti kao $(X,p(y|x),Y)$ gdje je χ ulazni set, Y - izlazni set i $p(y|x)$ uslovna vjerovatnoća (za svako $x\in\chi$). Diskretnim kanalom bez memorije se naziva $(X^n,p(y^n|x^n),Y^n)$ gdje je:

$$p(y_k\mid x^k,y^{k-1})=p(y_k\mid x_k)\,,$$

odnosno, gdje izlaz u posmatranom trenutku zavisi od ulaza u datom trenutku.

Definicija. Kod kanala $(X,p(y|x),Y)$ označen sa (M,n) se sastoji od:
1. Skupa indeksa $\{1,...,M\}$ (koji predstavljaju ulazni alfabet W).
2. Funkcija X^n: $\{1,2,...,M\}\to X^n$ koje daju kodne riječi $X^n(1)$, $X^n(2)$,...,$X^n(M)$.
3. Funkcije za dekodiranje g: $Y^n\to\{1,...,M\}$.

Drugim riječima, kod uzima simbol i W kodira ga i produkuje sekvencu od n simbola u χ. Vjerovatnoća greške se definiše kao:

$$\lambda_i=\Pr(g(Y^n)\ne i\mid X^n=X^n(i))$$

Ako je simbol poruke i, a izlazni simbol nije i tada imamo grešku. Ovo se može zapisati korišćenjem funkcije indikatora $I(\cdot)$

$$\lambda_i=\Pr(g(Y^n)\ne i\mid X^n=X^n(i))=\sum_{y^n}p(y^n\mid x^n(i))I(g(y^n)\ne i)$$

Mi ćemo ovdje baratati sa maksimalnom vjerovatnoćom greške. Ova veličina se definiše kao:

$$\lambda^{(n)}=\max_i\lambda_i$$

Prosječna vjerovatnoća greške je:

$$P_e^{(n)}=\frac{1}{M}\sum_{i=1}^{M}\lambda_i$$

Definicija. Brzina prenosa (kodni odnos, kodna brzina, ili na engleskom code rate) za kod (M,n) je:

$$R=[\log M]/n\ \text{ bita/transmisiji}$$

Primjer. Pretpostavimo da je M=4, i kodna riječ n=4 bita. Tada svaki ulazni simbol biva prikazan kao 2 bita informacija, a traži 4 simbola u transmisiji da bude prenesen $R\Rightarrow 1/2$.

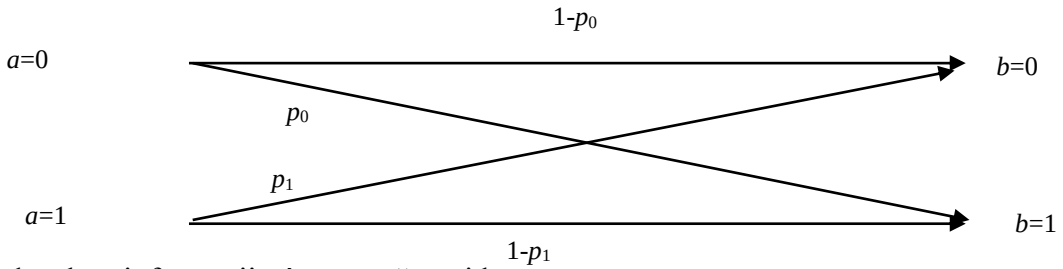
Definicija. R se naziva dostižnom, ako postoji niz od $(\lceil 2^{nR}\rceil,n)$ kodova takav da maksimalna vjerovatnoća greške $\lambda^{(n)}$ teži nuli kada $n\to\infty$.

Definicija. Kapacitet digitalnog kanala bez memorije (DMC) je jednak supremumu svih dostižnih brzina prenosa.

Prethodna definicija za kapacitet kanala se razlikuje od ove. Može se pokazati da su ove definicije ekvivalentne. Posljedica ove definicije je da za dostižnu brzinu prenosa vjerovatnoća greške teži nuli kako dužina bloka raste. Kako je kapacitet najveća dostižna brzina prenosa to znači da za brzinu prenosa manju od kapaciteta vjerovatnoća greške teži nuli sa porastom dužine kodne riječi.

Primjer. Posmatrajmo binarni nesimetrični kanal. Odredite mu kapacitet!

Rješenje. Riječ je o prilično složenom problemu iako na prvi pogled izgleda jako jednostavan, tek nešto složenijim od onog kod BSC-a. Sa p_0 ćemo označiti grešku koja se dešava kada se šalje 0 a sa p_1 grešku koja se dešava prilikom slanja simbola 1.



Međusobnu informaciju ćemo sračunati kao:

$$I(X;Y)=H(Y)-H(Y|X)$$

Događaj Y je binaran u kojem je entopija $H(P(1-p_0)+(1-P)p_1)=H(A(P))$. Uslovna entropija je jednaka

$$H(Y \mid X) = PH(p_0) + (1-P)H(p_1)$$

Sada je međusobna informacija

$$I(X;Y)=H(A(P))- PH(p_0) - (1-P)H(p_1)$$

Izvod po P ove međusobne informacije je jednak:

$$\frac{dI(X;Y)}{dP} = -(1-p_0-p_1)\log_2 A + (1-p_0-p_1)\log_2(1-A) - H(p_0) + H(p_1) = 0$$

Rješavajući ovu jednačinu po A dobijamo

$$A = \frac{1}{1+2^{\frac{H(p_0)-H(p_1)}{1-p_0-p_1}}}$$

Odnosno rješavajući ovo po P

$$P(1-p_0) + (1-P)p_1 = \frac{1}{1+2^{\frac{H(p_0)-H(p_1)}{1-p_0-p_1}}}$$

$$P = \frac{1}{1-p_0-p_1}\frac{1}{1+2^{\frac{H(p_0)-H(p_1)}{1-p_0-p_1}}} - \frac{p_1}{1-p_0-p_1}$$

Uvrstimo izraze za A i P u relaciju za međusobnu informaciju i dobijamo kapacitet:

$$C=\frac{1}{1+2^{\frac{H(p_0)-H(p_1)}{1-p_0-p_1}}}\log_2\left[1+2^{\frac{H(p_0)-H(p_1)}{1-p_0-p_1}}\right]-\frac{2^{\frac{H(p_0)-H(p_1)}{1-p_0-p_1}}}{1+2^{\frac{H(p_0)-H(p_1)}{1-p_0-p_1}}}\log_2\left[\frac{2^{\frac{H(p_0)-H(p_1)}{1-p_0-p_1}}}{1+2^{\frac{H(p_0)-H(p_1)}{1-p_0-p_1}}}\right]$$

$$-\frac{H(p_0)}{1-p_0-p_1}\frac{1}{1+2^{\frac{H(p_0)-H(p_1)}{1-p_0-p_1}}}-\frac{p_1H(p_0)}{1-p_0-p_1}-$$

$$-H(p_1)\left[1-\frac{1}{1-p_0-p_1}\frac{1}{1+2^{\frac{H(p_0)-H(p_1)}{1-p_0-p_1}}}+\frac{p_1}{1-p_0-p_1}\right]$$

$$=\log_2\left[1+2^{\frac{H(p_0)-H(p_1)}{1-p_0-p_1}}\right]-2^{\frac{H(p_0)-H(p_1)}{1-p_0-p_1}}\frac{H(p_0)-H(p_1)}{1-p_0-p_1}-\frac{H(p_0)-H(p_1)}{1-p_0-p_1}\frac{1}{1+2^{\frac{H(p_0)-H(p_1)}{1-p_0-p_1}}}$$

$$-\frac{p_1H(p_0)+H(p_1)(1-p_0)}{1-p_0-p_1}=$$

$$=\log_2\left[1+2^{\frac{H(p_0)-H(p_1)}{1-p_0-p_1}}\right]-\frac{H(p_0)-H(p_1)-p_1H(p_0)-H(p_1)(1-p_0)}{1-p_0-p_1}=$$

$$=\log_2\left[1+2^{\frac{H(p_0)-H(p_1)}{1-p_0-p_1}}\right]-\frac{H(p_0)(1-p_1)-H(p_1)p_0}{1-p_0-p_1}$$

Da bi provjerili naša izvođenja provjerimo što se dešava u slučaju kada je $p_0=p_1=p$. U tom slučaju izraz se znatno pojednostavljuje na:

$$C=\log_2[2]-\frac{(1-2p)H(p)}{1-2p}=1-H(p)$$

Ovim je potvrđena tačnost uvedenog izraza makar u navedenom partikularnom slučaju.

5.3. Združene tipične sekvence

U ovoj sekciji mi ćemo generalizovati teoreme o tipičnoj sekvenci, aproksimativnom broju sekvenci, vjerovatnoći tipične sekvence itd.

Definicija. Skup $A_\varepsilon^{(n)}$ je združena tipična senvenca $\{(x^n,y^n)\}$ u odnosu na distribuciju $p(x,y)$ je skup sekvenci sa empirijskom (statistički prosječnom) entropijom bliskoj tačnoj entropiji:

$$A_\varepsilon^{(n)}=\left\{(x^n,y^n)\in X^n\times Y^n:\left|-\frac{1}{n}\log p(x^n)-H(X)\right|<\varepsilon \text{ i } \left|-\frac{1}{n}\log p(y^n)-H(Y)\right|<\varepsilon\right.$$
$$\left.\text{i } \left|-\frac{1}{n}\log p(x^n,y^n)-H(X,Y)\right|<\varepsilon\right\}$$

Teorema. 1. $\Pr((X^n,Y^n)\in A_\varepsilon^{(n)})\to 1$ kada $n\to\infty$ (kako se veličina bloka povećava vjerovatnoća da smo primili združenu tipičnu sekvencu se približava 1).

2. $|A_\varepsilon^{(n)}|\leq 2^{n(H(X,Y)+\varepsilon)}$ (broj združenih tipičnih sekvenci je blizak broju određenom entropijom (X,Y)).

3. Ako $(\tilde{X}^n,\tilde{Y}^n)\sim p(x^n)p(y^n)$ tada su $\tilde{X}^n$ i $\tilde{Y}^n$ statistički nezavisni sa istom marginalnom distribucijom $p(x^n,y^n)$ tada: $\Pr((\tilde{X}^n,\tilde{Y}^n)\in A_\varepsilon^{(n)})\leq 2^{-n(I(X,Y)-3\varepsilon)}$. Za dovoljno veliko n:

$$\Pr((\tilde{X}^n,\tilde{Y}^n)\in A_\varepsilon^{(n)})\geq(1-\varepsilon)2^{-n(I(X;Y)+3\varepsilon)}$$

Grubo govoreći da tipičnih sekvenci X ima $2^{nH(X)}$, tipičnih sekvenci Y ima $2^{nH(Y)}$, dok tipičnih združenih sekvenci ima $2^{nH(X,Y)}$. Posljednji dio teoreme kaže da ako se izabere (slučajno) oko $2^{nI(X;Y)}$ parova sekvenci izabraćemo združeni tipični par.

Teorema. (teorema o kodiranju kanala). Sve brzine prenosa ispod kapaciteta C su dostižne. Odnosno, za svako $\varepsilon>0$ i $R<C$ postoji sekvenca od $(2^{nR},n)$ kodova sa maksimumom vjerovatnoće greške $\lambda^{(n)}\to 0$ (posljedica je da svaka sekvenca $(2^{nR},n)$ sa $\lambda^{(n)}\to 0$ mora imati $R\leq C$).

Napomena. Teorema je asimpotska i ovo važi za velike dužine blokova. Dalje, teorema nije konstruktivna odnosno ne ukazuje na način kako se do sekvence može doći. Dokaz teoreme je izuzetno složen i možete ga naći u dijelu sa primjerima.

5.4. Suprotnost kodne teoreme (praktično dokaz u suprotnom smjeru)

Posmatrajmo slučaj kanala bez grešaka i pokažimo da on zahtjeva brzinu prenosa koja je manja od kapaciteta. Kanal bez grešaka je onaj kod kojeg sekvenca Y^n određuje ulaz W bez greške tako da je $H(W|Y^n)=0$. Možemo postaviti granice pretpostavljajući da je W uniformno distribuirano preko 2^{nR} ulaznih simbola tao da je $H(W)=nR$. Sada slijedi:

$$nR = H(W) = H(W \mid Y^n) + I(W;Y^n) = I(W;Y^n) \le I(X^n;Y^n) \le \sum_{i=1}^{n} I(X_i;Y_i) \le nC$$

gdje $I(W;Y^n) \le I(X^n;Y^n)$ slijedi na osnovu nejednakosti procesiranja podataka (pogledajte teoremu 2.5.1). Sledeći korak će biti dokazan dok posljednji korak slijedi iz definicije kapaciteta kanala. Na osnovu ove relacije možemo da zaključimo da $R \le C$. Podsjetimo se Fanoove nejednakosti o odnosu vjerovatnoće greške i entropije. Koristivši notaciju vezanu za kanal možemo zapisati:

$$H(X^n \mid Y^n) \le 1 + P_e^{(n)} nR$$

Može se uočiti da (ovo se može i dokazati):

$$I(X^n;Y^n) \le nC$$

Sada imamo dovoljno sredstava da dokažemo suprotnost kodne teoreme:
Teorema. Za svaku sekvencu $(2^{nR},n)$ kodova sa $\lambda^{(n)} \to 0$ mora važiti $R \le C$.

Dokaz: Uočimo da $\lambda^{(n)} \to 0$ implicira $P_e^{(n)} \to 0$. Neka su simboli raspoređeni uniformno:

$$nR = H(W) = H(W \mid Y^n) + I(W;Y^n) \le H(W \mid Y^n) + I(X^n(W);Y^n) \le$$
$$\le 1 + P_e^{(n)} nR + I(X^n(W);Y^n) \le 1 + P_e^{(n)} nR + nC$$

Ovo se sada može zapisati:

$$P_e^{(n)} \ge 1 - \frac{C}{R} - \frac{1}{nR}$$

Ako je $R>C$ za dovoljno veliko n $P_e^{(n)}$ je ograničeno sa 0.

5.5. Združena izvor/kanal kodna teorema

Vidjeli smo da kodni odnos kod kodiranja izvora ne može biti manji od entropije izvora $H(X)$, $R>H$. Takođe, vidjeli smo da se prenos može obavljati sa brzinom koja je manja od kapaciteta kanala $R<C$. Postavlja se pitanje kako se ove dvije fundamantalne činjenice odnose jedna sa drugom. Da li je činjenica da je kodni odnos H manji od C neophodan i potreban uslov za slanje poruka sa izvora preko kanala? Združena teorema kaže da je moguće konstruisati kod sa $H(X)<C$ koji daje vjerovatnoću greške u prenosu koja teži nuli a za veliku dužinu kodne riječi da je $H(X)>C$ zasigurno kod koji nije dostižan odnosno koji ne postiže vjerovatnoću greške koja teži 0 ni za jednu dužinu kodne riječi. Ova teorema ukazuje i na moguću vezu kodiranja izvora i kodiranja kanala ali o tome nećemo dalje detaljisati u ovom materijalu.

II Šenonova teorema sa njenim posljedicama su asimptotske, odnosno, važe za slučaj kodova sa veoma dugačkim kodnim riječima. Dalje, ova teorema nije konstruktivna i ne kaže kako se do datih "dobrih" kodova može doći. Srećom kao što će se pokazati nešto kasnije ova teorema se može primjeniti kod kratkih (realnih) kodova, a tokom vremena su razvijene i strategije za dizajn ovih kodova.

5.6. Gausovski kanal

Pretpostavimo da šaljemo informacije kroz kanal koji je podvrgnut Gausovskom aditivnom šumu:

$$Y_i = X_i + Z_i$$

gdje je Y_i izlaz iz kanala, X_i ulaz iz kanala i Z_i Gausov šum sa nultom srednjom vrijednošću i varijansom N: $Z_i \sim \mathbf{N}(0,N)$. Ovo je model u kome izlaz može uzeti kontinualne vrijednosti. Ovaj model je uobičajen u brojnim komunikacionim primjenama. Pretpostavimo da predajnik šalje $\sqrt{P}$ i $-\sqrt{P}$, a da prijemnik gleda u izlazni signal i na osnovu odgovarajućeg praga odlučuje koju je poruku primio. Vjerovatnoća greške se može opisati kao:

$$P_e = \frac{1}{2}P(Y<0 \mid X=\sqrt{P}) + \frac{1}{2}P(Y>0 \mid X=-\sqrt{P}) = \frac{1}{2}P(Z<-\sqrt{P} \mid X=\sqrt{P}) + \frac{1}{2}P(Z>\sqrt{P} \mid X=-\sqrt{P})$$
$$= P(Z>\sqrt{P}) = 1-\Phi(\sqrt{P/N})$$

gdje je:

$$Q(x)=\frac{1}{\sqrt{2\pi}}\int_x^{\infty}e^{-x^2/2}dx \qquad\qquad \Phi(x)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x}e^{-x^2/2}dx$$

Definicija. Informacioni kapacitet Gausovskog kanala sa ograničenjem snage je:

$$P_e=Q(\sqrt{P/N})-\Phi(\sqrt{P/N})=\frac{1}{\sqrt{2\pi}}\int_{\sqrt{P/N}}^{\infty}\exp(-x^2/2)dx-\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\sqrt{P/N}}e^{-x^2/2}dx$$

$$C=\max_{p(x):EX^2\le P} I(X;Y)$$

Kapacitet kanala možemo zapisati kao

$$I(X;Y)=H(Y)-H(Y|X)=H(Y)-H(X+Z|X)=H(Y)-H(Z|X)$$

Dakle, izlaz je jednak sumi signala i Gausovskog procesa. Međutim, razlike ne zavisi od X-a već samo od Z aditivnog šuma pa stoji $H(X+Z|X)=H(Z|X)$. Pošto smatramo da je aditivni šum nezavisan od signala dobijamo da je $H(Z|X)=H(Z)$. Sada slijedi da je međusobna informacija:

$$I(X;Y)=H(Y)-H(Z)$$

Kako je Z Gausovski šum koji je analogna veličina to moramo za njega definisati entropiju kao za analognu veličinu. Entropija analognih veličina se definiše kao (naziva se diferencijalna entropija, ponekad se označava malim slovom):

$$H(Z)=-\int_{-\infty}^{\infty}p(x)\log_2 p(x)dx$$

gdje je $p(x)$ funkcija gustine raspodjele Gausovog šuma. Nakon relativno jednostavnih izvođenja (koja uključuju parcijalnu integraciju) slijedi:

$$H(Z)=\log(2\pi eN)/2$$

gdje je N varijansa šuma (spektralna gustina njegove snage). Preostaje nam da odredimo veličinu $H(Y)$. Pošto je Y jednako poslatom signalu plus šumu matematičko očekivanje Y^2 se može zapisati kao:

$$E(Y^2)=E(X^2)+2E(XZ)+E(Z^2)$$

Pošto su signal i šum međusobno nezavisni to je $E(XZ)=0$ odakle slijedi da je

$$E(Y^2)=E(X^2)+E(Z^2)=P+N$$

Pošto Gausovski šum ima najveću moguću entropiju to slijedi da je $H(Y)$ ograničeno na:

$$H(Y)\le\log(2\pi e(\mathrm{P}+N))/2$$

pa je međusobna informacija:

$$I(X;Y)\le\frac{1}{2}\log\left(1+\frac{P}{N}\right)$$

odakle slijedi da je kapacitet Gausovskog kanala:

$$C=\max I(X;Y)=\frac{1}{2}\log\left(1+\frac{P}{N}\right)\text{ bita po korišćenju kanala}$$

Maksimum se dobija kada je ulazni signal takođe distribuiran Gausovski.

Definicija. (M,n) kod za Gausovski kanal sa ograničenjem snage P sastoji se od sledećeg:
1. Skupa indeksa $\{1,2,...,M\}$.
2. Kodirajuće funkcije $x:\{1,...,M\}\to X^n$ koja preslikava ulazni skup indeksa u sekvencu koja je n elemenata dugačka, $x^n(1)$, $x^n(2)$,...,$x^n(M)$ tako da prosječno ograničenje snage zadovoljava:

$$\sum_{i=1}^{n}(x_i^n(w))^2\le nP$$

za w=1,2,...,M.
3. Dekodirajuće funkcije g: $y^n\to\{1,2,...,M\}$.

Definicija. R je dostižno za Gausovski kanal sa ograničenjem snage P ako postoji sekvenca od (2^{nR},n) kodova sa kodnim riječima koje zadovoljavaju ograničenje snage takvo da je maksimalna vjerovatnoća greške $\lambda^{(n)}\to 0$. Kapacitet kanala je supremum dostižnog skupa.

Teorema. Kapacitet Gausovski kanal sa ograničenjem snage P i varijansom šuma N je:

$$C = \frac{1}{2}\log\left(1+\frac{P}{N}\right) \text{ bita po prenosu}$$

Geometrijsko tumačenje. Za kodnu riječ dužine *n* primljeni vektor (u *n* prostoru) je normalno distribuiran sa srednjom vrijednošću jednakom tačnoj kodnoj riječi. Sa visokom vjerovatnoćom primljeni vektor sadrži sferu oko srednje vrijednosti sa radijusom $\sqrt{n(N+\varepsilon)}$. Zašto? Zato što sa visokom vjerovatnoćom vektor pada u zonu standardne devijacije oko sredine u svakom pravcu tako da je ukupna distanca u svim pravcima jednaka Euklidskoj sumi:

$$E[z_1^2 + z_2^2 + ... + z_n^2] = nN$$

Ovo je kvadrat očekivane distance u okviru koje mi očekujemo da se nalazi. Ako pridružimo sve u okviru sfere datoj kodnoj riječi do greške će doći samo ako je detektujemo van ove kodne riječi. Druge kodne riječi će imati drugačije sfere sa radijusima koji su aproksimativno $\sqrt{n(N+\varepsilon)}$. Primljeni vektori su ograničeni energijom sa *P* tako da moraju da leže u sferi radijusa $\sqrt{n(P+N)}$. Dakle, broj nepresjecajućih sekvenci koje se mogu dekodirati je približno:

$$\text{broj sfera} \approx \frac{\text{zapremina sfere u } n \text{ prostoru radijusa r}=\sqrt{n(P+N)}}{\text{zapremina sfere u } n \text{ prostoru radijusa r}=\sqrt{n(N+\varepsilon)}}$$

Zapremina sfere radijusa *r* u *n* prostoru je proporcionalna r^n. Uvrštavajući ovu činjenicu u prethodnu relaciju dobijamo:

$$\text{broj sfera} \approx \frac{(n(P+N))^{n/2}}{(n(N+\varepsilon))^{n/2}} \approx 2^{\frac{n}{2}\left(1+\frac{P}{N}\right)}$$

Dokaz. 1. Prvo generišemo kodnu knjigu *slučajno*. Kodnu knjigu ćemo generisati prema Gausovskoj raspodjeli: neka $X_i(w)$: i=1,2,...,n bude kodna sekvenca koja odgovara ulaznom indeksu *w* gdje svako $X_i(w)$ je selektovano u skladu sa slučajnom promjenljivom sa istom raspodjelom u skladu sa Gausovskim zakonom $\mathbf{N}(0, P-\varepsilon)$. Sa visokom vjerovatnoćom ovo ima prosječnu snagu *P*. Kodna knjiga je poznata i prijemniku i predajniku.

2. Kodirati kako je opisano prethodno.

3. Prijemnik dobija Y^n i gleda u listu kodnih riječi $\{X^n(w)\}$ i traži onu koja je združeno tipična sa primljenim vektorom. Ako postoji jedna primljena sekvenca onda je poznata predata sekvenca a ako ima više onda se došlo do greške. Greška se prijavljuje i ako nije zadovoljeno ograničenje snage.

Za vjerovatnoću greške pretpostavimo bez gubitaka opštosti da je poslata prva kodna riječ:

$$Y^n = X^n(1) + Z$$

Definišimo sljedeće događaje (da kodna riječ prelazi dato ograničenje):

$$E_0 = \left\{\frac{1}{n}\sum_{i=1}^{n} X_i^2(1) > P\right\} \qquad E_i\{(X^n(i), Y^n) \text{ je u } A_\varepsilon^{(n)}\}$$

Vjerovatnoća greške je jednaka:

$$P(\varepsilon) = P(E_0 \cup E_1^c \cup E_2 \cup ... \cup E_{2^nR}) \le P(E_0) + P(E_1^c) + \sum_{i=2}^{2^nR} P(E_i)$$

Posljednja relacija važi po pravilima vezanim za unije. Po zakonu velikih brojeva važi $P(E_0) \to 0$. Po združenoj ekviparticionoj osobini $P(E_1^c) \le \varepsilon$ za *n* dovoljno veliko. Na osnovu pravila o generisanju procesa $X^n(1)$ i $X^n(i)$ su nezavisni pa to važi i za Y^n i $X^n(i)$ za $i \ne 1$. Dakle, vjerovatnoća da su $X^n(1)$ i Y^n združeno tipični je $\le 2^{-n(I(X;Y)-3\varepsilon)}$ po asimptotskoj ekviparticionoj osobini. Dakle, slijedi:

$$P_e^{(n)} \le \varepsilon + \varepsilon + \sum_{i=2}^{2^{nR}} 2^{-n(I(X;Y)-3\varepsilon)} \le 2\varepsilon + (2^{nR}-1)2^{-n(I(X;Y)-3\varepsilon)} \le 2\varepsilon + 2^{nR}2^{-n(I(X;Y)-3\varepsilon)} \le 3\varepsilon$$

za dovoljno veliko *n* ako $R < I(X;Y) - \varepsilon$. Ovo nam daje prosječnu vjerovatnoću greške. Mi sada možemo proći kroz iste vrste argumenata da zaključimo da maksimum vjerovatnoće greške mora takođe biti nula.

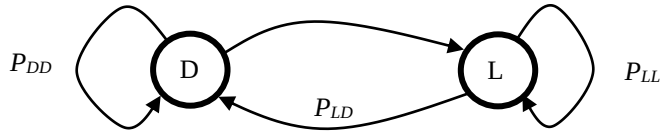
"Suprotnost" ove teoreme je da $R > C$ nije dostižno odnosno da ako $P_e^{(n)} \to 0$ mora da važi $R \le C$.

5.7. Modelovanje kanala

U praksi su razvijani brojni modeli kanala koji se koriste za slučaj kanala kod kojega se dešavaju rijetke ali uzastopne greške. Ovakve greške između ostalog izazivaju rad automobilskih motora, grmljavine, namjerno ometanje itd. Stoga se morao razviti model kanala kod kojega postoji jedna relativno mala greška u normalnim uslovima dok u uslovima impulsnih smetnji greška je velika i blizu 0.5. Takav kanal je modelovao Gilbert pa se po njemu i naziva Gilbertovim. Model ima dva

P_{DL}

stanja *D* (dobro) i *L* (loše). Svako stanje kanala se ponaša kao posebni binarni simetrički kanal. Pored toga postoje vjerovatnoće za prelazak iz jednog u drugo stanje P_{DD} (iz dobrog u dobro), P_{DL} (iz dobrog u loše), P_{LD} (iz lošeg u dobro), i P_{LL} (iz lošeg u loše).



U dobrom stanju vjerovatnoća greške može biti izuzetno mala (praktično 0) dok u lošem stanju vjerovatnoća greške može biti veoma velika (bliska 0.5). Matrica tranzicije se u slučaju dobrog i lošeg stanja može jednostavno opisati kao:

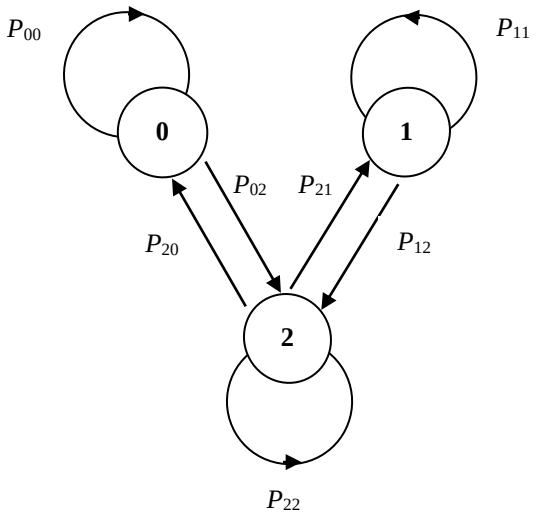
$$\begin{bmatrix} P_{DD} & P_{DL} \\ P_{LD} & P_{LL} \end{bmatrix}$$

a za dobro odnosno loše stanje se može opisati preko matrica tranzicije kao:

$$\begin{bmatrix} 1-p_0 & p_0 \\ p_0 & 1-p_0 \end{bmatrix} \qquad \begin{bmatrix} 1-p_1 & p_1 \\ p_1 & 1-p_1 \end{bmatrix}$$

gdje je $0.5>p_1>p_0>0$. Relativno je teško objediniti obije ove situacije ali možete pokušati uvođenjem 4 stanja L_0, L_1, D_0 i D_1. Gdje L_0 označava loše stanje sa velikom vjerovatnoćom greške u stanju 0 dok je stanje na primjer D_1 dobro stanje sa bitom 1.

Gilbertov model ne omogućava generisanje sekvence grešaka s grupisanim paketima grešaka. To je omogućeno korišćenjem Smit-Boven-Džojsovog modela čiji je model prikazan na slici dolje.



Stanja 0 i 1 su dobra s različitim ili istim vjerovatnoćama greške dok je stanje 2 loše sa verovatnoćom greške (obično) p_2=0.5 i odgovara paketima grešaka. Jedno od dobrih stanja odgovara intervalima između paketa grešaka u grupi, drugo intervalima između grupa paketa. Prelazak izmedju dva dobra stanja nije omogućen. Matrica prelaznih verovatnoća je

$$\begin{bmatrix} P_{00} & 0 & P_{02} \\ 0 & P_{11} & P_{12} \\ P_{20} & P_{21} & P_{22} \end{bmatrix}$$

Stacionarne verovatnoće pojednih stanja su

$$P_0 = \frac{P_{12}P_{20}}{D}, P_1 = \frac{P_{02}P_{21}}{D}, P_2 = \frac{P_{02}P_{12}}{D}$$

gde je $D=P_{12}P_{20}+P_{02}P_{21}+P_{02}P_{12}$. *P*roces na primjer započinje u stanju 0 i tu "boravi" izvesno vrijeme. Nakon toga prelazi u stanje 2 i generiše se paket grešaka. Zatim prelazi obično u stanje 1 (jer je izabrano da bude P_{21} veće od P_{20}). Sada proces kruži između stanja 1 i 2. Na kraju, proces prelazi u stanje 0 i ponovo se generiše dug interval između paketa grešaka.

5.8. Fizičke karakteristike kanala

Kanal smo (a i u daljem radu ćemo) posmatrali kao matricu uslovnih vjerovatnoća, ne vodeći računa o stvarnim (fizičkim) karakteristikama kanala kao i o signalu koji je korišćen za prenos informacija. Ovdje ćemo dati samo nekoliko napomena bez želje da ulazimo u detalje koji će biti razmatrani u drugim disciplinama. Vidjeli smo da imamo veličinu koju nazivamo kapacitet kanala *C* i da je ona u vezi sa veličinom koju smo zvali **kodni odnos** *R*. Za jedan od kodova koje ćemo uskoro učiti vidjećemo da je $R=4/7$. Ovaj kod će se moći koristiti kod kanala koji imaju kapacitet veći (ili barem jednak) sa 4/7. Što praktično znači 4/7? To znači da ako smo poslali 7 bita na ulaz kanala na izlazu možemo dobiti 4 bita korisne informacije. Znači po jednom ulaznom bitu dobijamo u prosjeku 4/7 bita korisne informacije. Ali, koliko je to bita u sekundi što je od ključne važnosti za nas. To je već pitanje koje zadire u fizičnost kanala. Kanalu je obično dodijeljen neki frekvencijski opseg širine *B* (uzeli smo ovu skraćenicu zbog engleske riječi **bandwidth –** propusni opseg). Nikvistov kriterijum i teorema Koteljnikova (obično se izučava u telekomunikacijama i/ili obradi signala) pokazuje se da je maksimalno moguće proslijediti bita u kanal proporcionalno širini propusnog opsega *cB* (konstanta proporcionalnosti *c* nas ne interesuje). Ova veličina se sada mjeri u 1/*s* a ovo dalje znači da je stvarna količina informacija koja se može dobiti iz kanala *cBC*, ali sada dobijena veličina se mjeri u bitima po sekundi. Često se koristi veličina koja se naziva iskorišćenost kanala koja je jednaka *G*/*cBC*, gdje je *G* stvarni prenos podataka u kanalu (teško se može postići veličina *cBC*).

Po Nyquistovoj teoremi koja kaže da ako je signal odabran sa korakom koji je $T \le 1/2f_m = 1/B$ moguće je rekonstruisati precizno bez greške. Znači da je preko komunikacionog kanala moguće prenijeti u jedinici vremena informacija predstavljenih sa *B* odbiraka. Ako je svaki odbirak kodiran sa *b* bita to znači da se putem komunikacionog kanala može prenijeti *Bb* bita u jednoj sekundi. Ako se zbog grešaka kapacitet kanala smanjuje na *CBb* i sada vidimo da je zapravo konstanta proporcionalnosti kod veze između kapaciteta i dostižnih granica kanala jednak broju bita kojim se jedan odbirak kodira.

Mnogo "priče" vezane za kanal posvećeno je idealizovanim slučajevima kada se podrazumjeva da su greške koje se u kanalu javljaju međusobno nezavisne. Nažalost, ovo je veoma rijetko slučaj. Npr. u telefonskom kanalu sa bakarnim žicama atmosferska pražnjenja uzrokuju greške. Ta pražnjenja nijesu međusobno nezavisna (ako se pražnjenje pojavi u ovom trenutku velika je vjerovatnoća da će ga biti opet uskoro). Da situacija bude interesantnija tokom vremena veliki broj kodova je razvijan baš za takve kanale, koji realno ne postoje u praksi. Postoji šala iz ranih 60-tih kada su počele prve komunikacije sa svemirskim objektima (vještačkim satelitima i drugim vasionskim brodovima) kad je otkriveno da je tzv. kosmički kanal približno Gausovski sa nezavisnim vjerovatnoćama grešaka. Naime, tada je rečeno da je **konačno otkriven kanal za naše kodove**. Naravno, analize kanala i dizajn kodova za slučaj realnih kanala. Stoga se mi početnici u ovoj oblasti nećemo previše osvrtati na realne kanale. Jedna od važnih činjenica kod realnih kanala je njihovo dobro modelovanje. Naime, mnogo je bolje vršiti ispitivanje naših sistema prvo na računaru na osnovu modela kanala, pa tek onda u praktičnim realizacijama.

Dajmo neke fizičke podatke za realne kanale. Telefonski kanal ima propusni opseg 4kHz (tačnije može i 3.4KHz). Ovaj kanal treba da obezbjedi odnos signal/šum od oko 30dB (da je signal oko 1000 puta veći od smetnji). Kod TV kanala propusni opseg je minimalno 5MHz i ovaj opseg treba da da barem 30-100 nijansi različitih boje (ako je crno-bijela TV to su nijanse sivog). Da bi se ovakav signal prenosio u kanalu potrebno je $2B\log r$ informacija po sekundi da bude preneseno ($B \geq 5$MHz dok je r broj nivoa sivog).

Zadaci za vježbu

5.1. Dokazati prvu teoremu iz Glave 5.3.

Rješenje: Po zakonu velikih brojeva slijedi da je:

$$-\frac{1}{n}\log p(X^n) \to -E\log p(X) = H(X)$$

Koristeći konvergenciju u vjerovatnoći slijedi da za neko $\varepsilon>0$ postoji n_1 takvo da $n>n_1$:

$$\Pr\left(\left|-\frac{1}{n}\log p(X^n) - H(X)\right| > \varepsilon\right) < \frac{\varepsilon}{3}$$

Na sličan način slijedi za $n>n_2$:

$$\Pr\left(\left|-\frac{1}{n}\log p(Y^n) - H(Y)\right| > \varepsilon\right) < \frac{\varepsilon}{3}$$

i za $n>n_3$:

$$\Pr\left(\left|-\frac{1}{n}\log p(X^n,Y^n) - H(X,Y)\right| > \varepsilon\right) < \frac{\varepsilon}{3}$$

Za $n>\max(n_1,n_2,n_3)$ slijedi da vjerovatnoća unije tri skupa mora biti manja od ε. Odnosno vjerovatnoća skupa $A_\varepsilon^{(n)}$ mora biti veća od $1-\varepsilon$. Drugi dio dokaza se obavlja na isti način kao kod tipične sekvence:

$$1 = \sum p(x^n,y^n) \geq \sum_{A_\varepsilon^{(n)}} p(x^n,y^n) \geq |A_\varepsilon^{(n)}|\, 2^{-n(H(X,Y)+\varepsilon)}$$

Odavde slijedi:

$$|A_\varepsilon^{(n)}| \leq 2^{n(H(X,Y)+\varepsilon)}$$

Za treći dio dokaza prvo ukažimo da za dovoljno veliko n: $\Pr(A_\varepsilon^{(n)}) \geq 1-\varepsilon$ pa se može pisati:

$$1-\varepsilon \leq \sum_{(x^n,y^n)\in A_\varepsilon^{(n)}} p(x^n,y^n) \leq |A_\varepsilon^{(n)}|\, 2^{-n(H(X,Y)-\varepsilon)}$$

odakle slijedi:

$$|A_\varepsilon^{(n)}| \geq (1-\varepsilon)2^{n(H(X,Y)-\varepsilon)}$$

Ako su $\tilde{X}$ i $\tilde{Y}$ nezavisne sa istim marginalnim vjerovatnoćama kao X i Y slijedi:

$$\Pr((\tilde{X}^n,\tilde{Y}^n) \in A_\varepsilon^{(n)}) = \sum_{(x^n,y^n)\in A_\varepsilon^{(n)}} p(x^n)p(y^n) \leq 2^{n(H(X,Y)+\varepsilon)}2^{-n(H(X)-\varepsilon)}2^{-n(H(Y)-\varepsilon)} = 2^{n(I(X;Y)-3\varepsilon)}$$

pošto ima aproksimativno $2^{n(H(X,Y)+\varepsilon)}$ elemenata u sumi. Na sličan način se može izvesti i posljednja nejednakost.

5.2. Dokažite teoremu o kodiranju kanala.

Rješenje: Fiksirajmo $p(x)$. Generišimo $(2^{nR},n)$ kodova na slučajan način koristivši distribuciju $p(x)$. Kodne riječi se formiraju na sledeći način:

$$p(x^n) = \prod_{i=1}^{n} p(x_i)$$

Postavimo ove kodne riječi u redove matrice:

$$C = \begin{bmatrix} x_1(1) & x_2(1) & \cdots & x_n(1) \\ x_1(2) & x_2(2) & \cdots & x_n(2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(2^{nR}) & x_2(2^{nR}) & \cdots & x_n(2^{nR}) \end{bmatrix}$$

Svaki elemenat u matrici je nezavisno generisan po distribuciji $p(x)$. Vjerovatnoća svakog posmatranog koda je:

$$\Pr(C) = \prod_{w=1}^{2^{nR}} \prod_{i=1}^{n} p(x_i(w))$$

Pretpostavimo sledeći model komuniciranja:

1. Slučajni kod C je kreiran kako je opisano. Kod je dostupan (poznat) i prijemniku i predajniku a i prijemnik i predajnik znaju tranizicionu matricu $p(y|x)$.

2. Poruka W je izabrana od strane pošiljaoca na slučajan način:

$$\Pr(W = w) = 2^{-nR}, \qquad w = 1, 2, ..., 2^{nR}$$

i ova kodna riječ odgovara w-tom redu matrice C i poslata preko kanala.

3. Prijemnik prima poruku na osnovu distribucije (pretpostavka je da je kanal bez memorije):

$$p(y^n \mid x^n(w)) = \prod_{i=1}^{n} p(y_i \mid x_i(w))$$

4. Na osnovu primljene poruke prijemnik pokušava da odredi koja je poruka poslana. Da bi mogli da pratimo proceduru donošenja odluke korišćena je procedura dekodiranja zasnovana na tipičnom skupu. Prijemnik odlučuje da je poruka $\hat{W}$ poslana ako: $(X^n(\hat{W}), Y^n)$ je združeno tipična i ako ne postoji nijedno drugo k za koje važi da je $(X^n(k), Y^n) \in A_\varepsilon^{(n)}$ (prijemnik se ne može odlučiti između više mogućnosti).

5. Ako je $W \neq \hat{W}$ došlo je do greške.

Sledeći korak je da nađemo prosječnu kodnu grešku (prosječnu preko svih kodnih riječi u kodnoj knjizi i prosječnu preko svih kodnih knjiga C). Ako je vjerovatnoća greške dobijena usrednjavanjem preko slučajnih kodnih knjiga je proizvoljno mala to znači da postoji barem jedna kodna knjiga koja daje malu prosječnu vjerovatnoću greške. Dakle, ovo je način da pokažemo da dobar kod postoji:

$$P(\varepsilon) = \sum_C P(C) P_e^{(n)}(C) = \sum_C P(C) \frac{1}{2^{nR}} \sum_{w=1}^{2^{nR}} \lambda_w(C) = \frac{1}{2^{nR}} \sum_C P(C) \sum_{w=1}^{2^{nR}} \lambda_w(C)$$

Na osnovu simetrije u konstrukciji koda vjerovatnoća greške ne zavisi od posmatranog indeksa koji je poslat tako da mi u opštem slučaju možemo posmatrati w=1. Definišimo događaj:

$$E_i = \{(X^{(n)}(i), Y^n) \in A_\varepsilon^{(n)}\} \text{ za } i \in \{1, 2, ..., 2^{nR}\}$$

Dakle, događaj E_i se pojavljuje kada je Y^n združena tipična sekvenca sa i-tom kodnom riječi. Do greške dolazi kada:

- E_1^c se pojavljuje: Y^n nije združena tipična sekvenca sa prvom kodom riječju i
- $E_2 \cup E_3 \cup ... \cup E_{2^{nR}}$ druga kodna riječ je tipično združena sa prvom kodnom riječju.

Neka $P(\varepsilon)$ označava $\Pr(\varepsilon \mid W = 1)$. Imamo da je:

$$P(\varepsilon) = P(E_1^c \cup E_2 \cup E_3 \cup ... \cup E^{2^{nR}}) \leq P(E_1^c) + \sum_{i=2}^{2^{nR}} P(E_i)$$

Mi možemo koristiti osobinu tipične sekvence po asimptotskoj ekviparticionoj teoremi:

$$P(E_1^c) \leq \varepsilon$$

za n dovoljno veliko. Kako je vjerovatnoća da su Y^b i $X^n(i)$ združeno tipične (za $i \neq 1$) je $\leq 2^{-nI(X;Y)-3\varepsilon}$. Odavde slijedi:

$$P(\varepsilon) \leq \varepsilon + \sum_{i=2}^{2^{nR}} 2^{-n(I(X;Y)-3\varepsilon)} = \varepsilon + (2^{nR} - 1) e^{-n(I(X;Y)-3\varepsilon)} \leq \varepsilon + 2^{-n(I(X;Y)-3\varepsilon-R)} \leq 2\varepsilon$$

za n dovoljno veliko i $R<I(X;Y)-3\varepsilon$. U stvari ako $R<I(X;Y)$ mi možemo izabrati ε i n tako da je vjerovatnoća greške usrednjena preko kodnih knjiga i kodnih riječi manja od 2ε. Finalni koraci u ovoj teoremi su:

1. U dokazu izabrati $p(x)$ da je to distribucija koja dostiže kapacitet. Tada $R<I(X;Y)$ može biti zamjenjena sa uslovom dostižnosti $R<C$.

2. Pošto je prosječna vjerovatnoća greške usrednjena preko svih kodnih knjiga mala tada postoji najmanje jedna kodna riječ C^* sa malom prosječnom dužinom greške $P_e^n(C^*) \le e\varepsilon$. Ovdje nijesmo diskutovali proceduru za njeno traženje.

3. Uočimo da bolja polovina kodnih riječi mora imati vjerovatnoću greške manju od 4ε (u suprotnom mi ne bi bili u prilici da dobijemo prosjek manji od 2ε). Možemo odbaciti goru polovinu kodnih riječi što nam daje brzina prenosa

$$R' = R - \frac{1}{n}$$

što je asimptotski zanemarljivo. Dakle, možemo zaključiti da za najbolju kodnu knjigu maksimalna vjerovatnoća greške je $\le 4\varepsilon$.

Ovo je kraj ove komplikovane teoreme. Zapamtite da se u praksi rijetko koriste tzv. slučajni kodovi već da se iz raznih razloga pristupa dizajnu praktično prihvatljivih kodova.

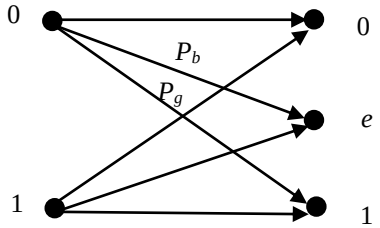
Napomena. U [9] postoji alternativni (mada sličan) dokaz ove teoreme. Pogledajte i uporedite.

5.3. Dokazati da je kod binarnog simetričnog kanala sa vjerovatnoćom pogreške u prenosu P_g i vjerovatnoćom brisanja simbola P_b (ovo je vjerovatnoća da na osnovu poslatog simbola nećemo moći donijeti odluku o primljenom) kapacitet jednak:

$$C = (1-P_g-P_b)\log\left[\frac{2(1-P_b-P_g)}{1-P_b}\right] + P_g \log\left[\frac{2P_g}{1-P_b}\right] \text{bita/simbolu}$$

Da li se ovdje može koristiti relacija za slabosimetrične kanale?

Rješenje: Posmatrajmo predmetni kanal:



Matrica tranzicije za ovaj kanal je:

$$\begin{bmatrix} 1-P_g-P_b & P_b & P_g \\ P_g & P_b & 1-P_g-P_b \end{bmatrix}$$

Međusobna informacija se sada može sračunati na neki pogodan način a u dosadašnjim izvođenjima smo vidjeli da je najpogodniji način da krenemo od relacije $I(X;Y)=H(Y)-H(Y|X)$

Međusobna informacija je jednaka:

$$\begin{aligned} H(Y \mid X) &= P[-(1-P_b-P_g)\log_2(1-P_b-P_g) - P_g \log_2 P_g - P_b \log_2 P_b] \\ &\quad + (1-P)[-(1-P_b-P_g)\log_2(1-P_b-P_g) - P_g \log_2 P_g - P_b \log_2 P_b] = \\ &= -(1-P_b-P_g)\log_2(1-P_b-P_g) - P_g \log_2 P_g - P_b \log_2 P_b \end{aligned}$$

Da bi odrediti entropiju događaja Y moramo da znamo vjerovatnoće pojedinih simbola. Događaj 0 se dešava sa vjerovatnoćom: $P(1-P_g-P_b)+(1-P)P_g$ dok se događaj 1 dešava sa vjerovatnoćom $(1-P)(1-P_g-P_b)+PP_g$ dok je vjerovatnoća stanja brisanja P_b. Dakle, sada možemo zapisati da je entropija $H(Y)$ jednaka:

$$H(Y)=-[P(1-P_g-P_b)+(1-P)P_g]\log_2[P(1-P_g-P_b)+(1-P)P_g]-$$
$$-[(1-P)(1-P_g-P_b)+PP_g]\log_2[(1-P)(1-P_g-P_b)+PP_g]-P_b\log_2 P_b$$

Međusobna informacija je jednaka:

$$I(X;Y)=H(Y)-H(Y|X)=$$
$$-[P(1-P_g-P_b)+(1-P)P_g]\log_2[P(1-P_g-P_b)+(1-P)P_g]-$$

$$-[(1-P)(1-P_g-P_b)+PP_g]\log_2[(1-P)(1-P_g-P_b)+PP_g]+(1-P_b-P_g)\log_2(1-P_b-P_g)+P_g\log_2 P_g$$

Međusobna informacija je očigledno funkcija od P tako da diferenciranjem po P i izjednačavanjem sa 0 dobijamo:

$$-[(1-P_g-P_b)-P_g]\log[P(1-P_g-P_b)+(1-P)P_g]-(1-P_g-P_b-P_g)$$
$$-[-(1-P_g-P_b)+P_g]\log[(1-P)(1-P_g-P_b)+PP_g]-(-(1-P_g-P_b)+P_g)=0$$

Poslije trivijalnih transformacija izraz se svede na:

$$\log[P(1-P_g-P_b)+(1-P)P_g]-\log[(1-P)(1-P_g-P_b)+PP_g]=0$$

Ovo dalje znači da argumenti logaritma moraju biti jednaki što konačno dovodi do toga da je jednakost zadovoljena za P=1/2. Dakle, dobijamo da je kapacitet jednak:

$$C=I(X;Y)=-[0.5(1-P_g-P_b)+0.5P_g]\log_2[0.5(1-P_g-P_b)+0.5P_g]-$$
$$-[0.5(1-P_g-P_b)+0.5P_g]\log_2[0.5(1-P_g-P_b)+0.5P_g]+(1-P_b-P_g)\log_2(1-P_b-P_g)+P_g\log_2 P_g$$
$$=-(1-P_b)\log_2[0.5(1-P_b)]+(1-P_b-P_g)\log_2(1-P_b-P_g)+P_g\log_2 P_g$$

Prvi član u izrazu možemo zapisati kao:

$$-(1-P_b)\log_2[0.5(1-P_b)]=-(1-P_b-P_g)\log_2[0.5(1-P_b)]-P_g\log_2[0.5(1-P_b)]$$

Sada možemo grupisati po dva člana izraza za kapacitet tako da dobijamo:

$$C=(1-P_b-P_g)\log_2\frac{2(1-P_g-P_b)}{1-P_b}+P_g\log_2\frac{2P_g}{1-P_b}$$

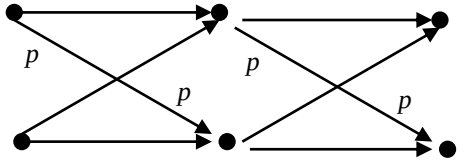
čime je dokaz okončan.

5.4. Izvedite (na osnovu relacije za slabosimetrične kanale) formulu za kapacitet opšteg simetrično kanala kod koga je vjerovatnoća greške uniformno raspodijeljena na sve simbole.

Rješenje: Pretpostavimo da imamo vjerovatnoću greške P koja je uniformno raspodjeljenja na sve simbole odnosno na N-1 simbol koji može da predstavlja pogrešku. U tom slučaju kapacitet kanala je dat relacijom:

$$C=\log_2 N+(N-1)P/(N-1)\log_2[P/(N-1)]+(1-P)\log_2(1-P)=$$
$$=\log_2 N+P\log_2[P/(N-1)]+(1-P)\log_2(1-P)=$$
$$\log_2 N-P\log_2(N-1)-H(P)$$

5.5. Na slici je prikazana kaskada dva simetrična binarna kanala. Odrediti transmisionu matricu rezultujućeg kanala, a zatim odrediti transmisionu matricu za kaskadu binarnih simetričnih kanala proizvoljne dužine u funkciji od broja kanala u kaskadi.



Rješenje: Kapacitet ovog kanala može se opisati formulom za slabosimetrične kanale pošto je u pitanju slabosimetrični kanal:

$$C=1-H(p_e)$$

gdje je p_e ekvivalentna vjerovatnoća greške ovog sistema. Posmatrajmo jednostavan eksperiment. Ako šaljemo 1 koja je vjerovatnoća da primimo 0. Potrebno je da napravimo grešku u prvom dijelu sistema p i da prenesemo uspješno simbol u narednoj kaskadi $(1-p)$ ili da u prvoj kaskadi napravimo tačan prenos $(1-p)$ a u drugoj pogrešan p. Dakle, ekvivalentna vjerovatnoća pogreške je:

$$p_e=2p(1-p)$$

Stoga je kapacitet jednak:

$$C=1-H(2p(1-p))$$

5.6. Data je transmisiona matrica binarnog kanala:

$$\mathbf{P}=\begin{bmatrix} 2/3 & 1/3 \\ 1/10 & 9/10 \end{bmatrix}$$

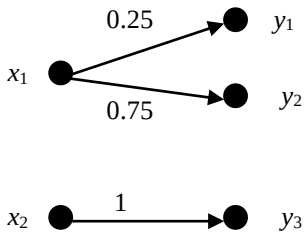
Odrediti kapacitet ovakvog kanala, odgovarajuće entropije, kao i sve vjerovatnoće (vjerovatnoću izlaznih simbola, uslovne vjerovatnoće i združene vjerovatnoće).

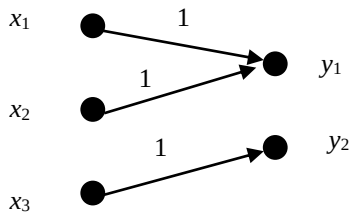
Rješenje: U pitanju je nesimetrični binarni kanal kod kojega je vjerovatnoća greške za jedan simbol 1/3 dok je za drugi simbol 1/10. Na osnovu formule koja je izvedena za slučaj binarnog nesimetričnog kanala slijedi da je kapacitet jednak:

$$C=\log_2\left[1+2^{\frac{H(p_0)-H(p_1)}{1-p_0-p_1}}\right]-\frac{H(p_0)(1-p_1)-H(p_1)p_0}{1-p_0-p_1}$$

Uvrštavajući $H(0.1)=0.469$ dok je $H(2/3)=0.9183$ dobijamo $C=0.2676$. Tačnost ovog izraza se može provjeriti numeričkim simulacijama kao što je urađeno za ternarni kanal u zadatku 9 koji je dat u nastavku.

5.7. Za kanale koji su dati na ilustraciji odrediti prenosnu (transmisionu ili kanalnu matricu) i odrediti kapacitet.





Rješenje: Prenosne matrice ova dva koda su:

$$\begin{bmatrix} 0.25 & 0.75 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Za prvi kanal:
Pretpostavimo da je vjerovatnoća pojavljivanja prvog simbola p dok je vjerovatnoća pojavljivanja drugog simbola $1-p$. Vjerovatnoće pojavljivanja simbola na strani prijema su: $[0.25p,\ 0.75p,\ 1-p]$. Združene vjerovatnoće simbola na ulazu i izlazu su $[0.25p,\ 0.75p,\ 1-p,\ 0,\ 0,\ 0]$

Međusobna informacija je jednaka:

$$I(X;Y) = H(X) + H(Y) - H(X,Y) = H(X) = H(p)$$

Dakle zaključujemo da je kapacitet ovog kanala C=1bit/simbolu.

Za drugi kanal pod pretpostavkom da su vjerovatnoća prvog i drugog simbola p, q i $1-p-q$. Simboli na izlazu se pojavljuju sa vjerovatnoćama $p+q$ i $1-p-q$. Združena entropija ovih simbola su $[p,\ q,\ 1-p-q]$. Međusobna informacija je u ovom slučaju:

$$I(X;Y) = H(X) + H(Y) - H(X,Y) = H(Y) = H(p+q)$$

Ponovo je i u ovom slučaju kapacitet maksimalan za $p+q=0.5$ i iznosi C=1bit/simbolu.

5.8. Pretpostavimo da imamo kod sa kodnom riječju dužine 3 gdje je jedan bit informacioni. Kod je u stanju da ispravi jednu pogrešku. Razmotriti vjerovatnoću grešku u kodu u zavisnosti od vjerovatnoće greške u kanalu i uporedite sa rezultatima koja daje kodna teorema. Ponoviti posao za slučaj kada imate kod sa 7 bita od kojih je 4 informacionih sa mogućnošću ispravke jedne pogreške. Zatim ponovite postupak ako imate kod sa 15 bita od kojih je 11 informacionih sa mogućnošću ispravke jedne pogreške. Nakon toga obaviti proceduru ako imate pet bita a samo jedan informacioni ali kod može da ispravi do dvije pogreške. Ponoviti proceduru nad kodom koji ima 15 bita od kojih je 7 informacionih i koji može da ispravi do 2 pogreške. Konačno uzeti slučaj 7 bita od kojih je jedan informacioni i koji može da ispravi tri pogreške i koda sa 15 bita od kojih je samo tri informaciona ali kod ima mogućnost da ispravi do tri pogreške.

Rješenje. Kodni odnosi u navedenim slučajevima su $R_{1,3}=1/3$, $R_{4,7}=4/7$, $R_{11,15}=11/15$, $R_{1,5}=1/5$, $R_{7,15}=7/15$, $R_{1,7}=1/7$ i $R_{5,15}=5/15=1/3$ gdje indeksi kodnog odnosa označavaju broj informacionih i ukupan broj bita u kodu. Kapacitet binarnog simetričnog kanala je $C=1-H(p)$ a kako po II Šenonovoj teoremi treba da važi $R\leq C$ da bi imali vjerovatnoću kodne pogreške koja teži nuli. Granične verzije ove vjerovatnoće su $p_{1,3}=0.1739$, $p_{4,7}=0.0876$, $p_{11,15}=0.0454$, $p_{1,5}=0.2430$, $p_{7/15}=0.1214$, $p_{1,7}=0.2812$ i $p_{5,15}=0.1739$. U slučaju kodova koji mogu da isprave jednu, dvije i tri pogreške imamo da su vjerovatnoće pogreške u tim situacjiama:

$$P_1=1-(1-p)^n-np(1-p)^{n-1}$$
$$P_2=1-(1-p)^n-np(1-p)^{n-1}-n(n-1)p^2(1-p)^{n-2}/2$$
$$P_3=1-(1-p)^n-np(1-p)^{n-1}-n(n-1)p^2(1-p)^{n-2}/2-n(n-1)(n-2)p^3(1-p)^{n-3}/6$$

Provjerite da li u navedenom slučaju dobijamo da je vjerovatnoća greške u kanalu za pomenute kodove za vrijednosti vjerovatnoće greške po jednom bitu u skladu sa kodnom teoremom i ako nije zbog čega.

Napomena. Dobićete da postoji greška i za uslove kada smatramo da je teorema "zadovoljena" zato što to nije slučaj odnosno ne važi fundamentalni uslov predmetne teoreme a to je da dužina kodne riječi teži beskonačno. Provjeriti što se dobija kod koda koji ima 1023 ukupan broj bita od kojih je 1013 informacionih. Provjerite što se dešava u ovom slučaju i uporediti sa slučajevima sa kraćim riječima.

5.9. Dokazati osobine kapaciteta kanala (ima ih ukupno pet).

Rješenje je dato u okviru lekcije.

5.10. Odrediti kapacitet kanala kanal koji prenosi ternarne poruke. Vjerovatnoća prenosa ispravnog simbola je P dok je vjerovatnoća prenosa pogrešnog Q (po $Q/2$ da će biti prenijeta ostala dva simbola).

Rješenje: Tranziciona matrica u ovom slučaju je:

$$\begin{bmatrix} P & Q/2 & Q/2 \\ Q/2 & P & Q/2 \\ Q/2 & Q/2 & P \end{bmatrix}$$

Stoga se na osnovu teoreme za slabosimetrične kanale lako pokazuje da je kapacitet ovog kanala:

$$C=\log 3-H(\mathbf{r})=\log 3+P\log P+2(Q/2)\log(Q/2)=$$
$$=\log 3+P\log P+Q\log Q-Q=\log 3-H(P)-Q$$

5.11. Odredite kapacitet binarnog nesimetričnog kanala.

Rješenje: Urađen u sklopu predavanja.

5.12. Data je transmisiona matrica kanala:

$$\begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.3 & 0.6 & 0.1 \\ 0.05 & 0.15 & 0.8 \end{bmatrix}$$

Kakav je kanal u pitanju? Koliki je kapacitet kanala. Ako ne možete da odrediti u zatvorenom obliku odredite numerički.

Rješenje: Očigledno riječ je o nesimetričnom kanalu. U procesu rješavanja ovdje je vjerovatno bolje uposliti malo numerike. Pretpostavićemo da se simboli na ulazu pojavljuju sa vjerovatnoćama p_1, p_2, $1-p_1-p_2$. Cilj nam je odrediti p_1 i p_2 takvo da kapacitet je maksimalan. Umjesto da krenemo težim putem kojim bi ovo računali ručno upotrijebili smo jednostavni MATLAB kod:

```
clear
K=[0.7 0.2 0.1; 0.3 0.6 0.1; 0.05 0.15 0.8];
Cmax=0;p1m=0;p2m=0;
for p1=0:0.001:1
    for p2=p1+0.001:0.001:1
        [p1,p2]
        if(p1+p2<=1 & 1-p1-p2>p2)
            p3=1-p1-p2;
            P1=p1*K(1,1)+p2*K(2,1)+p3*K(3,1);
            P2=p1*K(1,2)+p2*K(2,2)+p3*K(3,2);
            P3=p1*K(1,3)+p2*K(2,3)+p3*K(3,3);
            %%%Ovo su vjerovatnoce za Y pa je entropija HY
            HY=-P1*log2(P1)-P2*log2(P2)-P3*log2(P3);
            Hxy=-p1*sum(K(1,:).*log2(K(1,:)))-…
            p2*sum(K(2,:).*log2(K(2,:)))-p3*sum(K(3,:).*log2(K(3,:)));
            C=HY-Hxy;
```

```
            if(C>Cmax)
                Cmax=C;
                p1m=p1;
                p2m=p2;
            end
        end
    end
end
```

Programska realizacija je daleko od optimalne međutim krajnje je jednostavna. Usvajamo vjerovatnoće prva dva simbola a zatim na osnovu njih računamo vjerovatnoću trećeg simbola kao $1-p_1-p_2$. Zatim računamo kolika je vjerovatnoća simbola na izlazu iz kanala (P_1, P_2 i P_3) a to koristimo da bi sračunali $H(Y)$ dok za računanje $H(Y|X)$ nam je dovoljna matrica tranzicije i vjerovatnoće p_1, p_2 i p_3. Kada sračunamo međusobnu informaciju u slučaju da je veća od dotadašnjeg maksimuma proglašavamo je tekućim maksimumom i pamtimo. Konačni rezultati su:

$$C=0.4916$$

Koje se dobija za $p_1=0.2760$ i $p_2=0.2770$.

5.13. Na ulaz kanala moguće su poruke –1, 0 i 1. Kanal vrši kvadriranje ulazne poruke i nema brisanja ni grešaka. Čemu je jednak kapacitet kanala.

Rješenje: Pošto nema grešaka u kanalu $H(Y|X)=0$ jer sa znanjem X znamo i Y. Stoga je međusobna informacija jednaka:

$$I(X;Y)=H(Y)$$

Ako su vjerovatnoće ulaznih simbola redom $\{p(-1)=q,\ p(0)=p,\ p(1)=1-p-q\}$ tada je vjerovatnoća simbola 0 i 1 na izlazu: $\{P(0)=p(0)=p,\ P(1)=p(-1)+p(1)=1-p\}$ pa je entropija $H(Y)$ jednaka $H(p)$ koje maksimum postiže za $p=1/2$ i iznosi:

$$I(X;Y)=1\text{bit}$$

Napomena. Kao dodatni izazov pokušati da dokažete Teoremu iz Glave 5.6.

Poglavlje VI
OSNOVE KODIRANJA KANALA

6.1. Kratko podsjećanje na neke kodove

Binarni kod je definisan na alfabetu 0 i 1, oktalni kod je definisan na alfabetu 0, 1, ..., 7, heksadekadni kod je definisan na alfabetu 0, 1, 2, ..., 9, a, b, c, d, e, f. Ovi kodovi imaju jedinstvenu jednostavnu međusobnu vezu i vezu sa dekadnim zapisom broja. Pored toga dekadni brojevi se mogu predstavljati u BCD kodu. Zapis karaktera je obezbjeđen preko ASCII tabele. Uočimo da je ASCII tabela kod kod kojeg su svi kodni simboli jednake dužine. Postoji, recimo, Morzeov kod, kod kojeg se slova engleske abecede koja se češće pojavljuju kodiraju sa manjim brojem bita (tačkica i crtica) od onih koja se rjeđe pojavljuju. Bilo koji kod koji se prenosi preko komunikacionog kanala podložan je uticaju smetnji i očigledno da u kod koji je prošao kodiranje izvora radi otklanjanja redudancije mora se vratiti malo redudancije da bi se omogućio pouzdaniji prenos preko komunikacionog kanala.

6.2. Kodovi za detekciju greške

Prilikom prenosa poruka usljed šumova u kanalu dolazi do grešaka. Stoga se poruke moraju dodatno kodirati kako bi se obezbjedile detekcija greške. Najjednostavniji način da se detektuje greška u binarnoj poruci je na osnovu broja jedinica u njoj. Prebroje se jedinice i poruci se doda još jedan simbol (jedinica ili nula) tako da je broj jednica u poruci paran ili neparan po dogovoru. U kodu dužine n, n-1 bit nosi informacioni sadržaj, dok jedan bit se koristi za detekciju greške. Kod ne može da detektuje dvije greške, kao ni bilo koji paran broj grešaka. Ako je p vjerovatnoća greške na jednom bitu i ako je n mnogo manje od 1/p i ako je greška na svakom bitu nezavisna od greške na drugim bitovima, vjerovatnoća da će se na tačno jednom bitu u poruci pojaviti greška je $np(1\text{-}p)^{n\text{-}1}$. Vjerovatnoća dvostruke greške je $n(n\text{-}1)p^2(1\text{-}p)^{n\text{-}2}/2$. U teoriji se češće koristi kodovi sa parnim brojem jedinica, ali se u praksi češće koriste kodovi sa neparnim brojem, tako da "sve nule" nije ispravna poruka. Zapamtite da većina procesora ima funkciju koja broji jedinice u registrima tako da su u tom smislu ovi kodovi veoma pogodni. Postoje specijalni slučajevi koda sa provjerom parnosti koji se koriste u praksi. To su kodovi 2 od 5 i 3 od 7 gdje je broj jedinica (i nula) fiksiran na 2 ili na 3. Ovi kodovi mogu da vrše elementarno ispravljanje grešaka. Duge poruke se obično dijele na kraće i na svako n bita dodaje se po jedan bit za kontrolu greške. Model za određivanje vjerovatnoće greške podrazumjeva: svaki bit ima istu vjerovatnoću greške, greške na pojedinim pozicijama su međusobno nezavisne. Vjerovatnoća pojave k grešaka u nekoj poruci sa n bita je jednaka k-tom članu u binarnom razvoju:

$$1=[(1-p)+p]^n=(1-p)^n+np(1-p)+\frac{n(n-1)}{2}p^2(1-p)^2+\ldots+\binom{n}{k}p^k(1-p)^{n-k}+\ldots+p^n$$

Vjerovatnoća za pojavu parnog broja grešaka je:

$$\sum_{m=1}^{\lceil n/2\rceil}\binom{n}{2m}p^{2m}(1-p)^{n-2m}$$

Najčešće su samo prvi i drugi član relevantni za računanje, jer su ostali suviše mali.

Kod ASCII koda imamo slučaj da je kodni odnos jednak

$$R=\frac{n-1}{n}$$

Kod koda 2 od 5 kodni odnos je jednak

$$R = \frac{\log_2 10}{5}$$

jer se sa predmetnim kodom može kreirati $\binom{5}{2} = 10$ kodnih riječi. Kodni odnos koda 3 od 7 se na sličan način može pokazati da iznosi:

$$R = \frac{\log_2 \binom{7}{3}}{7} = \frac{\log_2 35}{7}$$

Odredimo sada vjerovatnoće kada predmetni kodovi (2 od 5 i 3 od 7) ne rade dobro. Pretpostavimo da je vjerovatnoća greške po bitu p i da je u pitanju i.i.d. proces. Kodovi rade dobro kada nema nijedne greške u kodnoj riječi. Dakle, sa vjerovatnoćama $(1\text{-}p)^5$ i $(1\text{-}p)^7$ u ova dva koda se ne javlja nijedna pogreška. Oba koda su u stanju da prepoznaju bilo koji neparan broj pogreški. Što se tiče parnog broja pogreški situacija je znatno komplikovanija jer ako se pogreške dogode recimo na dvije nule onda ćemo imati nijednu nulu (ako je 2 nule od 5 bita) i znaćemo da se greška dogodila dok ako se jedna pogreška dogodi na jednoj nuli a jedna na jednoj jedinici onda kod neće biti u stanju da uoči pogrešku. Slično i kod četiri odnosno kod šest pogreški. Svaka pojedinačna situacija sa dvije pogreške u kodu 2 od 5 ima vjerovatnoću $p^2(1\text{-}p)^3$ i ukupan broj takvih situacija je $\binom{5}{2} = 10$ međutim samo šest situacija je onih koje nama "škode" odnosno kada greška nije prepoznata $\binom{2}{1}\binom{3}{1} = 6$. Ako imamo četiri pogreške kod koda onda se greška događa kada imamo dvije greške na jednom simbolu i dvije na drugom. Od ukupno $\binom{5}{4} = 5$ situacija sa četiri pogreške $\binom{2}{2}\binom{3}{2} = 3$ nam ne odgovaraju. Dajmo sada konačne formule za tražene vjerovatnoće kod ovih kodova:

Vjerovatnoća da se kod ovih kanala ne desi pogreška je:

$$(1-p)^5 \qquad\qquad (1-p)^7$$

Vjerovatnoća da se dogodi pogreška i da je kod detektuje je respektivno:

$$5p(1-p)^4 + 4p^2(1-p)^3 + 10p^3(1-p)^2 + 2p^4(1-p) + p^5$$
$$7p(1-p)^6 + 9p^2(1-p)^5 + 35p^3(1-p)^4 + 17p^4(1-p^3) + 21p^5(1-p)^2 + 3p^6(1-p) + p^7$$

Vjerovatnoća da se dogodi pogreška i da kodovi nisu u stanju da je detektuju su:

$$6p^2(1-p)^3 + 8p^4(1-p)$$
$$12p^2(1-p)^5 + 18p^4(1-p^3) + 4p^6(1-p)$$

Pokušajte da vizuelizujete ove vjerovatnoće za p u granicama od [0,0.2] i to u logaritamskoj skali kako bi se preciznije vidjelo što se dešava na nižim vjerovatnoćama greške koje su za nas od većeg značaja.

Do sada smo govorili o slučaju da ima jedna greška koja je nezavisna od ostalih grešaka. Međutim, u prenosu podataka posebno kada je u pitanju pojava grmljavina može se dogoditi da veliki broj bita zaredom bude promjenjen (obično postavljeni svi na nulu ili svi na jedinicu). To se kontroliše provjerom parnosti po riječima. Npr. neka imamo poruke 0100, 0110, 1100 i 1111. Ovoj poruci se dodaje jedna kontrolna poruka koja predstavlja bitove parnosti po kolonama:

0100
0110
1100
1111
0001

Sada se jednostavno detektuju slučajevi "burst" promjena na bitovima. Sljedeći problem je kada ljudi rukuju kodovima. Naime, ljudi često prave uobičajene greške zamjene nekoliko cifara ili ponavljanja neke cifre u npr. matičnom broju. Za kodiranje ovakvih poruka koriste se težinski kodovi. Ovi kodovi zavise od nekog prostog broja koji se mora izabrati. Na osnovu promjene jednog kodnog simbola lako se može zaključiti o poziciji na kojoj se greška dogodila.

Posmatrajmo sledeći primjer. Neka se želi prikazati cifre od 0 do 9, slova engleske abecede od A do Z i znak blanko. Neka se svim simbolima dodjele vrijednosti redom od 0 (0=0, ..., 9=9, 10=A, 11=B, ..., 35=Z), do 36=blanko. Ovoj poruci pridodajemo broj 37 koji je prvi veći prosti broj od broja karaktera u poruci. Odredimo dodatni karakter koji će se koristiti za kodiranje za poruku A6 7:

	suma	suma sume
A=10	10	10
6=6	16	26
bl=36	52	78
7=7	59	137
x=x	59+x	196+x

Podjelimo 196+x sa 37 i odredimo ostatak. Ostatak je 11+x. Dakle, da bi suma sume ove poruke bila djeljiva sa 37 mora biti x=26 (karakter *Q*). Pojavu greške možemo jednostavno otkriti (npr. umjesto A se primi B, a umjesto 6 se primi 5) i pored toga što je suma poruke ista dobija se razlika u sumi sume, takođe i zamjena dva karaktera daje razliku u sumi sume. Ako je greška samo na jednom karakteru i to samo za "jedan" možemo otkriti i poziciju na kojoj do greške došlo. Posmatrajmo kod za jedinstveni matični broj koji se obračunava na osnovu ostatka od 11. Aritmetika po modulima (ostacima pri djeljenju) brojeva se često koristi u teoriji kodova. Uvijek je djelioc po kome se traži ostatak prost broj najčešće 2, ali može biti i neki drugi prost broj. Oznake koje se koriste su:

$$a = a' \bmod m$$

Važe sljedeća pravila: $a+b=a'+b' \bmod m$. Dalje, $ab=a'b' \bmod m$. Primjer a=15 b=12 m=10

$$15+12=5+2 \bmod 10$$

$$15 \cdot 12=5 \cdot 2 \bmod 10$$

Važan primjer težinskog koda je međunarodni jedinstveni serijski broj za knjige (ISBN). Ima deset cifara od kojih 9 predstavlja zemlju, izdavača i broj same knjige dok je deseta cifra broj koji predstavlja ostatak težinskog koda pri dijeljenju sa 11. Npr. koja se cifra dodaje na:

0-1321-2571- (tirei nemaju značaj sa stanovišta provjere koda)

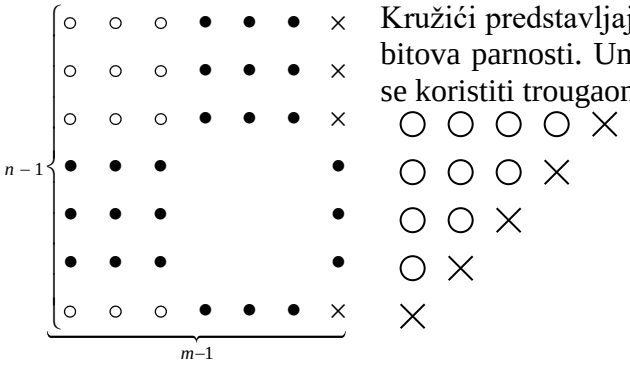
0	1	4	6	7	9	14	21	22	22+x
0	1	5	11	18	27	41	62	84	106+x

Da bi 106+x bilo djeljivo sa 11 potrebno je da x=4. U slučaju da je potreban broj koji je jednak 10 zapisuje se kao jedna cifra X.

Danas se umjesto ovog koda koristi njegova varijanta koja se naziva ISBN-13. Kod ovoga koda (važi od 2007-me godine) podaci o izdanju su u prvih 12 cifara dok su podaci o kontroli sakriveni u trinaestoj cifri. Kod ovoga koda kontrolna cifra se određuje tako što se prvo sračuna težinska suma prvih 12 cifara koja se dobija tako što se vrijednosti na neparnim pozicijama množe sa 1 a one na parnim pozicijama sa 3. Dodatna trinaesta cifra se dodaje tako da ostatak pri dijeljenju sume sa 10 bude 0. Ovdje nemamo provjeru sa 11 odnosno ne može se pojaviti specijalna kontrolna cifra X. Konačno postoji još jedan veoma poznati kod za detekciju pogreške kod dekadnih kodova a to je jedinstveni matični broj JMBG. Kod ovoga koda imamo 13 cifara od kojih prvih 7 predstavlja datum rođenja u formatu ddmmggg (kod godine se štedi na prvoj cifri). Sledeće dvije cifre je šifra mjesta rođenja naredne tri cifre predstavljaju broj djeteta rođenog toga dana u tom mjestu (0-499 za dječake i 500-999 za djevojčice) i posljednja trinaesta cifra je kontrolna. Prilikom težinskog sumiranja prvih 6 cifara su množene redom sa 7, 6, 5, 4, 3 i 2 dok su drugih šest cifara ponovo množene sa istom kombinacijom (7, 6, 5, 4, 3 i 2). Posljednja cifra je dodavana tako da bude zadovoljeno djeljenje sa ostatkom po prostom broju 11. Ako se desi da je potreban broj za ispunjavanje traženog ostatka 10 postavlja se 0 isto kao i kada je potrebna 0 (ne uvodi se X kao što je slučaj sa ISBN u osnovnoj varijanti).

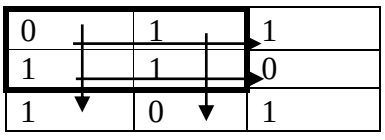
6.3. Kodovi za korekciju greške

Detekcija greške često nije dovoljna, već bi bilo neophodno imati mogućnost korekcije te greške. Najjednostavniji sistem za korekciju greške je slanje jedne poruke tri puta praćeno sa sistemom glasanja (primljen je onaj bit koji je dobio većinu glasova). Ovaj tip kodiranja se naziva i kodiranje sa ponavljanjem ili repetitivni kod. Da bi se korekcija greške vršila mora se u poruci dodati mala redudancija koja će da služi za korekciju. Osnovni razlog za postojanja grešaka su problemi u prenosu poruka na daljinu komunikacionim kanalima. Međutim, svi tipovi magnetnih i optičkih zapisa su podložni kvarenju tokom vremena, tako da se i u ovim slučajevima mora voditi računa o potrebi za korekciju greške. Ne uzimajući u obzir "sistem glasanja" najjednostavniji sistem za otkrivanje grešaka je tzv. pravougaoni kod. Pravougaoni kod podrazumjeva dodavanje bita parnosti po kolonama i po vrstama poruka. Ako se u određenoj poruci pojavi samo jedna greška onda se može detektovati u kojoj se koloni i vrsti pojavila (detektovati pozicija greške). Kako u binarnom sistemu ako je primljen bit i poznato je da je konkretni bit pogrešan to znači da je ispravan bit njegova negacija.



Kružići predstavljaju pozicije informacionih bitova dok je x pozicija bitova parnosti. Umjesto pravougaonih kodova kao efikasniji mogu se koristiti trougaoni kodovi:

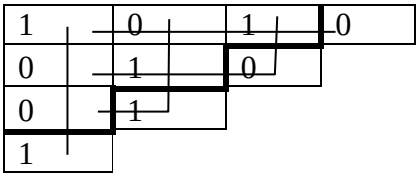
Posmatrajmo ilustraciju formiranja pravougaonog koda (9,4). Ovdje prva cifra podrazumijeva ukupan broj bita dok je druga cifra broj korisnih informacionih bita.



Strelice ukazuju na korespondentne bitove parnosti. Često postavljano pitanje je posljednje devete cifre i njene upotrebe. Prvo može se suštinski izostaviti ali ako se koristi onda je to kodni simbol koji kontroliše čitavu tabelu. Jednostavno se može pokazati da ako je broj jedinica u informacionim bitima paran da je onda broj jedinica i u kontrolnim ciframa po vertikali paran a isto tako je i broj jedinica u kontrolama po horizontali. Ako je broj jedinica u in-

formacionim bitima neparan onda je broj jedinica u kontolnim bitima po vertikali isto neparan baš kao i broj jedinica u kontrolnim bitima po horizontali. Dakle, ukupan broj jedinica je u provjeri parnosti u za čitavu tabelu skladu sa brojem jedinica u informacionim bitima (jer su kontrole po kolonama i vrstama dale paran broj jedinica).

Posmatrajmo primjer trougaonog koda (10,6) prikazanom u tabeli.



Svaki bit parnosti kontroliše svoju kolonu i vrstu. U slučaju da se greška dogodi na recimo drugom bitu prve kolone i da postane 1 onda će prvi bit parnosti koji se nalazi u prvoj vrsti indicirati grešku 1110 dok će to uraditi i treći bit parnosti koji se nalazi u trećoj vrsti (ovaj bit sada kontroliše 11 iz iste kolone i 0 iz iste vrsta a pošto je sam bit 1 to ukazuje na grešku). Presjek odnosno jedina pozicija koju kontrolišu oba bita parnosti koja su grešku indicirala.

Kod pravougaonog koda za blok [illegible] jedne greške u poruci. Redundancija je $(n^2+2n+1)/n^2=1+2/n+1/n^2$ dok kod trougaonog koda za kodiranje bloka veličine $1+2+...+n-1=n(n-1)/2$ bita je potrebno n bita pa je redudancija: $1+2/(n-1)$. Npr. za $n=5$ redudancija kod pravougaonog koda je 1.44 dok je kod trougaonog 1.33. Postavlja se sada pitanje koji bi kod bio optimalan sa stanovišta redudancije? Ako imamo prostorni kod predstavljen kockom za kodiranje bitova za provjeru nam je potrebno $3n-2$ pozicija ako je ukupna veličina kocke (uključujući bitove parnost n^3). Važno je zapamtiti da kod kodiranja u kocki je potrebno vršiti kodiranje po ravnima (provjera parnosti za čitavu ravan). Dakle, grubo redundancija je približno $3/n^2$ dodatih bitova. Porastom dimenzija dobija se sve manji i manji procenat neinformacionih bitova potreban za otkrivanje greške. Tako se za četvrtu dimenziju dobija $4/n^3$. Očigledno je da se najbolji rezultati dobijaju za najveći mogući prostor odnosno da za ukupno 2^n bita potrebno je $n+1$ bitova za provjeru parnosti. Ova analiza vodila je do definisanja Hammingovog koda. Posmatrajmo sledeći sistem sa 4 informaciona bita. Dakle, u ovom slučaju je $n=2$ i potrebno je 3 bita za korekciju greške. Znači ukupan broj bita u ovom slučaju je 7. Bitovi koji se koriste za korekciju greške se nazivaju **sindrom**. Bitovi sindroma se postavljaju na pozicije 2^k, $k=0,1,...,n$ u poruci. Ovaj slučaj je situacija kada su biti parnosti na pravilnim binarnim pozicijama. Prvi bit na poziciji $2^0=1$ predstavlja provjeru bita parnosti na neparnim pozicijama u poruci (pozicije 1,3,5 i 7). Drugi bit parnosti na poziciji $2^1=2$ predstavlja bitove parnosti na pozicijama 2, 3, 6 i 7, dok treći bit (pozicija $2^2=4$) predstavlja provjere parnosti na pozicijama 4, 5, 6 i 7. Pokažimo to na jednom primjeru. Neka je poruka koja se želi kodirati: 1011. Ovu poruku postavljamo u pozicije 2, 5, 6 i 7.

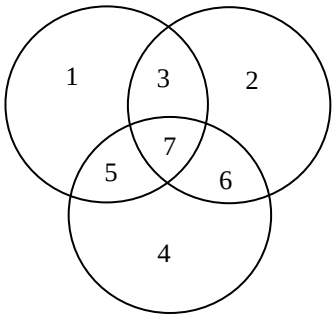
Pozicija	1	2	3	4	5	6	7
Poruka	-	-	1	-	0	1	1
Kodirana	0	1	1	0	0	1	1

Neka je primljena poruka sa greškom na trećem bitu:

Primljena:	0	1	0	0	0	1	1

Provjera parnosti daje da na pozicijama 1, 3, 5 i 7 nije ispunjena parnost upisujemo 1.
Provjera parnosti daje da na pozicijama 2, 3, 6 i 7 nije ispunjena parnost upisujemo 1.
Provjera parnosti po bitovima na pozicijama 4, 5, 6 i 7 je ispunjena i upisujemo 0.

Dakle, greška se javlja na poziciji $011_2=3_{10}$. Bitovi se čitaju u suprotnom redosljedu. Odnosno da budemo precizniji greška detektovana na četvrtom bitu ima težinu 4, greška na detektovana na drugom bitu ima težinu 2 dok greška identifikovana na prvom bitu ima težinu 1. Ovo se dosta lijepo može prikazati pomoću Venovog dijagrama.



Ovdje se sada jasno vidi da bit na poziciji 1 kontroliše bite na pozicijama 1, 3, 5 i 7, bit na poziciji 2 kontroliše bite na pozicijama 2, 3, 6 i 7 a bit na poziciji 4 kontroliše bite na pozicijama 4, 5, 6 i 7. Dalje, vidimo da biti 1, 2 i 4 sami sebe kontrolišu odnosno da ako se ne pojavi nijedna druga greška onda će pogreška na indicirana na pozicijama 1, 2 i 4 ukazivati da je pogrešan isključivo na kontrolnim bitima. Dalje, u presjecima dvije provjere parnosti nalaze se biti 3, 5 i 6 dok se u presjeku svih simbola nalazi simbol 7.

Postavlja se pitanje u generalnom slučaju proizvoljne dužine koje pozicije provjeravaju bitovi parnosti. Za ovo postoji jednostavna tehnika. Npr. u našem slučaju kada kodiranje se obavlja sa 7 bita možemo izvršiti binarni zapis brojeva od 1 do 7: 001, 010, 011, 100, 101, 110 i 111. Bit na poziciji 1 provjerava parnost na onim pozicijma, gdje je u ovoj sekvenci jedinica na posljednjem mjestu (mjestu bita najmanje važnosti), bit na poziciji 2 kontroliše pozicije, gdje je jedinica na narednom bitu a bit na poziciji 4 kontroliše one pozicije gdje je jedinica na prvom bitu (bitu najveće važnosti). Ako je poruka koja je dekodirana nakon provjere parnosti 000 nema greška u prenosu. Na prvi pogled ovaj sistem djeluje neefikasno (3 bita su redudancija u odnosu na 4 informaciona) ali u slučaju dužih sekvenci redudancija se značajno redukuje. Npr. za 1024 informacione poruke nam je neophodno svega 11 bita za provjeru parnosti. Naravno predloženi kod je samo jedna mogućnost. Postoje brojne ekvivalentne varijante. Postavlja se pitanje što se dešava kada imamo 8 bita informacionih bita u Hammingovom kodu. U tom slučaju ponovo postavljamo kontrolne bite na pozicijama koje su pravilno binarno raspoređene (1, 2, 4 i 8) a zatim raspoređujemo informacione bite na pozicijama 3, 5, 6, 7, 9, 10, 11, 12 i tu se zaustavljamo. Informacionih bita kojih nema na pozicijama 13, 14 i 15 možemo smatrati da su nule i u kodiranju i u dekodiranju odnosno možemo ih potpuno zanemariti.

Dobra osobina Hammingovog koda je činjenica da se veoma lako može povećati njegova fleksibilnost dodavanjem samo jednog bita parnosti. Na primjer, ako imamo kod sa 7 ukupno bita od kojih 4 informaciona (kod (7,4)) dodavanjem na poziciju 8 jednog bita parnosti koji provjerava sve bite. Ako se dogodi jedna pogreška onda ćemo je moći ispraviti putem prva tri bita parnosti dok će novododati bit indicirati da postoji jedna greška (ili eventualno neparan broj pogreški). Ako se dogode dvije pogreške prva tri bita parnosti će indicirati da postoji greška dok će poslednji bit indicirati da nema pogreški. Ovakva situacija indicira da su se dogodile dvije pogreške). Dakle, ovakav kod koji se obično naziva proširenim Hammingovim kodom može da ispravi jednu pogrešku i da detektuje dvije.

Primjer. Posmatrajmo sledećih 4 informaciona bita 1101. Kodirajmo ih sa proširenim Hammingovim kodom:

1 0 1 0 1 0 1 0

Ako se greška dogodi na petom bitu:

1 0 1 0 **0** 0 1 0

Prvi bit parnosti provjerava prvi, treći, peti i sedmi bit 1101 i detektuje grešku.
Drugi bit parnosti provjerava drugi, treći, šesti i sedmi bit 0101 i ne detektuje grešku.
Treći bit provjerava četvrti, peti, šesti i sedmi bit 0001 i detektuje grešku.

Pošto u riječ sada ima 3 jedinice znamo da se dogodila greška a pošto su je indicirali bitovi parnosti na pozicijama jedan i četiri zaključujemo da je greška na petoj poziciji.

Pretpostavimo sada da su se dogodile dvije pogreške na pozicijama četiri i pet:

1 0 1 **1** **0** 0 1 0

Prvi bit parnosti provjerava prvi, treći, peti i sedmi bit 1101 i detektuje grešku.
Drugi bit parnosti provjerava drugi, treći, šesti i sedmi bit 0101 i ne detektuje grešku.
Treći bit provjerava četvrti, peti, šesti i sedmi bit 1001 i ne detektuje grešku.
Pošto u riječ sada ima 4 jedinice znamo da su se dogodile dvije pogreške jer prva tri bita (odnosno preciznije samo prvi bit) ukazuje na pogrešku a konačni biti parnosti za čitavu riječ ne prepoznaje postojanje greške.

Za vježbu odredite vjerovatnoće da Hammingov i proširenі Hammingov kod prime ispravnu poruku, prime poruku sa jednom pogreškom i isprave je, da ne mogu da isprave pogrešku a kod proširenog Hammingovog koda i da se dogode dvije pogreške i da sistem indicira pogrešku.

Do sada smo opisali aritmetički pristup Hammingovom kodu. U literaturi se često daje i geometrijski pristup pomoću *n*-dimenzione geometrije. U ovom slučaju svaki bit je koordinata u *n*-dimenzionom prostoru i poruka koja se kodira je dužine *n*. Svaki čvor u datoj *n*-dimenzionoj kocki predstavlja potencijalno primljenu poruku. Jedna pogreška pomjera primljeni bit na poziciju nekog od susjeda (pod susjedima u ovom slučaju podrazumjevamo susjede po ivicama hiperkocke). Ako se sada podesi da je svaka legitimna pozicija na kocki udaljena za najmanje dva od naredne legitimne pozicije to se može lako detektovati koja je pozicije pogrešna. Ako je distanca (pod distancom podrazumjevamo kretanje duž ivica kocke) barem tri između pojedinih legitimnih pozicija tada greška na jednom bitu ostavlja da je primljena poruka bliža pravoj poruci nego bilo kojoj drugoj legitimnoj poruci. Distanca između pozicija se može shvatiti kao broj pozicija sa različitim bitovima između dvije poruke. Pravila koja važe kod distance su: Distanca između iste poruke je 0, distanca između poruka *x* i *y* je ista kao i distanca između *y* i *x*, važi pravilo trougla distanca - suma dvije stranice od *a* do *c* i od *c* do *b* je veća ili jednaka kao dužina treće stranice od *a* do *b*. Distanca se u ovom slučaju naziva Hammingovom distancom. Dokaz osobina distance za Hammingovu distancu je relativno jednostavan i može poći od jednog bita a za sve ostale bite važi u istom obliku tako da važi i za cijelu kodnu riječ.

Kod jednog bita distanca između riječi 0 i 0 i 1 i 1 je uvijek 0. Distanca između bilo koje četiri kombinacije *x* i *y* je ista kao distanca između *y* i *x*:

$$d(0,0)=d(0,0)\ \ d(0,1)=d(1,0),\ d(1,0)=d(0,1),\ d(1,1)=d(1,1)$$

Konačno najteže je dokazati nejednakost trougla $d(x,y)+d(y,z)\geq d(x,z)$. Ovdje možemo upotrijebiti sledeću tabelu:

x	*y*	*z*	$d(x,y)$	$d(x,z)$	$d(x,y)+d(y,z)$	$d(x,z)$
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	1	1	2	0
0	1	1	1	0	1	1
1	0	0	1	0	1	1
1	0	1	1	1	2	0
1	1	0	0	1	1	1
1	1	1	0	0	0	0

Kada navedene osobine važe po jednom bitu onda one važe i za kompletnu kodnu riječ.

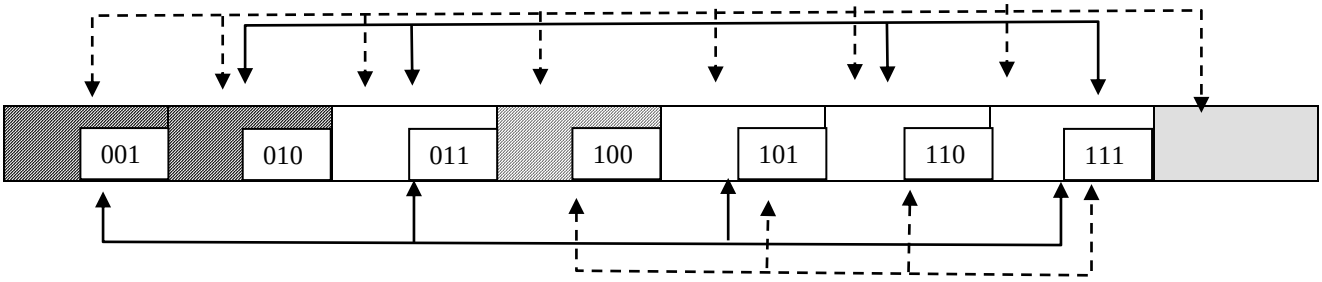
Površ sfere oko tačke predstavlja sve tačke koje su udaljene od nje za datu distancu. Ukupan broj tačaka koje su na udaljenju 1 od posmatrane tačke je *n*. Da bi kod bio jedinstven minimalna distanca između kodnih simbola mora biti jedan. Da bi se mogla detektovati greška distanca između kodnih simbola mora biti dva. Ako je distanca između kodnih simbola tri može se vršiti korekcija jedne greške ili detekcija dvije dok distanca od 4 daje simultanu jednu korekciju greške i dvije detekcije, distanca 5 daje dvije korekcije greške. Zapremina sfere prečnika 1 je 1+*n*. Očigledno zapremina sfere polupečnika *n* u *n*-dimenzionom prostoru je 2^n. Maksimalan broj bitova koji se koriste za slanje poruke mora zadovoljavati:

$$\frac{\text{ukupna zapremina}}{\text{zapremina sfere}} \geq \text{maksimalan broj sfera} \qquad \text{ili} \qquad \frac{2^n}{n+1} \geq 2^k$$

Kako je $n=m+k$ slijedi $2^{m+k} \geq 2^k(n+1)$ odnosno $2^m \geq n+1$. Npr. za dvostruku korekciju greške moramo imati distancu između kodnih simbola od 5 što znači da nepreklapajuće sfere moraju imati poluprečnik od 2. Ovo daje:

$$\frac{2^n}{n(n-1)/2+n+1} \geq 2^k$$

Pomenuli smo da postoji kod koji se može koristiti za korekciju jedne greške ili detekciju 2 i da je potreban razmak između kodnih simbola 4. Da bi se ovo postiglo u odnosu na kod koji koriguje jednu grešku dodaje se još jedan bit odnosno još jedna provjera parnosti. Dvostruka greška daje neke od kontrolnih sindroma jednake jedan ali ova provjera parnosti ostaje zadovojena. U ovom slučaju imamo 4 provjere parnosti. Ako prve tri sugerišu da je došlo do greške a četvrta to potvrđuje vršimo ispravku dok u slučaju da prve 3 sugerišu grešku a četvrta kaže da je sve u redu konstatujemo da je došlo do parnog broja grešaka.



Na slici je grafički ilustrovano što se dešava kod Hammingovog koda (8,4) odnosno koji biti parnosti kontrolišu koje informacione bite. Kao memotehnika može da posluži činjenica da biti parnosti na pravilnim binarnim pozicijama kontrolišu one bite koji imaju u binarnoj predstavi pozicije bita u kodnoj riječi jedinicu na istom mjestu kao i sami biti parnosti. Međutim, uskoro ćemo vidjeti da to ne mora da bude slučaj odnosno da se biti parnosti ne moraju raspoređivati na pravilne binarne pozicije.

Ovaj način se pozicioniranje bit može primjeniti kod svih formi Hammingovog koda. Kod (8,4) sa mogućnošću jedne ispravke i dvije detekcije greške je najpoznatiji samo-dualni kod. O samodualnosti, premda veoma bitnoj osobini kodova, nećemo govoriti zbog potrebe da objasnimo neke napredne koncepte. Sada možemo da izvedemo vjerovatnoće da pravougaoni kod, trougaoni kod i Hammingov kod rade odnosno ne rade. Kod svih kodova nemamo nikakav problem vezan za

situaciju kada nema pogreški odnosno svi ovi kodovi ako su greške na bitima jednakoraspoređene i nezavisne imaju ovu vjerovatnoću jednaku:

$$(1\text{-}p)^n$$

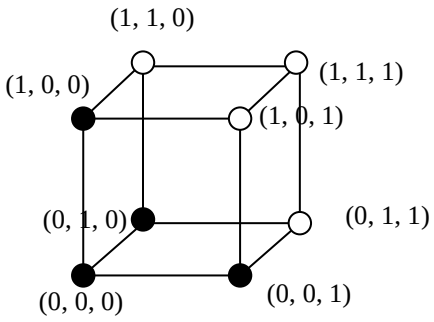
U slučaju jedne pogreške kod pravougaonog, trougaonog, Hammingovog i proširenog Hammingovog koda (jednom riječju kod blok kodova sa mogućnošću ispravke jedne pogreške) imamo vjerovatnoću koja je jednaka:

$$np(1\text{-}p)^{n\text{-}1}$$

Jedini kod koji je u stanju da sistematski detektuje dvije pogreške je prošireni Hammingov kod:

$$n(n\text{-}1)p^2(1\text{-}p)^{n\text{-}2}/2$$

Činjenica je međutim da višak bita parnosti kojim raspolažu pravougaoni i trougaoni kod isto mogu u pojedinim situacijama da pomognu da se detektuje više od jedne pogreške. Na primjer, ako kod pravougaonog koda se pojave dvije greške u jednoj vrsti u toj vrsti će biti detektovano da nema greške ali će se greške biti indicirane za dvije kolone. Slično ako se pojave dvije greške u dvije različite vrste i kolone imaćemo da su se indicirala četiri bita parnosti. Analiza rada pravougaonog koda u ovom slučaju je složena.



6.4. Kodovi za detekciju i ispravljanje pogreške u matričnom obliku – pojam sindroma

Blok (*n*,*k*) podrazumjeva kod koji se sastoji od kodiranja *n* bita od kojih je *k* za prenos informacije. Pod kodnom brzinom ili kodnim odnosom za binarni kanal podrazumjevamo $R=k/n$. Dekoder ponavlja pravila iz kodera, otkriva i ispravlja greške a zatim uklanja dodate simbole. Kod ovih kodova se koristi modulo-2 aritmetika. Sabiranje je zapravo eksluzivno ili operacija dok je množenje klasična logička i operacija. Operacija sabiranja je ista kao i operacija oduzimanja. Binarni sistem koji se sastoji od kodnih simbola 0 i 1 i za koji važe pravila sabiranja, oduzimanja, množenja itd naziva se konačnim poljem. Osnovni kod za detekciju greške je prosti paritet sa jednim bitom. Može se označiti kao:

$$i_1 \oplus i_2 \oplus ... \oplus i_k \oplus p_1 = 0$$

Ako paritet nije ispunjen onda je došlo do pogreške u prenosu. Takođe, do pogreške može doći ako je došlo do više od jedne greške što se kod parnih grešaka ne može detektovati. Ovakvo dekodiranje koje može otkriti samo neparan broj greška zovemo nepravilno dekodiranje. Nekoliko pravila:

$$P\{\text{ispravnog dekodiranja}\} = (1-P_g)^n \;\; P\{\text{otkrivanje pogreske}\} = \sum_{i=\text{neparno}}^{n} \binom{n}{i} P_g^i (1-P_g)^{n-i}$$

$$P\{\text{neispravnog dekodiranja}\} = \sum_{i=\text{parno}>0}^{n} \binom{n}{i} P_g^i (1-P_g)^{n-i}$$

Sledeći kod je tzv. blok kod sa kodiranjem po redovima i vrstama. Npr. kod (9,4) itd. Binarni kod sa ponavljanjem se sastoji od *n* identičnih bita koji se ponavljaju.

To znači da ako se prenosi *n* bita prenosi se u stvari samo jedan bit informacije. Tako je $R=1/n$. Odluka se na prijemu se donosi većinom glasova. Još je kompleksniji kod sa ponavljenjem i ugrađenim paritetom. Kod ovog koda se paritet može zapisati kao:

$$\begin{aligned}
&i_1 + p_1 = 0 \\
&i_1 + \quad + p_2 = 0 \\
&i_1 + \qquad\quad + p_3 = 0 \\
&\ldots \\
&i_1 + \qquad\qquad\quad + p_{n-1} = 0
\end{aligned}$$

Ako je primljena riječ $[r_1, r_2, ..., r_n]$ provjera pariteta se provodi kao:

$$\begin{aligned}
&r_1 + \; r_2 \qquad\qquad\quad = S_1 \\
&r_1 + \qquad r_3 \qquad\quad = S_2 \\
&r_1 + \qquad\quad r_4 \qquad = S_3 \\
&\ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \qquad \ldots \\
&r_1 + \qquad\quad \ldots \quad r_n \quad = S_{n-1}
\end{aligned}$$

$S = [S_1, S_2, ..., S_{n-1}]$ je nul vektor. Naravno ako je i broj grešaka mali mi ćemo detektovati u kome slučaju je greška nastupila a ako je veći od pola znači da nam je vjerovatno poruka neispravna.

Ovaj kod se može prikazati preko matrice oblika:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{cH}^T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Sindrom je jednak: $\mathbf{S} = \mathbf{rH}^T$. Matrica **H** se naziva kontrolnom matricom koda. Vjerovatnoća pravilnog dekodiranja je: $\sum_{i=0}^{(n-1)/2} \binom{n}{i} P_g^i (1-P_g)^{n-i}$. Vjerovatnoća pogrešnog dekodiranja je:

$$1 - P\{\text{pravilnog dekodiranja}\} = P_e$$

Na sličan način se može opisati i Hammingov kod. Kontrolni biti kod Hammingovog koda ne moraju biti restriktivno na pozicijama 1, 2, 4 itd. Posmatrajmo sada slučaj da su prva četiri bita informacioni a 3 naredna neinformacioni:

$$\begin{aligned}
&X_4 = X_1 + X_2 + X_3 \pmod 2 \\
&X_5 = X_0 + X_2 + X_3 \pmod 2 \quad \text{svaka jednačina izostavlja jedan od bita} \\
&X_6 = X_0 + X_1 + X_3 \pmod 2
\end{aligned}$$

Ovo se dalje može opisati matričnom jednačinom:

$$0 + X_1 + X_2 + X_3 + X_4 + 0 + 0 = 0$$
$$X_0 + 0 + X_2 + X_3 + 0 + X_5 + 0 = 0$$
$$X_0 + X_1 + 0 + X_3 + 0 + 0 + X_6 = 0$$

Matrica sistema je jednaka:

$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Ako je kodna riječ zapisana kao $\mathbf{X} = [X_0, X_1, X_2, ..., X_6]$ vrijedi jednačina: $\mathbf{XH}^T = [0\ 0\ 0]$. Neka se sada pridruži izvornom vektoru vektor pogreške **Z** koji ima jedan na jednoj poziciji i na svim ostalim ima vrijednost 0 to možemo zapisati kao **Y**=**X**+**Z**. Ovo se dalje može računati kao:

$$\mathbf{S} = \mathbf{YH}^T = (\mathbf{X} + \mathbf{Z})\mathbf{H}^T = \mathbf{XH}^T + \mathbf{ZH}^T = \mathbf{ZH}^T$$

Vektor **S** je sindrom vektora **Y**. Sindrom ne zavisi od predatog vektora već samo od pogreške. Da bi se odredio vektor **Z** potrebno je riješiti jednačinu $\mathbf{S} = \mathbf{ZH}^T$. Ovo se rješava tako što se provjerava koji se bit promjenio.

Sada dolazimo do veoma jednostavne mogućnosti da izvršimo realizaciju ovih kodova. Uzmimo na primjer da imamo pravougaoni kod (9,4). Ovaj kod možemo prikazati na sljedeći način:

i_1	i_2	c_1
i_3	i_4	c_2
c_3	c_4	c_5

Sa i_1-i_4 smo označiti informacione bite dok su c_1-c_5 kontrolni biti. Odgovarajući skup jednačina možemo zapisati kao:

$$i_1 \oplus i_2 \oplus c_1 = 0$$
$$i_3 \oplus i_4 \oplus c_2 = 0$$
$$i_1 \oplus i_3 \oplus c_3 = 0$$
$$i_2 \oplus i_4 \oplus c_4 = 0$$
$$i_1 \oplus i_2 \oplus i_1 \oplus i_2 \oplus c_4 = 0$$

Ako postavimo kodnu riječ vrstu po vrstu pravougaone matrice dobićemo kontrolnu matricu **H** kao:

$$\begin{matrix} i_1 & i_2 & c_1 & i_3 & i_4 & c_2 & c_3 & c_4 & c_5 \end{matrix}$$
$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Ova matrica ima dimenzije koje su (broj bita parnosti)x(broj bita u kodnoj riječi). Međutim, niko nam ne brani da postavimo bite parnosti (kontrolne bite) na bilo koju poziciju. Najprirodnije je da ta pozicija bude na poslednjim mjestima u kodnoj riječi. Na primjer ako je kodna riječ oblika:

$$[i_1\ i_2\ i_3\ i_4\ c_1\ c_2\ c_3\ c_4\ c_5]$$

U tom slučaju kontrolna matrica ima oblik:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Na primjer, ako su informacioni biti 0101 tada je kodna riječ 010111000. Pretpostavimo pogrešku na petom bitu (kontrolnom) dobijamo: 010101000. Množeći kodnu riječ sa kontrolnom matricom dobijamo:

$$[0\,1\,0\,1\,0\,1\,0\,0\,0]\begin{bmatrix} 1 & 1 & 0 & 0 & \boxed{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & \boxed{0} & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & \boxed{0} & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & \boxed{0} & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & \boxed{0} & 0 & 0 & 0 & 1 \end{bmatrix}^T = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Dakle, izuzetno elegantno dobijamo da je greška nastupila na petoj poziciji odnosno na poziciji kontrolnog bita što dalje znači da su informacioni biti korektno preneseni.

Posmatrajmo sada trougaoni kod (10,6) sa bitima parnosti raspoređenim na posljednjim pozicijama u kodu. Kod ima sljedeće informacione i kontrolne bite:

i_1	i_2	i_3	c_1
i_4	i_5	c_2	
i_6	c_3		
c_4			

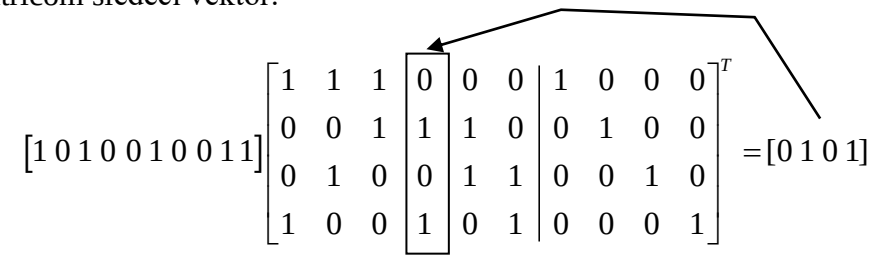
Kako svaki kontrolni bit kontroliše svoju kolonu odnosno vrstu možemo zapisati sledeće četiri relacije koje važe za slučaj kada nema greške:

$$i_1 \oplus i_2 \oplus i_3 \oplus c_1 = 0$$
$$i_3 \oplus i_4 \oplus i_5 \oplus c_2 = 0$$
$$i_2 \oplus i_5 \oplus i_6 \oplus c_3 = 0$$
$$i_1 \oplus i_4 \oplus i_6 \oplus c_4 = 0$$

Kontrolna matrica koda je:

$$\left[\begin{array}{cccccc|cccc} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

Ako je kodna riječ 101101 tada su kontrolni biti c_1=0, c_2=0, c_3=1 i c_4=1. Kodna riječ je: 1011010011. Ako se pogreška dogodila na četvrtom bitu dobijemo množenjem primljene riječi sa kontrolnom matricom sledeći vektor:



$$[1\,0\,1\,0\,0\,1\,0\,0\,1\,1]\begin{bmatrix}1&1&1&0&0&0&1&0&0&0\\0&0&1&1&1&0&0&1&0&0\\0&1&0&0&1&1&0&0&1&0\\1&0&0&1&0&1&0&0&0&1\end{bmatrix}^T=[0\,1\,0\,1]$$

Ponovo smo na ispravan način identifikovali poziciju na kojoj se greška dogodila.

Posmatrajmo finalni primjer Hammingovog koda (7,4) u kojem ćemo bite parnosti postaviti na posljednjim mjestima u kodnoj riječi. Dakle, kodna riječ se može zapisati kao: [i_1 i_2 i_3 i_4 c_1 c_2 c_3]. Bitovi parnosti vrše kontrolu na sledeći način:

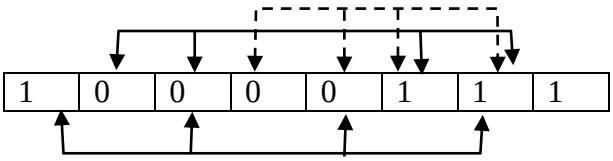
$i_1 \oplus i_2 \oplus i_3 \oplus c_1 = 0$
$i_1 \oplus i_2 \oplus i_4 \oplus c_2 = 0$
$i_1 \oplus i_3 \oplus i_4 \oplus c_3 = 0$

Kontrolna matrica se sada može zapisati u obliku:

$$H=\begin{bmatrix}1&1&1&0&1&0&0\\1&1&0&1&0&1&0\\1&0&1&1&0&0&1\end{bmatrix}$$

Sada je jasno da pomjeranje bita parnosti i informacionih bita u kodnoj riječi suštinski ne mijenja njenu fleksibilnost i funkcionalnost već samo postoji potreba da i prijemna i predajna strana (koder i dekoder) budu usklađeni sa navedenim izborom pozicija bita u kodnoj riječi.

Postavlja se pitanje ako je vjerovatnoća greške po bitu p kolika je vjerovatnoća da kodovi (7,4), (15,11) te za trougaoni kod (10,6) i pravougaoni (9,4) ne uspiju da dekodiraju poruku. Pretpostavimo da su greške na pojedinim bitovima nezavisne u odnosu na druge bitove (i.i.d. proces). U svim slučajevima vjerovatnoća uspješnog prenosa bez pojave grešaka vjerovatnoća je jednaka $(1-p)^n$ gdje je n broj bita u kodnoj riječi. Očigledno je da je ova vjerovatnoća manja što je kodna riječ duža a to dalje znači da od kodova sa istim brojem informacionih bita najbolji su oni koji su kraći a imaju istu spospobnosti za ispravljanje pogreški i po osnovu efikasnosti i po osnovu robusnosti na pojavu grešaka. Pojavu jedne greške svi navedeni kodovi su u stanju da isprave. Vjerovatnoća pojave jedne pogreške koju svi ovi kodovi mogu ispraviti je $np(1-p)^{n-1}$ gdje ponovo važi isti zaključak koji glasi da je bolje imati kraće kodove sa istom mogućnošću za ispravljanje pogreški. Postavlja se pitanje što je sa dvije pogreške i kakvo je stanje na tom polju. Očigledno da (7,4) kod nikada ne može detektovati dvije pogreške odnosno da će dvije pogreške posmatrati kao jednu i tu jednu (na pogrešnoj poziciji) i ispraviti na pogrešan način. Kod (9,4) koda međutim postoje situacije kada će dvije pogreške makar biti indicirane recimo kada se dogode na informacionim bitima koji su dijagonalno jedan u odnosnu na drugi (u tom slučaju 4 bita parnosti će indicirati greške ali nismo u stanju da ih ispravimo već samo da ih detektujemo). Greške u jednom redu ili vrsti međutim nismo u stanju da ispravimo. Ovaj detalj je interesantan za posebnu analizu međutim ovdje ćemo to preskočiti već ćemo ukazati na mogućnost kreiranja Hammingovih kodova koji imaju dodatni bit parnosti koji kontroliše kompletnu kodnu riječ. Prikažimo Hammingov kod (8,4) (ponekad se naziva prošireni Hammingov kod) kod kojega je dodat još jedan bit parnosti koji kontroliše kompletnu kodnu riječ:



U slučaju da se dogodi jedna pogreška tada će bit parnosti na posljednjem mjestu također sugerisati da je do pogreške došlo što nije sporno i doći će do ispravljanja te pogreške. U slučaju da se dese dvije pogreške prvih sedam bita će sugerisati da greška postoji dok će osmi bit sugerisati da je nema pa ćemo zaključiti da je došlo do višestrukih pogreški koje ovaj kod ne može ispraviti. Dakle, imamo slučaj detekcije dvije i ispravljanja jedne pogreške. Biti kodne riječi sada zadovoljavaju sledeći sistem jednačina

$$0+X_1+X_2+X_3+X_4+0+0+0=0$$
$$X_0+0+X_2+X_3+0+X_5+0+0=0$$
$$X_0+X_1+0+X_3+0+0+X_6+0=0$$
$$X_0+X_1+X_2+X_3+X_4+X_5+X_6+X_7=0$$

Ovome odgovara kontrolna matrica sa 4 vrste (odgovaraju četiri provjere parnosti) sa osam kolona (odgovaraju bitima u kodnoj riiječi):

$$H=\begin{bmatrix}0&1&1&1&1&0&0&0\\1&0&1&1&0&1&0&0\\1&1&0&1&0&0&1&0\\1&1&1&1&1&1&1&1\end{bmatrix}$$

Vidimo da Hammingov kod veoma dobro radi sa riječima kada imamo 4, 11, 26, ... bita. U opštem slučaju ako imamo 2^k-1-k informacionih bita situacija je veoma pogodna za Hammingov kod. Međutim, ako imamo broj informacionih bita koji se ne uklapa u ove brojeve onda imamo određeni problem. Međutim, navedeni problem nije znatan već samo treba skratiti informacionu riječ podrazumijevajući da su biti koji nedostaju nule. Pretpostavimo da imamo informacionu riječ dužine 6. To znači da će nam trebati makar 4 bita parnosti. U sledećoj tabeli su navedeni biti parnosti kao i informacioni biti. Za svaki bit parnosti navedene su pozicije bita koje kontroliše (u ovom slučaju su biti parnosti postaljeni na pravilne binarne pozicije. Biti parnosti su označeni sa P a informacioni sa I).

Pozicija	1	2	3	4	5	6	7	8	9	10
Bit	P	P	I	P	I	I	I	P	I	I
Kontrola	(1,3,5,7,9)	(2,3,6,7,10)		(4,5,6,7)				(8,9,10)		

Na sličan način se mogu formirati bilo koji tip skraćenih Hammingovih kodova.

Sada se može pokušati provjeriti kolika je vjerovatnoća greške prilikom primanja Hammingovog koda i nekog drugog koda za različite vjerovatnoće greške na jednom bitu.

Uzmimo opšti oblik paritetnog koda (n,k). Skup mogućih kodnih riječi se označava kao $[\mathbf{c}_i]\ i=0,1,2,...,2^k-1$ ili jednostavno sa **c**. Nakon prenosa dobijamo riječ **r** koja se može razlikovati u jednom ili više bita od poslate: **r=c+e**. Zadatak dekodiranja je pokušati odrediti **e** kako bi se

mogla izvršiti detekcija ispravne poruke **c'=r'-e'**. Svi kodovi sa kontrolom pariteta vrše računanje vektora sindroma dužine **(n-k)**. Polazeći od usvojenih relacija dobijamo:

$$\mathbf{cH}^T=\mathbf{0} \qquad \mathbf{S}=\mathbf{rH}^T$$

Sindrom je nezavisan od kodne riječi i zavisi samo od vektora pogreške:

$\mathbf{S}=\mathbf{rH}^T=(\mathbf{c}+\mathbf{e})\mathbf{H}^T=\mathbf{cH}^T+\mathbf{eH}^T=\mathbf{0}+\mathbf{eH}^T=\mathbf{eH}^T$

Za bilo koji vektor pogreške $\mathbf{2^k}$ vektora $\mathbf{e}_i$ se mogu odrediti kao:

$$\mathbf{e}_i=\mathbf{r}+\mathbf{c}_i, \quad i=0,...,2^k-1$$

Za isti sindrom važi da je $\mathbf{e}_i\mathbf{H}^T=(\mathbf{e}+\mathbf{c}_i)\mathbf{H}^T=\mathbf{eH}^T=\mathbf{e}_j\mathbf{H}^T$. Zaključujemo da sindrom ne može odrediti na jedinstven način poruku. Dakle, umjesto da tačno odredimo poruku mi ćemo biti zadovoljni i ako poruku odredimo približno, odnosno, ako odredimo najvjerovatniju pogrešku. Sindrom je suma redova matrica $\mathbf{H}^T$ čiji indeksi odgovaraju mjestima pogrešnog bita. Zapišimo matricu **H** u obliku:

$$\mathbf{H}^T=\begin{bmatrix}\mathbf{h}_1\\ \mathbf{h}_2\\ \cdots\\ \mathbf{h}_n\end{bmatrix} \qquad \mathbf{S}=\mathbf{eH}^T=\sum_{j=1}^{n}e_j\mathbf{h}_j$$

$\mathbf{e}_j$ je j-ti elemenat vektora pogreške i $\mathbf{e}_j=1$ znači da je do pogreške došlo.

6.5. Preciznija definicija Hammingovog koda

Hammingova distanca se definiše kao:

$$d_H(\mathbf{x},\mathbf{y})=\{\text{broj elemenata vektora koji se razlikuju } x_i\neq y_i=w_H(\mathbf{y}-\mathbf{x})\}$$

w_H se na naziva Hammingova težina koja predstavlja broj nenultih pozicija u vektoru. Minimalna distanca $d_{\min}(K)$ je minimalna težina razlike bilo koje dvije riječi koda $\mathbf{c}_i$-$\mathbf{c}_j$.

Neka je $K=\{\mathbf{c}_i\}$,i=0,1,2,...,2^k-1 (n,k) kod sa kojim se želi postići ispravljanje svih w-strukih pogreški koje su manje ili jednake od Hammingove težine. To znači da je poslata kodna riječ $\mathbf{c}_i$ i primljena kodna riječ $\mathbf{r}=\mathbf{c}+\mathbf{e}_i$ i da je $w_H(\mathbf{e})\leq w$ da dekoder na izlazu daje $\mathbf{c}'=\mathbf{c}_i$. Neka je raspodjela svih kodnih riječi uniformna i jednaka $1/2^k$. Najbolje rješenje je odabrati onu kodnu riječ koja je najbliža datoj riječi a to je kodna riječ za koju važi $d_H(\mathbf{c}_i,\mathbf{r})=w_H(\mathbf{e})$ najmanje. Dakle, na osnovu jednostavne geometrijske postavke da bi dekodiranje bilo moguće mora da važi:

$d_H(\mathbf{c}_i,\mathbf{c}_j)\geq 2w+1$ za svako i i j koja su međusobno različita. Ovo se sada može prikazati geometrijski. Minimalna distanca koda je:

$$d_{\min}(K)=\min\{d_H(\mathbf{c},\mathbf{c}'):\mathbf{c},\mathbf{c}'\in K,\mathbf{c}\neq\mathbf{c}'\}$$

Kod može da ispravi sve oblike pogreški do težine w ako važi: $w\leq(d_{\min}(K)-1)/2$. Ova razmatranja možemo dovesti u vezu sa strukturom kontrolne matrice koda **H**. $d_{\min}(K)$ je minimalan broj redova matrice **H** za koji je suma jednaka nuli:

$$\mathbf{H}=\begin{bmatrix}0&1&1&1&1&0&0\\ 1&0&1&1&0&1&0\\ 1&1&0&1&0&0&1\end{bmatrix}$$ $d_{\min}(K)$=3 odnosno kod može ispravljati jednostruke pogreške.

Definicija Hammingovog koda. Neka je **H** mx(2^m-1) binarna matrica takva da njeni redovi imaju 2^m-1 nenultih vektora po nekom rasporedu. Binarnim Hammingovim kodom (n,m) dužine 2^m-1 nazivamo linearni kod (m,k) n=2^m-1 k=2^m-1-m, i njegova je kontrolna matrica **H**.

Dekodiranje sindroma je za ovakav kod jednostavno. Ako je jednak nuli riječ je primljena ispravno a ako je $w_H(\mathbf{e})$=1 i ako je $\mathbf{e}_i$=1 tada je sindrom jednak i-toj koloni od **H**.

Hammingov kod ne gubi svojstva ako kolone promjene redosljed pa se često kod (7,4) piše u tzv. pravilnom binarnom redosljedu:

$$\mathbf{H}=\begin{bmatrix}0&0&0&1&1&1&1\\0&1&1&0&0&1&1\\1&0&1&0&1&0&1\end{bmatrix}$$

Kod ovakvog koda dekadni ekvivalent sindroma predstavlja poziciju u kodu gdje se dogodila greška.

Da bi se kreirao Hammingov kod koji će sa n nenultnih kolona koji će moći odrediti mjesto jednostruke pogreške mora da važi uslov:

$$2^{n-k}\geq n+1 \qquad\qquad m\leq n-\log[n+1]$$

Hammingov kod ima dosta dobru kodnu odnos za veliko n. Naravno za veliko n raste i vjerovatnoća da se dogode višestruke pogreške.

Zadaci za vježbu

6.1. Dokazati osobine Hammingove distance: distanca je nula ako su kodne riječi iste; distanca od x do y je jednaka distanci od y do x i kod Hammingove distance važi pravilo trougla. **Uputstvo.** Možete ove osobine dokazati za svaki bit u kodnoj riječi pojedinačno i dokazati da onda to važi za cijelu kodnu riječ.

Rješenje: Sve osobine se mogu dokazati primjenom na pojedinačnom bitu jer ako važi za svaki bit u kodnoj riječi važi i za kompletnu kodnu riječ. Prva osobina kaže da je:

$$d_H(x,x)=0 \text{ odnosno na jednom bitu } d_H(0,0)=0 \text{ odnosno } d_H(1,1)=0$$

Druga osobina znači da je distanca komutativna odnosno da važi da je:

$$d_H(a,b)=d_H(b,a)$$

A	b	$d_H(a,b)$	$d_H(b,a)$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

Osobina nejednakosti trougla se može zapisati kao:

$$d_H(x,y)+d_H(y,z)\geq d_H(x,z)$$

koju ćemo također dokazati putem pandana tabele istinitosti:

X	y	z	$d_H(x,y)$	$d_H(y,z)$	$d_H(x,y)+d_H(y,z)$	$d_H(x,z)$
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	1	1	2	0
0	1	1	1	0	1	1
1	0	0	1	0	1	0
1	0	1	1	1	2	0
1	1	0	0	1	1	1
1	1	1	0	0	0	0

6.2. Hammingova težina je broj nenultih mjesta u nekoj kodnoj riječi. Definisati Hammingovu distancu na osnovu Hammingove težine. Pokazati da je minimalna Hammingova težina nenulte kodne riječi (kodne riječi koja nije predstavljana sa svim nulama) jednaka minimalnoj Hammingovoj distanci između dvije različite kodne riječi. **Napomena.** Mada na prvi pogled izgleda sasvim uobičajen stav kakvih smo imali dosta u dosadašnjem toku nastave ovaj stav je veoma značajan za određivanje minimalne distance kao veoma važne karakteristike za determinisanje kvaliteta koda. Naime, ako bi određivali minimalnu distancu provjeravajući svaku kombinaciju dvije kodne riječi složenost bi bila N^2 gdje je N broj kodnih riječi u kodu koji može biti ogroman. Na ovaj način mi to radimo samo N puta.

Rješenje: Hammingova distanca se može definisati kao

$$d_H(\mathbf{x},\mathbf{y})=w_H(\mathbf{x}+\mathbf{y})$$

Kod Hammingove težine korišćena je oznaka + da naznači ekskluzivno ili po bitovima. Za neki kod minimalnu distancu računamo kao

$$\mathbf{d}_{\min}=\min\{d_H(\mathbf{c}_i,\mathbf{c}_j)\}$$

gdje je $i\neq j$ odnosno izraženo preko Hammingove težine ovo se može zapisati kao:

$$\mathbf{d}_{\min}=\min\{w_H(\mathbf{c}_i+\mathbf{c}_j)\}$$

Kako je kodna riječ u našem sistemu sigurno riječ nula (upotrijebimo informacione bite sa svim nulama dobićemo kodnu riječ sa svim nulama u svim našim kodovima). Najmanja distanca za tu kodnu riječ je jednaka:

$$\mathbf{d}_{\min}=\min\{w_H(\mathbf{c}_i+\mathbf{0})\}=\min\{w_H(\mathbf{c}_i)\}$$

čime smo dokazali predmetnu tvrdnju ako možemo da dokažemo da ne postoji nijedna druga kodna riječ od koje je rastojanje manje od neke druge kodne riječi. Suština je u tome da je postupak dobijanja kodne riječi na navedeni način linearan odnosno da suma dvije informacione kodne riječi daje sumu kodnih riječi. Sumiranjem sa istom kodnom riječju možemo sigurno dobiti nultu kodnu riječ (koja je isto ispravna kodna riječ) a na osnovu gore izvedene nejednakosti trougla jednostavno slijedi dokaz predmetne tvrdnje.

6.3. U praksi postoji značajan broj ARQ (Automatic Repaet Request) kodova kod kojih dekoder može da prepozna da se greška u poruci dogodila i da generiše automatski zahtjev za ponavljanje poruke. Jedan od tih kodova naziva se kod 3 od 7 i kod njega u poruci ima uvijek 3 jedinice (ili nule u zavisnosti od konvencije) i ostatak nula. Dekoder broji nule i jedinice i ako naiđe na neregularnost traži ponovno slanje poruke. Koliki je kodni odnos ovog koda? Kolika je vjerovatnoća da se

dogodila pogreška i da je dekoder zahtjevao ponovno slanje poruke? Kolika je vjerovatnoća da dekoder za pogrešnu poruku utvrdi da je ispravna? Pretpostaviti da je vjerovatnoća pogreške p i da su pogreške na pojedinim bitovima nezavisne. Ponoviti prethodnu proceduru ako je p=0.1 i ako je vjerovatnoća da se greška dogodila na narednom bitu ako se dogodila na prethodnom jednaka 0.3. Pretpostaviti da je poruka od 7 bita nezavisna od svih ostalih poruka u sistemu.

Rješenje: Kodni odnos (brzina) ovog koda je:

$$R = \frac{\log_2 \binom{7}{4}}{7} = \frac{\log_2 35}{7} = 0.7328 \text{bita/simbolu}$$

Svaki slučaj kada se dogodi neparan broj pogreški dekoder će detektovati. U slučaju kada imamo paran broj pogreški to nije slučaj. Na primjer, za jednu grešku na bitu nula i jednu na jedinici ne možemo da detektujemo pogrešku dok ako su obije pogreške na nulama ili jedinicama imamo da kod detektuje pogrešku. U skladu sa prethodno objašnjenim detaljima slijedi da je:

$$(1-p)^7$$

vjerovatnoća da kod radi ispravno odnosno da nema pogreški. Vjerovatnoća detekcije pogreški je:

$$7p(1-p)^6 + 9p^2(1-p)^5 + 35p^3(1-p)^4 + 17p^4(1-p^3) + 21p^5(1-p)^2 + 3p^6(1-p) + p^7$$

dok je vjerovatnoća da greški ima a da ih ne možemo detektovati:

$$6p^2(1-p)^3 + 8p^4(1-p)$$
$$12p^2(1-p)^5 + 18p^4(1-p^3) + 4p^6(1-p)$$

6.4. Date su kodne riječi 6 kodova. Odrediti minimalne distance i na osnovu toga procjeniti kakve su sposobnosti koda u ispravljanju i detektovanju pogreški.

Kod 1	Kod 2	Kod 3	Kod 4	Kod 5	Kod 6
00, 01, 10, 11	000, 011, 101 i 110	000, 111	0000, 1111	00000, 11111	00000, 01101, 10110, 11001

Rješenje: (a) Ovdje je minimalno rastojanje dvije kodne riječi 1. Znači kod je jednoznačno dekodabilan. (b) Minimalno Hammingovo rastojanje dvije riječi kod ovog koda je 2. Kod može detektovati jednu pogrešku. (c) Ovdje je rastojanje 3. Kod je u stanju da ispravi jednu pogrešku. (d) Rastojanje je 4 kod može da ispravi jednu pogrešku i da detektuje dvije. (e) Rastojanje je 5. Kod je u stanju da ispravi dvije pogreške. (f) Minimalno Hammingovo rastojanje je 3 a to znači da je kod u stanju da ispravi jednu pogrešku.

6.5. Kombinaciju od k_1xk_2 bita treba kodirati pravougaonim kodom. Ako je vjerovatnoća greške na bilo kom bitu jednaka p i ako su pogreške na bitima međusobno nezavisne odrediti vjerovatnoće da: (a) Nema greške u sistemu; (b) Kod detektuje i ispravi pogrešku; (c) Kod detektuje ali ne može da ispravi pogrešku; (d) Kod pogrešno ispravi nastalu pogrešku. Ukoliko imate problem sa određivanjem ovih veličina u opštem slučaju posmatrajte pravougaoni kod (9,4) pa pokušajte da izvedete generalne zaključke.

Rješenje: (a) Vjerovatnoća da nema greške u sistemu je jednaka $(1-p)^{(k_1+1)(k_2+1)}$.
(b) Vjerovatnoća da se dogodi pogreška i da je možemo ispraviti je jednaka:

$$(k_1+1)(k_2+1)\,p(1-p)^{(k_1+1)(k_2+1)-1}$$

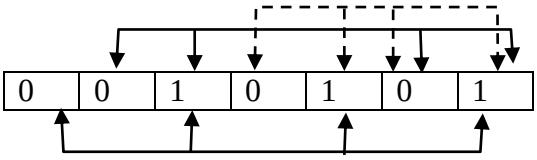
(c) Kod će biti u stanju da detektuje pojavu pogreške ako se to dogodi na dva bita. Naime, ako se pogreška dogodi u jednoj vrsti i/ili koloni onda će ta vrsta biti korektna sa stanovišta bita parnosti ali će se te greške pojaviti u kolonama/vrstama što će ukazati na dvije pogreške. U slučaju da se greške dogode u različitim kolonama vrstama onda će se na više od dva bita parnosti indicirati da je došlo do pogreški što ponovo ne možemo ispraviti. Dakle, pravougaoni kodovi sa dvije pogreške koje se pojavljuju sa vjerovatnoćom:

$$\frac{(k_1+1)(k_2+1)[(k_1+1)(k_2+1)-1]}{2}\,p^2(1-p)^{(k_1+1)(k_2+1)-2}$$

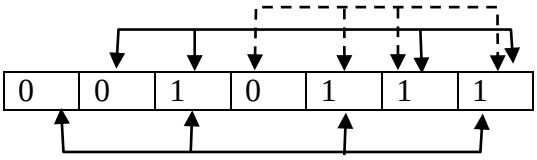
Situacija sa više od tri pogreške je znatno komplikovanija i neće biti dalje analizirana jer u tim slučajevima postoje obije varijante i da se može detektovati pogreška ali ne i ispraviti i varijanta da se konstatuje da se dogodila jedna pogreška te da se ta pogreška ispravi iako se dogodilo više od jedne greške.

6.6. Poruku 1101 kodirati Hammingovim kodom (7,4). Generisati pogrešku na poziciji 6 koju treba detektovati i ispraviti. Hammingov kod neka bude sa pravilnim binarnim rasporedom bitova.

Rješenje: Kodiranje ove poruke možemo obaviti tako što rasporedimo bitove onako kako smo učili (pozicija jedan, dva i četiri su kontrolni a ostali informacioni):



Nakon greške na šestom bitu dobijamo poruku:



Greške su uočene na drugoj poziciji i (0111) i četvrtoj (0111) pa zaključujemo da se pogreška dogodila na poziciji 6. Nakon njenog ispravljanja i ekstrakcije informacionih bita dobijamo 1110.

6.7. (a) Poruku 110011 treba kodirati sa pravougaonim kodom (12,6). Bitovi parnosti se postavljaju na posljednjim pozicijama u kodu. (b) Odrediti kontrolnu matricu ovog koda. (c) Do pogreške u rezultujućoj poruci je došlo na poziciji 7. Odrediti sindrom i uporediti sa odgovarajućom kolonom kontrolne matrice.

Rješenje: (a) Formirajmo tabelu:

1	1	0	0
0	1	1	0
1	0	1	0

Kodna riječ je sada 110011001010.

(b) Kontrolna matrica ovog koda je:

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

(c) Poruka sa greškom na sedmom bitu je 110011101010.

$$cH^T=S=[1\ 0\ 0\ 0\ 0\ 0]$$

Lako uočavamo da je ovaj sindrom jednak 7-moj koloni odnosno greška se desila na sedmoj poziciji.

6.8. (a) Dat je trougaoni kod (15,10). Izvršiti kodiranje poruke 1101101010. Bitove parnosti postaviti na posljednjim pozicijama u kodu. (b) Odrediti kontrolnu matricu za ovaj kod. (c) Do greške u prenosu je došlo na poziciji 6. Izvršiti dekodiranje poruke na osnovu kontrolne matrice.
Rješenje: (a) Kod ćemo formirati tako što formiramo trougaonu tabelu:

1	1	0	1	1
1	0	1	1	
0	1	0		
0	0			
0				

Kodna riječ je: [110110101011000].

(b) Kontrolna matrica ovog koda je:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

(c) Dobijena poruka sa greškom na šestom bitu je sada: [110111101011000]. Sindrom je sada:

$$\mathbf{cH}^T=\mathbf{S}=[0\ 1\ 0\ 0\ 1]$$

Sada možemo da dekodiramo poruku kao [110110101011000] a nakon ekstrakcije informacionih bita dobijamo: [1101101010].

6.9. (a) Dat je pravougaoni kod (9,4). Izvršiti kodiranje poruke 1101. Bitove parnosti postaviti na posljednjim pozicijama u kodu. (b) Odrediti kontrolnu matricu za ovaj kod. (c) Do greške u prenosu je došlo na poziciji 6. Izvršiti dekodiranje poruke na osnovu kontrolne matrice.

Rješenje: (a) Po potrebi možete raditi ovaj zadatak korak po korak ali ćemo ovdje ubrzati rješavanje a na vama je da izvršite provjeru. Kodna riječ je sljedećeg oblika:

1 1 0 1 0 1 1 0 1

(b) Kontrolna matrica ovog koda je:

$$H = \left[\begin{array}{cccc|ccccc} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

(c) Nakon prenosa kanalom poruka koja je primljena sa pogreškom je:

1 1 0 1 0 0 1 0 1

Dobijeni sindrom je jednak [0 1 0 0 0] što ukazuje da se pogreška dogodila na šestoj poziciji odnosno da je ispravna kodna riječ:

1 1 0 1 0 0 1 0 1

Nakon ekstrakcije informacionih bita dobijamo riječ: 1101.

Poglavlje VII
KODIRANJE I DEKODIRANJE PUTEM POLINOMA

7.1. Binarni linearni kodovi i pojam generišuće matrice

Definicija. Binarni linearni kod (n,k) je k-dimenzioni vektorski potprostor svih n-torki za koje važi $k<n$.

Svaki vektor dužine n bita oblikovan linearnom kombinacijom od k linearno nezavisnih osnovnih vektora $\mathbf{g}_1$, $\mathbf{g}_2$, ..., $\mathbf{g}_k$ je binarni (n,k) blok kod K. Tako će vektor $\mathbf{c}$ biti (n,k) kodna riječ koda K samo ako leži u k-dimenzionom vektorskom prostoru povezanom sa $\{\mathbf{g}_j\}$. Ako su $\{\mathbf{g}_j\}$ redovi kxn matrice $\mathbf{G}$ kodna riječ se može izraziti kao:

$$\mathbf{c} = [i_1, i_2, ..., i_k] \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ ... \\ \mathbf{g}_k \end{bmatrix} = \mathbf{iG}$$

gdje je $\mathbf{i}$ informacioni vektor od k bita, a $\mathbf{G}$ je generišuća matrica koda ili jednostavno generator koda. Skup kodnih riječi se dobija kada se uzmu svih 2^k kombinacija u kodnoj riječi $\mathbf{i}$. Svaki linearni kod posjeduje dvije osobine: suma dvije kodne riječi je opet kodna riječ i svaki kod sadrži nul vektor, to jest, kodnu riječ sa nulama na svim pozicijama.

$$\mathbf{c}_i + \mathbf{c}_j = \mathbf{i}_i\mathbf{G} + \mathbf{i}_j\mathbf{G} = (\mathbf{i}_i + \mathbf{i}_j)\mathbf{G} = \mathbf{iG}$$

Za kod sa trostrukim ponavljanjem važi:

$$\mathbf{c} = [\mathbf{i}_1 \;\; \mathbf{i}_1 \;\; \mathbf{i}_1] = \mathbf{i}_1\mathbf{G}$$

pa je generišuća matrica $\mathbf{G} = [1\ 1\ 1]$.

Kod sa jednostrukim paritetom je:

$$\mathbf{c} = [i_1, i_2, ..., i_k, (i_1 + i_2 + ... + i_k)]$$

U matričnom obliku ovo se može opisati kao

$$\mathbf{c} = [i_1, i_2, ..., i_k] \begin{bmatrix} 100...01 \\ 010...01 \\ ... \\ 000...11 \end{bmatrix} = \mathbf{iG}$$

Kod pravougaonog koda (9,4) može se pisati:

$$\mathbf{c} = [i_1, i_2, i_3, i_4] \begin{bmatrix} 100010101 \\ 010010011 \\ 001001101 \\ 000101011 \end{bmatrix} = \mathbf{iG}$$

U prethodnim primjerima možemo primjeti da su kodovi tako izgrađeni da su k bita svake kodne riječi neizmjenjivi informacijski biti, pa se generišuća matrica može zapisati kao:

$$\mathbf{G} = [\mathbf{I}_k \mid \mathbf{P}]$$

$\mathbf{I}_k$ je matrica dimenzija kxk sa jedinicama na glavnoj dijagonali, a $\mathbf{P}$ je kx(n-k) matrica čiji sastav determiniše provjere parnosti.

Neka je generišuća matrica:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \dots \\ \mathbf{g}_k \end{bmatrix} = \begin{bmatrix} 100\dots0, \mathbf{p}_1 \\ 010\dots0, \mathbf{p}_2 \\ \dots \\ 000\dots1, \mathbf{p}_k \end{bmatrix} = [\mathbf{I}_k \mid \mathbf{P}]$$

Svaki red ove matrice je kodna riječ. Odnos između informacionih i paritetnih bitova se može prikazati preko kontrolne matrice. Neka je m=n-k, kontrolna matrica mxn $\mathbf{H}$ će imati oblik:

$$\mathbf{H} = [\mathbf{P}^T \mid \mathbf{I}_m]$$

$$\mathbf{G}\mathbf{H}^T = [\mathbf{I}_k \mid \mathbf{P}] \begin{bmatrix} \mathbf{P} \\ \mathbf{I}_k \end{bmatrix} = \mathbf{P} + \mathbf{P} = \mathbf{0}$$

$\mathbf{0}$ predstavlja matricu sa kxm nula na svim mjestima.

$$\mathbf{c}\mathbf{H}^T = \mathbf{i}\mathbf{G}\mathbf{H}^T = \mathbf{0}$$

Za produktni kod matrica $\mathbf{H}$ ima oblik:

$$\mathbf{H} = \begin{bmatrix} 110010000 \\ 001101000 \\ 101000100 \\ 010100010 \\ 111100001 \end{bmatrix}$$

Za (7,4) Hammingov kod matrica $\mathbf{H}$ ima oblik:

$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{H} = [\mathbf{P}^T \mid \mathbf{I}_3]$$

$$\mathbf{G} = [\mathbf{I}_4 \mid \mathbf{P}]$$

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Prikažimo sada primjere ovih matrica za nekoliko popularnih kodova: Hammingov (7,4), trougaoni (10,6) i pravougaoni (9,4). Takođe vrijeme je i mjesto da prikažemo MATLAB realizaciju sa kontrolnim i generatorskim matricama.

$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} = [\mathbf{P}^T \mid \mathbf{I}_3]$$

gdje je

$$P = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Generatorska matrica koda je:

$$\mathbf{G} = [I_4 \mid \mathbf{P}] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Kontrolna matrica pravougaonog koda (9,4) sa bitima parnosti na krajnjim pozicijama u kodu ima oblik:

$$H = \left[\begin{array}{cccc|ccccc} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

Generatorska matrica za ovaj kod je:

$$G = \left[\begin{array}{cccc|ccccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{array}\right]$$

Konačno ista priča se može ponoviti za trougaoni kod što će ovdje biti i urađeno a na vama je da sve slučajeve provjerite:

$$H = \left[\begin{array}{cccccc|cccc} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$G = \left[\begin{array}{cccccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{array}\right]$$

Rad u Matlab-u sa kontrolnom matricom ilustrovaćemo samo za slučaj Hammingovog koda (7,4). Rad je više nego jednostavani ne zahtjeva velike izmjene ni za slučaj ostalih kodova (jedino treba podesiti *n*, *k* i *P*).

```
clear
n=7; k=4;
P=[1 0 1;1 0 1;1 1 0;1 1 1];
I=(rand(1,k)>0.5);  %slucajno odabrani informacioni vektor
H=[P',eye(n-k)];
G=[eye(k),P];
C=rem(I*G,2);  %kodna rijec
[m,p]=max(ceil(n*rand)); %slucajno generisana pozicija pogreske
E=zeros(1,n); E(p)=1; %vektor pogreske
R=xor(C,E); %poruka sa jednom pogreskom
S=rem(R*H',2); %sindrom
pz=1; ind=0; %potraga za pozicijom pogreske
while ((pz<=n) & (ind==0))

    if(all(S'-H(:,pz)==0)) %nadjena kolona koja odgovara sindromu
        ind=1;
    end
    pz=pz+1;
end
pz=pz-1;
R1=R;
R1(pz)=xor(R(pz),1);
ID=R1(1:k); %dekodirana poruka
disp('informacioni biti')
disp(I)
disp('kodirana poruka')
disp(C)
disp('poruka sa greskom')
disp(R)
disp('sindrom')
disp(S)
disp('ispravljena pogreska')
disp(R1)
disp('dekodirana poruka')
disp(ID)
```

Nadam se da ne postoji potreba za detaljnijom analizom programskog koda. Jedna naša realizacija ovog koda vodila je ka sljedećim rezultatima sa time da svaka realizacija zbog toga što smo slučajno generisali poziciju pogreške i informacione bite će biti različita.

```
informacioni biti
     1     0     1     0
kodirana poruka
     1     0     1     0     0     1     1
poruka sa greskom
     0     0     1     0     0     1     1
sindrom
     1     0     1
ispravljena pogreska
     1     0     1     0     0     1     1
dekodirana poruka
     1     0     1     0
```

Osnovni problem kod ovog programa što matrica *P* koja je najvažniji element u ovom kodu može da bude veoma neprijatno u slučaju velikih matrica. Međutim, kod Hammingov koda činjenica je da se matrica *H* sastoji od svih mogućih kombinacija bitova 0 i 1 isključujući kombinaciju sa svim

nulama. Napisati program koji je u stanju da rukuje sa ovim problemom sa što je moguće manje direktnog unošenja matrice *P*.

7.2. Pojam interlivera

Svi kodovi koji su do sada objašnjeni, kao i gotovo svi koji će biti uvedeni do kraja ovog kursa polaze od pretpostavke da je vjerovatnoća greške po kodnom simbolu (kod nas je to najčešće bit) jednako distribuirana i nezavisna veličina od kodnog simbola do kodnog simbola. U većini praktično primjenjivih kanala to nije slučaj. Na primjer, u digitalnom telefonskim kanalu procenat grešaka tokom perioda grmljavine je znatno veći nego u drugim intervalima. Takođe, ako se pojavi greška uzrokovana grmljavinom, mnogo je veća vjerovatnoća da će se greška ponovo dogoditi u nekom od narednih trenutaka (za neki od narednih kodnih simbola). Ako se u jednoj kodnoj riječi dogodi više od jedne pogreške u kodnoj riječi do sada uvedeni kodovi ne rade. Ovdje je trenutak da se ispriča i jedna šala iz vremena kada je kodna teorija bila u povoju. Naime, prvi i gotovo jedini kanal za koji je uočeno da generiše međusobno nezavisne i jednako distribuirane greške bio je astronomski kanal koji je analiziran tokom nekih od prvih komunikacija sa svemirskim brodovima. Naučnici su konstatovali da je "konačno pronađen kanal za naše kodove." Ispravljanje većeg broja grešaka je relativno komplikovana procedura i kvari neke od karakteristika koda, kao što je kodni količnik. Ovo je posebno nepotrebno u situaciji kada nam se ta višestruka greška dešava relativno rijetko. Kao tehnika za prevazilaženje ovog problema predložen je postupak interlivinga ili sklop koji se naziva interliver. Interliver radi na principu preuređivanja – permutacije poruke koju treba poslati. Ovdje to demonstrirajmo na sljedećem jednostavnom principu. Umjesto da se šalje kodna riječ po kodna riječ, interliver prikupi nekoliko kodnih riječi i šalje prve simbole svih kodnih riječi, zatim druge simbole svih kodnih riječi, treće simbole itd. Na primjer, neka nam je serija od 20 bita izdijeljena u 5 blokova po 4 bita. Svaki od blokova neka je kodiran Hammingovim kodom (7,4) sa bitima za provjeru parnosti raspoređenih u pravilni binarni raspored.

Originalna poruka:	1101	1011	1010	0010	1000
Kodirana poruka:	**1**0**1**0101	**0**1**1**0011	**1**0**1**1010	**0**1**0**1010	**1**1**1**0000

Boldirani su biti parnosti. Pretpostavimo da su se greške dogodile na pozicijama od 5-9 zaredom. U prvih sedam bita biće tri pogreške, dok će u drugih sedam bita biti dvije pogreške. U oba slučaja neće biti moguće pravilno dekodiranje prve grupe od četiri bita i druge grupe od četiri bita, jer predmetni Hammingov kod može da dekodira samo po jednu grešku u jednoj kodnoj riječi. Ako poruku propustimo kroz interliver koji šalje prve bite svih poruka, pa zatim druge bite itd. dobijeni kodni niz je:

1010**0** **1010**1 11101 00110 10000 01110 11000
(boldirani su sada biti na kojima su se dogodile pogreške)

Prijemni sklop sadrži deinterliver koji vrši suprotnu operaciju od interlivera. Poruka nakon deinterlivera je:

1**1**10101 0**0**10011 1**1**11010 0**0**01010 01**1**0000

U svakoj od poruka imamo po jednu pogrešku koju će Hammingov kod uspješno otkloniti. Postupak interlivinga podsjeća na pravougaonu matricu u čijim vrstama su upisane kodne riječi i čiji se sadržaj u kanal šalje ne vrstu po vrstu kako bi bilo očekivano već kolonu po kolonu. Očigledno je iz prethodnog primjera da broj kodnih riječi koje se koriste za smještaj u "matricu" odnosno broj vrsta matrice mora biti jednak najvećem očekivanom broju uzastopnih grešaka. Napominjemo da kod interlivera dužina vrste matrice ne mora biti jednaka dužini kodne riječi.

Kombinovani dizajn interlivera i blok koda nije jednostavna tema i postoje posebne klase kodova koje objedinjuju oba ova sklopa u jedan. To prevazilazi razmatranja u našem kursu.

Ovdje ćemo samo kratko demonstrirati jedan oblik interlivera koji se implementira kod optičkih memorijskih sredstava kao što su CD i DVD. Kod ovih sredstava zrnca prašine i ogrebotine preko nekih kritičnih dimenzija izazivaju greške u čitanju bitova podataka. Pretpostavimo da su podaci dati u tabeli (bloku) sa 24 elementa i da su raspoređeni na dolje naveden način:

i_1	i_5	i_9	i_{13}	i_{17}	i_{21}
i_2	i_6	i_{10}	i_{14}	i_{18}	i_{22}
i_3	i_7	i_{11}	i_{15}	i_{19}	i_{23}
i_4	i_8	i_{12}	i_{16}	i_{20}	i_{24}

Kod ovog tipa interlivera dolazi do pomjeranja svake vrste za po jedno mjesto više nego što je pomjerena prethodna vrsta. Dobijamo sledeći raspored bita:

i_1	i_5	i_9	i_{13}	i_{17}	i_{21}			
	i_2	i_6	i_{10}	i_{14}	i_{18}	i_{22}		
		i_3	i_7	i_{11}	i_{15}	i_{19}	i_{23}	
			i_4	i_8	i_{12}	i_{16}	i_{20}	i_{24}

Sada se poruke šalju kolonu po kolonu i umjesto originalne poruke dobijamo poruku:

[i_1 i_5 i_2 i_9 i_6 i_3 i_{13} i_{10} i_7 i_4 i_{17} i_{14} i_{11} i_8 i_{21} i_{18} i_{15} i_{12} i_{22} i_{19} i_{16} i_{23} i_{20} i_{24}].

7.3. Kreiranje kodova na osnovu prostih polinoma (elementi algebarske kodne teorije)

Vidimo da matrični zapis daje određenu slobodu u kreiranju koda, ali matrični zapis ima brojne nedostatke. Naime, danas korišćeni kodovi imaju često dužinu veću od n=100, pa i od n=1000 i u tim situacijama memorijski zahtjevi za kontrolnu i generišuću matricu su znatni, a ni matrično množenje nije jednostavna operacija. Još je veći problem u dizajnu kodova koji mogu da isprave veći broj pogreški što ćemo uskoro vidjeti. Stoga je dizajniran drugi (često mnogo pogodniji) način da se razvijaju kodovi. To je na osnovu polinoma. Tako se sekvenca 1,0,1,1,0 može zapisati kao polinom: $P_1(x)=x^4+x^2+x$ odnosno sekvenca 0,1,1,0,1 se prikazuje kao polinom $P_2(x)=x^3+x^2+1$. Ovi polinomi se sabiraju po modulu 2 (sabiraju se koeficijenti uz odgovarajuće stepene po modulu 2 odnosno po ekskluzivni ili operaciji) tako da važi $P_1(x)+P_2(x)=x^4+x^3+x+1$. Operacija sabiranja polinoma po modulu 2 je asocijativna $P_1+(P_2+P_3)=(P_1+P_2)+P_3$. Može se pokazati sa skup svih polinoma n-tog reda formira **grupu**. Situacija sa množenjem polinoma je znatno komplikovanija, jer se često vrši po modulu najvećeg stepena u dva polinoma. Na primjer, ako je najveći dozvoljeni stepen u sistemu q, odnosno član x^q i ako prilikom sumiranja se pojavi član x^r gdje je $r>q$ uzima se $x^{r \bmod q}$ odnosno u programerskoj notaciji iz programskog jezika C $x^{r\%q}$.

Prosti polinom se ne može prikazati kao produkt dva polinoma. Tako je P=1 prost polinoma. Takođe x i x+1 su prosti polinomi. $x^2=x\ x$, $x^2+1=(x+1)\ (x+1)$, $x^2+x=x\ (x+1)$ dok je x^2+x+1 prost polinom. Kako $x^2+1=(x+1)\ (x+1)$? Prosto važi $(x+1)\ (x+1)=x^2+2x+1$ ali srednji član nestaje zbog primjene sabiranja po modulu 2. Kako možemo dokazati da je x^2+x+1 prost. To se postiže dijeljenjem datog polinoma sa svim prostim polinomima (isključujući P=1) nižeg reda od posmatranog i ako se dobija dijeljenje sa ostatkom u svim slučajevima u pitanju je prost polinom. Kod polinoma trećeg reda prosti su x^3+x+1 i x^3+x^2+1.

Množenje polinoma u MATLAB-u se vrši naredbom `conv` međutim da bi se zadržala osobina da dobijemo polinome sa binarnim koeficijentima moramo izvršiti operaciju ekskluzivno ili odnosno sabiranje po modulu 2 odnosno odrediti ostatak dijeljenja sa 2:

```
>> c1=[1 0 1 1]; c2=[1 0 0 1];
>> rem(conv(c1,c2),2)
```

Prije nego detaljnije uđemo u materiju vezanu za aritmetičku kodnu teoriju koja je kompleksna i koju ćemo na ovom nivou razmatrati samo fenomenološki uvedimo neke elemente ove moćne teorije na nivou primjera. Neka želimo da kreiramo Hammingov kod (7,4) koristivši prosti polinom x^3+x+1 (pretpostavimo da to može da se uradi, a vidjećemo da može). Kodne riječi ovog polinoma ima najveći šesti stepen (do člana x^6). Pretpostavimo da su sve kodne riječi kreirane po pravilu da su djeljive bez ostatka sa prostim polinomom x^3+x+1. Da bi to bilo zadovoljeno kontrolna matrica koda mora da zadovoljava posebna pravila. Jedno od tih pravila je da u kolonama kodne matrice moraju da stoje stepeni nula polinoma x^3+x+1. Naime, ako je jedan polinom djeljiv sa drugim polinomom mora da se može prikazati kao polinom koji se anulira za istu nulu kao i osnovni polinom (ili za stepen te nule). Postavlja se pitanje što je nula polinoma x^3+x+1. Očigledno da nijedan od binarnih brojeva $x=0$ i $x=1$ ne zadovoljava ovu osobinu. Može se pokazati da ne postoji ni vektor sa dva elementa za koji se predmetni polinom anulira pa stoga je vektor najmanjeg reda za koji se poništavanje dešava vektor od tri člana. Stoga se može konvencionalno usvojiti da je ta nula za koju se ovaj prosti polinom $P(x)=x^3+x+1$ anulira $P(a)=0$ jednaka:

$$a=\begin{bmatrix}0\\1\\0\end{bmatrix}$$

Usvojimo još da je:

$$a^0=1=\begin{bmatrix}0\\0\\1\end{bmatrix}\quad a^2=\begin{bmatrix}1\\0\\0\end{bmatrix}$$

Napominjem da bez gubitka opštosti se moglo protpostaviti da su prva tri stepena nule prostog polinoma bilo koja tri različita nenulta vektora dužine 3.

U našoj aritmetici važi da je $a^3=1+a$ pa se svi viši redovi od 3 ovog korjena mogu zapisati preko polinoma koji je reda manjeg ili jednakog 2:

$$1=\begin{bmatrix}0\\0\\1\end{bmatrix}\quad a=\begin{bmatrix}0\\1\\0\end{bmatrix}\quad a^2=\begin{bmatrix}1\\0\\0\end{bmatrix}\quad a^3=\begin{bmatrix}0\\1\\1\end{bmatrix}\quad a^4=a^2+a=\begin{bmatrix}1\\1\\0\end{bmatrix}$$

$$a^5=a^2+a+1=\begin{bmatrix}1\\1\\1\end{bmatrix}\quad a^6=a^2+1=\begin{bmatrix}1\\0\\1\end{bmatrix}\quad a^7=a^3+a=\begin{bmatrix}0\\0\\1\end{bmatrix}$$

Dakle $a^7=a^0$. Slična procedura se obavlja i kod prostih polinoma višeg reda. Tada dimenzija stepena nule prostog polinoma mora da bude jednaka (najvećem) stepenu polinoma. Stepene nule polinoma sada možemo napisati u obliku već relativno dobro poznate kontrolne matrice Hammingovog koda ali sada sa nešto izmjenjenim redosljedom kolona, što se pokazalo nebitnim (dobijeni kod je

ekvivalentan do sada uvedenim Hammingovim kodovima kao i svi kodovi koji se dobijaju prostom zamjenom redosljeda kolona):

$$\begin{matrix} a^6 & a^5 & a^4 & a^3 & a^2 & a & 1 \end{matrix}$$
$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

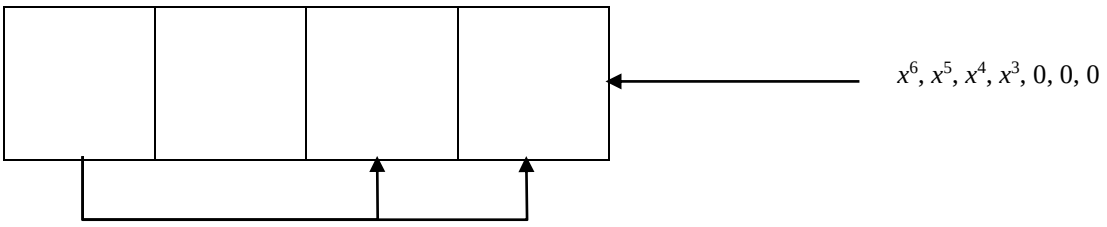
Slična se matrica može dobiti kada se koristi polinom x^3+x^2+1. Kako vršimo kodiranje? Pošto želimo da sindrom 0, 0, 0, ..., 0 znači da nema grešaka, moramo zahtjevati da poslati polinom bude djeljiv sa polinomom a^3+a+1. Možemo poslati poruku u stepenima a^6, a^5, a^4 i a^3 i privremeno postaviti nule na ostalim pozicijama. Kada izvršimo dijeljenje polinoma dobijamo odgovarajući ostatak. Ostatak zapravo predstavlja ono što trebamo dodati našem kodu kako bismo dobili kod koji se može dekodirati. Na prijemnom kraju prosto dijelimo polinom poruke sa prostim polinomom i onaj polinom koji ostaje je poruka.

Primjer kodiranja. 1 0 0 1 - - - ovome odgovara polinom a^6+a^3. Izvršimo dijeljenje sa a^3+a+1. Dobijamo ostatak a^2+a. Ovo znači da je kod koji treba poslati (1 0 0 1 1 1 0). Sada pretpostavimo grešku na 4 lokaciji: (1 0 0 0 1 1 0). Dijeljenjem odgovarajućeg polinoma dobijamo ostatak $a+1$ (0 1 1). Ostatak je dakle: $a^3=a+1$ što predstavlja četvrtu poziciju u odgovarajućoj matrici.

$$\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = a^3$$

Algebarska teorija je vodila jednostavnom kreiranju kodova hardverskim i softverskim putem. Najkorisnija hardverska komponenta je pomjerački registar. Pretpostavimo da šaljemo poruku 1 0 0 1 to znači da polinom koji želimo poslati je:

$$1 \cdot a^6 + 0 \cdot a^5 + 0 \cdot a^4 + 1 \cdot a^3 + *a^2 + *a + *1$$
$$a^6 + a^3 + a^2 + a = (a^3 + a + 1)(a^3 + a)$$



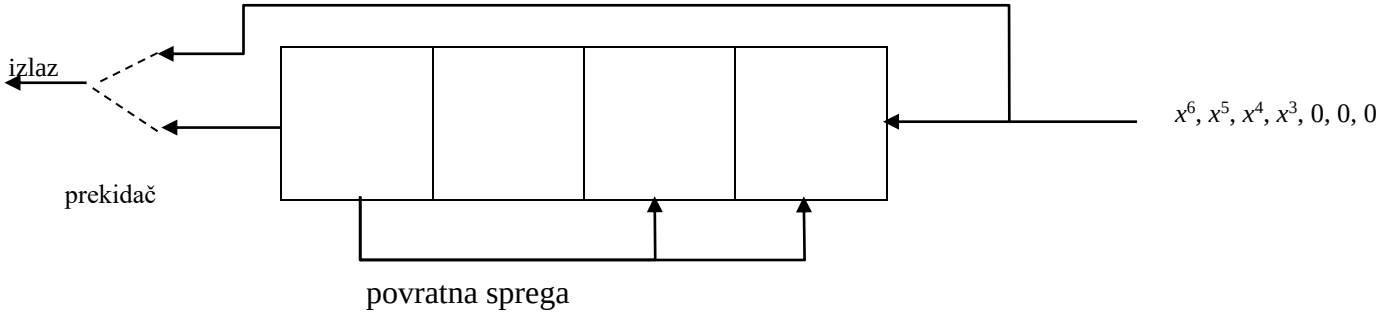
povratna sprega

Povratnu spregu možete razumjeti kao operaciju ekskluzivno ili stanja u prvoj ćeliji (prvom flip flopu) sa ostalim elementima na koje se odnosi. Da je polinom na osnovu kojeg je kod kreiran bio x^3+x^2+1 povratna sprega bi vodila sa prve na drugu i četvrtu ćeliju.

Stanje
povr.spreg.
Rezultat
Pomjeraj

1	0	0	1	0 0 0
1	0	1	1	
0	0	1	0	
0	1	0	0	0 0
0	0	0	0	
0	1	0	0	
1	0	0	0	0
1	0	1	1	
0	0	1	1	
0	1	1	0	
0	0	0	0	
0	1	1	0	

Dobijeni ostatak na kraju predstavlja posljednja 3 bita poruke. Kao kontrolni mehanizam da li je procedura obavljena korektno može da nam posluži činjenica da je nakon povratne sprege u prvoj ćeliji uvijek upisana 0 (treći red svakog pojedinačnog "takta"). Koder se sada može prikazati kao:



Koder radi tako što tokom nailaska informacionih bita propusti sadržaj informacionih bita, a zatim u narednim taktovima kupi sadržaj iz pomjeračkog registra.

Ovo se može veoma efikasno programski implementirati na sljedeći način.

```
clear
info_rijec=[1 0 0 1];
registar=info_rijec; %%nakon punjenja registra bitima informacione rijeci
for k=1:3
    registar=rem(registar+registar(1)*[1 0 1 1],2); %%povratna sprega i xor
operacija
    registar=[registar(2:4) 0];
end
kodna_rijec=[info_rijec,registar(2:4)]
```

Program je, nadamo se, jasan. Pokušajte ga generalizovati da može da radi i za druge tipove kodova te ga napraviti i u obliku programske funkcije.

Predmetni Hammingov kod, kao i mnogi drugi koji se mogu na sličan način uvesti pripada grupi cikličnih kodova. Kod je cikličan ako je riječ $\mathbf{c}=[c_1,c_2,c_3,...,c_n]$ ispravna kodna riječ, onda je svaka riječ nastala cikličnim pomjeranjem ove riječi kodna riječ datog koda. Osobinu ćemo dokazati u posebnom slučaju koda nastalog na osnovu prostog polinoma x^3+x+1. Ako je kodni polinom sa koeficijentom 0 uz član x^6 tada pomjeranje ulijevo za 1 zapravo predstavlja množenje tog polinoma sa x. Ako je polazni polinom djeljiv sa prostim polinomom, tada je i dobijeni polinom djeljiv sa njime (samo je količnik pomnožen sa x). Situacija se malo komplikuje, ako je koeficijent uz najveći član 1. Tada polazni polinom za kojeg pretpostavljamo da je prost možemo zapisati kao:

$$\frac{x^6 + p(x)}{x^3 + x + 1} = x^3 + q(x)$$

gdje je polinom $p(x)$ reda većeg ili jednakog x^5, polinom $x^3+q(x)$ je količnik, dok je $q(x)$ polinom reda ne većeg od 2. Ako ovaj polinom pomjerimo ulijevo za 1 dobijamo polinom $p(x)x+1$ za kojeg treba dokazati da je prost. Iz prve relacije jednostavno slijedi:

$$p(x)=x^4+x^3+q(x)(x^3+x+1)$$

pa se problem svodi na dokaz da je polinom:

$$x^5+x^4+xq(x)(x^3+x+1)+1$$

odnosno što je još jednostavnije potrebno je dokazati da je polinom x^5+x^4+1 djeljiv sa x^3+x+1. Količnik ova dva polinoma je: x^2+x+1 a djeljenje je obavljen bez ostatka čime smo u stvari dokazali tvrdnju za kod baziran na ovom prostom polinomu.

Postoji i drugi način da se možda nešto sistematičnije provjeri osobina da li je kod cikličan. Naime, ako je prvi bit u kodnoj riječi jednak 1 polinom se može zapisati kao:

$$x^{n-1}+q(x)$$

gdje je $q(x)$ polinom koji je najviše reda n-2. Pomnožimo ovaj polinom sa x i dobijamo:

$$x^n+xq(x)$$

Pod pretpostavkom da je polazni polinom djeljiv sa prostim generatorskim polinom onda je i ovaj polinom djeljiv sa prostim generatorskim polinomom. Međutim, mi umjesto ovog polinoma imamo polinom $xq(x)+1$ koji se od osnovnog polinoma, koji je djeljiv sa prostim polinomom, razlikuje u tome što je oduzeto x^n i dodato 1. Zaključujemo da ako se želi održati djeljivost sa prostim polinomom, koji je korišćen kao generator koda, mora da važi da je x^n-1 djeljivo sa tim prostim polinomom. U našoj binarnoj algebri sa binarnim alfabetom ovo znači da x^n+1 mora biti djeljivo sa prostim polinomom. Postoje i nebinarni alfabeti. I kod tih alfabeta u pridruženoj algebri važi prethodno izloženo pravilo (x^n-1) mora biti djeljivo sa prostim polinomom koji se koristi za generisanje koda). Jedino je pitanje što je u toj algebri operacija oduzimanja ali će o ovome biti više riječi kasnije.

7.4. Dekodiranje kodova za korekciju jedne greške

Dekodiranje kodova za korekciju jedne greške se obavlja na veoma prost način. Dobijena kodna riječ se pretvori u polinom i podjeli sa prostim polinomom. Ukoliko se dobije ostatak (sindrom) koji je jednak nuli u pitanju je ispravna kodna riječ i količnik je jednak informaciji. Ako se dobije ostatak dijeljenja taj polinom se može zapisati u obliku stepena a^k i greška se dogodila na poziciji koja odgovara datoj koloni matrice. Pretpostavimo da imamo poruku 1 0 0 1 1 1 0 i da je greška nastupila na poziciji a^2 1 0 0 1 0 1 0. Ostatak pri dijeljenju će dati 1 0 0. Sada se rješava jednačina:

$$a^k = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

za koju je jasno da je k=2 (na osnovu uvida u kontrolnu matricu). Kako želimo izbjeći rad sa kontrolnom matricom možemo vršiti množenje sa a^{-1} onoliko puta koliko je potrebno da dobijemo vektor a^0. Množenje sa a^{-1} je jednako sa množenjem sa a^6 jer je $a^6=a^7a^{-1}=a^0a^{-1}=a^{-1}$. U opštem slučaju se vrši množenje sa a^{n-1}. Množenje je ekvivalentno pomjeranju na dolje elemenata vektora sa upisivanjem nula na vrhu vektora. Ako se na dnu vektora pojavi element 1 ovaj elemenat onda se u narednom koraku vrši sabiranje pomjerenog ostatka sa $a^6=1+a^2$. Ovo se može shvatiti i kao da smo u prethodnom koraku umjesto 1 imali $1=a+a^3$. U našem jednostavnom slučaju se dobija:

$$a^k a^6 = a^{k-1} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad a^{k-1} a^6 = a^{k-2} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 1$$

pa se zaključuje da je k=2.

Prikažimo kako se može programirati dekodiranje u slučaju ovog koda. Uzmimo kodiranje sa kodnim polinomom x^3+x+1:

```
c=rem(conv([1 0 0 1],[1 0 1 1]),2)
c =     1     0     1     0     0     1     1
```

Unesimo grešku na poziciju 4, `c(4)=1`. Dijeljenje polinoma u MATLAB-u možemo izvršiti naredbom deconv:

```
[q,r]=deconv(c,[1 0 1 1]);
```

Potrebno je rezultate vratiti u domen binarnih brojeva:

```
q=rem(q,2);
r=rem(r,2);
r=r(end-2:end);
```

Posljednja naredba je potrebna kako bi se

```
k=0;
while sum(abs(r-[0 0 1]))~=0
k=k+1;
rc=conv(r,a6);
r=rem(rc(3:5)+rc(2)*[0 1 1]+ rc(1)*[1 1 0],2);
end
```

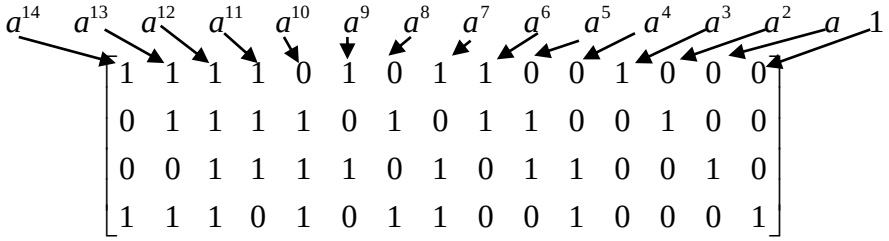
Nakon k=3 ponavljanja dobili smo da je $a^r=a^0$. To ukazuje da je pozicija pogreške `c(end-k)=xor(c(end-k),1)`. Za vježbu napisati program koji će u opštem slučaju za različite generatorske polinome vršiti dekodiranje. Detaljno protumačiti navedeni program.

7.5. Kodovi sa korekcijom dvije greške

Kod koda sa dvostrukom korekcijom greške moramo poći od Hammingovih kodova sa većim brojem bita. Posmatrajmo slučaj sa 15 bita. Za sada posmatrajmo uvođenje 4 provjere parnosti. Kao jedan potencijalni polinom može poslužiti x^4+x+1. Može se pokazati da je a^{15}=1. Naime važi:

$$1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad a = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad a^2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Na osnovu primitivnih korjena dobijamo kontrolnu matricu:



Ako se želi korigovati više grešaka nego jednu mora se dodati još nekoliko kolona matrice. Ovdje se dodaje četiri nova reda. U sadašnjoj situaciji dvije greške mogu da produkuju isti rezultat za sindrom:

$$a^k + a^m = a^{k'} + a^{m'}$$

Npr. kolone a i a^2 daju isti sindrom kao kolone a^{12} i a^{14}. Posmatrajmo sledeći primjer:

$$a_1 + a_2 = s_1 = \text{prvi sindrom} \qquad a_1^2 + a_2^2 = s_2 = \text{drugi sindrom}$$
$$s_1^2 = (a_1 + a_2)^2 = a_1^2 + 2a_1a_2 + a_2^2 = a_1^2 + a_2^2 = s_2$$

Ovo ne ide, pa posmatrajmo sledeći sindrom:

$$s_2 = a_1^3 + a_2^3$$
$$s_2 = (a_1 + a_2)(a_1^2 + a_1a_2 + a_2^2) = s_1(a_1^2 + a_1a_2 + a_2^2) = s_1(s_1^2 + a_1a_2)$$

Odavde slijedi:

$$a_1 + a_2 = s_1 \qquad a_1a_2 = s_1^2 + s_2 / s_1 \quad (s_1 \neq 0)$$

Ovo se može svesti na rješavanje kvadratne jednačine:

$$a^2 - (s_1 a) + \left(s_1^2 + \frac{s_2}{s_1} \right) = 0$$

Ako postoji samo jedna greška sindromi su $a_1 = s_1$ $a_1^3 = s_2$ i $s_1^2 + s_2 / s_1 = 2s_1^2 = 0$ pa kvadratna jednačina postaje $a^2 + s_1 a = 0$ ili a=s_1 ili a=0. Kada nema grešaka ovo je trivijalno s_1=s^2=0 a^2=0 a=0.

Kontrolna matrica se sada može prikazati kao nadovezana kodna matrica koda (15,11) sa kodnom matricom, čije kolone su 0, 3, 6, 9, 12 stepen osnovne matrice. Ostale kolone predstavljaju ponavljanje osnovnih kolona.

$$\left[\begin{array}{cccccccccccccc}
1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
\hdashline
1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1
\end{array}\right]$$

Pretpostavimo sada da se greška dogodila na poziciji koja odgovara k_1=14 i k_2=5. Dobijeni sindrom je jednak:

$$S = \begin{bmatrix} s_1 \\ \hline s_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ \hline 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} a^{12} \\ \hline a^{11} \end{bmatrix}$$

Ovo znači da je $a_1+a_2=a^{12}$ i da je $a_1a_2=a^{11}/a^{12}\text{-}a^{24}=a^{14}+a^9=a^4$ (pogledajte kontrolnu matricu). Sada se rješenje može potražiti 1D pretraživanjem uzimajući da je $a_1=a^k$. Tada je $a_2=a^4/a^k=a^{(19-k) \bmod 15}$. Zbog čega smo izvršili ovakvu modifikaciju eksponenta? Sada se rješenje traži mijenjanjem k u granicama od k=0 do k=14, dok se ne dobije zadovoljenje prvog uslova $a_1+a_2=a^k+a^m=a^{12}$. Lako se pokazuje da rješenja odgovaraju k=5 i m=14 odnosno kolonama sa a^5 i a^{14}. Prilikom pretraživanja čitav niz kombinacija k i m se ne uzima u obzir čime se donekle redukuje složenost ovog postupka.
Na primjer k=0 podrazumijeva m=4, sledeća kombinacija koja se provjerava je k=1 i m=3, kombinacija k=2 i m=2 se ne provjerava jer je u pitanju jednostruka pogreška a da se dogodila jednostruka pogreška sindrom bi bio jednak nekoj od kolona kontrolne matrice, k=3 i m=1 smo već provjerili (k=1 i m=3), kao i k=4 i m=0 pa ove kombinacije preskačemo. Kod ovog načina rada javlja se nekoliko problema: kontrolna matrica nije u formi koja bi nam razdvojila kontrole i informacione bite; problem predstavlja i način dalje generalizacije ovog problema koji očigledno nije trivijalan zbog rješavanja sistema jednačina; u ovom slučaju se javlja 1D pretraživanje dok tražimo rješenje, dok je vjerovatna situacija da ćemo za više grešaka imati 2D ili 3D pretraživanje; konačno, dobro bi bilo izbjeći korišćenje "obimne" kontrolne matrice *H*. Važno je zapaziti da mi zapravo nijesmo nijednom riječju pomenuli način kodiranja ovog koda jer će nam trebati neki mnogo sofisticiraniji metod od do sada uvedenih koji su bazirani na jednostavnim pravilima provjera parnosti.

U formi vježbe kreirajmo matricu *H* za gorenavedeni kod.

```
H=zeros(4,15);
H(:,15)=[0 0 0 1]';
for r=14:-1:1,H(:,r)=rem([H(2:4,r+1);0]+H(1,r+1)*[0 0 1 1]',2);end
H
  1  1  1  1  0  1  0  1  1  0  0  1  0  0  0
  0  1  1  1  1  0  1  0  1  1  0  0  1  0  0
  0  0  1  1  1  1  0  1  0  1  1  0  0  1  0
  1  1  1  0  1  0  1  1  0  0  1  0  0  0  1
H1=zeros(4,15);
for r=15:-1:1
H1(:,r)=H(:,15-rem(3*(15-r),15));
end
H=[H;H1];
H =
  1  1  1  1  0  1  0  1  1  0  0  1  0  0  0
  0  1  1  1  1  0  1  0  1  1  0  0  1  0  0
  0  0  1  1  1  1  0  1  0  1  1  0  0  1  0
  1  1  1  0  1  0  1  1  0  0  1  0  0  0  1
  1  1  1  1  0  1  1  1  1  0  1  1  1  1  0
  1  0  1  0  0  1  0  1  0  0  1  0  1  0  0
  1  1  0  0  0  1  1  0  0  0  1  1  0  0  0
  1  0  0  0  1  1  0  0  0  1  1  0  0  0  1
```

Za vježbu napisati program koji je u stanju da dekodira pojavljivanje jedne odnosno dvije greške kod ove kontrolne matrica.

7.6. Blok kodovi dobijeni množenjem polinoma

Hammingov kod koji ispravlja jednu pogrešku smo preko polinoma dobili na taj način što smo informacione bite postavili na prvim mjestima kodne riječi, a zatim smo dodali kontrolne bite tako da odgovarajući polinom bude djeljiv sa odabranim prostim polinomom. Moguć je i drugačiji postupak u kojem se polinom koji predstavlja informacione bite množi sa prostim polinomom:

$$\mathbf{c}(x)=\mathbf{i}(x)\mathbf{p}(x)$$

Ovako dobijeni polinom je zasigurno djeljiv sa prostim polinomom $\mathbf{p}(x)$. Uzmimo primjer informacione riječi sa 4 bita $\mathbf{i}(x)=i_3x^3+i_2x^2+i_1x+i_0$ i polinoma $\mathbf{p}(x)=x^3+x+1$. Dobijeni kodni polinom je:

$$c(x)=i_3x^6+i_2x^5+(i_3+i_1)x^4+(i_3+i_2+i_0)x^3+(i_2+i_1)x^2+(i_1+i_0)x+i_0$$

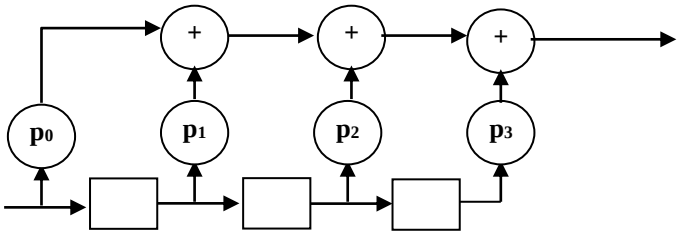
Uočavamo da samo tri bita kodnog polinoma (prvi, drugi i posljednji) informacioni, dok su ostali biti kombinacija informacionih (u stvari umjesto da kod ima četiri informaciona bita i 3 provjere parnosti mi imamo 3 informaciona bita i 4 provjere parnosti, ali takve da nijesu međusobno nezavisne sa time da su podaci o informacionim bitima izmješani sa podacima o provjerama parnosti pa i dalje zapravo možemo "izvući" četiri bita informacije iz kodne poruke).

Ovakav sistem se može realizovati nešto jednostavnije nego prethodno opisani Hammingov koder koji je zahtjevao pomjeračke registre sa povratnom spregom. Naime, ova realizacija zahtjeva samo običan pomjerački registar.

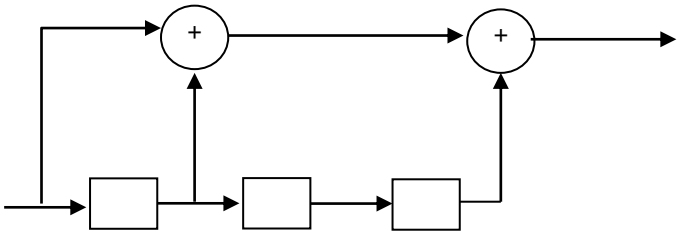
Ilustrujmo opšti oblik realizacije koda na osnovu množenja polinoma koji predstavlja informacione bite sa generatorskim polinomom koda. Registar koji se koristi mora da ima onoliko bita koliko je red informacionog polinoma (u našem slučaju maksimalni red informacionog polinoma je x^3, pa su potrebne 3 ćelije registra). U opštem slučaju za bilo koji prosti polinom 3 reda rezultat se dobija kao proizvod:

$$c(x)=i(x)p(x)=(i_3x^3+i_2x^2+i_1x+i_0)(p_3x^3+p_2x^2+p_1x+p_0)=$$
$$i_3p_3x^6+(i_3p_2+i_2p_3)x^5+(i_3p_1+i_2p_2+i_1p_3)x^4+(i_3p_0+i_2p_1+i_1p_2+i_0c_3)x^3+(i_2p_0+i_1p_1+i_0p_2)x^2+(i_1p_0+i_0p_1)x+i_0p_0$$

Hardverska realizacija ovog sistema je data na slici. Kružići sa upisanim koeficijentima p_i predstavljaju logičku operaciji I (ako nije binarni kod to je neka drugačija operacija, ali u te detalje za sada ne ulazimo). Najčešće se operacija I i ne realizuje već se ako je p_i=0 ta grana prekida, a ako je p_i=1 postavlja se "kratak spoj". Na početku su svi registri dovedeni na nulu. Ulazna sekvence dolazi po redosljedu: [i_0, i_1, i_2, i_3] dok izlazna sekvenca izlazi po redosljedu [c_0, c_1, c_2, c_3, c_4, c_5, c_6], gdje su c_i koeficijenti uz stepene x^i. Pogledajmo sada prvih nekoliko taktova. Na ulaz stiže i_0 a na izlaz stiže i_0p_0 dok iz ostalih izlaze sve nule tako da na izlaz stiže i_0p_0 što je jednako c_0. Ujedno informacioni bit ulazi u prvu ćeliju registra "pomjerajući" nulu koja se u njemu nalazila. U narednom taktu na izlaz direktno ide bit i_1 pomnožen sa p_0, ali i iz prve ćelije registra izlazi bit i_0 pomnožen sa p_1. Dobija se da je: $c_1=i_0p_1+i_1p_0$. U prvu ćeliju registra ulazi bit i_1 koji pomjera bit i_0 u narednu ćeliju. U trećem taktu se na izlazu dobija i_2p_0 direktno zatim i_1p_1 na osnovu i_1 koji je bio smješten u prvoj ćeliji i i_0p_2 na osnovu bita i_0 koji je bio smješten u drugoj ćeliji procesora. Dobijen je bit izlazne poruke $c_2=i_0p_2+i_1p_1+i_2p_0$. Procedura se nastavlja za naredne bite. Nakon ulaska bita i_3 u prvi registar ulaz se tretira kao 0 i ta nula ulazi tokom narednih nekoliko taktova u naredne registre pripremajući kolo za kodiranje naredne kodne riječi. Sami propratite naredne korake u algoritmu.



Kolo koje realizuje kodiranje na osnovu prostog polinoma x^3+x+1 je dato na narednoj slici.



Za vježbu sami kreirajte koder za kod koji je kreiran na osnovu prostog polinoma x^3+x^2+1 kao i Hammingove kodere za kodove tipe (15,11). Dekodiranje ovako dobijenog koda se može obaviti dijeljenjem sa odgovarajućim prostim polinomom. Može se smatrati da ako je dobijeni rezultat jednak 0 da je dijeljenje obavljeno uspješno. O ovim operacijama će nekoliko riječi više biti na narednom času kada ćemo obraditi i BCH kodiranje.

Kao što smo se osvjedočili kodiranje je relativno jednostavan problem. Hardverska realizacija dekodiranja može biti znatno komplikovanija. Kod kodova koji su kreirani putem cikličnog registra situacija je nešto jednostavnija i pokazali smo način određivanja sindroma za slučaj Hammingovog koda (7,4). Situacija je dosta složenija kod kodova koji su realizovani putem pomjeračkog registra koji vrši isključivo množenje informacionog sa prostim polinomom. Detalji hardverskog dekodiranja ovakvih sistema se mogu pronaći u literaturi a uglavnom su zasnovani na Meggittovoj teoremi možete potražiti u literaturi. Softverski sistemi polaze od dijeljenja polinoma i određivanja sindroma. Zbog performansi u praktičnim sistemima se koristi hardversko dekodiranje.

Programska realizacija za kodiranje ovog koda je krajnje jednostavna. Potrebno je samo izvršiti konvoluciju (po modulu 2) polinoma koji predstavlja informacione bite sa generatorskim polinomom (u MATLAB-u je koeficijent najvišeg reda na početku vektora):

```
i=[1 1 1 0];
g=[1 0 1 1];
c=rem(conv(i,g),2);
```

Dekonvolucija ako nije bilo pogreški daje:

```
[q,r]=deconv(c,g)
q=abs(rem(q,2)), r=rem(r,2)
q =
     1     1     1     0
r =
     0     0     0     0     0     0     0
```

U slučaju da postoji greška ostatak pri dekonvoluciji (dijeljenju polinoma) neće biti jednak nultom vektoru pa se potraga za pozicijom pogreške može obaviti na više načina. Idejno najprostiji način je da se provjerava bit po bit i da se nakon zamjene provjeri djeljivost sa generatorskim polinomom. Međutim, operacija djeljenja polinoma nije jednostavna pa bi ova pretraga uz navedenu operaciju nad polinomima pune dužine bila neprihvatljiva. Stoga se obično operacija vrši na isti način kako je

urađeno u slučaju kodova sa cikličnim registrom koji su prethodno objašnjeni ili se operacija može obaviti na način kako će to biti objašnjeno u narednom poglavlju kada budemo radili sa klasom kodova koja se naziva BCH.

Dobra je prilika da se upoznamo sa osnovnim naredbama za rad sa Hammingovim kodom u MATLAB-u. Naime, MATLAB je kreirao komunikacioni toolbox koji omogućava rad sa nizom popularnih blok kodova na pojednostavljen način. Ovdje ćemo se upoznati sa naredbama koje se odnose na Hammingovim kodom. Naredba `hammgen` formira kontrolu i generatorsku matricu koda. Argument ove naredbe je broj mjesta parnosti:

```
[kont,gener]=hammgen(3);
```

Generatorski polinom se može dobiti naredbom `cycpoly` čiji su argumenti dužina koda i broj mjesta parnosti:

```
genpoly = cyclpoly(7,3);
```

Naredba `cyclgen` sa dva argumenta – dužinom kodne riječi i generatorskim polinomom daje nam kontrolnu i generatorsku matricu koda:

```
[kont,gener]= cyclgen(7,genpoly);
```

Naredba `gen2par` konvertuje generatorsku u kontrolnu matricu koda. Napominjem da je suprotno od MATLAB-ove logike u radu sa polinomima ovdje generatorski polinomi imaju najniži red na prvom mjestu u vektoru koji predstavlja polinom. Ostatak priče je u principu već ispričan. Ako posjedujemo primljenu riječ sindrom se može odrediti prostim množenjem riječi sa kontrolnom matricom po modulu 2:

```
sindrom=rem(c*kont',2);
```

Sada se može na odgovarajući način utvrditi pozicija u kontrolnoj matrici koja odgovara sindromu itd. Dalje opcije će biti objašnjene nakon narednog poglavlja kada se upoznamo sa BCH kodovima.

Zadaci za vježbu

7.1. Kod koda sa provjerom parnosti koji je dobijen sa jednim bitom parnosti za p=0.001 i n=100 odrediti vjerovatnoću da nema greške u poruci, da je greška detektovana i da greška nije detektovana. Ako je vjerovatnoća greške na bitu p=0.01 koliko je potrebno n da bi se postigla vjerovatnoća nedetektovane pogreške manja od 0.005. Koliko je n maksimalno koje ovo omogućava? Za veoma malo p i veliko n kolika je aproksimativno vjerovatnoća da nema greške u poruci.

Rješenje: Vjerovatnoća da nema pogreške je jednaka:

$$(1-p)^n=0.9048$$

Vjerovatnoća da je greška detektovana je jednaka vjerovatnoći da u kodu postoji neparan broj pogreški što u konkretnom slučaju znači:

$$\sum_{r=1}^{50}\binom{n}{2r-1}p^{2r-1}(1-p)^{101-2r}\approx 0.0907$$

Vjerovatnoća da se greška ne detektuje je jednaka vjerovatnoći da postoji paran broj pogreški:

$$\sum_{r=1}^{50}\binom{n}{2r}p^{2r}(1-p)^{100-2r}\approx 0.0045$$

Za p=0.01 najveći broj za koji se postiže vjerovatnoća nedetektovane pogreške koja je manja od 0.05 je n=10 za koju ova vjerovatnoća iznosi 0.042:

$$\binom{10}{2}p^2(1-p)^8+\binom{10}{4}p^4(1-p)^6+\binom{10}{6}p^6(1-p)^4+\binom{10}{8}p^8(1-p)^2+p^{10}=0.042$$

Vjerovatnoća da nema greške u sistemu je:

$$(1\text{-}p)^n=1\text{-}np+n(n\text{-}1)p^2/2+...\approx 1\text{-}np$$

7.2. Koja je vjerovatnoća da Hammingov kod sa k informacionih bita neće moći da ispravi grešku u prenosu. Greške na svim bitima su jednake i međusobno nezavisne. Kolika je vjerovatnoća da Hammingov kod za ispravljanje jedne i detekciju dvije pogreške: (a) Nema grešaka u prenosu; (b) ispravi jednu pogrešku; (c) detektuje dvije pogreške; (d) ne može da posluži svojoj svrsi.

Rješenje: Pojednostavimo priču na taj način da kod ima n bita. Vjerovatnoća za slučaj da nema pogreške je $(1\text{-}p)^n$ dok je vjerovatnoća da se desi jedna greška i da je ona koriguje $np(1\text{-}p)^{n-1}$ konačno vjerovatnoća detekcije dvije pogreške je $n(n\text{-}1)p^2(1\text{-}p)^{n-2}/2$. U slučaju većeg broja pogreški kod neće moći služiti svojoj svrsi međutim tu se razlikuje slučaj parnog broja pogreški kada sistem detektuje da je do greške došlo dok kod neparnog broja pogreški sistem ispravlja "jednu pogrešku". Konačno ako je n paran broj i na svim bitima imamo pogrešku ta će se poruka shvatiti kao ispravna. Za vježbu formulišite izraze za navedene vjerovatnoće.

7.3. (a) Koliko je Hammingovo rastojanje između dva vektora koji predstavljaju susjedna tjemena n-dimenzione sfere. (b) Kolika je Hammingova distanca između poruke x i njoj komplementarne poruke ako su poruke od n bita.

Rješenje: (a) Hammingovo rastojanje dva susjedna tjemena na sferi je 1 jer se mogu razlikovati samo za jedan bit. (b) Komplementarne kodne riječi se razlikuju na svakom bitu a to znači da je Hammingovo rastojanje među navedene dvije riječi jednako dužini riječi n.

7.4. Kako izgleda kontrolna matrica Hammingovog koda (15,4). Kreirati poruku od 11 bita i odrediti kontrolne bite u ovom slučaju. Promjeniti jedan bit i prikazati proceduru za određivanje na kojoj se poziciji dogodila greška. Proceduru ponoviti u dva slučaja kada su bitovi parnosti na pozicijama 1, 2, 4 i 8 i kada su to posljednja četiri bita u poruci.

Rješenje: Neka je informaciona riječ **i**=[0 1 0 1 1 0 1 1 1 0 0]. U prvoj varijanti kodna riječ je:

c=[___ ____ 0 ____ 1 0 1 ____ 1 0 1 1 1 0 0]

Prvi bit kontroliše neparne pozicije pa je jednak 1, drugi bit kontroliše pozicije 2, 3, 6, 7, 10, 11, 14, 15 i bit je jednak 1, treći bit parnosti kontroliše pozicije 4, 5, 6, 7, 12, 13, 14 i 15 pa je ovaj bit 0, dok poslednji bit parnosti kontroliše poslednjih 8 bita i iznosi 0. Dakle, kodna riječ je: **c**=[1 1 0 0 1

0 1 0 1 0 1 1 1 0 0]. Pretpostavimo da se greška dogodila na šestoj poziciji u kodu pa da je primljena riječ: **r**=[1 1 0 0 1 1 1 0 1 0 1 1 1 0 0]. Kontrolna matrica ovog koda je:

$$\mathbf{H}=\begin{bmatrix}1&0&1&0&1&0&1&0&1&0&1&0&1&0&1\\0&1&1&0&0&1&1&0&0&1&1&0&0&1&1\\0&0&0&1&1&1&1&0&0&0&0&1&1&1&1\\0&0&0&0&0&0&0&1&1&1&1&1&1&1&1\end{bmatrix}$$

Sindrom je jednak:

$$\mathbf{S}=\mathbf{r}\mathbf{H}^{T}=[0\,1\,1\,0]^{T}$$

Jasno se pokazuje da je greška na šestom bitu jer je sindrom jednak šestoj koloni. Ispravljena riječ je: **c**=[1 1 0 0 1 0 1 0 1 0 1 1 1 0 0] a dekodirana riječ je:

$$\mathbf{i}=[0\,1\,0\,1\,1\,0\,1\,1\,1\,0\,0]$$

Posmatrajmo sada varijantu da su simboli parnosti na posljednjim pozicijama u kodu. Kontrolna matrica ovog koda može da bude preuređena matrica prethodnog koda:

$$\mathbf{H}=\begin{bmatrix}1&1&0&1&1&0&1&0&1&0&1&1&0&0&0\\1&0&1&1&0&1&1&0&0&1&1&0&1&0&0\\0&1&1&1&0&0&0&1&1&1&1&0&0&1&0\\0&0&0&0&1&1&1&1&1&1&1&0&0&0&1\end{bmatrix}$$

Ako je informaciona riječ ista kao u prethodnom slučaju **i**=[0 1 0 1 1 0 1 1 1 0 0], kodna riječ će imati sledeći oblik **c**=[0 1 0 1 1 0 1 1 1 0 0 1 1 0 0]. Pretpostavimo pogrešku na šestom bitu **r**=[0 1 0 1 1 1 1 1 1 0 0 1 1 0 0]. Sindrom u ovom slučaju je:

$$\mathbf{S}=\mathbf{r}\mathbf{H}^{\mathrm{T}}=[0\,1\,0\,1]$$

Ponovo zaključujemo da se pogreška dogodila na šestoj poziciji u kodu koji možemo da ispravimo i pravilno da dekodiramo kao [0 1 0 1 1 0 1 1 1 0 0].

7.5. Posmatrajte Hammingov kod sa 4 informaciona bita koji može da ispravi jednu i detektuje dvije greške. Kreirati kontrolnu matricu koda. Uzeti četiri proizvoljna bita i kreirati odgovarajući Hammingov kod. Zatim promjeniti jedan pa dva bita respektivno i prikazati način dekodiranja ovog koda.

Rješenje. Pošto drugačije nije specificirano uzećemo da su biti parnosti na pravilnim binarnim pozicijama. Kontrolna matrica koda u ovom slučaju je:

$$\mathbf{H}=\begin{bmatrix}1&0&1&0&1&0&1&0\\0&1&1&0&0&1&1&0\\0&0&0&1&1&1&1&0\\1&1&1&1&1&1&1&1\end{bmatrix}$$

Uzmimo da je informaciona riječ **i**=[1 0 1 1] tada je kodna riječ **c**=[0 0 1 0 0 1 1 1]. Promjenimo prvi bit **r**=[1 0 1 0 0 1 1 1]. Pomnožimo ovu kodnu riječ sa kontrolnom matricom čime dobijamo sindrom:

$$\mathbf{S}=\mathbf{r}\mathbf{H}^T=[1\ 0\ 0\ 1]$$

Kako je broj jedinica u kodnoj riječi neparan to znamo da postoji jedna pogreška. Pogreška se dogodila na prvom bitu pošto je dobijeni sindrom jednak prvoj koloni kontrolne matrice. Sada pretpostavimo da se pogreška dogodila na drugoj i petoj poziciji. Primljena riječ je: **r**=[0 1 1 0 1 1 1 1] pa je sindrom jednak:

$$\mathbf{S}=\mathbf{c}\mathbf{H}^T=[1\ 1\ 1\ 0]$$

Prvo imamo paran broj jedinica a to znači da iako je sindrom nenulti onda postoji parni broj pogreški ujedno vidimo da sindrom nije jednak nijednoj od kolona kontrolne matrice što dalje ukazuje na činjenicu da postoje pogreške ali da ne možemo da ih ispravimo.

7.6. a) Dati kontrolnu matricu za trougaoni kod (15,10). Prikazati kodiranje ovim kodom poruke 1001011001. U rezultujućem kodu promjeniti jedan bit na karakterističnoj lokaciji i izvršiti dekodiranje. Posmatrati što se dešava sa sindromom u ovom slučaju.
b) Proceduru ponoviti za pravougaoni kod (9,4). Kodiranje obaviti sa porukom 1100.

Rješenje: (a) Kontrolna matrica ovog koda je:

$$\mathbf{H}=\begin{bmatrix}1&1&1&1&0&0&0&0&0&0&1&0&0&0&0\\0&0&0&1&1&1&1&0&0&0&0&1&0&0&0\\0&0&1&0&0&0&1&1&1&0&0&0&1&0&0\\0&1&0&0&0&1&0&0&1&1&0&0&0&1&0\\1&0&0&0&1&0&0&1&0&1&0&0&0&0&1\end{bmatrix}$$

Generatorska matrica ovog koda je

$$\mathbf{G}=\begin{bmatrix}1&0&0&0&0&0&0&0&0&0&1&0&0&0&1\\0&1&0&0&0&0&0&0&0&0&1&0&0&1&0\\0&0&1&0&0&0&0&0&0&0&1&0&1&0&0\\0&0&0&1&0&0&0&0&0&0&1&1&0&0&0\\0&0&0&0&1&0&0&0&0&0&0&1&0&0&1\\0&0&0&0&0&1&0&0&0&0&0&1&0&1&0\\0&0&0&0&0&0&1&0&0&0&0&1&1&0&0\\0&0&0&0&0&0&0&1&0&0&0&0&1&0&1\\0&0&0&0&0&0&0&0&1&0&0&0&1&1&0\\0&0&0&0&0&0&0&0&0&1&0&0&0&1&1\end{bmatrix}$$

Kodna riječ predstavlja proizvod informacione riječi 1001011001 sa kontrolnom matricom. Prvih deset bita kodne riječi je isto kao u informacionoj riječi a kontrolni bitovi su (nastali množenjem kodne riječi sa pet posljednjih kolona generatorske matrice): 0 1 1 0 0. Za ostatak vježbe sami u ovoj riječi izmjenite jedan bit i na osnovu množenja sa kontrolnom matricom izvršite dekodiranje određujući sindrom koji je jednak koloni gdje se pogreška dogodila.

(b) Kontrolna matrica pravougaonog koda je:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Dok je generatorska matrica jednaka:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

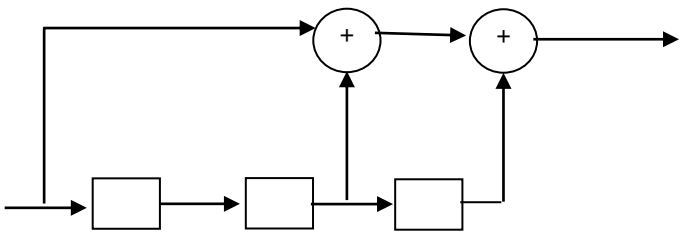
Ostatak zadatka možete uraditi sami.

7.7. Dokazati da je binarni polinom x^3+x^2+1 prost. Upotrijebiti ga za dobijanje odgovarajućeg Hammingovog koda. Kreirati odgovarajuću hardversku strukturu (pravilo za funkcionisanje pomjeračkih registara). Provjeriti rad ako je greška generisana na poziciji x^2.

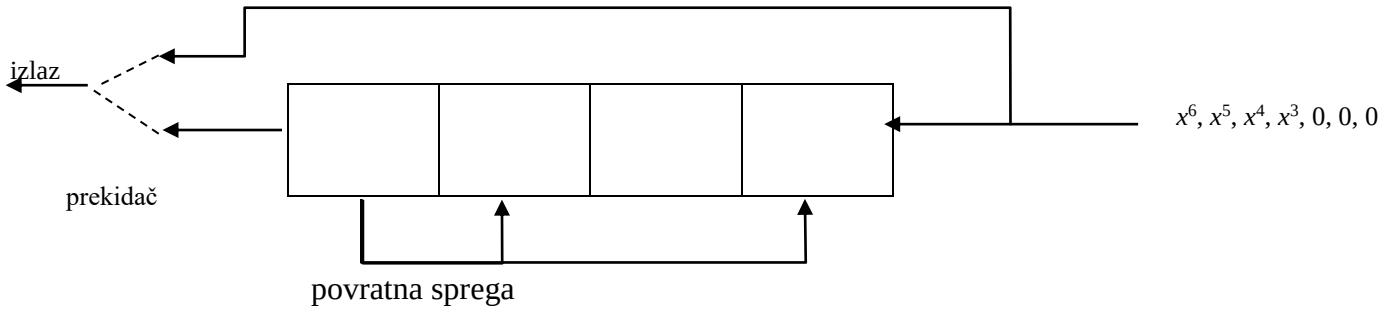
Rješenje: Polinom je prost jer nema djelioca koji ga djele bez ostatka a koji su polinomi manjeg reda. Uzmimo da ga podijelimo sa prostim polinomima nižeg reda:

$x^3+x^2+1 : x = x^2+x$ ostatak 1
$x^3+x^2+1 : x+1 = x^2$ ostatak 1
$x^3+x^2+1 : x^2+x+1 = x$ ostatak $x+1$

Ovo je dovoljno da bi dokazalo da je polinom prost jer kada nije djeljiv sa prostim polinomima nižeg reda nije djeljiv ni sa njihovim umnošcima.
Postoje dvije moguće realizacije jedna bez i druga sa cikličnim registrom i povratnom spregom. Varijanta bez cikličnog registra je:



Dok je varijanta sa cikličnim registrom i povratnom spregom:



Posmatrajmo sada slučaj da je informaciona riječ 1011 odnosno da ako uzmemo da je prvi bit onaj najmanje važnosti imamo polinom $1+x^2+x^3$ pa kada to pomnožimo sa generatorskim polinomom koji je ponovo x^3+x^2+1 dobijamo: x^6+x^4+1 odnosno 1 0 1 0 0 0 1. Greška na x^2 simbolu daje primljenu poruku 1 0 1 0 1 0 1 odnosno $x^6+x^4+x^2+1$. Kako je ostatak pri dijeljenju ovog polinoma sa generatorskim polinomom baš x^2 dolazimo do zaključka da se pogreška dogodila na navedenom bitu.

7.8. Odrediti sve proste binarne polinome 4-reda različite od x^4+x+1 i izvršiti kreiranje Hammingovog koda pomoću njih. Ukoliko ne postoje drugi polinomi četvrtog reda koji zadovoljavaju ovu osobinu proceduru odraditi na kodu koji je kreiran na osnovu polinoma x^4+x+1.

Rješenje: Polinom x^4+1 nije prost jer je djeljiv sa x^2+1 a ovaj opet sa $x+1$, polinom x^4 nije prost jer je djeljiv sa x, polinom x^4+x je između ostalog djeljiv sa x, polinomi x^4+x^2, x^4+x^2+x, x^4+x^3, $x^4+x^3+x^2$, $x^4+x^3+x^2+x$ su svi djeljivi sa x. Polinom x^4+x^2+1 nije prost jer je djeljiv sa x^2+x+1. Dakle ostali su samo polinomi x^4+x+1, x^4+x^3+1 i $x^4+x^3+x^2+x+1$. Sva ova tri polinoma su prosta, međutim samo prva dva od njih su pogodni za kreiranje Hammingovog koda. Razlozi zbog kojih treći polinom nije pogodan nisu sasvim jednostavni za objasniti. Međutim, pokušajmo da formiramo kod na osnovu ovog prostog polinoma pod pretpostavkom da su nule sa najnižim stepenima prostog polinoma $a^0=[0\ 0\ 0\ 1]^T$, $a^1=[0\ 0\ 1\ 0]^T$, $a^2=[0\ 1\ 0\ 0]^T$, $a^3=[1\ 0\ 0\ 0]^T$. Nula stepena a^4 se dobija na osnovu relacije $P(a)=a^4+a^3+a^2+a+1=0$ odnosno $a^4=a^3+a^2+a+1=[1\ \ 1\ \ 1\ \ 1]^T$. Naredni stepen $a^5=a^4a=a^4+a^3+a^2+a=a^3+a^2+a+1+a^3+a^2+a=1$. Dakle, ovo polje se ne zatvara tako da je najniži red za koji se stepen nule prostog polinoma svodi na a^0 je n odnosno ne odgovara dužini kodne riječi koja je projektovana u našem slučaju a^{15}. Stoga je navedeni polinom neupotrebljiv za ovu namjenu ali ćemo vidjeti kasnije da se može koristiti u kombinaciji sa drugim polinomima za kreiranje kodova koji su u stanju da isprave više od jedne pogreške.

7.9. Posmatrajte Hammingov kod za korekciju dvije greške nastao korištenjem polinoma x^4+x+1. Uzmite proizvoljnu binarnu poruku i kodirajte je ovim kodom. Koliko bita mora imati ta binarna poruka. Generišite jednu grešku i ispitajte što ste dobili. Zatim generišite dvije greške i pogledajte kako možete detektovati dobijenu pogrešku.

Rješenje: Kao što smo vidjeli prosti polinom x^4+x+1 daje Hammingov kod (15,11) a ako želimo da izvršimo njegovo proširenje tako da bude u stanju da detektuje dvije pogreške dobijamo kod (16,11). Kontrolna matrica ovog koda je:

$$\mathbf{H}=\begin{bmatrix}1&1&1&1&0&1&0&1&1&0&0&1&0&0&0&0\\0&1&1&1&1&0&1&0&1&1&0&0&1&0&0&0\\0&0&1&1&1&1&0&1&0&1&1&0&0&1&0&0\\1&1&1&0&1&0&1&1&0&0&1&0&0&0&1&0\\1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\end{bmatrix}$$

Binarna poruka kodirana ovim kodom očigledno mora da ima 16 bita (11 informacionih i 5 kontrolnih). Kada se dogodi jedna pogreška imamo da je sindrom jednak koloni matrice međutim u slučaju dvije pogreške npr. na šestom i jedanaestom bitu dobijamo sindrom koji je jednak:

$$\mathbf{S}=[1\ 0\ 1\ 0\ 1]+[0\ 0\ 1\ 1\ 1]=[1\ 0\ 0\ 1\ 0].$$

Kao što se može uočiti ne postoji navedena kolona koja odgovara sindromu. Dakle, u ovom slučaju i na osnovu kontrolne matrice i određivanja sindroma možemo zaključiti da postoji više od jedne pogreške i da je ne možemo dekodirati. Stoga zaključujemo da je greška detektovana ali ne i korigovana.

7.10. Neka je broj bita sa kojima želimo kodirati poruku n=7. Koliko informacija možemo poslati kanalom ako želimo da nam kod ispravi dvije greške? Da li moguća konstrukcija koda koji daje toliko informacija? Ako je moguća prikažite taj kod a ako nije objasniti zbog čega po vašem mišljenju nije?

Rješenje: Ukupan broj riječi koje se mogu konstruisati n=7 bita je 2^7=128. Oko svake legitimne kodne riječi mi sada možemo da formiramo sferu sa riječima koje se od nje razlikuju na jednom odnosno dva bita i nijedan takav skup ne treba da ima presjek. Svaka od sfera ima dakle $1+n+n(n-1)/2$ članova što u našem slučaju iznosi 1+7+21=29. Stoga možemo zaključiti da na osnovu pakovanja sfera možemo da konstruišemo 128/29 odnosno 4 kodne riječi. Postavlja se pitanje da li je ovo stvarno moguće. Pretpostavimo da imamo binarnu kodnu riječ *abcdefg* i da imamo kodnu riječ koja se od nje razlikuje na 5 pozicija (tolika Hammingova distanca je potrebna da bi dobili mogućnost prepoznavanja dvije pogreške) $ab\bar{c}\bar{d}\bar{e}\bar{f}\bar{g}$ gdje smo sa $\bar{x}$ komplement bita (bez gubitka opštosti uzeta su prva dva bita da su ista). Postavlja se pitanje da li postoji kodna riječ sa sedam bita koja se od obije prethodne legitimne kodne riječi razlikuje za 5 mjesta. Najveća moguća razlika na prva dva bita je dva i ona se postiže ako je početak treće kodne riječi $\bar{a}\bar{b}$. Dakle, u preostalih 5 bita potrebno je postići razliku od 3 bita u odnosu na obije riječi što je očigledno nemoguće ako su tri nekomplementirane onda će razlika sa prvom kodnom riječi biti samo dva dok ako su tri komplementirane onda će razlika sa drugom kodnom riječi biti samo dva. Stoga možemo na osnovu ovog prostog eksperimenta da zaključimo da je moguće imati najviše dva kodne riječi za koje važi navedeno pravilo. Stoga je uputno ili skratiti kodnu riječ na pet bita jer nam omogućuje istu fleksibilnost vezanu za broj riječi sa ispravkom dvije pogreške ili prosto se opredijeliti da vršimo ispravku 3 pogreške sa kodnom riječju od sedam bita. Zbog čega je ovo tako? Odgovorićemo na ovo pitanje sa manje matematičkog formalizma ali na prost način. Posmatrajmo kvadrat stranice a, njegova površina je a^2. Uzmimo kvadrat stranice $a\sqrt{2}/2$ čija je površina $a^2/2$. Poređenjem površina bi mogli da zaključimo da možemo smjestiti dva manja u jedan veći kvadrat ali svi znamo da to nije moguće. Ista situacija je i kod kodova gdje zbog geometrije prostora nije uvijek moguće idealno (a ponekad ni približno idealno) spakovati sfere. Očigledno da Hammingova distanca i prateća granica nam ne daje dovoljno informacija o mogućnosti konstrukcije kodova. Stoga smo dali nekoliko detalja o ovom složenom pitanju u Poglavlju XII.

7.11. Odrediti proste binarne polinome petog reda. Koji od njih su pogodni za konstrukciju Hammingovih kodova i na jednom primjeru konstruisati takav kod.

Rješenje. Binarnih polinoma petog stepena (sa nenultim koeficijentom uz x^5) ima 32. Svaki polinom mora da ima nenulti član x^0 jer u suprotnom bi bio djeljiv sa x a ovo svodi broj potencijalnih kandidata za proste polinome na 16. Kao što ćemo kasnije pokazati ako je

$$a_kx^k+a_{k-1}x^{k-1}+a_{k-2}x^{k-2}+\ldots+a_2x^2+a_1x+a_0$$

onda je prost i polinom kojem se koeficijenti mogu zapisati u suprotnom redosljedu:

$$a_0x^k+a_1x^{k-1}+a_2x^{k-2}+\ldots+a_{k-2}x^2+a_{k-1}x+a_k$$

Stoga smo dodatno redukovali prostor za pretraživanje (npr. već smo videli x^4+x^3+1 sa koeficijentima [11001] je prost polinom a onda je prost polinom i onaj koji ima koeficijente [10011] odnosno x^4+x+1).

Polinom je x^5+x^4+1 jer nije prost jer je bez ostatka djeljiv sa x^2+x+1:

$$x^5+x^4+1:x^2+x+1=x^3+x+1$$

Na osnovu prethodnog možemo zaključiti da nije prost ni polinom x^5+x+1. Polinom $x^5+x^4+x^3+x^2+x+1$ je djeljiv sa $x+1$ pa stoga nije prost. Provjerimo sada polinom $x^5+x^4+x^3+1$. Ovaj polinom je djeljiv sa $x+1$:

$$x^5+x^4+x^3+1:x+1=x^4+x^2+x+1$$

Dakle, nije prost ni polinom x^5+x^3+x+1. Sledeći polinom kojega provjeravamo je $x^5+x^4+x^3+x^2+1$. Ovaj polinom nije djeljiv ni sa $x+1$ ni sa x^2+x+1 pa zaključujemo da je prost. Postavlja se pitanje zašto ga ne nastavimo dijeliti sa prostim polinomima višeg reda da bi provjerili ovu činjenicu. Međutim, za ovo nema potrebe da je djeljiv sa polinomom trećeg reda taj bi se polinom pojavio kao količnik u djeljenju sa x^2+x+1 pa stoga nema ni potrebe da se provjerava činjenica djeljivosti sa polinomima trećeg reda kao što ne postoji potreba da se provjerava djeljivost sa polinomima četvrtog reda jer smo implicitno tu djeljivost provjerili dijeljenjem sa $x+1$ a vizuelnom inspekcijom u pogledu djeljivosti sa x. Dakle, zaključujučemo da su polinomi:

$$x^5+x^4+x^3+x^2+1$$
$$x^5+x^3+x^2+x+1$$

prosti. Sledeći polinom koga ćemo provjeriti je $x^5+x^4+x^3+x+1$ koji je takođe prost pa je prost i polinom $x^5+x^4+x^2+x+1$. Spisku prostih polinoma dodajemo:

$$x^5+x^4+x^3+x+1$$
$$x^5+x^4+x^2+x+1$$

Polinom $x^5+x^4+x^2+1$ nije prost jer je djeljiv sa $x+1$ pa nije prost ni polinom x^5+x^3+x+1. Polinom x^5+x^4+x+1 je djeljiv sa $x+1$. Interesantna je činjenica da kada se okrenu koeficijenti ovog polinoma dobijamo ponovo x^5+x^4+x+1. Sa ovim smo završili provjeru svih polinoma koji imaju dva najveća nenulta stepena x^5+x^4 a sada prelazimo na one kojima su najveća dva nenulta stepena x^5+x^3. Međutim, zbog simetrije već smo provjerili i sve one kod kojih je nenulti član uz x pa stoga je jedini polinomi koje treba provjeriti x^5+x^3+1 i x^5+x^2+1 međutim ako su oni međusobno "obrnuti" to je dovoljno provjeriti prvi od njih koji je prost pa je prost i njegova "obrnuta" kombinacija pa stoga zaključujemo da su oba ova polinoma prosti:

$$x^5+x^3+1$$
$$x^5+x^2+1$$

Jedini preostali polinom kojega treba provjeriti je x^5+1 koji je takođe prost polinom. Dakle, došli smo do spiska od 7 prostih polinoma petog reda:

$$x^5+x^4+x^3+x^2+1$$

$$x^5+x^3+x^2+x+1$$
$$x^5+x^4+x^3+x+1$$
$$x^5+x^4+x^2+x+1$$
$$x^5+x^3+1$$
$$x^5+x^2+1$$
$$x^5+1$$

Postavlja se pitanje koji su od ovih prostih polinoma pogodni za formiranje Hammingovog koda (31,26). Da bi kod bio pogodan za formiranje Hammingovog koda potrebno je da $x^{31}=1$ a da svi stepeni nižeg reda x^r za $r<31$ budu različiti od 1. Očigledno je da poslednji prosti polinom nije dobar za generisanje koda (31,26). Ostaje da se provjeri prethodnih šest kombinacija ali imajući na umu da ove kombinacije dolaze u tri para potrebno je izvršiti samo tri povjere.

Posmatrajmo prvi $x^5+x^4+x^3+x^2+1$. Ako je nula prostog polinoma a onda važi:

$$a^5=a^4+a^3+a^2+1$$

Sledeći stepen je $a^6=a^5+a^4+a^3+a=a^2+a+1$, sledeći stepeni su $a^7=a^3+a^2+a$, $a^8=a^4+a^3+a^2$, $a^9=a^2+1$, $a^{10}=a^3+a$, $a^{11}=a^4+a^2$, $a^{12}=a^4+a^2+1$, $a^{13}=a^4+a^2+a+1$, $a^{14}=a^4+a+1$, $a^{15}=a^4+a^3+a+1$, $a^{16}=a^3+a+1$, $a^{17}=a^4+a^2+a$, $a^{18}=a^4+1$, $a^{19}=a^4+a^3+a^2+a+1$, $a^{20}=a+1$, $a^{21}=a^2+a$, $a^{22}=a^3+a^2$, $a^{23}=a^4+a^3$, $a^{24}=a^3+a^2+1$, $a^{25}=a^4+a^3+a$, $a^{26}=a^3+1$, $a^{27}=a^4+a$, $a^{28}=a^4+a^3+1$, $a^{29}=a^3+a^2+a+1$, $a^{30}=a^4+a^3+a^2+a$, $a^{31}=1$.

Slično se može provjeriti da važi i za preostala dva para odnosno zaključujemo da postoji ukupno 6 pogodnih prostih polinoma za formiranje Hammingovog polinoma (31,26).

Poglavlje VIII
BCH KOD

8.1. Polja

Vidjeli smo da konstrukcija kodova postaje veoma složena ako se pred kod postavljaju relativno komplikovani zadaci (npr. ispravljanje većeg broja pogreški). Intuitivne predstave tu više ne mogu pomoći i stoga se mora kreirati metodologija za konstrukciju ovakvih kodova. U tu svrhu poslužila je algebarska kodna teorija čije ćemo osnovne elemente sada objasniti. Osnovni pojam u algebarskoj kodnoj teoriji je polje. Polje pripada grupi algebarskih struktura koje se definišu nad skupom P i operacijama koje se uslovno zovu sabiranje i množenje. Kod nas će sabiranje biti ex-ili operacija. Kod nebinarnih sistema umjesto ex-ili operacije također se koristi sabiranje po modulu. Množenje je za binarne alfabete logička i operacija, dok je za nebinarne množenje po modulu. Kada se bude vršilo množenje polinoma njihovi stepeni će biti sabirani po stepenu najvećeg dopuštenog reda polinoma u datom polju, dok će sabiranje koeficijenata polinoma biti obavljano kao ex-ili operacija. Sve algebarske strukture zadovoljavaju sledeće osobine:

P_1:	za svako x, y, z važi u skupu P	$x+(y+z)=(x+y)+z$
P_2:	postoji nulti član	$x+0=0+x=x$
P_3:	postoji negativni element	$x+(-x)=0$
P_4:	za svako x,y u skupu P	$x+y=y+x$
P_5:	za svako x, y, z u skupu P	$x\,(yz)=(xy)\,z$
P_6:	postoji jedinični element	$x\,1=1\,x=x$
P_7:	za svako x različito od 0 postoji x^{-1}	$x^{-1}\,x=x\,x^{-1}=1$
P_8:	za svako x,y	$x\,y=y\,x$
P_9:	za svako x, y, z	$x\,(y+z)=x\,y+x\,z$

Nama je polje već dobro poznato zato što je skup realnih brojeva sa operacijama sabiranjam i množenja polje. Obično smo mnogo familijarniji sa poljem nego sa algebarskim strukturama kao što su prsten, potprsten, integralni domen itd. koje ne zadovoljavaju neke od gorenavedenih osobina. Konačno polje se može definisati na nekom podskupu skupa realnih brojeva. Npr. ako je skup nad kojim je polje definisano {0,1} operacije ex-ili i logičko množenje zadovoljavaju sve karakteristike da bi se nešto zvalo poljem.

Osnovni elementi za izgradnju kodova su vektorski prostori definisani nad poljima F_p , gdje skup P čine brojevi $\{0,1,2,...,p-1\}$, gdje je p neki prost broj. Aritmetika u ovakvom polju se obavlja po modulu p. Npr. neka je p=7 i neka je x=2 i y=4. Proizvod x i y je 8 odnosno po modulu 7 (ostatak pri dijeljenju sa 7) je 1. Odavde vidimo da su ovo međusobno inverzni brojevi. Zbir ova dva broja je 6. Za brojeve x=6 i y=3 proizvod po modulu 7 je 4 dok je zbir po modulu 7 jednak 2, dok je za x=6 i y=1 proizvod 6 (1 je jedinični elemenat) dok je zbir 0 (1 je negacija od 6). Provjerite da li ovakav skup sa uvedenim operacijama sabiranja i množenja po modulu zadovoljava osobine polja.

Vektorski prostor definisan nad poljem F_p, $V_m(F_p)$, je skup uređenih nizova $\mathbf{a}=\{a_0,a_1,...,a_{m-1}\}$ gdje je svaki član ovog niza element konačnog polja. Očigledno je da članova vektorskog prostora definisanog na ovaj način ima p^m.

Svaki vektorski prostor $V_m(F_p)$ se može učiniti poljem na sledeći način. Uzmimo polinom: $f(x)=f_0+f_1x+...+f_mx^m$ čiji su koeficijenti iz skupa F_p. Pretpostavka je da se ovaj polinom ne može dalje uprostiti. Sada uzmimo dva vektora iz vektorskog prostora $V_m(F_p)$ koji su definisani kao: $\mathbf{a}=\{a_0,a_1,...,a_{m-1}\}$ i $\mathbf{b}=\{b_0,b_1,...,b_{m-1}\}$. Proizvod ovih polinoma u vektorskom prostoru se definiše kao:

$$[a_0 + a_1 x + ... + a_{m-1} x^{m-1}][b_0 + b_1 x + ... + b_{m-1} x^{m-1}] = c_0 + c_1 x + ... + c_{m-1} x^{m-1} \pmod{f(x)}$$

Prethodna notacija znači pomnožiti polinome, a zatim sračunati rezultat kao ostatak dijeljenja sa prostim polinomom $f(x)$. Sabiranje koeficijenata uz članove istog reda polinoma se obavlja po modulu p. Postoji međutim više definicija množenja polinoma. Najčešće se uzima ostatak pri dijeljenju sa prostim polinomom $f(x)$. Ovakva polja se nazivaju Galoaovim i označavaju sa $GF(p^m)$ ili F_q gdje je $q=p^m$. Uočite da sada polje F_q može biti takvo da q nije prost broj, ali se može definisati samo za kombinacije p^m (ne može se definisati GF sa 6 elemenata).

8.2. Multiplikativne strukture

Podsjetimo se još jednog pojma iz linearne algebre. To je pojam grupe. Grupa se definiše nad nekim skupom G i nad operacijom koju uslovno zovemo množenje. Ova operacija zadovoljava osobine P_5, P_6 i P_7 od gore definisanih za polje. Poseban tip grupe je ciklična grupa kod koje postoji element a takav da se svaki nenulti element grupe G može prikazati kao $a^n=a\cdot a\cdot ...\cdot a$ (množenje n puta). Nenulti elementi u polju F_q formiraju cikličnu grupu F_q^* koja ima ukupno q-1 elemenata. Ovo praktično znači da u polju F_q postoji neki nenulti element a, na osnovu kojeg se može dobiti kompletna grupa. Još preciznije svaki element ovog polja zadovoljava osobinu da se na osnovu njegovih stepena može dobiti kompletna grupa. Recimo za polje sa elementima {0, 1, 2, 3, 4, 5, 6} gdje su operacije sabiranja i množenja definisane po modulu 7, stepeni elementa a=3 daju sve nenulte elemente. Od posebnog značaja su takvi elementi (generatori polja) koji se mogu koristiti kod vektorskih polja. Već smo vidjeli da se Hammingov polinom može prikazati kao struktura gdje pojedine pozicije u informacionom polinomu odgovaraju stepenima nule prostog polinoma koji je usvojen kao generatorski polinom. Već smo kod Hamminogovog koda vidjeli i da sve ispravne kodne riječi možemo dobiti kao umnoške polinoma generatora koda što implicira činjenicu da je Hammingov kod cikličan.

8.3. Bose-Chaudhuri-Hocquenghem kodovi

Bose-Chaudhuri-Hocquenghem (BCH) kodovi su bili prvi sistematski kodovi koji su vodili ka mogućnosti ispravljanja više pogreški. Oni su samo algebarskom kodnom teorijom potkrijepljeni (proširenи) Hammingovi kodovi. Mi ćemo kad god je to moguće pokušati da izbjegnemo teške teorijske osnove ovih kodova i da što je više moguće izvršimo njihovo uvođenje poluintuitivnim metodama. Na nekoliko mjesta u našim algoritmima ćemo morati praviti iskorake u ovoj materiji da bi se koliko je to u najmanjoj mjeri potrebno držali striktnih matematičkih pravila. Kontrolna matrica Hammingova koda dužine $n=2^m-1$ može se prikazati u obliku:

$$\mathbf{H} = [\mathbf{h}_0, \mathbf{h}_1, ..., \mathbf{h}_{n-1}]$$

gdje $\mathbf{h_i}$ su ovo vektori dužine 2^m-1 iz vektorskog prostora $V_m=V_m(F_2)$ poređani po nekom redosljedu: Dimenzije matrice $\mathbf{H}$ su mxn, što znači da kod ima m paritetnih bita koji omogućavaju da kod ispravi jednu pogrešku. Za ispravljanje dvostrukih pogreški očigledno nam treba m dodatnih kontrolnih bita. Novu kontrolnu matricu možemo prikazati kao:

$$\mathbf{H}_2 = \begin{bmatrix} \mathbf{h}_0 & \mathbf{h}_1 & ... & \mathbf{h}_{n-1} \\ \mathbf{w}_0 & \mathbf{w}_1 & ... & \mathbf{w}_{n-1} \end{bmatrix}$$

$$[\mathbf{w}_0, \mathbf{w}_1, ..., \mathbf{w}_{n-1}] \in V_m$$

Sada treba da odredimo preslikavanje $\mathbf{h}_i \rightarrow \mathbf{w}_i$ kao funkciju vektorskog prostora V_m od njega samoga:

$$\mathbf{H}_2 = \begin{bmatrix} \mathbf{h}_0 & \mathbf{h}_1 & ... & \mathbf{h}_{n-1} \\ \mathbf{f}(\mathbf{w}_0) & \mathbf{f}(\mathbf{w}_1) & ... & \mathbf{f}(\mathbf{w}_{n-1}) \end{bmatrix}$$

Očigledno da bi $\mathbf{H}_2$ odredio kod sa mogućnošću ispravke pogreške, onda on može da odredi sve sindrome sa $1+n+n(n-1)/2$ oblika pogreške sa težinama 0, 1 i 2. Svaki takav sindrom je zbir vektora u $\mathbf{H}_2$. Sada možemo podijeliti sindrom na dva dijela: $\mathbf{S}_1 = [\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_m]$ i $\mathbf{S}_2 = [\mathbf{s}_{m+1}, \mathbf{s}_{m+2}, ..., \mathbf{s}_{2m}]$ oba u vektorskom prostoru V_m. Sindrom koji označava da nema pogreške je oblika $\mathbf{S} = [\mathbf{0}, \mathbf{0}]$. Jednostruke pogreške imaju oblik: $S = [h_i, f(h_i)]$ dok dvostruke imaju oblik $\mathbf{S} = [\mathbf{h}_i + \mathbf{h}_j, \mathbf{f}(\mathbf{h}_i) + \mathbf{f}(\mathbf{h}_j)]$. Očigledno mora da važi $\mathbf{f}(\mathbf{0})=\mathbf{0}$. Uslov da svi sindromi mogu biti određeni su zadati zahtjevom da sledeći sistem jednačina ima najmanje jedno rješenje ($\mathbf{u}$,$\mathbf{v}$) za svaki par vektora iz V_m.

$\mathbf{u}+\mathbf{v}=\mathbf{S}_1$
$\mathbf{f}(\mathbf{u})+\mathbf{f}(\mathbf{v})=\mathbf{S}_2$

Nakon ovoga moramo naći funkciju $\mathbf{f}: V_m \to V_m, \mathbf{f}(\mathbf{0}) = \mathbf{0}$ sa datim svojstvom. Možemo pokušati sa linearnom transformacijom: $\mathbf{f}(\mathbf{h})=\mathbf{T}\mathbf{h}$ gdje $\mathbf{T}$ označava operator linearne transformacije, ali nećemo moći doći do rješenja. Dakle, funkcija $\mathbf{f}$ mora biti nelinearna. Da bi opisali nelinearne funkcije od vektora $\mathbf{h} \in V_m$ treba poći od činjenice da je moguće definisati množenje vektora iz vektorskog prostora V_m koji kombinovano sa sabiranjem kreira polje prostora V_m. Posmatrajmo Galoaovo polje $GF(2^m) = F_{2^m}$. Svaka funkcija $\mathbf{f}: V_m \to V_m$ se može opisati polinomom. Polinomi reda manjeg ili jednakog od 2 nijesu dobre funkcije za ovu namjenu, dok polinomi trećeg i višeg neparnog stepena odgovaraju namjeni. Ako je $[\alpha_0, \alpha_1, ..., \alpha_{n-1}]$ bilo koji poredak elemenata iz F_{2^m}, tada se ova matrica može zapisati kao:

$$\mathbf{H}_3 = \begin{bmatrix} \alpha_0 & \alpha_1 & ... & \alpha_{n-1} \\ \alpha_0^3 & \alpha_1^3 & ... & \alpha_{n-1}^3 \end{bmatrix}$$

gdje je ovo kontrolna matrica koda dužine $n=2^m-1$ koji ispravlja dvije greške. Pošto matrica iznad F_2, $\mathbf{H}_2$ ima $2m$ redova, dimenzija koda je veća ili jednaka $n-2m=2^m-1-2m$. Ako želimo da ispravimo bilo koju pogrešku čija je težina manja ili jedna od w matrica za detekciju grešaka se može zapisati kao:

$$\mathbf{H} = \begin{bmatrix} \alpha_0 & \alpha_1 & ... & \alpha_{n-1} \\ \alpha_0^3 & \alpha_1^3 & ... & \alpha_{n-1}^3 \\ \alpha_0^5 & \alpha_1^5 & ... & \alpha_{n-1}^5 \\ ... & ... & ... & ... \\ \alpha_0^{2w-1} & \alpha_1^{2w-1} & ... & \alpha_{n-1}^{2w-1} \end{bmatrix}$$

Dakle, došli smo do kontrolne matrice koju smo poluintuitivno uveli na jednom od prethodnih časova. Vidjeli smo već da ako imamo kod sa 15 bita, može se konstruisati na osnovu prostog polinoma x^4+x+1. Izvršimo konstrukciju matrice koja može da izvrši kontrolu pojave tri pogreške. Napomenimo da smo već vidjeli da nam za slučaj da želimo da ispravimo dvije pogreške treba 8 bita za to, a da ostaje samo 7 informacionih. Kakva je situacija u slučaju ispravljanja 3 pogreške?

Ponovimo postupak koji smo vršili za slučaj korekcije dvije pogreške i dodajmo još 4 kolone koje predstavljaju pete stepene elemenata iz prve četiri kolone matrice. Jednostavno se uočava da je deveta kolona nula kolona, te da je parnost po ovoj koloni uvijek trivijalno zadovoljena, pa to dalje znači da se deveta kolona može bez gubitka izostaviti (u stvari ovo je dobitak, jer nam oslobađa prostor za jedan informacioni bit na račun bita za provjeru parnosti). Još važnije je to da su deseta i

jedanaesta kolona iste, tako da vrše istu provjeru parnosti, odnosno redundantne su. Stoga se jedna od ove dvije kolone može izostaviti tako da kod koji ispravlja tri pogreške sa 15 bita ima "samo" deset provjera parnosti. Očigledno analiza koju provodimo nije nimalo prijatna i u nekim drugim situacijama pronalaženje linearne zavisnosti kolona koje dodajemo i postojećih kolona može biti znatno složenije. Da bi situacija bila još gora, matrice postoju sve veće i veće sa rastućim memorijskim zahtjevima, a ni procedura dekodiranje nije nimalo jednostavna. Stoga jedini "spas" je u korišćenju kodova zapisanih preko odgovarajućih polinoma.

$$\left[\begin{array}{cccccccccccccc}
1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
\hline
1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1
\end{array}\right]$$

Ono što mi želimo postići je formiranje polinoma $g(x)$ koji će pomnožen sa informacionim polinomom $i(x)$ dati kodiranu informaciju $c(x)$. U primjeru koji je ovdje opisan nije teško zaključiti da ako želimo da imamo kodni polinom sa petnaest elemenata (polinom $c(x)$ 14-tog stepena) a broj mjesta parnosti je 10, to znači da će polinom $g(x)$ biti 10-tog reda, pa će informacioni polinom biti 4-tog reda (odnosno omogućiće 5 informacionih bita). Prva četiri reda ove matrice su formirana na osnovu prostog polinoma $p(x)=x^4+x+1$. Može da se pokaže da se rezultujući polinom koji će kreirati kod sa kontrolnom matricom koja je prethodno opisana može dobiti kao proizvod ovog prostog polinoma sa polinomima koji stvaraju naredna 4 odnosno naredna 2 reda matrice:

$$g(x)=p(x)g_3(x)g_5(x)$$

U opštem slučaju broj polinoma u ovom proizvodu će biti jednak broju pogreški koje želimo korigovati. Postavlja se pitanje kako možemo na osnovu oblika $p(x)$ i eventualnih dodatnih informacija dobiti polinome $g_3(x)$ i $g_5(x)$. Podsjetimo se da mi pretpostavljamo da se polinom $p(x)$ anulira za neki set rješenja α_i gdje broj rješenja ne može da pređe stepen tog polinoma $p(\alpha_i)=0$. Očigledno da ako je α_i korjen od $p(x)$ da zbog toga što su kolone matrice koja u matričnom zapisu kontrolne matrice predstavlja partnera vektora $g_3(x)$ su uzete kao treći stepeni od α važi: $g_3(x)=p(x^3)$ odnosno $g_5(x)=p(x^5)$. U našem primjeru se lako može zaključiti da je $g_3(x)$ polinom 4-tog reda a da je $g_5(x)$ polinom 2-og reda (jer predstavljaju toliko kolona kontrolne matrice). Kako mi ne konstruišemo kodove na osnovu matrice već na osnovu polinoma to se postavlja pitanje da li postoji neka alternativna metodologija za određivanje stepena ovih polinoma. Ne ulazeći, za sada, u proces dokazivanja (o ovome će više detalja biti kasnije kada uvedeno neke neophodne činjenice) stepen ovih polinoma se određuje na osnovu tzv. konjugata. Konjugati su stepeni reda 2^n stepena korjena prostog polinoma, koji se koriste u konstrukciji koda. U našem slučaju ti stepeni su: $[\alpha,\alpha^3,\alpha^5]$. Konjugati se ne ponavljaju odnosno uzima se samo broj različitih vrijednosti konjugata. Pretpostavimo da želimo da konstruišemo BCH kod BCH(15,3) gdje oznake kod BCH koda znače broj bita u kodnoj riječi (dužina kodne riječi) dok je drugi argument broj grešaka koje kod ispravlja. Važno je zapamtiti da za polje iz primjera važi $\alpha^{15}=1$. Dakle, konjugati dvojke su: $[\alpha,\alpha^2,\alpha^4,\alpha^8]$. Sledeći stepen je $(\alpha^8)^2=\alpha^{16}=\alpha^{15+1}=\alpha$ odnosno već se ponavlja u spisku konjugata stepena korjena α.

Konjugati α^3 su: [α^3,α^6,α^{12}, α^{24}= α^9] jer je naredni član α^{18}= α^3. Na kraju konjugati stepena α^5 su: [α^5,α^{10}] jer je α^{20}= α^5. Na osnovu broja konjugata u konkretnom slučaju se može zaključiti da je za korekciju jedne greške potreban polinom 4-tog reda, za narednu grešku ponovo polinom 4-tog reda, dok je za treću grešku dovoljan polinom drugog reda. Uočite da kada smo koristili konjugate nijesmo morali da imamo odgovarajuću tabelu što je relativno mala ali ne i beznačajna ušteda. Koeficijenti polinoma $g_3(x)$ i $g_5(x)$ se određuju tako što se formiraju jednačine: $g_3(x^3)=0$ i $g_5(x^5)=0$ i odrede koeficijenti polinoma koji za dato polje to obezbjeđuju.

$$g_3(x) = g_{30} + g_{31}x + g_{32}x^2 + g_{33}x^3 + g_{34}x^4$$

Za $x=\alpha^3$ važi da je $g_3(x)=0$. Ovo se svodi na (imajući u vidu čemu su jednaki odgovarajući stepeni):

$$g_{30}[0001] + g_{31}[1000] + g_{32}[1100] + g_{33}[1010] + g_{34}[1111] = [0000]$$
$$\alpha^0 \qquad \alpha^3 \qquad \alpha^6 \qquad \alpha^9 \qquad \alpha^{12}$$

Jedino netrivijalno rješenje (rješenje sa svim koeficijentima jednakim nula se naziva trivijalnim i očigledno nije pogodno za konstrukciju polinoma) ovog sistema od četiri jednačine sa pet nepoznatih je:

$$[g_{30}, g_{31}, g_{32}, g_{33}, g_{34}] = [11111]$$

pa slijedi: $g_3(x) = 1 + x + x^2 + x^3 + x^4$. Na sličan način se dobija: $g_5(x) = 1 + x + x^2$. Dalje dobijamo generatorski polinom u obliku:

$$g(x) = (x^4 + x + 1)(1 + x + x^2 + x^3 + x^4)(1 + x + x^2) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$$

Treba napomenuti da je ovo jedan mogući generatorski polinom, a da eventualna ostala rješenja zavise od polaznog prostog polinoma. Sada ćemo pokazati algoritam za kodiranje koji omogućuje jednostavnu realizaciju BCH koda.

Postavlja se pitanje zašto konjugati na predmetni način determinišu dimenzije pojedinih polinoma u kodnoj riječi. Premda navedeno tumačenje izlazi van okvira našeg osnovnog kursa u ovoj oblasti neki detalji će biti dati nešto kasnije.

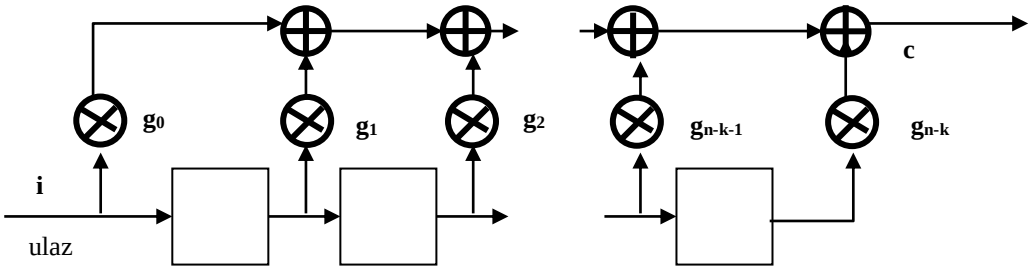
8.4. Realizacija BCH koda pomoću pomjeračkog registra

Realizacija BCH kodova se ni po čemu ne razlikuje od realizacije Hamingovog koda putem pomjeračkog registra. Algoritam za kodiranje bilo kojeg (n,k) linearnog koda je pravilo preslikavanja 2^k informacionih nizova $\mathbf{i} = [i_0, i_1, ..., i_{k-1}]$ u skupu 2^k kodnih riječi $\mathbf{c} = [c_0, c_1, ..., c_{n-1}]$. Kod BCH($n$,$w$) koda treba pomnožiti polinom $i(x) = i_0 + i_1 x + ... + i_{k-1}x^{k-1}$ sa generatorskim polinomom $g(x) = g_0 + g_1 x + ... + g_{n-k}x^{n-k}$ pa se dobija kao rezultat $c(x) = c_0 + c_1 x + ... + c_{n-1}x_{n-1}$ odnosno vektor $\mathbf{c} = [c_0, c_1, ..., c_{n-1}]$ pa je kodna riječ koja odgovara vektor **i**:

$$c(x) = i(x)g(x) = (i_0 + i_1 x + ... + i_{k-1}x^{k-1})(g_0 + g_1 x + ... + g_{n-k}x^{n-k}) = c_0 + c_1 x + ... + c_{n-1}x^{n-1}$$

Ovo se može prikazati preko sistema sa pomjeračkim registrom:

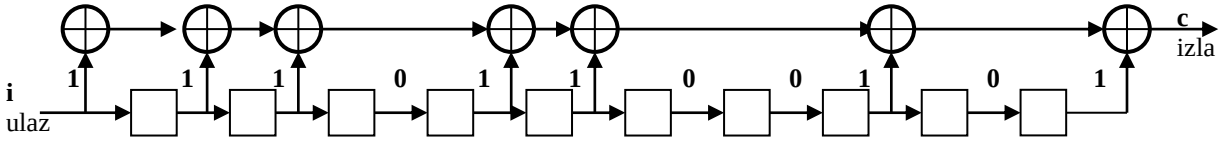
izlaz



Vrijeme	ULAZ	REGISTAR	IZLAZ
0	$\mathbf{i}_0$	[00...0]	$\mathbf{i}_0\mathbf{g}_0$
1	$\mathbf{i}_1$	[$\mathbf{i}_0$00...0]	$\mathbf{i}_1\mathbf{g}_0+\mathbf{i}_0\mathbf{g}_1$
2	$\mathbf{i}_2$	[$\mathbf{i}_1\mathbf{i}_0$0...0]	$\mathbf{i}_2\mathbf{g}_0+\mathbf{i}_1\mathbf{g}_1+\mathbf{i}_0\mathbf{g}_2$
...	...	...	...
j	$\mathbf{i}_j$	[$\mathbf{i}_j\mathbf{i}_1\mathbf{i}_0$0...0]	$\mathbf{i}_j\mathbf{g}_0+\mathbf{i}_{j-1}\mathbf{g}_1+...+\mathbf{i}_0\mathbf{g}_j$
...	...	...	...
n-1	0	[0,...,0,$\mathbf{i}_{k-1}$]	$\mathbf{i}_{k-1}\mathbf{g}_{n-k}$

$$c_0 = i_0 g_0$$
$$c_1 = i_0 g_1 + i_1 g_0$$
$$c_2 = i_0 g_2 + i_1 g_1 + i_2 g_0$$
...
$$c_j = i_0 g_j + i_1 g_{j-1} + ... + i_j g_0$$
...
$$c_{n-1} = i_{k-1} g_{n-k}$$

Realizacija BCH(15,3) koda sa generatorskim polinomom:



Kodni polinom je:

$$c(x) = (i_0 + i_1 x + i_2 x^2 + i_3 x^3 + i_4 x^4)(x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1) =$$
$$= i_0 + (i_0 + i_1)x + (i_0 + i_1 + i_2)x^2 + (i_1 + i_2 + i_3)x^3 + (i_0 + i_2 + i_3 + i_4)x^4 +$$
$$+ (i_0 + i_1 + i_3 + i_4)x^5 + (i_1 + i_2 + i_4)x^6 + (i_2 + i_3)x^7 + (i_0 + i_3 + i_4)x^8 + (i_1 + i_4)x^9 +$$
$$+ (i_0 + i_2)x^{10} + (i_1 + i_3)x^{11} + (i_2 + i_4)x^{12} + i_3 x^{13} + i_4 x^{14}$$

$$c_0 = i_0, c_1 = i_0 + i_1,\ c_2 = i_0 + i_1 + i_2, c_3 = i_1 + i_2 + i_3, c_4 = i_0 + i_2 + i_3 + i_4$$
$$c_5 = i_0 + i_1 + i_3 + i_4, c_6 = i_1 + i_2 + i_4, c_7 = i_2 + i_3, c_8 = i_0 + i_3 + i_4, c_9 = i_1 + i_4$$
$$c_{10} = i_0 + i_2, c_{11} = i_1 + i_3, c_{12} = i_2 + i_4, c_{13} = i_3, c_{14} = i_4$$

Za informacioni vektor $\mathbf{i} = [10110]$ kodna riječ je $\mathbf{c} = [1100101000001110]$. Ako smo primili poruku bez greške na relativno jednostavan način možemo obaviti dekodiranje. Naime, $i_0=c_0$, $i_3=c_{13}$ i $i_4=c_{14}$. Ostali informacioni biti se ne pojavljuju direktno u kodnoj riječi, pa ih moramo na neki način odrediti. Npr. $i_1=c_0+c_1$, odnosno $i_2=c_{12}+c_{14}$, ali se naravno ovim ne iscrpljuje broj mogućnosti. Ipak situacija kada nema grešaka nije i ono za što je BCH kod projektovan pa ćemo dekodiranje pomoću BCH koda u uslovima pojave grešaka biti razmatrana u daljem tekstu.

8.5. Dekodiranje BCH koda – Teorijski elementi

BCH kodiranu poruku možemo prikazati u obliku polinoma:

$$c(x) = c_0 + c_1 x + ... + c_{n-1} x^{n-1}$$

Ovaj polinom predstavlja proizvod informacionog polinoma sa prostim polinomom $p(x)$ i sa polinomima $g_{2j-1}(x)=p(x^{2j-1})$, $j=1,3,...,2w-1$. Ako znamo da je α korjen polinoma $p(x)$, onda bi to α trebalo da bude i korjen polinoma $c(x)$ (ako nema grešaka u sistemu) odnosno α^3, α^5, ..., α^{2w-1} trebali bi da budu korjeni polinoma $c(x)$, jer su to korjeni polinoma $g_{2j-1}(x)$. Uočimo dalje da različiti stepeni α, od nultog do $n-1$ stepena čine kolone osnovne kontrolne matrice koda, koja je kreirana na osnovu polinoma $p(x)$. Redosljed kolona ne utiče na tip koda, pa možemo da zapišemo da važi:

$$\sum_{i=0}^{n-1} c_i \alpha_i^j = 0, \quad j = 1,3,...,2w-1 \text{ (ili } \quad j = 1,2,...,2w)$$

gdje je $[\alpha_0, \alpha_1, ..., \alpha_{n-1}]$ bilo koji fiksni poredak nenultih elemenata iz polja F_{2^m}. Ako ova osobina važi naš kod je BCH (n,w). Ovo se može shvatiti i kao alternativna definicija BCH koda. Važna osobina BCH koda je i da je ovo ciklični kod odnosno da ako je $[c_0, c_1, ..., c_{n-1}]$ kodna riječ onda je svaki ciklični pomjeraj ove kodne riječi kodna riječ BCH koda. Uočimo da smo u zagradi prethodnog uslova napisali da važi ne samo za neparne stepene, već i za parne! Kako je to moguće? Ovo je posljedica jedne veoma interesante osobine kod polinoma, a to je:

$$c^2(x)=c(x^2)$$

Dokaz nije i suviše složen. Ako je

$$c(x) = c_0 + c_1 x + ... + c_{n-1} x^{n-1} = \sum_{i=0}^{n-1} c_i x^i$$

Kvadrat ovog polinoma se može zapisati kao:

$$c^2(x) = \left(\sum_{i=0}^{n-1} c_i x^i \right)^2 = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} c_i x^i c_j x^j = \sum_{i=0}^{n-1} c_i^2 x^{2i} + 2 \sum_{i=0}^{n-1} \sum_{\substack{j=0 \\ i \neq j}}^{n-1} c_i x^i c_j x^j$$

Iz dvostruke sume smo izdvojili podsumu za koju važi $i=j$. Drugi izraz je nula jer treba imati na umu da je u pitanju sabiranje po modulu 2, a to znači da bilo koji binarni broj sabran sa samim sobom daje nulu. Prva suma u gornjem izrazu je zapravo polazni polinom, ali kada se umjesto x uvrsti x^2. Svaka nula polinoma $c(x)$ je nula i njegovog kvadratnog stepena, pa to dalje znači da prethodno uvedena jednakost važi za sve parne stepene, a ne samo za neparne.

Sada možemo objasniti zbog čega nam konjugati pomažu u određivanju reda polinoma koji čine generatorski polinom. Prosti polinom koji je osnova za formiranje polinoma ima nulu $x=\alpha$. Sada smo vidjeli da je $x=\alpha^2$ zasigurno nula ovog polinoma pa je sledeća nula ovog polinoma α^4, pa $x=\alpha^8$ pa zatim $x=\alpha^{16}=\alpha^1$ a sada vidimo da se stepeni nule ovog polinoma ponašaju kao konjugati pa su nule polinoma $\{\alpha, \alpha^2, \alpha^4, \alpha^8\}$ a to dalje znači još da se izabrani prosti polinom kojega koristimo za formiranje generatorskog polinoma može zapisati kao $(x+\alpha)(x+\alpha^2)(x+\alpha^4)(x+\alpha^8)$. Za vježbu provjerite da li ovaj stav važi za naše generatorske polinome. Kao što smo rekli nule polinoma koji predstavlja drugi dio kontrolne matrice našeg polinoma je zasigurno $x=\alpha^3$. Po istom principu nula mora biti kvadrat ove nule $x=\alpha^6$, pa zatim njen kvadrat $x=\alpha^{12}$, pa njen kvadrat $x=\alpha^{24}=\alpha^9$ dok se sledeće nule ponavljaju pa zaključujemo da se polinom trećeg reda može zapisati kao $(x+\alpha^3)(x+\alpha^6)(x+\alpha^{12})(x+\alpha^9)$ odnosno za naš kod dovoljno je da konstatujemo (i zapravo

jednostavnije je) da je u pitanju polinom četvrtog reda. Na isti način vođeni navedenom logikom možemo odrediti i stepen narednog polinoma. Konačno svo vrijeme smo mogli da izbjegnemo rad sa polinomima ako koristimo navedenu tehniku zasnovanu na konjugatima.

Nastavljamo sa određivanjem sindromskog polinoma koji će nam biti osnova u dekodiranju. Ovo se može zapisati kao:

$$\sum_{i=0}^{n-1} c_i \alpha_i^j = 0, \quad j = 1, 2, ..., 2w$$

Umjesto usvojenog redosljeda vektora mi možemo da uvedemo bilo koji proizvoljan poredak vektora različitih nenultih vektora α_i i da ne izgubimo osobine BCH koda. Dakle umjesto $[\alpha_0, \alpha_1, ..., \alpha_{n-1}]$ proizvoljnog poretka $n=2^m-1$ nenultih elemenata iz polja F_{2^m} uvedimo poredak $[\alpha_0^{-1}, \alpha_0^{-1}, ..., \alpha_{n-1}^{-1}]$ pa se dobija kodni uslov:

$$\sum_{i=0}^{n-1} c_i \alpha_i^{-j} = 0, \quad j = 1, 2, ..., 2w$$

Ovaj uslov je uveden samo radi jednostavnijeg izvođenja algoritma za dekodiranje koda, a tumačićemo ga kad budemo povezivali sindromski polinom sa pozicijama na kojima se dogodila greška u kodnoj riječi. Pretpostavimo da smo za poslatu kodnu riječ $\mathbf{c} = [c_0, c_1, ..., c_{n-1}]$ dobili kodnu riječ $\mathbf{r} = [r_0, r_1, ..., r_{n-1}]$ koja se na ne više od w mjesta razlikuje od kodne riječi **c**. Tada, vrijednosti:

$$S_j = \sum_{i=0}^{n-1} r_i \alpha_i^{-j}, \quad j = 1, 2, ..., 2w$$

su različite od nule. Ove vrijednosti igraju ulogu sindroma kod dekodiranja kodova na osnovu kontrolne matrice. Ovi koeficijenti mogu da formiraju vektor $S(x)=S_1+S_2x+S_3x^2+...+S_{2w}x^{2w-1}$ koji se naziva sindromski polinom i koji se može zapisati i kao:

$$S(x) = \sum_{j=1}^{2w} S_j x^{j-1} = \sum_{j=1}^{2w} x^{j-1} \sum_{i=0}^{n-1} r_i \alpha_i^{-j} = \sum_{i=0}^{n-1} r_i \sum_{j=1}^{2w} x^{j-1} \alpha_i^{-j} = \sum_{i=0}^{n-1} r_i \alpha_i^{-1} \frac{1 - x^{2w} \alpha_i^{-2w}}{1 - x\alpha_i^{-1}} = \sum_{i=0}^{n-1} r_i \frac{x^{2w} \alpha_i^{-2w} - 1}{x - \alpha_i}$$

Sada možemo zapisati $r_i=c_i+e_i$ gdje je e_i vektor pogreške koji uzima vrijednost 1 na pozicijama gdje je nastala greška u sistemu i na ostalim pozicijama uzima vrijednost 0. Kada uvrstimo ovu relaciju u izraz za sindromski polinom dobijamo:

$$S(x) = \sum_{i=0}^{n-1} r_i \frac{x^{2w} \alpha_i^{-2w} - 1}{x - \alpha_i} = \sum_{i=0}^{n-1} c_i \frac{x^{2w} \alpha_i^{-2w} - 1}{x - \alpha_i} + \sum_{i=0}^{n-1} e_i \frac{x^{2w} \alpha_i^{-2w} - 1}{x - \alpha_i} = \sum_{i=0}^{n-1} e_i \frac{x^{2w} \alpha_i^{-2w} - 1}{x - \alpha_i}$$

Suma sa koeficijentima c_i je jednaka nuli zato što su koeficijenti sindroma za originalnu poruku jednaki 0. Uočimo dalje da je samo manje ili jednako od w elemenata iz prethodne sume jednako 1, jer smo pošli od pretpostavke da želimo kreirati kod koji može da dekodira poruku sa najviše w pogreški. Neka se pogreška dogodila na pozicijama β. Sada možemo sumirati prethodni izraz tako što ga dovedemo do istog zajedničkog sadržaoca (zajednički sadržalac za međusobno proste polinome u imeniocu je proizvod tih polinoma):

$$S(x) = \frac{\sum_{i \in \beta} (x^{2w} \alpha_i^{-2w} - 1) \prod_{\substack{j \in \beta \\ j \neq i}} (x - \alpha_j)}{\prod_{i \in \beta} x - \alpha_i} = \frac{x^{2w} \sum_{i \in \beta} \alpha_i^{-2w} \prod_{\substack{j \in \beta \\ j \neq i}} (x - \alpha_j) - \sum_{i \in \beta} \prod_{\substack{j \in \beta \\ j \neq i}} (x - \alpha_j)}{\prod_{i \in \beta} x - \alpha_i}$$

Ova relacija se sada može zapisati kao:

$$S(x)=\frac{x^{2w}u(x)}{l(x)}-\frac{w(x)}{l(x)} \qquad \text{odnosno} \qquad S(x)l(x)=x^{2w}u(x)-w(x)$$

gdje je:

$$l(x)=\prod_{i\in\beta} x-\alpha_i \qquad u(x)=\sum_{i\in\beta}\alpha_i^{-2w}\prod_{\substack{j\in\beta\\ j\neq i}}(x-\alpha_j) \qquad w(x)=\sum_{i\in\beta}\prod_{\substack{j\in\beta\\ j\neq i}}(x-\alpha_j)$$

Dakle, imamo jednačinu sa 3 suštinski nepoznata polinoma, ali od njih nas zapravo samo jedan interesuje. To je polinom $l(x)$ čije nule su α_i, i to za ono i koje pripada skupu β, odnosno, nule ovog polinoma na jedinstven način označavaju poziciju gdje se greška dogodila. Podsjetimo se da je pretpostavka da su svi α_i različiti. Stoga se ovaj polinom naziva i polinom mjesta pogreške ili polinom lokator greške. Preostala dva polinoma kod binarnih kodova nijesu od interesa, jer ako znamo poziciju pogreške umijemo i da izvršimo ispravku. Polinom $u(x)$ (ponekad se naziva polinomom vrednovanja pogreške) je interesantan kod nebinarnih kodova, jer nam daje i težinu greške, pa na osnovu polinoma $l(x)$ i $u(x)$ možemo da vršimo ispravku pogreški i kod nebinarnih kodova. Opet napominjemo da ćemo se zadržati na binarnim kodovima. Ostaje da vidimo kako iz prethodne jednačine, gdje je poznato samo $S(x)$ i to da kod može da ispravi do w pogreški možemo da odredimo $l(x)$.

8.6. Euklidski algoritam

Na dekodiranje BCH koda, na osnovu jednačine zasnovane na sindromskom polinomu u cilju određivanja polinoma lokatora greške (polinoma mjesta pogreške), standardno se primjenjuje procedura nazvana po grčkom matematičaru Euklidu - Euklidski algoritam. Euklid je u 3. vijeku prije nove ere u svojoj knjizi "Elementi" dokazao tvrdnju broj VII.2 koja kaže da ako je $a\geq b$ i $b\neq a$ da se određivanja najvećeg zajedničkog djelioca brojeva a i b svodi na određivanje najvećeg zajedničkog djelioca brojeva b i a mod b (ostatka pri dijeljenju a sa b ili u notaciji programskog jezika C $a\%b$). Na primjer za brojeve 1071 i 1029, ostatak pri dijeljenju je 42, pa se procedura nastavlja nad manjim od početnih brojeva (1029) i na dobijenom ostatku 42. Ostatak pri dijeljenju je sada 21. Procedura se zaustavlja kada su brojevi međusobno djeljivi i tada je manji od njih najveći zajednički djelilac. Kuriozitet ovog veoma efikasnog algoritma je činjenica da je originalno dokazan geometrijskim putem (Euklid je kao što vjerovatno znate otac geometrije). Euklidski algoritam ima i niz nešto složenijih tumačenja. Dva broja a i b se mogu zapisati kao:

$$a= q\ b+r$$

gdje je q njihov cjelobrojni količnik dok je r ostatak pri dijeljenju. Pored toga najveći zajednički djelioc a i b se može zapisati kao linearna kombinacija a i b:

$$\text{gcd}=u\ a+v\ b$$

U prethodnom izrazu iskorišćena je skraćenica **gcd** (najveći zajednički djelilac ili na engleskom *greatest common divisor*) da označi najveći zajednički djelilac. Npr. za a=52 i b=32 parametri u=-3 i v=5 daju najveći zajednički djelilac dok se za a=910 i b=143 gcd dobija za u=3 i v=-19. Postavlja se pitanje kako da dobijemo koeficijente u i v. To ćemo obaviti rekurzivnom procedurom koja je slična samom Euklidskom algoritmu. U prvoj iteraciji ćemo postaviti da je $r_{-1}=a$ dok je $r_0=b$. U svakoj iteraciju formiraćemo novo r_k , u_k i v_k tako da važi linearna kombinacija:

$$r_k=u_k\ a+v_k\ b$$

Prvo se određuje vrijednost r_k koja je ostatak pri dijeljenju r_{k-1} i r_k tako da važi:

$$r_{k-1}=q_{k+1}\ r_k+r_{k+1}$$

gdje je q_{k+1} količnik r_{k-1} i r_k. Pretpostavimo da smo već postigli:

$$r_k=u_k\,a+v_k\,b$$
$$r_{k-1}=u_{k-1}\,a+v_{k-1}\,b$$

Tada slijedi:

$$r_{k+1}=r_{k-1}-q_{k+1}r_k=u_{k-1}b+v_{k-1}b-q_{k+1}(u_ka+v_kb)=(u_{k-1}-q_{k+1}u_k)a+(v_{k-1}-q_{k+1}v_k)b$$

Sada se jasno mogu uspostaviti rekurzivne relacije:

$$u_{k+1}=u_{k-1}-q_{k+1}u_k \qquad\qquad v_{k+1}=v_{k-1}-q_{k+1}v_k$$

Da bi relacije bile kompletirane prve dvije iteracije su postavljene na $u_{-1}=1$, $u_0=0$, $v_{-1}=0$ i $v_0=1$. Za $a=910$ i $b=143$ možemo formirati sledeću tabelu sa međurezultatima algoritma.

k	***Q***	***R***	***U***	***V***
-1		910	1	0
0		143	0	1
1	6	52	1	-6
2	2	39	-2	13
3	1	13	3	-19
4	3	0	-11	70

Kada dobijemo 0 u ***R*** koloni znamo da je u prethodnom redu dobijen gcd. Nekoliko izuzetno interesantnih relacija se može pokazati za brojeve koji se dobijaju u pojedinim iteracijama Euklidskog algoritma. Ovdje su pobrojene najvažnije relacije:

$$r_{k-1}u_k-r_ku_{k-1}=\pm b \qquad r_{k-1}v_k-r_kv_{k-1}=\pm a \qquad u_{k-1}v_k-u_kv_{k-1}=\pm 1$$

Ovdje ćemo dokazati samo jednu od ovih važnih relacija, a dokaz ostalih se može obaviti na sličan način. Dokaz se obavlja matematičkom indukcijom. Pretpostavimo da važi neka od relacija i dokazujemo za naredno k:

$$r_ku_{k+1}-r_{k+1}u_k=r_k(u_{k-1}-q_{k+1}u_k)-(r_{k-1}-q_{k+1}r_k)u_k=r_ku_{k-1}-r_{k-1}u_k=\mp b$$

Ovo dakle važi ako važi u početnom koraku za $k=0$ a po datim uslovima važi. Treći izraz je možda najinteresantniji i odnosi sa na to da su u_k i v_k u svakoj iteraciji međusobno prosti. Da apostrofiramo da za svaki red prethodne tabele važi: $r_k=u_ka+v_kb$. To važi dakle i za poslednji red gdje se r_k anulira pa slijedi da je $u_ka+v_kb=0$ za ovaj red, odnosno važi da se v_k/u_k redukuje na $-a/b$. Važno je još uočiti da apsolutna vrijednost brojeva u koloni **R** strogo opada dok u kolonama **U** i **V** striktno raste počevši od kolone $k=1$.

8.7. Euklidski algoritam za polinome - Primjer

Posmatrajmo polinome $a(x)=x^8$ i $b(x)=x^6+x^4+x^2+x+1$. Odredimo njihov najveći zajednički djelilac na osnovu Euklidskog algoritma.

i	**U**	**V**	**R**	**Q**
-1	1	0	x^8	...
0	0	1	$x^6+x^4+x^2+x+1$	...
1	1	x^2+1	x^3+x+1	x^2+1
2	x^3+1	$x^5+x^3+x^2$	x^2	x^3+1
3	x^4+x+1	$x^6+x^4+x^3+x^2+1$	$x+1$	x
4	$x^5+x^4+x^3+x^2$	$x^7+x^6+x^3+x+1$	1	$x+1$
5	$x^6+x^4+x^2+x+1$	x^8	0	$x+1$

Nekoliko interesantnih činjenica. Najveći zajednički djelilac je 1 (poslednja nenulta kolona tabele **R**). Druga interesantna činjenica je da stepeni u koloni **R** strogo opadaju (kao što kod primjene kod

brojeva opada apsolutna vrijednost broja) dok stepeni u kolonama **U** i **V** rastu za $i>0$ kao što raste apsolutna vrijednost elemenata u kolonama **U** i **V** kod primjene ovog algoritma kod brojeva. Takođe, za elemente pojedinih vrsta važe veze (u istoj ili malo izmjenjenoj formi) koje smo uspostavili između redova ove matrice kod brojeva. Uočite da pošto smo dobili da je najveći zajednički djelilac 1 u posljednjoj vrsti ponovo imamo polazne polinome.

Za vježbu provjerite funkcionisanje Euklidskog algoritma na slučaju dva polinoma kojima je najveći zajednički djelilac različit od 1.

8.8. Primjena Euklidskog algoritma na polinome i u sklopu dekodiranja BCH koda

Postavlja se pitanje kakve veze ima Euklidski algoritam sa dekodiranje BCH koda, a posebno ove na prvi pogled iskomplikovane verzije ovog algoritma. Pogledajmo sada opet relaciju koju smo izveli u sekciji 8.7:

$$w(x) = x^{2w}u(x) - S(x)l(x)$$

Ovo podsjeća na relaciju: gcd=$u\ a+v\ b$ odnosno Euklidski algoritam po iteracijama $r_k=u_k\ a+v_k$ gdje su b polinomi x^{2w} dok je a polinom $S(x)$. Euklidski algoritam se u iterativnoj formi primjenjuje na ova dva polinoma u cilju određivanja najvećeg zajedničkog djelioca. Prva kolona u kojoj je stepen polinoma r_k manji od w determiniše sledeću relaciju:

$$v_k(x)=\lambda\ l(x)$$

gdje je λ multiplikativna konstanta različita od 0. Dakle, Euklidski algoritam nam determiniše polinom lokator greške! Pored toga za datu kolonu važe i sledeće relacije od značaja za nebinarni kod: $r_k(x)=\lambda\ w(x)$ i $u_k(x)=\lambda\ u(x)$. Prije nego pokažemo i dokažemo da je ovo istina moramo pokazati da su $l(x)$ i $u(x)$ međusobno prosti odnosno da im je najveći zajednički djelilac 1. Podsjetimo se $l(x)$ i $u(x)$.

$$l(x) = \prod_{i\in\beta} x-\alpha_i \qquad u(x) = \sum_{i\in\beta} \alpha_i^{-2w} \prod_{\substack{j\in\beta \\ j\neq i}} (x-\alpha_j)$$

Potencijalni djelioci $u(x)$ su polinom oblika $(x-\alpha_l)$ za $l\in\beta$. Polinom $u(x)$ je djeljiv sa $(x-\alpha_l)$ ako je $u(\alpha_l)=0$. Međutim ovo nije ispunjeno jer:

$$u(\alpha_l) = \sum_{i\in\beta} \alpha_i^{-2w} \prod_{\substack{j\in\beta \\ j\neq i}} (\alpha_l-\alpha_j) = \alpha_l^{-2w} \prod_{\substack{j\in\beta \\ j\neq i}} (\alpha_l-\alpha_j)$$

predstavlja proizvod nenultih elemenata koji daje nenultu vrijednost. Pogledajmo sada ekvivalent prethodne tabele u poslednjem redu prije nego se izvršavanje algoritma zaustavi (odnosno prije nego red polinoma r_k padne ispod w):

q_{j-1}	$r_{j-1}(x)$	$u_{j-1}(x)$	$v_{j-1}(x)$
q_j	$r_j(x)=w^*(x)$	$u_j(x)=u^*(x)$	$v_j(x)=l^*(x)$

Na svaki red pa i ovaj posljednji može se primjeniti odgovarajuća relacija koja povezuje **U** i **V** kolone sa polaznim polinomima. U ovom slučaju to je:

$$w^*(x) = u^*(x)x^{2w} + l^*(x)S(x)$$

Ako je $l^*(x)=\lambda\ l(x)$ dobili smo metodologiju za određivanje ovog važnog polinoma. Prije nego to pokažemo direktnim putem moramo prvo da pokažemo da je: stepen polinoma $l^*(x)$ manji ili jednak od w, dok su stepeni polinoma $u^*(x)$ i $w^*(x)$ manji od w:

$$\deg(l^*(x))\leq w \qquad \deg(u^*(x))<w \qquad \deg(w^*(x))<w$$

Treća relacija koja se odnosi na polinom $w^*(x)$ predstavlja zapravo uslov za zaustavljanje algoritma tako da ovo nije potrebno dokazivati. Na uzastopne kolone matrice može se primijeniti relacija:

$$l^*(x)r_{j-1}(x)\text{-}v_{j-1}(x)w^*(x)=\pm x^{2w}$$

Po istoj logici po kojoj je amplituda u koloni **R** opadala kod određivanja gcd brojeva i amplituda elemenata u kolonama **U** i **V** rasla po istoj logici i elementi u kolonama **U** i **V** imaju rastuće stepene, a elementi u koloni **R** imaju opadajuće stepene. Sada je lako uočiti da je stepen prvog člana u izrazu na lijevoj strani veći od izraza na desnoj strani pa slijedi da je:

$$\deg(l^*(x)r_{j-1}(x))=2w$$

Kako je po uslovima algoritma $\deg(r_{j-1}(x))\geq w$ to je $\deg(l^*(x))\leq w$. Na sličan se način iz izraza:

$$u^*(x)r_{j-1}(x)\text{-}u_{j-1}(x)w^*(x)=\pm S(x)$$

može dokazati da je $\deg(u^*(x))<w$, ali vam taj dokaz prepuštamo. Sada pođimo od početnog reda tabele i od reda sa polinomima koji su označeni sa *:

$$u(x)x^{2w}+l(x)S(x)=w(x)$$
$$u^*(x)x^{2w}+l^*(x)S(x)=w^*(x)$$

Kada pomnožimo prvu jednačinu sa $l^*(x)$ a drugu sa $l(x)$ i oduzmemo ove dvije jednačine dobijamo:

$$(l^*(x)u(x)-l(x)u^*(x))x^{2w}=l^*(x)w(x)-l(x)w^*(x)$$

Na lijevoj strani očigledno imamo polinom koji je stepena većeg ili jednakog od $2w$. Međutim, na desnoj strani imamo polinom koji je manji od $2w$ (dokažite). Da bi prethodna jednakost važila lijeva i desna strana izraza moraju biti jednake nuli, odnosno:

$$l^*(x)u(x)=l(x)u^*(x)$$
$$l^*(x)w(x)=l(x)w^*(x)$$

Veoma smo blizu da dokažemo da se Euklidski algoritam može koristiti za dekodiranje BCH koda. Već smo pokazali da je gcd za polinome $l(x)$ i $u(x)$ jednak 1. Ovo znači da postoje polinomi $f(x)$ i $g(x)$ takvi da važi:

$$f(x)l(x)+g(x)u(x)=1$$

Pomnožimo obije strane ove relacije sa $l^*(x)$ i zamijenimo $l^*(x)u(x)$ sa $l(x)u^*(x)$:

$$l^*(x)=(f(x)l^*(x)+g(x)u^*(x))l(x)=k(x)l(x)$$

Slično se može dobiti da je $w^*(x)=k(x)w(x)$ kao i $u^*(x)=k(x)u(x)$. Kako je $u_j(x)=u^*(x)$ i $v_j(x)=l^*(x)$ te kako se ± 1 može napisati kao linearna kombinacija $u_j(x)$ i $v_j(x)$ slijedi da je $(u^*(x),v^*(x))=1$. Odavde bi se moglo pokazati da važi recimo $u^*(x)=z(x)\ u(x)$, a ovo je moguće samo ako je polinom $k(x)$ multiplikativna konstanta odnosno $k(x)=\lambda$, i $z(x)=1/\lambda$. Ovim smo dokazali da Euklidski algoritam primijenjen na polinome do reda koji zadovoljava uvedeni uslov određuje polinom lokator greške sa tačnošću do muliplikativne konstante (kod binarnih kodova ova konstanta može biti samo ± 1) $l^*(x)=\lambda\ l(x)$. Nule ovog polinoma odgovaraju pozicijama na kojima su se dogodile pogreške. Obratite pažnju da smo na početku izvođenja prilikom određivanja sindromskog polinoma uzeli negativne stepene pojedinih kolona, pa to ima uticaj na pozicije grešaka u datom polinomu (praktično je riječ o obrnutim pozicijama).

Koraci u BCH algoritmu

1. Sračunati koeficijente sindromskog polinoma: $S_j=\sum_{i=0}^{n-1} r_i\alpha_i^{-j},\ \ j=1,2,...,2w.$

2. Izvesti Euklidov algoritam za x^{2w} i $S(x)=S_1+S_2x+...+S_{2w}x^{2w-1}$ i algoritam zaustavi kada je stepen polinoma $r_j(x)$ padne ispod w. Postaviti $l(x)=v_j(x)$ i $u(x)=u_j(x)$.
3. Odrediti B, skup nula polinoma $l(x)$.
4. Naći oblik pogreške $\mathbf{e}=[e_0,e_1,...,e_{n-1}]$ na ovaj način $e_i=1$ ako je $\alpha^{-i}\notin B$ i $e_i=0$ ako je $\alpha^{-i}\in B$.
5. Procijenjena kodna riječ je: $\mathbf{c}'=\mathbf{r}-\mathbf{e}$. U slučaju nebinarnog koda potrebno je na osnovu $u(x)$ sračunati težine pogreški da bi se obavilo uspješno dekodiranje. Mi to nećemo raditi.

BCH kod i dekodiranje BCH koda moguće je izvršiti na više načina, ali smo se opredijelili na ovaj jer za ostale tehnike neophodno je mnogo šire znanje iz algebarske kodne teorije.

8.9. Primjer dekodiranja poruke na osnovu Euklidskog algoritma

Neka je dat BCH(15,3) iz prethodnog primjera i neka je kodna riječ $\mathbf{r}=[111000110011110]$. Sindrom ima oblik:

$$S_j=1+\alpha^j+\alpha^{2j}+\alpha^{6j}+\alpha^{7j}+\alpha^{10j}+\alpha^{11j}+\alpha^{12j}+\alpha^{13j}$$

Za pojedine sindrome slijedi: $S_1=\alpha^7$, $S_2=\alpha^{14}$, $S_3=\alpha^{11}$, $S_4=\alpha^{13}$, $S_5=1$ i $S_6=\alpha^7$. Primjenimo Euklidski algoritma na polinome x^6 i $S(x)=\alpha^7+\alpha^{14}x+\alpha^{11}x^2+\alpha^{13}x^3+x^4+\alpha^7x^5$

i	V	R	Q
-1	[*]	[0,*,*,*,*,*,*]	...
0	[0]	[7,0,13,11,14,7]	...
1	[8,1]	[11,9,2,*,8]	[8,1]
2	[4,2,*]	[8,6,9,*]	[11,14]
3	[7,5,8,1]	[7,*,8]	[3,*]

Polinomi označeni listama eksponenata njihovih koeficijenata, dok je izostanak nekog stepena označen sa *. $r_2(x)=[8,6,9,*]=\alpha^8x^3+\alpha^6x^2+\alpha^9x$. Algoritam je zaustavljen kod i=3, jer je deg(r_3)<3 pa važi: $\sigma(x)=t_3(x)=\alpha^7x^3+\alpha^5x^2+\alpha^8x+\alpha$.

Korak (3) algoritam za dekodiranje pokazuje da $\sigma(x)$=0 ima tri rješenja i to: $x=\alpha^3,\alpha^9,\alpha^{12}$. Tako će algoritam za dekodiranje odrediti da se pogreške nalaze na mjestima n+1-stepen eksponenta (za stepen eksponenta≠0 dok je poziciju 0 zapravo prva pozicija), tako da se u našem slučaju dobijaju pozicije 13, 7, 4 pa konačno možemo procjeniti da je poslana riječ:

$$\mathbf{c}'=\mathbf{r}-\mathbf{e}=[111000110011110]+[000100100000100]=[111100010011010].$$

Dekodirana poruka je [1 0 0 1 0] (zapamtite da je primljena poruka "obrnuta" pa je u istom smislu "obrnuta" i dekodirana poruka).

Na sličnom principu kao BCH kodovi definisane su klase linearnih kodova (Reed Solomovi kodovi, Goppa kodovi i Golay kodovi).

Zadaci za vježbu

8.1 Provjeriti da li sistem definisan nad realnim alfabetom {0,1} sa operacijom sabiranja koje je ex-ili i sa operacijom množenja koje je logička i operacija predstavlja polje.

Rješenje. Pogledajmo osobine koje su opisane u sekciji 8.1 a odnose se na polje.

P_1:	za svako x, y, z važi u skupu P	$x+(y+z)=(x+y)+z$
P_2:	postoji nulti član	$x+0=0+x=x$
P_3:	postoji negativni element	$x+(-x)=0$
P_4:	za svako x,y u skupu P	$x+y=y+x$
P_5:	za svako x, y, z u skupu P	$x\,(yz)=(xy)\,z$
P_6:	postoji jedinični element	$x\,1=1\,x=x$
P_7:	za svako x različito od 0 postoji x^{-1}	$x^{-1}\,x=x\,x^{-1}=1$
P_8:	za svako x,y	$x\,y=y\,x$
P_9:	za svako x, y, z	$x\,(y+z)=x\,y+x\,z$

Osobine P_1, P_2, P_4, P_5, P_6, P_8 i P_9 nije teško dokazati u najgorem slučaju ako vam nisu poznate možete se poslužiti tabelom istinitosti logičkih operacija. Kod osobine P_3 negativni element je sam taj element (kod nas važi $x+x=0$). Kod osobine P_7 jedini nenulti element u skupu je $x=1$ pa kod njega važi da je $x^{-1}=x=1$.

8.2. Izvršiti Euklidski algoritam za polinome ako su polinomi od interesa su $a(x)=x^7+1$ i $b(x)=x^6+x^4+x^2+x+1$.

Rješenje: Kreirajmo Euklidski algoritam za ova dva polinoma:

i	U	V	R	Q
-1	1	0	x^7+1	...
0	0	1	$x^6+x^4+x^2+x+1$	...
1	1	x	$x^5+x^3+x^2+x+1$	x
2	x	x^2+1	x^3+1	x
3	x^3+x+1	x^4+x+1	x	x^2+1
4	$x^5+x^3+x^2+x$	x^6+x^3+1	1	x^2
5	$x^6+x^4+x^2+x+1$	x^7+1	0	x

8.3. Dokazati da ako je polinom:

$$c(x)=\sum_{k=0}^{n} a_k x^k$$

prost tada je prost i polinom:

$$c(x)=\sum_{k=0}^{n} a_k x^{n-k}$$

Rješenje: Dokaz se može provesti prilično intuitivno. Prvo je potrebno dokazati da ako imamo proizvod dva polinoma $a(x)b(x)$ koji je jednak nekom polinomu:

$$a(x)b(x)=\sum_{p=0}^{P} \alpha_p x^p$$

Tada će proizvod polinoma sa obrnutim koeficijentima (označimo ih sa $\bar{a}(x)$ i $\bar{b}(x)$) dobijamo polinom sa obrnutim koeficijentima:

$$\bar{a}(x)\bar{b}(x)=\sum_{p=0}^{P} \alpha_{P-p} x^p$$

Sada možemo zaključiti da ako $\sum_{p=0}^{P} \alpha_p x^p$ nije prost odnosno ako ima djelioce onda ni polinom $\sum_{p=0}^{P} \alpha_{P-p} x^p$ nije prost a isto važi i za proste polinome.

8.4. Dokazati da ako polinom ima nenulte koeficijente samo uz parne stepene ne može biti prost. **Napomena.** Zadaci 8.3 i 8.4 pomažu da se redukuje pretraživanje za proste polinome.

Rješenje. Uzmimo da imamo polinom sa samo parnim koeficijentima:

$$c(x) = \sum_{p=0}^{P} a_p x^{2p}$$

Posmatrajmo sada polinom

$$q(x) = \sum_{p=0}^{P} a_p x^p$$

Izvršimo kvadriranje ovog polinoma:

$$q^2(x) = \sum_{p_1=0}^{P} \sum_{p_2=0}^{P} a_{p_1} a_{p_2} x^{p_1+p_2} =$$
$$= \sum_{p=0}^{P} a_p^2 x^{2p} + 2 \sum_{p_1=0}^{P} \sum_{\substack{p_2=0 \\ p_1 \neq p_2}}^{P} a_{p_1} a_{p_2} x^{p_1+p_2}$$

Već smo vidjeli da važi da je stepen binarnog broja jednak samom tom broju te da je druga suma u predmetnom izrazu jednaka 0. Stoga možemo zaključiti da je $q^2(x)=c(x)$ a samim tim da je $c(x)$ djeljivo sa $q(x)$ pa da $c(x)$ nije prost polinom.

8.5. Provjeriti da li je polinom x^5+x^2+1 prost. Zatim provjeriti da li može poslužiti za stvaranje koda dužine 31. Ako može iskorititi ga za kreiranje BCH koda odgovarajuće dužine kodne riječi koji može da ispravi 2 odnosno 3 pogreške. **Napomena.** Može da se dogodi da kod ovog polinoma ne dobijete da su polinomi $p_3(x)$ i $p_5(x)$ sa svim jediničnim koeficijentima.

Rješenje. Već smo vidjeli da je predmetni polinom prost pa ćemo za početak da konstruišemo odgovarajuću kontrolnu matricu za Hammingov kod koji je u stanju da ispravi jednu pogrešku.

$$H = \begin{bmatrix}
1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

Odredimo sada redove polinoma $p_3(x)$ i $p_5(x)$. Konjugati za $p_3(x)$ su {3, 6, 12, 24, 48=48%31=17}. Sledeći konjugat je 34 a on je jednak 34%31=3 pa možemo zaključiti da je 5 različitih konjugata broja 3 odnosno da je $p_3(x)$ polinom petog reda. Konjugati petice su {5, 10, 20, 40=40%31=9, 18} dok bi sledeći konjugati bio 36=36%31=5 odnosno ponovljen prvi element skupa pa zaključujemo

da je i ovaj polinom petog stepena. Sada je potrebno da odredimo koeficijente ovih polinoma. Neka su:

$$p_3(x)=p_{35}x^5+p_{34}x^4+p_{33}x^3+p_{32}x^2+p_{31}x+p_{30}$$
$$p_5(x)=p_{55}x^5+p_{54}x^4+p_{53}x^3+p_{52}x^2+p_{51}x+p_{50}$$

Uvrstimo $x=a^3$ u prvi izraz i $x=a^5$ u drugi trebali da dobijemo nul-vektor:

$$p_3(a^3)=p_{35}[11111]+p_{34}[01110]+p_{33}[11010]+p_{32}[01010]+p_{31}[01000]+p_{30}[00001]=[00000]$$
$$p_5(a^5)=p_{55}[11001]+p_{54}[01100]+p_{53}[11111]+p_{52}[10001]+p_{51}[00101]+p_{50}[00001]=[00000]$$

Odavde slijedi $p_{35}+p_{33}=0$ a po istoj logici koju smo ranije objašnjavali slijedi da su oba $p_{35}=1$ i $p_{33}=1$. Kako je $p_{35}+p_{30}=0$ slijedi da je $p_{30}=1$. Dalje, imamo da je $p_{35}+p_{34}=1$ (za srednji bit) pa je odatle $p_{34}=1$. Za četvrti bit od početka slijedi $p_{35}+p_{34}+p_{33}+p_{32}=0$ a to dalje znači da je $p_{32}=1$. Konačno za prvi bit od početka slijedi $p_{35}+p_{34}+p_{33}+p_{32}+p_{31}=0$ a to onda znači da je $p_{31}=0$ (ne jedan kao u dosadašnjim primjerima). Time dobijamo da je $p_3(x)$ jedan od prostih polinoma 5-tog reda:

$$p_3(x)=x^5+x^4+x^3+x^2+1$$

Što se tiče polinoma $p_5(x)$ da bi pojednostavili potragu možemo poći od činjenice da je on zasigurno polinom petog reda što implicira da je $p_{55}=1$. Uvidom u ostale jednačine slijedi da je $p_{53}=0$ (za četvrti bit od početka) a to dalje znači da je $p_{54}=1$, $p_{52}=1$, $p_{51}=1$ i $p_{50}=1$. Na ovaj način ponovo dobijamo jedan od prostih polinoma petog reda:

$$p_5(x)=x^5+x^4+x^2+x+1$$

Generatorski polinom koda BCH(31,3) je:

$$g(x)=p(x)p_3(x)p_5(x)=(x^5+x^2+1)(x^5+x^4+x^3+x^2+1)(x^5+x^4+x^2+x+1)=$$
$$=x^{15}+x^{11}+x^{10}+x^9+x^8+x^7+x^5+x^3+x^2+x+1$$

Za dalji rad vam prepuštamo određivanje odgovarajućeg hardvera za kodiranje ovog koda i provjera rada sistema i dekodiranje za slučaj od jedne do tri pogreške.

8.6. Kod BCH(255,2) je najpopularnih BCH kod. Zbog čega? Kreirati polinom koji formira barem jedan takav kod.

Rješenje: Bez želje da detaljnije ulazimo u ovu problematiku dajemo osnovni prosti polinom osmog stepena koji se može koristiti za kreiranje BCH kodova sa 255 bita dugom kodnom riječju uz napomenu da ako koristimo samo ovaj polinom dobijamo odgovarajući Hammingov kod (255,247).

$$p(x)=x^8+x^4+x^3+x^2+1$$

Na osnovu predmetnog polinoma procedurom koja postaje prilično komplikovana za ručnu obradu sa porastom reda polinoma i sa dužom kodnom riječju dobijamo polinom:

$$g(x)=x^{16}+x^{14}+x^{13}+x^{11}+x^{10}+x^9+x^8+x^6+x^5+x+1$$

255 bita koliko je kodna riječ ovog koda predstavlja veoma pogodnu kombinaciju jer je broj informacionih bita 239 a to predstavlja (gotovo) 30 riječi ASCII koda po 8 bita dok se kodiranje vrši putem (približno) sa 3e kodne riječi po 8 bita.

8.7. Dokazati sledeće stavove za pojedine vrste u tabeli koja se dobija primjenom Euklidskog algoritma (radi skraćivanja zapisa izostavljena je zavisnost od *x*-a).

A	$v_i r_{i-1} - v_{i-1} r_i = (-1)^i a$	$0 \le i \le n+1$
B	$u_i r_{i-1} - u_{i-1} r_i = (-1)^i b$	$0 \le i \le n+1$
C	$u_i v_{i-1} - u_{i-1} v_i = (-1)^{i+1}$	$0 \le i \le n+1$
D	$u_i a + v_i b = r_i$	$-1 \le i \le n+1$
E	$\deg(u_i) + \deg(r_{i-1}) = \deg(b)$	$1 \le i \le n+1$
F	$\deg(v_i) + \deg(r_{i-1}) = \deg(a)$	$0 \le i \le n+1$

Napomena. deg označava stepen polinoma uz najveći nenulti koeficijent.

Rješenje. Dokazaćemo jednu po jednu osobinu.

(A) $v_i r_{i-1} - v_{i-1} r_i = (-1)^i a$

Riješimo ovo rekurzivno putem indukcije.

Za i=0 slijedi da treba da dokažemo da je

$$v_0 r_{-1} - v_{-1} r_0 = a$$

Kako je v_0=1, v_{-1}=0, r_{-1}=a i r_0=b time je navedena činjenica dokazana. Sada po pravilima indukcije treba dokazati da pod uslovom da relacija važi za i

$$v_i r_{i-1} - v_{i-1} r_i = (-1)^i a$$

da važi i za i+1:

$$v_{i+1} r_i - v_i r_{i+1} = (-1)^{i+1} a$$

Sada uvrstimo u ovu relaciju:

$$v_{i+1} = v_{i-1} - q_{i+1} v_i$$
$$r_{i+1} = r_{i-1} - q_{i+1} r_i$$

pa dobijamo:

$$(v_{i-1} - q_{i+1} v_i) r_i - v_i (r_{i-1} - q_{i+1} r_i) = v_{i-1} r_i - q_{i+1} v_i r_i - v_i r_{i-1} + q_{i+1} v_i r_i = v_{i-1} r_i - v_i r_{i-1} = -(-1)^i a = (-1)^{i+1} a$$

čime je ovaj dokaz izvršen.

(B) $u_i r_{i-1} - u_{i-1} r_i = (-1)^i b$. Dokazuje se na potpuno isti način kao prethodno a već je dokazano u tekstu.

(C) $u_i v_{i-1} - u_{i-1} v_i = (-1)^{i+1}$

Prvo dokažimo da predmetna relacija važi za i=0:

$u_0 v_{-1} - u_{-1} v_0 = -1$

Uvrštavajući početne vrijednosti u_0=0, u_{-1}=1, v_0=1, v_{-1}=0 dobijamo -1=-1 (odnosno u aritmetici po modulu 2: 1=1). Sada treba da dokažemo da ako važi gore navedena realcija ((C)) da važi relacija i za i=i+1:

$u_{i+1} v_i - u_i v_{i+1} = (-1)^{i+2}$

Uvrštavajući u ovu jednačinu pravila ažuriranja u kolonama **U** i **V**:

$u_{i+1} = u_{i-1} - q_{i+1} u_i$
$v_{i+1} = v_{i-1} - q_{i+1} v_i$

$(u_{i-1} - q_{i+1} u_i) v_i - u_i (v_{i-1} - q_{i+1} v_i) =$
$u_{i-1} v_i - u_i v_{i-1} - q_{i+1} u_i v_i + q_{i+1} u_i v_i =$
$u_{i-1} v_i - u_i v_{i-1} = -(-1)^{i+1} = (-1)^{i+2}$

(D) $u_i a + v_i b = r_i$

Za i=0 predmetna relacija zasigurno važi jer predstavlja uslov inicijalizacije algoritma. Pod pretpostavkom da važi (D) dokažimo da važi relacija za i=i+1:

$u_{i+1} a + v_{i+1} b = r_{i+1}$

Uvrštavanjem relacija za u_{i+1} i v_{i+1} u lijevu stranu izraza dobijamo:

$(u_{i-1} - q_{i+1} u_i) a + (v_{i-1} - q_{i+1} v_i) b =$
$u_{i-1} a + v_{i-1} b - q_{i+1} u_i a - q_{i+1} v_i b =$
$r_{i-1} - q_{i+1} r_i = r_{i+1}$

Poslednja relacija slijedi iz rekurzivne relacije za ostatak koja je korišćena i pokazana u teorijskom dijelu.

(E) $\deg(u_i) + \deg(r_{i-1}) = \deg(b)$

Ovdje ćemo odstupiti od dokazivanja matematičkom indukcijom pa ćemo se prebaciti na direktni dokaz. Pođimo od sledećeg izraza:

$$r_{i-1} u_i - r_i u_{i-1} = \pm b$$

Stepen polinoma r_i opada sa rastom i pa važi $\deg(r_i) < \deg(r_{i-1})$ dok stepen polinoma u_i raste $\deg(u_i) > \deg(u_{i-1})$. Stoga je $\deg(r_{i-1} u_i) > \deg(r_i u_{i-1})$ pa odatle slijedi da je $\deg(r_{i-1} u_i - r_i u_{i-1}) = \deg(r_{i-1} u_i) = \deg(r_{i-1}) + \deg(u_i) = \deg(b)$ čime je dokaz završen.

(F) $\deg(v_i) + \deg(r_{i-1}) = \deg(a)$

Dokazati kao pod (E)!

8.8. a) Uzmite polinom $a(x)=x^{10}$ i $b(x)=x^4+x^3+1$ i odraditi Euklidski algoritam. b) Da li je polinom $P(x)=x^4+x^3+x^2+x+1$ prost i ako jeste konstruisati kod za korekciju greške sa 15 kodnih bita (bita informacije plus provjera parnosti). Da li je moguće kreirati ovakav kod i ako nije zbog čega? c) Odredite polinome 5-tog reda (sa stepenom petog reda kao najvećim članom) koji mogu da posluže za kreiranje Hammingovog koda (31,26).

Rješenje: (a) Izvršimo Euklidski algoritam sa navedenim polinomima.

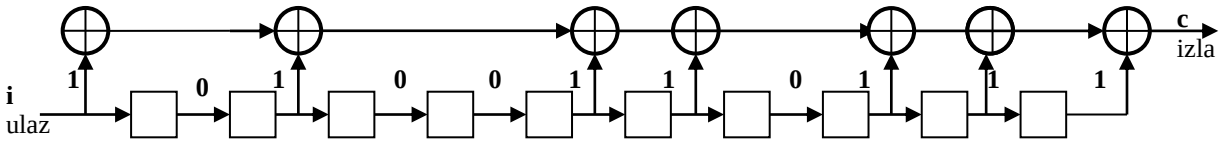
i	U	V	R	Q
-1	1	0	x^{10}	...
0	0	1	x^4+x^3+1	...
1	1	$x^6+x^5+x^4+x^3+x$	x^3+x	$x^6+x^5+x^4+x^3+x$
2	$x+1$	$x^7+x^3+x^2+x+1$	x^2+x+1	$x+1$
3	x^2	$x^8+x^7+x^6+x^5+x^3+x+1$	$x+1$	$x+1$
4	x^3+x+1	$x^9+x^8+x^6+x^4+x^3+1$	1	x
5	x^4+x^3+1	x^{10}	0	$x+1$

8.9. Odrediti generatorski polinoma za BCH kod (15,3) koristivši prostog polinom x^4+x^3+1. Prikazati realizaciju pomoću pomjeračkog registra.

Rješenje. Procedura za određivanje predmetnog generatorskog polinoma je ista kao i u slučaju prostog polinoma x^4+x+1. Ponovo se dobija da su $p_3(x)=x^4+x^3+x^2+x+1+$ i $p_5(x)=x^2+x+1$ (provjerite). Generatorski polinomi za BCH(15,2) i BCH(15,3) su:

$$g_1(x)=(x^4+x^3+1)(x^4+x^3+x^2+x+1)=x^8+x^4+x^2+x+1$$
$$g_2(x)=(x^4+x^3+1)(x^4+x^3+x^2+x+1)(x^2+x+1)=x^{10}+x^9+x^8+x^6+x^5+x^2+1$$

Koder BCH(15,3) koda je dat na slici dolje.



Sami za vježbu prikažite BCH(15,2) kod.

8.10. Data je poruka 10111. Kodirati je BCH kodom BCH(15,3) iz prethodnog zadatka. Generisati 1 pogrešku i izvršiti dekodiranje. Generisati dvije pogreške i izvršiti dekodiranje i generisati tri pogreške i izvršiti dekodiranje.

Rješenje. Kodni polinom u ovom slučaju je:

$$c(x)=i(x)g_2(x)=(x^4+x^2+x+1)(x^{10}+x^9+x^8+x^6+x^5+x^2+1)=$$
$$x^{14}+x^{13}+x^9+x^6+x^5+x^3+x+1$$

Ovu kodnu riječ možemo zapisati kao [110001001101011]. Uzmimo sada da imamo tri slučaja. U prvom jednu grešku i to na poziciji 5:

[110011001101011]

U drugom sluačaju dvije greške na pozicijama 10 i 14:

[110001001001001]

Konačno u trećem slučaju imamo tri pogreške na pozicijama 3, 8 i 12:

[111001011100011]

Koeficijenti sindromskog polinoma u prvom slučaju su:

$$S_j=1+a^j+a^{4j}+a^{5j}+a^{8j}+a^{9j}+a^{11j}+a^{13j}+a^{14j}$$

Pogledajmo sada koeficijente:

$$S_1=1+a+a^4+a^5+a^8+a^9+a^{11}+ a^{13}+a^{14}=a^4$$

$$S_2=1+a^2+a^8+a^{10}+a+a^3+a^7+a^{11}+a^{13}=a^8$$

$$S_3=1+a^3+a^{12}+1+a^9+a^{12}+a^3+a^9+a^{12}=a^{12}$$

$$S_4=1+a^4+a^1+a^5+a^2+a^6+a^9+a^7+a^{11}=a^1$$

$$S_5=1+a^5+a^5+a^{10}+a^{10}+1+a^{10}+a^5+a^{10}=a^5$$

$$S_6=1+a^6+a^9+1+a^3+a^6+a^6+ a^3+a^9=a^9$$

i	V	R	Q
-1	[*]	[0,*,*,*,*,*,*]	...
0	[0]	[4,8,12,1,5,9]	...
1	[6,2]	[6]	[6,2]

Polinom lokator greške je $l(x)=a^6x+a^2$ pa je nula ovog polinoma $x=a^2/a^6=a^{-4}=a^{11}$. Zaključujemo da je pozicija pogreške 5-ta što odgovara stvarnosti.

U drugom slučaju koeficijenti sindromskog polinoma su

$$S_j=1+a^j+a^{5j}+a^{8j}+a^{11j}+a^{14j}$$

Konkretne vrijednosti koeficijenata sindromskog polinoma su redom:

$$S_1=a^{12},\ S_2=a^9,\ S_3=a^{13},\ S_4=a^3,\ S_5=a^{10} \text{ i } S_6=a^{11}$$

i	V	R	Q
-1	[*]	[0,*,*,*,*,*,*]	...
0	[0]	[11,10,3,13,9,12]	...
1	[4,3]	[0, 5, 2, 0, 0]	[4,3]
2	[0, 5, 8]	[5]	[11,3]

Polinom lokator greške je $l(x)=x^2+a^5x+a^8$. Njegove nule su $x=a^2$ i $x=a^6$ što ukazuje da se pogreške dešavaju na pozicijama 14 i 10.

Slučaju sa tri pogreške odgovara primljena poruka:

[111001011100011]

Sindrom je dakle:

$$S_j=1+a^j+a^{2j}+a^{5j}+a^{7j}+a^{8j}+a^{9j}+a^{13j}+a^{14j}$$

Koeficijenti sindromskog polinoma su:

$$S_1=a^8,\ S_2=a,\ S_3=a^3,\ S_4=a^2,\ S_5=a^5,\ S_6=a^6$$

Obavimo sada Euklidski algoritam.

i	V	R	Q
-1	[*]	[0,*,*,*,*,*,*]	...
0	[0]	[6,5,2,3,1,8]	...
1	[9,8]	[5, 4, 7, 0, 1]	[9,8]
2	[10, 9, 0]	[10, 10, 13, 8]	[1,*]
3	[5, 5, 8, 0]	[5, *, 8]	[10, 6]

Polinom lokator greške je $l(x)=a^5x^3+a^5x^2+a^8x+1$. Nule ovog polinoma su $x=a^4$, $x=a^8$ i $x=a^{13}$. Odavde slijedi da su pogreške na pozicijama 12, 8 i 3 odnosno pravilno smo identifikovali pozicije pogreške.

8.11. Odabrati neki prosti polinom 5 stepena i na osnovu njega kreirati generatorski polinom koda koji može da ispravi 4 pogreške. Za željenu dužinu informacione riječi k kolika je potrebna dužina kodne riječi.

Rješenje: Već smo formirali BCH kod BCH(31,3). Uzmimo primjer iz zadatka 8.5

$$g(x)=p(x)p_3(x)p_5(x)=(x^5+x^2+1)(x^5+x^4+x^3+x^2+1)(x^5+x^4+x^2+x+1)=$$
$$=x^{15}+x^{11}+x^{10}+x^9+x^8+x^7+x^5+x^3+x^2+x+1$$

a sada je vrijeme da formiramo kod BCH(31,4). Konjugati a^7 su stepeni: (7, 14, 28, 56%31=25, 50%31=19) dok je naredni 38%31=7. Dakle, ponovo imamo 5 različitih konjugata odnosno polinom 5-reda. Dakle, kod koji je u pitanju je (31, 11) kod.

$$p_7(a^7)=p_{75}[10000]+p_{74}[10110]+p_{73}[11000]+p_{72}[11101]+p_{71}[10100]+p_{70}[00001]=[00000]$$

Lako možemo poći od pretpostavke da je $p_{75}=1$ (polinom je petog stepena). Odavde slijedi $p_{74}+p_{73}+p_{72}+p_{71}=1$. Za sumiranje bitova $p_{73}+p_{72}=0$ pa sada prvi uslov svodimo $p_{74}+p_{71}=1$. Za treće bite važi $p_{74}+p_{72}+p_{71}=0$ a odavde lako zaključujemo da je $p_{72}=1$ a odavde slijedi da je $p_{73}=1$. Provjerom za četvrte bite $p_{74}=0$ pa je $p_{71}=1$. Konačno provjerom petih bita slijedi $p_{70}=1$. Dakle, polinom $p_7(x)$ je:

$$p_7(x)=x^5+x^3+x^2+1$$

Generatorski polinom je sada:

$$g'(x) = g(x)p_7(x) = x^{20} + x^{18} + x^{17} + x^{16} + x^{13} + x^{12} + x^{11} + x^8 + x^7 + x^3 + x + 1$$

Za ilustrativne potrebe realizovan je MATLAB program koji prati osnovne korake u kodiranju i dekodiranje BCH(15,3) koda (sa oba osnovna generatorska polinoma).

Osnovni program main_bch.m

```
clear
global MM
c=input('Unesi kodni vektor sa 5 bitova ');
disp('Kodna rijec koja se kodira');
disp(c)
pause(1)
tip=input('Ako je gen. polinom x^4+x+1 unijeti 1 a za x^4+x^3+1 unijeti 2 ');
MM=parametri_bch(2,tip);
x=bch(c,tip);
disp('Kodirana kodna rijec');
disp(x)
pause(1)
izmj=input('Unesi pozicije na kojima se mijenja kodna rijec ');
r=izmjeni(x,izmj);
disp('Izmjenjena kodna rijec: ')
disp(r)
pause(1)
sindrom=sindr_pol(r);
disp('Sindrom ')
help_prikaz(sindrom)
disp('Koraci u euklidovom algoritmu')
ts=euklidovalgoritam(sindrom);
npoz=nule_polin(ts);
disp('Nule polinoma t su: ')
disp(npoz)
pause(1)
npoz=(npoz>0).*(16-npoz)+(npoz==0);
disp('Sto znaci da su zapravo se greske dogodile: ')
disp(npoz)
pause(1)
xr=izmjeni(r,npoz);
disp('Popravljena kodna poruka: ')
disp(xr)
pause(1)
[ce,ost]=dij_bin_pol(xr(15:-1:1),parametri_bch(1,tip));
ce=[zeros(1,5-length(ce)),ce];
ce=ce(5:-1:1);
if(isempty(ost))
    disp('Dekodiranje uspjelo ')
    disp(ce)
    pause(1)
    disp('Uporedi sa polaznim vektorom ')
    disp(c)
    pause(1)
else
   disp('Dekodiranje nije uspjelo')
end
```

Objašnjenje glavnog programa:

Nakon unosa informacionog polinoma, naredba `MM=parametri_bch(2,tip);` suštinski formira kontrolnu matricu (ovdje to radimo zbog lakšeg obavljanja operacija sabiranja i množenja).

Iz navedenih razloga `MM` je deklarisana kao globalna promjenljiva. Naredba `bch` ima namjenu da kreira kodni polinom. Ova funkcija ima jednostavnu realizaciju:

```
function y=bch(c,tip)
y=rem(conv(c(5:-1:1),parametri_bch(1,tip)),2);
y=y(15:-1:1);
```

Uočimo da smo ovdje koristili naredba `parametri_bch(1,tip)` i vidimo da ova funkcija sa prvim argumentom jednakim `1` daje generatorski polinom dok sa `2` daje generatorsku matricu. Promjenljiva `tip` se odnosi na to koji je osnovni prosti polinom izabran (x^4+x+1 ili x^4+x^3+1). Nakon toga korisnik zadaje pozicije gdje su se greške dogodile pa se funkcijom `izmjeni` vrši modifikacija kodnog polinoma (dogodile se pogreške u kanalu). Realizacija ove proste funkcije je:

```
function z=izmjeni(y,poz)
e=zeros(1,length(y));
e(1,poz)=1;
z=xor(y,e);
```

Naredbom `sindrom=sindr_pol(r);` vrši se generisanje sindromskog polinoma:

```
function s=sindr_pol(r)
for k=1:6
    s(2,k)=0;
    for l=1:15
            if(r(l)==1)
                if(s(2,k)==0)
                    s(1,k)=rem(k*(l-1),15);
                    s(2,k)=1;
                else
                    zz=zbirstep(s(1,k),rem(k*(l-1),15));
                    if(isempty(zz))
                        s(2,k)=0;
                    else
                        s(1,k)=zz;
                    end
                end
            end

    end
end
```

Razmotrite sami realizaciju ove funkcije. Unutar ove funkcije koristi se naredba `zbirstep` koja sabira dvije kolone generatorske matrice osnovnog polinoma (onog koji ispravlja jednu pogrešku) i vraća odgovarajući stepen koji odgovara rezultatu:

```
function z=zbirstep(z1,z2);
global MM
x=xor(MM(z1+1,:),MM(z2+1,:));
for k=1:15
    if(sum(abs(x-MM(k,:)))==0)
        z=k-1;
        return
    end
end
z=[];
```

Najsloženiji dio programa je `ts=euklidovalgoritam(sindrom);` koja služi za provođenje Euklidskog algoritma

```
function ts=euklidovalgoritam(sind)
tp=[0;0];
rp=[0 0 0 0 0 0 0;0 0 0 0 0 0 1];
ts=[0;1];
rs=sind;
n=size(rp,2);
while(n>3)
    [kol,ost]=dijeli_polin_f(rp,rs);
    disp('Kolicnik ')
    help_prikaz(kol)
    disp('Ostatak ')
    help_prikaz(ost)
    tr=saberi_polin_f(tp,mnozi_polin_f(kol,ts));
    rp=rs;
    rs=ost;
    tp=ts;
    ts=tr;
    disp('Tekuci vektor t ')
    help_prikaz(ts)
    disp('Tekuci vektor r ')
    help_prikaz(rs)
    n=size(rs,2);
end
```

Ključna funkcija kod Euklidskog algoritma je ona koja vrši dijeljenje polinoma sa koeficijentima koji su stepeni nule prostog polinoma:

```
[kol,ost]=dijeli_polin_f(rp,rs);
```

Realizacija ove funkcije je data kao:

```
function [kol,ost]=dijeli_polin_f(imen,djel)
M=max(find(imen(2,:)==1));
imen=imen(:,1:M);
N=max(find(djel(2,:)==1));
djel=djel(:,1:N);
kol=zeros(2,M-N+1);
k=M-N+1;
while M>=N
    kol(1,M-N+1)=rem(imen(1,M)-djel(1,N)+15,15);
    kol(2,M-N+1)=1;
    imen=saberi_polin_f(imen,mnozi_polin_f(kol(:,1:M-N+1),djel));
    M=max(find(imen(2,:)==1));
    imen=imen(:,1:M);
end
ost=imen;
```

Kod Euklidskog algoritma smo prikazali sve međurezultate putem naredbe `help_prikaz` jer nam je ideja da studenti mogu da kontrolišu svoje rezultate. Funkcije `saberi_polin_f` i `mnozi_polin_f` se koriste za sabiranje i množenje polinoma sa koeficijentima koji su stepeni nula prostog polinoma. Realizacija funkcije `saberi_polin_f`

```
function q3=saberi_polin_f(q1,q2)
M=size(q1,2); N=size(q2,2);
q3=zeros(2,max(M,N));
for m=1:min(M,N)
    if(q1(2,m)==1 & q2(2,m)==1)
        aq=zbirstep(q1(1,m),q2(1,m));
        if(isempty(aq))
```

```
            q3(2,m)=0;
        else
            q3(2,m)=1;
            q3(1,m)=aq;
        end
    elseif(q1(2,m)==1)
        q3(1,m)=q1(1,m);
        q3(2,m)=1;
    elseif(q2(2,m)==1)
        q3(1,m)=q2(1,m);
        q3(2,m)=1;
    end
end
if(M>N)
    q3(:,min(M,N)+1:M)=q1(:,min(M,N)+1:M);
else
    q3(:,min(M,N)+1:N)=q2(:,min(M,N)+1:N);
end
```

Realizacija funkcije `mnozi_polin_f`

```
function q3=mnozi_polin_f(q1,q2)
M=size(q1,2); N=size(q2,2);
q3=zeros(2,M+N-1);
for m=1:M
    for n=1:N
        if(q1(2,m)==1 & q2(2,n)==1 & q3(2,m+n-1)==0)
            q3(1,m+n-1)=rem(q1(1,m)+q2(1,n),15);
            q3(2,m+n-1)=1;
        elseif(q1(2,m)==1 & q2(2,n)==1)
            aq=zbirstep(q3(1,m+n-1),rem(q1(1,m)+q2(1,n),15));
            if(isempty(aq))
                q3(2,m+n-1)=0;
            else
                q3(1,m+n-1)=aq; q3(2,m+n-1)=1;
            end
       end
    end
end
```

Naredba `npoz=nule_polin(ts);` daje nule polinoma lokatora pogreške. Realizacija ove funkcije je:

```
function npoz=nule_polin(ts)
pp=0;
for k=0:14
    ind=0;
    for l=1:size(ts,2)
        if(ind==0 & ts(2,l)==1)
            tek=rem(ts(1,l)+(l-1)*k,15);
            ind=1;
        elseif(ts(2,l)==1)
            tek=zbirstep(tek,rem(ts(1,l)+(l-1)*k,15));
            if(isempty(tek))
                ind=0;
            end
        end
    end
    if(ind==0)
        npoz(pp+1)=k;
        pp=pp+1;
    end
```

```
end
    if(pp==0),npoz=[];end
```

Prava pozicija pogreški se dobija kao:

```
npoz=(npoz>0).*(16-npoz)+(npoz==0);
```

Kao što znamo nule koje dobijamo a^k za k>0 odgovaraju pozicije 16-k dok stepenu a^0 odgovara pozicija 1. Provjera da li je dekodiranje uspješno obavljeno postiže se dijeljenjem polinoma:

```
[ce,ost]=dij_bin_pol(xr(15:-1:1),parametri_bch(1,tip));
```

Realizacija funkcije `dij_bin_pol`

```
function [kol,ost]=dij_bin_pol(x1,x2)
m1=min(find(x1==1)); x1=x1(m1:length(x1));
m2=min(find(x2==1)); x2=x2(m2:length(x2));
if(length(x2)>length(x1))
    kol=0;
    ost=x1;
else
    kol1=[1,zeros(1,length(x1)-length(x2))];
    pro1=rem(conv(kol1,x2),2);
    kol2=pol_bin_sum(x1,pro1);
    mk=min(find(kol2==1));
    if(isempty(mk))
        ost=[];
        kol=kol1;
    else
        kol2=kol2(mk:length(kol2));
        [kolx,ost]=dij_bin_pol(kol2,x2);
        kol=pol_bin_sum(kol1,kolx);

    end
end
```

U slučaju da na kraju programa dobijemo nenulti ostatak pri dijeljenju ili da se dobijeni količnik razlikuje od polaznog polinoma prijavljuemo pogrešku u suprotno imamo uspješno obavljeno dekodiranje.

U okviru MATLAB-ovog komunikacionog toolbox-a postoji naredba `fec.bchenc` kojom se mogu generisati parametri BCH koda. Naredbom `enc=fec.bchenc(15,5);` dobijamo sve važne podatke za koder u ovom slučaju to je BCH kod (15,3) (parametri naredbe su dužina riječi i broj informacionih bita). Kodiranje se obavlja naredbom `code=encode(enc,[1 1 0 1 1]');`. Promjenimo sada tri bita `code([3 9 11])=xor(code([3 9 11]),[1 1 1]')`. Dekodiranje se obavlja obrnutom procedurom:

```
dec=fec.bchdec(15,5);
decode=decode(dec,code);
```

Dobijeni rezultat je jednak polaznoj kodoj riječi:

```
decode'
     1     1     0     1     1
```

Preporučujemo studentima da izuče ove naredbe kao i prateći toolobox u kojem postoje mnoge korisne opcije u dijelu BCH kodiranja i u opšte kodne teorije.

Provjera da li ste savladali ključne elemente kodiranja kanala

1. Osnovni pojmovi o informacionom kanalu: kanal, tranzitivna matrica uslovnih vjerovatnoća, kapacitet kanala (definicija i osobine sa dokazom), simetrični i slabosimetrični kanali, formulacija kodne teoreme i njene posljedice (bez dokaza), kapacitet kanala sa Gausovskim šumom (definicija bez dokaza), određivanje kodnog količnika (code rate) za sve definisane kodove.

2. Izračunavanje kapaciteta kanala preko definicije i preko teoreme za slabosimetrične kanale.

3. Kodovi za detekciju greške (ASCII, suma sume kod, kodovi 2 od 5 i kod 4 od 7). Kodovi za ispravljanje jedne pogreške: pravougaoni, trougaoni, Hammingovi kodovi (7,4) i (15,11). Hammingovi kodovi za jedno ispravljanje ili detekciju dvije greške. Izračunavanje vjerovatnoće da kod uspješno radi i da kod ne uspije da ispravi ili detektuje pogreške. Hammingova distanca, Hammingova težina, minimalna težina koda i postupci za određivanje minimalne težine.

4. Kontrolna matrica kodova, generišuća matrica kodova, Hammingovi kodovi sa provjerom parnosti na pravilnim binarnim pozicijama, Hammingovi kodovi sa provjerom parnosti na posljednjim bitovima u kodu, generišuća matrica kodova, preuređivanje kolona kodne matrice Hammingovog koda. Generisanje kodova preko prostih binarnih polinoma. Kreiranje kontrolne matrice za Hammingov kod koja može da posluži za ispravljanje dvije pogreške i postupak otkrivanja dvije pogreške. Pojam interlivinga.

5. Obnoviti gradivo vezano za kodiranje izvora..

Za svaku od tema 1-5 treba savladati prateću teoriju i definisati jedan zadatak koji reflektuje materiju koja je opisana.

Poglavlje IX
NEBINARNI KODOVI

9.1. Nebinarni Hammingovi kodovi – Kontrolna i generišuća matrica

Prije nego uvedemo kontrolnu matricu za nebinarne Hammingov kodove (ovdje ćemo za sada demonstrirati samo mogućnost jednog ispravljanja greške). Napišimo na sledeći način kontrolnu matricu Hammingovog koda (7,4). Neka polazeći od poslednje ka prvim kolonama svaka kolona predstavlja binarno zapisan broj počevši od 1 do 7:

$$-\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Dekodiranje ovog koda se obavlja množenjem primljene kodne riječi sa (transponovanom) kontrolnom matricom.

$$\mathbf{S} = \mathbf{c}\mathbf{H}^T$$

U slučaju da dobijemo da je sindrom **S** jednak nekoj od kolona onda se greška pojavila na toj lokaciji (uočite da se svaka kolona pojavljuje samo jednom u kontrolnoj matrici čime je dekodiranje moguće na jedinstven način). Ako dobijemo sindrom sa svim nulama (ova kombinacija se ne pojavljuje u kontrolnoj matrici) podrazumijevamo da nema greške u kodnoj riječi. Napominjemo da je množenje obavljano kao logička i operacija (mada isti smisao ima i standardno matematičko množenje) dok je sabiranje obavljano preko ex-ili operacija koja se može smatrati i operacijom "po modulu 2" odnosno od dobijenog rezultata se odredi ostatak pri dijeljenju sa 2. Alternativno kontrolnu matricu možemo definisati tako što prvo postavimo jediničnu matricu na kraju kontrolne matrice a ispred njeređamo sve kolone koje se razlikuju od prethodno napisanih kolona:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Ako je kontrolna matrica zapisana u ovom obliku: $\mathbf{H} = [\mathbf{P} \mid \mathbf{I}_{n-k}]$ tada se generišuća matrica sa kojom množimo informacione bite da bi dobili kodnu riječ mogla zapisati kao:

$$\mathbf{G} = \left[\mathbf{I}_k \mid \mathbf{P}^T \right]$$

Nebinarni kodovi su definisani obično u aritmetici po modulu nekog većeg prostog broja. Npr. kod ternarnog koda kodni simboli se usvajaju kao 0, 1, 2 pa se operacije množenja i sabiranja obavljaju po modulo 3 aritmetici:

$$\begin{array}{lll} 0\cdot 0 = 0 & 0\cdot 1 = 0 & 0\cdot 2 = 0 \\ 1\cdot 0 = 0 & 1\cdot 1 = 1 & 1\cdot 2 = 2 \\ 2\cdot 0 = 0 & 2\cdot 1 = 2 & 2\cdot 2 = 1 \end{array} \qquad \begin{array}{lll} 0+0 = 0 & 0+1 = 1 & 0+2 = 2 \\ 1+0 = 1 & 1+1 = 2 & 1+2 = 0 \\ 2+0 = 2 & 2+1 = 0 & 2+2 = 1 \end{array}$$

Prije nego damo matematički formalniju definiciju nebinarnog Hammingovog koda posmatrajmo ternarni Hammingov kod (4,2). Neka je njegova kontrolna matrica (dobijena s neba pa u rebra) data kao:

$$\mathbf{H} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

Uočava se da je da smo postavili da su informacioni biti poslednji u kodnoj riječi. Neka je dobijena kodna riječ **c**=[1 2 2 0]. Sindrom kojeg dobijamo je u ovom slučaju:

$$\mathbf{c}\mathbf{H}^T = \begin{bmatrix} 1 & 2 & 2 & 0 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = [0 \;\; 0]$$

Dakle, u pitanju je ispravna kodna riječ jer je dobijen sindrom koji je jednak nul-vektoru. Neka se sada dogodila greška na prvom bitu i neka je prvi bit jednak 2. Dobijeni sindrom je:

$$\begin{bmatrix} 2 & 2 & 2 & 0 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = [2 \;\; 1]$$

Sindrom nam kaže da se greška dogodila na prvom simbolu jer je sadržaj sindroma jednak prvoj koloni kontrolne matrice. Za sada sve funkcioniše po logici kod binarnih kodova. Sada posmatrajmo sledeću situaciju neka je prvi simbol 0.

$$\begin{bmatrix} 0 & 2 & 2 & 0 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = [1 \;\; 2]$$

Dobili smo kolonu koja ne postoji u kontrolnoj matrici *H* ali i dalje identifikuje da se pogreška pojavljuje na prvoj poziciji jer je dobijeni sindrom jednak dvostrukoj vrijednosti prve kolone. Naime,

$$2[2 \;\; 1] = [4 \;\; 2] = [1 \;\; 2]$$

Dakle, sada možemo da ovo zapišemo nešto opštije ako je kodna riječ **c** i ako joj je dodat vektor pogreške **e** koji ima jedan nenulti elemenat dobijeni sindrom je:

$$\mathbf{r}\mathbf{H}^T = (\mathbf{c} + \mathbf{e})\mathbf{H}^T = \mathbf{c}\mathbf{H}^T + \mathbf{e}\mathbf{H}^T = \mathbf{e}\mathbf{H}^T$$

Ako je težina pogreške jednaka 1 onda je sindrom jednak koloni matrice **H** i dekodiranje poruke se obavlja tako što se od kodne riječi **x** na odgovarajućoj poziciji oduzme 1 dok ako je jednak umnošku kolone matrice **H** onda se oduzima taj umnožak sa odgovarajuće pozicije. U našem slučaju (slučaju ternarnog koda) oduzimanje 1 je ekvivalentno sabiranju sa 2 odnosno oduzimanje 2 je jednako sabiranju sa 1.

Sad možemo i da razumijemo način na koji se dobija kontrolna matrica (a ne da je dajemo ad-hok). Jasno je da se u kontrolnoj matrici ne može dva puta pojaviti ista kolona ali ni njen umnožak. Pretpostavimo da imamo dva informaciona bita. Polazna matrica ima dvije vrste ostaje da se ispita koliko nam treba kolona za provjere "parnosti":

$$\begin{bmatrix} ? & 1 & 0 \\ ? & 0 & 1 \end{bmatrix}$$

Očigledno se sada u kontrolnoj matrici ne mogu pojaviti kolone $[2\ 0]^T$ i $[0\ 2]^T$ jer su to umnošci već dobijenih kolona. Prva naredna moguća kombinacija je $[1\ 1]^T$ koja eliminiše kombinaciju $[2\ 2]^T$. Konačno preostaje još samo kombinacija $[1\ 2]^T$ koja eliminiše kombinaciju $[2\ 1]^T$. Dobijena kontrolna matrica je:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{bmatrix}$$

Ali jednako njoj Hammingov ternarni kod (4,2) može da ima i kontrolnu matricu:

$$\mathbf{H} = \begin{bmatrix} 2 & 1 & 2 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

Jednostavno rečeno kod ternarnog koda postoje dva konkurenta za svaku kolonu kontrolne matrice a ne treba zaboraviti ni da kad se dobije kontrolna matrica možemo izmiješati njene kolone na bilo koji pogodan način. Postavlja se pitanje koje dimenzije moraju biti ovakvog koda. Pretpostavimo da imamo *r*-arni Hammingov kod sa *k* informacionih simbola (teško sada više možemo govoriti o bitovima). Broj kontrolnih bita je u ovom slučaju *n*-*k* a toliko je i vrsta kontrolne matrice. Postavlja se pitanje kolika nam dužina kodne riječi treba da bi izvršili jedno ispravljanje pogreške. Očigledno po prethodnom izlaganju da svaka kolona matrice eliminiše *r*-2 mogućih kolona matrice koja predstavljaju umnoške date kolone sa brojevima 2, ..., *r*-1 (množenje sa 1 daje samu tu kolonu dok množenje sa 0 daje kolonu koja je neupotrebljiva kod korekcije pogreški). Sa *r* simbola ukupno se može kreirati r^{n-k} različitih kolona u matrici koja ima *n*-*k* vrsta. Jedna kolona sa svim nulama je neupotrebljiva pa stoga prilikom konstrukcije kontrolne matrice mi možemo koristiti r^{n-k}-1 kolona. Kada odaberemo jednu kolonu mi eliminišemo još *r*-2 kolona stoga je maksimalan broj kolona matrice (odnosno broj bita kontrolnih + informacionih) jednak:

$$n = \frac{r^{n-k} - 1}{r - 1}$$

Provjera na poznatim slučajevima koda sa *r*=2 i recimo *n*-*k*=3 daje *n*=7 (Hammingov kod (7,4)) dok recimo *r*=3 i *n*-*k*=2 daje *n*=4 (ternarni Hammingov kod (4,2) koji je korišten kao primjer). Napominjemo da u gornjoj relaciji može da stoji znak ≤. Naime, za datu redundanciju *n*-*k* možemo da imamo maksimalno *k* informacionih simbola ili maksimalno *n* ukupnih simbola. Kodovi koji zadovoljavaju ovu relaciju sa jednakošću nazivaju se perfektnim. Sada se postavlja pitanje kako recimo na osnovu poznatog *k* i *r* doći do *n*. To je zapravo problem koji nas interesuje jer obično znamo koliko želimo da šaljemo informacionih bita (to je *k*) kao i koji kod koristimo (to je *r*). Postoje različiti pristupi ali se uglavnom svode na malo numerike.

Naredni problem sa kojim se moramo suočiti je da li postoji način da se ovakav tip koda generiše na osnovu odgovarajuće generatorske matrice **G**. Generatorska matrica mora da ima dimenzije *k*x*n* i

mora da zadovoljava da je $\mathbf{GH}^{\mathrm{T}}=\mathbf{0}$. Po analogiji sa binarnim kodovima generatorsku i kontrolnu matricu možemo zapisati kao:

$$\mathbf{G}=[\mathbf{I}_k \mid \mathbf{R}] \qquad\qquad \mathbf{H}=[\mathbf{P}^T \mid \mathbf{I}]$$

gdje smo uveli novu matricu **R** u generatorsku matricu (napominjemo da kod binarnih kodova imamo matricu **P** na ovom mjestu). Jasno je da mora da važi:

$$\mathbf{GH}^T=\mathbf{P}+\mathbf{R}=\mathbf{0}$$

Dakle, u aritmetici sa *r*-arnim kodom **R**=-**P** odnosno ovo se može zapisati (pomalo u MATLAB notaciji) kao **R**=*r*-**P**. Npr. za posmatrani ternarni kod to se svede na:

$$\mathbf{H}=\begin{bmatrix}1&1&1&0\\2&1&0&1\end{bmatrix} \qquad \mathbf{G}=\begin{bmatrix}1&0&2&1\\0&1&2&2\end{bmatrix} \qquad \mathbf{GH}^T=\begin{bmatrix}1&0&2&1\\0&1&2&2\end{bmatrix}\begin{bmatrix}1&2\\1&1\\1&0\\0&1\end{bmatrix}=\begin{bmatrix}0&0\\0&0\end{bmatrix}$$

9.2. Nebinarni Hammingovi kodovi – Definicija preko polinoma

Nebinarni Hammingovi kodovi se mogu uvesti i preko prostih polinoma:

$$\sum_{i=0}^{Q} a_i x^i$$

čiji koeficijenti uzimaju vrijednosti iz skupa $a_i \in \{0,1,...,r-1\}$. Stepen ovih polinoma odgovara broju provjera "parnosti" (*r*-arnosti). Radi jednostavnosti posmatrajmo ternarni kod sa 2 provjere parnosti. Postavlja se pitanje koji su prosti polinomi reda koji je manji ili jednak sa *Q*. Prosti polinomi prvog reda su: 1, *x*, *x*+1, *x*+2, ali recimo nijesu 2*x*+1 jer podijeljen sa 2 daje *x*+2 za kojeg smo usvojili da je prost. Po istoj logici možemo reći da polinom *x*+2 nije prost jer 2(2*x*+1)=*x*+2. Da bi sebi olakšali život usvojimo da prost polinom mora imati koeficijenat uz član najvišeg reda jednak 1 (ovakvi polinomi se nazivaju monični) i ne provjeravajmo dijeljenja sa konstantom. Broj mogućih *r*-arnih polinoma drugog reda sa koeficijentom uz član najvišeg reda 1 je r^2. U slučaju ternarnog koda ti polinomi su:

x^2 (nije prost jer je djeljiv sa x)	x^2+1 (prost! što nije slučaj kod binarnog koda)
x^2+2 (nije prost djeljiv sa $x+1$ i $x+2$)	x^2+x (nije prost, jer je djeljiv sa x i $x+1$)
x^2+x+1 (nije prost $=(x+2)^2$)	x^2+x+2 (prost)
x^2+2x (nije prost $=x(x+2)$)	x^2+2x+1 (nije prost $=(x+1)^2$)
x^2+2x+2 (prost)	

Detektovali smo (prostom pretragom) 3 prosta polinoma traženog oblika. Međutim, nijesu svi prosti polinomi pogodni za kodiranje Hammingovog koda. Ako se sjećate izvođenja kod binarnih kodova potrebno je da stepeni nule prostog polinoma generatora koda a^l budu jednaki $a^0=1$ za $l=n$ gdje je n dužina kodne riječi ali ne i prije $l=n$. Stoga otpada polinom x^2+1 jer je sa njime djeljiv x^l+1 za $l=2<n$. Preostaju samo dva prosta polinoma koja se mogu upotrijebiti za datu namjenu oba zadovoljavaju osobinu da ni x^2+1 ni x^3+1 nije djeljivo sa njima bez ostatka dok x^4+1 je djeljivo što ćemo pokazati na primjeru polinoma x^2+x+2 (isto važi i za polinom x^2+2x+2).

x^4+1 : $x^2+x+2 =$ x^2
$-(x^4+x^3+2x^2)$
$\underline{-x^3-2x^2+1=2x^3+x^2+1}$: $x^2+x+2 =$ $2x$
$-(2x^3+2x^2+4x)$
$\underline{-x^2-4x+1=2x^2+2x+1}$: $x^2+x+2 =$ 2
$-(2x^2+2x+4)$
1-4=-3=0

Dakle: $x^4+1:x^2+x+2=x^2+2x+2$ i to bez ostatka (ujedno smo pokazali i da je dati polinom djeljiv i sa x^2+2x+2. Povežimo sada dobijene kontrolne matrice sa polinomom. Neka je α nula prostog polinoma. U kontrolnoj matrici upisujemo stepene nule prostog polinoma usvajajući da je:

$$\alpha^0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \alpha^1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Svi ostali stepeni se mogu opisati preko ova dva stepena:

$$\alpha^2 = -\alpha - 2 = 2\alpha + 1 = 2\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$\alpha^3 = 2\alpha^2 + \alpha = 2\begin{bmatrix} 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

Stoga smo dobili kontrolnu matricu:

$$\mathbf{H} = \begin{array}{c} \alpha^3 \ \ \alpha^2 \ \ \alpha^1 \ \alpha^0 \\ \begin{bmatrix} 2 & 2 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{bmatrix} \end{array}$$

Posmatrajmo sada sledeći stepen:

$$\alpha^4 = \alpha^3 \cdot \alpha = (2\alpha + 2)\alpha = 2\alpha^2 + 2\alpha = 4\alpha + 2 + 2\alpha = 2 = 2\alpha^0$$

Ovdje sada vidimo da kod nebinarnih kodova ne važi $\alpha^n = \alpha^0$ kao kod binarnih kodova već je ovdje situacija nešto složenija odnosno važi: $\alpha^n = 2\alpha^0 = (r-1)\alpha^0 = -\alpha^0$. U opštem slučaju kod nebinarnih kodova važi ova relacija dok ona važi i kod binarnih kodova jer je kod binarnih kodova 1 i -1 isti broj odnosno sabiranje i oduzimanje su ista operacija.

Kodiranje preko polinoma se obavlja množenjem informacionog polinoma sa generatorskim polinomom koda. Neka je informaciona riječ u našem slučaju [2 1] ona se može prikazati preko polinoma kao $\mathbf{i}(x)=2x+1$. Kodna riječ je sada:

$$\mathbf{c}(x)=\mathbf{i}(x)\mathbf{g}(x)=(2x+1)(x^2+x+2)=2x^3+3x^2+5x+2=2x^3+0x^2+2x+2$$

Dobijena kodna riječ je [2 0 2 2]. Dekodiranje putem polinoma se može obaviti dijeljenjem sa generatorskim polinomom. Ako je dobijen rezultat bez ostatka tada je dobijeni polinom količnik informacija koja je bila poslata. U slučaju da dobijemo ostatak pri dijeljenju moramo odrediti kojem stepenu korjena prostog generatorskog polinoma odgovara dobijeni ostatak. Na primjer, neka je

primljeni polinom: $2x^3+2x^2+2x+2$. Obavimo proceduru njegovog dijeljenja sa generatorskim polinomom:

$2x^3+2x^2+2x+2$: x^2+x+2 = $2x$
$-(2x^3+2x^2+4x)$
$x+2$

Znači dobijeni ostatak je $x+2$. Ovo odgovara vektor koloni $[1\ 2]^T$. Iz kontrolne matrice mi lako uočavamo da je u pitanju umnožak druge kolone matrice (koja odgovara stepenu prostog polinoma α^2) $[1\ 2]^T = 2[2\ 1]^T$ na osnovu čega zaključujemo da se pogreška dogodila na drugoj poziciji i da je težina greške 2. U praksi do ovog zaključka se ne dolazi tako lako zato što se zbog dimenzija kontrolna i generatorska matrica izbjegavaju. Umjesto toga se polinom ostatak množi sa α^{-1} što je u datom polju ekvivalentno sa množenjem sa $-\alpha^{n-1}$. U našem slučaju je $\alpha^3=2\alpha^2+\alpha=4\alpha+2+\alpha=2\alpha+2$ pa je $-\alpha^3=\alpha+1$. Pomnožimo ovaj polinom sa našim ostatkom $(\alpha+2)(\alpha+1)= \alpha^2+2= (2\alpha+1)+2=2\alpha$. Odavde zaključujemo da se greška dogodila na α^2 stepenu a da je težina pogreške 2 odnosno da treba oduzeti dvoju na poziciji x^2 u polinomu.

9.3. Nebinarni kodovi koji su u stanju da isprave više od jedne pogreške

Najpoznatiji nebinarni kod koji je u stanju da ispravi više od jedne pogreške je Golayev ternarni (11,6) kod koji je u stanju da ispravi dvije pogreške. Ovaj polinom je definisan na polju u kojem važi $x^{11}=1$ sa generatorskim polinomom $f(x)=x^5+x^4-x^3+x^2-1=x^5+x^4+2x^3+x^2+2$. Golay je ovaj kod otkrio 1949-te i do danas je ovo jedini poznat perfektan nebinarni kod koji je u stanju da dekodira više od jedne pogreške! Golay je inače razvio i binarni perfektni kod (23,11) koji je u stanju da ispravi do tri pogreške. Kontrolna i generatorska matrica ovog koda se obično daju u sljedećoj formi:

$$\mathbf{H}=\begin{bmatrix} 1 & 2 & 2 & 2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 2 & 2 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 2 & 2 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 2 & 2 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 2 & 2 & 1 & 0 & 1 \end{bmatrix} \mathbf{G}=\begin{bmatrix} 2 & 0 & 1 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 1 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 & 2 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 1 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 1 & 2 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 & 2 & 1 & 1 \end{bmatrix}$$

Golayevi kodovi imaju i veoma interesantnu istoriju pošto ih je uveo Juhani Virtakallio za potrebe zapisa rezultata fudbalskih utakmica pa ih je tek nakon dvije godine "otkrio" Golay 1949. te godine.

Alternativno Golayevom kodu mogu se koristiti i varijante BCH kodova vodeći računa samo o činjenici da je prostor odnosno algebra koja se provodi različita od binarne po modulu 2 aritmetike. Realizacija BCH kodova se obavlja na potpuno isti ili veoma sličan način kao kod binarnih kodova. Osnovni problem se ipak ogleda u činjenici što ne postoje BCH kodovi koji su perfektni tako da izbor pravog prostog polinoma može da bude veoma složen i problematičan. Poseban problem što se teško dizajniraju kodovi koji su relativno kratki i upotrebljivi za ispravljanje većeg broja pogreški. Naravno ovo ne znači da nema nebinarnih BCH kodova koji su veoma upotrebljivi već samo da ih ovdje nećemo navoditi jer je teško doći do nekog relativno kratkog BCH koda sa malim r na kome bi demonstrirali kako se sa ovim kodovima radi. Pored ovoga koriste se nebinarni Reed Solomonovi (RS) kodovi. Ovi kodovi su generalizacija BCH kodova te je kodiranje i dekodiranje gotovo potpuno isto kao u slučaju BCH kodova (posebno u dijelu euklidskog algoritma). Osnovna razlika između RS i BCH kodova je u alfabetu odnosu polju koeficijenata polinoma – koda preko kojega su kodovi definisani.

Zadaci za vježbu

9.1. Formirati kontrolnu matricu koda sa r=5 simbola. U kodnoj riječi ima n-k=2 provjera parnosti.

Rješenje. Kontrolna matrica koja predstavlja jedno moguće rješenje postavljenog problema je:

$$\mathbf{H} = \begin{bmatrix} 4 & 3 & 2 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Prvo treba provjeriti da li je ovakav kod može da postoji. Kao što smo vidjeli između dužine kodne riječi, broja informacionih simbola i veličine alfabeta važi sledeća relacija:

$$n = \frac{r^{n-k} - 1}{r - 1}$$

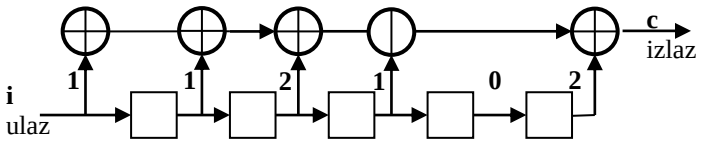
Uvrštavanjem gorenavedenih parametara dobijamo da je n=6 pa je zatim kontrolna matrica dimenzija (n-k)xn=2x6 baš kao što je u ovom slučaju naša **H** matrica. Predmetna matrica predstavlja kontrolnu matricu koda a generatorska se može dobiti kao:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 4 \\ 0 & 1 & 0 & 0 & 2 & 4 \\ 0 & 0 & 1 & 0 & 3 & 4 \\ 0 & 0 & 0 & 1 & 4 & 4 \end{bmatrix}$$

Postavlja se pitanje kakve su karakteristike ovog koda u pogledu mogućnosti za ispravljanje pogreški. To pitanje uz primjer realizacije vam prepuštamo.

9.2. Kreirati hardversku strukturu koja kodira Golayev kod. Zatim dokazati da je ovaj kod cikličan. Prikazati primjer dekodiranja ovog koda. Prikazati sistematičniji postupak dekodiranja i dokazati da je kod cikličan.

Rješenje. Posmatrajmo generatorski polinom: $f(x)=x^5+x^4+2x^3+x^2+2$. Hardver koji realizuje kodiranje ovog koda je dat na slici dolje sa time da treba imati na umu da se u ovom slučaju množenja i sabiranja obavljaju po modulu 3.



Kao što smo već pokazali ranije kod je cikličan ako je polinom x^n-1 (u našem slučaju x^{11}+2) djeljiv sa generatorskim polinomom:

$x^{11}+2 : x^5+x^4+2x^3+x^2+2 = x^6+2x^5+2x^4+2x^3+x^2+1$
$x^{11}+x^{10}+2x^9+x^8+2x^6$

$2x^{10}+x^9+2x^8+x^6+2$
$2x^{10}+2x^9+x^8+2x^7+x^5$

$2x^9+x^8+x^7+x^6+2x^5+2$
$2x^9+2x^8+x^7+2x^6+x^4$

$2x^8+2x^6+2x^5+2x^4+2$
$2x^8+2x^7+x^6+2x^5+x^3$

$x^7+x^6+2x^4+2x^3+2$
$x^7+x^6+2x^5+x^4+2x^2$

$x^5+x^4+2x^3+x^2+2$

Pošto smo dobili djeljivost bez ostatka možemo zaključiti da je u pitanju ciklični kod.

Izvršimo sada demonstraciju funkcionisanja Golayevog koda. Uzmimo da je riječ koju treba kodirati:

[0, 1, 1, 2, 0, 2]

Putem generatorske matrice dobijamo kodnu riječ:

$$[0,\ 1,\ 1,\ 2,\ 0,\ 2]\begin{bmatrix} 2 & 0 & 1 & 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 1 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 & 2 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 1 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 1 & 2 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 & 2 & 1 & 1 \end{bmatrix} = [0\ 2\ 2\ 2\ 0\ \ 0\ 0\ 2\ 0\ 2\ \ 2]$$

Provjerimo da li je sada u pitanju korektna kodna riječ na osnovu kontrolne matrice.

$$\mathbf{cH}^T = [0\ \ 0\ 0\ 0\ 0]$$

Čime smo pokazali da kod radi korektno u uslovima kada nema pogreški u kodu. Posmatrajmo sada slučaj kada postoje dvije pogreške (stanje kada ima samo jedna pogreška je relativno jednostavno jer podrazumijeva da je sindrom jednak koloni ili umnošku neke kolone).

Neka je na primjer primljena kodna riječ:

$$[0\ 2\ 2\ \mathbf{1}\ 0\ \ 0\ 1\ 2\ 0\ 2\ \ 2]$$

Greške su se dogodile na četvrtom i na sedmom simbolu. Sada možemo da odredimo rezultat množenja kodne riječi sa kontrolnom matricom kako bi odredili sindrom:

$$[0\ 1\ 0\ 0\ 1]$$

Pošto dobijena vrijednost nije jednaka nijednoj koloni niti umnošku ijedne kolone to onda može da ima dvije pogreške. Ova situacija se može ispitati provjerom i svođenjem mogućih kombinacija koji je 11x10/2=55 ali svaka kombinacija se može pojaviti sa dvije moguće težine. Nula kao prva vrijednost sindromskog vektora i 1 kao druga vrijednost sindromskog vektora se svodi na sljedeće moguće kombinacije: [1,2] (1,1), [1,3] (2,2), [1,4] (2,2), [1,5] (1,2), [2,3] (2,1), [2,4] (2,1), [2,7] (1,1), [3,5] (1,1), [3,7] (2,2), [4,5] (1,1), [4,7] (2,2), [5,7] (2,1), [6,8] (2,2), [6,9] (1,*), [6,10] (1,*), [6,11] (1,*), [8,9] (1,*), [8,10] (1,*) i [8,11] (1,*) gdje srednje zagrade označavaju moguće pozicije pogreške dok male zagrade označavaju težine pogreške (* znači da može biti bilo 1 bilo 2). Dakle imamo 20 kombinacija od kojih sedam sa dvije moguće težine. Slijedi provjera treće pozicije na kojoj je sindromski vektor jedna nuli. Od kombinacija koje smo prethodno dobili ovo zadovoljavaju samo: [1,2] (1,1), [3,5] (1,1), [4,7] (2,2), [6,9] (1,1), [8,10] (1,*) i [8,11] (1,*). Sveli smo pretragu na šest parova pozicija pogreške od kojih četiri imaju jedinstvene težine a dvije imaju mogućnost za dvije različite kombinacija težina pogreške. Sljedeća težina u sindromu je 0 nakon koje nam preostaju samo tri kombinacije [1,2] (1,1), [4,7] (2,2) i [8,10] (1,2). Konačno peti bit sindroma je 1 a to nam dalje svodi moguće kombinacije [4,7] (2,2) što je tačno jer se tačna vrijednost poruke može dobiti tako što se na pozicijama 4 i 7 od dobijene poruke oduzmu težine 2.

Postoji i sistematičniji način da se izvrši dekodiranje Golayevih kodova koji je upitan zbog činjenice da je ovaj kod relativno kratak i da se može napraviti program koji sve kombinacije mogućih pogreški provjeri veoma brzo. Ovdje dajemo osnovne elemente algebra vezane za dekodiranje ovog Golayevog koda. Pretpostavimo da su se pogreške dogodile na pozicijama i i j koje odgovaraju nulama prostog polinoma stepena a^i i a^j. Stoga se sindrom može zapisati kao:

$$S_1=ua^i+va^j$$

gdje su u i v težine grešaka. Ovo očigledno nije dovoljno da bi se sistem riješio (posjedujemo 4 nepoznate u, v, i i j) pa stoga uvodimo još jedan sindrom:

$$S_5=ua^{5i}+va^{5j}$$

Kao što smo očekivali potreban je sindrom sa neparnim stepenima nule prostog polinoma u ovom slučaju se pokazuje da je taj stepen koji nam odgovara peti ali detalje nećemo iznositi. Sada uvedimo smjene $y_1=ua^i$ i $y_2=va^j$ uz činjenicu da je u algebri po modulu 3 $0^5=0$, $1^5=1$ i $2^5=32\%3=2$ odnosno $u^5=u$:

$$S_1=y_1+y_2$$
$$S_5 = y_1^5 + y_2^5$$

U slučaju pojave dvostruke pogreške pokazuje se da su korjeni rješenja kvadratne jednačine:

$$y^2 - \sigma_1 y + \sigma_2 = 0$$

gdje je $\sigma_1=y_1+y_2$ i $\sigma_2=y_1y_2$. Međutim, riječ je na neki način o vrzinom kolu pošto za određivanje navedene kvadratne jednačine je neophodno poznavati rješenja jednačine y_1 i y_2. Međutim, lako se pokazuje da je

$$S_1=\sigma_1$$
$$S_5 = S_1^5 + \sigma_2 S_1^3 - \sigma_2^2 S_1$$

uz napomenu da je -1=2 u ovoj algebri. Rješavanjem kvadratne jednačine dobijamo:

$$\sigma_2 = -S_1^2 \pm \sqrt{\frac{-S_5 - S_1^5}{S_1}}$$

Pošto postoje dva riješenja treba izabrati ono koje zadovoljava da je $\sigma_2^{11} = \pm 1$. Konačno jednačina posjeduje dva rješenja:

$$y_{1,2} = -S_1 \pm \sqrt{S_1^2 - \sigma_2}$$

Težine pogreške se mogu sračunati kao:

$$u = y_1^{11} \text{ i } v = y_2^{11}$$

Dalje se pozicije mogu dobiti (u diskretnom obliku):

$$i = \log_a(y_1/u) \quad j = \log_a(y_2/v)$$

9.3. Pokušajte da na Internetu ili drugim izvorima pronađene podatke o nebinarnim BCH kodovima i proceduri njihovog dekodiranja.

Poglavlje X
KONVOLUCIONI KODOVI

10.1. Razlozi za kreiranje konvolucionih kodova

Po drugoj Šenonovoj teoremi kodna riječ bi morala da bude što je moguće duža, da bi se za date parametre blok koda (i kanala) postigla vjerovatnoća greške pri dekodiranju koja teži nuli. Ovo bi usložilo koder i dekoder i često predstavlja neprihvatljivu strategiju, pa umjesto da kodiramo kodovima beskonačne dužine, mi se odlučujemo za odstupanje od Šenonove teoreme odnosno za kodove koji produkuju određenu vjerovatnoću pogreške. Stoga se umjesto kreiranju blok kodova često pristupa i kreiranju **konvolucionih** kodova. Konvolucioni kodovi se nazivaju i **sistematski**, ako se informacioni biti pojavljuju direktno u sekvenci na odgovarajućim pozicijama. Konvolucioni kodovi se koriste u radio i satelitskim linkovima, a djelovi su i nekih komunikacionih standarda za savremene mobilne sisteme uključujući WLAN. Posebno su na popularnost dobili kod sondi Voyager, kada su korišćeni konvolucioni kodovi sa kodnim odnosom R=1/7. Ovi kodovi su dio komunikacionog paketa za većinu aktuelnih svemirskih brodova uključujući Mars program. Danas se postepeno prevazilaze upotrebom turbo kodova, koji pokazuju još bolje preformanse. Međutim, u osjetljivim aplikacijama kao što su svemirska istraživanja, gdje kapacitet kanala može da padne i do 1/100 dobri se rezultati ne mogu postići pojedinačnom primjenom ni konvolucionih ni turbo kodova, već se ovi kodovi moraju kombinovati sa nekim kvalitetnim blok kodom. U ovim aplikacijama konvolucioni kodovi su neprevaziđeni, pošto postoje odlični algoritmi da se povežu sa Reed-Solomonovim kodovima (RS kodovi su generalizacija BCH kodova) što, u ovom trenutku, nije slučaj sa turbo kodovima. Kod konvolucionih kodova, kodni simboli na izlazu ne zavise samo od trenutne poruke koja se kodira, već mogu zavisiti i od nekoliko prethodnih kodnih simbola. Ovo dovodi do toga da se na lakši način (sa manjom dužinom kodne riječi) primičemo cilju druge Šenonove teoreme. Nažalost, kao što ćemo vidjeti kasnije, ovo se plaća složenošću procesa dekodiranja.

Konvolucija se može povezati sa određivanjem koeficijenata polinoma proizvoda kao:

$$c_n = \sum_{\nu} a_{\nu} b_{n-\nu}$$

Linearni (n,k) blok kod K određuje se nxk generatorskom matricom **G** nad poljem F_2. Konvolucioni kod (KK) (n,k) je takođe određen generatorskom matricom **G**, s tom razlikom što su elementi generatorske matrice g_{ij} polinomi nad poljem F_2. Možemo zaključiti da su svi linearni blok kodovi zapravo specijalni slučajevi konvolucionih kodova. Posmatrajmo primjere dva konvoluciona koda:

$\mathbf{G} = [x^2+1, x^2+x+1]$ je generatorska matrica KK (2,1)

Generatorska matrica KK (3,2) je:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & x+1 \\ 0 & 1 & x \end{bmatrix}$$

Osnovne karakteristike KK su:

memorija: $M = \max_{ij}[\deg(g_{ij})]$ (deg označava stepen polinoma)

ograničena dužina: $N = M + 1$

Kodna (prenosna) brzina: $R=k/n$ (gdje su kxn dimenzije matrice **G**).

Prethodna dva koda imaju respektivno: M=2, N=3, R=1/2
M=1, N=2, R=2/3

Da bi se polinomska matrica **G** upotrijebila za zaštitno kodiranje informacioni biti se moraju preslikati u koeficijente *k*-torki polinoma $\mathbf{i}=[i_0(x),i_1(x),...,i_{k-1}(x)]$. Kodna riječ $\mathbf{c}=[c_0(x),c_1(x),...$ $c_{n-1}(x)]$ će biti određena kao:

$$\mathbf{c}=\mathbf{iG}$$

Neka je data prva matrica konvolucionog koda i neka je $\mathbf{i}=[x^3+x+1]$ će biti kodirana kao: $\mathbf{c}=[x^5+x^2+x+1,x^5+x^4+1]$ dok za informaciju: $\mathbf{i}=[x^2+x,x^3+1]$ dobijamo polinomijalnu kodnu riječ $\mathbf{c}=[x^2+x,x^3+1,x^4+x^2]$. Ovo se obavlja kao klasično matrično množenje:

$$\mathbf{c}=[x^2+1,x^3+1]\begin{bmatrix}1 & 0 & 1+x\\ 0 & 1 & x\end{bmatrix}=[x^2+1,x^3+1,(x+1)(x^2+1)+x(x^3+1)]=$$
$$=[x^2+1,x^3+1,x^4+x^3+x^2+1]$$

Najpogodniji način prikaza polinomne kodne riječi $\mathbf{c}=[c_0(x),c_1(x),...,c_{n-1}(x)]$ je preko polinomnih koeficijenata. Tako se *j*-ti polinom može napisati: $c_j(x)=c_{j0}+c_{j1}x+...$ pa iz ovoga slijedi skalarni prikaz kodne riječi kao:

$$\mathbf{c}=[c_{00}c_{10}...c_{n-1;0},c_{01}c_{11}...c_{n-1;1},...]$$

Matrica **G** se može napisati i u pogodnijoj (ponekad se naziva skalarnoj) formi kao

$$\mathbf{G}=\begin{bmatrix}G_0 & G_1 & G_2 & ... & G_M & 0 & 0\\ 0 & G_0 & G_1 & ... & G_{M-1} & G_M & 0\\ 0 & 0 & G_0 & ... & G_{M-2} & G_{M-1} & G_M\\ ... & ... & ... & ... & ... & ... & ...\\ 0 & 0 & 0 & ... & ... & ... & ...\end{bmatrix}$$

gdje su koeficijenti u ovoj matrici mogu odrediti na osnovu relacije:

$$\mathbf{G}=\sum_{i=0}^{M}G_i x^i$$

Na primjer, generatorska matrica za prvi uvedeni kod se može zapisati kao:

$$\mathbf{G}=[1,1]+[0,1]x+[1,1]x^2$$

odakle slijedi oblik matrice:

$$\mathbf{G}=\begin{bmatrix}11 & 01 & 11 & & & \\ & 11 & 01 & 11 & & \\ & & 11 & 01 & 11 & \\ & & & 11 & 01 & 11\\ & & & & \downarrow & \rightarrow\end{bmatrix}$$

Pretpostavimo da je informacija data kao $\mathbf{i}=[x^3+x+1]$. Ovo se u skalarnoj formi zapisuje kao [1101] odnosno koeficijenti se pišu počevši od koeficijenta uz polinom najmanje težine (nije [1011]). Dobijena kodna riječ se može zapisati kao $\mathbf{c}=[x^5+x^2+x+1,x^5+x^4+1]$. Skalarni zapis ove kodne riječi glasi [111010000111]. Ovo se jednostavno dokazuje množenjem vektora **i** sa matricom **G** (uz odgovorajuće ograničenje dužine ove matrice):

$$\mathbf{c} = [1\,1\,0\,1]\begin{bmatrix} 11 & 01 & 11 & & & \\ & 11 & 01 & 11 & & \\ & & 11 & 01 & 11 & \\ & & & 11 & 01 & 11 \end{bmatrix}$$

Za slučaj druge generatorske matrice slijedi:

$$G = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} x$$

i odgovarajuća skalarna generatorska matrica za konvolucioni kod je:

$$\mathbf{G} = \begin{bmatrix} 101 & 001 & & \\ 010 & 001 & & \\ & 101 & 001 & \\ & 010 & 001 & \\ & & \downarrow & \rightarrow \end{bmatrix}$$

Skalarna informacija pridružena polinomnoj informaciji $\mathbf{i} = [x^2 + x, x^3 + 1]$ je [01101001] i skalarna kodna riječ pridružena polinomnoj kodnoj riječi $\mathbf{c} = [x^2 + x, x^3 + 1, x^4 + x^3]$ je [010100100011001].

Uočite da nijesmo diskutovali način izbora polinoma kod konvolucionih kodova. Obično se koriste prosti polinomi, ali uočite da neki od korištenih polinoma ne ispunjavaju neke od preduslova koje smo uveli kod određivanja polinoma za kodiranje kod Hammingovog i BCH koda. Naprosto, do danas nije pronađen jednostavan postupak da se opišu ovi polinomi, već se posebno kvalitetni polinomi određuju na osnovu simulacija. Dobri rezultati se obično publikuju u obliku tablica.

10.2. Konvolucioni kodovi sa ograničenjima

Stepen informacionog polinoma bi u principu trebao biti što je moguće veći, da bi se primakli uslovima kodne teoreme (za velike dužine koda moguće je kreirati kod koji ne daje grešku, a čija je kodni odnos blizak kapacitetu kanala), ali zbog raznih praktičnih razloga se moraju uvesti ograničenja. To nas vodi na definiciju L-tog ograničenja konvolucionog koda. Informacioni polinomi moraju zadovoljavati uslov $\deg[i_j(x)] \le L-1$, j=0, 1, ..., k-1. Na osnovu ovoga se dobija kodna riječ $\mathbf{c} = [c_0(x), c_1(x), ..., c_{n-1}(x)]$ čija svaka komponenta ima stepen manji od $M + L - 1$. Tako se informacija $\mathbf{i} = [i_0(x), i_1(x), ..., i_{k-1}(x)]$ može predstaviti pomoću kL bita, a kodnu riječ $\mathbf{c}$ pomoću $n(M+L)$ bita. Kodno preslikavanje se može prikazati u skalarnoj notaciji kao:

$$\mathbf{c} = \mathbf{i}G_L$$

	$M+L$ blokova od n bita							
	$\mathbf{G}_0$	$\mathbf{G}_1$	$\mathbf{G}_2$	...	$\mathbf{G}_M$	0	0	L blokova
	0	$\mathbf{G}_0$	$\mathbf{G}_1$	...	$\mathbf{G}_{M-1}$	$\mathbf{G}_M$	0	dužine k
G=	0	0	...		...	...	...	bita
	0	0	$\mathbf{G}_0$		$\mathbf{G}_{M-2}$	$\mathbf{G}_{M-1}$	$\mathbf{G}_M$	

Ovdje se vidi zašto se pomoću konvolucionih kodova sa osnovnom kodnom riječju koja je bitno kraća nego kod blok kodova, postiže približavanje performansama koje predviđa II Shannonova teorema. Naime, u slučaju konvolucionog koda imamo činjenicu da trenutni izlaz zavisi od određenog broja prethodnih informacionih poruka, što nije slučaj kod blok kodova. L-to ograničenje konvolucionog koda (n,k) možemo razmatrati kao $(n(M+L),kL)$ linearni blok-kod i u tom smislu su

konvolucioni kodovi posebna vrsta blok-kodova (mada se blok kodovi na neki način mogu posmatrati kao specijalni slučaj konvolucionih kodova sa samo jednim specijalno konstruisanim kodnim polinomom). Kodni odnos (kodna brzina) ograničenog koda je data kao:

$$R_L = \frac{kL}{n(M+L)} = R\left(1 - \frac{M}{M+L}\right)$$

gdje je R kodni odnos neograničenog konvolucionog koda. Kako u mnogim aplikacijama uzimamo L mnogo veće od M, to će brzina ograničenog koda biti bliska brzini neograničenog koda $R_L \approx R$.

Ako za primjer 1 uvedemo ograničenje L=6 dobijamo (16,6) linearni blok kod koji ima generatorsku matricu (uočite da nulte elemente izostavljamo).

$$\mathbf{G}_6 = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & & & & & & & & \\ & & 0 & 1 & 0 & 1 & 1 & 1 & & & & & & \\ & & & & 1 & 1 & 0 & 1 & 1 & 1 & & & & \\ & & & & & & 1 & 1 & 0 & 1 & 1 & 1 & & \\ & & & & & & & & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Za drugi kod koji je usvojen kao primjer i ograničenje L=2 dobijamo linearni kod sa generatorskom matricom:

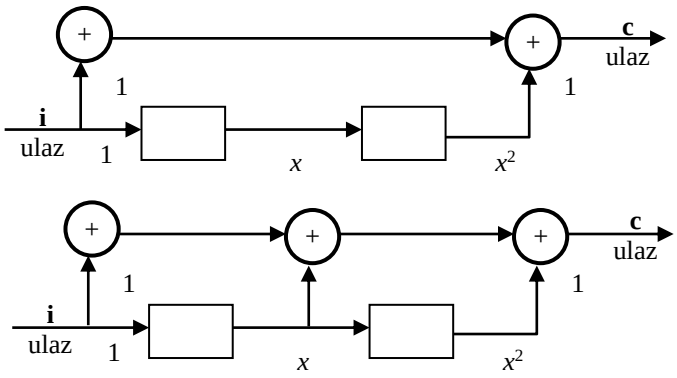
$$\mathbf{G}_2 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & & & \\ 0 & 1 & 0 & 0 & 0 & 1 & & & \\ & & & 1 & 0 & 1 & 0 & 0 & 1 \\ & & & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

10.3. Realizacija konvolucionih kodova

U slučaju prvog koda $\mathbf{i} = [i_0, i_1, ...]$ na ulazu dobijeni kodni niz zapisan i proizvodi kodni redosljed $\mathbf{c} = [c_{00}, c_{01}, c_{10}, c_{11}, ...]$. Odnos između informacionog niza **i** i kodirane povorke se može prikazati kao:

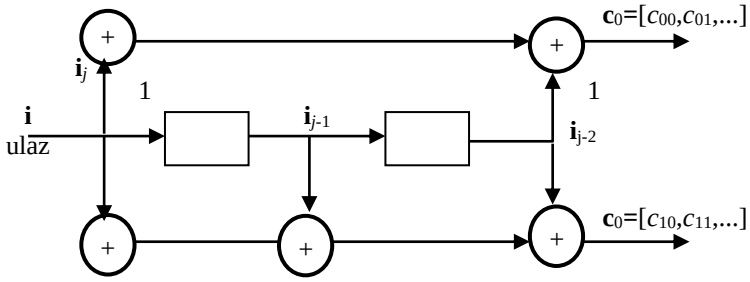
$$\mathbf{c}_0(x) = c_{00} + c_{01}x + ... = (x^2+1)(i_0 + i_1 x + ...) = (x^2+1)\mathbf{i}(x)$$
$$\mathbf{c}_1(x) = c_{10} + c_{11}x + ... = (x^2 + x + 1)\mathbf{i}(x)$$

Realizacija ovog koda se može obaviti preko sledećih sklopova



$$c_{0j} = i_j + \quad + i_{j-2} \qquad c_{1j} = i_j + i_{j-1} + i_{j-2} \qquad j = 0,1,...$$

Ova dva sklopa se mogu objediniti u jedan sa pomjeračkim registrom:

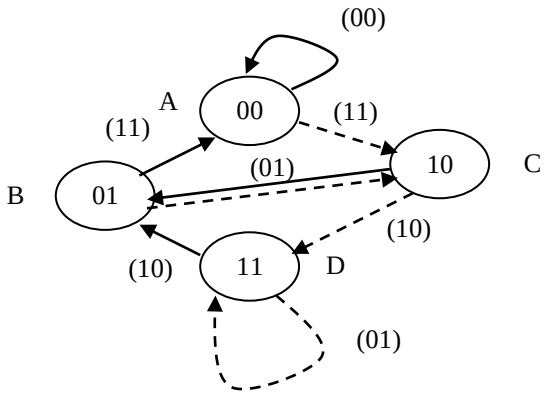


Pošto izlazi zavise od tekućeg i prethodna dva bita to znači da sa ova tri bita mora raspolagati koder da bi mogao da kreira izlaz. Memorija je M=2, a kako tekući bit nije potrebno pamtiti, to znači da je dužina koda $N=M+1$. Važna veličina kod konvolucionih kodova je *ukupna ograničena dužina* koja se definiše kao $V=K(M+1)$. Kako jedan informacioni bit utiče na najviše V bita, ova se veličina naziva i domašaj. Očigledno je da se za svaki bit sa ulaza proizvedu dva bita na izlazu, pa je brzina koda (odnosno kodni odnos) R=1/2 (uostalom, ovo smo uočili odmah prilikom definisanja konvolucionih kodova). Za opšti kod sa pomjeračkim registrom (n,k) nam je potrebno k pomjeračkih registara, svaki za jedan ulazni informacioni niz i_0, i_1,..., i_{k-1} pa će i-ti pomjerački registar informacijskog niza sa n polinoma biti $g_{i0}(x)$, $g_{i1}(x)$, ..., $g_{i,n-1}(x)$, pa se izlazni niz na j-tom izlazu dobija sabiranjem j-tog izlaza sa svakog od pomjeračkih registara.

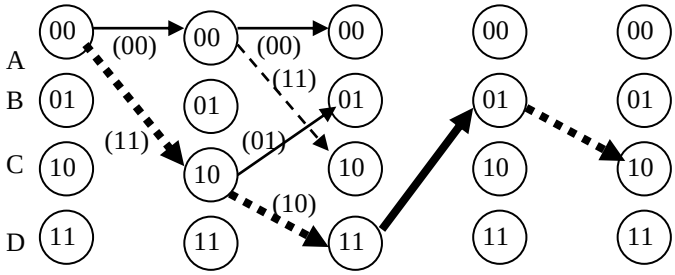
-Postoje varijante konvolucionih kodova koji se nazivaju "punctured" (probušeni ili redukovani) kodovi kod kojih se konstruiše koder određenog kodnog odnosa R. Ako se na osnovu odgovarajućeg snimanja utvrdi da su uslovi u kanalu povoljniji, može se kodni odnos popraviti tako što se neki od bitova koje bi inače produkovao koder ne šalju uopšte u kanal. Srećna okolnost što je softverska realizacija odbacivanja dijela kodnih bita, kao i prilagođavanje dekodera ovoj situaciji, relativno jednostavno.

10.4. Grafički prikaz kodiranja konvolucionim kodom

Prikažimo ovo i preko dijagrama stanja. U čvorovima dijagrama se nalaze kombinacije i_{j-1} i i_{j-2}. Nakon primanja novog bita i_j koder se pomjera u stanje i_j i_j-1, dok se na izlazu kreiraju dva izlazna bita c_{0j} i c_{1j}. Prelazi koji odgovaraju bitu 0 su prikazani sa punim linijama, dok su oni koji odgovaraju jedinici prikazani crtkanim linijama. Oznake u zagradama predstavljaju odgovarajuće bitove kodne riječi koja je produkovana.

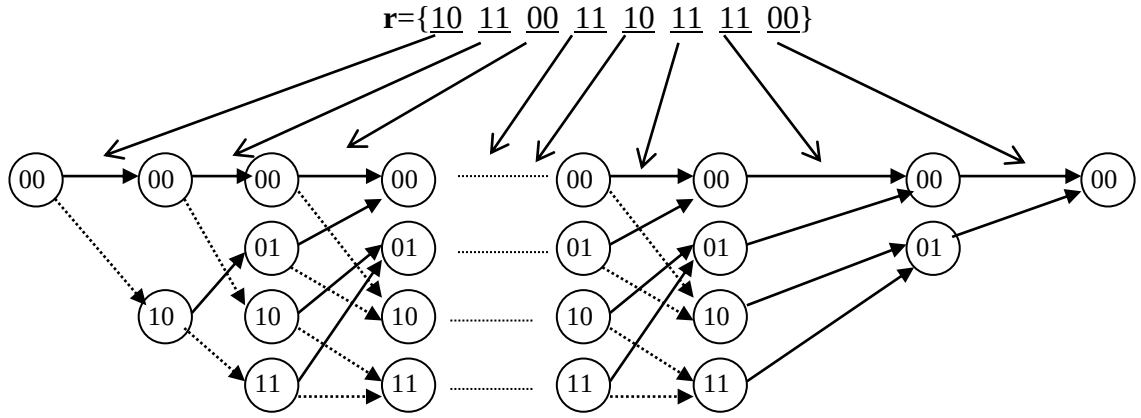


Sada se lako može učiti da za ulazni informacioni kod [110100] dobijamo put ACDBCBA odnosno izlazni kod [111010000111]. Dijagram stanja je često nepregledan, pa se stoga mnogo češće koriste rešetkasti dijagrami (kodne rešetke ili trelis) koji naglašavaju i vremensku dimenziju dijagrama. Konvolucioni kodovi se stoga često nazivaju i trelis kodovima.



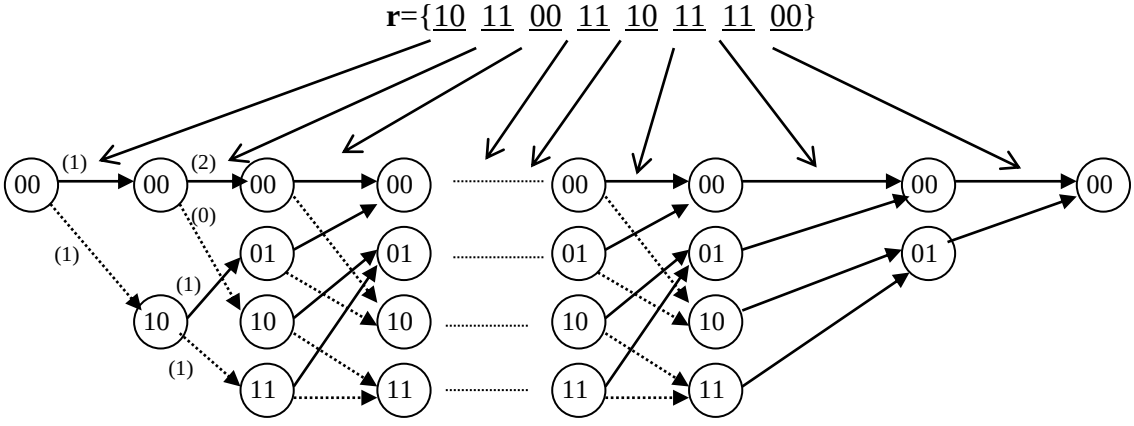
Prethodnu kodnu rešetku bi trebalo popuniti i ostalim stanjima i dobiti kompletnu kodnu rešetku.

Za praktične primjene se koristi konvolucioni kodovi sa L-tim ograničenjem. L-tom ograničenju odgovara i ograničenje u prikazu kodne rešetke. Ako primjenimo L=6 ograničenje na kod sa prethodne slike, dobijamo ulazni niz $[i_5,i_4,i_3,i_2,i_1,i_0]$, što daje dužinu puta kroz dijagram od 8 grana. Naime, zbog dva pomjeračka registra na krajevima, početak i kraj procesa kodiranja izgleda ovako: $[i_0,0,0]$, $[i_1,i_0,0]$, $[i_2,i_1,i_0]$,...,$[i_5,i_4,i_3]$,$[0,i_5,i_4]$,$[0,0,i_5]$.



10.5. Viterbijev algoritam za dekodiranje konvolucionih kodova

Performanse konvolucionih kodova zavise od algoritma dekodiranja (kodiranje obično nije previše problematično) i osobina vezanih za rastojanje između kodnih sekvenci (ne i između kodnih riječi). Najvažnija mjera za rastojanje koja se koristi kod konvolucionih kodova je slobodno rastojanje (*free distance*) d_{free}. Ovo je minimalno moguće rastojanje bilo koje dvije sekvence na izlazu iz kodera, koje odgovaraju različitim ulaznim sekvencama. Rešetkasti dijagram (trelis) se može primjeniti i za dekodiranje poruke. Neka se kod koji smo uzeli za primjer prenosi preko binarnog simetričnog kanala sa vjerovatnoćom greške P_g<0.5 i neka je primljeno **r**={10 11 00 11 10 11 11 00}. Primjenjeni kod je linearni blok kod (16,6). Dekoder mora od mogućih 64 kodnih riječi naći onu koja je najbliža po Hammingovoj distanci zadatoj kodnoj riječi. Nacrtajmo ovaj dijagram u drugom obliku.



Uz (neke) oznake prelaza umjesto vrijednosti izlaza dekodera date su Hammingove distance između te vrijednosti i dva bita primljene vrijednosti. Ako je primljeno 11, a grana je 00, riječ je o Hammingovoj distanci 2. Problem određivanja kodne riječi koja je najbliža datom nizu se svodi na određivanje najkraćeg puta kroz rešetkasti dijagram sa slike. Da bi ovo uspješno opisali posmatrajmo još neke oznake. Neka je skup svih stanja $S=\{A, B, C, D\}$. Ova stanja su naznačena na grafu tranzicije definisanom u sekciji 9.4. Ako su stanja $s,t \in S$ postoji grana koja povezuje *s*-*t* i mi možemo definisati veličinu $B(s,t)$ koja uzima vrijednost 0 ili 1 u zavisnost od toga koja vrijednost uzrokuje promjenu (da li je puna ili isprekidana linija). Ako prelaz ne postoji $B(s,t)$ je neodređena.

		t			
		A	*B*	*C*	*D*
s	*A*	0	-	1	-
	B	0	-	1	-
	C	-	0	-	1
	D	-	0	-	1

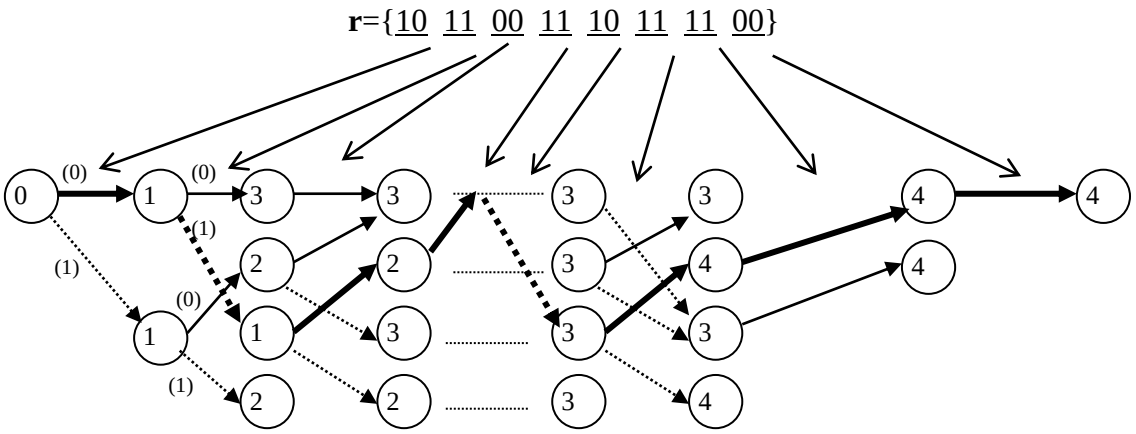
Sada za $s,t \in S$ se mogu definisati $l_{i-1;j}(s,t)$, kao oznaku koja se nalazi na grani rešetkastog dijagrama spaja sa s_{j-1} i t_j. Ako grana ne postoji postavlja se: $l_{i-1;j}(s,t)=\infty$. Npr. $l_{0;1}(A,C)=1$, $l_{2;3}(D,B)=1$, $l_{7;8}(A,C)=\infty$. Jedan od algoritama za traženje najkraćeg puta je Viterbijev. Viterbijev algortima je imao jako interesantnu istoriju. Tokom njegovog razvoja pokazano je da je ovo najbolji postupak (ML postupak) za dekodiranje. Našao je primjenu i van kodne teorije da se koristi i u rješavanju problema iz teorije grafova, gdje se traži najkraće rastojanje kroz graf i to tako da grane grafa imaju određenu težinu. On polazi od računanja dvije vrijednosti: metrike i prethodnika. Metrika $\mu_j(s)$, $s \in S$ je dužina najkraćeg puta od A_0 do s_j dok je prethodnik $B_j(s)$ niz od *j* binarnih simbola koji označavaju najkraću put od A_0 do s_j. Npr. $B_4(B)$=1010 znači da je najkraći put od A_0 do B_4: A_0 C_1 B_2 C_3 B_4. Viterbijev algoritam se može opisati po sljedećim koracima:

(1) Staviti $\mu_0(A)<-0$ i $\mu_0(s)=\infty$ za sve $s \neq A$. Postaviti $B_0(A)=\{0\}$ i j=1.

(2) Za svaki $s \in S$ naći $t \in S$ za koji je $\mu_{j-1}(t)+l_{j-1;j}(t,s)$ minimalno. Postaviti: $\mu_j(s)<-\mu_{j-1}(t)+l_{j-1;j}(t,s)$ i $B_j(s)<-B_{j-1}(s)B(t,s)$.

(3) Ako je $j=L+M$ tada je prvih *L* bita u $B_j(A)$ dio najkraćeg puta inače postaviti $j<-j+1$ i preći na korak (2).

Za dijagram sa prethodne slike ilustrujmo Viterbijev algoritam. Kod Viterbijevog algoritma oznaka * znači nastavljanje prethodnika sa narednim simbolom (10110*1=101101).



Na ovoj slici je unutar čvorova prikazana metrika $\mu_j(s)$ a put prethodnika je označen najkraćim putem od A_0 do s_j. Npr. $\mu_3(B) = 2$ i $B_3(B) = \{010\}$. Najkraći put od A_0 do A_8 je $A_0\, A_1C_2\, B_3A_4\, C_5B_6$ - A_7A_8 koji je podebljan. Njegova ukupna dužina je 4, $B_8(A)$=[010001000], pa od prvih 6 simbola dobijamo procjenjenu informaciju kao **i**'=[0100010].

Da bi pokušali da na još plastičniji način objasnimo Viterbijev algoritam posmatrajmo prethodni dijagram i pokušajmo da odredimo najbolji put do svih čvorova u drugoj koloni. Obije putanje imaju Hammingovu distancu 1. Treba ih obije zapamtiti: putanju 00-00 i putanju 00-10. Odredimo sada najbolje putanje do čvorova u drugom trenutku. Te putanje su konkatenacija (nadovezivanje) putanja koje su bile najbolje do čvorova iz prethodnog trenutka sa putanjom od druge do treće kolone. U svaki od čvorova se može stići preko jedne putanje:

U čvor 00 preko putanje 00-00-00 koja ima distancu 3 (Put I);
U čvor 01 preko putanje 00-10-01 koja ima distancu 2 (Put II);
U čvor 10 preko putanje 00-00-10 koja ima distancu 1 (Put III);
U čvor 11 preko putanje 00-10-11 koja ima distancu 2 (Put IV).

Najbolja putanja je treća putanja, ali treba zapamtiti sve putanje koje su najbolje (u ovom slučaju i jedine moguće) do datih čvorova. U terminologiji Viterbijevog algoritma ove putanje se često nazivaju *parcijalni najbolji putevi*. Pogledajmo sada četvrti red čvorova. Najbolje putanje do pojedinih čvorova su:

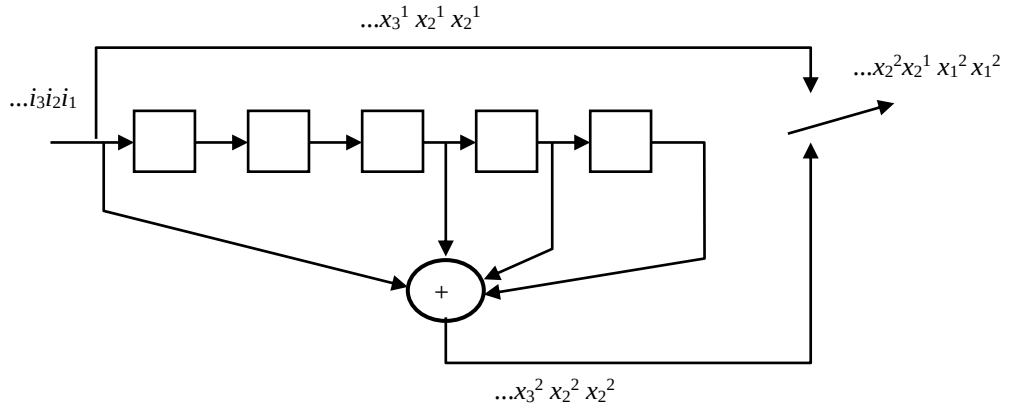
Do čvora 00 to je konkatenacija Put I-00 (novi Put I) sa Hammingovom distancom 3;
Do čvora 01 to je konkatenacija Put III-01 (novi Put II) sa Hammingovom distancom 2;
Do čvora 10 to je konkatenacija Put II-10 (novi Put III) sa Hammingovom distancom 2;
Do čvora 11 to je konkatenacija Put III-11 (novi Put IV) sa Hammingovom distancom 2.

U ovom trenutku ne bi mogli da donesemo odluku koja je putanja najbolja, ali mi moramo da pamtimo ove 4 putanje. Probajte da sami rekonstruišete naredne korake, a posebno peti red čvorova koji je na dijagramu izostavljen.

Metrika koja se koristi kod Viterbijevog algoritma ne mora biti "hard" – zasnovana na Hammingovoj distanci, već može biti "soft" zasnovana na veličini koja se transformiše u bite. Npr. mi primamo veličinu i sa 0 proglašavamo stanje manje od 2.5 dok sa 1 proglašavamo veće stanje. Umjesto tako dobijenih stanja mi možemo koristiti vrijednosti veličine jer nije nam svaka jedinica jednako pouzdana: ona koja je primljena sa 2.6 i ona koja je primljena sa 5.

10.6. Dekodiranje pomoću majoritetne logike

Viterbijev algoritam je jedna od najefektnijih procedura koja se koriste u nauci. Glavni problem koji je posebno ranije postojao kod Viterbijevog algoritma je memorijska zahtjevnost. Stoga se ponekad traže druge metode za dekodiranje. Danas, se uglavnom smatra da je memorijska zahtjevnost Viterbijevog algoritma prihvatljiva, pa se drugi algoritmi za dekodiranje konvolucionih kodova rjeđe koriste. Napominjemo da kod izuzetno dugih kodova memorijski zahtjevi Viterbijevog algoritma rastu eksponencijalno, pa ipak se dešavaju situacije kada treba izvršiti implementaciju nekog drugog dekodirajućeg algoritma. Preostala dva algoritma za dekodiranje konvolucionih kodova su sekvencijalni algoritam i algoritam sa majoritetnom logikom. Majoritetna logika i Viterbijev algoritam mogu da dekodiraju i blok kodove. Dok se blok kodovi koriste za otkrivanje i za ispravljanje grešaka, dotle se konvolucioni kodovi koriste isključivo za ispravljanje grešaka. Dekodiranje na osnovu majoritetne logike je specijalni slučaj dekodiranja na osnovu praga. Kod majoritetne logike se koriste ortogonalne provjere na parnost. Za svaki bit se na osnovu ovakve provjere donosi odluka da li je ispravan ili nije. Ako je ispravan pušta se u dekoder, a ako nije ispravlja se (i koriguju dobijene provjere parnosti). Procedura se nastavlja ka narednim informacionim bitima. Na slici je prikazan konvolucioni dekoder sa kodnim odnosom R=1/2.



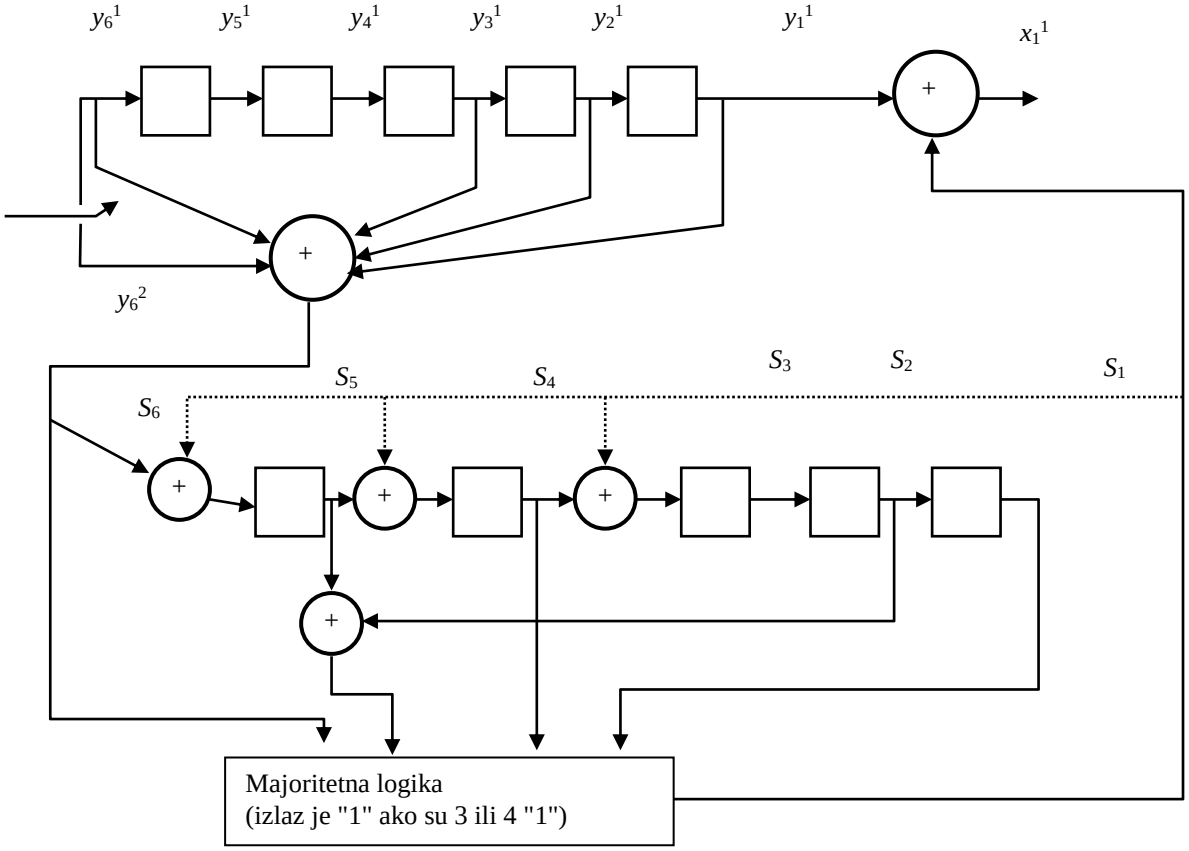
Informacioni biti su označeni sa i_i dok su izlazni $x_i^{1(2)}$. Gornji indeks označava da li je to prvi ili drugi bit u paru koji se emituje kada koder uđe u informacioni blok. Kanalni biti u donjoj grani su identični sa informacionim bitima, pa je u pitanju sistemski koder. Koder je na početku resetovan (ispunjen nulama). Kada se pojavi bit i_i sledeći kanalni biti su:

$$x_i^{\,1} = i_i \qquad x_i^{\,2} = i_1 \oplus i_{i-3} \oplus i_{i-4} \oplus i_{i-5} = x_i^{\,1} \oplus x_{i-3}^{\;\;1} \oplus x_{i-4}^{\;\;1} \oplus x_{i-5}^{\;\;1}$$

Kanalni biti traju dva puta manje od informacionih. Nova sekvenca sa greškama se može opisati kao:

$$y_i^{\,1(2)} = x_i^{\,1(2)} \oplus e_i^{\,1(2)}$$

Dekoder za posmatrani slučaj je prikazan na slici.



Dekoder kod kojega su replika kodera i sindrom na početku postavljeni na 0 računa niz sindroma S_i koristivši provjere parnosti unesene kao:

$$S_i = y_i^2 \oplus y_i^1 \oplus y_{i-3}^2 \oplus y_{i-4}^2 \oplus y_{i-5}^2$$

Ako su se dogodile greške sindromi će biti 0, a u suprotnom će biti 1. Naime, za kod bez greške sindrom iznosi

$$S_i = x_i^2 \oplus x_i^1 \oplus x_{i-3}^2 \oplus x_{i-4}^2 \oplus x_{i-5}^2 = x_i^1 \oplus x_{i-3}^2 \oplus x_{i-4}^2 \oplus x_{i-5}^2 \oplus x_i^1 \oplus x_{i-3}^2 \oplus x_{i-4}^2 \oplus x_{i-5}^2 = 0$$

Ako zamijenimo odgovarajuće vrijednosti dobijamo (pošto važi $x_i^2 \oplus x_i^1 \oplus x_{i-3}^2 \oplus x_{i-4}^2 \oplus x_{i-5}^2 = 0$):

$$S_i = e_i^2 \oplus e_i^1 \oplus e_{i-3}^2 \oplus e_{i-4}^2 \oplus e_{i-5}^2$$

Za vrijednosti i između 1 i 6 dobija se (znajući da $e_i^j = 0$ za $i \leq 0$):

$$S_1 = e_1^2 \oplus \mathbf{e_1^1} \qquad S_2 = e_2^2 \oplus e_2^1$$
$$S_3 = e_3^2 \oplus e_3^1 \qquad S_4 = e_4^2 \oplus e_4^1 \oplus \mathbf{e_1^1}$$
$$S_5 = e_5^2 \oplus e_5^1 \oplus e_2^1 \oplus \mathbf{e_1^1} \qquad S_6 = e_6^2 \oplus e_6^1 \oplus e_3^1 \oplus e_2^1 \oplus \mathbf{e_1^1}$$

$\mathbf{e_1^1}$ se pojavljuje u 4 pozicije u sindromu. Ako je $\mathbf{e_1^1}$=1 i ako na drugim bitima nema greške to znači da će te četiri pozicije u sindromu biti 1. Majoritetna logika kaže da je $\mathbf{e_1^1}$=1, ako je većina od četiri sindroma S_1, S_4, S_5 i S_6 jednaka 1. Postoji problem uticaja $\mathbf{e}_2^1$ koji se pojavljuje u više sindromskih koeficijenata, ali se on može izbjeći sabiranjem S_2 i S_5 kada se dobija sledeći skup sindroma:

$$S_1 = e_1^2 \oplus \mathbf{e_1^1} \qquad S_2 \oplus S_5 = e_5^2 \oplus e_5^1 \oplus e_2^2 \oplus \mathbf{e_1^1} \qquad S_4 = e_4^2 \oplus e_4^1 \oplus \mathbf{e_1^1}$$
$$S_6 = e_6^2 \oplus e_6^1 \oplus e_3^1 \oplus e_2^1 \oplus \mathbf{e_1^1}$$

U ovom skupu se $\mathbf{e_1^1}$ pojavljuje uvijek, dok se ostali elementi pojavljuju jednom, pa se kaže da je sistem ortogonalan u odnosu na $\mathbf{e_1^1}$. Ako su tri ili sva četiri sindroma 1 onda se dogodila greška. To radi blok sa majoritetnom logikom. Kada se donese odluka o x_1^1 onda se indeks *i* uveća za 1, *i*=7, i dobijaju se sledeći sindromi:

$$S_2 = e_2^2 \oplus \mathbf{e_2^1} \qquad S_5 = e_5^2 \oplus e_5^1 \oplus \mathbf{e_2^1} \oplus e_1^1 \qquad S_3 \oplus S_6 = e_6^2 \oplus e_6^1 \oplus e_3^2 \oplus \mathbf{e_2^1} \oplus e_1^1$$
$$S_7 = e_7^2 \oplus e_7^1 \oplus e_4^1 \oplus e_3^1 \oplus \mathbf{e_2^1}$$

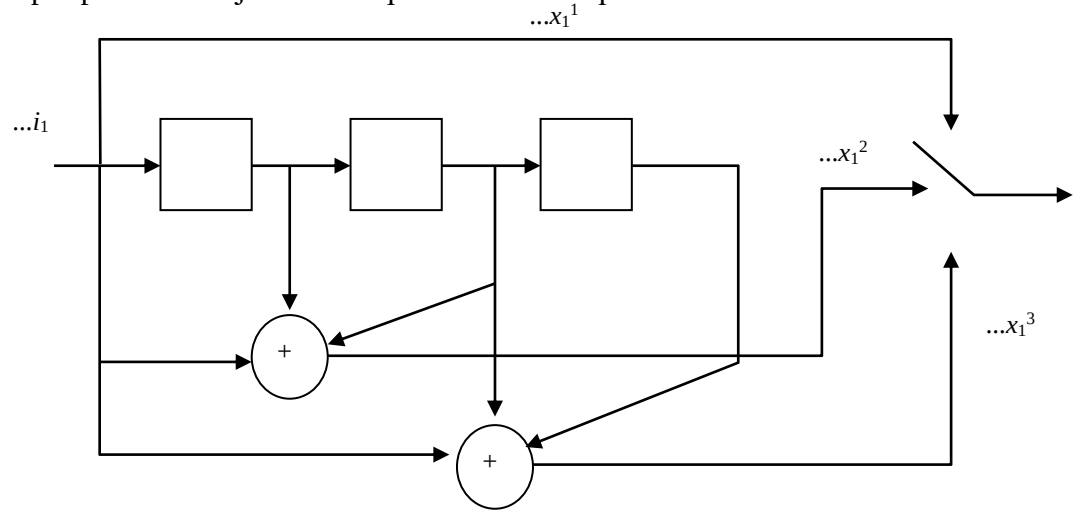
Ove su riječi sada ortogonalne u odnosu na $\mathbf{e_2^1}$, pa se može vršiti korekcija. Ako se detektuje greška mora se vršiti korekcija informacionog bita, ali se mora vršiti i korekcija sindroma. Ako se pogrešno detektuje pogreška, ona će propagirati kroz ostale kodne riječi. Može se pokazati da ako zaredom stigne veliki broj bita bez pogreške dolazi do oporavljanja kodne riječi, odnosno od jednog bita na dalje nastavlja se uspješno dekodiranje (mada postoje kodovi koji se ne mogu oporaviti i koji se nazivaju katastrofalnim). Majoritetna logika se može primjenjivati samo kod kodova koji su ortogonalni ili kod onih koji se mogu ortogonalizovati (odnosno kod kodova kod kojih se može podesiti da svi sindromi zavise od jednog bita pogreške, a da nijedan drugi bit pogreške se ne pojavljuje više od jednom u sindromima).

10.7. Sekvencijalno dekodiranje

Sekvencijalno dekodiranje je posljedica dugog razvoja u ovoj oblasti i mnogo je naučnika doprijenijelo njegovoj realizaciji. Postoje varijante zvane Fanoovo algoritma, kao i stek algoritam. Posmatrajmo primjer koda sa parametrima: *K*=1, *M*=3, *N*=3, *R*=1/3 čiji je sistemski koder dat na narednoj ilustraciji.

Na početku su u koderu sve nule. Sve kodne sekvence se u ovom slučaju mogu prikazati preko kodnog stabla. Kodno stablo za prva četiri koraka je dato ispod naredne slike, dok su kanalni biti raspoređeni obrnuto u odnosu na sliku gore. Kretanje u gornju granu je forsirano jedinicom na ulazu, dok je kretanje u bilo koju donju granu stabla prouzrokovano nulom na ulazu.

Kako je ovo sistematski kod, to je prvi bit ujedno i informacioni. Ako je kodna sekvenca 1010 to će izlazna sekvenca biti 111 010 101 001. Neka se greška dogodila na drugom bitu u sekvenci i neka je primljena poruka: 101 010 101 001. Dekoder pretpostavljajući da se dogodila jedna greška kreće kroz gornju granu stabla. Sljedeća kodna riječ je 010 i ona zbilja postoji u kodnom stablu. Može se pretpostaviti da je koder na početku donio ispravnu odluku.

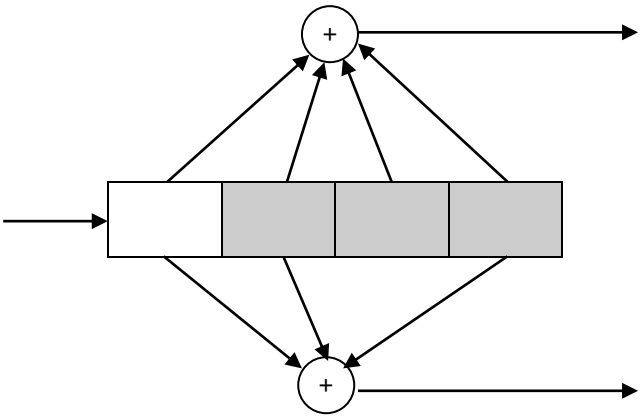


$x_1^1\ x_1^2\ x_1^3$	$x_2^1\ x_2^2\ x_2^3$	$x_3^1\ x_3^2\ x_3^3$	$x_4^1\ x_4^2\ x_4^3$
		1 1 0	1 0 1
	1 0 1		0 1 0
		0 0 1	1 1 1
1..1..1			0 0 0
1		**1 0 1**	1 1 0
	0 1 0		**0 0 1**
0		0 1 0	1 0 0
			0 1 1
		1 0 1	1 1 0
	1 1 1		0 0 1
		0 1 0	1 0 0
0 0 0			0 1 1
		1 1 1	1 0 1
	0 0 0		0 1 0
		0 0 0	1 1 1
			0 0 0

Ako je primljena sekvenca 010 010 101 001 koder će poći u donju granu, ali sljedeća kodna riječ nije moguća 010, a ni sljedeće trojke se ne mogu naći u produžetku izabrane grane, ali se mogu pridružiti kombinacijama 111 i 101. Vidimo da Hammingovo rastojanje u ovom slučaju raste. Kada kumulativna greška naraste previše sekvencijalni dekoder se vraća unazad i ponavlja postupak. Na žalost i pored jednostavne ideje ovo je izuzetno komplikovan procedura. Sekvencijalni dekoder radi jako dobro kada je nivo šuma (broj grešaka) mali, ali kad nivo šuma raste performanse algoritma značajno slabe. Često se čak i kada postoje uslovi za rad sekvencijalnog dekodera koristi Viterbijev postupak pošto je jednostavniji za realizaciju u VLSI tehnici.

Zadaci za vježbu

10.1 Dat je sledeći konvolucioni koder.



Napomena. Ovo je alternativni način da se vizuelizuju koder konvolucionog koda kod kojega smo spojili pojedine ćelije pomjeračkog registra. Bijela ćelija je zapravo ulazni bit dok su osjenčeni bitovi memorija ovog kodera.
(a) Odrediti karakteristike koda koji se formira zahvaljujući ovom sklopu (*N*, *M*, *R*, *V* itd).
(b) Formirati look-up tabelu sa 16 kolona u kojima su upisane vrijednosti ulaznog bita, početna stanja u ćelijama registra na osnovu kojih treba sračunati naredna stanja u ćelijama registra kao i izlazne bitove.
(c) Odrediti polinomijalni zapis funkcije ovog kodera kao i matrični zapis preko matrice **G**.
(d) Nacrtati dijagram stanja ovog kodera (ovaj graf ima 8 čvorova).
(e) Formirati trelis dijagram.
(f) Kodirati poruku 1011 koristivši dijagram stanja kao i pomoću trelisa (nemojte pretpostavljati da je trelis ograničen).
(g) Pretpostaviti da je primljena poruka 01 11 01 11 01 01 11. Ispraviti poruku (odrediti koliko je grešaka napravljeno i na kojim pozicijama). Dekodirati poslatu poruku. Prilikom kretanja kroz sekvencijalno stablo postaviti da se može ići u neku granu, dok je Hammingova distanca između primljene riječi manja od 3. Kad postane veća od 3 trebate se vratiti na granu koja nije još provjerena a koja ima Hammingovo rastojanje manje od 3.
(h) Dekodiranje izvršiti Viterbijevim algoritmom. Uočite da ovaj kod ima 8 čvorova odnosno 8 redova sa čvorovima u trelisu.

Rješenje. (a) U gornjoj grani se nalazi koder sa generatorskim polinomom $1+x+x^2+x^3$ dok je u donjoj grani koder sa generatorskim polinomom $1+x+x^3$. Memorija ovog kodera je jednaka najvećem stepenu generatorskog polinoma što je u ovom slučaju jednako $M=3$. Ograničena dužina koda je $N=M+1=4$. Matrica **G** (uslovno je nazovimo generatorskom matricom) za predmetni koder ima oblik:

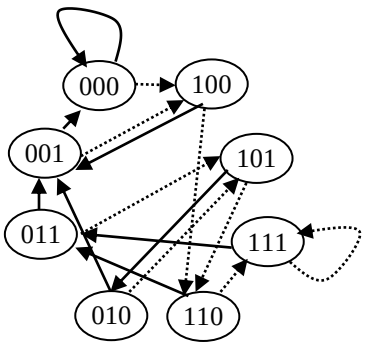
$$\mathbf{G}=[x^3+x^2+x+1, x^3+x+1]$$

Stoga je kodni odnos ovog koda $R=1/2$ (broj vrsta kroz broj kolona generatorske matrice). Ukupna ograničena dužina koda je $V=2x4=8$ bita jer postoje dvije grane sa oba stepena jednaka tri odnosno sa 4 bita.
(b) Formirajmo predmetnu tabelu:

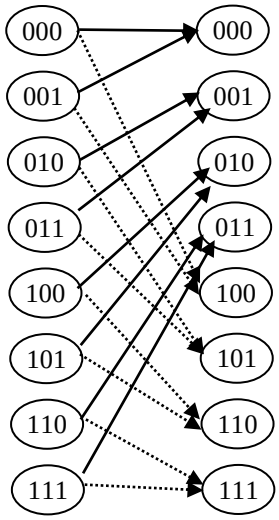
Ulazni bit	Stanje u ćelijama	Naredno stanje	Izlaz gore	Izlaz dolje
0	000	000	0	0
0	001	000	1	1
0	010	001	1	0
0	011	001	0	1
0	100	010	1	1
0	101	010	0	0
0	110	011	0	1
0	111	011	1	0
1	000	100	1	1
1	001	100	0	0
1	010	101	0	1
1	011	101	1	0
1	100	110	0	0
1	101	110	1	1
1	110	111	1	0
1	111	111	0	1

(c) Obavljeno pod (a).

(d)



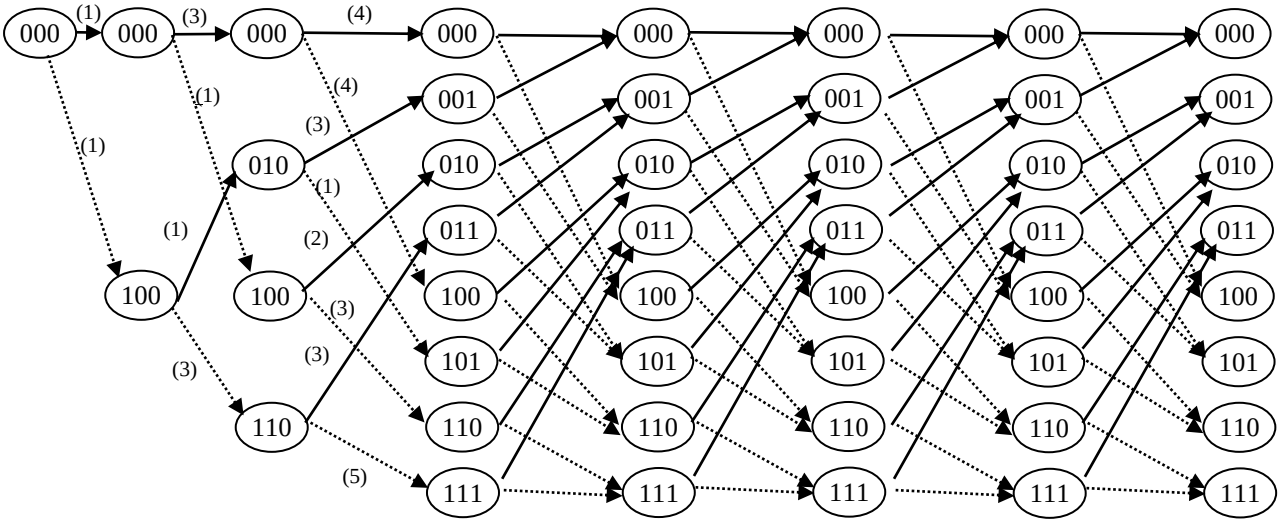
(e) Trelis dijagram se može prikazati sa jednom laticom.



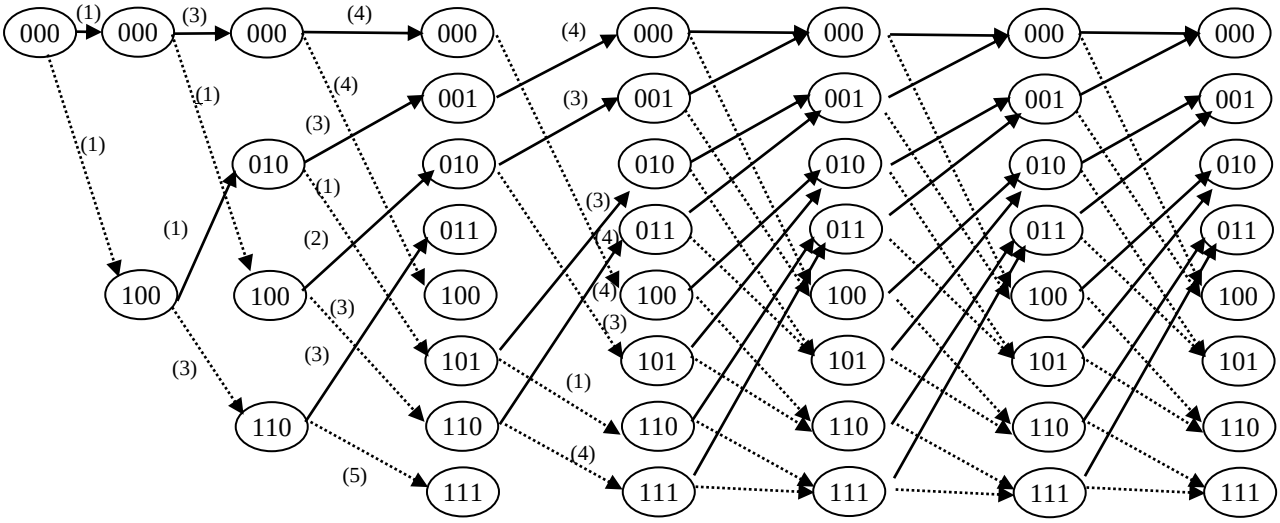
(f) 1011. Na početku pretpostavimo da je na početku koder prazan 000. Na početku ulazi 1 na izlazu dobijamo dva bita 11 a sistem prelazi u stanje 100. Zatim (čitajući poruku od početka) stiže 0. Na izlazu se pojavljuju dvije jedinice (11) a sistem prelazi u stanje 010. Zatim ulazi 1 izlazi su 01 a sistem prelazi u stanje 101. Konačno dolazi 1 i iz sistema izlaze dvije jedinice a sistem prelazi u stanje 110. Dakle, zaključujemo da je izlaz iz sistema:

11 11 01 11

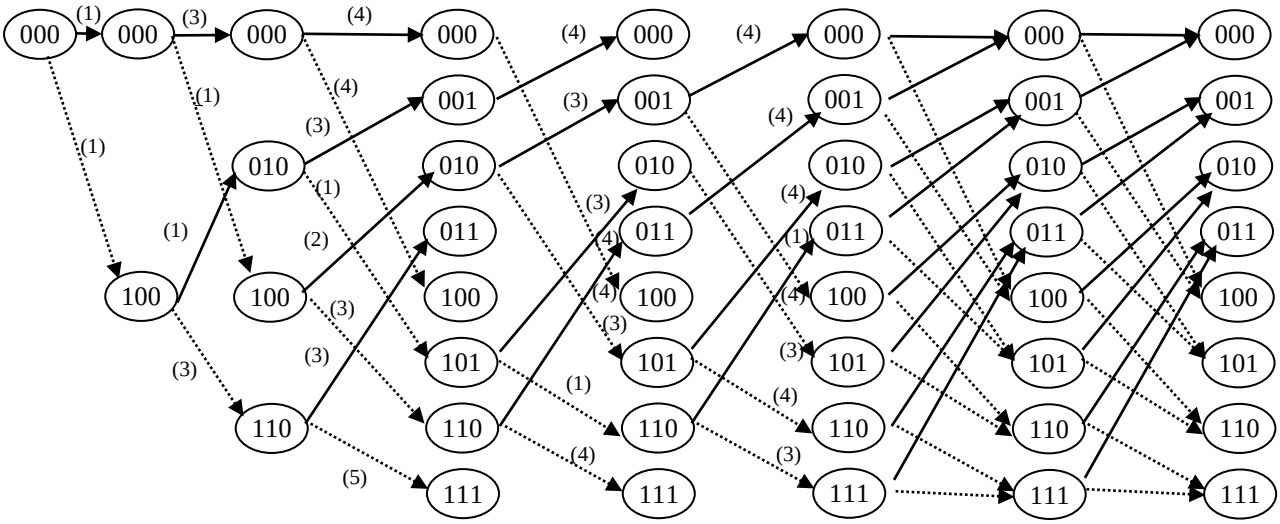
(g) Sekvencijalno dekodiranje probajte da sami obavite. Mi ćemo samo ilustrovati Viterbijev algoritam.
(h) Izvršimo sada dekodiranje poruke 01 11 01 11 01 01 11 uz prikazivanje odgovarajućeg dijagrama sa početnim stanjem 000.
U prva tri koraka imamo sljedeću situaciju gdje su Hammingove distance između mjerenih putanja i dobijenih prikazane u zagradama. Radi daljeg rada sa algoritmom sve ove putanje moraju biti sačuvane.



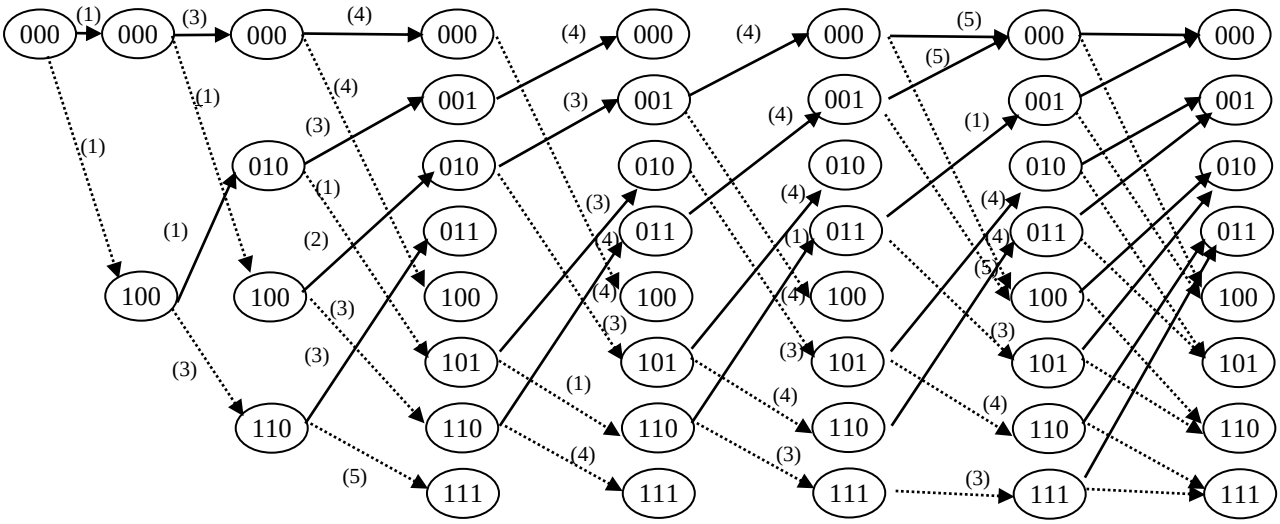
Do svakog čvora u četvrtom koraku postoje dvije putanje. Pamtimo samo bolju od njih (eventualno obije ako su im težine iste). Ove putanje se u terminologiji Viterbijevog algoritma nazivaju parcijalni najbolji putevi. Jedna od putanja (ona do čvora 110) ima Hammingovu razliku od samo 1 i ona predstavlja trenutnu dekodiranu vrijednost).



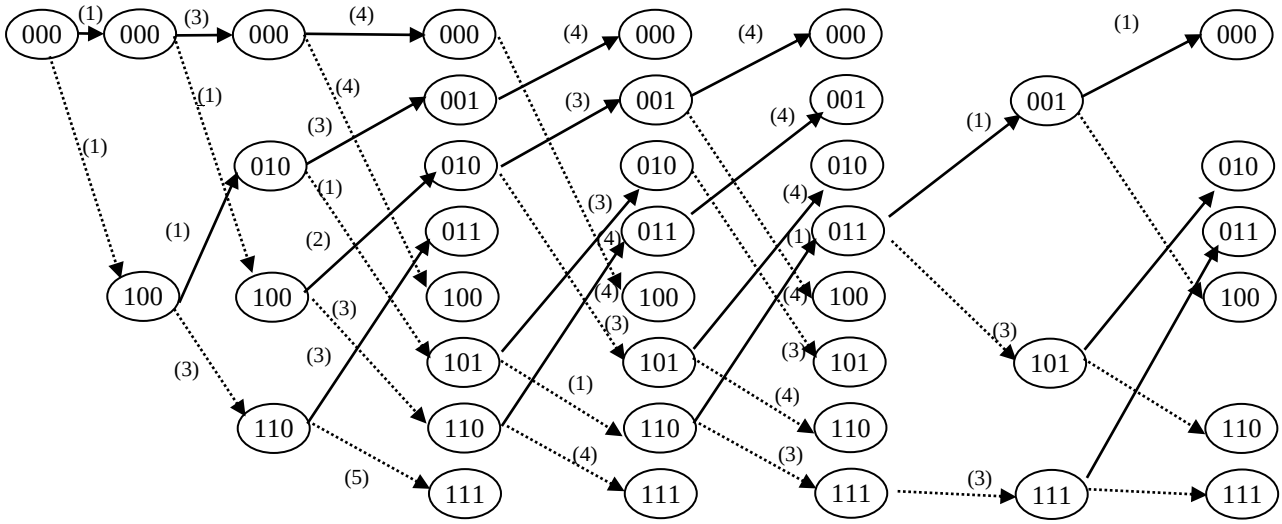
Stanje u petom koraku je dato dolje. Trenutno je najbolja ona putanja koja stiže do čvora 011 i koja ima Hammingovu distancu od primljenog koda koja je jednaka 1.



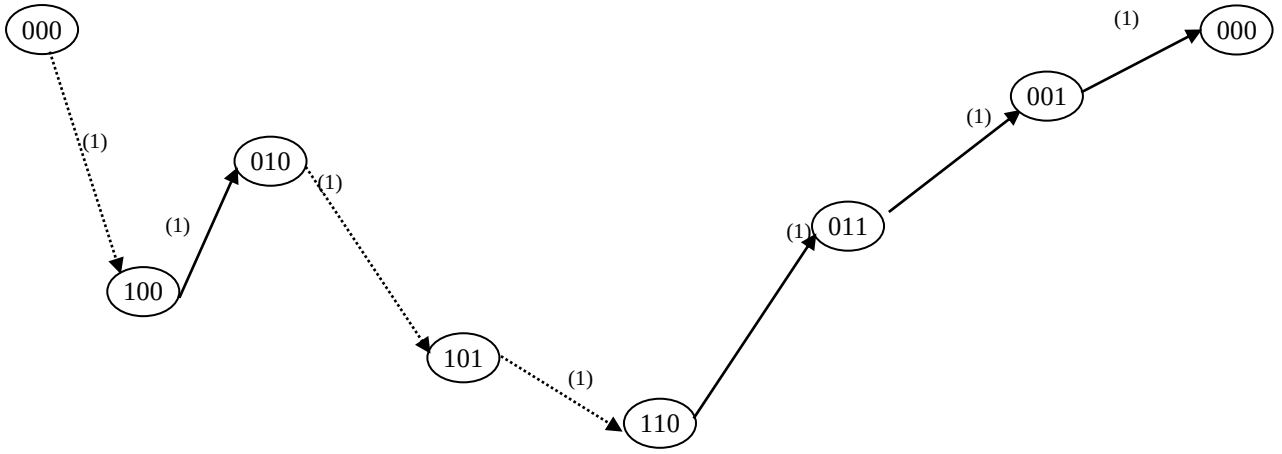
Stanje u šestom koraku je dato dolje. Najbolja putanja je ona do čvora 001 koja ima težinu samo 1. Pošto je ostalo dekodiranje samo još u jednom koraku postoje samo tri konkurentne putanje koje mogu da predstavljaju rezultat dekodiranja to su pored putanje koju smo pomenuli i one putanje do čvorova 101 i 111 koje imaju težinu 3.



U sedmom (finalnom koraku) pratimo samo tri parcijalna najbolja puta iz prethodnog koraka.



Vidimo da je putanja do čvora 000 ona koja daje najmanju težinu tako da nismo dalje ispitivali ostale putanje. Prikažimo sada samo ovu "najbolju" putanju.



Sa ovako prikazanim trelisom dekodiranje je krajnje jednostavno. Isprekidane linije su prouzrokovane 1 dok su pune prouzrokovane sa 0 pa je poruka koju dekodiramo 1011000. Kreirajte sami primljenu poruku i analizirajte i poredite složenost sekvencijalnog dekodiranja i Viterbijevog algoritma. U ovom posmatranom primjeru složenost će kod oba pravila dekodiranja biti prilično mala što je uzrokovano malom težinom pogreške u primljenoj riječi ali će kod većih težina pogreške prednosti Viterbijevog algoritma.

10.2. Napišite MATLAB program koji vrši konvoluciono kodiranje putem sekvencijalnog postupka i putem Viterbijevog algoritma.

Poglavlje XI
TURBO KODOVI

11.1. Istorija turbo kodova - Uvod

Podsjetimo se kodne teoreme. Kodna teorema kaže da je moguće konstruisati kod sa kodnim odnosom R koji sa vjerovatnoćom greške koja je bliska nuli može preko kanala koji je kapaciteta C ako je $R \leq C$. Obično je i dužina kodne riječi unaprijed poznata (n) pa se ova teorema svodi na činjenicu da broj informacionih bita u riječi mora biti $k \leq nC$ da bi se mogla postići vjerovatnoća greške koja teži 0. Šenonova teorema nije konstruktivna i ne kaže kako se do datih kodova može doći i što je još gore, asimptotska je jer važi samo za $n \to \infty$. Nama je cilj da za dati kanal postignemo $k=nC$ jer alternativa podrazumijeva da šaljemo manji procenat korisnih bita, odnosno da dio energije koristimo neracionalno (svaki preneseni bit podrazumijeva potrošnju određene količine energije) da bi nepotrebno povećali redundanciju. Još preciznije, nama je cilj da se primaknemo ovom odnosu za n konačne dužine pošto kodovi ekstremno velike dužine ili se ne mogu realizovati ili bi njihova realizacija zahtjevala neprihvatljivo velike hardverske ili softverske zahtjeve. Drugi problem kod upotrebe predstavlja činjenica da veliko n iziskuje da imamo ekstremno veliki propusni opseg za prenos signala kodiranog sa ovako velikim n. Osnovni problem je bio taj, što je sve do sredine devedesetih godina dvadesetog vijeka energija potrebna za prenos informacija preko najbezazlenijeg kanala bila oko 2 puta veća nego što je to predviđeno kodnom teoremom. Na jednoj konferenciji grupa francuskih naučnika Berrou, Glavieux i Thijtimasjshima je izložila koncept turbo kodova 1993-će godine. Tvrdili su da je moguće se primaći Shannonovoj granici na samo 12% prekomjernog trošenja energije korišćenjem koncepta tzv. "turbo kodova". Njihov rad je u prvo vrijeme dočekan sa podsmjehom, ali su nakon nekoliko godina njihovi rezultati bili simulaciono potvrđeni od drugih naučnika. Od tada turbo kodovi u teoriji informacija i kodova postaju jedno od najvažnijih polja istraživanja iz dva razloga: očigledna je njihova potencijalna upotrebljivost kao i potreba da se istraže mane ovog sistema kodiranja kao i načini da se te mane prevaziđu. Termin "turbo" potiče iz načina dekodiranja ovih kodova koji je podsjeća na rad turbo motora.

Krajem dvadesetog vijeka razvijeni su turbo kodovi za brojne praktične aplikacije tako da se koriste: u tzv. "deep space" komunikacijama od strane NASA-e; digitalnom video broadcastingu (DVB-T), te u komunikacionim sistemima za mobilne komunikacije treće generacije: code division multiplex access (CDMA2000) i universal mobile telecommunication standard-u (UMTS).

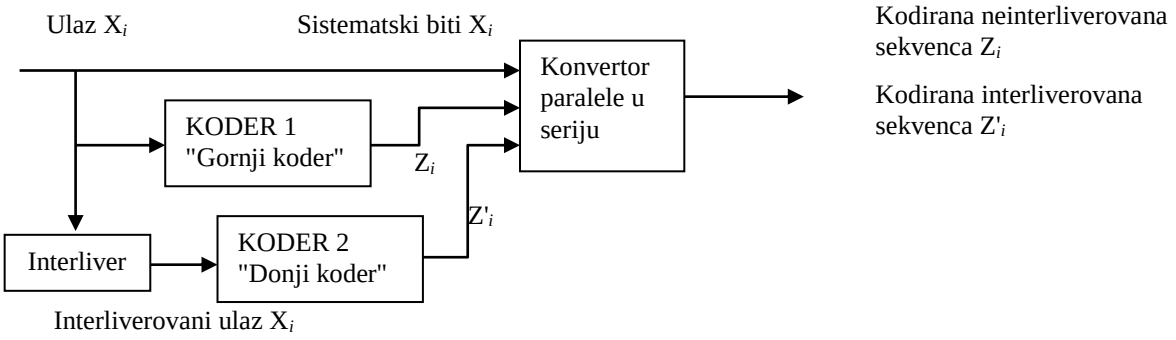
U ovom poglavlju dajemo samo najosnovnije informacije o turbo kodiranju i dekodiranju. Na nekoliko mjesta napuštamo matematičke formalizme i činjenice objašnjavamo na simplifikovanim modelima koji se ne upotrebljavaju na prezentirani način u praksi. Razlog je želja da ne komplikujemo izlaganje suvišnim detaljima koji ne mogu da budu jasni studentima u okviru osnovnog kursa teorije informacija i kodova, ali istovremeno pokušavamo da iznesemo sve važne činjenice za razumijevanje ove materije.

Ovaj materijal je zasnovan na [5] i [6] koje vam preporučujem kao prve korake u daljem istraživanju ove materije.

11.2. Kako funkcioniše turbo kodiranje

Turbo kod nije jedan kod već je to zapravo kombinacija dva (ili više) kodova koji funkcionišu zajedno. Koderi pojedinačnih kodova se po pravilu nadovezuju paralelno, ali u posljednje vrijeme postoje sistemi i sa redno nadovezanim koderima. Postoje teorijski proučeni sistemi i sa većim brojem kodera ali takvi sistemi rijetko su implementirani u komunikacionim standardima. Važan

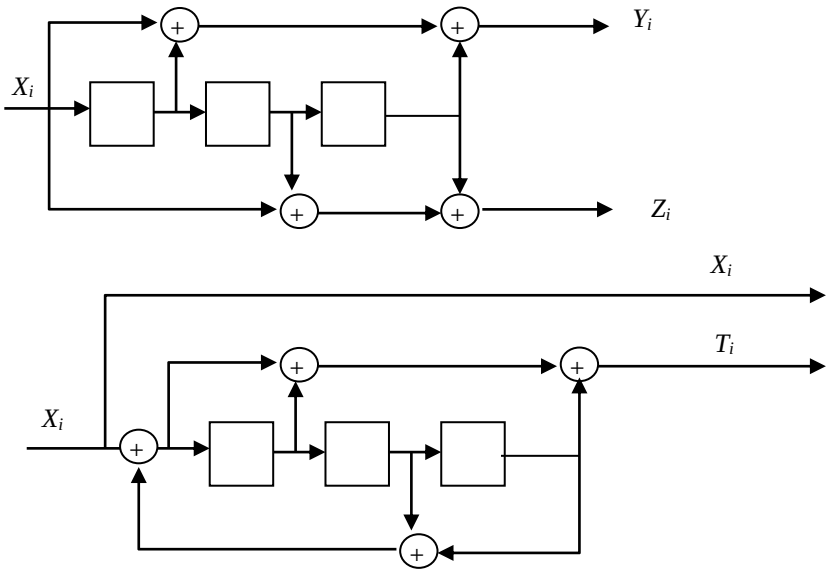
detalj u kodnom sistemu je interliver koji postoji između dva (ili više kodera). Napominjemo da ovaj interliver preuređuje (permutuje) originalnu sekvencu na neki složen ali dogovoren način. Naknadno ćemo objasniti kako funkcioniše interliver na principskom nivou, kao i o razlozima zbog kojih je izuzetno važan. Po nepisanom pravilu koderi koji se koriste u sistemu turbo kodera su isti ali to ne mora biti tako. Struktura jednog prostog turbo kodera je data na slici.



Što je zadatak interlivera? Pretpostavimo da pojedinačni kod koji se koristi za kodiranje ulazne sekvence (može biti bilo koji od onih koji su već obrađeni: Hammingovi, BCH, jednom riječju blok kodovi, kao i konvolucioni kodovi) predstavlja množenje ulazne informacione sekvence (ulaznog informacionog polinoma) sa generatorskim polinomom koda. Jedna moguća kombinacija na ulazu je kombinacija sa *k* nula koja kod klase kodova koju posmatramo na izlazu daje sekvencu u kojoj su takođe sve nule. Jedan od načina da odredimo minimalnu Hammingovu distancu koda je da nađemo onu riječ koja se najmanje razlikuje (na najmanje bita) od neke kodne riječi. Nama je stoga najlakše to uvijek provjeriti na primjeru kodne riječi sa svim nulama. Najmanju razliku u odnosu na ovu kodnu riječ ima ona kodna riječ koja ima najmanje jedinica (najmanju Hammingovu težinu). Kako minimalna Hammingova distanca određuje sposobnost koda da ispravi veći broj pogreški to nam je cilj maksimizovati ovu veličinu odnosno povećati minimalan broj jedinica koji se pojavljuje u kodnoj riječi. Stoga se interliver dizajnira tako da ako koder 1 za datu ulaznu sekvencu može da produkuje mali broj jedinica koder 2 koji kodira permutovanu sekvencu treba da produkuje sekvencu sa velikim brojem jedinica. Na ovaj način se izbjegava mogućnost da na izlazu dobijemo sekvencu sa malim brojem jedinica odnosno sa malom Hammingovom težinom što odmah implicira da ne možemo da dobijemo kod koji ima malu Hammingovu distancu. Još ćemo se jednom osvrnuti na funkcionisanje interlivera kod konkretnih kodova nešto kasnije.

11.3. Tipovi kodera kod turbo kodiranja

Važno pitanje koje se postavlja kod dizajna turbo kodova je koje kodere izabrati u gornjoj i donjoj grani. Premda ne postoje čisto teorijski problemi da to budu koderi sasvim prozvoljnog tipa ipak su to po pravilu konvolucioni koderi. U praksi se koriste dva tipa konvolucionih kodera. Predstavnici ova dva tipa su prikazani na slikama dolje. Prvi koder nije sistematski, jer se u izlaznoj sekvenci ne nalaze direktno zastupljeni bitovi koji predstavljaju poruku. Kako je dekodiranje konvolucionih kodova već i onako složeno to je veoma važno da se u izlaznoj sekvenci pojave bitovi ulaza direktno a ne u formi provjera parnosti. Stoga se često koristi druga forma, gdje se izlaz iz jednog od kodera provodi po povratnoj sprezi na ulaz drugog kodera (sistem se ponekad zbog toga zove rekurzivnim) iskoristi kada se želi postići da imamo sistematski kod odnosno da se kodna riječ javi direktno u kodnoj riječi. Druga varijanta se koristi i kod UMTS standarda. Postoji i mnogo značajniji razlog za korišćenje rekurzivnih konvolucionih kodova, a to je činjenica da za visoke kodne odnose ovi kodovi daju znatno manju vjerovatnoću greške nego najbolji koderi nerekurzivnog tipa za bilo koji nivo šuma koji se pojavljuje u kanalu.



11.4. Likelihood odnosi – osnova za turbo dekodiranje

Kada smo radili konvoluciono dekodiranje rekli smo da se osim bitova dekodiranje može obaviti na osnovu "mekih" poruka koje su stigle na ulaz dekodere. Naime, ako se pošalje 1 i 0 mi na ulaz prijemnika dobijamo kontinualne vrijednosti koje uzimaju vrijednosti u nekom intervalu. Obično se uzima prag da je između dvije vrijednosti između 0 i 1, odnosno 0.5. Npr. vrijednosti 0.3 se proglašava kao 0 i sa tom nulom se vrši dekodiranje. Međutim, postoje algoritmi koji mogu da tretiraju ove tzv. "meke" vrijednosti. Tako se 0.3 tretira i dalje ako 0, ali postoji dosta velika vjerovatnoća da je to zapravo 1 koje je pod uticajem šuma palo ispod praga. Isto tako 0.1 se proglašava sa 0, ali ovdje je mnogo manja vjerovatnoća da je to 1 koje je palo ispod praga. Kod turbo dekodiranja se po pravilu vrši prenos putem bipolarne sekvence, što znači da bitu 0 zapravo odgovara negativni naponski nivo dok bitu 1 odgovara pozitivni naponski nivo. Stoga ćemo ponekad ove nivoe nazivati -1 i +1.

Osnovni elemenat koji se koristi prilikom turbo dekodiranja je logaritamski odnos uslovnih vjerovatnoća koji je ponekad naziva "odnos sličnosti" ili "odnos vjerodostojnosti". Umjesto ove terminologije možemo koristiti engleski naziv i odgovarajuću skraćenicu: likelihood ratio (LR) odnosno za logaritam od likelihood odnosa (LLR). Ovaj odnos u nauci ima ogromnu primjenu i jedan je od osnova testiranja hipoteza, teorije estimacije (procjene) itd. Matematička osnova za testiranje hipoteza zasniva se na Bayesovoj teoremi, koja je izvedena da definiše relaciju između združene i uslovne vjerovatnoće događaja A i B kao:

$$P(A|B)P(B)=P(AB)$$

Ova teorema može da se veže i za određivanje aposteriori vjerovatnoće:

$$P(A\,|\,B)=\frac{P(B\,|\,A)P(A)}{P(B)}$$

pomoću koje određujemo vjerovatnoću da je događaj B koji se dogodio uzrokovan događajem A ako znamo vjerovatnoću događaja A, B i uslovnu vjerovatnoću B u zavisnosti od A. Ono što smo često u okviru našeg kursa prenebregavali je činjenica da je signal usljed djelovanja u kanalu izobličen i da to izobličenje na ulaz u prijemnik uzrokuje da je signal različit od vrijednosti 0 i 1

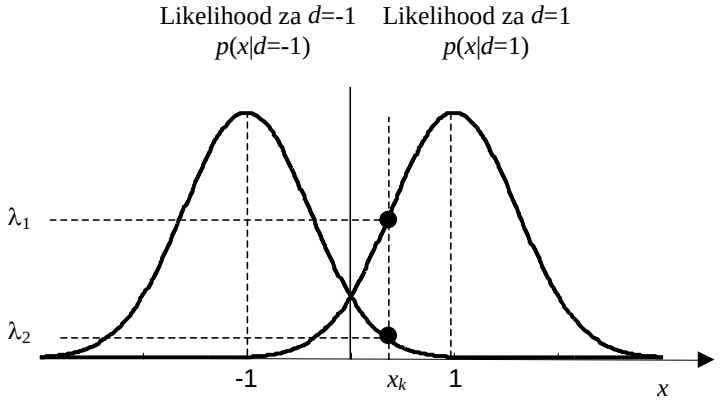
odnosno da može uzeti neke kontinualne vrijednosti u nekom intervalu. Stoga mi možemo zapisati Bayesovu teoremu za kontinualne događaje kao:

$$P(d = i \mid x) = \frac{p(x \mid d = i)P(d = i)}{p(x)}$$

gdje je

$$p(x) = \sum_{i=1}^{M} p(x \mid d = i)P(d = i)$$

Pretpostavljamo da je $d=i$ neki mogući događaj iz skupa događaja – alfabeta (recimo {0, 1}) dok su $p(x)$ i $p(x|d=i)$ funkcije rasporedjele kontinualne slučajne promjenljive x (direktna i uslovna). Iz prethodne relacije je očigledno da je $p(x)$ nezavisno od i. Pretpostavimo da imamo binarni alfabet a da su vrijednosti iz ovog alfabeta prikazane fizičkim vrijednostima +1 i –1 (recimo u voltima, amperima itd). Dakle, pretpostavljamo da d može uzeti te dvije vrijednosti. Za kanal u kome slučajni proces može biti modelovan kao Gausov bijeli šum gdje su uslovne vjerovatnoće $p(x|d=1)$ i $p(x|d=-1)$ prikazane na slici. Ove uslovne vjerovatnoće se ponekad nazivaju funkcijama sličnosti ili vjerodostojnosti (likelihood functions). Na slici se vidi vrijednost x_k i označene su vrijednosti ove dvije likelihoood funkcije za dato x_k (λ_1 i λ_2).



Na osnovu ovih veličina možemo da odredimo kojoj klasi pripada dobijena vrijednost x_k. Ujedno uočavamo prag i taj prag nam služi da izvršimo selekciju gdje sve veće od praga pripada događaju uzrokovanom sa $d=1$ i suprotnom $d=-1$. Slično pravilo je zasnovano na maksimalnoj aposteriori vjerovatnoći (minimalnoj grešci) gdje važi:

$$P(d = 1 \mid x) \underset{H_2}{\overset{H_1}{\gtrless}} P(d = -1 \mid x)$$

Ova jednačina kaže da treba prihvatiti hipotezu H_1 ($d=1$) ako je $P(d=1|x)$ veće od $P(d=-1|x)$ i obrnuto. Pomoću Bayesove formule ovo se može zapisati kao:

$$p(x \mid d = 1)P(d = 1) \underset{H_2}{\overset{H_1}{\gtrless}} p(x \mid d = -1)P(d = -1)$$

Ova relacija se obično zapisuje u obliku likelihood odnosa:

$$\frac{p(x\,|\,d=1)}{p(x\,|\,d=-1)} \underset{H_2}{\overset{H_1}{\gtrless}} \frac{P(d=-1)}{P(d=1)} \qquad \text{ili} \qquad \frac{p(x\,|\,d=1)}{p(x\,|\,d=-1)}\frac{P(d=1)}{P(d=-1)} \underset{H_2}{\overset{H_1}{\gtrless}} 1$$

Uzimajući logaritam likelihood odnosa dobijamo veoma korisnu metriku koja se naziva log-likelihood odnos. Ona se daje u obliku realnog broja koji predstavlja meku odluku na izlazu iz detektora označenu kao $L(d|x)$:

$$L(d\,|\,x) = \log\left[\frac{P(d=1\,|\,x)}{P(d=-1\,|\,x)}\right] = \log\left[\frac{p(x\,|\,d=1)}{p(x\,|\,d=-1)}\frac{P(d=1)}{P(d=-1)}\right]$$

Algebra u domenu log-likelihood odnosa je veoma interesantna i važna posebno u procesu dekodiranja turbo kodova. Posmatrajmo sledeći log-likelihood odnos:

$$L(d) = \log\frac{P(d=1)}{P(d=-1)} = \log\frac{P(d=1)}{1-P(d=1)}$$

Pod pretpostavkom da je osnova logaritma e (da je u pitanju prirodni logaritam) možemo na osnovu log-likelihood odnosa odrediti vjerovatnoću $P(d=1)$ (na istovjetan način se određuju i uslovne vjerovatnoće za odnos $L(d|x)$):

$$P(d=1) = \frac{e^{L(d)}}{1+e^{L(d)}} \qquad \text{dok je} \qquad P(d=-1) = \frac{1}{1+e^{L(d)}}$$

Ovo će nam biti pomoćne relacije u izvođenju log-likelihood odnosa sa $d_1 \oplus d_2$. Ovaj odnos se obično označava kao:

$$\begin{aligned} L(d_1 \boxplus d_2) = L(d_1 \oplus d_2) &= \log\frac{P(d_1 \oplus d_2 = 1)}{P(d_1 \oplus d_2 = -1)} = \\ &= \log\frac{P(d_1=1)P(d_2=-1)+P(d_1=-1)P(d_2=1)}{P(d_1=1)P(d_2=1)+P(d_1=-1)P(d_2=-1)} = \\ &= \log\frac{\frac{e^{L(d_1)}}{1+e^{L(d_1)}}\frac{1}{1+e^{L(d_2)}}+\frac{1}{1+e^{L(d_1)}}\frac{e^{L(d_2)}}{1+e^{L(d_2)}}}{\frac{e^{L(d_1)}}{1+e^{L(d_1)}}\frac{e^{L(d_2)}}{1+e^{L(d_2)}}+\frac{1}{1+e^{L(d_1)}}\frac{1}{1+e^{L(d_2)}}} = \log\frac{e^{L(d_1)}+e^{L(d_2)}}{e^{L(d_1)+L(d_2)}+1} \end{aligned}$$

U realnim sistemima sa porukama koje imaju dužinu od desetina hiljada pa i miliona bitova ovakva relacija koja uključuje računanje logaritma i stepene funkcije je neprihvatljiva. Naime, ove operacije ponekad usporavaju sistem i do 20 puta. Stoga je potrebno izbjeći računanje ovih izraza kad god je to moguće pa čak i po cijenu određene netačnosti. Najčešće se koristi sledeća aproksimacija (mada postoje i preciznije):

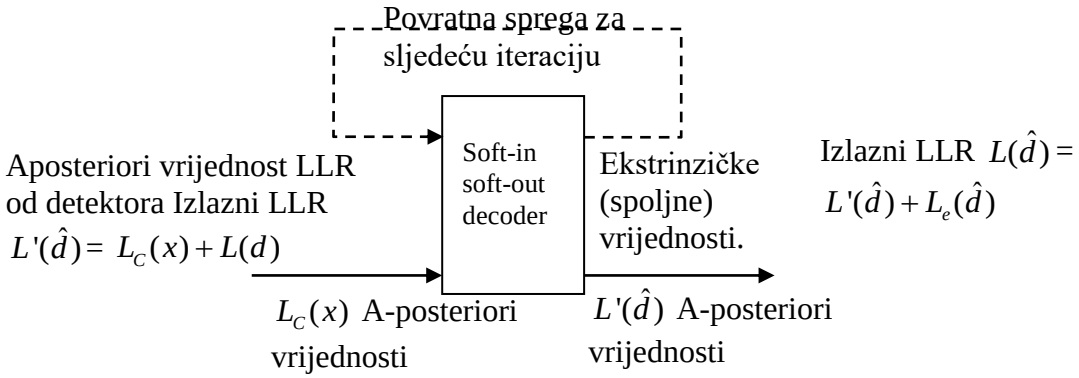
$$L(d_1 \boxplus d_2) \approx (-1)\text{sign}[L(d_1)]\text{sign}[L(d_2)]\min[|L(d_1)|,|L(d_2)|]$$

gdje sign predstavlja funkciju znaka, dok je min minimum dvije vrijednosti. Uzmimo recimo da je $L(d_1)=1$ a $L(d_2)=0.5$. Tačan izraz daje –0.2273 dok približan daje –0.5. Situacija je znatno bolja ako

je razlika likelihooda veća tako da za $L(d_1)$=4 i $L(d_2)$=1 tačan izraz daje –0.9843 dok približan daje –1. Dakle, spremni smo da zbog jednostavnijeg računanja platimo malu nepreciznost u log-likelihood odnosima. Postoje aproksimacije koje daju nešto preciznije vrijednosti log-likelihooda (ako su potrebne) dok istovremeno nešto su složenije za računanje (ali ne koliko bi bilo računanje logaritma i exp funkcija).

11.5. SISO dekoder (srce sistema za dekodiranje)

Dekoderska strana je znatno složenija od strane za kodiranje. Njeno "srce" čini kolo koje se naziva SISO (Soft-In-Soft-Out) dekoder. Ovaj dekoder vrši proračun log-likelihooda za primljenu sekvencu. Prilikom proračuna uzima informaciju o sistematskim bitima i popravlja je na osnovu informacija o bitovima parnosti. Sistem radi iterativnom procedurom



Da bi objasnili kako radi SISO dekoder zapišimo LLR u sledećem obliku:

$$L(d \mid x) = \log\left[\frac{p(x \mid d=1)}{p(x \mid d=-1)}\right] + \log\left[\frac{P(d=1)}{P(d=-1)}\right] = L(x \mid d) + L(d)$$

gdje je $L(x|d)$ log-likelohood odnos izlaza iz kanala x pod uslovom d=1 ili d=-1 dok je $L(d)$ apiori log-likelihood odnos za bite koji su poslati sa predajnika. Da bi pojednostavili ovo se može zapisati kao:

$$L'(\hat{d}) = L_C(x) + L(d)$$

gdje oznaka $L_C(x)$ koji predstavlja član koji označava da log-likelihood odnos je rezultat mjerenja napravljenog na detektoru. Sve prethodne jednačine su uvedene imajući na umu samo dekoder podataka. Međutim, uvodeći informaciju o procesu kodiranja imaćemo neke dobitke i u procesu donošenja odluka. Za sistematske kodove log-likelihood odnos (meki izlaz) $L(\hat{d})$ je jednak:

$$L(\hat{d}) = L'(\hat{d}) + L_e(\hat{d})$$

gdje je $L'(\hat{d})$ log-likelihood odnos za bitove van detektora (ulaz u dekoder) dok $L_e(\hat{d})$ je log-likelihood dobijen ekstra znanjem o procesu dekodiranja. Izlazna sekvenca sistematskog dekodera sastoji se od bitova poruke i bitova parnosti. Stoga se prethodni likelihood odnos može shvatiti kao suma log-likelihood odnosa za poruku i dodatak koji se može dobiti na osnovu bitova parnosti. Konačno možemo zapisati:

$$L(\hat{d}) = L_C(x) + L(d) + L_e(\hat{d})$$

Meka odluka $L(\hat{d})$ je realni broj koji daje "hard" odluku sa odgovarajućom sigurnošću. Znak od $L(\hat{d})$ predstavlja "hard" odluku, odnosno za pozitivne vrijednosti odlučujemo da je d=1 dok u suprotnom donosimo zaključak da je d=-1. Apsolutna vrijednost $L(\hat{d})$ daje nam sa kolikom sigurnošću je odluka donesena. U prvoj iteraciju u dekodiranju SISO dekoder obično pretpostavlja da su svi binarni podaci jednakovjerovatni, dajući inicijalnu apriori log likelihod vrijednost $L(d)$=0 za treći član u prethodnoj jednakosti. Predetekciona vrijednost kanala $L_C(x)$ je određena formirajući logaritam odnosa λ_1 i λ_2. Izlaz $L(\hat{d})$ je kreiran na osnovu LLR za dekoder, $L'(\hat{d})$, spoljnjeg LLR izlaza $L_e(\hat{d})$ koji predstavlja znanje do kojega se došlo u procesu dekodiranja. Tokom procesa iterativnog dekodiranja spoljni likelihood se povratnom spregom ponovo dovodi na ulaz dekodera u cilju poboljšavanja apriori vrijednosti u narednoj iteraciju. Posmatrajmo pravougaoni kod sa slike. Konfiguracija se može opisati kao matrica koja se sastoji od k_1 redova i k_2 kolona kojoj se dodaju biti parnosti. Različiti djelovi strukture su označeni sa d za podatke, p_h za horizontalne bitove parnosti, i p_v za vertikalne bitove parnosti. Dodatno postoje blokovi označeni sa L_{eh} i L_{ev} koji sadrže spoljne LLR vrijednosti određene na za horizontalni i vertikalni dekodirajući korak respektivno. Napomenimo da pravougaoni kod je jednostavan primjer koji nije od naročite praktične važnosti, ali može da posluži da na računskom primjeru objasnimo proces dekodiranja. Njegova struktura se sastoji od dva odvojena dekodirajuća koraka – horizontalnog i vertikalnog. Algoritam za iterativno dekodiranje za pravougaoni kod se sastoji od sledećih koraka:

1. Postaviti apriori informacije $L(d)$=0.
2. Dekodirati horizontalno i odrediti horizontalno spoljnu informaciju:

$$L_{eh}(\hat{d}) = L(\hat{d}) - L_C(x) - L(d)$$

3. Postavi se $L(d) = L_{eh}(\hat{d})$.
4. Dekodirati vertikalno i dobiti:

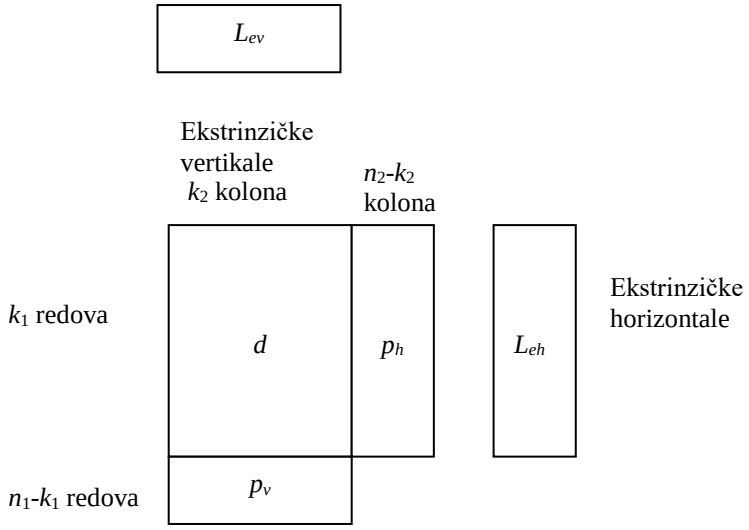
$$L_{ev}(\hat{d}) = L(\hat{d}) - L_C(x) - L(d)$$

5. Postavi se $L(d) = L_{ev}(\hat{d})$.

6. Ako postoji dovoljno iteracija da se donese validna odluka preći na korak 7 a u suprotnom ponoviti korak 2.

7. Meki izlaz se računa kao:

$$L(\hat{d}) = L_C(x) + L_{eh}(\hat{d}) + L_{ev}(\hat{d})$$



Kod predmetnog pravougaonog koda važe sledeće relacije:

$$d_i \oplus d_j = p_{ij} \qquad d_i = d_j \oplus p_{ij} \qquad d_j = d_i \oplus p_{ij}$$

Bitovi koji se prenose se prikazuju kao sekvenca $d_1d_2d_3d_4p_{12}p_{34}p_{13}p_{24}$. Na ulaz prijemnika primljeni simboli su prikazani kao sekvenca $[x_i, x_{ij}]$ gdje je $x_i=d_i+n$ za svaki primljeni podatak $x_{ij}=p_{ij}+n$ za svaki primljeni simbol parnosti i n predstavlja nezavisni i jednakodistribuirani šum. Zbog notacione jednostavnosti označićemo primljenu riječ sa jednim indeksom $\{x_k\}$ gdje k se može tretirati kao vremenski indeks. Koristivši prethodne relacije i pretpostavljajući da je aditivni šum Gausovski možemo pisati LLR za mjerenja kanala kao:

$$L_C(x_k) = \log\left[\frac{p(x_k \mid d=1)}{p(x_k \mid d=-1)}\right] = \log\left[\frac{\frac{1}{\sigma\sqrt{2\pi}}\exp\left[-\frac{1}{2}\frac{(x_k-1)^2}{\sigma^2}\right]}{\frac{1}{\sigma\sqrt{2\pi}}\exp\left[-\frac{1}{2}\frac{(x_k+1)^2}{\sigma^2}\right]}\right] =$$

$$= -\frac{1}{2}\left(\frac{x_k-1}{\sigma}\right)^2 + \frac{1}{2}\left(\frac{x_k+1}{\sigma}\right)^2 = \frac{2}{\sigma^2}x_k$$

Ako izvršimo dalje pojednostavljenje ovog izraza postavljajući da je $\sigma^2=1$ dobijamo:

$$L_c(x_k)=2x_k$$

Posmatrajmo sledeći primjer gdje je sekvenca podataka $d_1d_2d_3d_4$ kreirana od binarnih cifara 1001 kao što je prikazano u tabeli. Na osnovu uvedenih relacija slijedi da su biti parnosti: $p_{12}p_{34}p_{13}p_{24}$ jednaki 1111. Dakle, poslata sekvenca je jednaka:

$$[d_i, p_{ij}]=1\ 0\ 0\ 1\ 1\ 1\ 1\ 1$$

$d_1=1$	$d_2=0$	$p_{12}=1$
$d_3=0$	$d_4=1$	$p_{34}=1$
$p_{13}=1$	$p_{24}=1$	

Neka je poruka prenešena sa bipolarnim impulsima +1 –1 –1 +1 +1 +1 +1 +1 i neka zbog šuma u prenosu dobijena poruka ima oblik

$$[x_k]=0.75\ 0.05\ 0.10\ 0.15\ 1.25\ 1.00\ 3.00\ 0.50$$

Log likelihood odnos je u ovom slučaju:

$$[L_c(x_k)]=1.50\ 0.10\ 0.20\ 0.30\ 2.50\ 2.00\ 6.00\ 1.00$$

Dekoderski ulazni log-likelihood odnosi su dati u narednoj tabeli.

$L_C(x_1)=1.5$	$L_C(x_2)=0.1$	$L_C(x_{12})=2.5$
$L_C(x_3)=0.2$	$L_C(x_4)=0.3$	$L_C(x_{34})=2.0$
$L_C(x_{13})=6.0$	$L_C(x_{24})=1.0$	

Zbog toga što je na drugom i trećem bitu log-likelihood odnos pozitivan ovaj sistem bi imao greške na ta dva bita. Da bi objasnili iterativni turbo postupak za dekodiranje uvešćemo nekoliko osnovnih elemenata log-likelihood algebre. Prilikom računanja log-likelihood odnosa korišćena je prethodno uvedena aproksimacija:

$$L(d_1) \mp L(d_2) \approx (-1)\text{sign}[L(d_1)]\text{sign}[L(d_2)]\min(|L(d_1)|,|L(d_2)|)$$

Za uvedeni produktni kod koristimo relaciju da se meki izlaz $L(\hat{d}_1)$ za podatke d_1 računa kao:

$$L(\hat{d}_1) = L_c(x_1) + L(d_1) + ([L_c(x_2) + L(d_2)] \mp L_c(x_{12}))$$

gdje je $([L_c(x_2) + L(d_2)] \mp L_c(x_{12}))$ spoljna (ekstrinzička) vrijednost koja pomaže kodu (signal koji odgovara podacima d_2 i njegovoj apriori vjerovatnoći u konjukciji sa signalom koji predstavlja bite parnosti p_{12}). U opštem slučaju meki izlaz $L(\hat{d}_i)$ za primljeni signal d_i koji odgovara podacima je:

$$L(\hat{d}_i) = L_c(x_i) + L(d_i) + ([L_c(x_j) + L(d_j)] \mp L_c(x_{ij}))$$

gdje su $L_c(x_i)$, $L_c(x_j)$ i $L_c(x_{ij})$ log-likelihood vrijednosti kanala i primljenih signala koji odgovaraju d_i, d_j i p_{ij}. $L(d_i)$ i $L(d_j)$ su log-likelihood odnosi apriornih vjerovatnoća za d_i i d_j respektivno dok je $([L_c(x_j) + L(d_j)] \mp L_c(x_{ij}))$ spoljni doprinos koda. Na primjer, meki izlaz $L(\hat{d}_1)$ je prikazan sa detektorom log-likelihood odnosa mjere primljenog signala koji odgovara podacima d_1 i spoljne log-likelihood dobijene na osnovu činjenice da podatak d_2 i parnost p_{12} takođe daju znanje o podatku d_1. Na primjer, horizontalne kalkulacije $L_{eh}(\hat{d})$ i vertikalne kalkulacije $L_{ev}(\hat{d})$ se izražavaju kako slijedi:

$$\begin{aligned}
L_{eh}(\hat{d}_1) &= [L_c(x_2) + L(d_2)] \mp L_c(x_{12}) \\
L_{ev}(\hat{d}_1) &= [L_c(x_3) + L(d_3)] \mp L_c(x_{13}) \\
L_{eh}(\hat{d}_2) &= [L_c(x_1) + L(d_1)] \mp L_c(x_{12}) \\
L_{ev}(\hat{d}_2) &= [L_c(x_4) + L(d_4)] \mp L_c(x_{23}) \\
L_{eh}(\hat{d}_3) &= [L_c(x_4) + L(d_4)] \mp L_c(x_{34}) \\
L_{ev}(\hat{d}_3) &= [L_c(x_1) + L(d_1)] \mp L_c(x_{13})
\end{aligned}$$

$$L_{eh}(\hat{d}_4) = [L_c(x_3) + L(d_3)] \boxplus L_c(x_{34})$$
$$L_{ev}(\hat{d}_4) = [L_c(x_2) + L(d_2)] \boxplus L_c(x_{24})$$

Kada uvrstimo log-likelihood vrijednosti iz tabele u izraze za $L_{eh}(\hat{d})$ uz inicijalno postavljanje $L(d)$ vrijednosti na nulu dobijamo:

$$L_{eh}(\hat{d}_1) = (0.1 + 0) \boxplus 2.5 = -0.1 = \text{novi } L(d_1)$$
$$L_{eh}(\hat{d}_2) = (1.5 + 0) \boxplus 2.5 = -1.5 = \text{novi } L(d_2)$$
$$L_{eh}(\hat{d}_3) = (0.3 + 0) \boxplus 2.0 = -0.3 = \text{novi } L(d_3)$$
$$L_{eh}(\hat{d}_4) = (0.2 + 0) \boxplus 2.0 = -0.2 = \text{novi } L(d_4)$$

gdje je log-likelihood sabiranje vršeno upotrebom uvedene aproksimacije. Sada obavljamo računanje prve vertikalne log-likelihood vrijednosti $L_{ev}(\hat{d})$. Za vrijednosti $L(d)$ koristimo vrijednosti koje su sračunate u prethodnom koraku za $L_{eh}(\hat{d})$:

$$L_{ev}(\hat{d}_1) = (0.2 - 0.3) \boxplus 6.0 = 0.1 = \text{novi } L(d_1)$$
$$L_{ev}(\hat{d}_2) = (0.3 - 0.2) \boxplus 1.0 = -0.1 = \text{novi } L(d_2)$$
$$L_{ev}(\hat{d}_3) = (1.5 - 0.1) \boxplus 6.0 = -1.4 = \text{novi } L(d_3)$$
$$L_{ev}(\hat{d}_4) = (0.1 - 1.5) \boxplus 1.0 = 1.0 = \text{novi } L(d_4)$$

Rezultati prve pune iteracije za dva dekodirajuća koraka (horizontalni i vertikalni) je prikazan u narednim tabelama.

1.5	0.1
0.2	0.3

$L_c(x_k)$

-0.1	-1.5
-0.3	-0.2

$L_{eh}(\hat{d})$ nakon prvog horizontalnog koraka

0.1	-0.1
-1.4	1.0

$L_{ev}(\hat{d})$ nakon prvog vertikalnog koraka

Svaki dekodirajući korak popravlja originalni log-likelihood odnos. Ovo se može vidjeti na osnovu računanja log-likelihood odnosa pomoću uvedene aproksimacije. Originalni log-likelihood odnos plus horizontalni plus spoljni log-likelihod odnos daje popravku u narednoj tabeli (spoljne vertikalne vrijednosti nijesu još posmatrane):

1.4	-1.4
-0.1	0.1

Popravka log-likelihood odnosa uzrokovana $L_{eh}(\hat{d})$

Originalni log-likelihood odnos sa dodatim horizontalnim i vertikalnim spoljnim vrijednostima daje popravku iz naredne tabele.

1.5	-1.5
-1.5	1.1

Popravka loglikelihood odnosa zbog $L_{eh}(\hat{d}) + L_{ev}(\hat{d})$

Za ovaj primjer možemo da vidimo da pomoću isključivo horizontalnih bitova parnosti dobijamo korektnu (hard) odluku o izlazu iz dekodera ali sa malom sigurnošću za bite d_3 i d_4. Nakon što popravimo dekoderom log-likelihood odnos sa vertikalnim spoljnim odnosom novo log-likelihood rješenje daje veću sigurnost. Posmatrajmo jednu dodatnu horizontalnu i vertikalnu dekodirajuću iteraciju da vidimo da li postoji neko dodatno značajno uvećanje rezultata. Ponavljamo praktično istu proceduru koja je prethodno odrađena:

$$L_{eh}(\hat{d}_1) = (0.1 - 0.1) \mp 2.5 = 0 = \text{nova } L(d_1)$$
$$L_{eh}(\hat{d}_2) = (1.5 + 0.1) \mp 2.5 = -1.6 = \text{nova } L(d_2)$$
$$L_{eh}(\hat{d}_3) = (0.3 + 0.1) \mp 2.0 = -1.3 = \text{nova } L(d_3)$$
$$L_{eh}(\hat{d}_4) = (0.2 - 1.4) \mp 2.0 = 1.2 = \text{nova } L(d_4)$$

U sledećem koraku nastavljamo sa drugom vertikalnom kalkulacijom da bi izračunali $L_{ev}(\hat{d})$ koristivši novu $L(d)$ iz druge horizontalne relacije:

$$L_{ev}(\hat{d}_1) = (0.2 - 1.3) \mp 6.0 = 1.1 = \text{nova } L(d_1)$$
$$L_{ev}(\hat{d}_2) = (0.3 + 1.2) \mp 1.0 = -1.0 = \text{nova } L(d_2)$$
$$L_{ev}(\hat{d}_3) = (1.5 + 0) \mp 6.0 = -1.5 = \text{nova } L(d_3)$$
$$L_{ev}(\hat{d}_4) = (0.1 - 1.6) \mp 1.0 = 1.0 = \text{nova } L(d_4)$$

Nakon druge iteracije sa horizontalnom i vertikalnom dekodirajućim proračunima meki izlaz likelihood vrijednosti se ponovo računa kao:

$$L(\hat{d}) = L_c(x) + L_{eh}(\hat{d}) + L_{ev}(\hat{d})$$

Finalna horizontalna i vertikalna spoljna vrijednost i rezultujući dekodirajuća log-likelihood vrijednost su prikazani u tabelama dolje. U ovom primjeru druga kompletna iteracija za oba koda (ukupno 4 iteracije) sugeriše umjereno poboljšanje u odnosu na jednu iteraciju.

1.5	0.1
0.2	0.3

$L_c(x_k)$

0	-1.6
-1.3	1.2

$L_{eh}(\hat{d})$ nakon drugog horizontalnog koraka

1.1	-1.0
-1.5	1.0

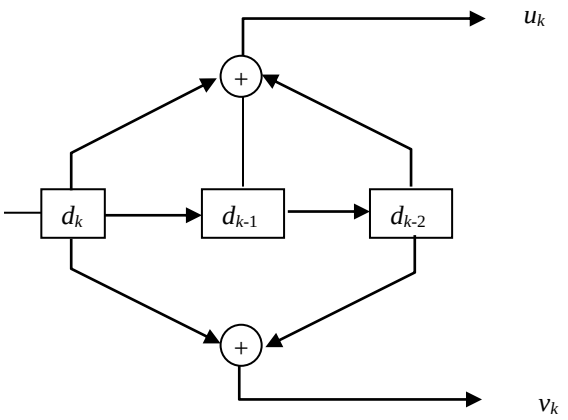
$L_{ev}(\hat{d})$ nakon drugog vertikalnog koraka

Meki izlazi su:

2.6	-2.5
-2.6	2.5

11.6. Turbo kodovi korak bliže stvarnoj realizaciji

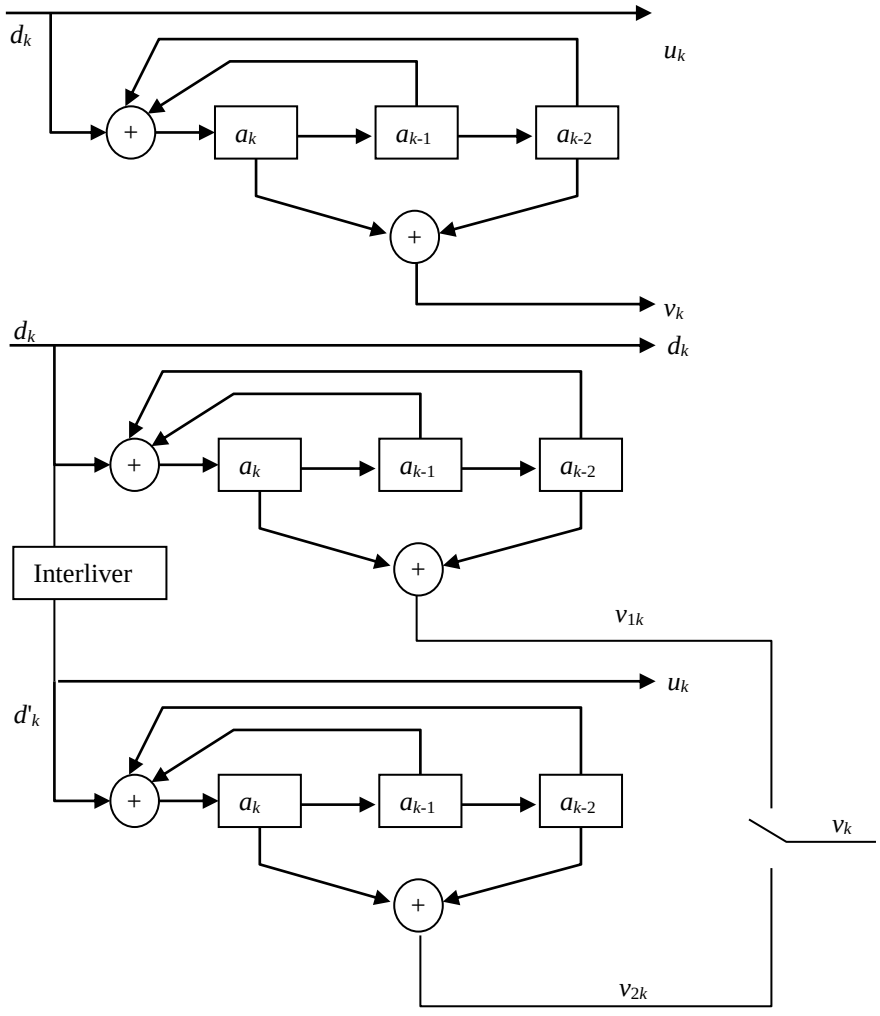
Nakon što smo opisali osnovne koncepte nadovezivanja, iteracija i mekog odlučivanja korišćenjem produktnog koda, sada ćemo ove ideje primjeniti na implementaciju turbo kodova. Naravno, daćemo samo osnovne grube informacije o ovom procesu bez želje da ga previše detaljno proučavamo. Turbo kod ćemo formirati kao paralelno nadovezivanje komponenti kodova. Prvo, ostatak od jednostavnog binarnog konvolucionog kodera sa R=1/2 sa ograničenom dužinom K i memorijom K-1. Ulaz u koder u trenutku k je bit d_k, i odgovarajuća kodna riječ je par bitova (u_k, v_k), gdje su koeficijenti pojedinih generatorskih polinoma dati kao $G_1=[g_{1i}]$ i $G_2=[g_{2i}]$. Jedan mogući ovakav koder je dat na slici dolje. Napominjemo da je dati koder nesistematski i da ima konačni impulsni odziv (jedan ulazni bit ima uticaj na konačan broj izlaznih bita u ovom slučaju 3).



U ovom slučaju ograničenje dužine je K=3, i dva generatora koda su opisana sa G_1=[111] i G_2=[101]. Napominjemo da to koji je od metoda kodiranja (nesistematski ili sistematski sa beskonačnim odzivom odnosno za rekurzivne sistematske kodove) utiče ne toliko na proces dekodiranja, već na performanse dekodiranja za signale zahvaćene šumom. Pokazuje se da za veliki odnos signal-šum (mali uticaj šuma) nesistematski kodovi rade znatno bolje dok za mali odnos signal-šum (veliki uticaj šuma) bolje rade sistematski rekurzivni koderi. Kako se turbo kodovi ipak dizajniraju da rade za složenije uslove rada onda se češće primjenjuju sistematski kodovi, s time što neki sistemi posjeduju mogućnost da vrše "prebacivanje" sa jednog na drugi kodirajući metod u zavisnosti od stanja u kanalu. Primjer sistematskog rekurzivnog kodera koji se koristi kod turbo kodiranja kao i način na koji se dva ovakva kodera mogu povezati u turbo koder su prikazani na narednim slikama.

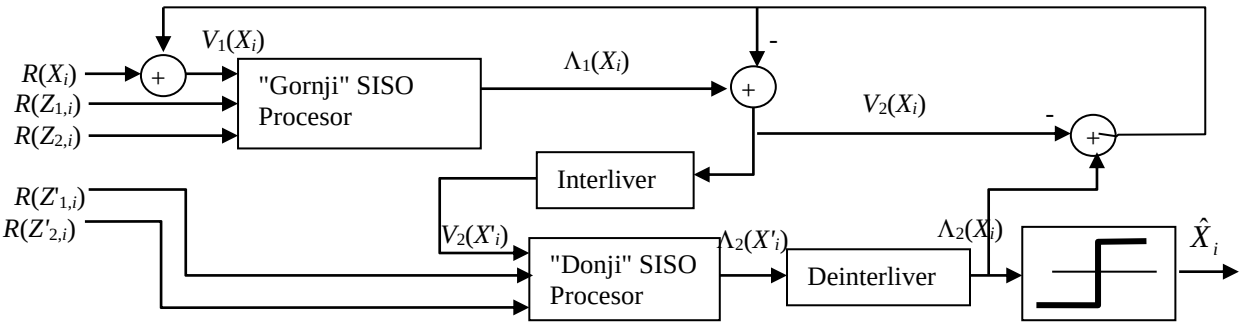
Već smo ukazali na razloge zbog kojih je važan interliver i zbog čega ovakav kodni sistem dobro radi kod rekurzvnih kodova. Ipak, zbog značaja, vrijedi ovo još jednom apostrofirati. Da bi neki kodni sistem imao dobre performanse u otklanjanju grešaka mora da ima što je veću moguću minimalnu Hammingovu distancu a za kodove kod kojih ulaz sa svim nulama produkuje izlaz sa svim nulama (kao što su naši koderi) to se svodi na cilj da imamo kod koji ima što je veću moguću minimalnu Hammingovu težinu. Odnosno cilj je da broj jedinica u riječi koja ih ima najmanje bude što je veći moguć. Kod našeg sistema u pojedinačnim koderima situacija da se na izlazu pojavi riječ sa malim brojem jedinica se ne može izbjeći ali se može izbjeći da istovremeno oba kodera produkuju riječ sa malim brojem jedinica. Uparivanje riječi sa malim brojem jedinica se izbjegava

pažljivim dizajnom interlivera. Interliver može da ima dobru kontrolu nad težinom kodne riječi samo ako je kod rekurzivan. Kod rekurzivnih kodova ono što je na prvi pogled najgora ulazna sekvenca (sekvenca sa jednom jedinicom i ostalim nulama) nije kritična jer rekurzivni kodovi rade na principu filtara sa beskonačnim impulsnim odzivom, odnosno jedna jedinica će produkovati beskonačno mnogo jedinica na izlazu. Suprotno logici povećanje broja jedinica u ulaznoj sekvenci ne mora da produkuje izlazne riječi sa većim brojem jedinica. Naime, može da se dogodi ono što se kod teorije filtara sa beskonačnim impulsnim odzivom naziva "poništavanja para nula-pol" i da za neke ulazne sekvence dobijemo konačan impulsni odziv. Za određenu ulaznu sekvencu može se dogoditi da na izlazu dobijemo relativno mali broj jedinica na izlazu. Kod predmetnog sistema jedna ulazna jedinica daje na izlazu beskonačan odziv. Kritične sekvence su sa dvije jedinice koje su razmaknute sa dvije nule (svi ostali bitovi su nule) i sa tri uzastopne jedinice i ostalim nulama. Dakle, interliver u predmetnom slučaju mora biti dizajniran tako da za date slučajeve ne produkuje u drugoj grani ulaznu sekvencu koja ima iste osobine!



U sekciji 11.5 smo demonstrirali na primjeru jednog jednostavnog koda osnovnu ideju turbo dekodiranja. Nadamo se da je ta ideja jasna. Na žalost, fundament je jedno a stvarna realizacija nešto sasvim drugo. Mi u stvarnu realizaciju ovog sistema zapravo nećemo ni ulaziti iz razloga što za predmetne konvolucione kodove koji su dio sistema turbo kodiranja taj dekodirajući stepen sadrži komplikovane tehnike za dekodiranje koje su naslijeđene iz konvolucionih kodova a koje su u poglavlju sa konvolucionim kodovima već opisani i ilustrovani primjerom. Najčešći oblik sistema za dekodiranje je zapravo Viterbijev algoritma kojega smo ranije upoznali, odnosno već smo se susreli sa njegovom relativnom složenošću. Da bi situacija bila dodatno komplikovanija ovdje moramo da kombinujemo preko SISO procesora rezultate dekodiranja dva konvoluciona koda pa stoga da bi se izašlo u susret svim ovim izazovima razvijene su posebne varijante Viterbijevog

algoritma za dekodiranje kao što je npr. SOVA (Soft-Output-Viterbi-Algorthm). U detalje ovog i većeg broja drugih algoritama koji su razvijeni za predmetnu namjenu nećemo ulaziti. Na slici je prikazana blok shema sistema za dekodiranje turbo kodova.

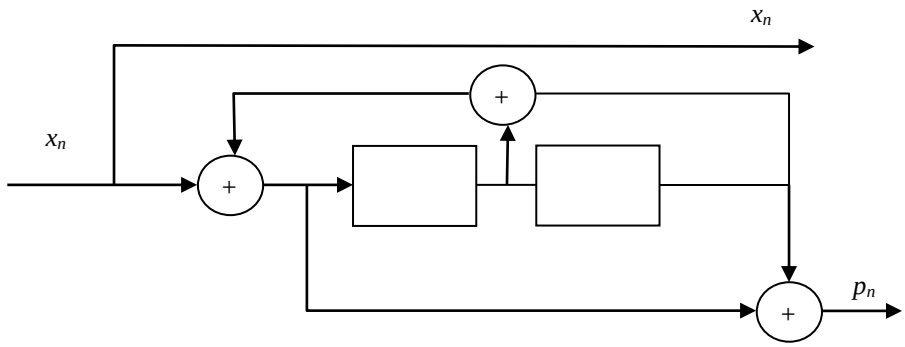


11.7. Zaključak o turbo kodovima

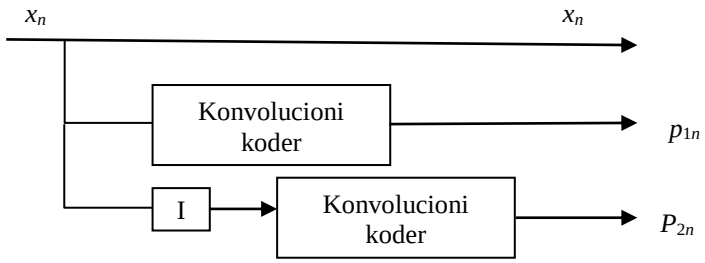
Demonstrirali smo turbo kodove kao jednu od najvažnijih novijih tehnika u razvoju kodova za dekodiranje pogreški. Rad turbo kodova a posebno dekodiranje ovih kodova smo opisali na principskom nivou i dali jedan ilustrativni primjer za koji se nadamo da otkriva čitaocima kvalitete ovih kodova. Nadamo se da studenti koji pročitaju ovaj kratak materijal imaju dovoljno elemenata da nastave da izučavaju ovu materiju samostalno.

Zadaci za vježbu

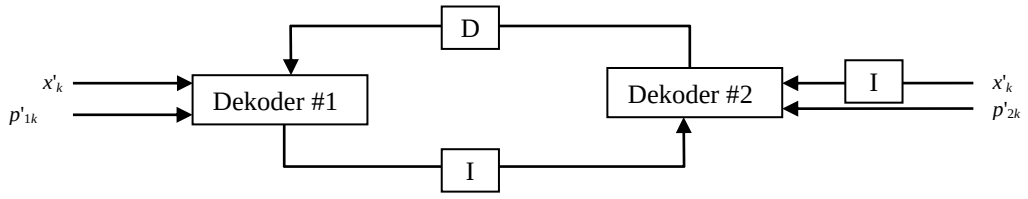
11.1. Dat je rekurzivni sistemski koder sljedećeg oblika:



Dat je turbo kod sljedećeg oblika:



Dekoder ovog sistema je zasnovan na maksimalnoj aposteriori vjerovatnoći u sljedećem obliku:



Blok **I** je interliver isti kao u koderu dok je **D** deinterliver odnosno suprotan blok koji vraća signal u "normalu". Pretpostaviti da je interliver formiran na taj način što se uzima niz od po 8 odbiraka signala i da se oni preuređuju tako da prvi odbirak ide na posljednju poziciju, drugi i treći pomjeraju na treću i četvrtu poziciju, četvrta pozicija se pomjera na šestu, peta ostaje na svom mjestu, šesta se pomjera na drugu poziciju, sedma ostaje na svom mjestu dok osmi bit ide na poziciju prvog bita.

(a) Opisati prelaze između stanja u zavisnosti od ulaza i stanja u registrima.
(b) Prikazati trelis stanja u ćelijama registra.
(c) Prikažite interliver šematski i putem matrice permutacije.
(d) Razmotrite sistem za određivanje odgovarajuće likelihood algebra odnosno algoritam dekodiranja sistema.
(e) Kodirajte sekvencu $\{X_1, X_2, X_3, X_4, X_5, X_6\} = \{1,1,0,0,1,0\}$. Nedostajuće bite postavite na nulu.
(f) Kodirani signal je prošao kroz kanal sa Gausovim aditivnim šumom sa jediničnom varijansom. Neka je konkretna realizacija šumnog procesa data kao:

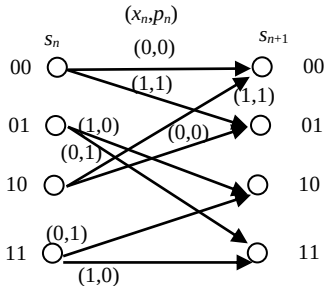
1.966099	2.132927	-0.701887
-1.232363	-0.443420	-0.696641
0.750745	0.823265	0.823463
1.832447	-0.088392	1.036052
-1.262811	0.551007	-2.051227
0.205224	0.277622	0.462560
-0.569778	0.978633	1.105726
0.257169	0.465353	-0.700940

Kolone predstavljaju signal koji dolazi iz pojedinih grana kodera dok su vrste pojedini intervali. Dekodirati primljenu poruku.
(g) Napišite MATLAB program koji realizuje turbo dekoder.

Rješenje. (a) U narednoj tabeli prikazano je stanje na ulazu x_n, početna stanja u ćelijama registrima, naredna stanja u ćelijama registra i stanje na izlazu p_n.

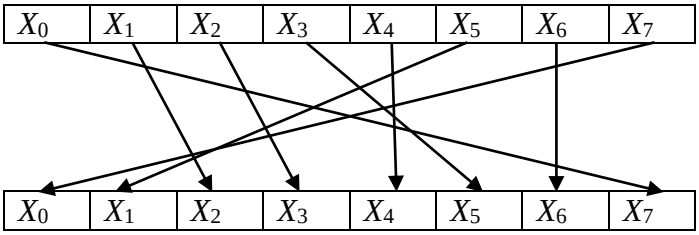
x_n	Prethodno stanje	Naredno stanje	p_n
0	00	00	0
1	00	10	1
0	01	10	0
1	01	00	1
0	10	11	1
1	10	01	0
0	11	01	1
1	11	11	0

(b) Pošto sistem ima dvije ćelije u registru to znači da je ukupno 4 stanja. Prikažimo jednu laticu trelisa.



Nadovezivanjem trelisa možemo da dobijemo generisanu kodnu riječ odnosno navedeni trelis može da se koristi za dekodiranje koda.

(c) Prikažimo rad interlivera šematski:



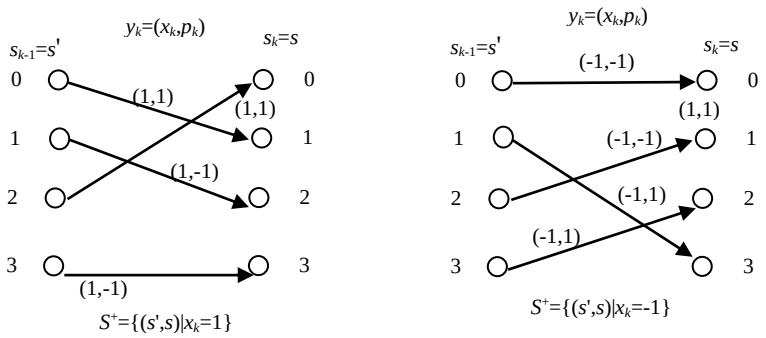
Matrica tranzicije sistema se može opisati na sljedeći način:

$$\mathbf{I} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(d) Za dekodiranje ćemo koristiti likelihood odnose kao:

$$L(x_k \mid y') = \log\left[\frac{P(x_k = 1 \mid y')}{P(x_k = -1 \mid y')}\right] = \log\left[\frac{\sum_{(s',s)\in S^+} P(s_{k-1} = s', s_k = s, y')}{\sum_{(s',s)\in S^-} P(s_{k-1} = s', s_k = s, y')}\right]$$

U imeniocu se nalaze sve kombinacije dvobita koji daju rezultat 0 (odnosno -1) dok se u brojiocu nalaze sve kombinacije dvobita (odnosno njihovih vjerovatoća) koje daju na izlazu 1. Prelazi za ova dva stanja su prikazani na dijagramima.



Predmetni dijagrami su suštinski bipolarna verzija trelisa koji je prethodno prikazan ali sada razdvojeni za stanja +1 i -1.

Vjerovatnoća svakog pojedinačnog stanja se može izraziti Bayesovom teoremom putem sljedećih veza:

$$P(s_{k-1}=s', s_k=s, y'_{1,K}) = P(s_{k-1}=s', y'_{1,k-1})P(s_k=s, y'_k \mid s_{k-1}=s')P(y'_{k+1,K} \mid s_k=s) = \\ = \alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)$$

Sada se likelihood odnos može zapisati kao:

$$L(x_k \mid y') = \log\left[\frac{\sum_{(s',s)\in S^+} \alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)}{\sum_{(s',s)\in S^-} \alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)}\right]$$

Član $\alpha_k(s')$ je vjerovatnoća dolaska u granu u partikularnom stanju i niz šumnih observacija $y'_{1,k} = y'_1, y'_2, ..., y'_k$ koje su ga dovele u navedeno stanje. Ova vjerovatnoća se može dobiti sabiranjem preko svih putanja koje vode u navedeno stanje:

$$\alpha_k(s) = P(s_k=s, y'_{1,k}) = \sum_{s'} P(s_{k-1}=s', y'_{1,k-1})P(s_k=s, y'_k \mid s_{k-1}=s') = \sum_{s'} \alpha_{k-1}(s')\gamma_k(s',s)$$

Inicijalizacija se može izvršiti tako što se uzme (kako je uobičajeno) da svi turbo kodovi započinju u stanju 0:

$$\alpha_0(s) = P(s_0=s) = 1 \quad s=0 \\ s\neq 0$$

Član $\beta_k(s)$ je vjerovatnoća postojeće grane preko posmatranog stanja s i niza šumnih opservacija $y'_{k+1,K} = y'_{k+1}, y'_{k+2}, ..., y'_K$ koje završavaju sa trelisom. Sumiranjem preko svih putanja koje postoje u stanju mi idemo nazad u rekurziju računajući $\beta_k(s)$ u obliku vrijednosti $\gamma_k(s',s)$:

$$\beta_k(s') = P(y'_{k+1,K} \mid s_k=s) = \sum_s P(s_{k+1}=s, y'_{k+1} \mid s_k=s')P(y'_{k+2,K} \mid s_k=s') = \sum_s \gamma_{k+1}(s',s)\beta_{k+1}(s)$$

I u ovom slučaju je neophodno izvršiti odgovarajuću inicijalizaciju. Prvi koder obično završava u stanju 0 međutim finalno stanje drugog konvolucionog kodera je zavisno od poruke pa stoga smo se

opredijelili da pretpostavimo da je konačno stanje drugog dekodera uniformno raspoređeno preko svih mogućih dozvoljenih stanja:

Prvi dekoder

$$\beta_K(s) = P(s_K = s) = \begin{cases} 1 & s = 0 \\ 0 & s \neq 0 \end{cases}$$

Drugi dekoder

$$\beta_K(s) = P(s_K = s) = \frac{1}{2^v} \quad \text{za svako } s$$

Sada možemo da dobijemo eksplicitni izraz za računanje $\gamma_k(s', s)$ koji se može računati u dekoderu. Za dato stanje preneseni signal je bit podataka i bitovima parnosti y_k. Za dato početno stanje naredno stanje je potpuno određeno sa bitom podataka. Koristivši Bayesovu formula vjerovatnoća u grani se može izraziti kao:

$$\gamma_k(s', s) = P(s_k = s, y'_k \mid s_{k-1} = s') = P(y'_k \mid s_{k-1} = s', s_k = s) P(s_k = s \mid s_{k-1} = s') = P(y'_k \mid y_k) P(x_k)$$

Vjerovatnoća da bit podataka uzme datu vrijednost može se izraziti putem log-likelihood odnosa kao:

$$P(x_k) = \frac{\exp\left[\frac{1}{2} L_a(x_k)\right]}{1 + \exp\left[\frac{1}{2} L_a(x_k)\right]} \exp\left[\frac{1}{2} x_k L_a(x_k)\right] = B_k \exp\left[\frac{1}{2} x_k L_a(x_k)\right]$$

gdje je

$$L_a(x_k) = \log\left[\frac{P(x_k = 1)}{P(x_k = -1)}\right]$$

Vjerovatnoća da dobijemo dati par simbola poruke i simbola parnosti koji su zahvaćeni greškama (šumom) može se opisati Gausovskom distribucijom:

$$P(y'_k \mid y_k) = P(x'_k \mid x_k) P(p'_k \mid p_k) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x'_k - x_k)^2}{2\sigma^2}\right] \Delta \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(p'_k - p_k)^2}{2\sigma^2}\right] =$$

$$= A_k \exp\left[\frac{x'_k x_k - p'_k p_k}{\sigma^2}\right]$$

Vrijednost Δ predstavlja malu zonu oko navedene vrijednosti koja se uzima da bi se izvršilo aproksimiranje kontinualne funkcije gustine vjerovatnoće sa pravougaonikom da bi dobili diskretnu slučajnu promjenljivu. Ne treba previše brinuti oko ovog člana jer će on i nestati u dijeljenju koje je sastavni dio proračuna log-likelihod odnosa. Sada se može izraziti tranziciona vjerovatnoća $\gamma_k(s', s)$ u obliku log-likelihood odnosa i šumnih observacija kao:

$$\gamma_k(s', s) = A_k B_k \exp\left[\frac{1}{2}(x_k L_a(x_k) + x_k L_c x'_k + p_k L_c p'_k)\right] \quad L_c = \frac{2}{\sigma^2}$$

Na osnovu estimatora sa maksimalnom aposteriori vjerovatnoćom slijedi:

$$L_{map}(x_k) = \log\left[\frac{P(x_k = 1 \mid y')}{P(x_k = -1 \mid y')}\right] = \log\left[\frac{P(y'_k \mid x_k = 1)}{P(y'_k \mid x_k = -1)}\right] + \log\left[\frac{P(x_k = 1)}{P(x_k = -1)}\right]$$

Na osnovu prethodnih izvođenja sada slijedi:

$$L_{map}(x_k) = \log\left[\frac{\sum_{(s,s')\in S^+} \alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)}{\sum_{(s,s')\in S^-} \alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)}\right]$$

$$= \log\left[\frac{\sum_{(s',s)\in S^+} \exp\left(\frac{L_a(x_k)}{2}\right)\exp\left(\frac{L_c x'_k}{2}\right)\alpha_{k-1}(s')\exp\left(\frac{p_k L_c p'_k}{2}\right)\beta_k(s)}{\sum_{(s',s)\in S^-} \exp\left(-\frac{L_a(x_k)}{2}\right)\exp\left(-\frac{L_c x'_k}{2}\right)\alpha_{k-1}(s')\exp\left(\frac{p_k L_c p'_k}{2}\right)\beta_k(s)}\right]$$

Napomenimo da sumiranje u brojioca preko svih stanja tranzicija prenesenih sa bitovima simbola jednakim 1 i sumiranje u brojiocu sa simbolom podataka -1. Imamo:

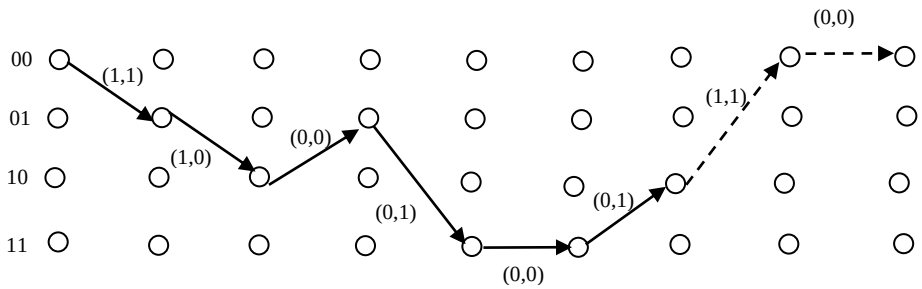
$$L_{map}(x_k) = L_a(x_k) + L_c x'_k + \log\left[\frac{\sum_{(s,s')\in S^+} \alpha_{k-1}(s')\exp\left(\frac{p_k L_c p'_k}{2}\right)\beta_k(s)}{\sum_{(s,s')\in S^-} \alpha_{k-1}(s')\exp\left(\frac{p_k L_c p'_k}{2}\right)\beta_k(s)}\right]$$

Sada se može prikazati log-likelihood odnos za ovaj estimator:

$$L_{map}(x_k) = L_a(x_k) + L_c x'_k + L_e(x_k)$$

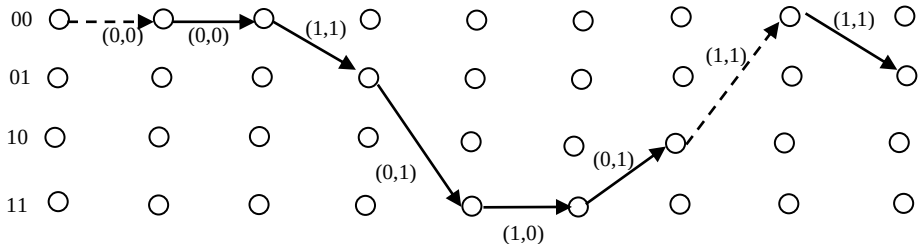
Prvi član $L_a(x_k)$ je apriori informacija koja predstavlja našu odluku prije nego započnemo sa MAP (turbo) algoritmom. Drugi član $L_c x'_k$ je dio koji nam daje šumna observacija i koji ne zavisi od ograničenja prouzrokovanih upotrijebljenim kodom. Treći dio je informacija koju dobijamo iz podataka o ograničenjima koje unosi parnost. Ova se informacija naziva ekstrinzičkom i ona se dobija iz koda. Kod turbo kodova ekstrinzička informacija o jednom kodu dobijena u procesu dekodiranja se koristi kao ulaz u dekoder za dekodiranje drugog MAP dekodera. Proces se zatim obavlja iterativno popravka sa jednog dekodera se prenosi na drugi i obrnuto.

(e) Izvršimo kodiranje sekvenca $\{X_1, X_2, X_3, X_4, X_5, X_6\} = \{1,1,0,0,1,0\}$. U prvom trelisu imamo.



Na kraj sekvence smo dodali dva bita informacije $\{X_7, X_8\} = \{1,0\}$. Pretpostavimo da je sada izvršena permutacija i da je ovako dobijeni signal doveden na ulaz drugog kodera. Permutovana

sekvenca je {1, 1, 0, 0, 1, 0, 1, 0}. Ova sekvenca je zatim propuštena kroz drugi koder rezultujući u trelis.



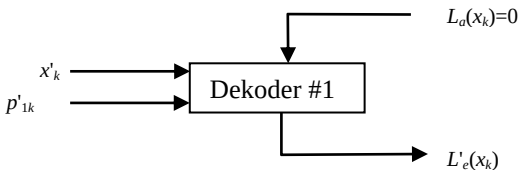
Kako ćemo posmatrati bipolarni sistem kojim se biti podataka i biti parnosti preslikavaju u:

X_k	P_{1k}	P_{2k}	→	x_k	p_{1k}	p_{2k}
1	1	0		+1	+1	-1
1	0	0		+1	-1	-1
0	0	1		-1	-1	+1
0	1	1		-1	+1	+1
1	0	0		+1	-1	-1
0	1	1		-1	+1	+1
1	1	1		+1	+1	+1
0	0	1		-1	-1	+1

(f) Nakon superponiranja sa Gausovim šumom dobijamo signal:

x'_k	p'_{1k}	p'_{2k}
2.966099	3.132927	-1.701887
-0.232363	-1.443420	-1.696641
-0.249255	-0.176735	1.823463
0.832447	0.911608	2.036052
-0.262811	-0.448993	-3.051227
-0.794776	1.277622	1.462560
0.430222	1.978633	2.105726
-0.742831	-0.534647	0.299060

U prvoj iteraciji pretpostavljamo da dekoder #1 nema ekstrinzičkih informacija pa je možemo postaviti na nulu. Stanje se sada može opisati dijagramom.



Vjerovatnoće grana se mogu računati iz zašumljenih opservaciji u skladu sa:

$$\gamma_k(s',s) = \exp\left[\frac{1}{2}(x_k L_a(x_k) + x_k L_c x'_k + p_k L_c p'_k)\right] \quad L_c = \frac{2}{\sigma^2} = 2$$

Koristivši trellis dijagram dat pod (b) računamo vjerovatnoće grana u prvom koraku na osnovu x_0, p_0 dobijamo vrijednosti $\gamma_1(s',s)$ kao:

$$\gamma_1(0,0) = \exp(-x'_1 - p'_1) = \exp(-2.966099 - 3.132927) = 0.002245$$

$$\gamma_1(0,1) = \exp(x_1' + p_1') = \exp(2.966099 + 3.132927) = 445.4235$$
$$\gamma_1(1,2) = \exp(x_1' - p_1') = \exp(2.966099 - 3.132927) = 0.846345$$
$$\gamma_1(1,3) = \exp(-x_1' + p_1') = \exp(-2.966099 + 3.132927) = 1.181551$$
$$\gamma_1(2,0) = \exp(x_1' + p_1') = \exp(2.966099 + 3.132927) = 445.4235$$
$$\gamma_1(2,1) = \exp(-x_1' - p_1') = \exp(-2.966099 - 3.132927) = 0.002245$$
$$\gamma_1(3,2) = \exp(-x_1' + p_1') = \exp(-2.966099 + 3.132927) = 1.181551$$
$$\gamma_1(3,3) = \exp(x_1' - p_1') = \exp(2.966099 - 3.132927) = 0.846345$$

Ponavljajući procedure za sva stanja u trelisu dobijamo.

s',s	$\gamma_1(s',s)$	$\gamma_2(s',s)$	$\gamma_3(s',s)$	$\gamma_4(s',s)$	$\gamma_5(s',s)$	$\gamma_6(s',s)$	$\gamma_7(s',s)$	$\gamma_8(s',s)$
0,0	0.002245	5.342977	1.531105	0.064309	2.037664	0.617025	0.089918	3.587580
0,1	445.4235	0.187162	0.653123	15.549912	0.490758	1.620680	11.121220	0.278739
1,2	0.846345	3.357031	1.075214	2.511397	0.830122	0.125884	0.212586	0.812058
1,3	1.181551	0.297882	0.930047	0.398184	1.204641	7.943850	4.703990	1.231440
2,0	445.4235	0.187162	0.653123	15.549912	0.490758	1.620680	11.121220	0.278739
2,1	0.002245	5.342977	1.531105	0.064309	2.037664	0.617025	0.089918	3.587580
3,2	1.181551	0.297882	0.930047	0.398184	1.204641	7.943850	4.703990	1.231440
3,3	0.846345	3.357031	1.075214	2.511397	0.830122	0.125884	0.212586	0.812058

Rekurzija unaprijed se može sračunati kao:

$$\alpha_k(s) = \sum_{s'} \alpha_{k-1}(s')\gamma_k(s',s)$$

U predmetnom primjeru rezultujuće vrijednosti su date u tabeli. U svakoj vrsti su date dobijene vrijednosti pa normalizovane tako da suma kolone se dobija da je 1. Normalizovana vrijednost ulazi u računanje narednog koraka:

S	$\alpha_0(s)$	$\alpha_1(s)$	$\alpha_2(s)$	$\alpha_3(s)$	$\alpha_4(s)$	$\alpha_5(s)$	$\alpha_6(s)$	$\alpha_7(s)$	$\alpha_8(s)$
0	1	0.002245 0.000005	0.000027 0.000007	0.599899 0.276497	0.279384 0.086266	0.272254 0.125209	0.657920 0.125318	5.329578 0.611503	2.241861 0.645632
1	0	445.4237 0.999995	0.000001 0	1.406312 0.648177	1.588757 0.490561	0.442903 0.203690	0.423994 0.080761	1.436693 0.164843	0.788824 0.227173
2	0	0	3.357014 0.918491	0.087632 0.040390	0.636660 0.196582	0.779049 0.358283	2.510613 0.478213	1.502255 0.172365	0.197021 0.056740
3	0	0	0.297881 0.081501	0.075800 0.034937	0.733850 0.226591	0.680187 0.312817	1.657463 0.315708	0.447013 0.051289	0.244644 0.070455

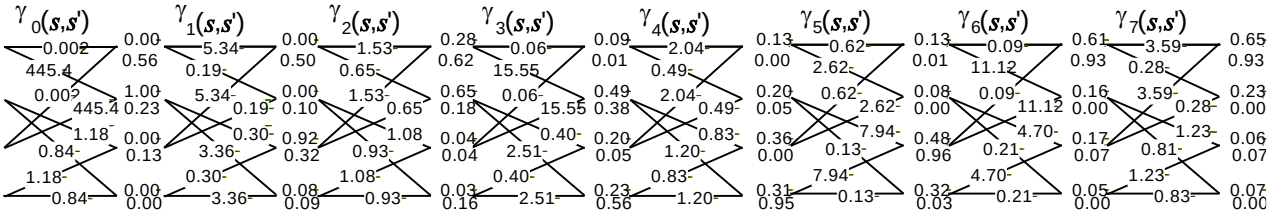
Rekurzija unazad se može računati kao:

$$\beta_{k-1}(s') = \sum_{s} \gamma_k(s',s)\beta_k(s)$$

U našem primjeru dobija se tabela prikazana na isti način sa vrijednostima i normalizacijama kao prethodna:

s	$\beta_8(s)$	$\beta_7(s)$	$\beta_6(s)$	$\beta_5(s)$	$\beta_4(s)$	$\beta_3(s)$	$\beta_2(s)$	$\beta_1(s)$	$\beta_0(s)$
0	1	3.587580	0.083436	0.007094	0.024528	2.202315	1.069151	2.676299	102.274177
	1	0.927906	0.007756	0.000885	0.011918	0.620645	0.497532	0.561278	0.289971
1	0	0	0.015326	0.371193	0.791577	0.645871	0.206641	1.094823	0.204413
	0	0	0.001425	0.046305	0.384623	0.182016	0.096161	0.229608	0.000580
2	0	0.278739	10.319444	0.013449	0.094790	0.135414	0.684043	0.606906	250.007162
	0	0.072094	0.959294	0.001678	0.046058	0.038162	0.318321	0.127281	0.708827
3	0	0	0.339131	7.624453	1.147165	0.564829	0.189074	0.390194	0.219647
	0	0	0.031525	0.951132	0.557401	0.159177	0.087986	0.081832	0.000622

Rezultujući trelis je prikazan na narednom grafiku.



Sada možemo sračunati meku odluku putem dekodera sa maksimalnom aposteriori vjerovatnoćom pa dobijamo da je:

$$L_{map}(x_k) = \log\left[\frac{\sum_{(s',s)\in S^+}\alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)}{\sum_{(s',s)\in S^-}\alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)}\right]$$

$$= \frac{\alpha_{k-1}(0)\gamma_k(0,1)\beta_k(1)+\alpha_{k-1}(1)\gamma_k(1,2)\beta_k(2)+\alpha_{k-1}(2)\gamma_k(2,0)\beta_k(0)+\alpha_{k-1}(3)\gamma_k(3,3)\beta_k(3)}{\alpha_{k-1}(0)\gamma_k(0,0)\beta_k(0)+\alpha_{k-1}(1)\gamma_k(1,3)\beta_k(3)+\alpha_{k-1}(2)\gamma_k(2,1)\beta_k(1)+\alpha_{k-1}(3)\gamma_k(3,2)\beta_k(2)}$$

Log-likelihood funkcija za $L_{map}(x_k)$ je jednaka:

11.3042 3.7075 0.3936 0.5056 -0.4363 -4.3760 3.7377 -3.8213

Primjenom "tvrdog" praga (pozitivno je 1 a negativno je nula) dobijamo:

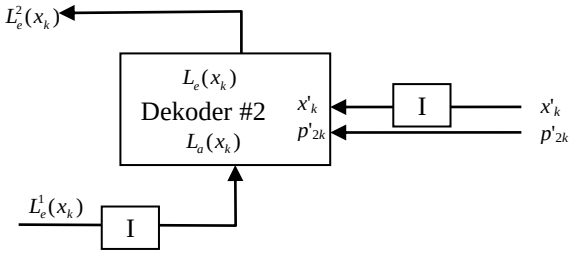
1 1 1 1 0 0 1 0

Uočite da imamo razliku u odnosu na polaznu poruku na 3 bita. Formirajmo sada ekstrinzičku informaciju nakon prvog koraka kao:

$$L_{map}(x_k) - [L_a(x_k) + L_C x_k'] = L_e(x_3)$$

$L_{map}(x_k)$	$-L_a(x_k)$	$-L_C x_k'$	$L_e(x_3)$
11.3042	0	-5.9322	5.3720
3.7075	0	0.4647	4.1722
0.3936	0	0.4985	0.8921
0.5056	0	-1.6649	-1.1593
-0.4363	0	0.5256	0.0893
-4.3760	0	1.5896	-2.7864
3.7377	0	-0.8604	2.8773
-3.8213	0	1.4857	-2.3356

U drugoj iteraciji se koristi ekstriznička informacija iz prvog dekodera kao apriori informacija u drugom dekoderu. Dobijeni bitovi podataka su permutovani da dobijemo šumne bite koji su u skladu sa redosljedom koji se koristi u ovom dekoderu.



U skladu sa permutacijom u interliveru imamo sljedeći raspored $L_a(x_k)=L_a([8\ 6\ 2\ 3\ 5\ 4\ 7\ 1])$ (ovdje je upotrijebljena MATLAB notacija). Na isti način se permutuje x_k':

$L_a(x_k)$	x'_k	p'_k
-2.3356	-0.7428	-1.7019
-2.7864	-0.7948	-1.6966
4.1722	-0.2324	1.8235
0.8921	-0.2493	2.0361
0.0893	-0.2628	-3.0512
-1.1593	0.8324	1.4626
2.8773	0.4302	2.1057
5.3720	2.9661	0.2991

Funkcija $\gamma_k()$ se u ovoj iteraciji formira kao:

$$\gamma_k(0,0) = \exp((-L_a(x_k)/2 - x'_k) - p'_k)$$
$$\gamma_k(0,1) = \exp(L_a(x_k)/2 + x'_k + p'_k)$$
$$\gamma_k(1,2) = \exp(L_a(x_k)/2 + x'_k - p'_k)$$
$$\gamma_k(1,3) = \exp(-L_a(x_k)/2 - x'_k + p'_k)$$
$$\gamma_k(2,0) = \exp(L_a(x_k)/2 + x'_k + p'_k)$$
$$\gamma_k(2,1) = \exp(-L_a(x_k)/2 - x'_k - p'_k)$$
$$\gamma_k(3,2) = \exp(-L_a(x_k)/2 - x'_k + p'_k)$$
$$\gamma_k(3,3) = \exp(L_a(x_k)/2 + x'_k - p'_k)$$

Formirajmo tabelu sa ovom funkcijom:

s',s	$\gamma_1(s',s)$	$\gamma_2(s',s)$	$\gamma_3(s',s)$	$\gamma_4(s',s)$	$\gamma_5(s',s)$	$\gamma_6(s',s)$	$\gamma_7(s',s)$	$\gamma_8(s',s)$
0,0	37.0592	48.6487	0.0253	0.1072	26.2955	0.1799	0.0188	0.0026
0,1	0.0270	0.0206	39.5357	9.3263	0.0380	5.5586	53.2278	384.2006
1,2	0.8116	0.6118	1.0307	0.1589	16.9973	0.2983	0.7891	211.2505
1,3	1.2321	1.6345	0.9702	6.2919	0.0588	3.3527	1.2673	0.0047
2,0	0.0270	0.0206	39.5357	9.3263	0.0380	5.5586	53.2278	384.2006
2,1	37.0592	48.6487	0.0253	0.1072	26.2955	0.1799	0.0188	0.0026
3,2	1.2321	1.6345	0.9702	6.2919	0.0588	3.3527	1.2673	0.0047
3,3	0.8116	0.6118	1.0307	0.1589	16.9973	0.2983	0.7891	211.2505

Sračunajmo sada (normalizovano) $\alpha_k(s)$ i $\beta_k(s)$ koje prikazujemo u tabelama:

s	$\alpha_0(s)$	$\alpha_1(s)$	$\alpha_2(s)$	$\alpha_3(s)$	$\alpha_4(s)$	$\alpha_5(s)$	$\alpha_6(s)$	$\alpha_7(s)$	$\alpha_8(s)$
0	1	0.9993	0.9995	0.0006	0.0000	0.0001	0.0064	0.9875	0.0030
1	0	0.0007	0.0004	0.9993	0.0009	0.0375	0.0004	0.0076	0.9918
2	0	0	0.0000	0.0000	0.0246	0.0042	0.8808	0.0030	0.0042
3	0	0	0.0000	0.0000	0.9744	0.9582	0.1124	0.0019	0.0010

s	$\beta_8(s)$	$\beta_7(s)$	$\beta_6(s)$	$\beta_5(s)$	$\beta_4(s)$	$\beta_3(s)$	$\beta_2(s)$	$\beta_1(s)$	$\beta_0(s)$
0	0.2500	0.3226	0.3417	0.0359	0.0528	0.5906	0.4226	0.9319	0.9794
1	0.2500	0.1774	0.0173	0.0572	0.4116	0.3643	0.0042	0.0161	0.0017
2	0.2500	0.3226	0.6211	0.4323	0.0840	0.0684	0.5689	0.0098	0.0176
3	0.2500	0.1774	0.0198	0.4746	0.4516	0.0766	0.5689	0.0098	0.176

Sračunajmo sada $L_{\text{map}}(x_k)$ na osnovu prethodnih relacija pa dobijamo:

-11.2839 -11.0077 7.0541 -5.1885 4.8451 -4.5711 5.7372 11.8904

sa odgovarajućom "hard" odlukom:

0 0 1 0 1 0 1 1

čime se dobila sekvenca u kojoj nema grešaka.

Sada sračunajmo meku odluku na izlazu iz MAP dekodera koja iznosi:

$L_{map}(x_k)$	$-L_a(x_k)$	$-L_C x'_k$ =	$L_e(x_3)$
-11.2839	2.3356	1.4857	-7.4626
-11.0077	2.7864	1.5896	-6.6317
7.0541	-4.1722	0.4647	3.3466
-5.1885	-0.8921	0.4985	-5.5821
4.8451	-0.0893	0.5256	5.2815
-4.5711	1.1593	-1.6649	-5.0767
5.7372	-2.8773	-0.8604	1.9995
11.8904	-5.3720	-5.9322	0.5862

Ponavljajući kompletan postupak nekoliko iteracija dobijamo sljedeću tabelu sa ekstrinzičkim informacijama u pojedinim iteracijama.

#0	#1	#2	#3	#4	#5	#6	#7
$L_e^1(x_k)$	$L_e^2(x_k)$	$L_e^1(x_k)$	$L_e^2(x_k)$	$L_e^1(x_k)$	$L_e^2(x_k)$	$L_e^1(x_k)$	$L_e^2(x_k)$
5.3720	-7.4626	14.3560	-20.0202	17.3927	-20.0697	17.3926	-20.0697
4.1722	-6.6317	10.6861	-18.1967	18.6237	-18.2476	18.6717	-18.2476
0.8921	3.3466	-9.2185	18.1499	-19.9789	18.6310	-20.0279	18.6310
-1.1593	-5.5821	-11.2938	-13.1898	-14.8526	-13.6978	-14.8530	-13.6978
0.0893	5.2815	8.7022	13.1898	21.3719	13.2377	21.4453	13.2377
-2.7864	-5.0767	-10.8954	-7.7337	-19.7240	-7.7347	-19.7598	-7.7347
2.8773	1.9995	12.6339	4.8091	15.2023	4.8096	15.3032	4.8096
-2.3356	0.5862	-12.0293	0.5981	-21.0834	0.5981	-21.1307	0.5981

(g) Prikažimo sada MATLAB kod koji realizuje prve dvije iteracije ovog algoritma. Dodajte odgovarajuću `for` ili `while` petlju i izvršite neophodne popravke kako bi program mogao da produkuje više od jedne kompletne iteracije.

```
Clear
% Unipolarna sekvenca (bitovi podatka i dva seta bitova parnosti)
Z=[
    1   1   0
    1   0   0
    0   0   1
    0   1   1
    1   0   0
    0   1   1
    1   1   1
    0   0   1
    ];

% Bipolarna sekvenca
Z=Z(:,1:3)*2-1;

% Jedna realizacija šuma, testirati sa slučajnim procesom!
N=[1.966099 2.132927    -0.701887
    -1.232363   -0.443420   -0.696641
    0.750745    0.823265    0.823463
    1.832447    -0.088392   1.036052
    -1.262811   0.551007    -2.051227
    0.205224    0.277622    0.462560
    -0.569778   0.978633    1.105726
    0.257169    0.465353    -0.700940];
% Signal+Šum
Y=Z+N;

% \gamma_k(s,s'), promjenljiva Tabela služi za jednostavnije predstavljanje
Gamma=zeros(4,4,size(N,1));
Tabela=zeros(8,size(N,1));
for k=1:size(N,1)
    Gamma(1,1,k)=exp(-Y(k,1)-Y(k,2));
    Gamma(1,2,k)=exp(Y(k,1)+Y(k,2));
    Gamma(2,3,k)=exp(Y(k,1)-Y(k,2));
    Gamma(2,4,k)=exp(-Y(k,1)+Y(k,2));
    Gamma(3,1,k)=Gamma(1,2,k);
    Gamma(3,2,k)=Gamma(1,1,k);
    Gamma(4,3,k)=Gamma(2,4,k);
    Gamma(4,4,k)=Gamma(2,3,k);
    Tabela(:,k)=[Gamma(1,1,k) Gamma(1,2,k) Gamma(2,3,k) Gamma(2,4,k) …
Gamma(3,1,k) Gamma(3,2,k) Gamma(4,3,k) Gamma(4,4,k)]';
end

% Proračun \alpha_k(s)
Alfa1=zeros(4,9);
Alfa2=zeros(4,9);
Alfa1(:,1)=[1 0 0 0]';
Alfa2(:,1)=Alfa1(:,1);
for k=2:9
    for r=1:4
        for p=1:4

            Alfa1(r,k)=Alfa1(r,k)+Gamma(p,r,k-1)*Alfa2(p,k-1);
        end
    end
    Alfa2(:,k)=Alfa1(:,k)/sum(Alfa1(:,k));
end
```

```
% Proračun \beta_k(s')
Beta1=zeros(4,9);
Beta2=zeros(4,9);
Beta1(:,9)=[1 0 0 0]'
Beta2(:,9)=Beta1(:,9);
for k=8:-1:1
    for r=1:4
        for p=1:4
            Beta1(r,k)=Beta1(r,k)+Gamma(r,p,k)*Beta2(p,k+1);
        end
        Beta2(:,k)=Beta1(:,k)/sum(Beta1(:,k));
    end
end

% Proračun L_{map}(x_k)
for k=1:8
    gornji=Alfa2(1,k)*Beta2(2,k+1)*Gamma(1,2,k)+…
      Alfa2(2,k)*Beta2(3,k+1)*Gamma(2,3,k)+…
      Alfa2(3,k)*Beta2(1,k+1)*Gamma(3,1,k)+Alfa2(4,k)*Beta2(4,k+1)*Gamma(4,4,k);
    donji=Alfa2(1,k)*Beta2(1,k+1)*Gamma(1,1,k)+…
      Alfa2(2,k)*Beta2(4,k+1)*Gamma(2,4,k)+…
      Alfa2(3,k)*Beta2(2,k+1)*Gamma(3,2,k)+Alfa2(4,k)*Beta2(3,k+1)*Gamma(4,3,k);
    Lmap(k)=log(gornji/donji)
end

%%%Proračun ekstrinzičkih vrijednosti
La=zeros(8,1);
La=Lmap'-La-2*Y(:,1);
La=La([8 6 2 3 5 4 7 1]);     %Permutacija
YS=[La,Y([8 6 2 3 5 4 7 1],1),Y(:,3)];

%Nova \gamma
Gamma=zeros(4,4,size(N,1));
Tabela=zeros(8,size(N,1));
for k=1:size(N,1)
    Gamma(1,1,k)=exp(-YS(k,1)/2-YS(k,2)-YS(k,3));
    Gamma(1,2,k)=exp(YS(k,1)/2+YS(k,2)+YS(k,3));
    Gamma(2,3,k)=exp(YS(k,1)/2+YS(k,2)-YS(k,3));
    Gamma(2,4,k)=exp(-YS(k,1)/2-YS(k,2)+YS(k,3));
    Gamma(3,1,k)=Gamma(1,2,k);
    Gamma(3,2,k)=Gamma(1,1,k);
    Gamma(4,3,k)=Gamma(2,4,k);
    Gamma(4,4,k)=Gamma(2,3,k);
    Tabela(:,k)=[Gamma(1,1,k) Gamma(1,2,k) Gamma(2,3,k) Gamma(2,4,k) …
      Gamma(3,1,k) Gamma(3,2,k) Gamma(4,3,k) Gamma(4,4,k)]';
end

%Nova \alpha
Alfa1=zeros(4,9);
Alfa2=zeros(4,9);
Alfa1(:,1)=[1 0 0 0]';
Alfa2(:,1)=Alfa1(:,1);
for k=2:9
    for r=1:4
        for p=1:4

            Alfa1(r,k)=Alfa1(r,k)+Gamma(p,r,k-1)*Alfa2(p,k-1);
        end
    end
    Alfa2(:,k)=Alfa1(:,k)/sum(Alfa1(:,k));
end

%Nova \beta
```

```
Beta1=zeros(4,9);
Beta2=zeros(4,9);
Beta1(:,9)=[0.25 0.25 0.25 0.25]'
Beta2(:,9)=Beta1(:,9);
for k=8:-1:1
    for r=1:4
        for p=1:4
            Beta1(r,k)=Beta1(r,k)+Gamma(r,p,k)*Beta2(p,k+1);
        end
        Beta2(:,k)=Beta1(:,k)/sum(Beta1(:,k));
    end
end

%proračun Lmap
for k=1:8
    gornji=Alfa2(1,k)*Beta2(2,k+1)*Gamma(1,2,k)+…
Alfa2(2,k)*Beta2(3,k+1)*Gamma(2,3,k)+…
      Alfa2(3,k)*Beta2(1,k+1)*Gamma(3,1,k)+Alfa2(4,k)*Beta2(4,k+1)*Gamma(4,4,k);
    donji=Alfa2(1,k)*Beta2(1,k+1)*Gamma(1,1,k)+…
      Alfa2(2,k)*Beta2(4,k+1)*Gamma(2,4,k)+…
      Alfa2(3,k)*Beta2(2,k+1)*Gamma(3,2,k)+Alfa2(4,k)*Beta2(3,k+1)*Gamma(4,3,k);
    Lmap(k)=log(gornji/donji)
end

%Radi grafičkog predstavljanja korišćenog u knjizi
[Lmap',-La,-2*YS(:,2),Lmap'-La-2*YS(:,2)]
%Proračun ekstriničkih brjednosti
La=Lmap'-La-2*YS(:,2);
```

U trenutku pisanja ovog materijala nije bilo MATLAB-ove funkcije za dekodiranje turbo kodova. Međutim, nije teško naći programe za realizaciju turbo kodova u MATLAB-u (na primjer preko razmjene fajlova – File Exchange na web sajtu MATLAB Central (http://www.mathworks.com/matlabcentral).

Poglavlje XII
CRC KODOVI, GRANICE ZA KODIRANJE

12.1. CRC Kodovi – Uvod i osnovni podaci

CRC (Cyclic Redundancy Check) kodovi su klasa kodova koji su prevashodno namjenjeni za detekciju a ne i korekciju većeg broja grešaka. CRC kodovi se kreiraju na osnovu generatorskog polinoma čiji koeficijenti uzimaju vrijednosti iz nekog konačnog polja. Najčešće je to polje binarno. Posebno je značajna sposobnosti CRC kodova da detektuju pojavu više grešaka u relativno dugačkim kodnim riječima. Pretpostavimo da je generatorski polinom ovog koda $g(x)$. Neka je $i(x)$ informaciona poruka dok je $d(x)$ dobijeni polinom koji se tokom prenosa u kanalu može deformisati do polinoma $\tilde{d}(x)$. Greška u prenosu se može izraziti kao:

$$e(x)=d(x)-\tilde{d}(x)$$

Broj nenultih koeficijenata u $e(x)$ je broj pogreški. Neka je $r(x)$ ostatak pri dijeljenu $d(x)$ sa $g(x)$. Ova veličina se naziva CRC od $d(x)$. Sada možemo pisati:

$$d(x)=q(x)g(x)+r(x)$$

Neka je $\tilde{r}(x)$ ostatak dobijen dijeljenjem $\tilde{d}(x)$ sa $g(x)$. Sada se može zapisati:

$$\begin{aligned} e(x) &= d(x)-\tilde{d}(x) = (q(x)g(x)+r(x))-(\tilde{q}(x)g(x)+\tilde{r}(x)) = \\ &= (q(x)-\tilde{q}(x))g(x)+r(x)-\tilde{r}(x) \end{aligned}$$

Dakle, može da se zaključi da je ostatak pri dijeljenju $e(x)$ sa $g(x)$ jednak $r(x)-\tilde{r}(x)$. Da bi CRC pogriješio u detekciji pogreške neophodno je da $r(x)-\tilde{r}(x)$ bude jednako 0 što je ekvivalentno sa činjenicom da $g(x)$ dijeli $e(x)$ (sa ostatkom 0).

Ako se greška dogodi samo na jednom bitu tada se polinom greške može zapisati kao $e(x)=x^i$. U ovom slučaju se greška ne može detektovati samo u slučaju kada $g(x)$ dijeli $e(x)=x^i$ bez ostatka. Pošto je x^i proizvod i kopija od x, $g(x)$ može da dijeli x^i samo ako je $g(x)$ jednako x^j za neko $j\leq i$. Dakle, već polinom $g(x)=x+1$ je sposoban za detekciju jedne greške. Ovo je suštinski kod sa provjerom parnosti kakav se koristi kod ASCII sistema. Stoga možemo da smatramo da je detekcija jedne pogreške jednostavna. Ako postoje dvije pogreške na m-toj i na n-toj poziciji (sa $m<n$) tada je polinom greške:

$$e(x)=x^m+x^n$$

Do pogreške u detekciji sa CRC-om dolazi samo ako $g(x)$ dijeli $e(x)=x^m+x^n$. Polinom $e(x)$ možemo faktorisati kao:

$$e(x)=x^m+x^n=x^m(1+x^{n-m})$$

Relativno lako se obezbijediti da $g(x)$ ne dijeli x^m ali nije jednostavna analiza kako dizajnirati $g(x)$ da ne dijeli bilo koju varijantu $1+x^{n-m}$. Uočimo da važi $x^N-1=(x-1)(x^{N-1}+x^{N-2}+...+x+1)$ za bilo koje N. Slično važi ako zamjenimo recimo x sa x^8 pa se može zapisati:

$$x^{16}-1=(x^8-1)(x^8+1)$$

$$x^{24}-1=(x^8-1)(x^{16}+x^8+1)$$
$$x^{32}-1=(x^8-1)(x^{24}+x^{16}+x^8+1)$$
$$x^{40}-1=(x^8-1)(x^{32}+x^{24}+x^{16}+x^8+1)$$
$$\ldots$$
$$x^{8N}-1=(x^8-1)(x^{8(N-1)}+\ldots+x^8+1)$$

Dakle, uočavamo da predmetni kod zasnovan na generatorskom polinomu x^8-1 neće detektovati dvije greške ako se one nalaze na rastojanju koje je umnožak od 8. Predmetna osobina može biti postignuta i sa polinomima manjih dimenzija. Npr. CRC sa generatorskim polinomom x^3+x+1 iako je samo stepena 3 a ne 8 daje veoma sličnu osobinu odnosno ne detektuje dvije pogreške samo ako su ove pogreške razmaknute za broj cifara koji je multipl broja 7. Dakle, x^3+x+1 dijeli x^N-1 sa ostatkom 0 samo ako je N množioc broja 7. Ova osobina je u direktnoj vezi sa upotrebom ovog prostog polinoma kod kreiranja Hammingovog koda (7,4). Slično važi da x^4+x+1 prosti polinom ne može da detektuje dvije greške samo kada je rastojanje između pozicija na kojima su se pogreške dogodile jednak multiplu broja 15. Slično koristivši iskustva iz Hammingovih kodova prosti polinom x^5+x^2+1 ne može da detektuje dvije greške samo kada su na rastojanju koje je multipl broja 31. Treba znati da ne može svaki polinom 5-stepena da ponovi ovako kvalitetan rezultat. Tako polinom x^5+x+1 može da otkrije greške koje nijesu razmaknute za broj cifara koji je multipl 21 dok polinom x^5+x^4+x+1 ne može da detektuje pogreške koje su razmaknute za multipl 8 bitova. Jedan izuzetno moćan CRC kod se može dobiti sa generatorskim polinomom:

$$g(x)=x^{16}+x^{15}+x^2+1=(x+1)(x^{15}+x+1)$$

je u stanju da detektuje dvije pogreške osim ako se one nalaze na rastojanju koje je umnožak broja 2^{15}-1 (32767). Dakle, dobri generatorski polinomi za CRC kodove moraju biti prosti (nesvodljivi polinomi). Druga osobina je da dati polinom mora biti djelilac x^N-1 gdje je $N=2^d$-1 a d je stepen predmetnog polinoma. Napomenimo dalje da je $g(x)$ takav generatorski polinom da je u stanju da detektuje bilo koju kombinaciju od 3 greške u setu podataka koji je kraći od 2^{15}-1 (32767). Razlog je u činjenici da se kao faktor kod ovog polinoma pojavljuje polinom $x+1$ koji je u stanju da detektuje uvijek neparan broj grešaka (prva osobina vezana za dvije greške jedino je vezana za prosti polinom $x^{15}+x+1$ a nema veze sa $x+1$).

Pored mogućnosti detekcije nekoliko pogreški kod CRC kodova važna je i osobina da se pomoću njih može detektovati i nekoliko uzastopnih pogreški (tzv. burst errors). CRC kod reda n uvijek je u stanju da detektuje uzastopne greške koje imaju dužinu koja je kraća od n. Da bi ovo dokazali zapisaćemo polinom pogreške kao:

$$e(x)=x^n p(x)$$

gdje je $p(x)$ polinom reda koji je manji od n. CRC ne bi bio u stanju da detektuje ovakav tip pogreške u slučaju da $g(x)$ dijeli $e(x)$. Jasno je da $g(x)$ nije djelilac od x^n a ujedno pošto je $p(\boldsymbol{x})$ manjeg reda od n $g(x)$ ne može podijeliti ni $p(x)$.

Na ovaj način smo u kratkim crtama opisali osnovne osobine CRC kodova.

12.2. Reed-Mullerovi kodovi

Reed-Mullerovi (čita se Rid-Maler) kodovi predstavljaju proširenje standardnih Hammingovih kodova. Naime, u teoriji kodova postoje strategije pomoću kojih se na osnovu dobro dizajniranih kodova (npr. Hammingovih, BCH i drugih sličnih) dizajniraju kodovi sa izmjenjenim osobinama. Jedna od strategija za dizajniranje ovih kodova je proširenje kodova. Reed-Mullerovi kodovi su uspješna strategija koja je primjenjena kod proširenja. Ova grupa kodova je nastala 1954-te od

strane Reeda i Mullera koji su do istih zaključaka došli nezavisno. Mi ćemo ovdje samo demonstrirati binarne Reed-Mullerove kodove prvog reda koji se relativno jednostavno mogu povezati sa Hammingovim kodovima. Ovo su kodovi dužine 2^m gdje je m cijeli broj. Generatorska matrica Reed-Mullerovog koda se može definisati rekurzivnom relacijom:

$$EL_m = \left[\begin{array}{c|c} 00\ldots0 & 11\ldots1 \\ \hline EL_{m-1} & EL_{m-1} \end{array}\right]$$

gdje je prva generatorska matrica definisana kao:

$$EL_1 = \left[\begin{array}{c|c} 0 & 1 \\ \hline 1 & 1 \end{array}\right]$$

Prvih nekoliko generatorskih matrica kod Reed-Mullerovih kodova su:

$$EL_2 = \left[\begin{array}{cc|cc} 0 & 0 & 1 & 1 \\ \hline 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{array}\right]$$

$$EL_3 = \left[\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ \hline 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right]$$

$$EL_4 = \left[\begin{array}{cccccccc|cccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right]$$

Uočimo da recimo EL_3 matrica predstavlja zapravo kontrolnu matricu proširenog Hammingovog koda (8,4). Na osnovu generatorske matrice Reed-Mullerovog koda može se zaključiti da je u pitanju kod koji ima m+1 informacioni simbol dok je ukupna dužina kodne riječi 2^m. Za relativno veliko m riječ je o kodu koji ima ekstremno veliku dužinu kodne riječi ali na račun toga dobijamo i ogromno minimalno Hammingovo rastojanje koje iznosi 2^{m-1} pa možemo zaključiti da ovaj kod ima solidnu sposobnost za ispravljanje velikog broja grešaka za veliko m. Sledeća interesantna činjenica vezana za Reed-Mullerove kodove je ta da moguće kodne riječi su kodna riječi sa svim nulama, kodne riječi sa svim jedinicama dok ostale kodne riječ imaju težinu 2^{m-1}. Pokušajte dokazati ovu činjenicu. Reed-Mullerov kod zbog nekih interesantnih osobina često se prikazuje u bipolarnoj formi. Naime, nule se zamijene sa minus jedinicama. Skalarni proizvod dvije ovakve kodne riječi u tom slučaju ima vrijednost koja je jednaka broju pozicija na kojima se pojavljuju isti bitovi umanjen za broj pozicija na kojima se pojavljuju različiti simboli. Ovaj produkt se može zapisati kao n-$2d_H(x,y)$. U ovom slučaju je n dužina koda a $d_H(x,y)$ Hammingovo rastojanje dvije kodne riječi (broj pozicija na kojima se kodne riječi razlikuju. Za kodne riječi koje pripadaju ovom kodu (bipolarnom kodu koji je dobijen zamjenom 0 sa -1) važi da je skalarni proizvod dvije riječi jednak:

$$\mathbf{c}_i \cdot \mathbf{c}_j = \begin{cases} \pm 2^m & \mathbf{c}_i = \pm \mathbf{c}_j \\ 0 & \mathbf{c}_i \neq \pm \mathbf{c}_j \end{cases}$$

Ova osobina se može koristiti kao osnova dekodirajućeg algoritma. Pretpostavimo da je primljena kodna riječ **r** odluka o tome koja je kodna riječ primljena može se donijeti na osnovu (skalarnog) množenja **r** sa svim kodnim riječima $\mathbf{r}\cdot\mathbf{c}_i$. Odluka se može donijeti tako što se poredi dobijeni rezultat i odabere ona kodna riječ koja daje najveći rezultat. Posebna pogodnost je ta da se algoritam može primjeniti i za slučaj kada je kodna riječ **r** nebinarna odnosno ako umjesto standardnih vrijednosti za **r** (binarne unipolarne ili bipolarne sekvence) možemo koristiti primljene vrijednosti (naravno normalizovane) koje mogu da uzmu vrijednosti koji nijesu cijeli brojevi.

Dekodiranje se u ovom domenu može obaviti na osnovu Hadamardove transformacije. Hadamardova transformacija se definiše rekurzivno (na sličan način kao matrica Reed-Mullerovog koda):

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad \mathbf{H}_4 = \begin{bmatrix} \mathbf{H}_2 & \mathbf{H}_2 \\ \mathbf{H}_2 & -\mathbf{H}_2 \end{bmatrix} \qquad \mathbf{H}_{2^n} = \begin{bmatrix} \mathbf{H}_{2^{n-1}} & \mathbf{H}_{2^{n-1}} \\ \mathbf{H}_{2^{n-1}} & -\mathbf{H}_{2^{n-1}} \end{bmatrix}$$

Za Hadamardovu matricu važi da je:

$$\mathbf{H}_m \mathbf{H}_m^T = m\mathbf{I}_m$$

gdje je $\mathbf{I}_m$ kvadratna matrica dimenzija mxm. Npr. Hadamardova matrica dimenzija 8x8 je:

$$\mathbf{H}_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

Pretpostavimo da je primljena sekvenca r=[1, 1, -1, 1, -1, -1, 1, 1]. Hadamardova transformacija ove sekvence je:

$$\mathbf{H}_8 r^T = \begin{bmatrix} 2 & -2 & 2 & -2 & 2 & -2 & 6 & 2 \end{bmatrix}^T$$

Kako je najveća apsolutna vrijednost dobijena na poziciji 7 možemo zaključiti da ispravna kodna riječ odgovara sedmoj koloni (ili vrsti jer je Hadamardova matrica simetrična):

$$\mathbf{c}_7^8 = [1 \quad 1 \quad -1 \quad -1 \quad -1 \quad -1 \quad 1 \quad 1]$$

U slučaju da najveća apsolutna vrijednost u Hadamardovoj transformaciji primljenog vektora negativna neophodno je primljenu kodnu riječ zamjeniti sa njenom negacijom (promijeniti predznak svim elementima). Posebno atraktivna osobina Reed Mullerovih kodova je činjenica da se oni mogu primjeniti na sekvencama na koje nije primjenjen prag odnosno na sekvencama koje ne pokazuju isključivo vrijednosti ±1. Ovo je u skladu sa učenim kod turbo kodova kada primjenjujemo meko odlučivanje. Posmatrajmo sljedeći primjer. Neka je primljena sekvenca:

$$r = \begin{bmatrix} -0.7 & 1 & 0 & -0.8 & -0.9 & 1 & 0.9 & -1 \end{bmatrix}$$

Hadamardova transformacija je u ovom slučaju:

$$\mathbf{H}_8 r^T = \begin{bmatrix} -0.5 & -0.9 & 1.3 & -6.3 & -0.5 & -0.9 & 0.9 & 1.3 \end{bmatrix}^T$$

Najveću apsolutnu vrijednost sada ima četvrti član niza ali kako je negativan to zaključujemo da negacija četvrte kolone (vrste) Hadamardove matrice predstavlja odgovarajuću kodnu riječ:

$$-\mathbf{c}_4^* = \begin{bmatrix} -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

Predmetni kod zbog datih osobina naziva se i Hadamardovim. Da bi ovaj kod mogao da radi uspješno kodna riječ mora imati dužinu 2 ili da bude umnožak četvorke. Dekodiranje koje se obavlja za slučaj nebinarnih vrijednosti naziva se meko dekodiranje (slično kao kod turbo dekodiranja). Dobra strana primjene Hadamardove matrice u dekodiranju je izuzetna efikasnost množenja kontrolne (Hadamardove) matrice sa dobijenim vektorom. Naime, u opštem slučaju za množenje matrice dimenzija nxn čiji su elementi ± 1 sa vektorom dužine n zahtjeva nx(n-1) sabiranja (zanemarimo za trenutak množenje sa 1 i promjenu znaka). Međutim, postoji mogućnost da se postigne značajna ušteda u broju potrebnih operacija. Naime, operaciju množenja vektora r dužine n sa Hadamardovom matricom možemo zapisati preko dva vektora r_1 i r_2 od kojih vektor r_1 predstavlja prvih n/2 elemenata vektora r dok je r_2 vektor preostalih n/2 odbiraka vektora r.

$$\mathbf{H}_n \mathbf{r}^T = \begin{bmatrix} \mathbf{H}_{n/2} & \mathbf{H}_{n/2} \\ \mathbf{H}_{n/2} & -\mathbf{H}_{n/2} \end{bmatrix} \begin{bmatrix} r_1^T \\ r_2^T \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{n/2} r_1^T + \mathbf{H}_{n/2} r_2^T \\ \mathbf{H}_{n/2} r_1^T - \mathbf{H}_{n/2} r_2^T \end{bmatrix}$$

Izrazi $\mathbf{H}_{n/2} r_1^T$ i $\mathbf{H}_{n/2} r_2^T$ predstavljaju Hadamardove transformacije odgovarajućih vektora. Za proračun svake od ovih Hadamardovih transformacija bilo je potrebno po n/2(n/2-1) sabiranja tako da je ukupno za proračun Hadamardove transformacije $\mathbf{H}_n r^T$ preko razdvajanja na dvije podmatrice potrebno:

n(n/2-1)+n sabiranja

Ovo je zapravo 2 puta koliko je potrebno za Hadamardovu transformaciju po n/2 članova plus n dodatnih sabiranja za svaki elemenat u matrici dimenzija nxn odnosno n^2/2. Dakle, ako bi ovo primjenili na Hadamardovu transformaciju vektora dužine n=8 bilo bi nam potrebno 32 sabiranja dok je standardnim algoritmom potrebno 56 sabiranja. Ako nastavimo sa daljim dekomponovanjem za Hadamardovu matricu dimenzija n/2 je potrebno n^2/8 operacija dok je za Hadamardovu transformaciju dimenzija n/4 potrebno n^2/32 operacije. Pretpostavljajući da je za n elemenata vektora potrebno $f(n)$ operacija a za n/2 potrebno $f(n/2)$ operacija. Veza između broja operacija za n elemenata i n/2 elementa se može opisati kao:

$$f(n)=2f(n/2)+n$$

Funkcija koja zadovoljava ovu relaciju je $f(n)=n\log_2 n$. Dakle, za n=8 ako je Hadamardova dekompozicija vršena "do kraja" odnosno do matrica dimenzija 2x2 dobija se da je potrebno 24 operacije što je značajna ušteda. Npr. za n=1024 operacije umjesto preko milion operacija potrebno je samo nešto više od 10 hiljada.

12.3. Granice za kodove

Granice za kodove predstavljaju neke od činjenica vezanih za kodove koje se mogu pokazati bez izvođenja partikularnih kodova. Granice zapravo ukazuju na to što se može postići kod kodova a što je zapravo nemoguće. Postoji više tipova granica i mi ćemo u okviru ovoga materijala pokazati tri. Prva granica je Hammingova ili granica za pakovanje sfera. Grubo govoreći ova granica daje limite koliko dobar može biti posmatrani kod. Premda Hammingova granica nije naročito precizna ona se može lako shvatiti na osnovu fizičkih pravila pa je veoma popularna. Mi smo je i do sada koristili na primjeru binarnih kodova a sada ćemo je generalizovati za slučaj opštih q-arnih kodova. Posmatrajmo alfabet $\mathbf{F}_q$ preko konačnog polja od q elemenata. Neka je $\mathbf{F}_q^n$ skup svih vektora dužine n koji se mogu kreirati od predmetnog alfabeta. Hammingova težina $wt(v)$ za opšti q-arni kod se definiše kao broj nenultih mjesta u kodnoj riječi: $v=(v_1, ..., v_n)$ dok je Hammingova distanca $d(v,w)$ broj mjesta na kojima se kodne riječi v i w razlikuju. Veza Hammingove distance i težine je $wt(v-w)=d(v,w)$. Lopta radijusa r (mjereno Hammingovom distancom) u $\mathbf{F}_q^n$ centrirana na vektoru $\mathbf{v}$ je skup svih vektora w takvih da je $d(v,w)\leq r$. Zapreminom se naziva broj takvih vektora u lopti. Lopta radijusa r u $\mathbf{F}_q^n$ sadrži:

$$1+(q-1)\binom{n}{1}+(q-1)^2\binom{n}{2}+\cdots+(q-1)^r\binom{n}{r}$$

Prvi broj (1) je sama kodna riječ v, sledeći elemenat je broj kodnih riječi koje se razlikuju na jednoj poziciji. Pozicija je n a za svaki simbol postoji (q-1) simbola koji se razlikuju od njega. Slično se mogu analizirati i ostali članovi u predmetnom redu.

Lema. Neka su x i y dva vektora u $\mathbf{F}_q^n$ sa $d(x,y)>2e$ gdje je e cijeli broj. Ako za vektor z važi $d(x,z)\leq e$ tada važi $d(y,z)>e$.

Dokaz. Dokaz je posljedica nejednakost trougla: $d(x,y)\leq d(x,z)+d(z,y)$ za bilo koju trojku vektora x, y, z. Ako je $d(x,y)>2e$ i $d(x,z)\leq e$ tada slijedi da je: $2e<e+d(z,y)$ odnosno slijedi $e<d(z,y)$.

Posljedica. Ako se vektori x i y nalaze na rastojanju koje je veće od $2e$, $d(x,y)>2e$, tada su sfere koje su radijusa e centrirane u x i y razdvojene.

Dokaz prethodne posljedica izvršite sami.

Teorema. (Ova teorema se naziva Hammingovom ili granicom u pakovanju sfera). Posmatrajmo kod koji je podskup $\mathbf{F}_q^n$ ima minimalnu distancu $2e+1$ i ima l kodnih riječi:

$$l\left(1+(q-1)\binom{n}{1}+(q-1)^2\binom{n}{1}+\cdots+(q-e)^2\binom{n}{e}\right)\leq q^n$$

Dokaz. Pošto minimalna distanca je $2e+1$ tada po nejednakost trougla i na osnovu prethodne posljedice bilo koje dvije sfere radijusa e locirane u kodnim riječima su razdvojene (nemaju presjek). Dakle, suma zapremina l razdvojenih sfera će biti manja ili jednaka ukupnoj zapremini čitavog skupa q^n.

Kod koji predmetnu granicu dostiže sa jednakošću naziva se **perfektnim**. Postoji svega nekoliko perfektnih linearnih kodova: Hammingovi koji ispravljaju jednu grešku, dva Golayeva koda kao i nekoliko nelinearnih kodova. Predmetna teorema stoga daje dosta široke granice ali je fizikalnost njenog tumačenja takva da se ona obično jedino i uči na osnovnim kursevima teorije informacija i kodova.

Druga granica koju ćemo ovdje uvesti je Gilbert-Varshamova koje se izvodi u suprotnom pravcu u odnosu na Hammingovu. Ona pokušava da dokaže postojanje linearnog koda sa datim parametrima (uočite da je ograničena na linearne kodove) a ujedno ne dajući previše usku granicu za formiranje kodova. Posmatramo linearni blok kod na alfabetu $\mathbf{F}_q$ dužine n, dimenzije (broja informacionih simbola) k i sa distancom d. Generatorska matrica ovog koda je kxn a mi ćemo za sada označiti ovaj kod kao [n, k, d].

Teorema: Linearni kod [n, k, d] definisan preko alfabeta $\mathbf{F}_q$ postoji ako je zadovoljeno:

$$q^{n-k}-1>(q-1)\binom{n-1}{1}+\cdots+(q-1)^{d-3}\binom{n-1}{d-3}+(q-1)^{d-2}\binom{n-1}{d-2}$$

Teoremu ćemo dokazati za pojednostavljeni slučaj binarnog alfabeta (q=2) kada treba pokazati:

$$2^{n-k}-1>\binom{n-1}{1}+\binom{n-1}{2}+\cdots+\binom{n-1}{d-3}+\binom{n-1}{d-2}$$

Prije nego dokažemo predmetnu teoremu vrijedi istaći da ova teorema ne daje potvrdu da dobar kod postoji niti daje bilo kakvu efikasnu proceduru kako ga odrediti. Ujedno ovo nije garancija da kod ne može bolje već jedino da mi očekujemo predmetne performanse.

Dokaz. Pretpostavimo da smo u konstrukciji kontrolne matrice koda sa uspjehom izabrali l kolona tako da je $l\geq d$-2 tako da su l-1 linerno zavisne. Sada želimo da izaberemo (l+1) kolonu. Izbor se mora napraviti između vektor kolona dužine n-k. Ukupno postoji 2^{n-k} takvih elemenata. Moramo isključiti kolonu sa svim nulama, sve kolone koje su se prethodno pojavljivale, sumu svih kombinacija prethodne dvije kolone, i tako sve do sume bilo koje d-2 prethodne kolone. U najgorem slučaju svi elementi koji se "izbacuju" moraju biti različiti. U najgorem slučaju dostupnih vektora može da bude samo:

$$2^{n-k}-\left(1+\binom{l}{1}+\binom{l}{2}+\cdots+\binom{l}{d-2}\right)$$

Da bi predmetna selekcija bila dostupna ovaj broj mora biti pozitivan. Za $i<d$ nejednakost koja mora biti zadovoljena da bi se izabrala i-ta kolona je:

$$2^{n-k}\geq 1+\binom{i-1}{1}+\binom{i-1}{2}+\cdots+\binom{i-1}{i-1}$$

Po binomnom razvoju desna strana je 2^i. Pošto važi $i\leq d$-1$\leq n$-k (kolone imaju dužinu najmanje d-1) ova nejednakost je zasigurno zadovoljena. Binomni koeficijenti $\binom{l}{i}$ su rastući sa porastom l. Dakle ako za l=n-1 (da bi pokušali da izaberemo n-tu kolonu) sledeća nejednakost važi:

$$2^{n-k}\geq 1+\binom{l}{1}+\binom{l}{2}+\cdots+\binom{l}{d-2}$$

onda ona važi i za manje l. Stoga konstatujemo da smo postigli uslove teoreme.

Gilbert-Varshamova nejednakost ukazuje da možemo kreirati kod sa parametrima [n, k, d] ako je odgovarajuća nejednakost zadovoljena. Nažalost do sada nije nađena nijedna procedura osim direktnog pretraživanja koja bi konstruisala predmetni kod. Suprotna teorema Gilbert-Varshamovoj nije tačna. Naime, možda postoji linearni kod koji prevazilazi limite ove teoreme. Takvi kodovi postoje kao što su geometrijski Goppa kodovi.

Singletonova granica se može primjeniti na bilo koji kod. Inače primjenjuje se da bi pokazala nepostojanje određenog koda a ne da bi pokazala postojanje koda sa datim parametrima.

Teorema. Neka je C kod sa kodnom riječju dužine n koji koristi alfabet $\mathbf{F}_q$ sa minimalnom distancom d i l kodnih riječi: Tada:

$$q^{n-(d-1)} \geq l$$

Napomena. Ako je kod linearan, na primjer, [n,k] kod tada broj kodnih riječi je $l=q^k$ i uzimajući logaritam osnovne q Singletonova granica postaje:

$$n-(d-1) \geq k$$

Alternativno ovo se može zapisati kao:

$$n+1 \geq k+d$$

Dakle, ako je $k+d \leq n$ tada ne postoji odgovarajući kod.

Dokaz. Pošto svaki par kodnih riječi se razlikuje na najmanje d pozicija čak i ako ignorišemo posljednjih d-1 pozicija nema dvije kodne riječi takve da će biti iste u prvih n-(d-1) kodnih riječi. Ako otkinemo posljednjih d-1 pozicija svakih l kodnih riječi su i dalje različite. Dakle dobićemo kod sa l kodnih riječi i dužinom bloka n-(d-1). Pošto postoji $q^{n-(d-1)}$ riječi dužine n-(d-1) zajedno zasigurno $l \leq q^{n-(d-1)}$. Ovim je teorema dokazana.

Kodovi koji postižu predmetnu granicu sa jednakošću nazivaju se MDS (Minimum Distance Separating) kodovima.

12.4. Vandermodova matrica, RS i BCH kodovi

Reed-Solomonovi kodovi su prvi kodovi koji su razvijeni da se mogu primjeniti na dekodiranje proizvoljnog broja pogreški. Njihova generalizacija su BCH kodovi. Za kreiranje ovakvih kodova trebamo velika konačna polja gdje polje $\mathbf{F}_2=\{0,1\}$ nije adekvatno. Alternativno mi možemo da koristimo konačno polje proizvoljnih dimenzija (sa p članova gdje je p neki prosti broj) ali praksa, najviše zbog pogodnosti memorijske reprezentacije, zahtjeva da se kodovi uglavnom predstavjaju u binarnom formatu. Stoga je najpopularniji način da se radi sa ovim kodovima da se uzme konačno polje $GF(2^n)$ sa 2^n elemenata. Razvoj klase Hammingovih, RS i BCH kodova dizajniranih na predmetni način trajao je relativno dugo, već gotovo 50 godina sa limitiranim uspjehom posebno za slučajeve kada je kodna riječ veoma dugačka. To je i jedan od razloga što se u međuvremenu umjesto blok kodova pristupilo razvoju i radu sa drugim tipovima kodova, u prvom redu sa konvolucionim i turbo kodovima.

Međutim, naš cilj je da u ovoj sekciji damo još nekoliko teorijskih elemenata za uvođenje ovih kodova kako bi ih bolje osvijetliti i dali više ideja i za njihovu generalizaciju kao i za kodiranje i dekodiranje. Izuzetno važan pojam kojega ćemo uvesti je Vandermondova matrica preko koje se ovi kodovi mogu lakše opisivati u pojedinim situacijama. Poznato je da linearni kod može da

koriguje *e* pogreški ako je 2*e* kolona njegove kontrolne matrice linearno nezavisno. Još u linearnoj algebri smo učili da *l* vektora dužine *l* su linearno nezavisni ako korespondirajuća matrica ima determinantu različitu od nule. Nažalost, računanje determinante nije jednostavno pa nam treba neki trik koji bi nam ukazivao da je neka klasa matrica nesingularna (sa determinantom različitom od nule). Ovdje ćemo predstaviti dva tipa Vandermondovih matrica koje zadovoljavaju ove osobine pod veoma lako-provjerljivim uslovima. Posmatrajmo *n* vektora dužine *n* koji čine matricu:

$$\begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & \cdots & v_{1n} \\ v_{21} & v_{22} & v_{23} & v_{24} & \cdots & v_{2n} \\ v_{31} & v_{32} & v_{33} & v_{34} & \cdots & v_{3n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & v_{n3} & v_{n4} & \cdots & v_{nn} \end{bmatrix}$$

Jedan od oblika Vandermondove matrice za koji se pouzdano zna da produkuje nenultu determinantu je Vandermondova matrica koja je nesingularna kada je svako $x_i \neq x_j$ za $i \neq j$:

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & x_3 & x_4 & \cdots & x_n \\ x_1^2 & x_2^2 & x_3^2 & x_4^2 & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & x_3^{n-1} & x_4^{n-1} & \cdots & x_n^{n-1} \end{bmatrix}$$

Determinanta ove matrice je takođe poznata i ima oblik:

$$\det(M) = (-1)^{n(n-1)/2} \prod_{i<j} (x_i - x_j)$$

Na osnovu matematike iz srednje škola znamo da je prozvod nenultih elemenata različit od nule ali ovo ne mora da važi za konačna polja već je osobina koju zadovoljavaju integralni domeni kao što je recimo skup cijelih brojeva.

Posmatrajmo polje od *k* konačnih elemenata koji će biti stepeni od α: 1, α, α^2, α^3, ..., α^l. Onda važi:

$$\det \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \alpha^3 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & (\alpha^2)^2 & (\alpha^3)^2 & \cdots & (\alpha^{n-1})^2 \\ 1 & \alpha^3 & (\alpha^2)^3 & (\alpha^3)^3 & \cdots & (\alpha^{n-1})^3 \\ 1 & \alpha^4 & (\alpha^2)^4 & (\alpha^3)^4 & \cdots & (\alpha^{n-1})^4 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{n-1} & (\alpha^2)^{n-1} & (\alpha^3)^{n-1} & \cdots & (\alpha^{n-1})^{n-1} \end{bmatrix} \neq 0$$

Nenulti elementi se mogu preurediti na bilo koji pogodan poredak α^{l_1}, α^{l_2}, α^{l_3}, ..., α^{l_n} pa i tada važi:

$$\det\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha^{l_1} & \alpha^{l_2} & \alpha^{l_3} & \cdots & \alpha^{l_{n-1}} \\ 1 & (\alpha^{l_1})^2 & (\alpha^{l_2})^2 & (\alpha^{l_3})^2 & \cdots & (\alpha^{l_{n-1}})^2 \\ 1 & (\alpha^{l_1})^3 & (\alpha^{l_2})^3 & (\alpha^{l_3})^3 & \cdots & (\alpha^{l_{n-1}})^3 \\ 1 & (\alpha^{l_1})^4 & (\alpha^{l_2})^4 & (\alpha^{l_3})^4 & \cdots & (\alpha^{l_{n-1}})^4 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & (\alpha^{l_1})^{n-1} & (\alpha^{l_2})^{n-1} & (\alpha^{l_3})^{n-1} & \cdots & (\alpha^{l_{n-1}})^{n-1} \end{bmatrix} \neq 0$$

Ako se redovi ili vrste nesingularne matrice pomnože sa nenultim vrijednostima matrica ostaje nesingularna. Ova osobina se koristi da bi se dobila druga forma Vandermondove matrice koja se dobija tako što se svaka vrsta matrice pomnoži redom sa x_1, x_2, ..., x_n:

$$M = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & \cdots & x_n \\ x_1^2 & x_2^2 & x_3^2 & x_4^2 & \cdots & x_n^2 \\ x_1^3 & x_2^3 & x_3^3 & x_4^3 & \cdots & x_n^3 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^n & x_2^n & x_3^n & x_4^n & \cdots & x_n^n \end{bmatrix}$$

Već smo se uvjerili da za svaki blok kod možemo koristiti kontrolnu matricu i to sa elementima preko proizvoljnog polja $\mathbf{F}_q$. Svaki ciklični blok kod dužine n se može prikazati preko generatorskog polinoma $\mathbf{g}(x)$ koji je djelioc x^n-1. Kolekcija svih kodnih riječi specificiranih sa $\mathbf{g}(x)$ je kolekcija svih polinoma stepena $<n$ koji su polinomi množioci $\mathbf{g}(x)$. Kod $[n,k]$ ima osobinu $k=n-\deg(\mathbf{g})$ i $\mathbf{g}(x)=c_0+c_1x+\ldots+c_sx^s$. Generatorska matrica koda se može zapisati kao:

$$\mathbf{G} = \begin{bmatrix} c_0 & c_1 & \cdots & c_s & 0 & 0 & \cdots & 0 \\ 0 & c_0 & \cdots & c_{s-1} & c_s & 0 & \cdots & 0 \\ 0 & 0 & \cdots & c_{s-2} & c_{s-1} & c_s & \cdots & 0 \end{bmatrix}$$

Neka je:

$$\mathbf{h}(x) = \frac{x^n - 1}{\mathbf{g}(x)} = b_0 + b_1x + \ldots + b_tx^t \quad (t+s=n)$$

Jedan tip kontrolne matrice se može zapisati kao:

$$H = \begin{bmatrix} b_t & b_{t-1} & b_{t-2} & b_{t-3} & \cdots & b_1 & b_0 & 0 & \cdots & 0 \\ 0 & b_t & b_{t-1} & b_{t-2} & \cdots & b_2 & b_1 & b_0 & \cdots & 0 \\ 0 & 0 & b_t & b_{t-1} & \cdots & b_3 & b_2 & b_1 & \cdots & 0 \\ & & \cdots & & & & & & & \end{bmatrix}$$

Sada se mogu kreirati različiti oblici kontrolnih matrica koje bolje ilustruju sposobnosti kodova za ispravljanje grešaka preko linearne nezavisnosti kolona. Neka je $\mathbf{F}_{q^m}$

Zadaci za vježbu

12.1. Pokazati da CRC sa generatorskim polinomom $1+x^2+x^3+x^4$ ne uspijeva da detektuje dvije pogreške kada su razdvojene za distancu koja je multipl broja 7.

Rješenje. Uzmimo da su se pogreške dogodile na razmaku od 7 mjesta što znači da se sindrom može zapisati kao:

$$s(x)=(x^k+x^{k+7})d(x)$$

Stoga je potrebno provjeriti da li je generatorski polinom djelioc polinoma $(1+x^7)$.

$x^7+1 : x^4+x^3+x^2+1 = x^3+x^2+1$
$x^7+x^6+x^5+x^3$

$x^6+x^5+x^3+1$
$x^6+x^5+x^4+x^2$

$x^4+x^3+x^2+1$

Dakle, možemo zaključiti da u slučaju da se pojave pogreške na razmaku koji je multipl broja 7 neće biti detektovane.

12.2. Pokazati da CRC sa generatorskim polinomom $1+x+x^2+x^3$ ne uspijeva da detektuje dvije pogreške kada su razdvojene za 4 bita.

Rješenje. Postupak je isti kao u prethodnom zadatku. Dvije greške se neće uočiti na rastojanju koje je multipl četvorke ako je x^4+1 djeljivo sa predmetnim generatorskim polinomom. Dakle, provjeravamo:

$x^4+1 : x^3+x^2+x+1 = x+1$
$x^4+x^3+x^2+x$

---

x^3+x^2+x+1

12.3. Neka je $g(x)=x^3+x+1$ generatorski polinom za CRC. Odrediti "pametan" algoritam za računanje CRC-a za niz:

11100011010100011001110

tako da se izbjegne "dosadno" dijeljenje.

Rješenje. Dijeljenje je operacija koja se svodi na oduzimanje a oduzimanje se kod binarne algebre svodi na sabiranje po modulu 2 odnosno na ekskluzivno ili operaciju. Procedura koja može biti relativno brza i efikasna u predmetnom slučaju se može sastojati od određivanja jedinice najviše važnosti a zatim se od te jedinice i naredna 4 bita oduzme kombinacija [1,0,1,1]. Procedura se zatim nastavlja sa jedinicom najveće važnosti u ostatku riječi. Navedeni koraci u predmetnoj poruci su dati:

11100011 01010001 10011110
01010011 01010001 10011110
00001011 01010001 10011110

00000000 01010001 10011110
00000000 00001001 10011110
00000000 00000010 10011110
00000000 00000000 01011110
00000000 00000000 00000110

Ovdje vidimo da je ostatak dijeljenja 110. Procedura je obavljena relativno efikasno.

12.4. Postoji li binarni kod sa 17 riječi i minimalnom distancom 3 dužine 7?

Rješenje. Kod sa minimalnom distancom 3 može da ispravi jednu riječ. Dužine 7 podrazumijeva da ima ukupno mogućih riječi 2^7=128. Ako želite da postignete da može da ispravlja jednu kodnu riječ oko svake ispravne kodne riječi mora se formirati sfera koja pokriva pored ispravne kodne riječi i susjednih sedam od koje se svaka ispravna kodna riječ razlikuje za 1. Stoga svakoj ispravnoj kodnoj riječi odgovara sfera dimenzija 8 a to dalje znači da možemo imati najviše

27/8=16 ispravnih kodnih riječi

Dakle nije moguće spakovati u datom prostoru 17 ispravnih kodnih riječi odnosno spakovati 17 sfera.

12.5. Postoji li binarni kod sa 29 kodnih riječi, minimalnom distancom 3 dužine 8?

Rješenje. Ponovimo postupak ispitivanja pakovanja sfera dobijamo:

$$2^8/9=28.44$$

čime ponovo potvrđujemo da nije moguće imati kod datih karakteristika.

12.6. Postoji li binarni kod sa 7 kodnih riječi, minimalnom distancom 5 dužine 8?

Rješenje. Ponovimo proceduru provjere na osnovu pakovanja sfera. Ukupan broj mogućih kombinacija sa 8 bita je 2^8=256. Za ostvarivanje pakovanja potrebno je u ovom slučaju formirati sferu koja pored ispravne kodne riječi obuhvata i one riječi koje su na rastojanju 1 (ima ih 8) i one koje su na rastojanju 2 od ispravne kodne riječi (ima ih $\binom{8}{2}=28$). Dakle, ukupan broj sfera koje je teorijski moguće spakovati u datom prostoru je:

$$\frac{2^8}{1+8+28}=6.92$$

I u ovom slučaju smo prosto na osnovu pakovanja sfera zaključili da nije moguć kod sa traženim karakteristikama.

12.7. Postoji li binarni linearni kod dimenzije 2 sa minimalnom distancom 3 i dužinom bloka 5?

Rješenje. Po uslovu zadatka k=2, q=2, d=3 i n=5. Po Gilbert-Varshamovoj granici slijedi:

$$q^{n-k}-1>(q-1)\binom{n-1}{1}+\cdots+(q-1)^{d-3}\binom{n-1}{d-3}+(q-1)^{d-2}\binom{n-1}{d-2}$$

$$2^3 - 1 > \binom{4}{1}$$

Dakle, pošto je navedena nejednakost zadovoljena moguće je konstruisati ovakav kod.

12.8. Postoji li binarni linearni kod dimenzije 3 sa minimalnom distancom 3 sa blok dužine 6?

Rješenje. Po uslovu zadatka k=3, d=3, q=2 i n=6. Po Gilbert-Varshamovoj granici slijedi:

$$2^3 - 1 > \binom{5}{1}$$

Dakle, pošto je navedena nejednakost zadovoljena moguće je konstruisati ovakav kod.

12.9. Postoji li binarni linearni kod dimenzija 3 sa minimalnom distancom 3 i dužinom bloka 5?

Rješenje. Po uslovu zadatka k=3, d=3, n=5 i q=2. Po Gilbert-Varshamovoj granici slijedi:

$$q^{n-k} - 1 > (q-1)\binom{n-1}{1} + \cdots + (q-1)^{d-3}\binom{n-1}{d-3} + (q-1)^{d-2}\binom{n-1}{d-2}$$

$$2^3 - 1 > \binom{4}{1}$$

Dakle, pošto je navedena nejednakost zadovoljena moguće je konstruisati ovakav kod.

12.10. Postoji li binarni linearni kod dimenzija 3 sa minimalnom distancom 4 i dužinom bloka 8?

Rješenje. Posmatrajmo Gilbert-Varshamovu granicu:

$$2^{8-3} - 1 > \binom{7}{1} + \binom{7}{2}$$

$$31 > 7 + 21$$

Dakle, Gilbert Varshamova granica je zadovoljena pa predmetni kod postoji.

12.11. Postoji li binarni linearni kod dimenzija 2 sa minimalnom distancom 4 i dužinom bloka 7?

Rješenje. Ponovo posmatrajmo Gilbert-Varshamovu granicu:

$$2^5 - 1 > \binom{4}{1} + \binom{4}{2} = 4 + 6$$

Kako je uslov ove granice zadovoljen možemo zaključiti da je predmetni kod moguće konstruisati.

12.12. Postoji li binarni kod sa 9 kodnih riječi minimalnom distancom 3 i sa dužinom 5?

Rješenje. Ne, navedeni kod ne postoji pošto nije zadovoljena Hammingova granica:

$$\frac{2^5}{1+5} \geq 9$$

12.13. Postoji li binarni kod sa 17 kodnih riječi sa minimalnom distancom 5 i sa dužinom 8?

Rješenje. Oko svake od 17 kodnih riječi treba da možemo da kreiramo sferu na distanci 2. Ovo se može provjeriti sa sljedećom nejednakošću:

$$\frac{2^8}{1+8+\binom{8}{2}} \geq 17$$

koja nije zadovoljena pa kod sa predmetnim karakteristikama ne može postojati.

12.14. Imamo primljenu sekvencu [0.8 0.2 -0.3 0.5 0.4 0.1 0.9 -0.8]. Putem Hadamardove matrice dekodirati poruku.

Rješenje. Pomnožimo Hadamardovu matricu sa vektorom koji predstavlja primljenu sekvencu:

$$\mathbf{H}_8 r^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 0.8 \\ 0.2 \\ -0.3 \\ 0.5 \\ 0.4 \\ 0.1 \\ 0.9 \\ -0.8 \end{bmatrix} = \begin{bmatrix} 1.8 \\ 1.8 \\ 1.2 \\ 0 \\ 0.6 \\ -2.2 \\ 0.4 \\ 2.8 \end{bmatrix}$$

Najveća apsolutna vrijednost primljenog vektora je na poziciji 8 i iznosi 2.8. Dakle, dekodirana kodna riječ je [1 -1 -1 1 -1 1 1 -1].

12.15. Posmatrajmo sljedeću kodnu riječ 1 1 0 0 1 1 0 0 1 0 0 1 0 1 0 1. Poruku dekodirati Reed Mullerovim kodom.

Rješenje. Ako je kodna riječ smještena u vektor **z** tada možemo izvršiti njeno dekodiranje množenjem EL_4 sa $\mathbf{z}^T$ čime dobijamo:

$$EL_4 \mathbf{z}^T = \begin{bmatrix} 4 \\ 4 \\ 2 \\ 5 \\ 8 \end{bmatrix}$$

Dakle, peta kolona koja predstavlja sve jedinice daje najveći "odziv" pa zaključujemo da u ovom slučaju predstavlja ispravnu kodnu riječ.

12.16. Dokazati izraz za determinantu Vandermondove matrice i pokazati da navedeni izraz važi za slučaj konačnih polja.

Rješenje. Potrebno je za matricu:

$$M=\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & x_3 & x_4 & \cdots & x_n \\ x_1^2 & x_2^2 & x_3^2 & x_4^2 & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & x_3^{n-1} & x_4^{n-1} & \cdots & x_n^{n-1} \end{bmatrix}$$

dokazati da joj je determinanta jednaka

$$\det(M)=(-1)^{n(n-1)/2}\prod_{i<j}(x_i-x_j)$$

te da navedena relacija važi za konačna polja $x_i\in\{0, 1, \ldots, P\text{-}1\}$ gdje je P prost broj dok su sabiranje i oduzimanje definisani po modulu prostog broja P. Dokaz obavimo rekurzivno. Posmatrajmo za n=2 dobijamo matricu:

$$M_2=\begin{bmatrix} 1 & 1 \\ x_1 & x_2 \end{bmatrix}$$

Determinanta ove matrice je $\det(M_2)=x_2\text{-}x_1$. Ovo je u skladu sa tvrdnjom. Sada pretpostavimo da je navedena tvrdnja tačna za matricu dimenzija n pa dokažimo za matricu dimenzija n+1. Determinanta dimenzija n+1 je jednaka:

$$M_{n+1}=\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & \cdots & x_n & x_{n+1} \\ x_1^2 & x_2^2 & x_3^2 & x_4^2 & \cdots & x_n^2 & x_{n+1}^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_1^{n-1} & x_2^{n-1} & x_3^{n-1} & x_4^{n-1} & \cdots & x_n^{n-1} & x_{n+1}^{n-1} \\ x_1^{n} & x_2^{n} & x_3^{n} & x_4^{n} & \cdots & x_n^{n} & x_{n+1}^{n} \end{bmatrix}$$

Determinantu ove matrice možemo sračunati razvojem po posljednjoj vrsti kao:

$$\det(M_{n+1})=\sum_{i=1}^{n+1}(-1)^{n+i}\,x_i^{n+1}\det\left(\begin{bmatrix} 1 & \ldots & 1 & 1 & \cdots & 1 \\ x_1 & \cdots & x_{i-1} & x_{i+1} & \cdots & x_n \\ x_1^2 & \cdots & x_{i-1}^2 & x_{i+1}^2 & \cdots & x_n^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & \cdots & x_{i-1}^{n-1} & x_{i+1}^{n-1} & \cdots & x_n^{n-1} \end{bmatrix}\right)$$

Pod pretpostavkom da navedena teorema važi za matrice manjih dimenzija tada slijedi

$$\det(M_{n+1}) = \sum_{i=1}^{n+1} (-1)^{n+i} x_i^{n+1} (-1)^{n(n-1)/2} \prod_{\substack{l<k \\ l\neq i, k\neq i}} (x_l - x_k) =$$

$$= (-1)^{n(n+1)/2} \sum_{i=1}^{n+1} (-1)^{i} x_i^{n+1} \prod_{\substack{l<k \\ l\neq i, k\neq i}} (x_l - x_k)$$

$$= (-1)^{n(n+1)/2} \prod_{\substack{i,j=1 \\ i<j}}^{n+1} (x_i - x_j)$$

LITERATURA

[1] R. W. Hamming, Coding and information theory, Prentice-Hall, 1986.
[2] A. Gersho, R. Gray, "Vector quantization and signal compression," Kluwer Academic Publishers, 2003 (deveto izdanje), dostupno i putem Google books.
[3] vashe.org/turbo/turbo_primer_0.0.pdf, pristup 29.12.2013.
[4] C. Berrou, A. Glavieux, P. Thijtimasjshima: "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes", Proc. of IEEE Int. Conf. on Communications, May 1993, Ženeva, Švajcarska, 1064-1070.
[5] B. Sklar: "A primer on turbo code concepts," IEEE Communications Magazine, Dec. 1997, str. 94-102.
[6] M. C. Valenti and J. Sun: "Turbo codes", DOWLA: Handbook of RF and wireless technologies, str. 375-400.
[7] Matlab Central turbo kodovi, http://www.mathworks.com/matlabcentral/fileexchange/39423-turbo-code/content/turbo.m, pristupljeno 29.12.2013.
[8] http://www.mathworks.com/matlabcentral/fileexchange/authors/13257, LZW kodovi.
[9] P. N. Ivaniš, D. Drajić, Uvod u teoriju informacija i kodovanje, Akademska misao, Beograd, 2009.
[10] V. Sinković, Informacija, simbolika, semantika, Školska knjiga, 1997.
[11] P. B. Garrett, The mathematics of coding theory, Pearson, 2003.
[12] R. Roth, Introduction to coding theory, Cambridge University Press, 2006.
[13] B. Ryabko, A. Fionov, Basics of contemporary cryptography for IT practioners, World Scientific, 2005.
[14] J. I. Hall, Notes on coding theory, Department of Mathematics, Michigan State University, 2003.
[15] S. A. Tretter, Predavanja i drugi materijali iz teorije kodova, Department of Electrical and Computer Engineering, University of Maryland.
[16] D. J. C. MacKay, Information Theory, Inference, and Learning Algorithms, Cambridge University Press, 2003.
[17] T. Moon, Information theory, Utah State University, predavanja i ostali materijali.
[18] J. Bierbrauer, Introduction to coding theory, Champan & Hall/CRC, 2005.
[19] G. Jovanović-Doleček, Teorija vjerovatnoće, Sarajevo, 1990.