Kodno stablo

Jedan od pogodnih načina da se kod vizuelno predstavi, pa i konstruiše, tj. ispišu kodne riječi, je **kodno stablo**. Pomoću stabla se istovremeno može pratiti način rada dekodera.

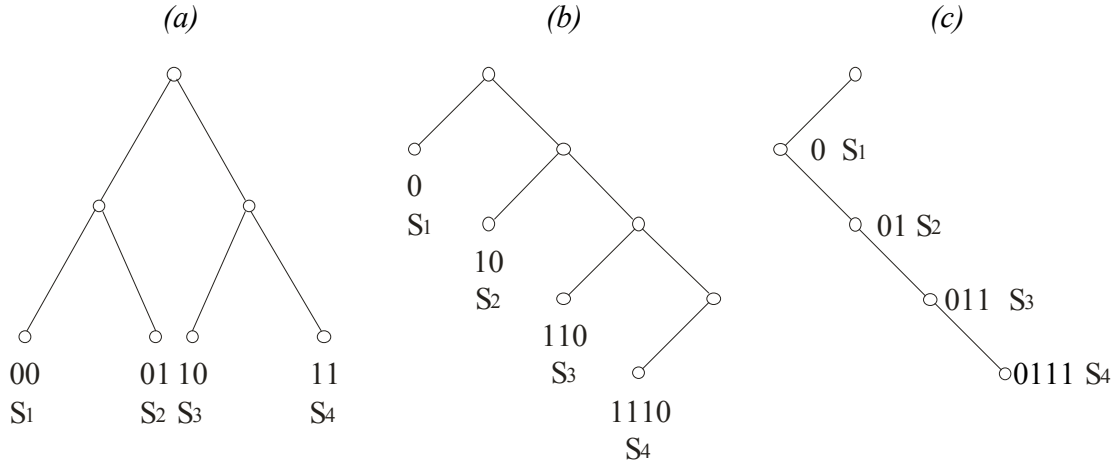
Dekoder kreće iz korjena stabla i zavisno od primljenog simbola ide jednom od grana. Grana ima onoliko koliko ima različitih simbola u kodu, tj. kolika je kodna osnova. Kada stigne sledeći simbol, grana se račva i iz čvora opet izlaze nove grane. Kada stigne poslednji simbol kodne riječi, dekodovan je odgovarajući simbol izvora, a koder se vraća u korjen stabla.

Primjer:

Neka su zadati sledeći kodovi koji kodiraju simbole izvora, čija izvorna lista ima 4 simbola $S = \{S_1, S_2, S_3, S_4\}$. Nacrtati kodna stabla.

S	*(a)*	*(b)*	*(c)*
S_1	00	0	0
S_2	01	10	01
S_3	10	110	011
S_4	11	1110	0111

Dogovorno je usvojeno da se pri pojavi **0** u kodnoj riječi ide ulijevo po stablu, a pri pojavi **1** – udesno.



Kodna lista za ovaj primjer ima dva simbola {0, 1}, pa iz čvora možemo imati **dvije grane**.

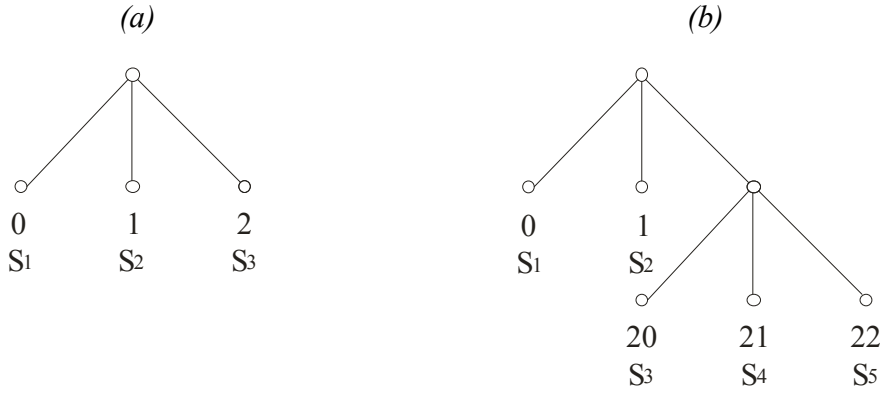
Ranije smo već rekli da je kod **trenutan**, ako se svaka kodna riječ u sekvenci može dekodovati bez oslanjanja na sljedeće kodne simbole. Potreban i dovoljan uslov da kod bude trenutan je da ni jedna cijela kodna riječ ne bude prefiks neke druge kodne riječi.

U prethodnom primjeru kodovi *(a)* i *(b)* su trenutni, a kod *(c)* nije trenutan, jer je svaka kodna riječ prefiks sljedeće kodne riječi. Takođe, ako pogledamo kodna stabla ovih kodova vidimo da je kodno stablo koda *(c)* **degenerisano**, pa on nije trenutan.

Na osnovu ovoga možemo zaključiti da korišćenje **nedegenerisanog kodnog stabla**, za konstrukciju koda, garantuje dobijanje trenutnog koda.

Primjer:

Neka je dat ternarni kod (broj simbola kodne liste je $r = 3$). Kodni simboli su označeni sa 0, 1 i 2. Na slici *(a)* je dato kodno stablo za izvor sa tri simbola ($g = 3$) $S = \{S_1, S_2, S_3\}$, a na slici *(b)* je dato kodno stablo za izvor sa pet simbola $S = \{S_1, S_2, S_3, S_4, S_5\}$



Kraftova nejednakost

Kraftova nejednakost daje potreban i dovoljan uslov za **postojanje trenutnog koda** sa kodnim riječima određene dužine.

Neka je dat izvor $S=\{S_1, S_2, ..., S_g\}$ i neka je data kodna lista $X=\{X_1, X_2, ..., X_r\}$, tj. neka je r osnova koda. Neka je dužina kodne riječi X_i kojom se koduje simbol izvora S_i jednaka l_i ($i = 1, 2, ..., g$). Tada je potreban i dovoljan uslov za postojanje trenutnog koda sa kodnim riječima dužine $l_1, l_2, ..., l_g$ zadovoljenje Kraftove nejednakosti:

$$\sum_{i=1}^{g} r^{-l_i} \leq 1$$

Za binarni kod ($r = 2$) ova nejednakost ima oblik:

$$\sum_{i=1}^{g} 2^{-l_i} \leq 1$$

Ova Kraftova nejednakost daje dogovor da li se koristeći kodne riječi određene dužine može napraviti trenutan kod, ali ne i kako konstruisati taj kod.

Primjer:

Neka su dati sledeći binarni kodovi (r = 2) za izvor sa g = 4 simbola. Provjeriti da li zadovoljavaju Kraftovu nejednakost i da li su trenutni.

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 kod *(a)* $\sum_{i=1}^{4} 2^{-li} = 2^{-2} + 2^{-2} + 2^{-2} + 2^{-2} = \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} = 1 \leq 1$ (zadovoljava)

Za kod *(b)* $\sum_{i=1}^{4} 2^{-li} = 2^{-1} + 2^{-3} + 2^{-3} + 2^{-3} = \frac{1}{2} + \frac{3}{8} = \frac{7}{8} \leq 1$ (zadovoljava)

Za kod *(c)* $\sum_{i=1}^{4} 2^{-li} = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = \frac{1}{2} + \frac{1}{4} + \frac{2}{8} = 1 \leq 1$ (zadovoljava)

Za kod *(d)* $\sum_{i=1}^{4} 2^{-li} = 2^{-1} + 2^{-3} + 2^{-3} + 2^{-2} = \frac{1}{2} + \frac{2}{8} + \frac{1}{4} = 1 \leq 1$ (zadovoljava)

Za kod *(e)* $\sum_{i=1}^{4} 2^{-li} = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-2} = \frac{1}{2} + \frac{2}{4} + \frac{1}{8} = \frac{9}{8} > 1$ (ne zadovoljava)

Na osnovu prethodnih provjera smo dobili da kodovi *(a)*, *(b)*, *(c)*, *(d)* zadovoljavaju Kraftovu nejednakost, dok kod *(e)* ne zadovoljava tu nejednakost, pa ne može biti trenutni.

Treba obratiti pažnju na kod *(d)*, iako on zadovoljava Kraftovu nejednakost, ne može biti trenutan, jer je simbol izvora S_4 prefiks od S_3.

Srednja dužina kodne riječi

Za dati izvor se može sa datom kodnom listom konstruisati **veći broj trenutnih kodova**.

- Koje izabrati?

- Kakav kriterijum pri izboru treba primjeniti?

Kod statističkog kodovanja **dužina kodne riječi** je osnovni kriterijum pri izboru trenutnog koda.

Neka su dati izvor $S=\{S_1, S_2, ..., S_g\}$ sa vjerovatnoćama pojavljivanja simbola izvora $P(S_i)$ $(i = 1, 2, ..., g)$ i kodna lista $X\ \{X_1, X_2, ..., X_r\}$ i neka kodna riječ kojom se koduje simbol S_i ima dužinu l_i, tada je srednja dužina kodne riječi data relacijom:

$$L = \sum_{i=1}^{g} P(S_i) \cdot l_i$$

Zaključak:

Pri statističkom kodovanju od više mogućih kodova treba odabrati onaj koji ima manju srednju dužinu kodne riječi.

Definicija:

Jedan kod je **kompaktan (optimalan)** za jedan izvor, ako je srednja dužina kodne riječi manja ili jednaka od srednjih dužina kodnih riječi svih ostalih trenutnih kodova za isti izvor i istu kodnu listu.

Prema tome, **cilj statističkog kodovanja** je traženje kompaktnog (optimalnog) koda.

-Treba postaviti pitanje kako se može ocijeniti da li je neki kod kompaktan ili ne i koja je donja granica srednje dužine kodne riječi ispod koje se ne može ići.

Donja granica za izvore bez i sa memorijom je određena sledećim uslovom:

$$L \geq \frac{H(S)}{\log r}$$

gdje je $H(S) = -\sum_{i=1}^{g} P_i(S_i) \log P_i(S_i)$ entropija izvora bez memorije (kad imamo slučaj izvora sa memorijom koristimo izraz za entropiju izvora sa memorijom).

Za slučaj binarnog koda (r = 2) donja granica je definisana:

$$L \geq H(S) \qquad l_i = \log_r \frac{1}{P_i(S_i)}$$

Primjer:

Izvor sa $g = 4$ simbola, treba kodovati binarnim kodom ($r = 2$), pri čemu su moguća dva skupa vjerovatnoća:

	(a)			*(b)*		
S	P_i	$l_i = \log(1/P_i)$	X_i	P_i	$l_i = \log(1/P_i)$	X_i
S_1	$\frac{1}{4}$	2	00	$\frac{1}{2}$	1	0
S_2	$\frac{1}{4}$	2	01	$\frac{1}{4}$	2	10
S_3	$\frac{1}{4}$	2	10	$\frac{1}{8}$	3	110
S_4	$\frac{1}{4}$	2	11	$\frac{1}{8}$	3	111

$$H_a(S) = \log 4 = 2 \text{ Sh/simb}$$

Za kod *(a)*

$$L_a = 4 \cdot \frac{1}{4} \cdot 2 = 2 \text{ b/simb}$$

$$H_b(S) = \frac{1}{2}\log 2 + \frac{1}{4}\log 4 + 2 \cdot \frac{1}{8}\log 8 = 1{,}75 \text{ Sh/simb}$$

Za kod *(b)*

$$L_b = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + 2 \cdot \frac{1}{8} \cdot 3 = 1{,}75 \text{ b/simb}$$

Oba koda *(a)* i *(b)* su optimalni kodovi, pri čemu kod *(b)* ima manju srednju dužinu kodne riječi.

U slučaju kada vjerovatnoće pojavljivanja simbola izvora nijesu oblika $\frac{1}{r^k}$ (kod prethodnog primjera vjerovatnoće su bile oblika $\frac{1}{2^k}$) tada logaritam recipročne vrijednosti vjerovatnoće nije cio broj, pa se postavlja pitanje kako uzeti dužinu kodne riječi?

Dužina kodne riječi se sada bira prema sledećem pravilu:

$$\boxed{\log \frac{1}{P_i} \le l_i \le \log \frac{1}{P_i} + 1} \quad (i = 1, 2, ..., g)$$

Odnosno srednja dužina kodne riječi mora zadovoljavati sledeću vrijednost:

$$\boxed{H(S) \le L \le H(S) + 1}$$

Sve ovo važi i za *n-to* proširenje izvora (S^n), u tom slučaju je:

$$\boxed{H(S^n) \le L_n \le H(S^n) + 1}$$

gdje je L_n - **srednja dužina kodne riječi za simbole proširenog izvora**, tj. za sekvencu od po *n* simbola orginalnog izvora.

Pošto za entropiju važi: $H(S^n) = n \cdot H(S)$ dobija se:

$$n \cdot H(S) \le L_n \le n \cdot H(S) + 1 \text{ tj.}$$

$$\boxed{H(S) \le \frac{L_n}{n} \le H(S) + \frac{1}{n}}$$

Veličina $\frac{L_n}{n}$ predstavlja srednju dužinu kodne riječi po simbolu orginalnog izvora.

Primjer:

Neka je dat binarni izvor bez memorije $S=\{S_1, S_2\}$ koje treba kodovati binarnim kodom ($r = 2$) i neka se simboli izvora pojavljuju sa vjerovatnoćama $\frac{7}{8}$ i $\frac{1}{8}$.

S	P_i	X_i
S_1	$\frac{7}{8}$	0
S_2	$\frac{1}{8}$	1

Ovdje je $H(S)$ = 0,54 Sh/simb, a srednja dužina kodne riječi $L = 1 \cdot \frac{7}{8} + 1 \cdot \frac{1}{8} = 1$ b/simb.

Ako se izvrši drugo (n = 2) proširenje izvora, dobija se:

S^2	P_i	X_i
$S_1\ S_1$	$\frac{49}{64}$	0
$S_1\ S_2$	$\frac{7}{64}$	10
$S_2\ S_1$	$\frac{7}{64}$	110
$S_2\ S_2$	$\frac{1}{64}$	111

$$H(S^2) = 2 \cdot H(S) = 1{,}0871 \text{ Sh/simb}$$

Prosječna dužina kodne riječi je sada L_2 = 1,359 b/simb skupa S_2, odnosno $L_2/2$ = 0,68 b/simb orginalnog izvora *S,* što je srednja dužina kodne riječi znatno bliža entropiji (umjesto 1 b/simb dobijeno je 0,68 b/simb). Daljim proširivanjima dobijale bi se vrijednosti sve bliže vrijednosti entropije. Međutim, cijena koja se plaća je proširivanje kodne liste.

Šenon–Fanoov postupak kodiranja

Ovaj postupak će biti prikazan detaljnije za slučaj binarnog koda. Osnovna Šenonova ideja je da se simboli izvorne liste urede po nerastujućim vjerovatnoćama, tj. $P_1 \geq P_2 \geq ... \geq P_g$ (ako simboli imaju jednake vjerovatnoće njihov raspored u okviru tog podskupa nije važan), a zatim se skup simbola podijeli na dva podjednako vjerovatna dijela (ili bar približno podjednako vjerovatna dijela). Kodne riječi za simbole jednog podskupa počinjaće sa **0**, a za simbole drugog podskupa sa **1**. Svaki od ovih podskupova se dalje dijeli na dva podjednako vjerovatna ili približno vjerovatna podskupa dodajući na odgovarajuće mjesto u kodnoj riječi **0** ili **1**. Kada u svim podskupovima ostane samo po jedan simbol kodovanje je završeno.

Primjer:

a)

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

$$H(S)=\frac{1}{2}\log 2+\frac{1}{4}\log 4+2\cdot\frac{1}{8}\log 8=1{,}75 \text{ Sh/simb}$$

$$L=\frac{1}{2}\cdot 1+\frac{1}{4}\cdot 2+\frac{1}{8}\cdot 3+\frac{1}{8}\cdot 3=1+\frac{3}{4}=1{,}75 \text{ b/simb}$$

Dobijen je kompaktni (optimalan) kod i srednja dužina kodne riječi jednaka je entropiji.

b)

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

c)

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

d)

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	110
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

Napomena: Postojala je mogućnost da prva podjela na dva podskupa bude između simbola S_1 i S_2.

Hafmenov postupak kodiranja

Za razliku od Šenon-Fanoove metode, Hafmenova metoda omogućava **direktno nalaženje kompaktnog koda**.

Kao i u prethodnom slučaju, Hafmenov postupak kodovanja će biti opisan na primjeru binarnog koda.

Neka je dat izvor S sa simbolima $\{S_1, S_2, ..., S_g\}$ čije su vjerovatnoće pojavljivanja

P_i (i=1, 2, ..., g). Simbole treba urediti po nerastujućim vjerovatnoćama tj. $P_1 \geq P_2 \geq ... \geq P_g$. Ukoliko simboli imaju iste vjerovatnoće, njihov redoslijed pri ovome uređivanju nije važan. Sada treba dva najmanje vjerovatna simbola S_{g-1} i S_g zamijeniti jedinstvenim ekvivalentnim simbolom S'_{g-1} , tj. formirati njihovu uniju. Vjerovatnoća pojavljivanja ovoga simbola je jednaka zbiru vjerovatnoća simbola koje on zamjenjuje. Na taj način je izvršena **redukcija** izvora S sa g simbola na izvor S_1 koji ima g-1 simbol. Sada se ponovo uređuju simboli redukovanog izvora i vrši nova redukcija. Ovaj proces se nastavlja sve dok se ne izvrši g-1 redukcija i ne dobije izvor S_{g-1} sa samo dva simbola.

Sada se polazi od redukovanog izvora sa dva simbola i počinju da se pišu kodne riječi na taj način što se jednom od simbola dodjeljuje bit **0** kao kodna riječ, a drugom bit **1**. Ide se korak po korak unazad i unije se rastavljaju. Pri svakom rastavljanju dodaje se na odgovarajuće mjesto u kodnu riječ bit **0** ili **1**, dok se ne dođe na prvobitni izvor.

Primjer:

S	P_i	X_i	S_1		S_2		S_3		S_4	
S_1	0,65	0	0,65	0	0,65	0	0,65	0	0,65	0
S_2	0,15	11	0,15	11	0,15	11	•0,20*	10	*0,35	1
S_3	0,08	101	0,08	101	*0,12•	100	0,15*	11		
S_4	0,05	1001	•0,07*	1000	0,08•	101				
S_5	0,04•	10000	0,05*	1001						
S_6	0,03•	10001								

Srednja dužina kodne riječi je L = 1,74 b/simb.

U primjeru koji slijedi biće prikazano šta se dešava ako se u okviru Hafmenovog koda pojavi više simbola sa istim vjerovatnoćama i ako se novodobijeni simbol pri redukovanju uvijek stavlja na poslednje mjesto u skupu vjerovatnoća ili se stavlja na proizvoljno.

Primjer:

a) Simbol dobijen redukcijom stavlja se uvijek na poslednje mjesto u skupu simbola međusobno jednakih vjerovatnoća.

S	P_i	X_i	S_1		S_2		S_3		S_4	
S_1	0,5	0	0,5	0	0,5	0	0,5	0	0,5	0
S_2	0,2	11	0,2	11	0,2	11	•0,3*	10	*0,5	1
S_3	0,1	101	0,1	101	*0,2•	100	0,2*	11		
S_4	0,1	1000	0,1*	1000	0,1•	101				
S_5	0,07•	10010	•0,1*	1001						
S_6	0,03•	10011								

Srednja dužina kodne riječi je L = 2,1 b/simb, a entropija je $H(S)$ = 2,05 Sh/simb.

b) Simbol dobijen prvom redukcijom stavlja se na **prvo mjesto** u skupu simbola međusobno jednakih vjerovatnoća, a u kasnijim redukcijama na poslednje mjesto.

S	P_i	X_i	S_1		S_2		S_3		S_4	
S_1	0,5	0	0,5	0	0,5	0	0,5	0	0,5	0
S_2	0,2	11	0,2	11	0,2	11	•0,3*	10	*0,5	1
S_3	0,1	1000	•0,1	101	*0,2•	100	0,2*	11		
S_4	0,1	1001	0,1*	1000	0,1•	101				
S_5	0,07•	1010	0,1*	1001						
S_6	0,03•	1011								

Srednja dužina kodne riječi je opet ista L = 2,1 b/simb, ali struktura dužina kodnih riječi je različita. To, međutim, ne smeta, jer je po definiciji kompaktan kod onaj koji ima srednju dužinu kodne riječi koja je **manja ili jednaka** od svih srednjih dužina svih ostalih trenutnih kodova koji se mogu konstruisati pod datim uslovima.

Nebinarno Hafmenovo kodiranje

Primjenjuje se u slučaju rada sa višom kodnom osnovom ($r > 2$).

Neka je data izvorna lista $S=\{S_1, S_2, ..., S_g\}$ sa g simbola koje treba kodirati sa kodnom listom $X=\{X_1, X_2, ..., X_r\}$ koja ima r simbola.

Procedura kodiranja je ista kao i u prethodnom slučaju, samo što treba u svakom koraku odrediti broj simbola koji se redukuju. Uslov za određivanje broja simbola koji se redukuju u koracima je sledeći:

$$g - (r - 1) \leq r$$

Primjer:

Neka je data izvorna lista sa 4 simbola $S = \{S_1, S_2, S_3, S_4\}$ $g = 4$, koju treba kodovati ternarnim kodom $X = \{0, 1, 2\}$ $r = 3$.

S	$P(S_i)$	X_i	$\boldsymbol{S_1}$	
S_1	0,4	0	0,4	0
S_2	0,3	1	0,3	1
S_3	0,2•	20	•0,3	2
S_4	0,1•	21		

Broj redukovanih simbola u prvom koraku mora biti:

$g - (r - 1) \leq r$ $\quad$ $4 - (3 - 1) \leq 3$ $\quad$ $2 \leq 3$ (tačno)

broj simbola koji se redukuju

Slijedi da u prvom koraku redukujemo dva simbola.

Pošto već poslije prvog koraka dobijamo da je broj simbola jednak broju simbola kodne liste, redukcija je završena i vraćamo se unazad i kodujemo simbole.

Srednja dužina kodne riječi je: $L = 1 \cdot 0,4 + 1 \cdot 0,3 + 2 \cdot 0,2 + 2 \cdot 0,1 = 1,3$ b/simb.

Primjer:

Neka je data izvorna lista sa $g = 8$ simbola, čije su vjerovatnoće poznate i kodna lista $X = \{0, 1, 2, 3\}$ sa $r = 4$ simbola.

S	P_i	X_i	S_1		S_2	
S_1	0,22	1	0,22	1	∗0,40	0
S_2	0,20	2	0,20	2	0,22	1
S_3	0,18	3	0,18	3	0,20	2
S_4	0,15	00	0,15∗	00	0,18	3
S_5	0,10	01	0,10∗	01		
S_6	0,08	02	0,08∗	02		
S_7	0,05•	030	•0,07∗	03		
S_8	0,02•	031				

1. Korak: Broj redukovanih simbola u prvom koraku mora zadovoljiti nejednakost:

$$g - (r - 1) \leq r$$

što je u našem slučaju

$8 - (4 - 1) \leq 4$ $\quad 5 \leq 4$ (nije tačno, pa ponovimo još jednom)

$5 - (4 - 1) \leq 4$ $\quad 2 \leq 4$ (tačno)

broj simbola koji se redukuju u prvom koraku

2. Korak: Broj simbola koje treba redukovati u drugom koraku je:

$7 - (4 - 1) \leq 4$ $\quad 4 \leq 4$ (tačno)

broj simbola koje treba redukovati u drugom koraku

Broj simbola u drugom koraku je jednak broju simbola kodne liste, pa je redukcija završena.

Srednja dužina kodne riječi je:

$$L = 1 \cdot 0,22 + 1 \cdot 0,20 + 1 \cdot 0,18 + 2 \cdot 0,15 + 2 \cdot 0,1 + 2 \cdot 0,08 + 3 \cdot 0,05 + 3 \cdot 0,02 = 1,44 \text{ b/sim}$$

Često se u praksi Hafmenov kod ne primjenjuje za sve simbole izvorne liste već samo za određeni broj simbola koji se relativno često pojavljuju, dok se oni simboli koji se pojavljuju ekstremno rijetko kodiraju kodom fiksne dužine.

Hafmenov postupak kodiranja se može koristiti i kod Markovljevih procesa sa memorijom.

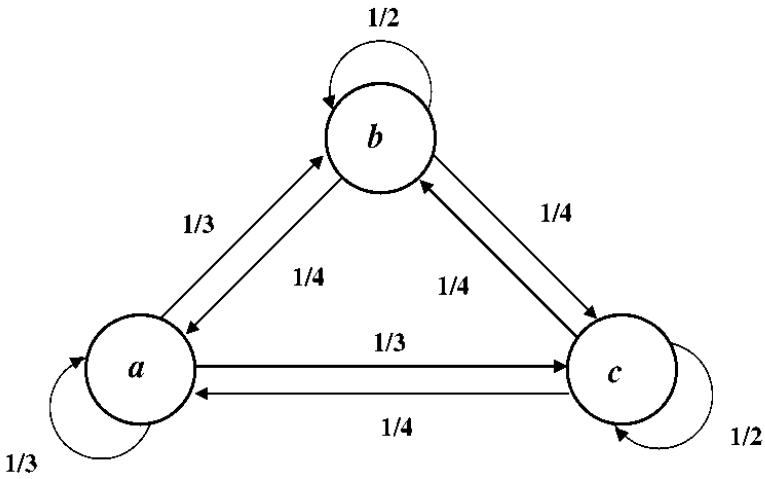
Primjer:

Posmatrajmo primjer koji je više puta korišćen:

$S = \{a, b, c\}$ i Markovljev izvor I reda

$$P(a \mid a) = \frac{1}{3} \quad P(b \mid a) = \frac{1}{3} \quad P(c \mid a) = \frac{1}{3}$$
$$P(a \mid b) = \frac{1}{4} \quad P(b \mid b) = \frac{1}{2} \quad P(c \mid b) = \frac{1}{4}$$
$$P(a \mid c) = \frac{1}{4} \quad P(b \mid c) = \frac{1}{4} \quad P(c \mid c) = \frac{1}{2}$$

Graf tranzicije je:



U stacionarnom stanju smo dobili da je:

$$P(a) = \frac{3}{11}, \; P(b) = \frac{4}{11}, \; P(c) = \frac{4}{11}$$

Ako bi izvršili Hafmenovo kodiranje simbola izvorne liste u stacionarnom stanju, oni bi bili kodirani na sljedeći način:

S	$P(S_i)$	X_i	S_1	
b	$\frac{4}{11}$	1	$\bullet\ \frac{7}{11}$	0
c	$\frac{4}{11}\bullet$	00	$\frac{4}{11}$	1
a	$\frac{3}{11}\bullet$	01		

Simbol *a* je kodiran sa **01** (2 bita)
Simbol *b* je kodiran sa **1** (1 bit)
Simbol *c* je kodiran sa **00** (2 bita)

Srednja dužina kodne riječi je: $L = 1 \cdot \frac{4}{11} + 2 \cdot \frac{4}{11} + 2 \cdot \frac{3}{11} = \frac{18}{11} = 1{,}6364$ b/simb.

Međutim, mi možemo kodirati prelaske pošto kod trenutnih kodova znamo prethodni simbol.

1. Pretpostavimo da se desilo ***a***:

S	$P(S_i)$	X_i	S_1	
$P(a\|a)$	$\frac{1}{3}$	1	$\bullet\ \frac{2}{3}$	0
$P(b\|a)$	$\frac{1}{3}\bullet$	00	$\frac{1}{3}$	1
$P(c\|a)$	$\frac{1}{3}\bullet$	01		

Prelazak iz *a* u *a* je kodiran sa **1**
Prelazak iz *a* u *b* je kodiran sa **00**
Prelazak iz *a* u *c* je kodiran sa **01**

$L_a = 1 \cdot \frac{1}{3} + 2 \cdot \frac{1}{3} + 2 \cdot \frac{1}{3} = \frac{5}{3}$ b/simb.

2. Pretpostavimo da se desilo ***b***:

$\boldsymbol{S}$	$P\,(S_i)$	X_i	$\boldsymbol{S_1}$	
$P(b\|b)$	$\frac{1}{2}$	0	$\frac{1}{2}$	0
$P(a\|b)$	$\frac{1}{4}\bullet$	10	$\bullet\frac{1}{2}$	1
$P(c\|b)$	$\frac{1}{4}\bullet$	11		

Prelazak iz *b* u *b* je kodiran sa **0**
Prelazak iz *b* u *a* je kodiran sa **10**
Prelazak iz *b* u *c* je kodiran sa **11**

$$L_b = 1\cdot\frac{1}{2} + 2\cdot\frac{1}{4} + 2\cdot\frac{1}{4} = \frac{3}{2}\ \text{b/simb.}$$

3. Pretpostavimo da se desilo ***c***:

$\boldsymbol{S}$	$P\,(S_i)$	X_i	$\boldsymbol{S_1}$	
$P(c\|c)$	$\frac{1}{2}$	0	$\frac{1}{2}$	0
$P(a\|c)$	$\frac{1}{4}\bullet$	10	$\bullet\frac{1}{2}$	1
$P(b\|c)$	$\frac{1}{4}\bullet$	11		

Prelazak iz *c* u *c* je kodiran sa **0**
Prelazak iz *c* u *a* je kodiran sa **10**
Prelazak iz *c* u *b* je kodiran sa **11**

$$L_c = 1\cdot\frac{1}{2} + 2\cdot\frac{1}{4} + 2\cdot\frac{1}{4} = \frac{3}{2}\ \text{b/simb.}$$

Ukupna srednja dužina kodne riječi se dobija:

$$L = L_a P(a) + L_b P(b) + L_c P(c) = \frac{17}{11}\ \text{b/simb.}$$

Zaključujemo da je srednja dužina kodne riječi malo kraća nego u prethodnom slučaju.

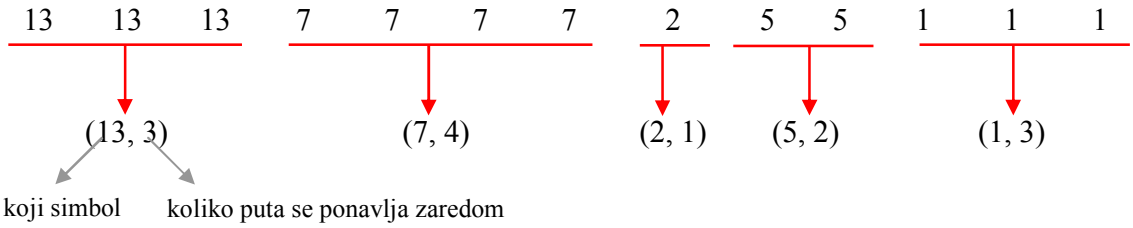
Hafmenov kod daje dobre rezultate kod stacionarnih izvora i kod poznatih vjerovatnoća pojavljivanja pojedinih simbola. Međutim, kod nestacionarnih izvora i kada nijesu poznate vjerovatnoće koristi se **adaptivni Hafmenov kod**. Ovaj kod mjeri vjerovatnoće pojavljivanja simbola i kada usvojeni Hafmenov kod prestane da daje optimalnu dužinu koda vrši se ponovno izračunavanje Hafmenovog koda.

RLE (Run Length Encoding) kod

RLE je veoma jednostavna, ali u brojnim praktičnim slučajevima, veoma korisna kodna šema. Koristi se u slučajevima kada izvor emituje simbole koji se ponavljaju veliki broj puta uzastopno.

Na primjer: Fax dokument se dijeli u tačkice, koje se zovu pikseli. Svaka tačka je jedno crno ili bijelo polje. Ako se crno polje ponavlja veliki broj puta uzastopno, potrebno je zapamtiti da je polje crno i broj piksela koji se ponavljaju. RLE kodiranje se rijetko sprovodi samostalno, već često u kombinaciji sa nekim drugim kodom, kao što je Hafmenov kod.

Primjer:



Diferencijalni kod

Diferencijalni kod se koristi kod sporo promjenljivih podataka. Kod njega se pamti prva vrijednost i zatim se umjesto orginalnih vrijednosti sledećih podataka pamti razlika između susjednih podataka. Kao i RLE kod, rijetko se koristi samostalno, već ćesto se iz diferencijalnog kodera izlaz vodi do sistema za Hafmenovo kodiranje.

Primjer:

17	18	19	19	20	20	20	19	19	18	17
17	1	1	0	1	0	0	-1	0	-1	-1

Kodiranje: $Y_i = X_i - X_{i-1}$ Dekodiranje: $X_i = Y_i + X_{i-1}$

Lempel-Zivov (LZ) kod

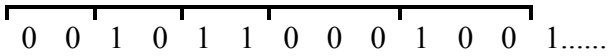
Dugo vremena je Hafmenov postupak skoro isključivo korišćen za statističko kodovanje. Međutim, on ima izvjesne suštinske nedostatke. Jedan od osnovnih nedostataka je što njegova primjena zahtijeva poznavanje vjerovatnoća pojavljivanja simbola. U mnogim situacijama nije moguće odrediti ove vjerovatnoće, zbog čega su činjeni mnogi pokušaji modifikacije Hafmenovog koda.

Lempel-Zivov (LZ) postupak kodovanja je jedna vrsta univerzalnog kodovanja. Ovaj kod je zasnovan na rječniku u kojem se pamte do sada viđeni stringovi. Osnovne ideje LZ kodovanja biće izložene na primjeru binarnog izvora, ali ih je lako proširiti na slučaj kada izvor ima više od dva simbola.

U suštini, može se reći da se u okviru LZ algoritma, ulazna sekvenca dijeli na najkraće segmente koji se još nisu pojavili u prethodno primljenom dijelu sekvence. Segmenti koji su se prethodno pojavili zapamćeni su od strane kodera, tako da koder umjesto cijelog novog segmenta šalje samo adresu prethodno zapamćenog segmenta i novi bit. Na ovaj način se definiše novi segment duži za jedan bit i taj novi segment sada dobija svoju (novu) adresu.

Primjer:

Neka je data sledeća sekvenca koju treba kodovati LZ postupkom:

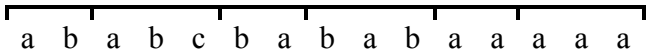


Neka su u koderu već zapamćeni bit **0** (sa adresom 1) i bit **1** (sa adresom 2)

Adrese	Stringovi	Kodirana sekvenca	
1	0		
2	1		
3	00	(1, 0)	ili (001, 0)
4	10	(2, 0)	ili (010, 0)
5	11	(2, 1)	ili (010, 1)
6	000	(3, 0)	ili (011, 0)
7	100	(4, 0)	ili (100, 0)
	...	...	...

Primjer:

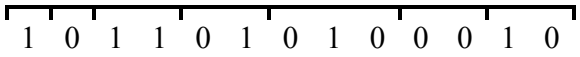
Neka je dat alfabet izvora $\{a, b, c\}$ i neka koder u rječniku ima alfabet izvora. Kodirati sekvencu LZ postupkom:



Adrese	Stringovi	Kodirana sekvenca
1	a	
2	b	
3	c	
4	ab	(1, b)
5	abc	(4, c)
6	ba	(2, a)
7	bab	(6, b)
8	aa	(1, a)
9	aaa	(8, a)

Primjer:

Kodirati LZ postupkom sekvencu:



polazeći od praznog rječnika.

Adrese		Stringovi	Kodirana sekvenca
0	000	NULL	
1	001	1	(000, 1) ili (0, 1)
2	010	0	(000, 0) ili (0, 0)
3	011	11	(001, 1) ili (1, 1)
4	100	01	(010, 1) ili (2, 1)
5	101	010	(100, 1) ili (4, 0)
6	110	00	(010, 1) ili (2, 0)
7	111	10	(001, 0) ili (1, 0)

Na osnovu prethodnih primjera možemo zaključiti da se LZ postupkom dobijaju **kodne riječi fiksne dužine**.