UNIVERZITET CRNE GORE

Elektrotehnički fakultet, Podgorica

Materijal sa drugog termina predavanja iz

EKSPERTNIH SISTEMA

NEINFORMISANE STRATEGIJE PRETRAŽIVANJA

Prof. dr Vesna Popović-Bugarin

Podgorica, 2015.

4.1 Strategije pretraživanja

Kada je izvršeno predstavljanje problema u prostoru stanja može se započeti pretraživanje putanje do ciljnog stanja. Pretraživanja koja će se u ovom poglavlju izučavati se mogu predstaviti kao pronalaženje putanje kroz stablo. Stablo je graf u kojem svaki čvor ima samo jednog roditelja. Ovo zapravo znači da, kada kažemo da ćemo izučavati algoritme za stablo pretraživanja, polazimo od pretpostavke da se u neko stanje može direktno doći iz samo jednog stanja. Izuzetak predstavlja ciljno stanje, u koje se može doći direktno iz više stanja (pojavljuje se kao čvor na više mjesta u stablu). Ipak, ovo neće narušiti koncepciju stabla, jer se iz ciljnog stanja dalje nećemo kretati (predstavljaće list stabla), i neće se pojavljivati zatvorene putanje i petlje.

Često se prostor pretraživanja ne može predstaviti kao stablo (konačne dubine), već imamo ponovljena stanja, čvorove koji imaju više roditelja, i koji se mogu pojaviti više puta u jednoj putanji, čime se stvaraju ciklusi (zatvorene putanje) i potrebno je posmatrati graf pretraživanja (ili stablo pretraživanja, beskonačne dubine, u kojem se dio nekih putanja naizmjenično ponavlja). Sve strategije pretraživanja će biti ilustrovane preko stabla pretraživanja, a na kraju će se dati uputstva šta treba biti modifikovano kada se naiđe na graf, odnosno kada se pojave ponovljena stanja na putu ka cilju.

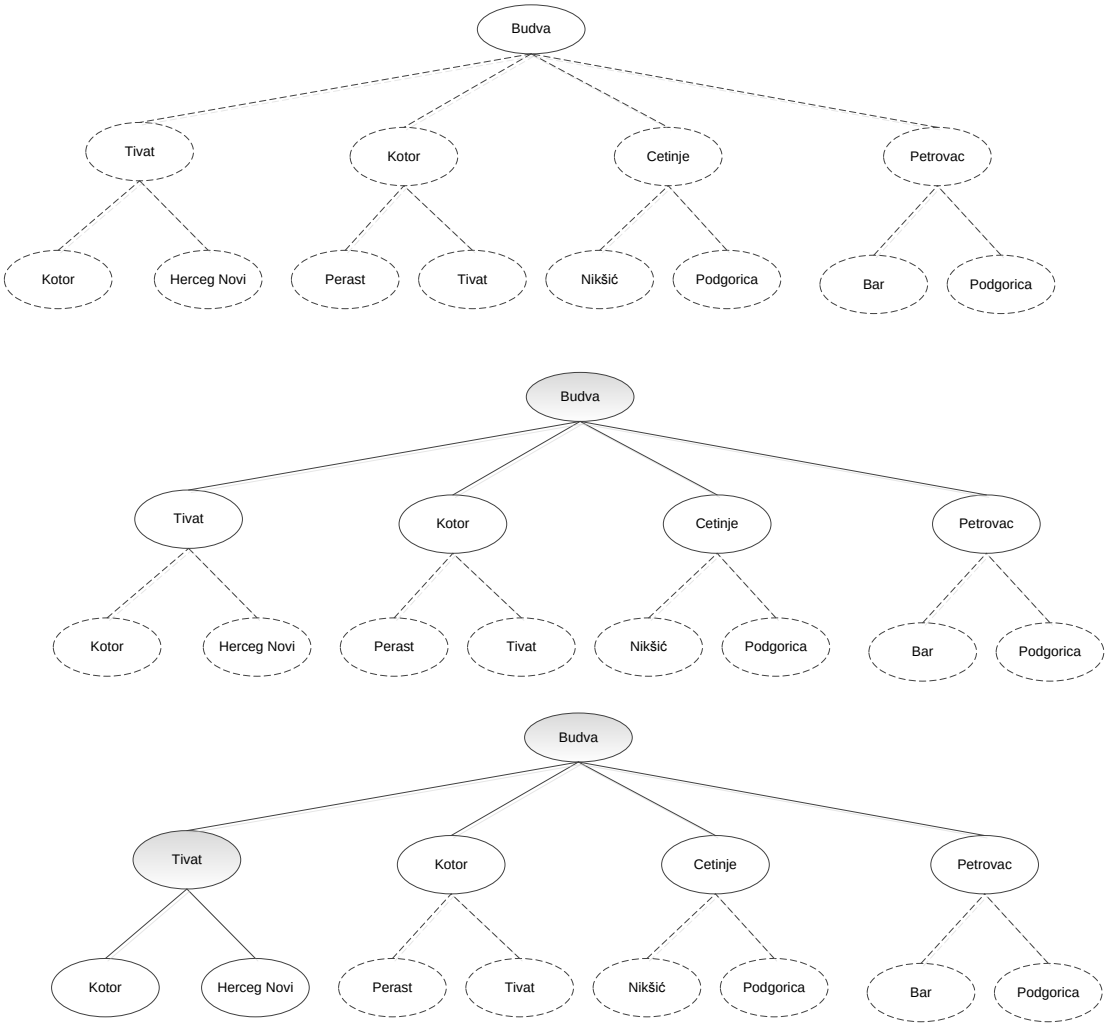
Svako stablo pretraživanja se dobija polazeći od korijena (početnog stanja) i generisanjem novih čvorova (stanja iz prostora stanja) proširivanjem trenutnog čvora upotrebom svih dozvoljenih operatora (prelaza) za to stanje. Proširivanje čvorova se vrši sve dok se dođe do ciljnog stanja ili se zaključi da dalje generisanje čvorova nije moguće, odnosno da više nema čvorova koji bi se mogli proširiti. Način generisanja novih čvorova zavisi od **strategije pretraživanja.**

U slučaju traženja putanje od Budve do Podgorice, postupak pretraživanja bi započeo iz Budve, koja bi bila korijen stabla, Slika 1. Provjeravalo bi se da li je to ciljno stanje i nakon zaključka da nije, čvor bi se proširivao generisanjem sve njegove djece - upotrebom svih mogućih prelaza na susjedne gradove. Nakon ovog proširivanja dobilo bi se stablo u kojem bi se čvor Budva preko odgovarajućih operatora proširio generisanjem čvorova Kotor, Petrovac, Tivat, Cetinje, Slika 1 – drugi red. Sada bi morali odlučiti koja od ova četiri grada (četiri mogućnosti) dalje da razmatramo. Jedna od mogućnosti bi bila da se odabere, npr. Tivat, ispita da li je on ciljno stanje izvrši njegovo proširivanje, čime se dobijaju gradovi Kotor i Herceg Novi, Slika 1 – treći red. Dalje bi se moglo vršiti ispitivanje da li je jedan od ova dva grada cilj, i njegovo proširivanje nakon utvrđivanja da nije, ili vraćanje nazad i biranje nekog od neispitanih gradova (Petrovac, Tivat, Cetinje). Ovo ispitivanje da li je trenutni čvor ciljni i njegovo proširivanje se vrši sve dok se ne dođe do cilja, ili se eventualno zaključi da su svi čvorovi ispitani i prošireni i da se ne može doći do cilja u datom prostoru stanja. Način na koji se vrši biranje čvora koji će se ispitivati i proširivati određuje se strategijom pretraživanja.

Kod gore opisanog postupka pretraživanja prostora stanja jasno se izdvaja *stablo pretraživanja* - koje u nekoj fazi pretraživanja čini dio stanja i prelaza iz prostora stanja koji je do te faze otkriven. Pod otkrivenim dijelom prostora stanja se podrazumijevaju svi generisani čvorovi, bilo da je ispitano da li su ciljno stanje ili ne. Ispitani čvorovi, koji su dalje prošireni generisanjem njihove djece, se često nazivaju

zatvorenim stanjima (osjenčene elipse, Slika 1), dok se *otvorenim stanjima* nazivaju oni čvorovi koji su generisani (neosjenčene elipse punih ivica, Slika 1), ali nije još uvijek ispitano da li su ciljni čvor i nijesu proširenи generisanjem njihove djece. Isprekidanim linijama su prikazani čvorovi koji još uvijek nijesu generisani, ali mogu biti na osnovu datog prostora stanja, Slika 1.

Čvor u stablu pretraživanja nije prosta reprezentacija stanja iz prostora stanja, već je to struktura podataka od kojih se sastoji stablo pretraživanja. Sadrži podatke o čvoru od kojeg je potekao (roditelju), o operatoru koji je primijenjen na čvor roditelj kako bi se generisao ovaj čvor, funkciji cijene putanje od korijena ili cijenu prelaza od roditeljskog čvora do njega i dubini koja je jednaka broju čvorova koji čine putanju koja ga povezuje sa korijenim čvorom – početnim stanjem. Dubina početnog stanja je nula, njegove djece 1, njihovih potomaka 2, itd.



Slika 1 Parcijalno stablo pretraživanja za nalaženje putanje od Budve do Podgorice. Prošireni čvorovi su osjenčeni, okvir čvorova koji su generisani, ali nijesu još uvijek ispitani, (otvoreni čvorovi) su dati punom linijom, dok su neotvoreni čvorovi sa okvirima prikazanim isprekidanim linijama

Jasno je da bez obzira na jednostavnost generalnog opisa pretraživanja, proširivanjem čvorova i ispitivanjem novogenerisanih čvorova na ciljno stanje, ipak

neće sve strategije pretraživanja biti jednako dobre. Potrebno je na neki način uporediti kvalitet raznih strategija pretraživanja, pa se često vrši njihovo poređenje po:

- **Potpunosti** – zagarantovanost pronalaženja rješenja ukoliko ono postoji.
- **Optimalnosti** – nalaženja optimalnog rješenja, pri čemu pod optimalnim rješenjem smatramo rješenje sa najmanjom vrijednošću funkcije cijene puta.
- **Vremenskoj kompleksnosti** – vremenu koje je potrebno da bi se pronašlo rješenje.
- **Prostornoj kompleksnosti** – memoriji koja je neophodna da bi se došlo do rješenja.

Naravno, prilikom definisanja vremenske i prostorne kompleksnosti ima se na umu da je svaka od izučavanih strategija pretraživanja namijenjena realizaciji na računaru. Dalje, kada su u pitanju vremenska i prostorna složenost, one se uvijek mjere imajući na umu složenost ispitivanog problema. Prostor stanja u problemima vještačke inteligencije predstavljen grafom često se ne može definisati prostim nabrajanjem stanja, jer će funkcije prelaza često omogućiti generisanje praktično beskonačno mnogo stanja (gradova u koje se može stići iz Budve i mogućnosti da se preko njih stigne do Podgorice), bez obzira na to što će ovaj graf (ili funkcija za njegovo generisanje) biti ulazni argument programa kojim će se realizivati pretraživanje, za utvrđivanje prostorne složenosti se ne uzima u obzir veličina strukture podataka kojom je prestavljen cijeli graf, već:

- *Faktor grananja **(b)*** - najveći broj grana koje izlaze iz nekog čvora. Ukoliko ne postoji prevelika razlika u broju grana koje izlaze iz pojedinih čvorova uzima se i prosječni faktor grananja, prosječan broj grana koje izlaze iz nekog čvora.
- *Dubina najplićeg rješenja **(d)*** – broj čvorova od početnog stanja do čvora koji predstavlja rješenje, a koji ima najmanju dubinu. Kažemo da je za poređenje algoritama bitna dubina najplićeg rješenja. To što imamo rješenja različitih dubina, ne znači da rješenje može biti dostizanje jednog od više različitih stanja, već da se bilo koje stanje, pa i ciljno, može generisati upotrebom dozvoljenih prelaza od strane većeg broja ispitivanih i proširenih stanja, te se samim tim nalazi kao čvor na više mjesta u stablu pretraživanja. U primjeru sa putanjom, ovo predstavlja činjenicu da se u Podgoricu može doći iz više gradova, pa iako u prostoru stanja nećemo imati osim jedno stanje koje će predstavljati činjenicu da se turista nalazi u Podgorici, u grafu će se generisati čvor u(Podgorici), proširivanjem bilo kojeg čvora koji predstavlja grad susjedan Podgorici, Slika 1. Ponavljanje ciljnog stanja nije problem, jer će se tu završiti generisanje čvorova za tu granu, i neće nastati petlje.
- *Maksimalna dužina putanja* u prostoru stanja ***(m)***. Ovo je zapravo jednako maksimalnoj dubini stabla pretraživanja, t.j. dubini najdubljeg lista stabla pretraživanja. Podsjetimo se da je list čvor stabla koji nema djecu.

4.2 Slijepo pretraživanje

Strategije slijepog pretraživanja su strategije neinformisanog pretraživanja. Ove strategije nemaju nikakve dodatne informacije osim onih koje su im date samom definicijom problema. One znaju samo koje stanje je početno, koje je ciljno i koji su dozvoljeni prelazi iz pojedinih stanja. Nemaju informacije o tome da li neko otkriveno

stanje ima veću vjerovatnoću da povede do ciljnog stanja, u odnosu na neko drugo. Strategije koje imaju ove informacije se nazivaju informisanim, odnosno heurističkim stategijama pretraživanja.

Najjednostavnija strategija slijepog pretraživanja bi bilo slučajno biranje otvorenog čvora koji će sledeći ispitivati i proširivati, dok postoji i veliki broj strategija koje biraju sljedeći čvor koji će se ispitivati na cilj i eventualno proširivati po nekom prethodno definisanom pravilu, pa tako imamo pretraživanje u širinu, u dubinu i njihove modifikacije.

4.2.1 Pretraživanje u širinu (breadth-first search)

Pretraživanje u širinu je jednostavna strategija pretraživanja u kojoj se kreće od korijena, izvrši se ispitivanje i proširivanje korijena, nakon čega se vrši ispitivanje i proširivanje sve njegove djece, i postupak se ponavlja dok se ne pronađe cilj ili se ispitaju svi čvorovi iz stabla pretraživanja. Dakle, prelazak na ispitivanje i proširivanje čvorova u stablu pretraživanja koji se nalaze na dubini *d+1* se vrši tek nakon što se prošire svi čvorovi na dubini *d*. Kako bi se obezbijedio da se prvi ispitaju i prošire čvorovi koji se nalaze na dubini *d*, pa da se tek onda pređe na veću dubinu, za čuvanje otvorenih čvorova se koristi princip reda FIFO (prvi čvor koji se postavi na red, se prvi i ispituje i proširuje), pa se svi novootvoreni čvorovi postavljaju na kraj reda otvorenih čvorova.

Algoritam za pretraživanje u širinu bi bio:

1. Formirati listu čvorova koja inicijalno sadrži samo startni čvor.
2. Dok se lista čvorova ne isprazni provjeriti da li je prvi element liste ciljni čvor.
   a. Ako je prvi element ciljni čvor, vratiti uspješno nađeno rješenje i prekinuti dalje pretraživanje.
   b. Ako prvi element nije ciljni čvor, ukloniti ga iz liste i dodati njegove sledbenike iz stabla pretrage (ako ih ima) na kraj liste.
3. Ako je pronađen ciljni čvor, pretraga je uspješno završena; u suprotnom pretraga je neuspješna.

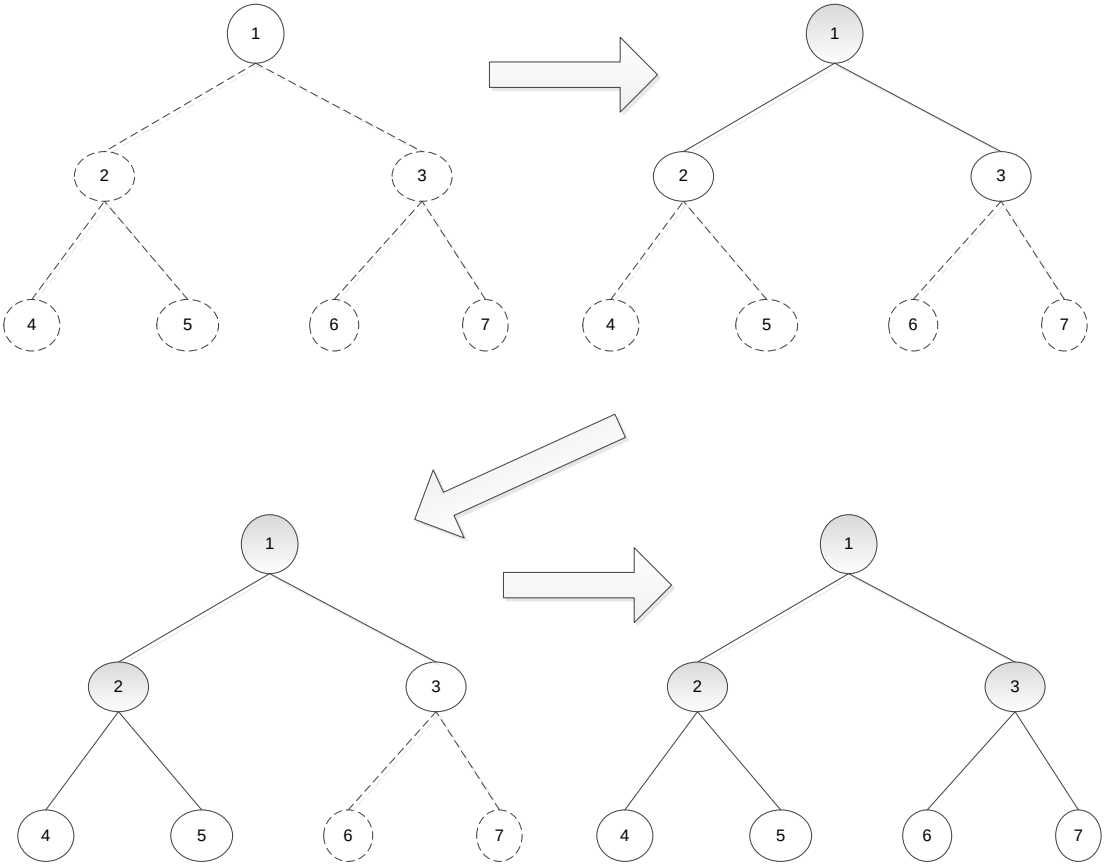
Slika 2 predstavlja ilustraciju pretraživanja u dubinu. Broj kojim su obilježeni čvorovi predstavlja redosljed ispitivanja.

Ukoliko bi se ovaj algoritam pretraživanja analizirao po četiri ranije uvedena kriterijuma, zaključili bi da je:

- **Potpun** – Uvijek će naći rješenje ukoliko ono postoji i najpliće rješenje se nalazi na nekoj konačnoj dubini, jer vrši sistematsku pretragu po svim čvorovina na dubini *d* prije nego što pređe na veću dubinu.
- Ovaj način pretrage će nam dati rješenje koje se nalazi na najmanjoj dubini, što ga ne čini automatski optimalnim rješenjem. Ovaj algoritam **daje optimalno rješenje ukoliko je funkcija cijene putanje ne može opadati sa dubinom stabla pretraživanja.** Odnosno, ukoliko je funkcija cijene putanje

na dubini d+1 uvijek veća od cijene putanje na dubini d. Ovakav bi slučaj imali kada bi svaka akcija imala isti cijenu – Hanojske kule.

- Kada je u pitanju vremenska i prostorna složenost, može se izvršiti analiza u najgorem slučaju (engl. worst case analysis), uz pretpostavku da svaki čvor ima faktor grananja b. Ukoliko se najpliće rješenje nalazi na dubini d i to kao krajnji desni čvor, moramo proširiti $b+b^2+b^3+...b^d+b^{d+1}-b$ čvorova. Za dubinu 0 proširujemo b čvorova, za dubinu 1 svaki od b čvorova ima po b djece čime dobijamo b^2 čvorova na dubini 2, i tako redom. Na dubini d vršimo proširivanje b^d-1 čvorova, i tek nam je b^d-ti čvor rješenje, pa njegovo proširivanje ne vršimo, otud $-b$ na kraju. Kažemo da je vremenska i prostorna složenost za algoritam pretraživanja u širinu $O(b^{d+1})$, (veliko o O() predstavlja red veličine računske složenosti). Dakle, za algoritam pretraživanja po širini, računska složenost eskponencijalno raste sa dubinom najplićeg rješenja. Bez obzira na veliki broj operacija proširivanja čvorova, **glavni nedostatak ovog algoritma je zapravo prostorna složenost** jer bi za faktor grananja $b=3$, dubinu najplićeg rješenja $d=9$ i memoriju potrebnu za smještanje jednog čvora od 100byte (što nije mnogo s obzirom na to šta je sve potrebno da se smjesti u srtukturu koja predstavlja čvor stabla pretraživanja) bilo nam potrebno 5.9GB radne memorije za smještanje generisanih čvorova!!!



Slika 3 Primjer stabla pretrage za metod pretraživanja u širinu.

4.2.2. Pretraživanje sa uniformnom cijenom

Ovo pretraživanje predstavlja jednostavnu modifikaciju pretraživanja po širini, a u cilju postizanja optimalnosti za bilo koju funkciju cijene putanje. Modifikacija se sastoji u tome da se sljedeći čvor koji će biti ispitivan i proširivan bira među otvorenim čvorovima u zavisnosti od cijene putanje, a ne od dubine. Naime, kod ovog pretraživanja se u nekom trenutku *proširuje onaj čvor koji ima najmanju cijenu putanje od početnog*, bez obzira na dubinu na kojoj se nalazi. Ovakav način pretraživanja je i potpun ukoliko je cijena svih prelaza veća od nule. Takođe je i optimalan u ovom slučaju jer se pretraživanje vrši tako da uvijek proširuje onaj otvoreni čvor koji će nam dati putanju sa najmanjom funkcijom cijene.

Mana ovog tipa pretraživanja je mogućnost dobijanja još veće prostorne i vremenske složenosti, nego kod pretraživanja u širinu. Prostorna i vremenska složenost, u najgorem slučaju mu se mogu predstaviti kao $O(b^{1+\lfloor C/\varepsilon \rfloor})$ uz pretpostavku da je ukupna cijena C i da je svaki prelaz ima cijenu najmanje ε, dok je $\lfloor C/\varepsilon \rfloor$ oznaka da se uzima cijeli dio količnika i procjenjuje dubinu optimalnog rješenja. Uniformno pretraživanje često daje veće stablo u odnosu na pretraživanje po širini jer daje prioritet putanjama sa manjim koracima kojih može biti mnogo, a ne moraju voditi do rješenja, za razliku od putanja koje se sastoje od većih koraka koji brže mogu povesti do rješenja. Manji koraci su zapravo prelazi koji imaju manju funkciju cijene.

4.2.3 Pretraživanje u dubinu (depth first search)

U ovoj strategiji pretraživanja u svakom trenutku se proširuje otvoreni čvor koji se nalazi na najvećoj dubini. Pretraživanja kreće od korijena i vrši se ispitivanje i proširivanje korijena, nakon čega se vrši ispitivanje i proširivanje njegovog krajnjeg lijevog djeteta i postupak se nastavlja u dubinu sve dok se ne pronađe cilj ili se dođe do lista (čvora koji nema djecu). U slučaju da se ne može vršiti dalje proširivanje trenutno ispitivanog čvora, vraća se na plići zatvoreni čvor koji još uvijek ima neispitane otvorene djece i vrši se njihovo proširivanje i ispitivanje u dubinu po istoj proceduri. Svi čvorovi koji pripradaju putanji koja se prethodno ispitivala, a nalaze se nakon ovog čvora na koji se vraćamo, se mogu odbaciti (izbaciti iz memorije). Može se vršiti odbacivanje ovih čvorova jer se pokazalo da putanja koja njih sadrži sigurno ne vodi do cilja. Kako bi se obezbijedilo da se prvi ispitaju i prošire čvorovi koji se nalaze na najvećoj dubini i da se tek ako se oni ne mogu dalje proširivati (stiglo se do lista) vrati na ispitivanje i proširivanje plićih otvorenih čvorova, za čuvanje otvorenih čvorova se koristi princip LIFO (zadnji čvor koji se postavi u listu se prvi i ispituje i proširuje), pa se svi novootvoreni čvorovi postavljaju na početak liste. Ovakva lista se često naziva stekom.

Algoritam za pretraživanje u dubinu bi bio:

1. Formirati listu čvorova koja inicijalno sadrži samo startni čvor.
2. Dok se lista čvorova ne isprazni provjeriti da li je prvi element liste ciljni čvor.
   a. Ako je prvi element ciljni čvor, vratiti uspješno nađeno rješenje i prekinuti dalje pretraživanje.

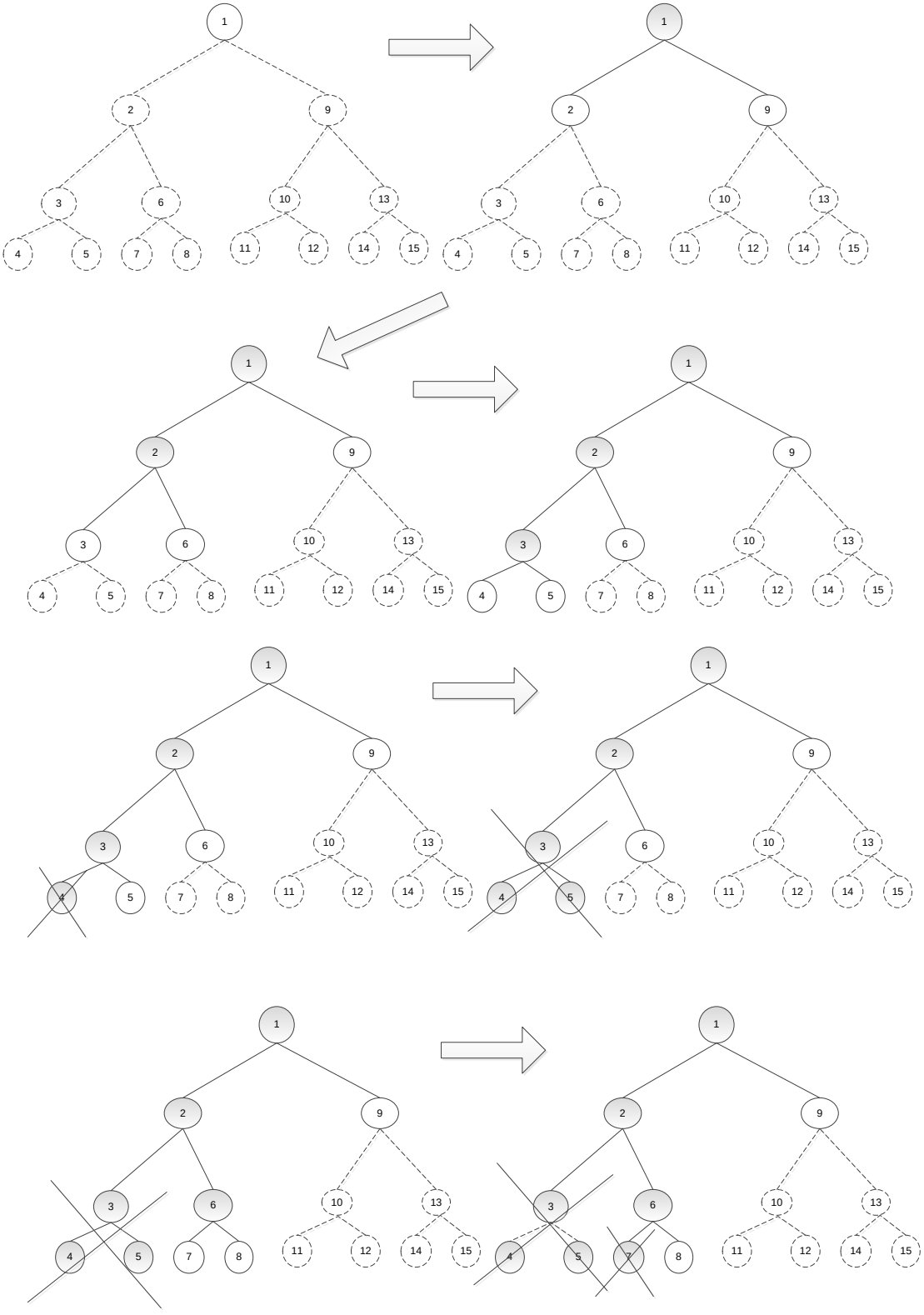
b. Ako prvi element nije ciljni čvor, ukloniti ga iz liste i dodati njegove sledbenike iz stabla pretrage (ako ih ima) na početak liste.

3. Ako je pronađen ciljni čvor, pretraga je uspješno završena; u suprotnom pretraga je neuspješna.

Slika 4 predstavlja ilustraciju pretraživanja po dubini. Brojevi kojima su obilježeni čvorovi predstavljaju redosljed njihovog ispitivanja.

Ukoliko bi se ovaj algoritam pretraživanja analizirao po četiri ranije uvedena kriterijuma, zaključili bi da:

- Ovaj način pretraživanja **obezbijeđuje potpunost** ukoliko je razmatrano stablo konačne dubine.
- **Ne obezbjeđuje optimalnost**. Može se desiti da se ovim algoritmom proširuje neko od lijevih podstabala do veoma velike dubine i da se na toj dubini nađe rješenje, dok rješenje na mnogo manjoj dubini i sa manjom cijenom postoji u nekom od stabala koje bi se dobilo proširivanjem nekog od otkrivenih i neispitanih čvorova na manjoj dubini.
- Osnovna prednost ove strategije pretraživanja jeste njena **mala memorijska zahtijevnost**. U svakom trenutku je potrebno memorisati samo putanju do čvora koji se trenutno proširuje i otvorene a neispitane čvorove. Uz istu pretpostavku kao kod pretraživanja u dubinu, dakle da svaki čvor ima faktor grananja *b* i uz pretpostavku da je maksimalna dubina stabla *m*, najveći broj čvorova koji se u nekom trenutku mora pamtiti je *(mb+1)* čvorova, odnosno za isti slučaj kao ranije *m=9, b=3* i memorija za jedan čvor 100byte, potrebno je 2.7KB radne memorije. Prilikom računanja prostorne složenosti smatra se da se svi prošireni čvorovi koji nemaju otvorenih neispitanih sljedbenika mogu maći iz memorije, Slika 5.
- Vremenska složenost u najgoram slučaju je $O(b^m)$ gdje je *m* najveća dubina nekog čvora. Ukoliko se ovo uporedi sa $O(b^{d+1})$ kod pretraživanja u širinu, i ima na umu da je *d* dubina najplićeg reješenja i da je često $d<<m$, vidi se da je vremenska složenost pretraživanja u dubinu veća. Ipak, ova strategija pretraživanja bi se trebala koristiti kada znamo da se rješenje nalazi na velikoj dubini, ili kada ima više ciljnih čvorova - velika je vjerovatnoća da ćemo pretraživanjem u dubinu naići na neko od rješenja.



Slika 5 Primjer stabla pretrage za metod pretraživanja u dubinu

4.2.4 Ograničeno pretraživanje u dubinu

Ova strategija pretraživanja **rješava problem nekompletnosti pretraživanja u dubinu kada postoji stablo neograničene dubine**. Modifikacija koja do ovoga dovodi se svodi na ograničenje dubine *l* čvorova koji se mogu proširivati. Ukoliko se

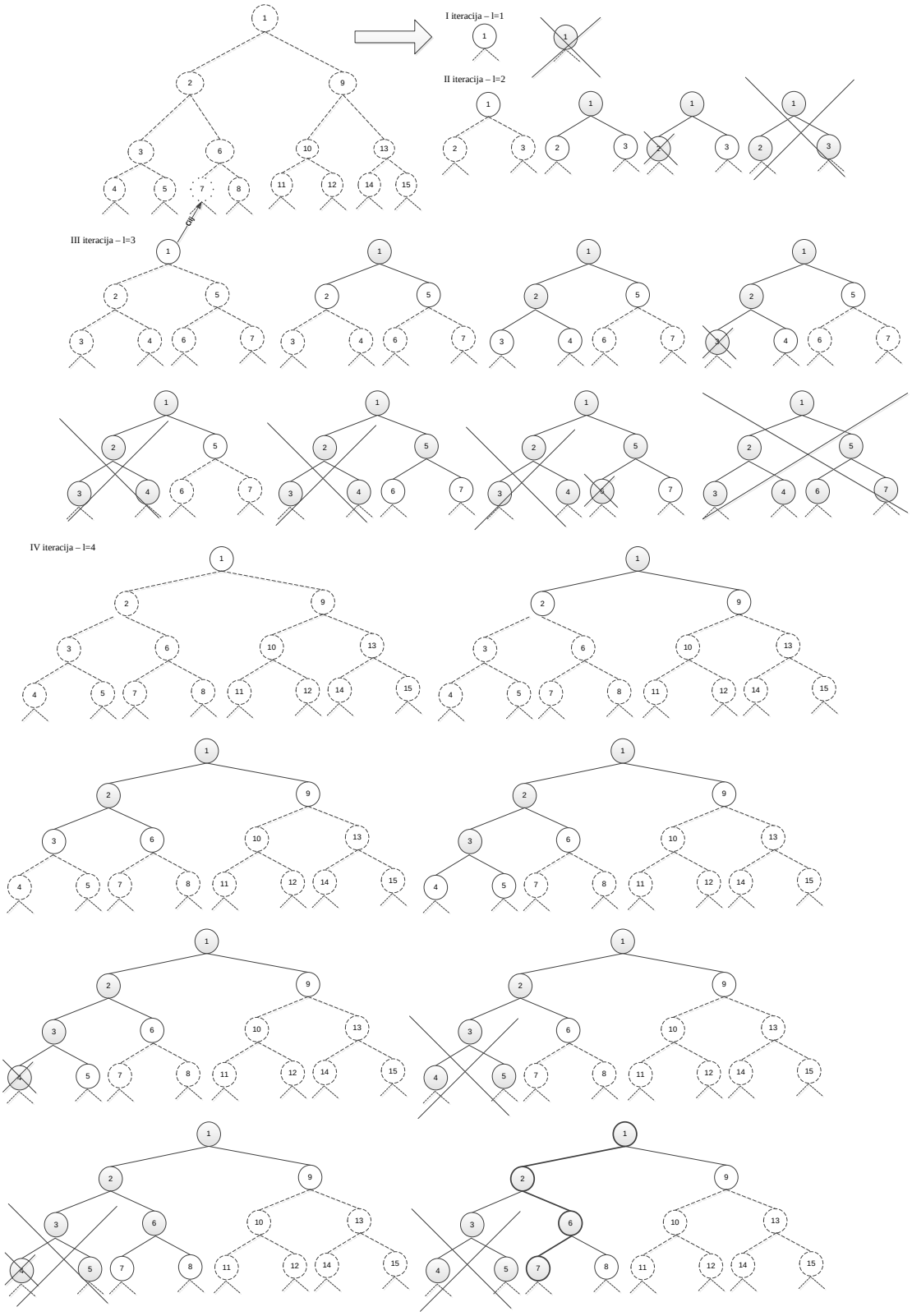
dostigne čvor na dubini *l* ponaša se isto kao da ovaj čvor nema sljedbenike, i prelazi se na plići zatvoreni čvor koji ima još uvijek neispitane djece. Glavni nedostatak je u činjenici da se najčešće ne posjeduje informacija na kojoj dubini se nalazi rješenje, pa će ovaj algoritam ipak biti nepotpun ukoliko je dubina najplićeg rješenja *d* veća od *l*. Ukoliko je $d<l$ ne garantuje se optimalnost, iz istog razloga kao kod pretraživanja u dubinu, bez ograničenja dubine. Vremenska složenost ovog algoritma je $O(b^l)$, dok je prostorna složenost $O(bl)$.

4.2.5 Iterativno pretraživanje u dubinu

Iterativno pretraživanje u dubinu omogućava dinamičko određivanje optimalne dubine, odnosno dubine najplićeg rješenja. Sastoji se u iterativnom povećavanju dubine do koje se može primijeniti ograničeno pretraživanje u dubinu. Počinje se sa dubinom nula (ispituje se samo korijeni čvor), pa se ova dubina u svakoj iteraciji povećava za 1, dakle sljedeće dubine su 2, 3, itd. Pretraživanje se prekida kada se pronađe cilj ili nema više čvorova za ispitivanje. Nakon svake iteracije odbacuju se iz memorije svi čvorovi koji su generisani u toj iteraciji i pretraga se počinje iznova.

Ilustracija iterativnog pretraživanja u dubinu je data kroz primjer, Slika 6. Prikazan je postupak za slučaj kada se rješenje nalazi na dubini 4, u čvoru koji je na početku prikazan tačkastim linijama. Putanja koja čini rješenje je podebljana. Stablo ima veću dubinu od 4.

Iterativnim pretraživanjem u dubinu **se prevazilazi problem pretraživanja u dubinu, kada se mogao dobiti nepotpun algoritam pretraživanja u slučaju postojanja beskonačne dubine**. Ovaj algoritam povećava ograničenje za dubinu do koje se vrši pretraživanje, tek nakon što se ispitaju svi čvorovi na dubini koja odgovara trenutnom limitu. Šta više, **ovaj algoritam je optimalan ukoliko je funkcija cijene puta neopadajuća funkcija sa dubinom.** Ovo su zapravo prednosti koje je imalo pretraživanje u širinu, a koje su navedenom modifikacijom prenijete na iterativno pretraživanje u dubinu. Mana pretraživanja u širinu je bila velika prostorna kompleksnost, ali kod iterativnog pretraživanja u dubinu, vrši se za svaku dubinu pretraživanje u dubini, pa je i dalje **prostorna složenost linearna i jednaka *O(bd)***.



Slika 6 Iterativno pretraživanje u dubinu. Rješenje se nalazi na dubini 4, u čvoru koji je na početku prikazan tačkastim linijama. Putanja koja čini rješenje je podebljana. Stablo ima veću dubinu od 4.

Postavlja se pitanje da li je ovom modifikacijom dobijena prevelika vremenska složenost. Vremenska složenost se jednostavno može izračunati, ukoliko imamo na

umu da je za recimo dubinu *d* korijeni čvor generisan *d* puta, njegova djeca jedan put manje, itd, sve do čvora na dubini *d* koji je generisan jedan put, pa imamo

$$N(IPD)=d + b(d-1) + b^2(d-2) + ...+ b^d$$

što daje vremensku složenost $O(b^d)$, koja je manja od vremenske složenosti algoritma za pretraživanja u širinu $O(b^{d+1})$. Ovo se može objasniti činjenicom da se čvorovi na nižoj dubini generišu više puta, a da je uobičajeno kod stabala sa konačnim faktorom grananja da mnogo više čvorova ima na višim dubinama, koji se generišu manji broj puta. Pored toga, kod iterativnog pretraživanja u dubinu, ukoliko se najpliće rješenje nalazi na dubini *d*, to će biti limit u kojem se nalazi rješenje, dok se kod pretraživanja u širinu često desi da se neki od čvorova na dubini *d* prošire i generišu čvorovi na dubini *d+1* prije nego se pronađe najpliće rješenje na širini *d*.

Smatra se da je iterativno pretraživanje u dubinu poželjno koristiti kada se razmatra problem sa velikim prostorom stanja, a dubina najplićeg rješenja nije unaprijed poznata.

4.2.5 Bidirekciono pretraživanje

Ideja bidirekcionog pretraživanja je da se istovremeno vrši pretraživanje od korijena (početnog stanja) ka cilju i od cilja ka korijenu sve dok se ova dva pretraživanja ne susretnu. Pretraživanje se prekida kada se utvrdi da čvor koji se trenutno ispituje postoji kao posjećeni čvor drugog pretraživanja. U tom slučaju, rješenje je pronađeno. Može se odabrati bilo koja od pometnutih strategija pretraživanja, ili se mogu kombinovati različite strategije pretraživanja za pretraživanje unaprijed i unazad.

Ukoliko se za oba pretraživanja odabere pretraživanje u širinu, i najplići cilj se nalazi na dubini *d=8* u najgorem slučaju se moraju ispitati u oba smjera svi čvorovi na dubini 4, osim posljednjeg čvora, pa će vremenska složenost biti $b^{d/2}+ b^{d/2}$, odnosno $O(b^{d/2})$ što je mnogo manje od vremenske složenosti koja bi se dobila kada bi se radilo jednosmjerno pretraživanje u širinu $O(b^{d+1})$, ili u dubinu $O(b^d)$. Što se tiče prostorne složenosti, mora se čuvati cijelo stablo pretraživanja u makar jednom smjeru, pa je prostorna složenost $O(b^{d/2})$, dakle veća od složenosti pretraživanja u dubinu, ali manja od složenosti pretraživanja u širinu. Algoritam je kompletan i optimalan ako su pretraživanja u oba smjera u širinu i cijena prelaza konstantna, druge kombinacije pretraživanja ne garantuju kompletnost i optimalnost.

Ovaj vid pretraživanja je popularan zbog male vremenske složenosti, ali ima niz nedostataka. Prvi nedostatak se javlja u slučaju kada imamo više od jednog ciljnog stanja, pa je teško definisati stanje od kojeg se treba krenuti. Još je veći problem kada nam cilj nije definisan nekim stanjem, već zadovoljenem uslova (šah). Dalje, ovakav način pretraživanja se može primijeniti samo kada imamo inverzne operatore, t.j. moguć je prelaz sa roditeljskog na čvor djeteta, i obratno.

Za efikasnu realizaciju ovog algoritma, uz gore navedene uslove, potrebno je i naći efikasan način poređenja novotvorenih stanja u jednom smjeru pretraživanja, sa svim posjećenim stanjima drugog smjera pretraživanja.

4.2.6. Izbjegavanje ponovljenih stanja

Pri definisanju ranije navedenih algoritama za slijepo pretraživanje, nije se vodilo računa o tome da se u neka stanja (čvorovi) može stići direktno iz više čvorova, i da

će u tom slučaju biti potrebno vršiti ispitivanje i proširivanje već ispitanog čvora, što je nista drugo do gubljenje vremena. Ovakva situacija se može pojaviti u slučajevima kada prostor stanja nije stablo, već graf sa ciklusima, što je slučaj u svim problemima u kojima su prelazi reverzibilni. Jedan takav primjer je određivanje optimalne putanje, gdje se u isti grad može stići iz više pravaca. Slična situacija je i kod slagalice. Pored povećanja vremenske kompleksnosti ovakvi prostori stanja mogu rezultovati u beskonačnom stablu pretraživanja.

S obzirom da će ponovljena stanja izazvati pojavu stabala beskonačne dubine, ona mogu izazvati i nerješivost problema koji su inače rješivi. Rješenje je detekcija i eliminacija ovakvih stanja. Detekcija se može vršiti tako što se svaki čvor koji se trenutno ispituje uporedi sa već posjećenim i proširenim čvorovima. Ukoliko se utvrdi da je taj čvor već proširivan, znači da smo pronašli dvije različite putanje do istog stanja i trebamo odbaciti jednu od njih. U ovom slučaju postoje dvije strategije; prva strategija bi bila da se trenutno ispitivani čvor odbaci, a druga da se vidi koji čvor se nalazi u kraćoj putanji i da se on zadrži.

Da bi se moglo vršiti adekvatno poređenje sa čvorovima koji su već prošireni, moraju se svi prošireni čvorovi pamtiti, pa prostorna kompleksnot algoritama u dubinu neće više biti linearna. Ukoliko se želi dobiti optimalnost za ovu vrstu pretraživanja (koja se još naziva pretraživanje grafova, zbog uzimanja u obzir ponovljenih stanja koja su posljedica postojanja ciklusa) mora se imati na umu sljedeće:

- Kod algoritama pretraživanja sa uniformnom cijenom i pretraživanja u širinu sa konstantnom cijenom prelaza, obije strategije dovode do optimalnog rješenja.
- Kod svih pretraživanja u dubinu potrebno je ispitati koja putanja ima manju cijenu i zadržati čvor koji vodi do nje a odbaciti drugi čvor.

Minimum zahtijevanog znanja sa drugog termina

1. Zadatak sa problemom koji je potrebno predstaviti u prostoru stanja i naći rješenje primjenom neke od strategija pretraživanja. Može se tražiti i primjena više strategija i uporedna analiza na konkretnom problemu.
2. Dati preporuke za koji tip problema je dobro koristiti pojedine analizirane strategije pretraživanja

LITERATURA

[1] Russell S., Norvig P.: *Artificial Intelligence: A Modern Approach*, Prentice Hall, NJ, 1995.

[2] http://ri4es.etf.rs/, posljednji put pristupano, 08. 02. 2010. godine.

[3] *http://www.zemris.fer.hr/predmeti/tes/nastava.html* - zadnji put pristupano, 10. 02. 2010. godine.