Ekspertski sistemi

Lekcija 3: Strategije pretraživanja

Strategije pretraživanja

Strategije pretraživanja se mogu podeliti na:

- neka putanja
  - po dubini,
  - po širini,
  - generiši i testiraj,
  - planinarenje,
  - prvo najbolji,
  - Grananje i ograničavanje
- optimalna putanja
  - A*
  - AO*
  - Zadovoljavanje ograničenja
  - Metoda sukcesivnih aproksimacija
  - Britanski muzej
  - Skoči i ograniči
- Igre
  - Minimaks
  - Alfa-beta podešavanje
  - Progresivno proširivanje
  - Heurističko podešavanje
  - Heurističko nastavljanje

Po širini (Breadth-first)

Ciljni čvor se traži između svih čvorova na datom nivou pre nego što se krene sa njihovim sledbenicima

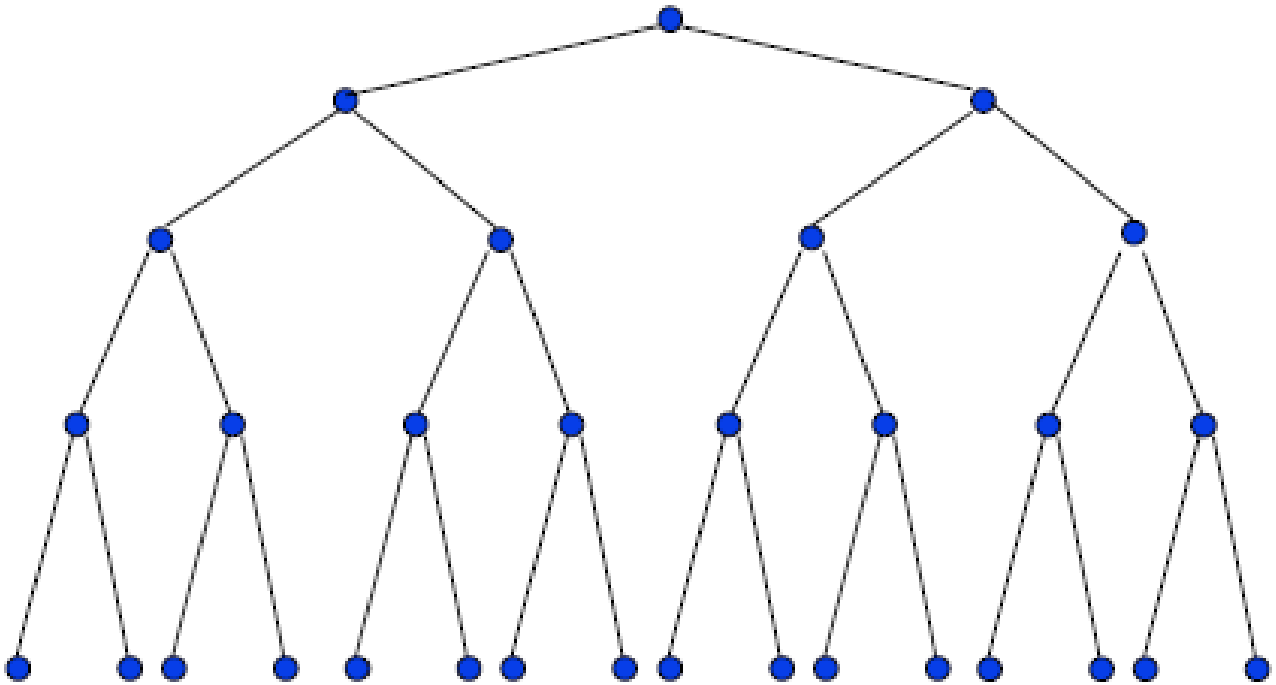
1. Formirati listu čvorova koja inicijalno sadrži samo startni čvor.
2. Dok se lista čvorova ne isprazni ili se ne dođe do ciljnog čvora, proveriti da li je prvi element liste ciljni čvor

2.1. Ako je prvi element ciljni čvor, ne raditi ništa.

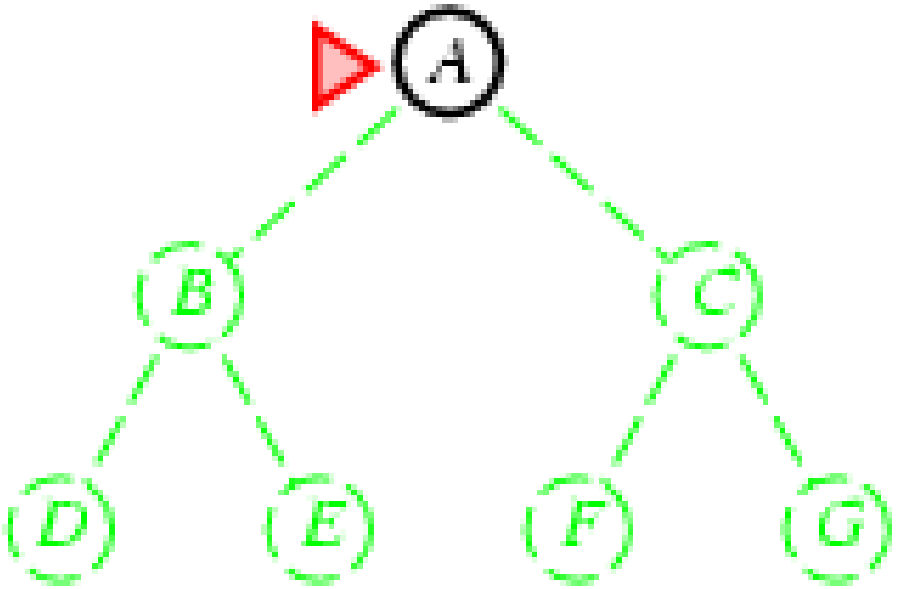
2.2. Ako prvi element nije ciljni čvor, ukloniti prvi element iz liste i dodati njegove sledbenike iz stabla pretrage (ako ih ima) *na kraj* liste.

3. Ako je pronađen ciljni čvor, pretraga je uspešno završena; u suprotnom pretraga je neuspešna.

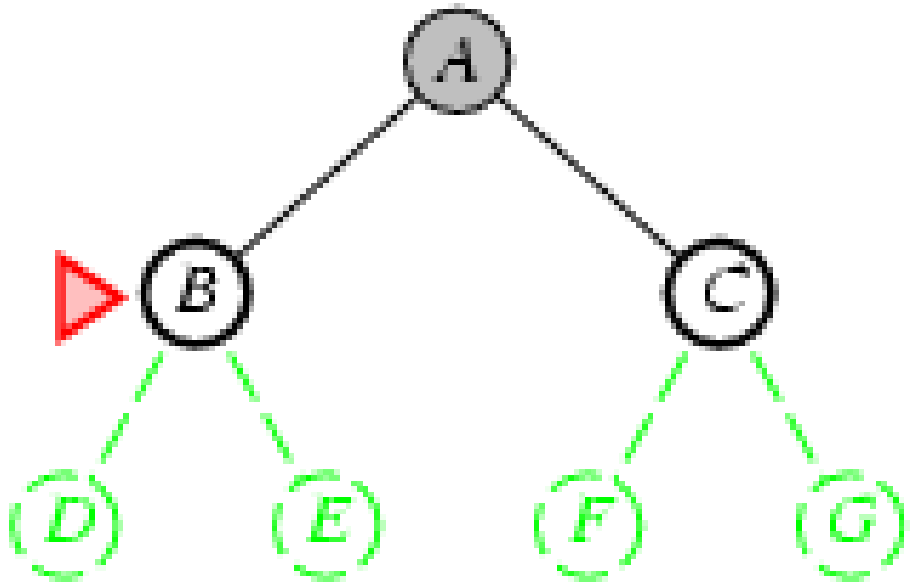
Po širini (Breadth-first)



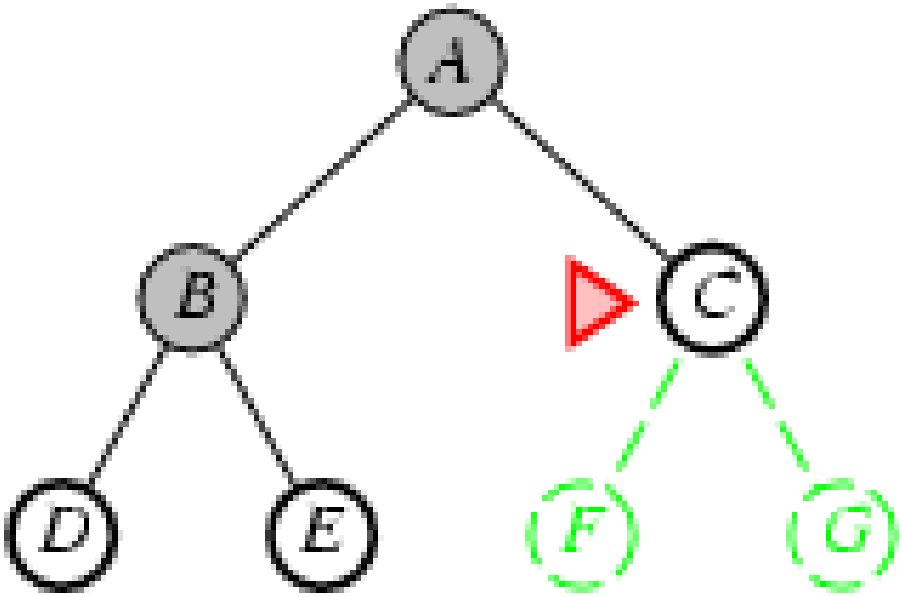
Po širini



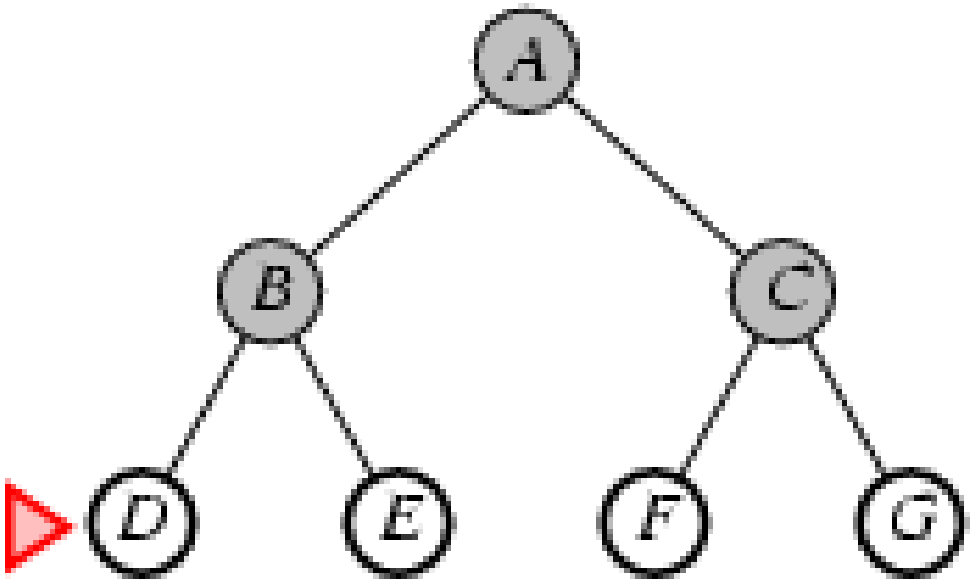
Po širini



Po širini



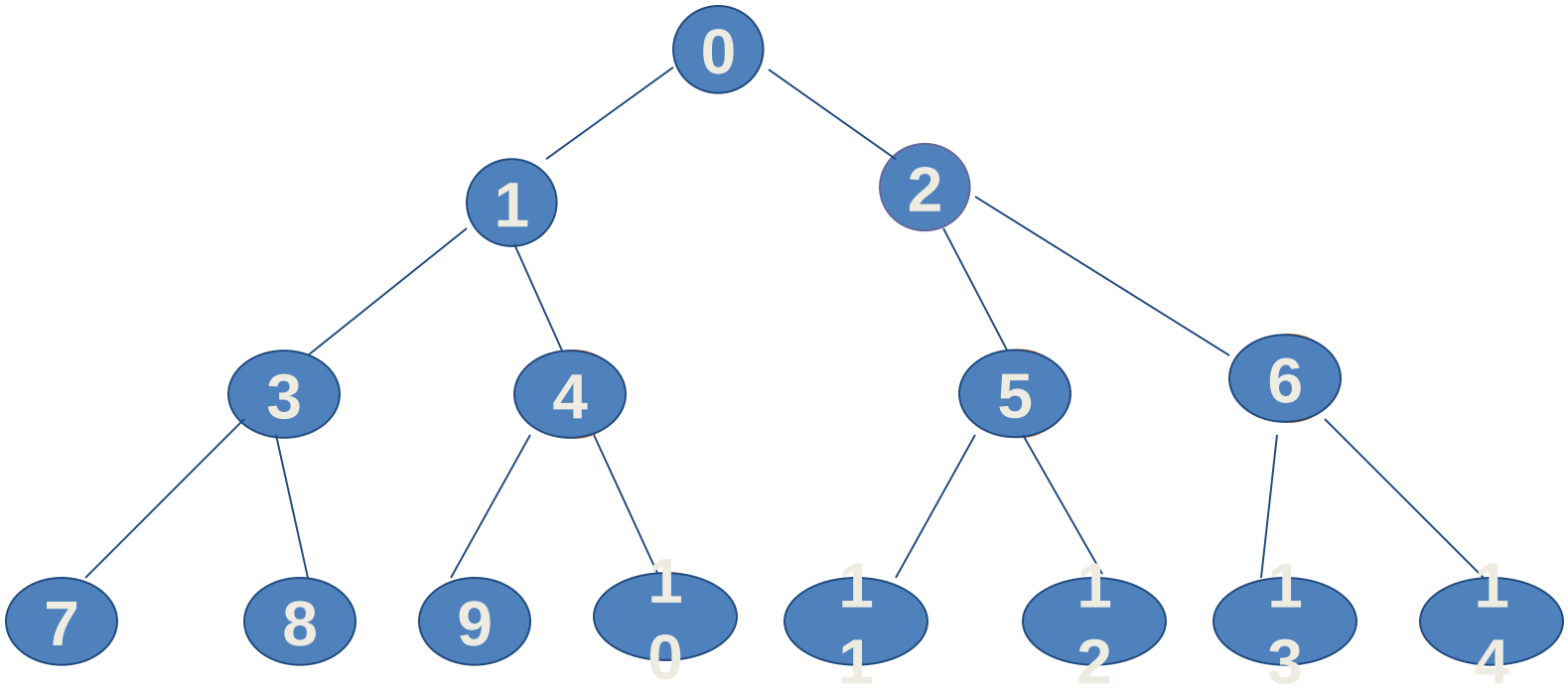
Po širini



Osobine

- Kompletno? Da (ako *b* je b konacno)
- Vreme? $1+b+b^2+b^3+\ldots+b^d + b(b^d-1) = O(b^{d+1})$
- Prostor? $O(b^{d+1})$ (svaki čvor u memoriji)
- Optimalno rešenje? Da
- Prostor je veći problem nego vreme

Po širini (Breadth-first)



Brojevi prikazuju redosled posete pri BF pretraživanju

Po dubini (Depth-first)

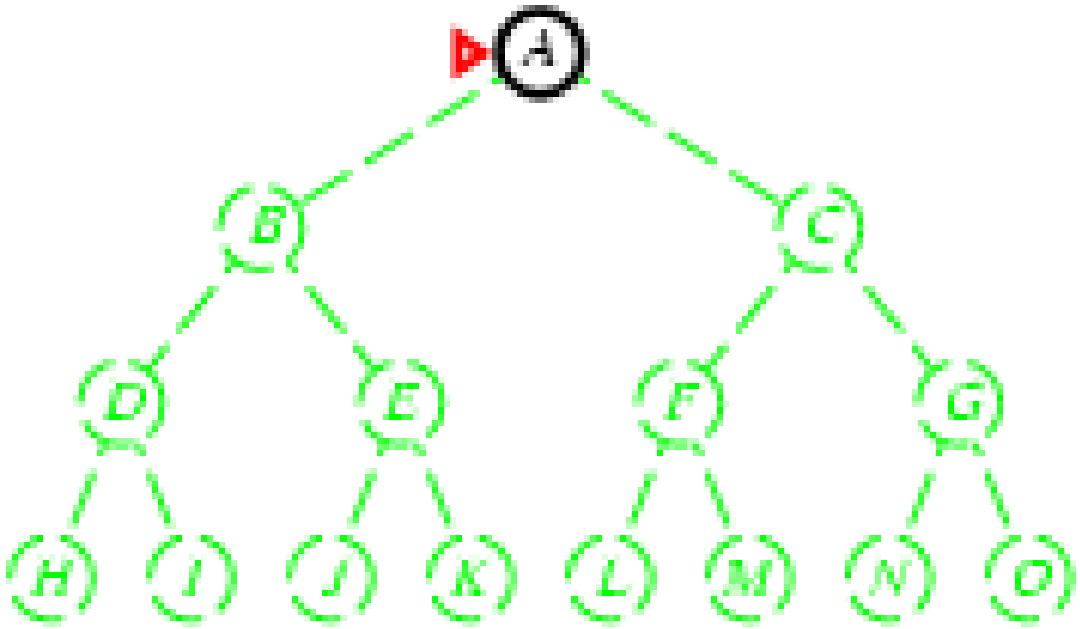
1. Formirati listu čvorova koja inicijalno sadrži samo startni čvor.
2. Dok se lista čvorova ne isprazni ili se ne dođe do ciljnog čvora, proveriti da li je prvi element liste ciljni čvor

2.1. Ako je prvi element ciljni čvor, ne raditi ništa.

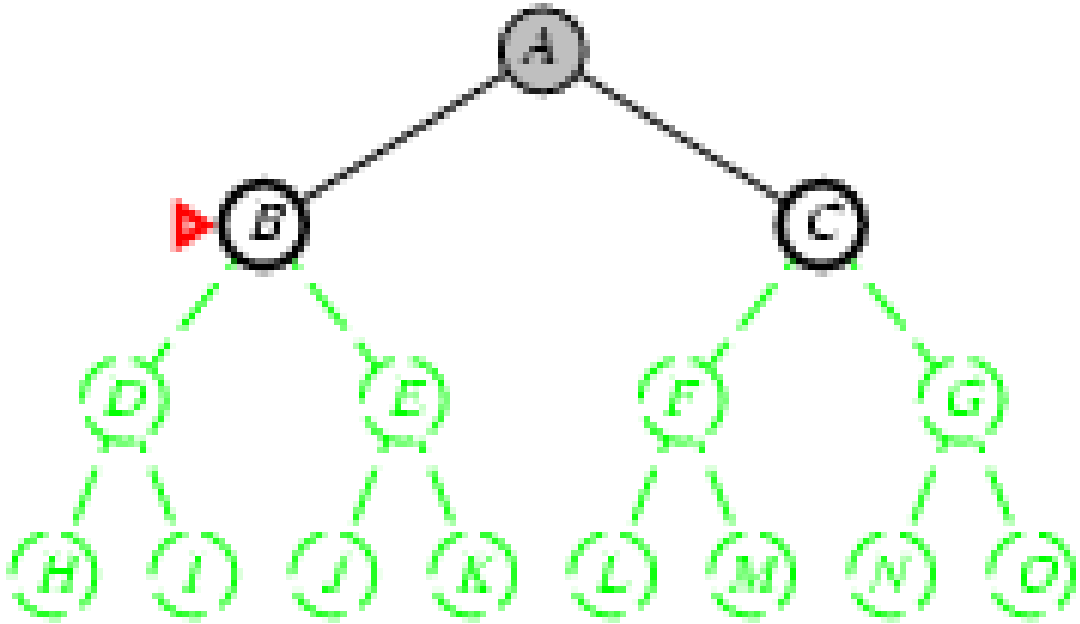
2.2. Ako prvi element nije ciljni čvor, ukloniti prvi element iz liste i dodati njegove sledbenike iz stabla pretrage (ako ih ima) *na početak* liste.

3. Ako je pronađen ciljni čvor, pretraga je uspešno završena; u suprotnom pretaga je neuspešna.

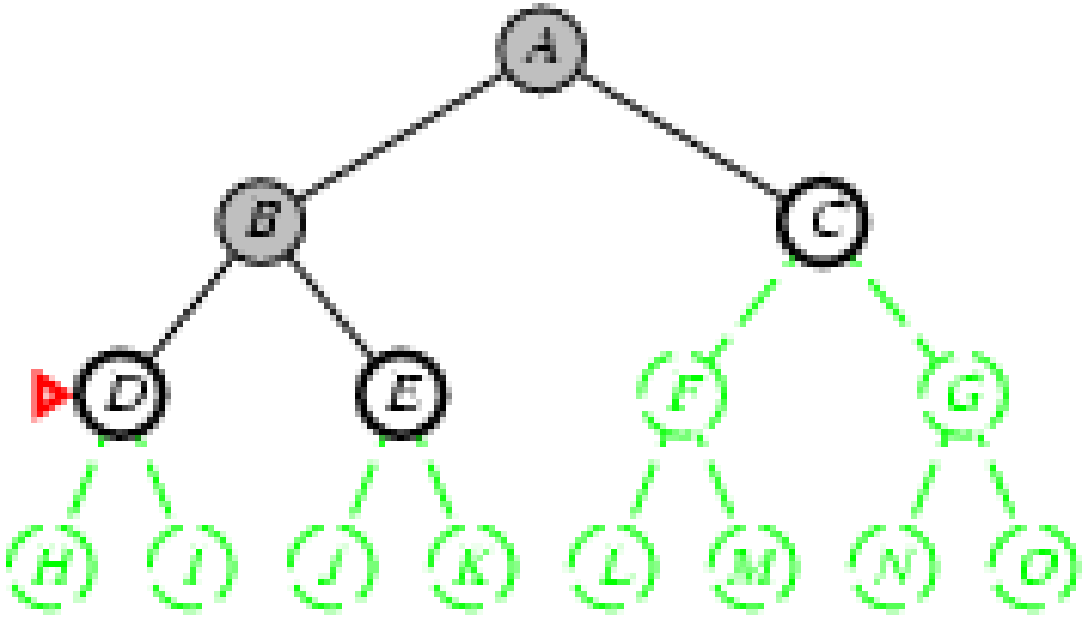
Po dubini



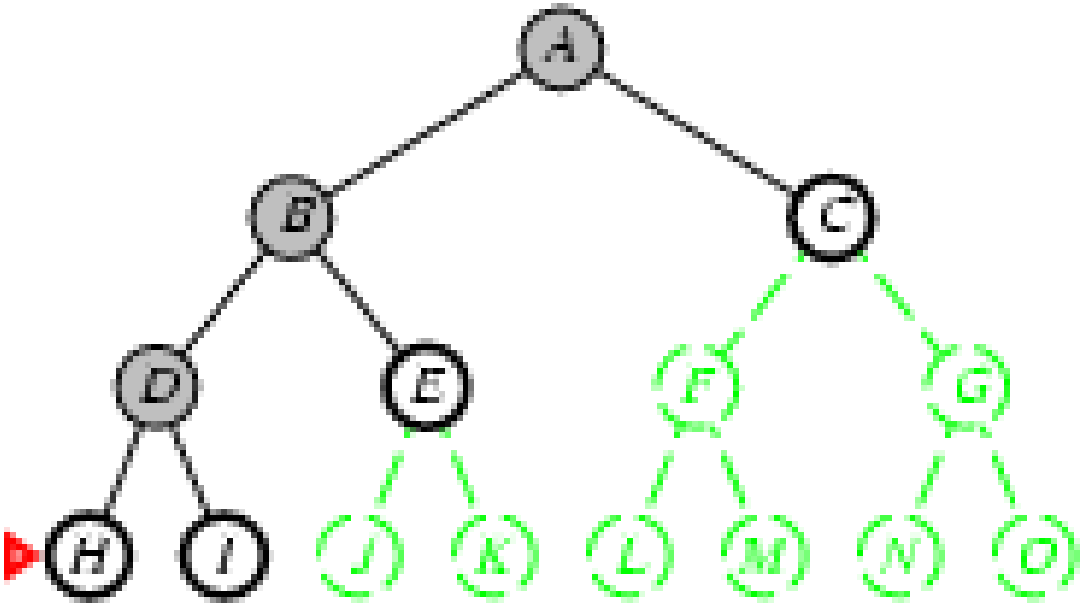
Po dubini



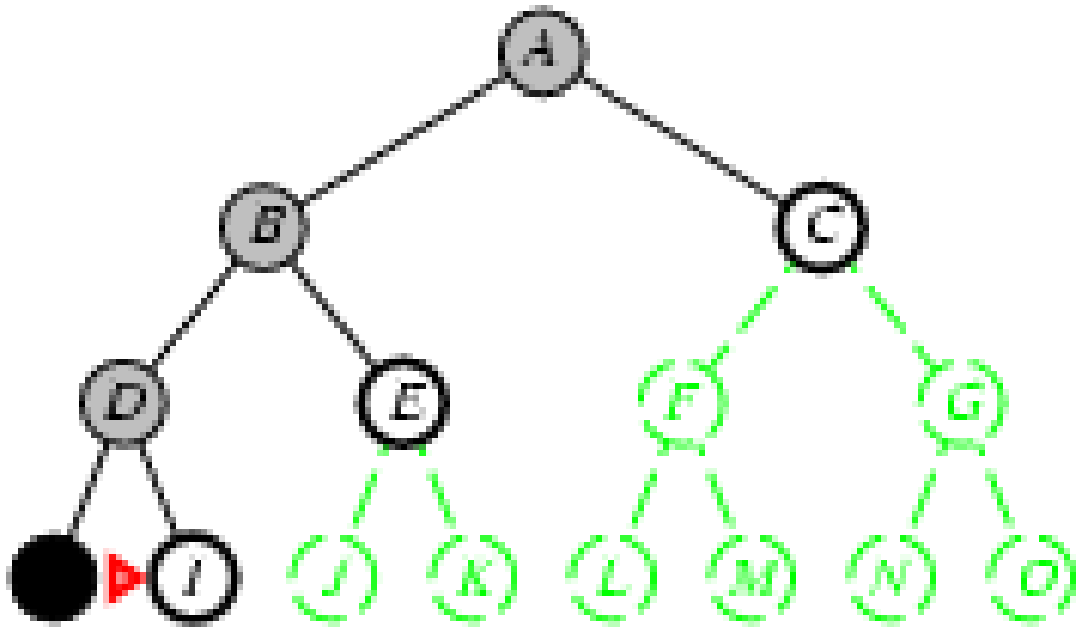
Po dubini



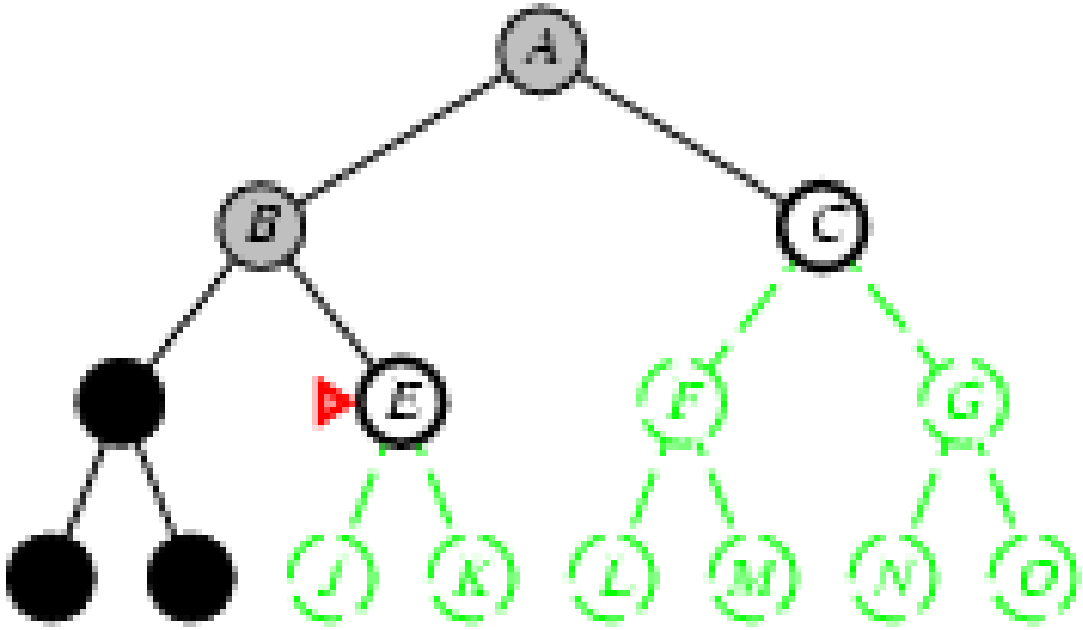
Po dubini



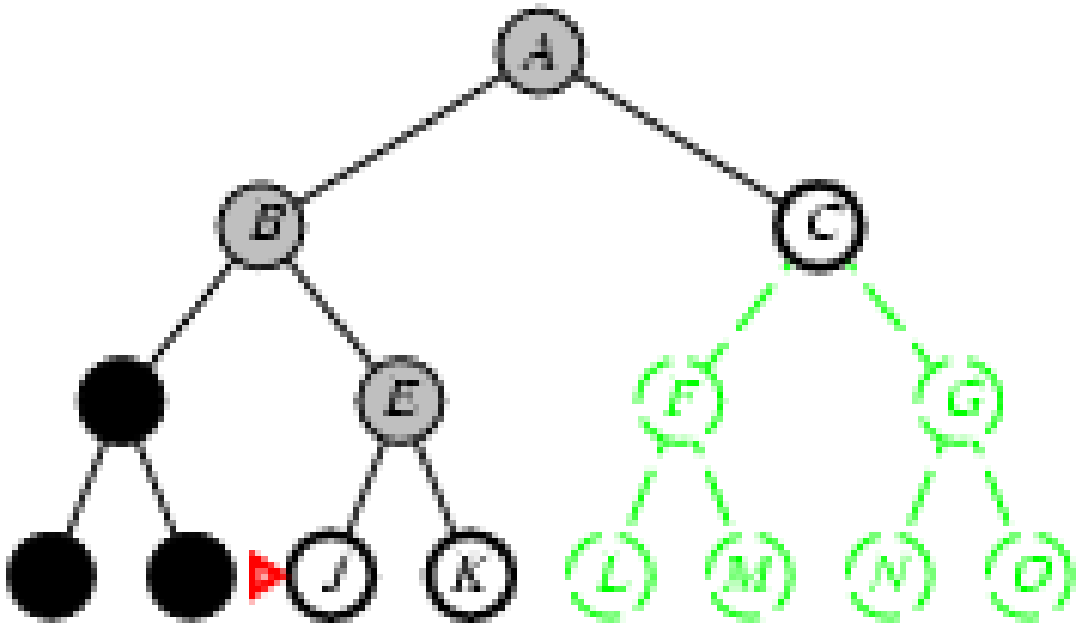
Po dubini



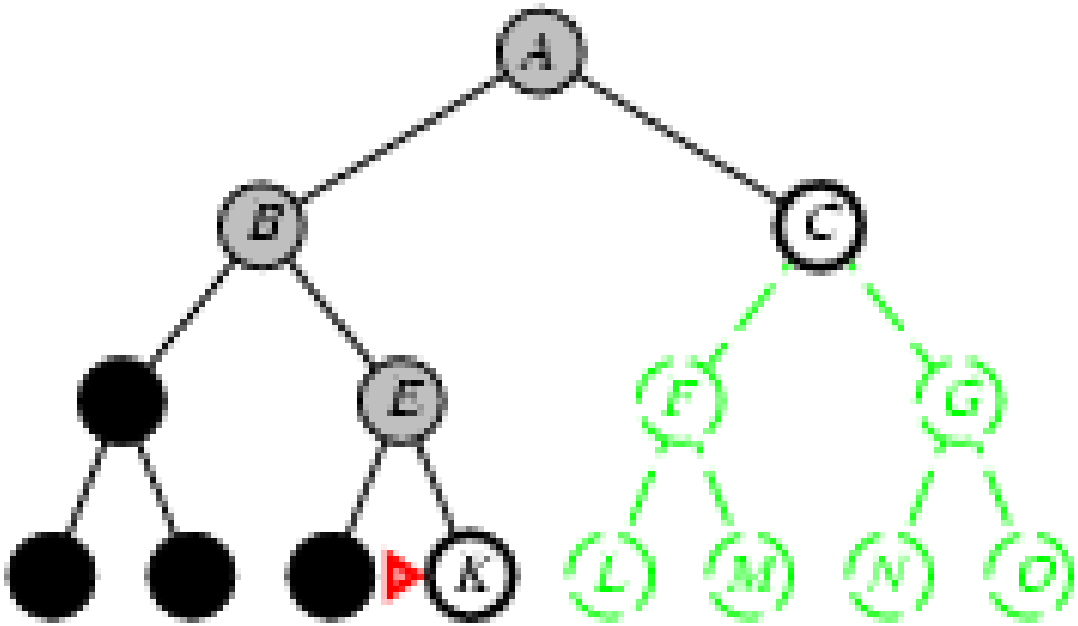
Po dubini



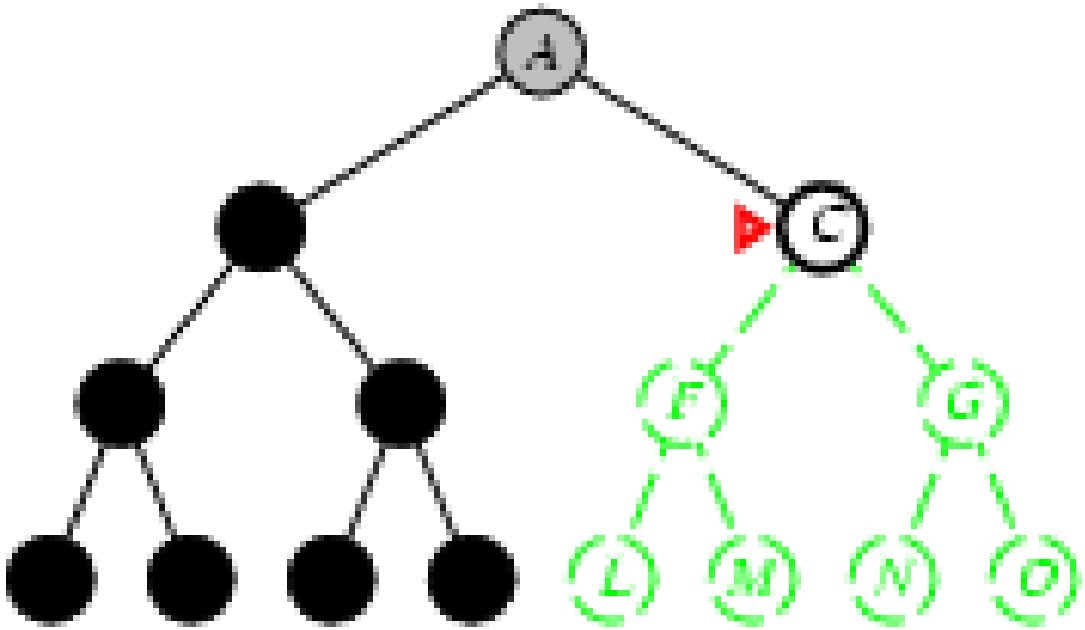
Po dubini



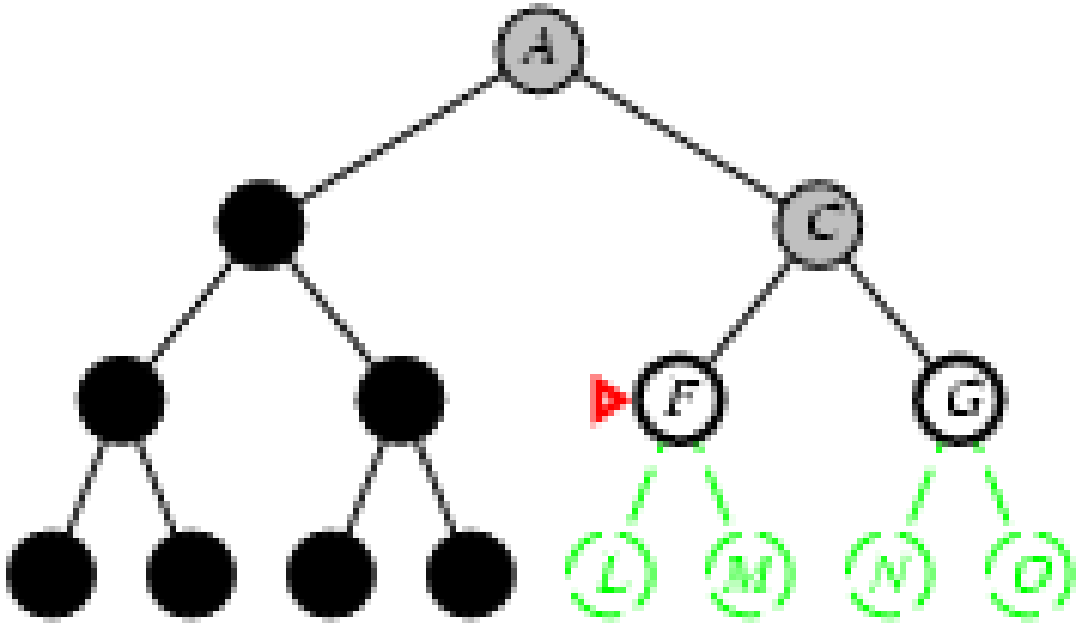
Po dubini



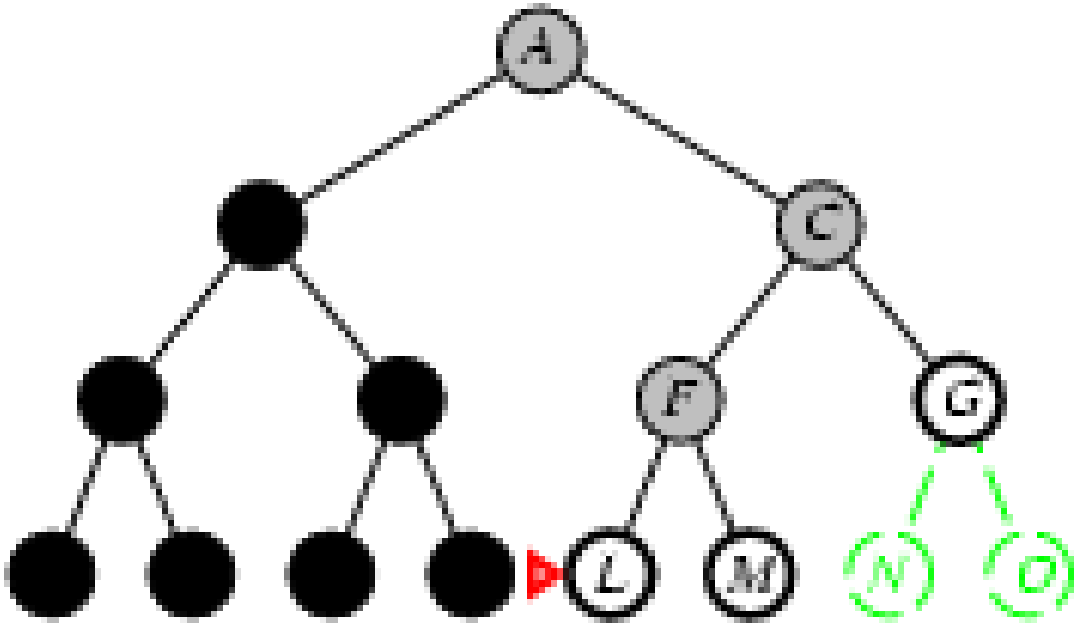
Po dubini



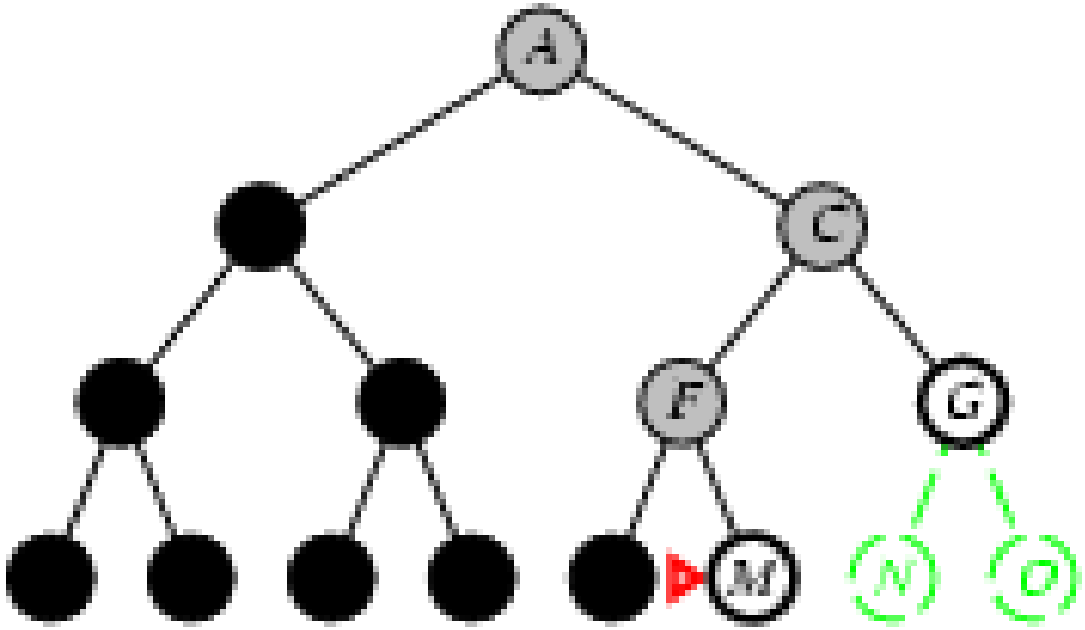
Po dubini



Po dubini



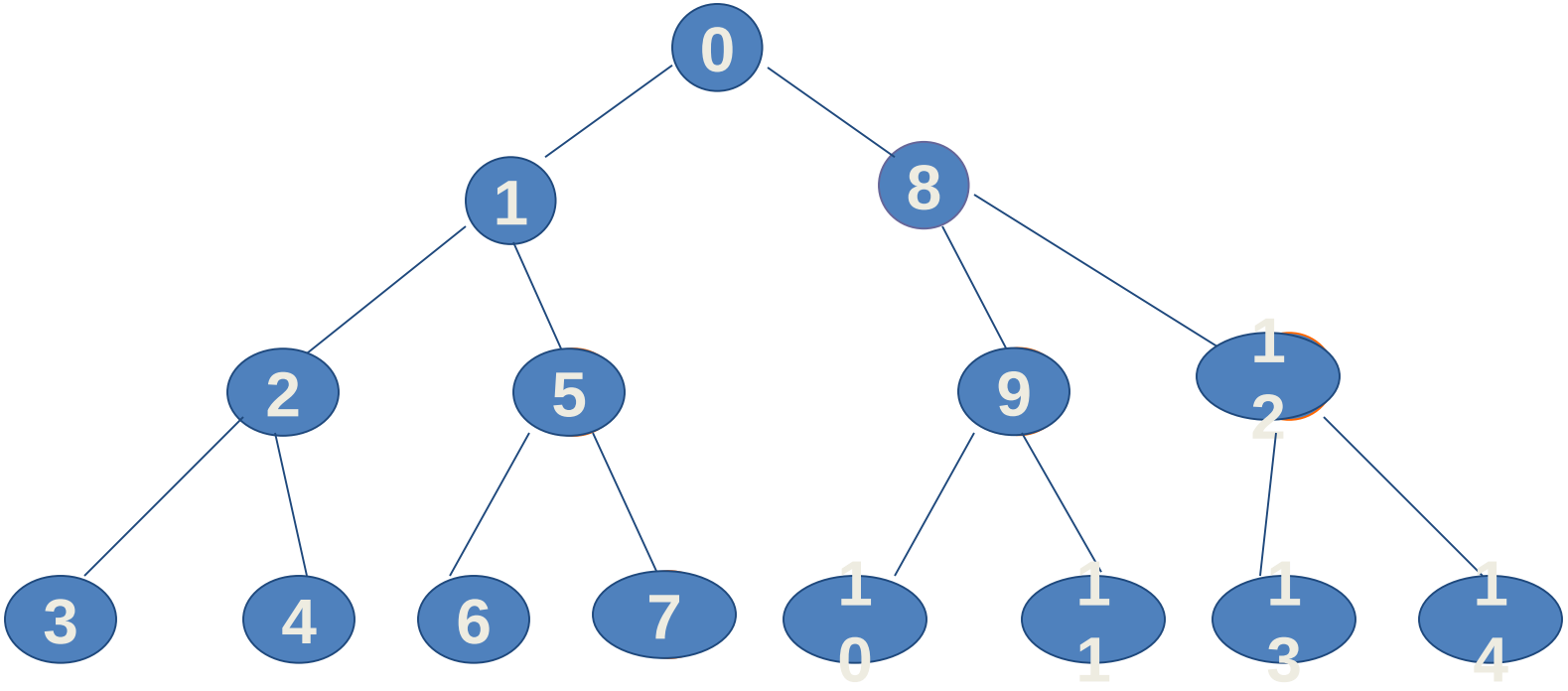
Po dubini



Osobine

- Kompletno? Ne: problem kod beskonačnih dubina, prostori sa petljama
- Vreme? $O(b^m)$
- Prostor? $O(bm)$, linearno
- Optimalno? Ne

Po dubini (Depth-first)



Brojevi prikazuju redosled posete pri DF pretraživanju

Po dubini (Depth-first)

- Prednosti:
  - zahteva manje memorije, jer se posmatra ograničen prostor pretraživanja za dati nivo,
  - problemi koji imaju rešenja na dubokim nivoima biće pre rešeni
- Nedostaci su:
  - ako su pretraživanja zauzeta na nekom nivou (zbog mogućnosti beskonačne putanje), tada se može desiti da se ne nađe rešenje iako ono postoji,
  - ako se pronađe rešenje, nema garancije da je ono najkraće

Planinarenje (hill-climbing)

1. Formirati listu čvorova koja inicijalno sadrži samo startni čvor.
2. Dok se lista čvorova ne isprazni ili se ne dođe do ciljnog čvora, proveriti da li je prvi element liste ciljni čvor

2.1. Ako je prvi element ciljni čvor, ne raditi ništa.

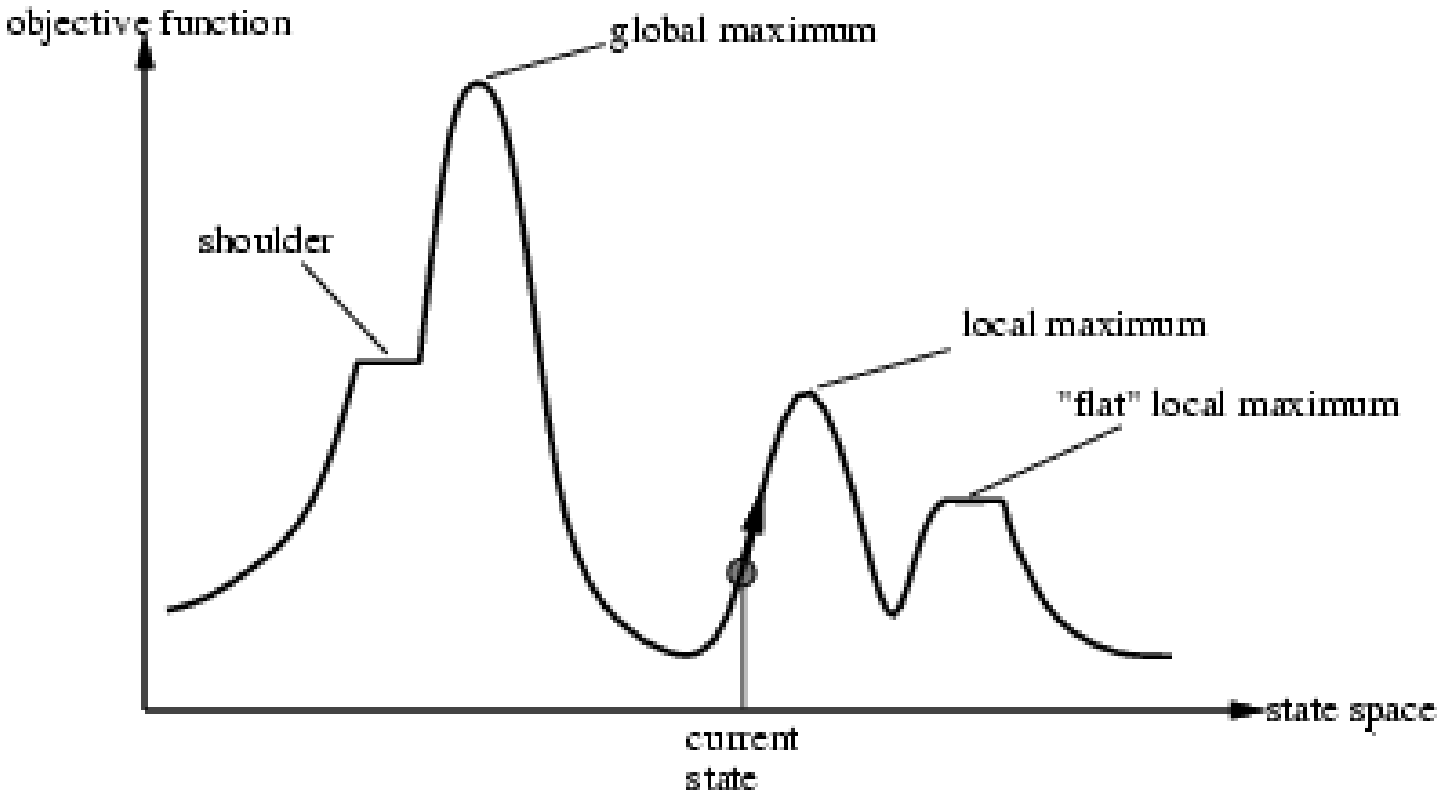
2.2. Ako prvi element nije ciljni čvor, ukloniti prvi element iz liste, sortirati njegove sledbenike iz stabla pretrage (ako ih ima) po rastućim vrednostima heurističke funkcije. Zatim te sledbenike dodati *na početak* liste tako da prvi element liste bude sledbenik sa najmanjom vrednošću heurističke funkcije.

3. Ako je pronađen ciljni čvor, pretraga je uspešno završena; u suprotnom pretaga je neuspešna.

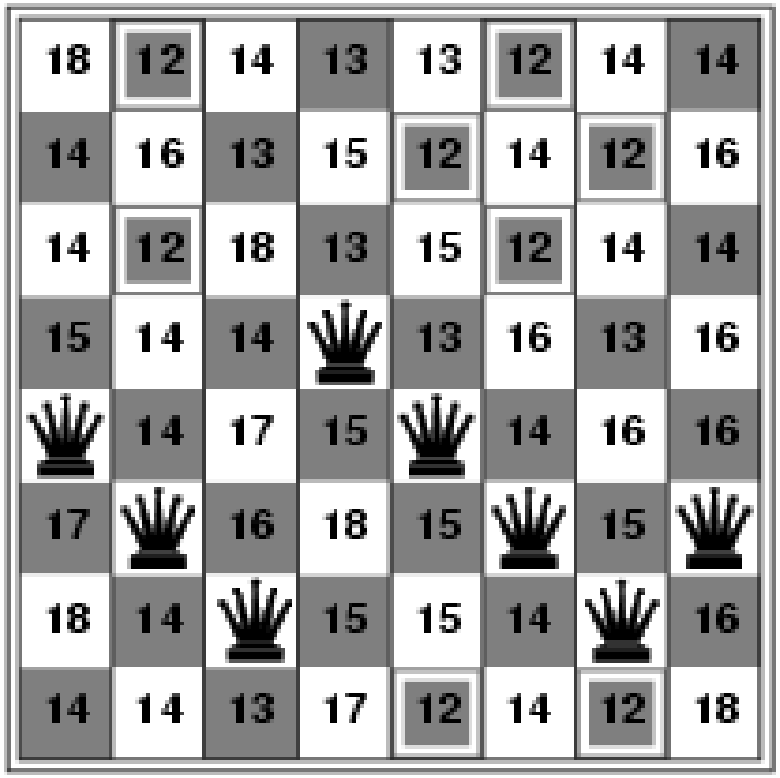
Planinarenje (hill-climbing)

- Prednost je:
  - manja je kombinatorna eksplozija u poređenju sa globalnim metodom
- Nedostatak je:
  - lokalna metoda, pa ne postoji garancija da će da bude efektivna
- Ova metoda nije pogodna kod velikih prostora pretraživanja, jer je veća verovatnoća da će doći do nekog od navedenih neželjenih efekata.

Planinarenje



Planinarenje : 8-kraljica



- *h* = broj parova kraljica koje se medjusobno napadaju
- *h = 17* za prikazanu situaciju

Grananje i ograničavanje (branch and bound)

1. Formirati listu parcijalnih putanja. Inicijalno lista sadrži samo jednu putanju nulte dužine koja koja sadrži samo startni čvor.
2. Dok se lista čvorova ne isprazni ili se ne dođe do ciljnog čvora, proveriti da li je prvi element liste putanja koja dostiže ciljni čvor.

2.1. Ako je prva putanja dostigla ciljni čvor, ne raditi ništa.

2.2. Ako prva putanja nije dostigla ciljni čvor, uraditi sledeće:

2.2.1. Ukloniti prvu putanju iz liste.

2.2.2. Za svaki sledbenik poslednjeg čvora na uklonjenoj putanji formirati po jednu novu putanju produžujući sledbenikom uklonjenu putanju.

2.2.3. Za svaku od novodobijenih putanja izračunati ukupnu (kumulativnu) cenu koštanja c kao zbir cena koštanja operatora na toj putanji.

2.2.4. Dodati nove putanje u listu parcijalnih putanja.

2.2.5. Sortirati listu putanja po rastućim vrednostima cena koštanja putanja.

3. Ako je pronađen ciljni čvor, pretraga je uspešno završena; u suprotnom pretaga je neuspešna.

Generiši i testiraj (Generate and Test)

1. Generisati moguće rešenje, u nekim slučajevima to znači generisanje neke tačke u prostoru problema, u drugim, to znači generisanje putanje od startnog stanja.
2. Izvršiti ispitivanje da bi se ustanovilo da li je dobijeno rešenje, poredeći dobijenu tačku ili krajnju tačku generisane putanje sa skupom prihvatljivih ciljnih stanja.
3. Ako je pronađeno rešenje završiti pretraživanje, inače vratiti se na korak 1.

Generiši i testiraj (Generate and Test)

- Ovaj algoritam je neka vrsta pretraživanja po dubini, jer se mora generisati kompletno rešenje pre samog testiranja.
- U svojoj najsistematičnijoj formi, algoritam predstavlja iscrpno pretraživanje.
- Generisanje može da radi na bazi slučajnog generisanja rešenja, ali u tom slučaju nema garancije da će doći do rešenja.

Prvo najbolji (OR graf)

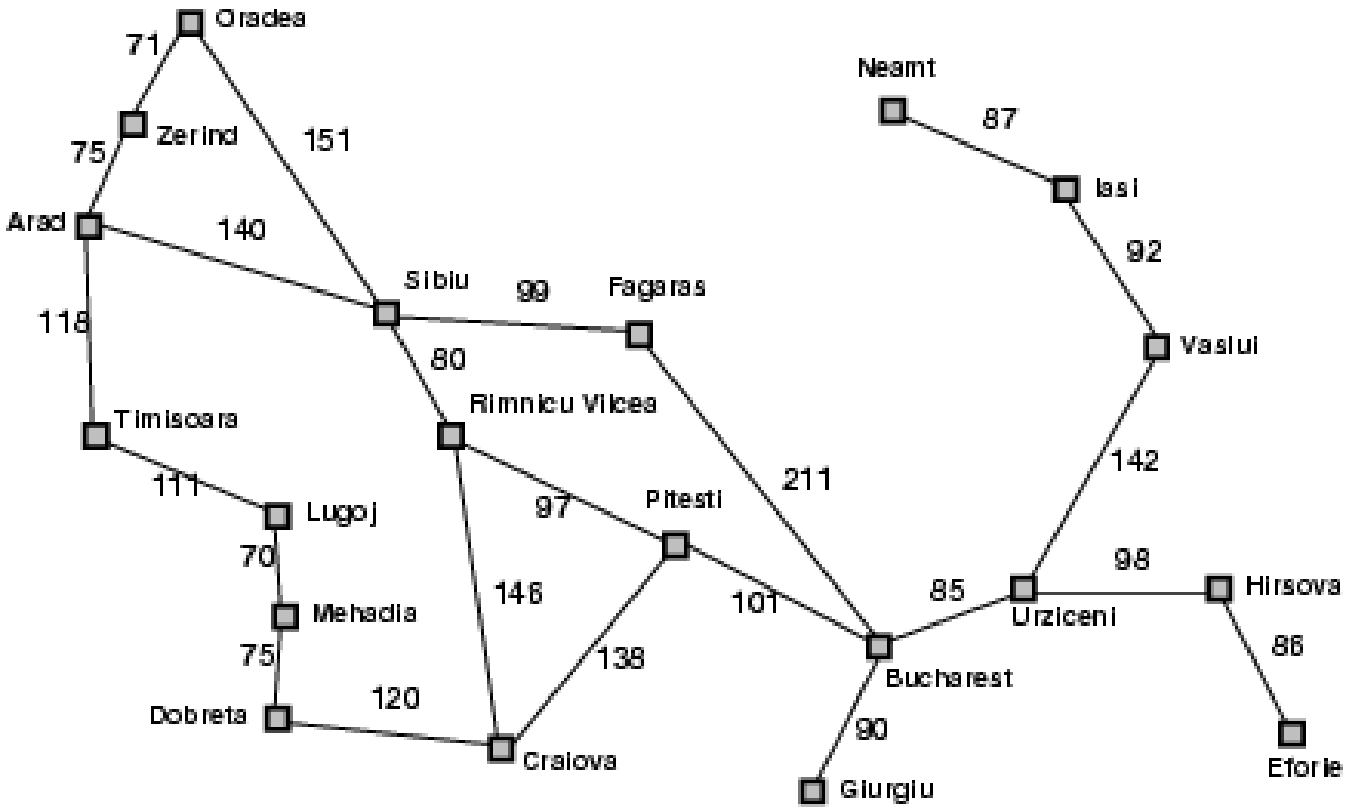
1. Formirati listu čvorova koja inicijalno sadrži samo startni čvor.
2. Dok se lista čvorova ne isprazni ili se ne dođe do ciljnog čvora, proveriti da li je prvi element liste ciljni čvor

2.1. Ako je prvi element ciljni čvor, ne raditi ništa.

2.2. Ako prvi element nije ciljni čvor, ukloniti prvi element iz liste i dodati njegove sledbenike iz stabla pretrage (ako ih ima) u listu. Celokupnu listu sortirati po rastućim vrednostima heurističkih funkcija čvorova.

3. Ako je pronađen ciljni čvor, pretraga je uspešno završena; u suprotnom pretaga je neuspešna

Put u Rumuniju

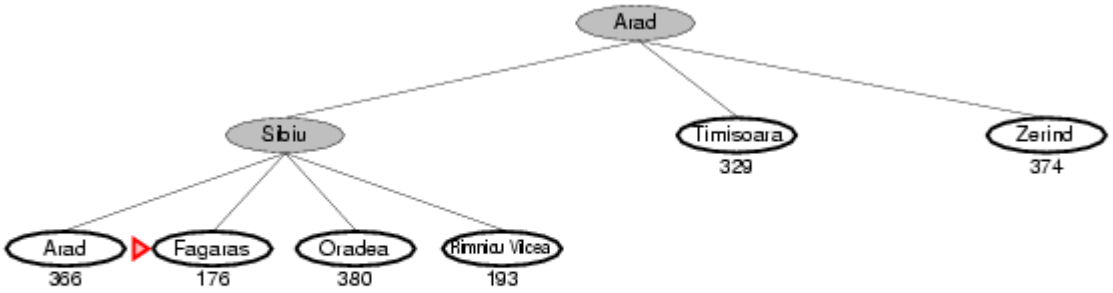


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

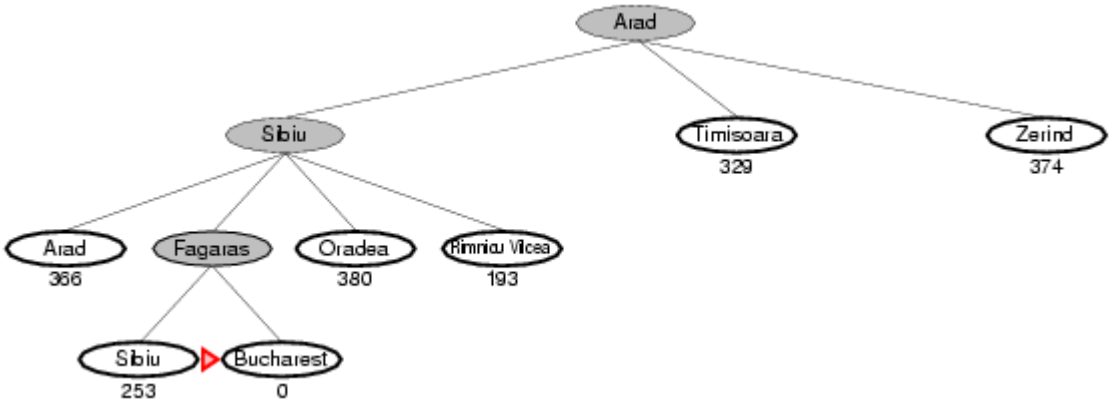
Prvo najbolji

Prvo najbolji

Prvo najbolji



Prvo najbolji



A*

1. Formirati listu parcijalnih putanja. Inicijalno lista sadrži samo jednu putanju nulte dužine koja koja sadrži samo startni čvor.
2. Dok se lista čvorova ne isprazni ili se ne dođe do ciljnog čvora, proveriti da li je prvi element liste putanja koja dostiže ciljni čvor.

2.1. Ako je prva putanja dostigla ciljni čvor, ne raditi ništa.

2.2. Ako prva putanja nije dostigla ciljni čvor, uraditi sledeće:

2.2.1. Ukloniti prvu putanju iz liste.

2.2.2. Za svaki sledbenik poslednjeg čvora na uklonjenoj putanji formirati po jednu novu putanju produžujući sledbenikom uklonjenu putanju.

2.2.3. Za svaku od novodobijenih putanja izračunati ukupnu (kumulativnu) cenu koštanja c kao zbir cena koštanja operatora na toj putanji; za poslednji čvor na putanji izračunati heurističku funkciju h. Funkciju procene f za svaku od novih putanja izračunati kao zbir heurističke funkcije h i cene koštanja putanje c ($f = h + c$).

2.2.4. Dodati nove putanje u listu parcijalnih putanja.

2.2.5. Sortirati listu putanja po rastućim vrednostima funkcije procene f.

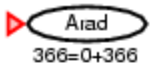
2.2.6. Ako dve ili više putanja iz liste imaju isti poslednji čvor, ukloniti iz liste sve takve putanje osim jedne koja ima najmanju cenu koštanja (princip dinamičkog programiranja).

3. Ako je pronađen ciljni čvor, pretraga je uspešno završena; u suprotnom pretaga je neuspešna.

A*

- Izbor funkcije g – cena koštanja – omogućava da se izabere sledeći čvor za ekspanziju, ne samo na osnovu toga koliko nam je dobar čvor, već i koliko je dobra putanja do njega.
- Ako se ne zna da li će se doći do rešenja, postavlja se g=0.
- Ako se želi da se pronađe najkraća putanja iz najmanje koraka postavlja se g=1.
- Ako se želi pronaći najadekvatnija putanja postavlja se vrednost g da odgovara stvarnoj ceni koštanja.
- Izbor funkcije h. Ako je h:
  - tačna (konverguje direktno ka cilju bez pretraživanja),
  - približna (bliže direktnom pristupu),
  - h=0 i g≠0 (pretraživanje na osnovu cene koštanja)
  - h=0 i g=0 (pretraživanje slučajno)
  - g=0 i h≠0 (pretraživanje po širini)
- Ako h nije precenjeno, A* vodi ka optimalnom rešenju,ako rešenje postoji

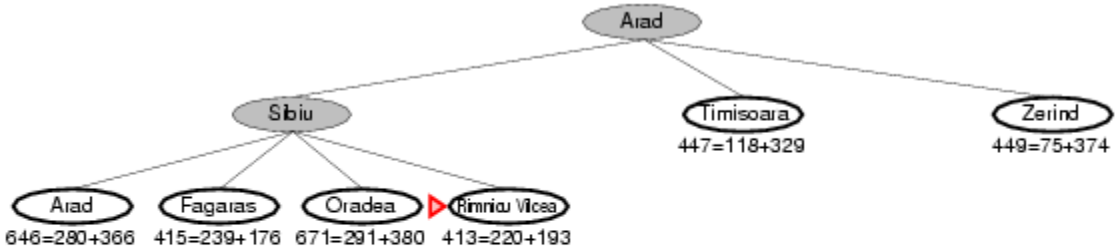
A*



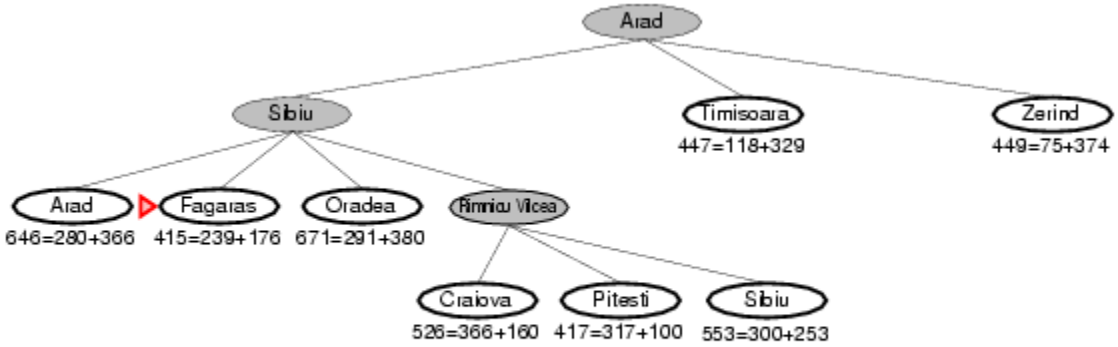
A*


Arad
Sibiu
393=140+253
Timisoara
447=118+329
Zerind
449=75+374

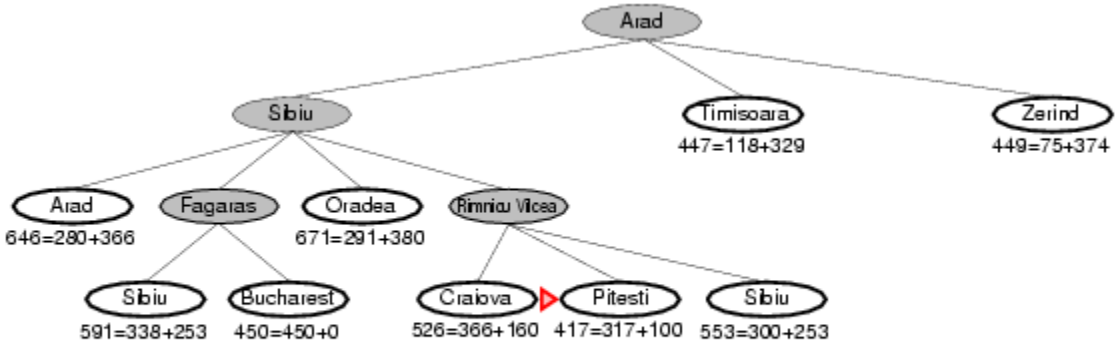
A^*


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu Vilcea
413=220+193

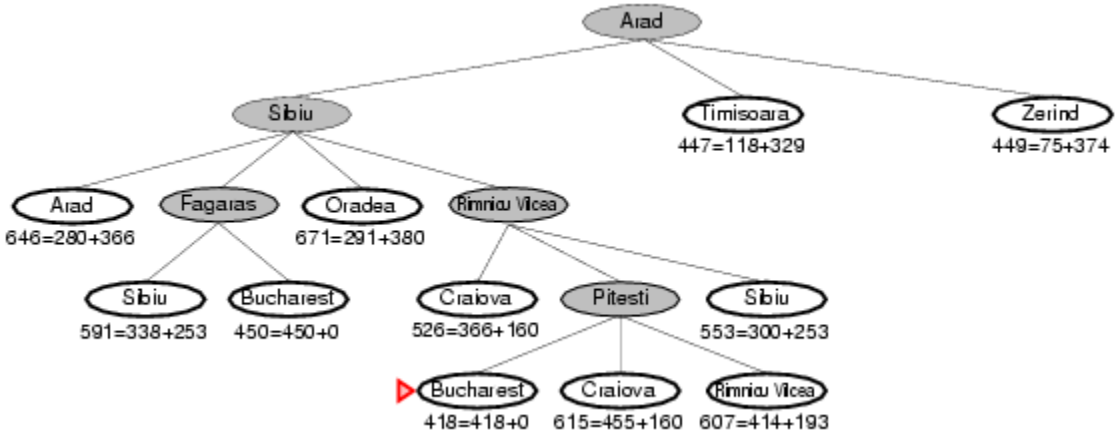
A*


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu Vilcea
Craiova
526=366+160
Pitesti
417=317+100
Sibiu
553=300+253

A*


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
Oradea
671=291+380
Rimnicu Vilcea
Sibiu
591=338+253
Bucharest
450=450+0
Craiova
526=366+160
Pitesti
417=317+100
Sibiu
553=300+253

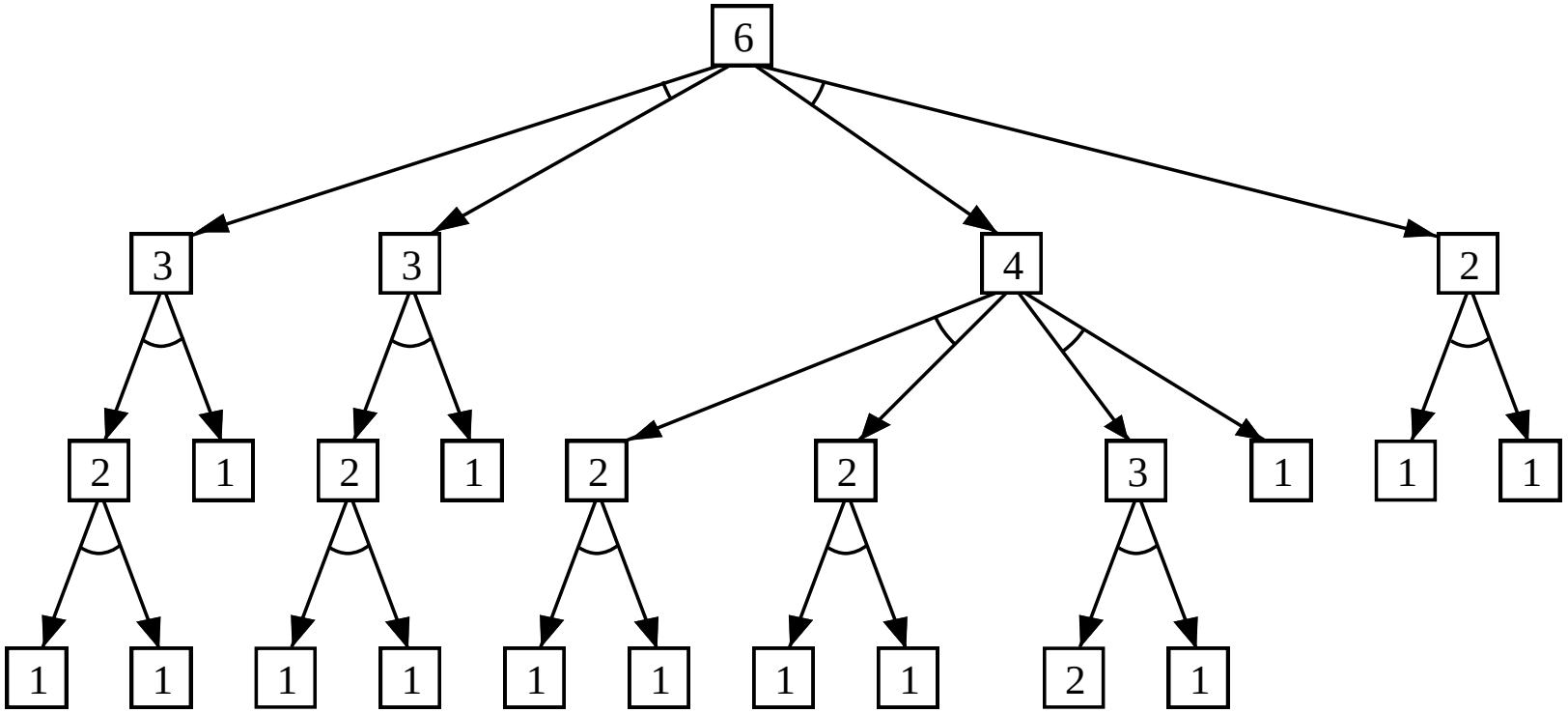
A^*


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
Oradea
671=291+380
Rimnicu Vilcea
Sibiu
591=338+253
Bucharest
450=450+0
Craiova
526=366+160
Pitesti
Sibiu
553=300+253
Bucharest
418=418+0
Craiova
615=455+160
Rimnicu Vilcea
607=414+193

AO*

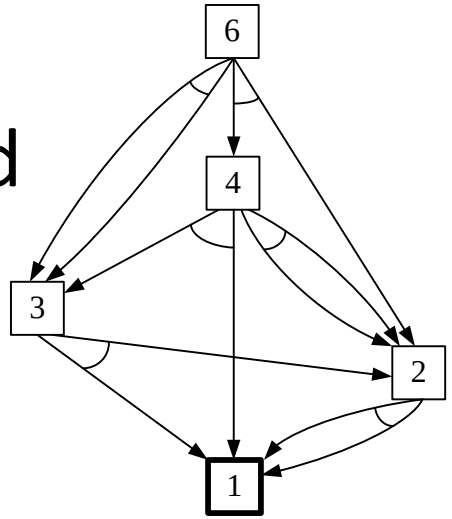
- Neki problem se može se razložiti (dekomponovati) na niz potproblema tako da svaki problem rešavamo nezavisno
- Pogodna predstava takvih problema je uz upotrebu AND-OR stabla.
- Čvorovi stabla su pojedinačni potproblemi, koji su povezani takozvanim *k-konektorima*.
- Radi se o generalizovanim granama koje povezuju jedan čvor-roditelj sa k čvorova naslednika.
- Konektor je predstavljen nizom grana koje čvor roditelj povezuju sa svakim od naslednika i koje su sve međusobno povezane lukom.
- U opštem slučaju, potproblem je rešen ako je rešen primenom bilo kog od konektora iz odgovarajućeg čvora grafa, pri čemu po tom konektoru svi čvorovi naslednici moraju biti rešeni.
- Konektor izražava I relaciju (AND) dok postojanje više konektora iz istog čvora izražava ILI relaciju (OR) - AND-OR stabla.

AO*



AO*

- Rešenje problema se može predstaviti još kompaktnije koristeći, umesto AND-OR stabla, AND-OR acikličkи graf.
- AND-OR acikličkи graf je vrsta acikličkog grafa kod koga su grane generalizovane k-konektorima.
- U literaturi se ovakvi grafovi ponekad nazivaju i hipergrafovima



AO*

- Kod 'klasičnih' metoda pretrage, rešenje je predstavljeno putanjom u grafu pretrage od početnog do nekog od ciljnih čvorova.
- Pri korišćenju AND/OR grafova, cilj se predstavlja skupom ciljnih čvorova *N*. Rešenje je predstavljeno podgrafom G' kompletnog grafa pretrage G.
- Rešenje se, ako postoji, dobija tako što se, polazeći od startnog čvora *n*, izabere jedan od konektora koji od čvora *n* vodi ka čvorovima-naslednicima *n*1, *n*2,..., *n*k.
- Ukoliko svaki od čvorova naslednika predstavlja ciljni čvor, rešenje je pronađeno i sastoji se od izabranih čvorova povezanih izabranim konektorom.
- U suprotnom slučaju, za svaki od čvorova naslednika koji nije ciljni čvor, potrebno je izabrati jedan od konektora i uključiti taj konektor i njegove čvorove-naslednike u rešenje.
- Procedura uključivanja novih konektora i čvorova u rešenje ponavlja se sve dok u podgrafu G' postoji čvor koji nije ciljni, a za koji nije izabran konektor.
- S obzirom da procedura određivanja rešenja u AND/OR grafu uključuje proizvoljan izbor konektora, u opštem slučaju postoji više rešenja određenog problema pretrage

AO*

- Konektorima u AND/OR grafu, mogu se pridružiti cene koje reprezentuju cene upotrebe određenih pravila pri rešavanju problema.
- Na osnovu ovih cena može se definisati cena određenog rešenja čime se ustanovljava kriterijum za poređenje različitih rešenja.
- Cena k(*n*,*N*) za neki podgraf G' grafa G od startnog čvora *n* do skupa ciljnih čvorova *N* definiše se sledećom rekurzivnom formulom:
  - Ako je *n* element skupa *N,* onda je *k*(*n, N*) = 0.
  - Inače, čvor *n* poseduje izlazni konektor prema skupu čvorova n1,n2,...,ni. Neka je cena ovog konektora cn. Tada je cena k(*n* , *N*) kompletnog rešenja jednaka zbiru cena izlaznog konektora čvora *n* i cena svih podgrafova od čvorova naslednika do ciljnih čvorova iz skupa *N*:

    k(*n*,*N*) = cn + k(*n*1 , *N*) + k(*n*2 , *N*) + ... + k(*n*i , *N*)

AO*

- Pretragu algoritmom AO* usmerava heuristička funkcija koja se definiše za svaki čvor AND/OR grafa.
- Heuristička funkcija za čvor *n* mora da ispunjava određeni uslov da bi pronađeno rešenje bilo optimalno, analogno slučaju kada se koristi algoritam A*.
- U slučaju algoritma AO* heuristička funkcija h(n) za proizvoljan čvor n AND/OR grafa mora da predstavlja potcenjenu cenu optimalnog podgrafa rešenja od čvora n do ciljnih čvorova.
- Izvršavanje algoritma AO* sastoji se iz ponavljanja dve glavne faze:
  - ekspanzije izabranog čvora grafa
  - revizije funkcija procene čvorova grafa

AO*

- Prva faza je ekspanzija izabranog čvora AND/OR grafa i dodavanje njegovih izlaznih konektora i čvorova-naslednika u graf.
- Jedan od konektora, koji ima najbolju funkciju procene, pri tome biva obeležen kao rezultat druge glavne faze algorima.
- U svakom trenutku pretrage *podgraf najboljeg parcijalnog rešenja* može se dobiti polazeći od startnog čvora grafa i prateći obeležene konektore.
- Sledeći čvor koji će biti ekspandovan uvek je jedan od čvorova koji pripadaju podgrafu najboljeg parcijalnog rešenja koji nije ciljni čvor.
- Rešenje je nađeno kada se startni čvor obeleži kao REŠEN
- Čvor *n* je rešen kada su ekspandovani svi čvorovi koji nisu ciljni u podgrafu koji polazi od čvora *n* i ide preko obeleženih konektora do ciljnih čvorova.

AO*

- Druga glavna faza algoritma je revizija funkcija procene čvorova u grafu.
- Funkcija procene f(*n*) čvora *n* predstavlja procenu cene podgrafa optimalnog rešenja od čvora *n* do skupa ciljnih čvorova.
- Kada se neki čvor *n* unese u graf, a pre nego što se taj čvor ekspanduje, njegova funkcija procene f(*n*) je inicijalno jednaka vrednosti njegove heurističke funkcije h(*n*).
- Za razliku od algoritma A*, gde se funkcija procene računa samo jedanput za svaki čvor stabla pretrage, u AO* algoritmu postoji potreba za revizijom funkcija procene čvorova.
- Kada se čvor *n* ekspanduje, njegova funkcija procene se ažurira na osnovu cena izlaznih konektora iz tog čvora i vrednosti funkcija procene čvorova naslednika.
- Ako iz čvora *n* ide više izlaznih konektora, uzima se najbolja vrednost po nekom od konektora i taj konektor se obeležava čime se produžava najbolje parcijalno rešenje.
- Revidiranu vrednost funkcije procene čvora *n* potrebno je proslediti nagore u grafu.
- Pri tome se (eventualno) ažuriraju funkcije procene čvorova-prethodnika čvora *n* po *obeleženim* konektorima.
- Kako je procena uvek potcenjena veličina, a ažuriranjem se dobijaju preciznije procene, vrednost funkcije procene ažuriranjem može samo da se poveća.
- Ažuriranje može da dovede do toga da za određeni čvor funkcija procene po nekom od konektora koji nije obeležen postane povoljnija od vrednosti po obeleženom konektoru, pa je tada potrebno premestiti obeležje na neobeleženi konektor.

AO*

1. Graf pretrage G sastoji se inicijalno samo od startnog čvora n0. Funkcija procene f za startni čvor jednaka je njegovoj heurističkoj funkciji f(n0) = h(n0).
2. Sve dok startni čvor ne bude obeležen kao REŠEN, raditi sledeće:

2.1. Odrediti graf parcijalnog rešenja G' sledeći markirane konektore u grafu pretrage G od startnog čvora. Inicijano G' se sastoji samo od startnog čvora. Izabrati (na proizvoljan način) neki od nerazvijenih čvorova iz G'.

2.2. Razviti izabrani čvor n. Ako se neki od čvorova naslednika čvora n ne nalazi u grafu, uneti ga u graf i vrednost njegove funkcije procene f postaviti na vrednost heurističke funkcije toga čvora. Ako uneti čvor predstavlja jednog od ciljnih čvorova, obeležiti ga kao REŠENOG.

2.3. Ažurirati funkciju procene izabranog čvora n na sledeći način:

2.3.1. Funkcija procene f(n) čvora n jednaka je minimumu svih funkcija procene fKi po svakom od izlaznih konektora K1, K2 do Kp:

$$f(n) = \min(fK1, fK2, ..., fKp)$$

pri čemu se funkcija procene fKi izlaznog konektora Ki računa kao zbir cene cKi konektora Ki i (ranije izračunatih) vrednosti funkcija procene fnj svih čvorova n1, n2,...,nm koji su konektorom Ki spojeni sa izabranim čvorom n:

$$fKi = cKi + f(n1) + f(n2) + ... + f(nm).$$

2.3.2. Obeležiti onaj konektor Ki među izlaznim konektorima izabranog čvora n koji ima minimalnu vrednost funkcije procene (prethodno ukloniti oznaku sa ranije obeleženog konektora, ako takav postoji).

2.3.3. Ako su svi čvorovi naslednici čvora n po izabranom konektoru Ki obeleženi kao REŠENI, obeležiti i čvor n kao REŠEN.

2.3.4. Ako je funkcija procene čvora n izračunata u koraku 2.3.1 različita (veća od) stare ili je čvor n u koraku 2.3.3. obeležen kao rešen, ponoviti korak 2.3. (ažurirati njihove funkcije procene i status rešenosti) za svaki od prethodnika čvora n po obeleženom konektoru u grafu G'.

Zadovoljenje ograničenja

1. Dok se ne nađe potpuno rešenje ili dok sve putanje ne dovedu u ćorsokak, uraditi sledeće:

1.1. Izabrati nerazvijen čvor grafa pretraživanja

1.2. Primeniti sva ograničenja na izabrani čvor kako bi se generisala sva moguća nova ograničenja

Ako skup ograničenja sadrži protivrečnost, obavestiti da je putanja ćorsokak

Ako skup ograničenja opisuje potpuno rešenje, tada obavestiti o uspehu

Ako se ne radi o protivrečnosti ili o potpunom rešenju, tada se primenjuje pravilo za generisanje novog parcijalnog rešenja koje je konzistentno sa tekućim skupom ograničenja. Ubaciti parcijalno rešenje u graf pretraživanja.

Sukcesivne aproksimacije (Means-ends Analysi's)

Da bi se iz tekućeg stanja došlo u ciljno stanje, treba uraditi sledeće:

1. Formirati listu koja će inicijalno sadržati samo tekuće stanje.
2. Nazovimo stanje na čelu liste tekućim stanjem. Dok se lista ne isprazni ili dok se ne dostigne ciljno stanje, raditi sledeće:

2.1. Ako se pregledom tabele razlika ustanovi da ne postoji neupotrebljen operator za smanjenje razlike između tekućeg i ciljnog stanja, ukloniti tekuće stanje iz liste.

2.2. Inače, izabrati operator za smanjivanje razlike iz tabele razlika.

2.2.1. Ako preduslovi za primenu operatora nisu zadovoljeni, pokušati njihovo zadovoljavanje formiranjem novog ciljnog stanja od tih preduslova i rekurzivnim pozivom GPS algoritma radi dostizanja novog ciljnog stanja iz tekućeg.

2.2.2. Ako su preduslovi zadovoljeni primeniti operator i novodobijeno stanje staviti na početak liste stanja.

3. Ako se dostigne ciljno stanje, obavestiti o uspehu; u suprotnom, obavestiti o neuspehu.

Primer – robot i tetka

Tetka Marija koja živi u Tivtu pozvala je robota iz Beograda da dođe kod nje. Robot koristi sledeća pravila:

- ako je put duži od 250 km, putovati avionom ili vozom.
- ako je put duži od 50 km i kraći od 250 km, putovati vozom ili kolima.
- ako je put duži od 1 km, a kraći od 50 km, putovati kolima ili uzeti taksi.
- ako je put kraći od 1 km, ići pešice.

Preduslov za putovanje avionom je da se bude na aerodromu, za putovanje vozom da se bude na železničkoj stanici, a da se putuje kolima je da se poseduju kola.

Odrediti na koji će način Nenad, koji u Beogradu ima svoja kola, doputovati tetki Mariji.

operator	**Preduslov**	**akcija**
leteti avionom u mesto *x*	lokacija = aerodrom Surčin	lokacija = aerodrom u mestu *x*
putovati vozom u mesto *x*	lokacija = beogradska železnička stanica	lokacija = železnička stanica u mestu *x*
voziti se kolima u mesto *x*	lokacija = Nenadov parking	lokacija = *x*
voziti se taksijem u mesto *x*	-	lokacija = *x*
hodati do mesta *x*	-	lokacija = *x*

razlika r	putovati avionom	putovati vozom	voziti se kolima	voziti se taksijem	hodati
r > 250 Km	DA	DA			
50 Km< r < 250 Km		DA	DA		
1 Km < r < 50 Km			DA	DA	
r < 1 Km					DA

Nivo rekurzije: 1
Tekuće stanje: lokacija = Kuća robota
Ciljno stanje: lokacija = tetkina kuća u Tivtu
Razlika: r = 300 Km

Nivo rekurzije: 2
Tekuće stanje: lokacija = Kuća robota
Ciljno stanje: lokacija = aerodrom Surčin
Razlika: r = 10 Km

Nivo rekurzije: 3
Tekuće stanje: lokacija = Kuća robota
Ciljno stanje: lokacija = Parking
Razlika: r = 100 m

Nivo rekurzije: 3
Tekuće stanje: lokacija = Parking
Ciljno stanje: lokacija = Parking
Razlika: r = 0

Nivo rekurzije: 2
Tekuće stanje: lokacija = Parking
Ciljno stanje: lokacija = aerodrom Surčin
Razlika: r = 10 Km

Nivo rekurzije: 2
Tekuće stanje: lokacija = aerodrom Surčin
Ciljno stanje: lokacija = aerodrom Surčin
Razlika: r = 0

Nivo rekurzije: 1
Tekuće stanje: lokacija = aerodrom Surčin
Ciljno stanje: lokacija = tetkina kuća u Tivtu
Razlika: r = 300 km

Nivo rekurzije: 1
Tekuće stanje: lokacija = tivatski aerodrom
ciljno stanje: lokacija = tetkina kuća u Tivtu
Razlika: r = 7 km

Nivo rekurzije: 1
Tekuće stanje: lokacija = tetkina kuća u Tivtu
Ciljno stanje: lokacija = tetkina kuća u Tivtu
Razlika: r = 0

Primer – robot i tetka - rešenje