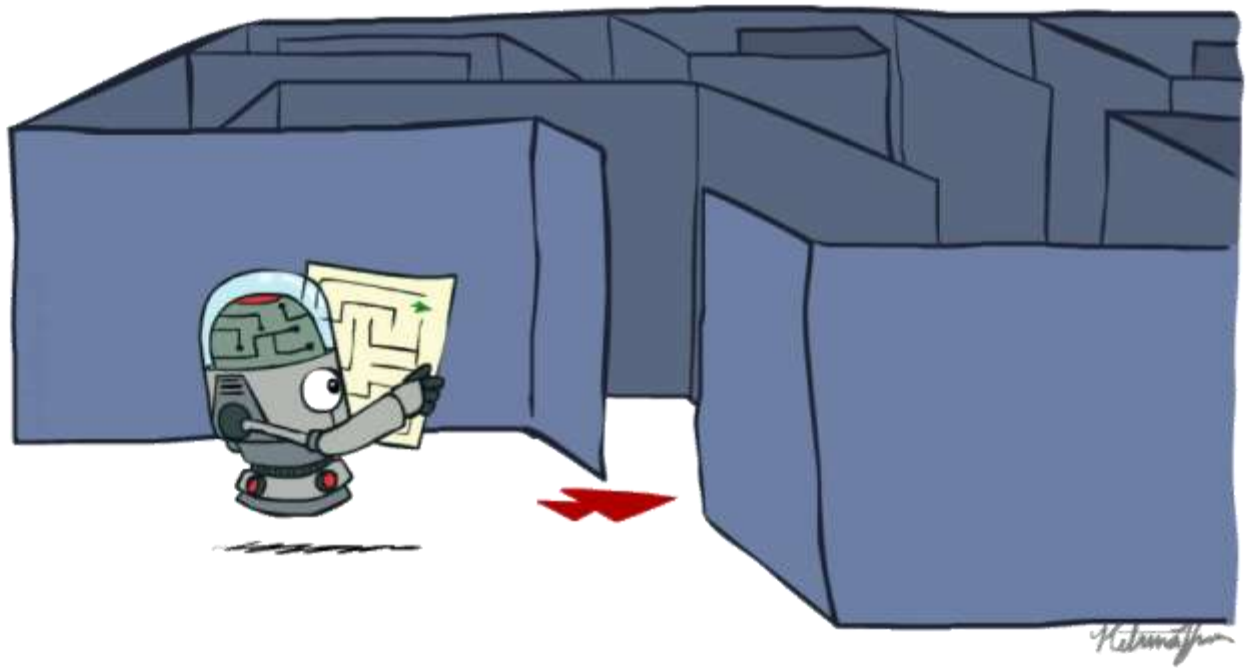
Osnovi Računarske Inteligencije

Pretrage



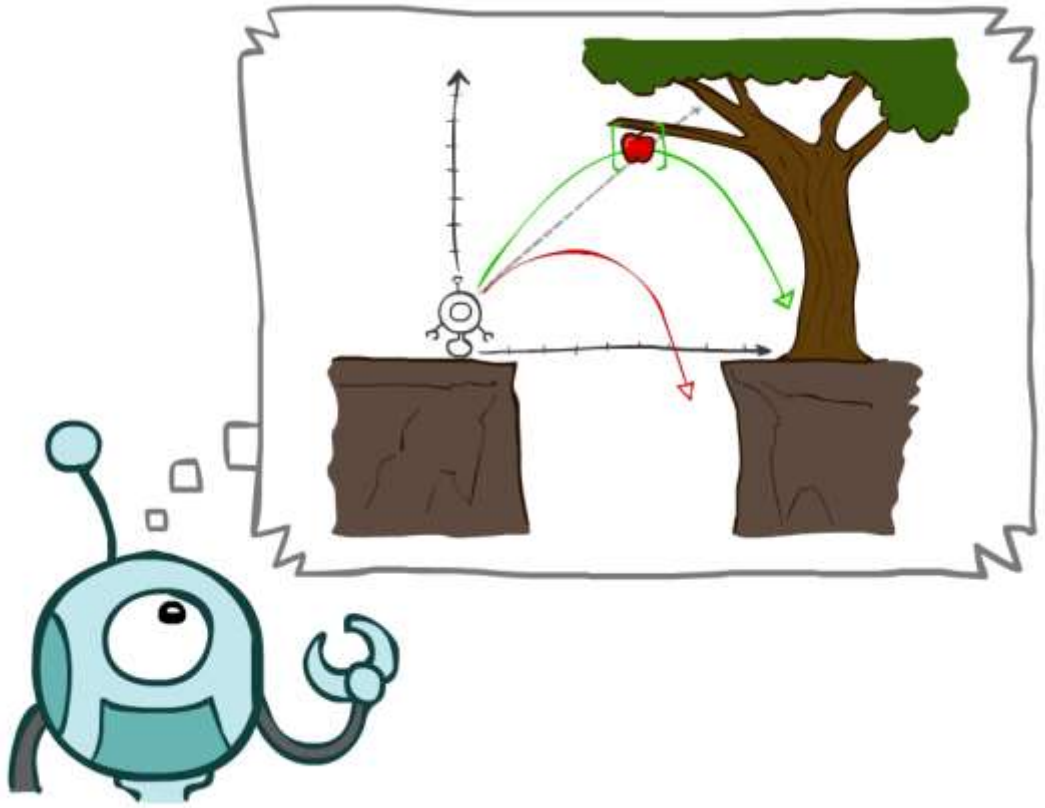
Predavač: Aleksandar Kovačević

Slajdovi preuzeti sa kursa CS188, University of California, Berkeley

http://ai.berkeley.edu/

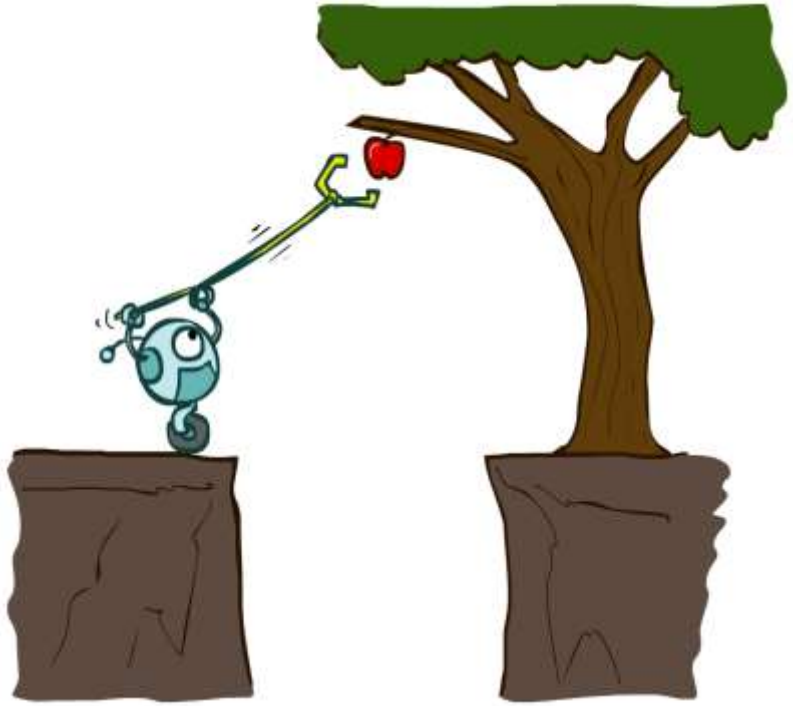
Prvi Deo

- Agenti Koji Planiraju Unapred
- Formulacija Problema Pretraga
- Uniformne Metode Pretrage
  - Prvi po Dubini (DFS)
  - Prvi u Širinu (BFS)
  - Pretraga sa Uniformnom Cenom (*Uniform-Cost Search*, UCS)

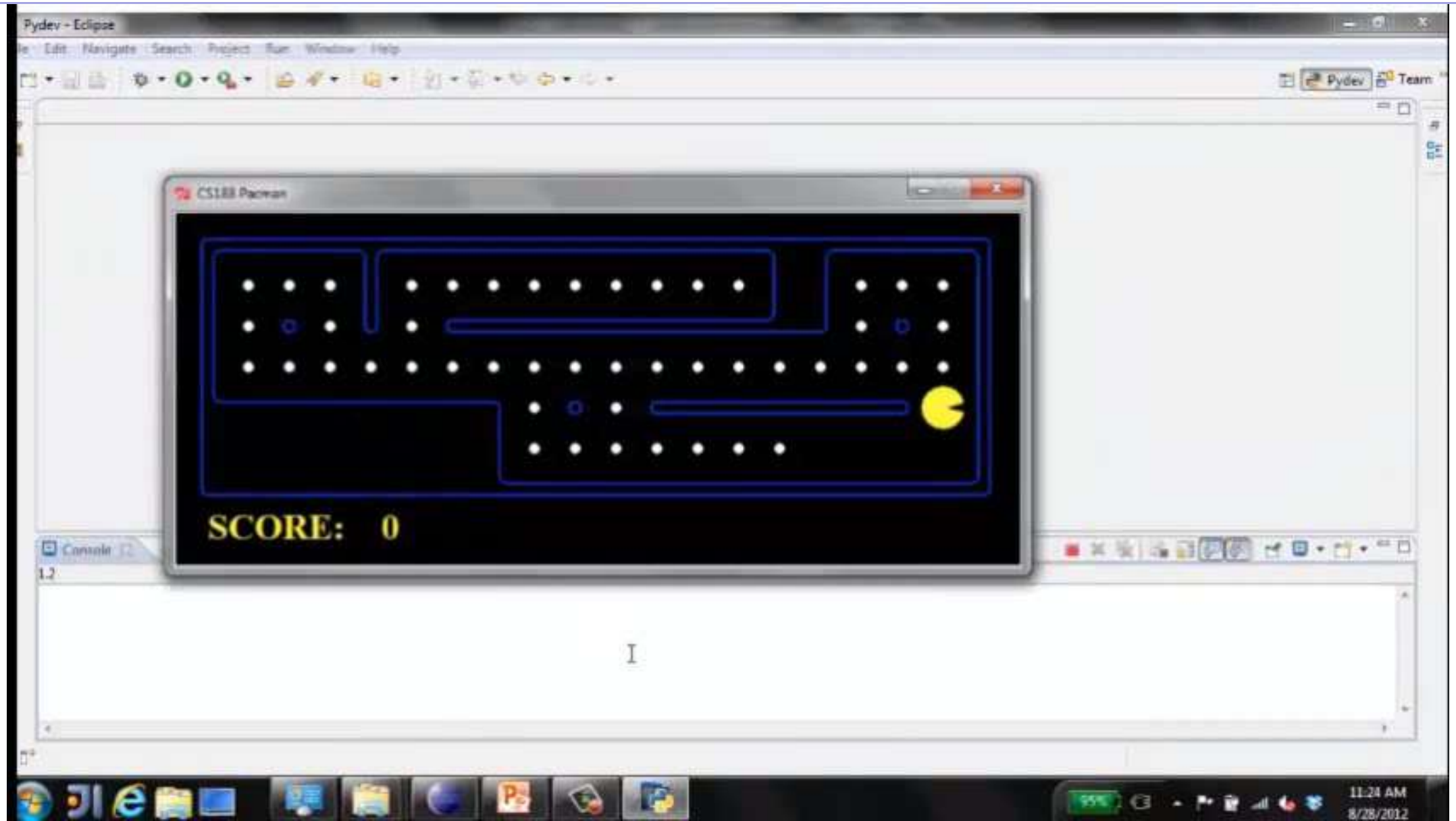


Agenti Koji Planiraju

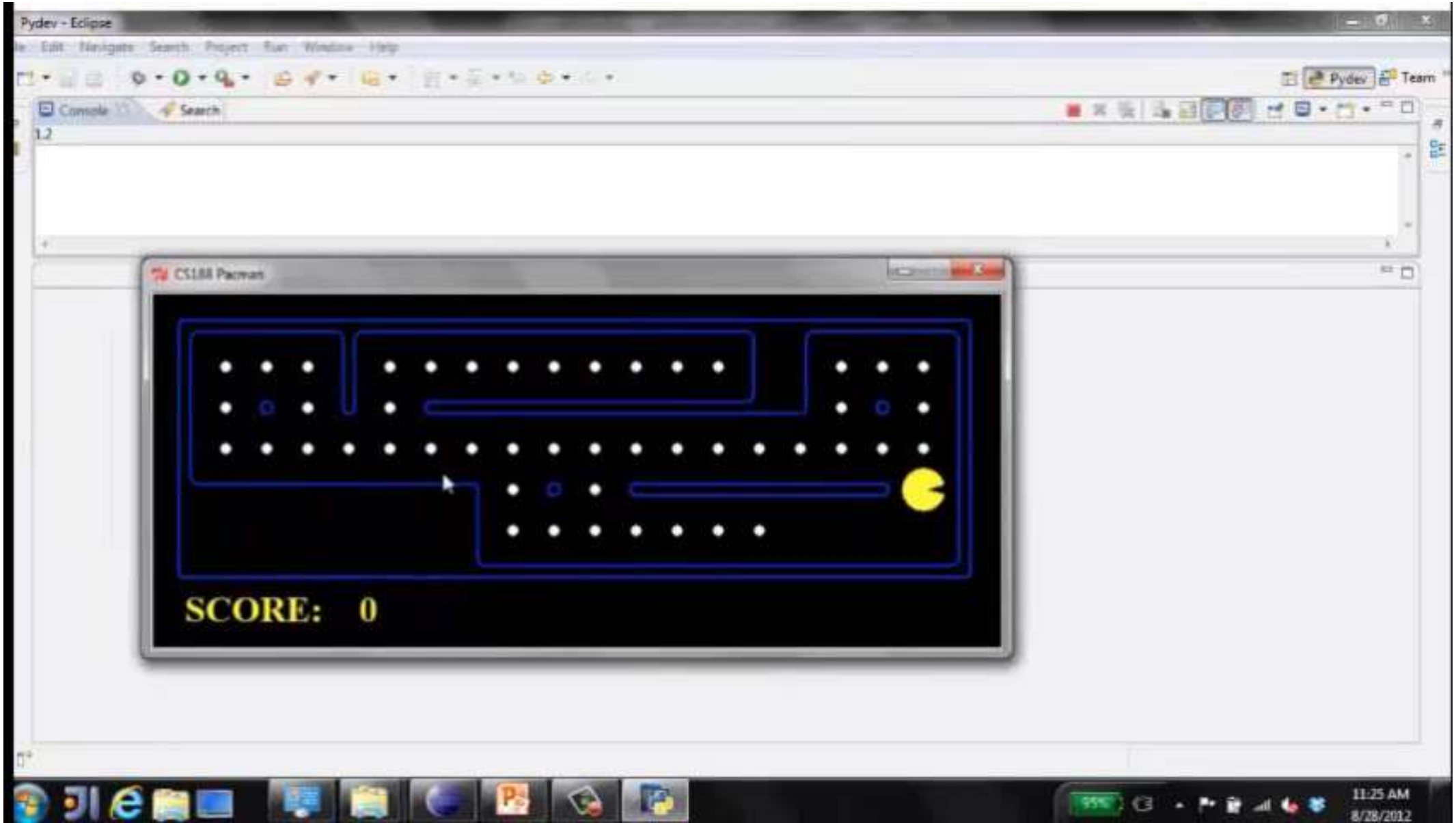
- Karakteristike:
  - Pitaju “šta ako”
  - Odluke su zasnovane na posledicama akcija
  - Agent mora imati uvid u model sveta tj. način na koji se svet menja kao posledica akcija.
  - Mora da postoji jasno formulisan cilj i test za cilj.
- Optimalni planovi (pretrage)
- Kompletni planovi (pretrage)
- Planiranje i Re-planiranje (promena plana)



Demo – Re-planiranje



Demo Mastermind

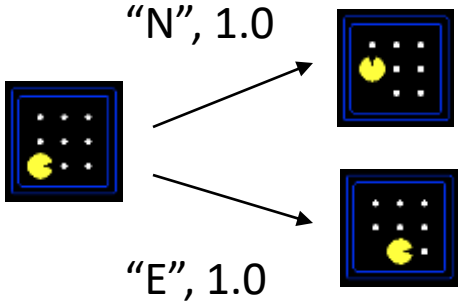


Formulacija Problema Pretraga

Formulacija Problema Pretraga

- Problem pretraga sastoji se od:
  - Prostora stanja

  - Funkcije za generisanje sledećih stanja (uz akcije i cene)

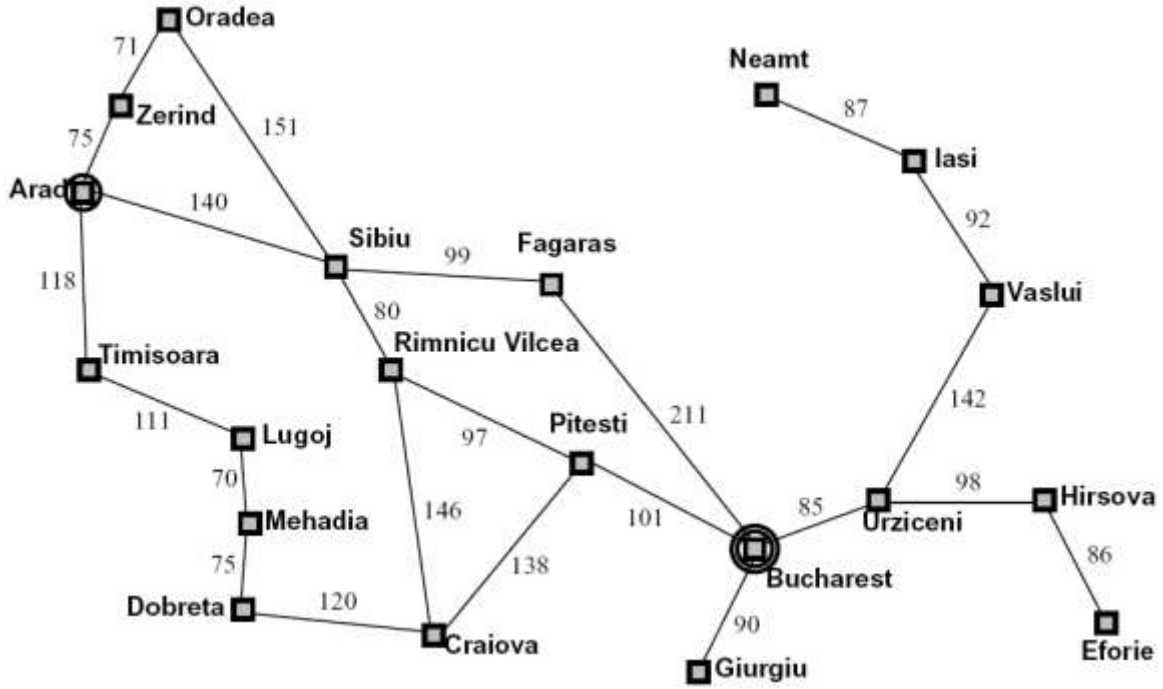


  - Početnog i Ciljnog stanja

- Rešenje je niz akcija (plan) koji nas od početnog dovodi do ciljnog stanja.

Problemi Pretraga su Modeli

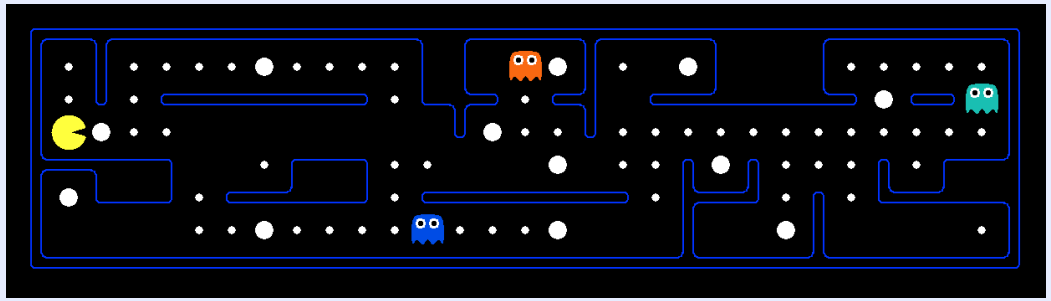
Primer: Putovanje u Rumuniji



- Prostor stanja:
  - Gradovi
- Funkcija za generisanje sledećih stanja:
  - Putevi: možemo preći iz jednog u susedni grad, cena = udaljenost
- Početno stanje:
  - Arad
- Test za ciljno stanje:
  - Da li je stanje == Bucharest?
- Rešenje?

Šta je prostor stanja?

Stanje sveta sadrži svaki detalj vezan za okruženje



Stanje pretrage sadrži samo detalje potrebne za pretragu (planiranje)

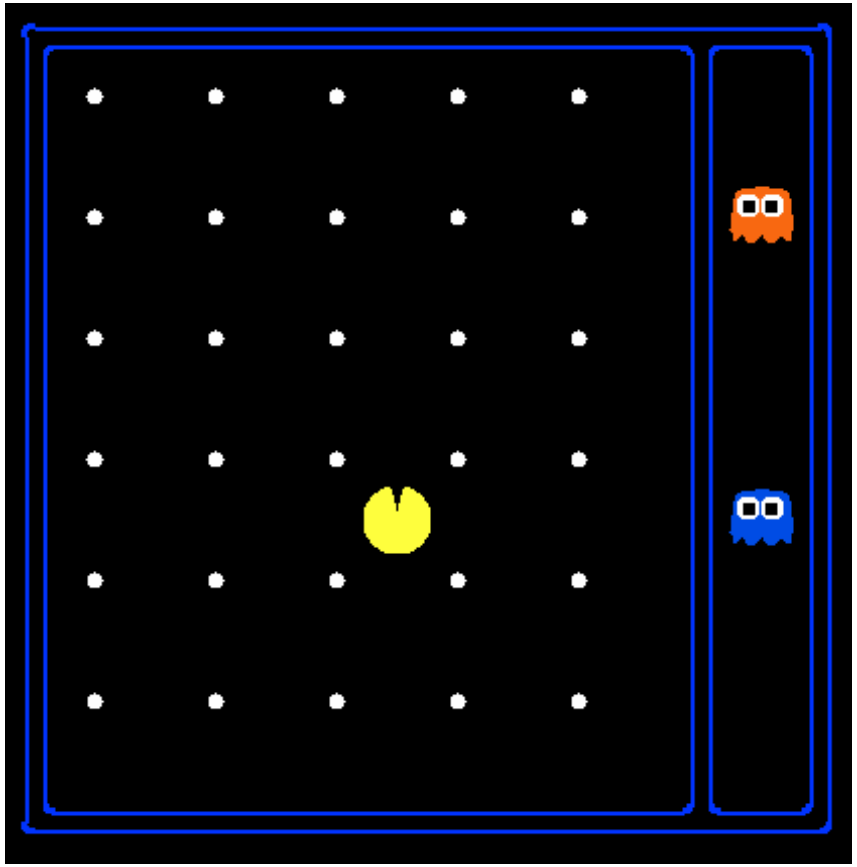
- Problem: Pronalaženje puta
  - Stanja: (x,y) lokacije
  - Akcije: NorthSouthEastWest
  - Za sledeće stanje: menjamo samo (x,y)
  - Test za cilj: da li je (x,y)=END

- Problem: Pojesti sve tačke
  - Stanja: {(x,y), boolean promenljive za sve tačke}
  - Actions: NSEW
  - Za sledeće stanje: menjamo (x,y) i ponekad boolean za tačku
  - Test za cilj: sve tačke = false

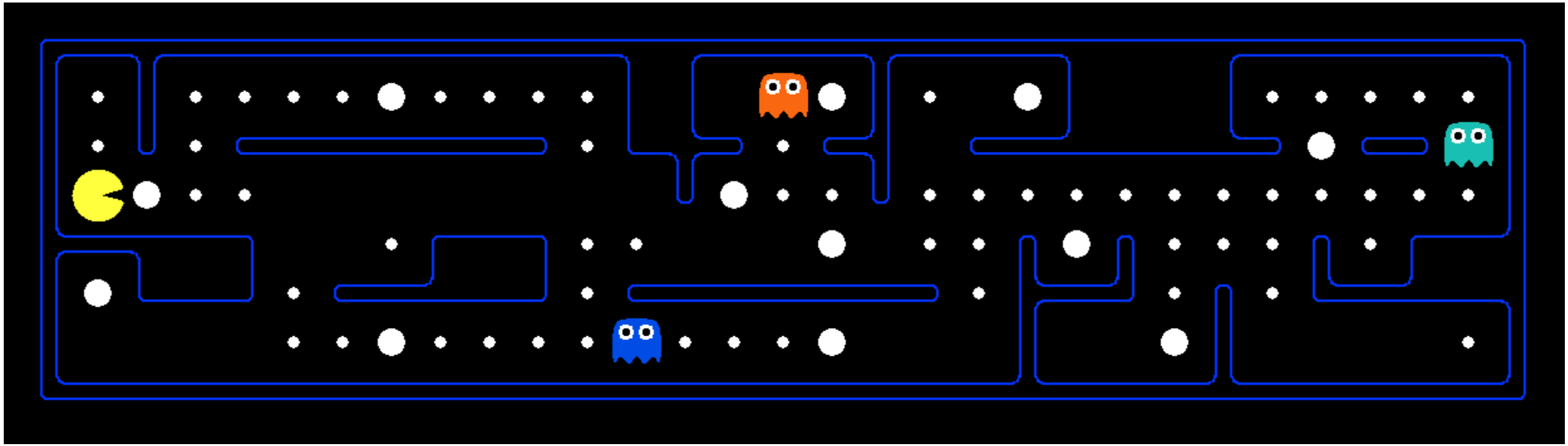
Veličine Prostora Stanja?

- Karateristike ove Pacman igre:
  - Moguće pozcije agenta: 120
  - Broj tačaka: 30
  - Pozicije duhova: 12
  - Orijentacija Pacmana: NSEW

- Koliko čega ima:
  - Broj stanja ove Pacman igre?
    $120x(2^{30})x(12^2)x4$
  - Broj stanja ako samo radimo određivanje putanje?
    120
  - Broj stanja ako nam je cilj da pojedemo sve tačke?
    $120x(2^{30})$

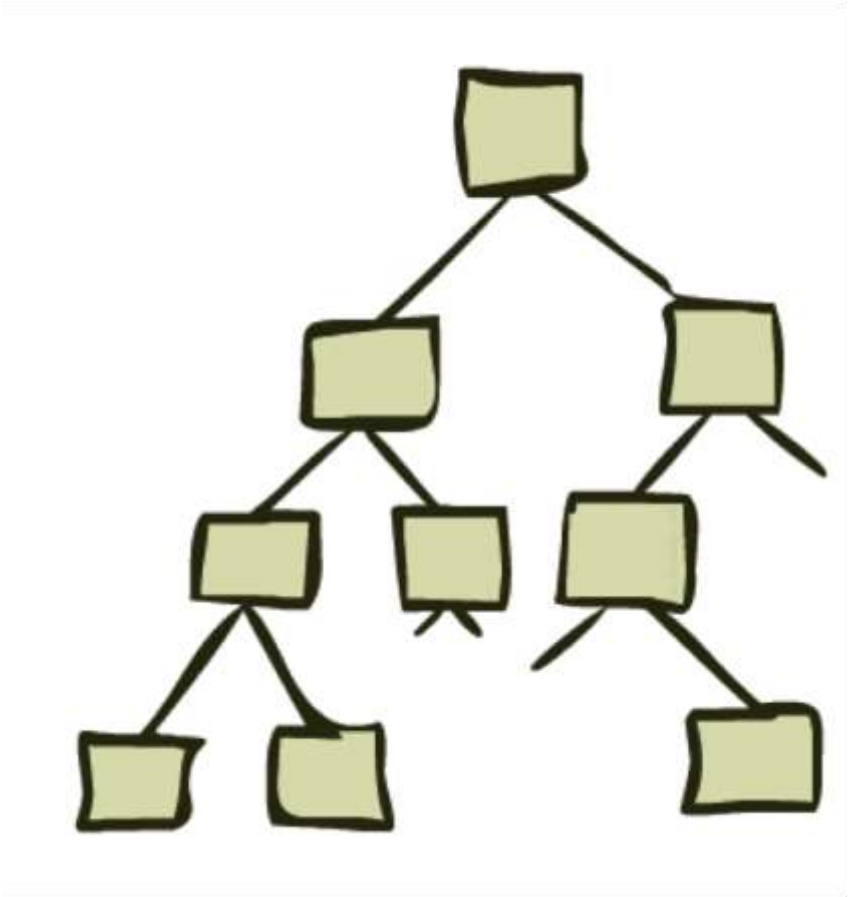


Siguran prolaz



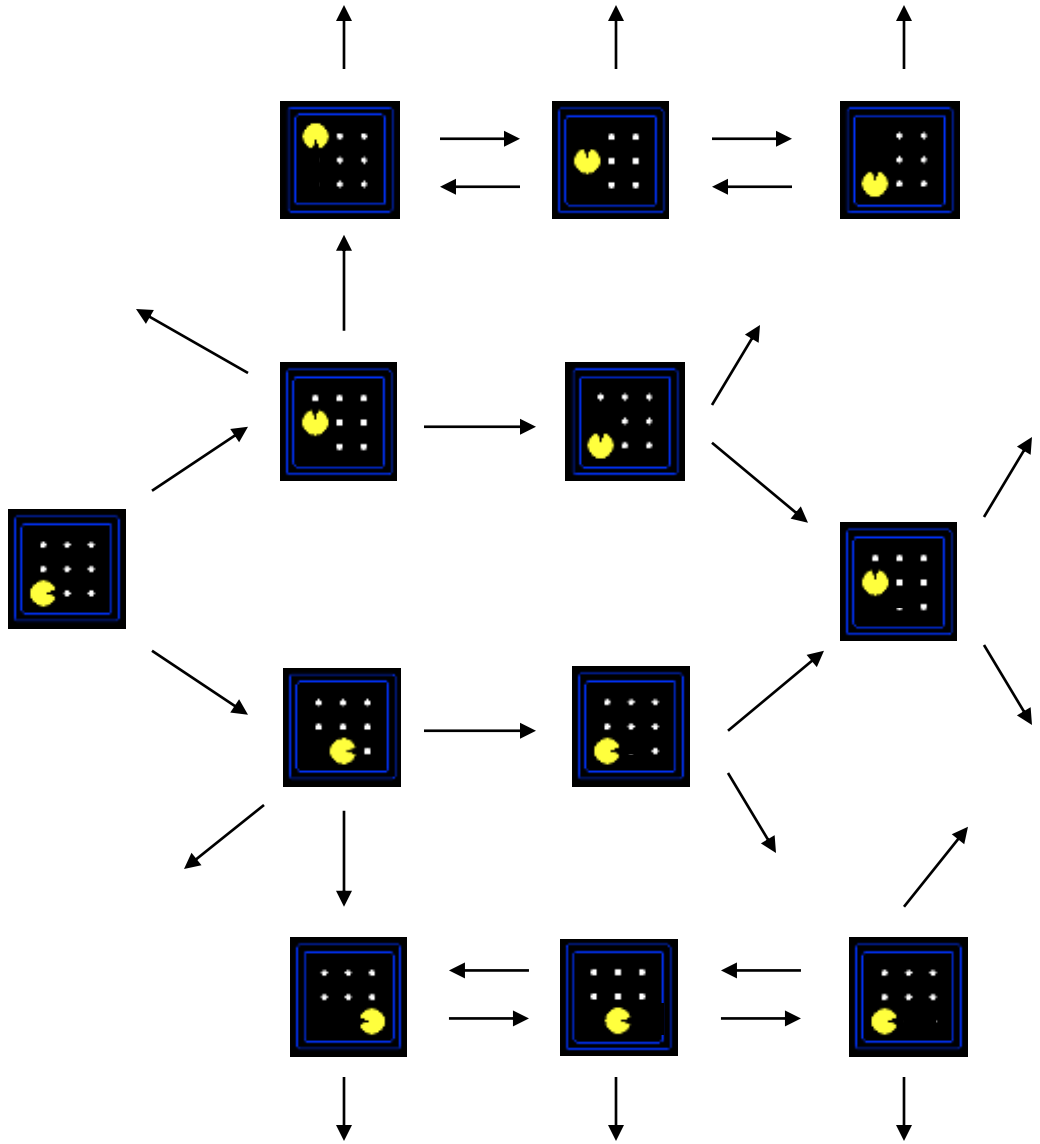
- Problem: pojesti sve tačke, ali tako da su duhovi stalno „uplašeni“
- Koje sve informacije moramo da imamo u stanju?
  - (pozciju agenta, booleane za tačke, booleane za velike tačke, vreme prestanka delovanja efekta velike tačke,...)

Graf Prostora Stanja i Stablo Pretraživanja



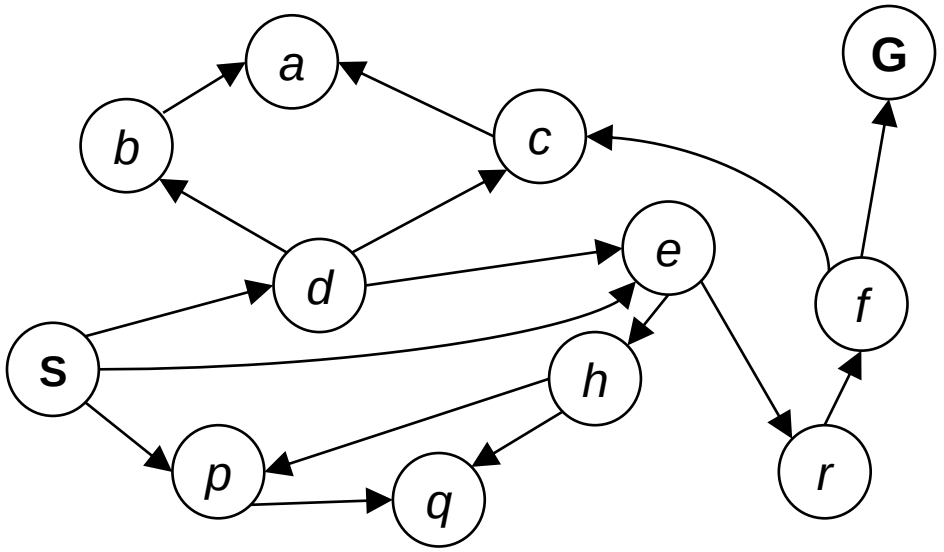
Grafovi Prostora Stanja

- Graf prostora stanja: Matematička reprezentacija problema pretraga
  - Čvorovi su reprezntacije stanja sveta u kome radimo pretragu.
  - Grane su moguća sledeća stanja (akcije).
  - Ciljno stanje je jedan ili više čvorova u grafu.
- U grafu prostora stanja svako stanje se pojavljuje samo jednom!
- Retko kad je moguće držati kompletan graf u memoriji, ali nam je koristan kao ideja.

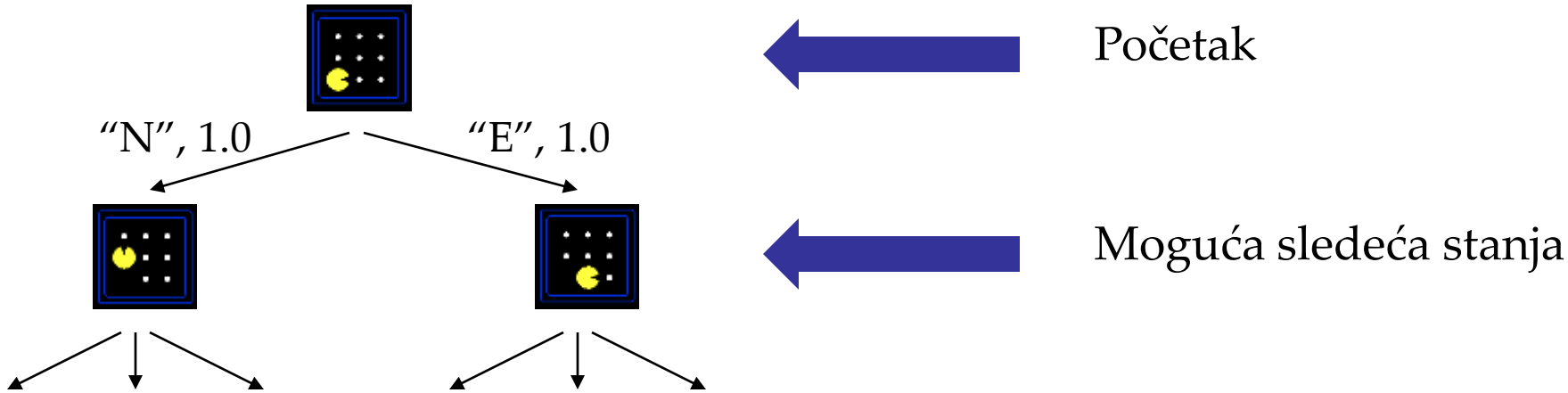


Grafovi Prostora Stanja

- Graf prostora stanja: Matematička reprezentacija problema pretraga
  - Čvorovi su reprezntacije stanja sveta u kome radimo pretragu.
  - Grane su moguća sledeća stanja (akcije).
  - Ciljno stanje je jedan ili više čvorova u grafu.
- U grafu prostora stanja svako stanje se pojavljuje samo jednom!
- Retko kad je moguće držati kompletan graf u memoriji, ali nam je koristan kao ideja.



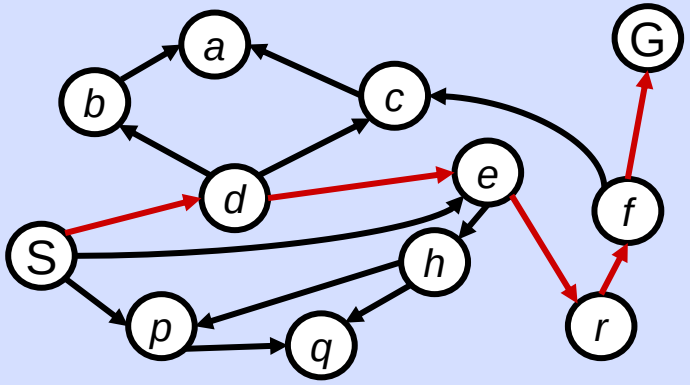
Stabla Pretraživanja



- Stablo pretraživanja:
  - Stablo prikazuje “šta bi bilo ako...” i uključuje planove i njihove ishode.
  - Početno stanje je koren.
  - Potomci su moguća sledeća stanja iz trenutnog
  - Čvorovi reprezentuju stanja sveta, ali ih treba posmatrati u kontekstu planova (isti čvor se može pojaviti više puta ako je rezultat različitih planova)
  - Za većinu problema ne možemo da izračunamo celo stablo pretraživanja

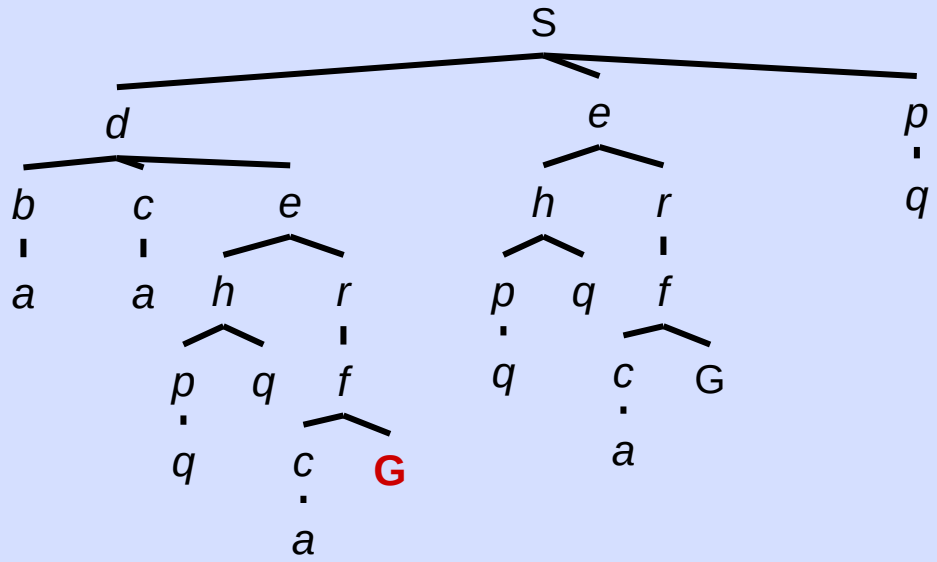
Graf Prostora Stanja vs. Stablo Pretraživanja

Graf Prostora Stanja



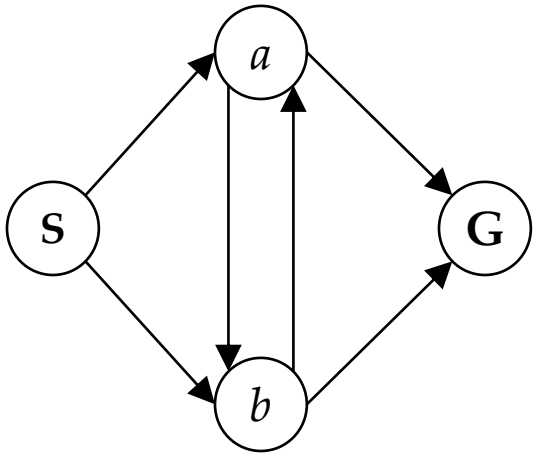
Obe strukture izračunavamo po potrebi i što manje moguće

Stablo Pretraživanja



Graf Prostora Stanja vs. Stablo Pretraživanja

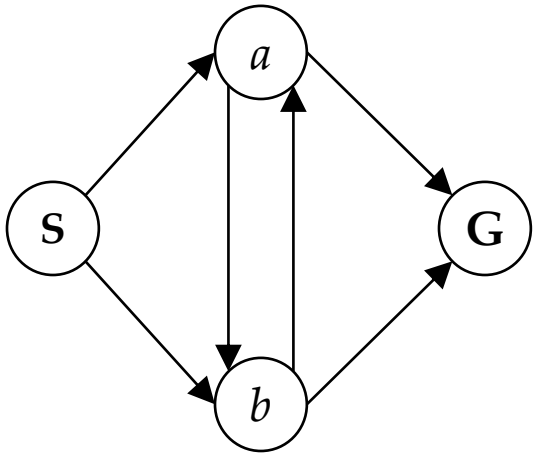
Recimo da imamo ovaj graf prostora stanja sa 4 čvora:



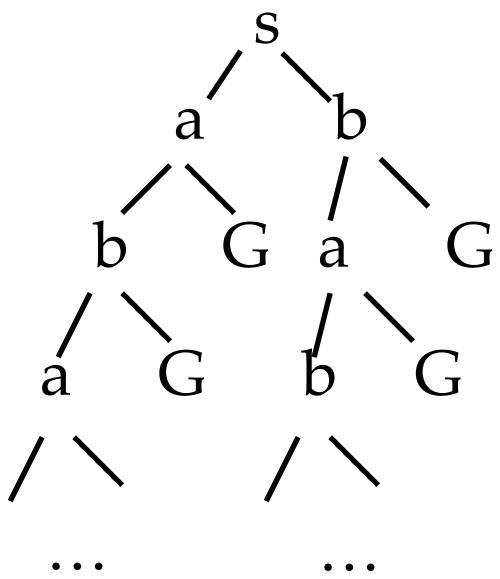
Ako krenemo iz S koliko je stablo pretraživanja?

Graf Prostora Stanja vs. Stablo Pretraživanja

Recimo da imamo ovaj graf prostora stanja sa 4 čvora:

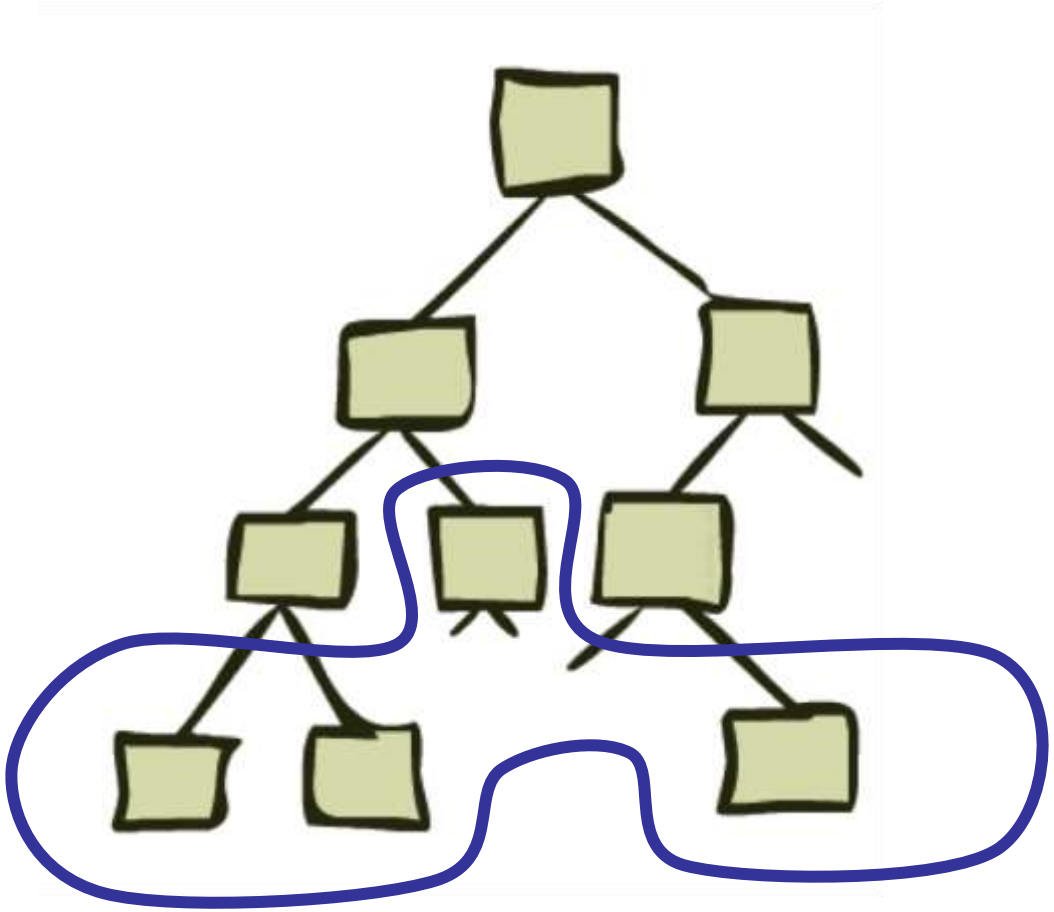


Ako krenemo iz S koliko je stablo pretraživanja?

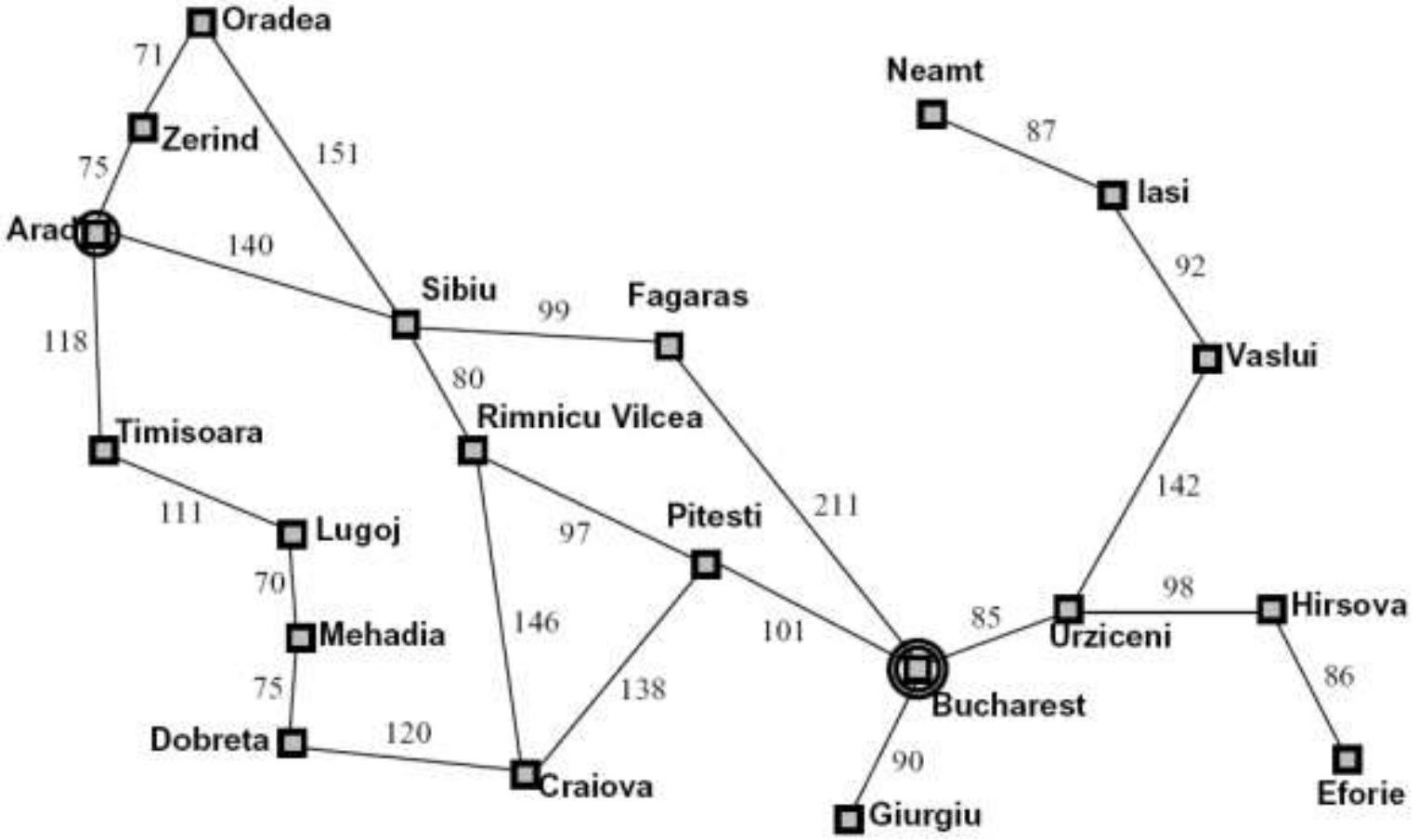


Jako puno ponavljanja u stablu pretraživanja!

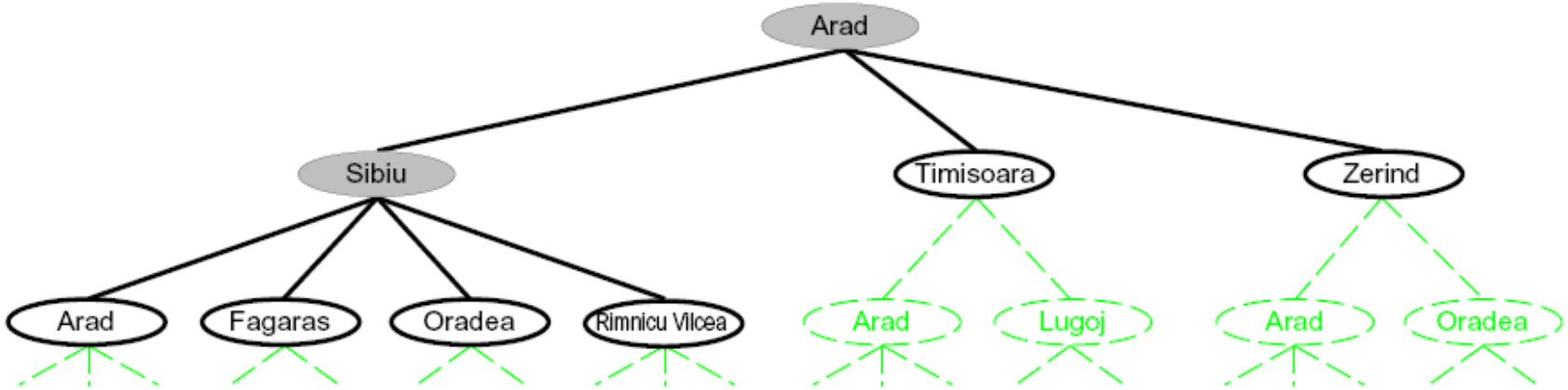
Pretrage



Primer: Putovanje u Rumuniji



Na koji način koristimo stablo pretraživanja



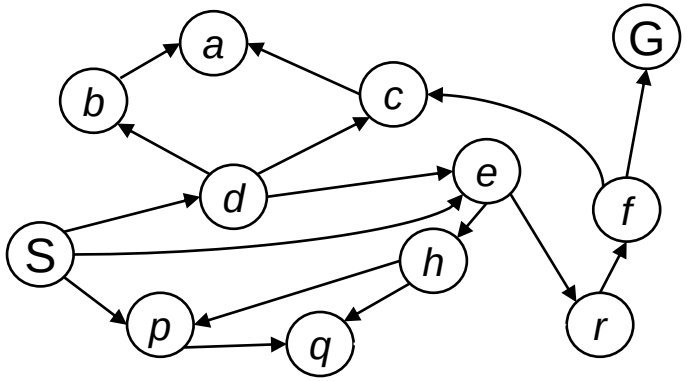
- Pretraga:
  - Razvijamo stablo korak po korak
  - Održavamo strukturu sa čvorovima koje još nismo razvili
  - Trudimo se da razvijamo što je manje moguće

Opšti algoritam za pretrage

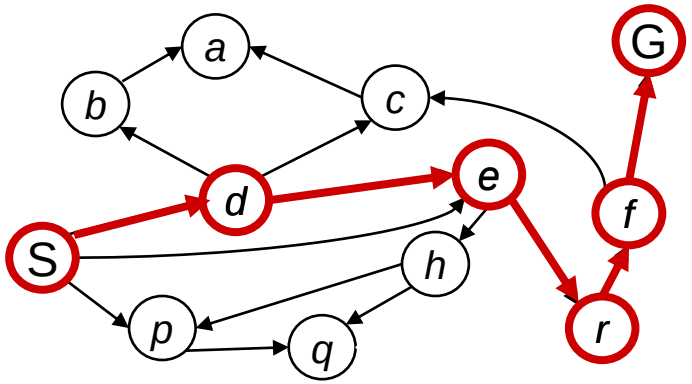
```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
    end
```

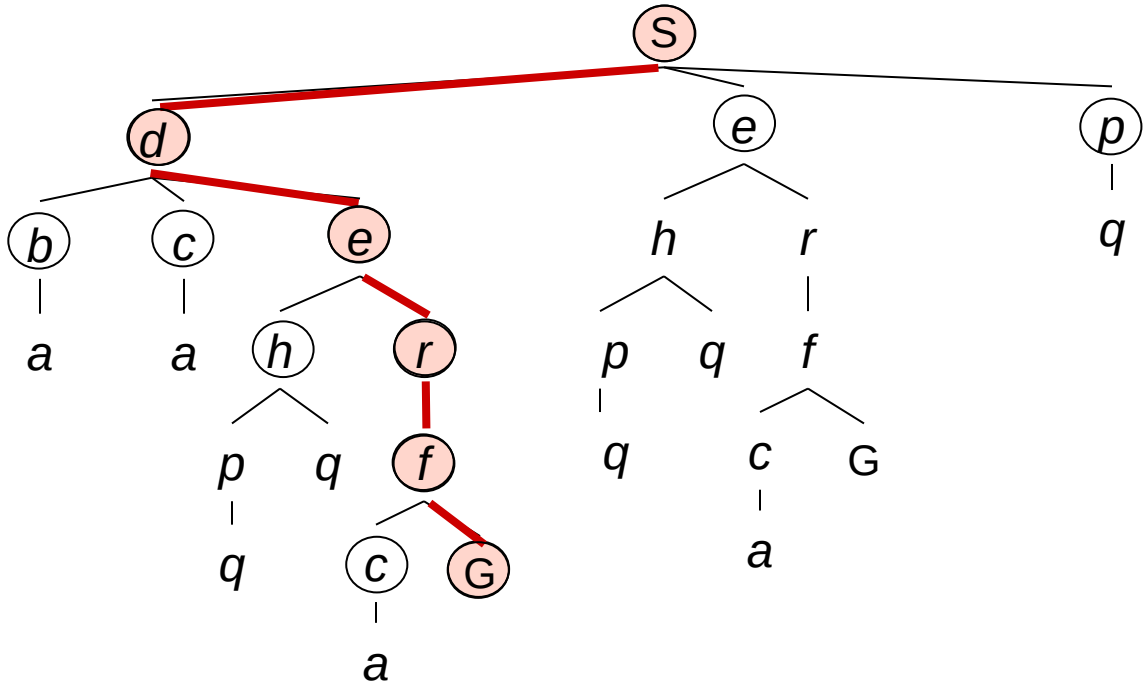
- Važni koncepti:
  - Struktura za čuvanje čvorova koje još nismo razvili
  - Razvijanje čvora (*Expansion*)
  - Strategija po kojoj razvijamo čvorove (*Exploration strategy*)
- Važno pitanje: na koji način biramo čvor koji ćemo sledeći da razvijemo?

Primer: Pretrage



Primer: Pretrage





~~s~~

~~s → d~~

s → e

s → p

s → d → b

s → d → c

~~s → d → e~~

s → d → e → h

~~s → d → e → r~~

~~s → d → e → r → f~~

s → d → e → r → f → c

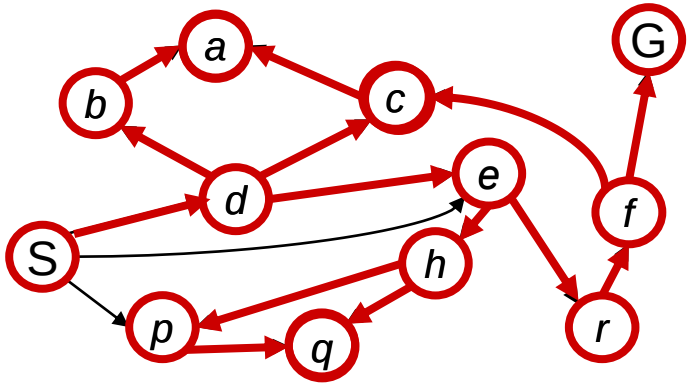
~~s → d → e → r → f → G~~

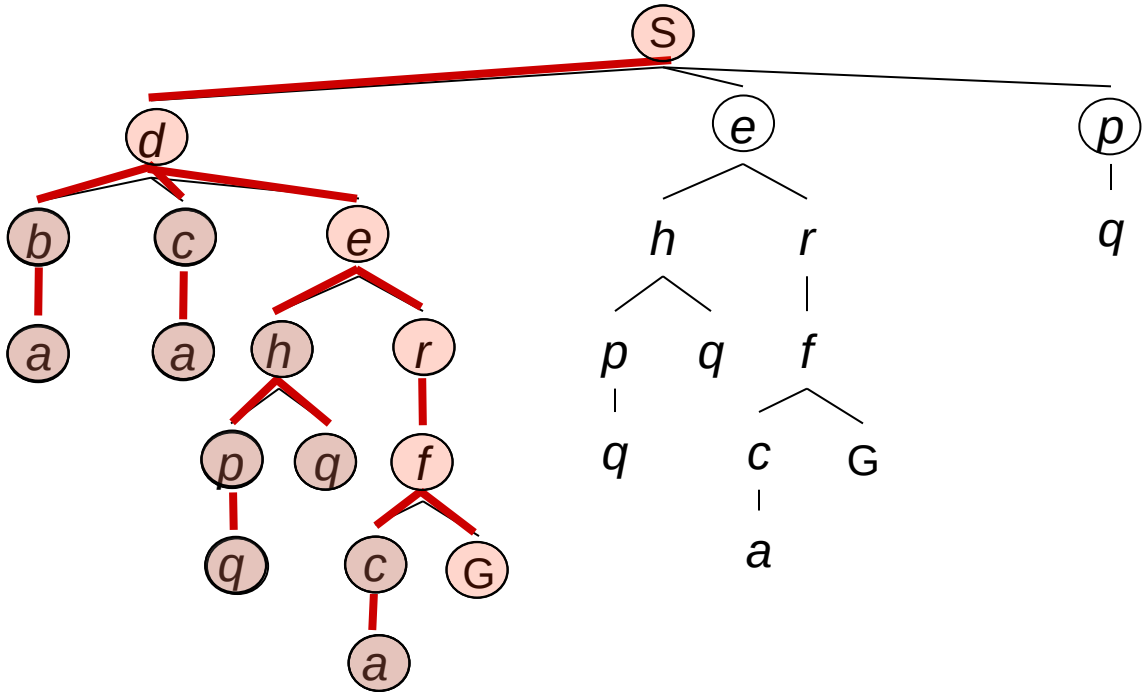
Prvi Po Dubini - *Depth-First Search, DFS*

Depth-First Search

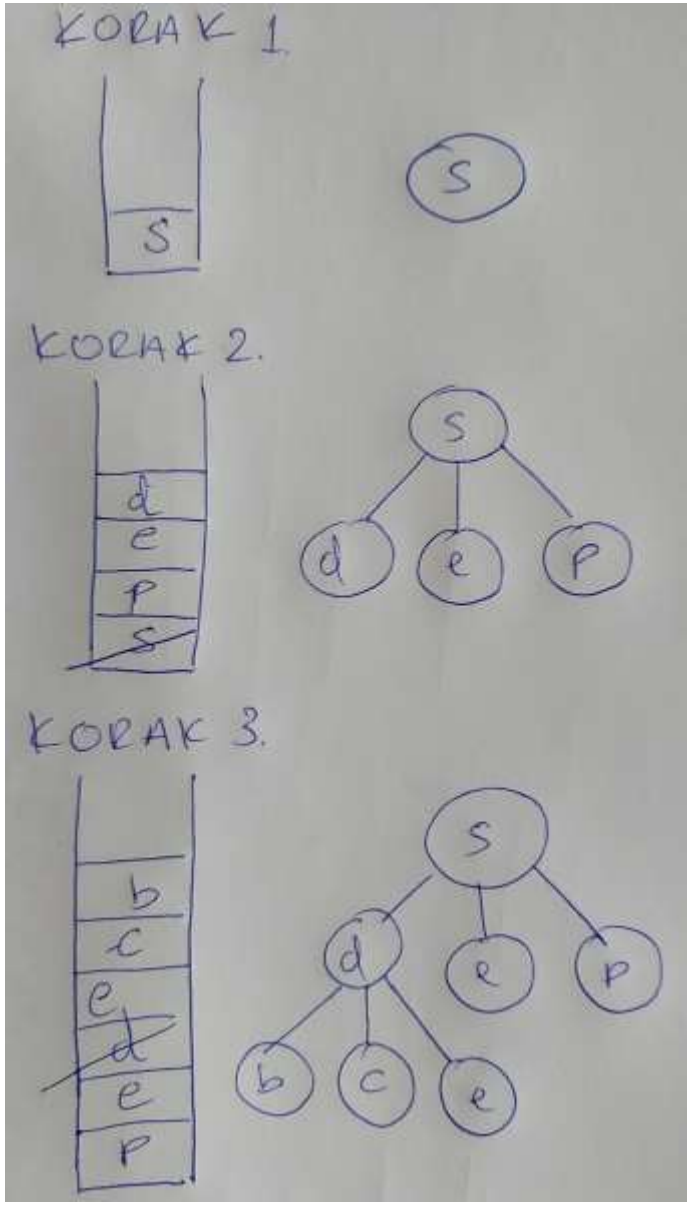
Stragedija: razvijamo prvo čvorove na najvećoj dubini

Implentacija: Struktura koju koristimo je: stek tj. LIFO lista.

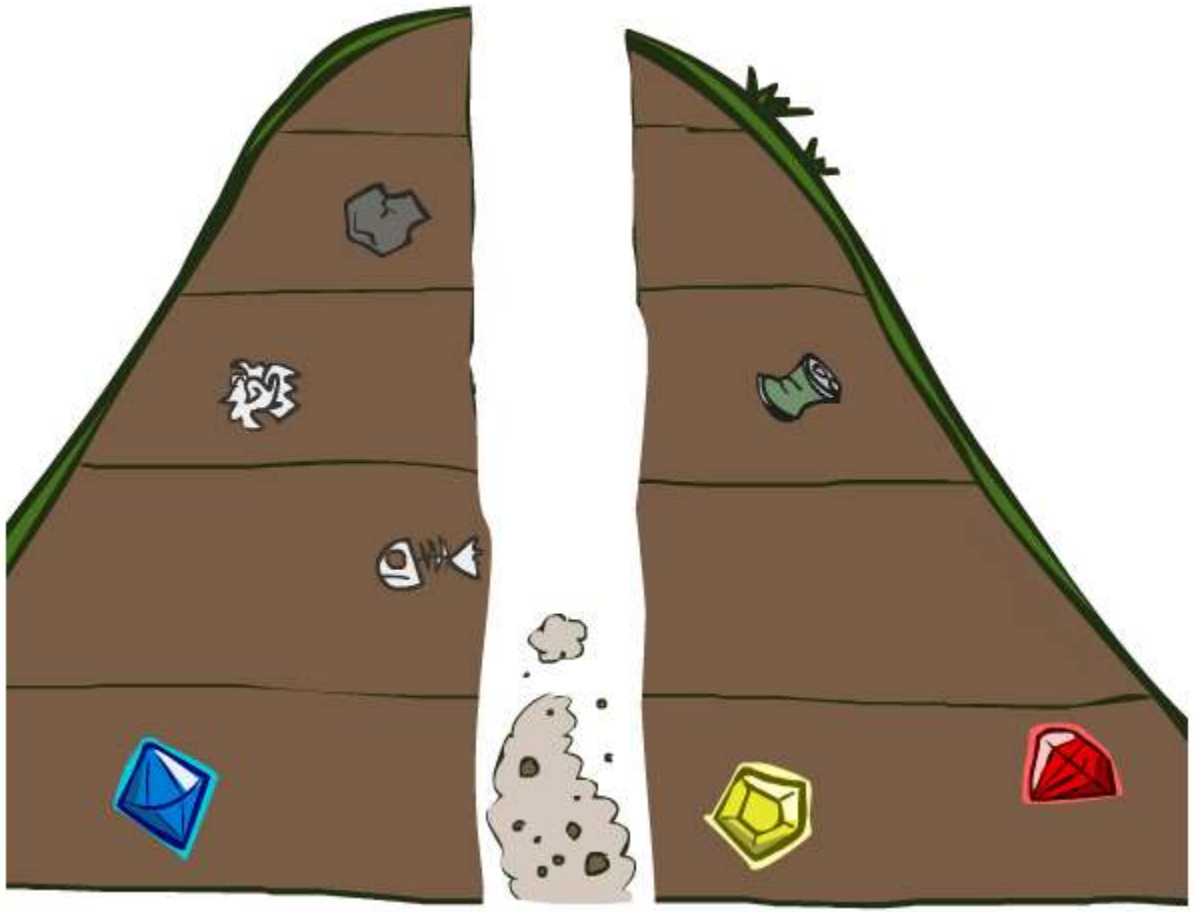




DFS - pojašnjenje



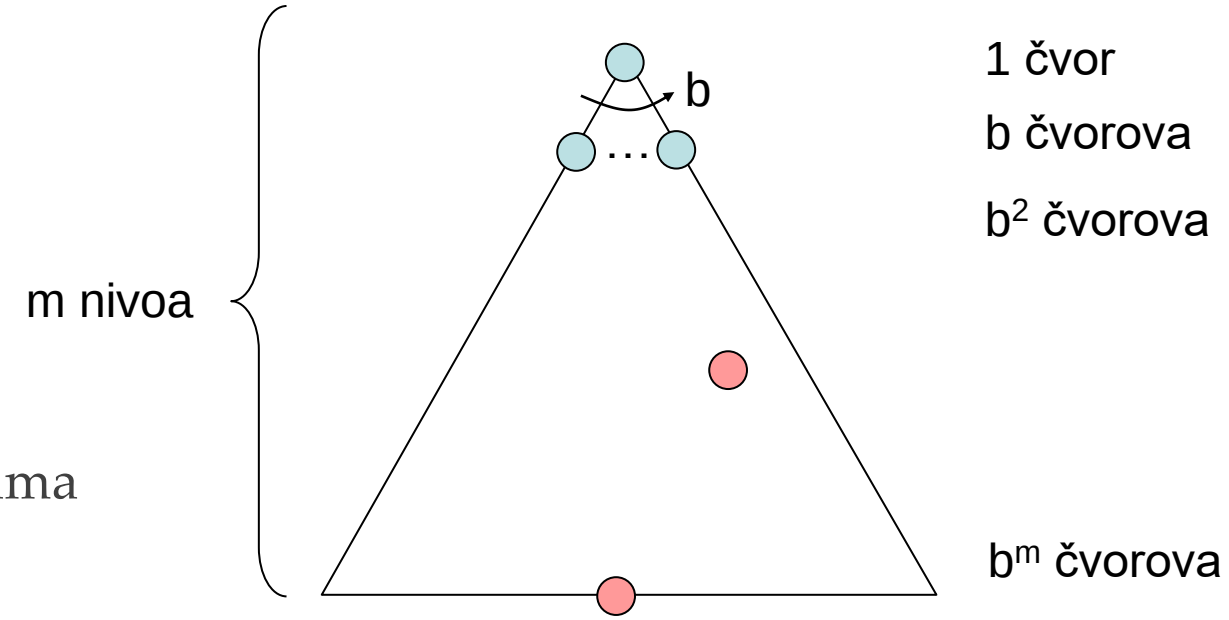
Karakteristike Algoritama Za Pretrage



Karakteristike Algoritama Za Pretrage

- Kompletnost: Garantovano pronalazi rešenje ako postoji bar jedno?
- Optimalnost: Garantovano pronalazi najbolje rešenje?
- Vremenska složenost?
- Memorijska složenost?

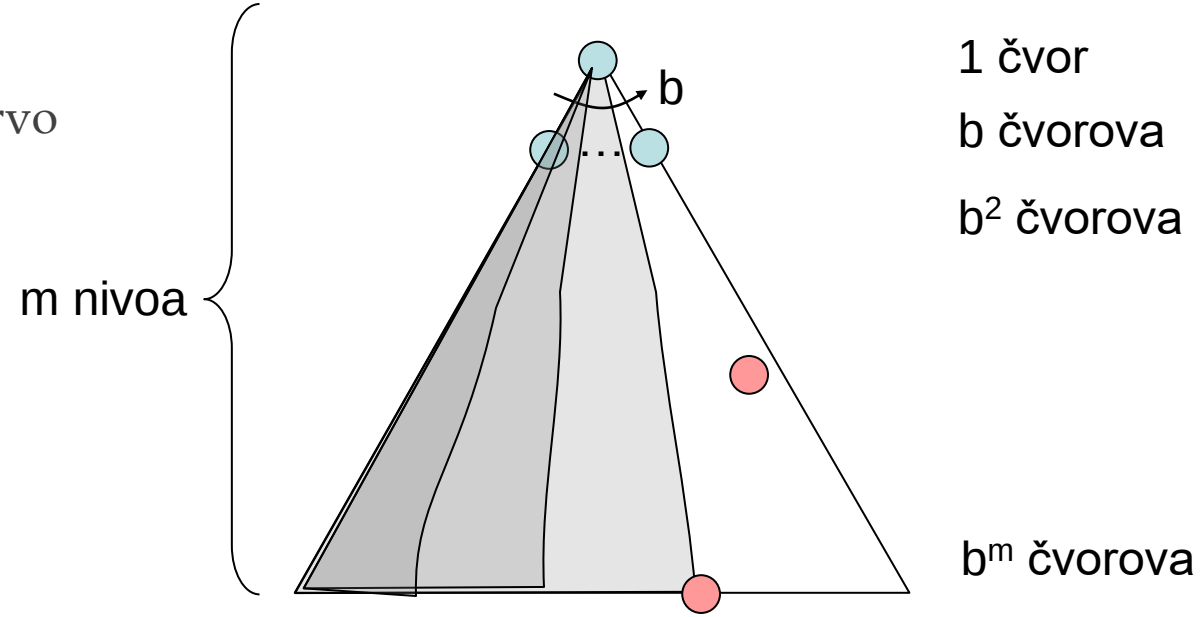
- Crtež stabla pretraživanja:
  - b je faktor grananja
  - m je maksimalna dubina
  - Imamo rešenja na različitim dubinama



- Broj čvorova u celom stablu?
  - $1 + b + b^2 + \ldots. b^m = O(b^m)$

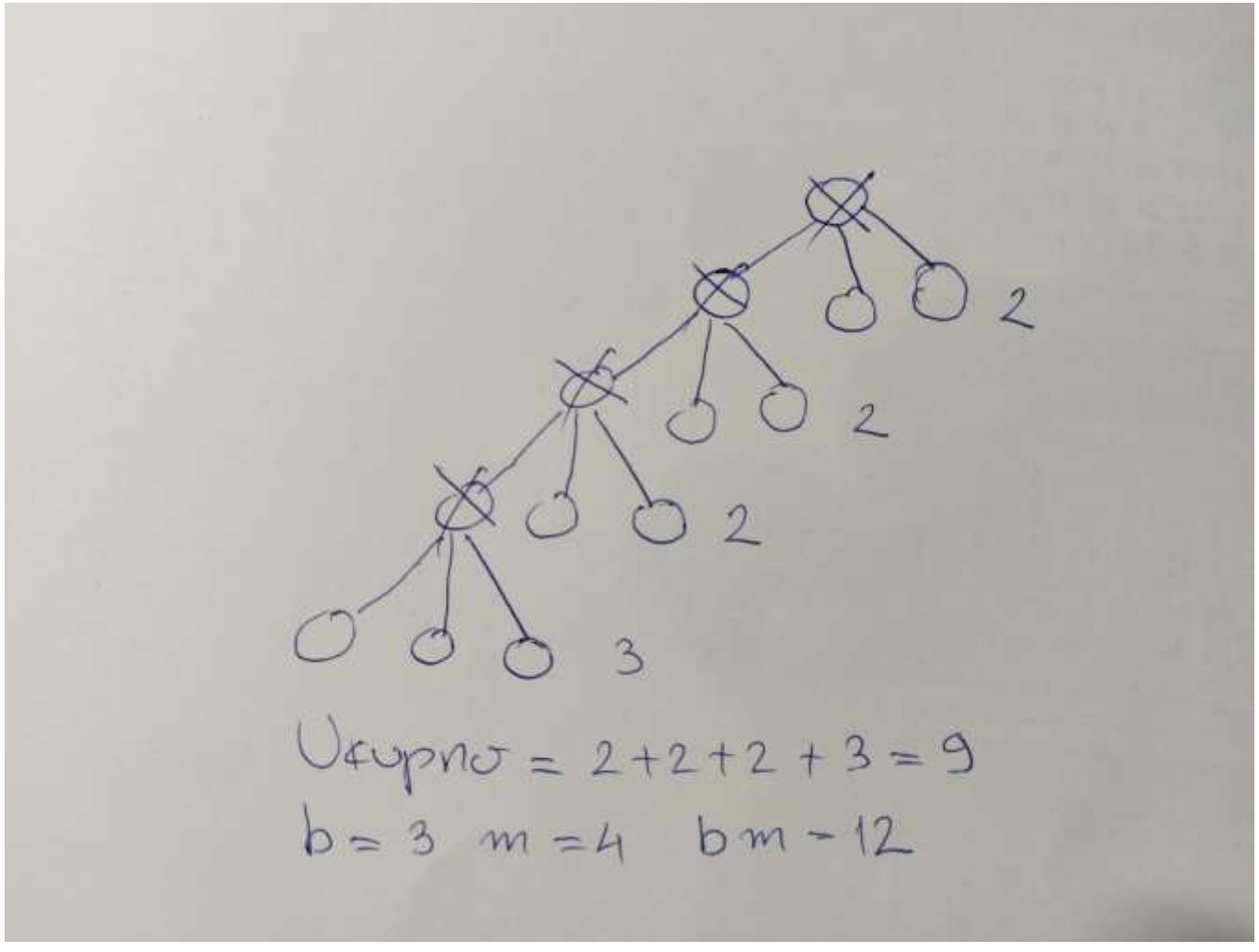
Karakteristike DFS algoritma

- Koje čvorove razvija DFS?
  - Neki levi deo stabla (ako preferiramo da razvijamo prvo leve čvorove).
  - Moguće je da razvije celo stablo!
  - Ako je m konačno, vremenska složenost je $O(b^m)$
- Kolika je prostorna složenost steka?
  - U odnosu na trenutni čvor čuvamo samo njegove roditelje (i „braću ili sestre ☺“), tako da je O(bm)
- Da li je kompletan?
  - m može da bude beskonačno, tako da je DFS kompletan samo ako sprečavamo petlje u stablu
- Da li je optimalan?
  - Ne, pronalazi rešenje koje je “najviše levo” bez obzira na dubinu ili cenu

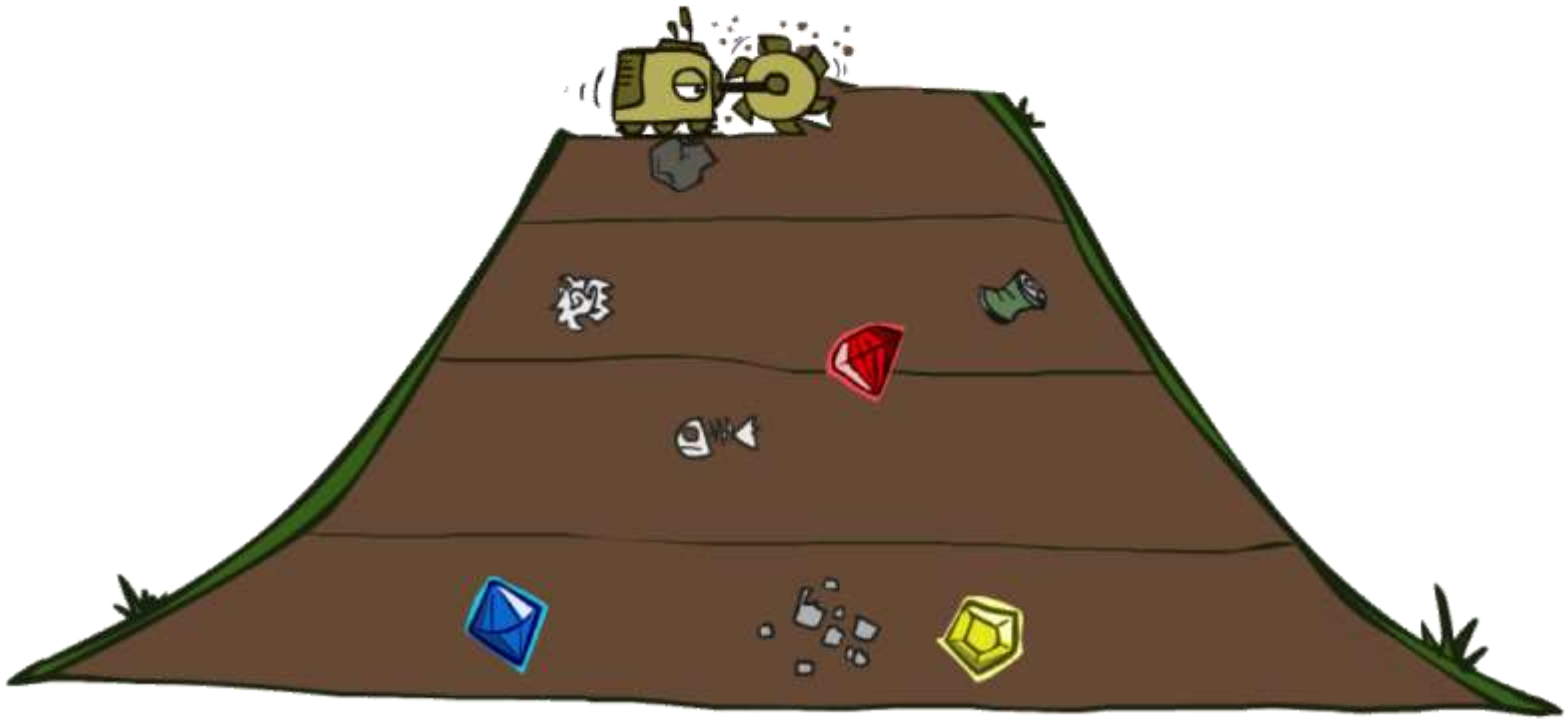


Prostorna složenost - pojašnjenje

- Precrtani čvorovi su skinuti sa steka pre nego što su njihovi potomci dodati na stek.



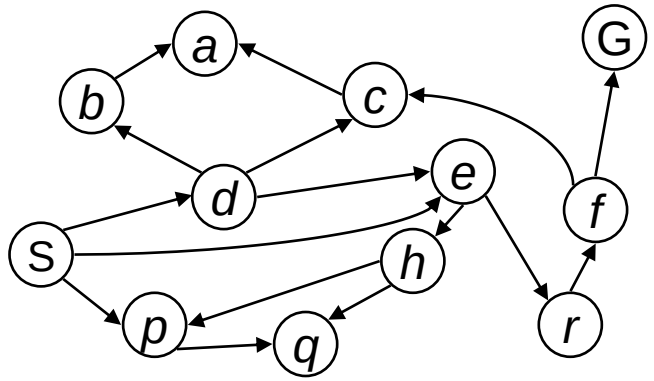
Prvo Po Širini - Breadth-First Search, BFS



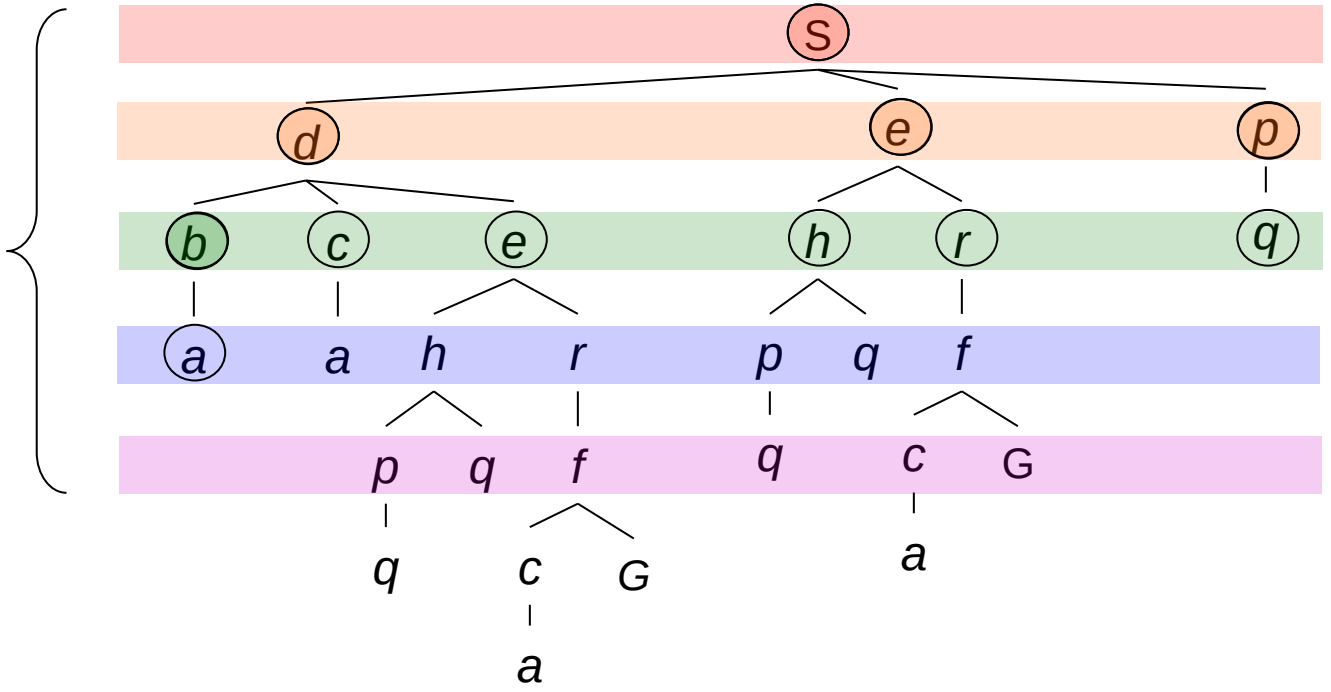
Breadth-First Search

Strategija: razvijamo prvo sve čvorove na istoj dubini.

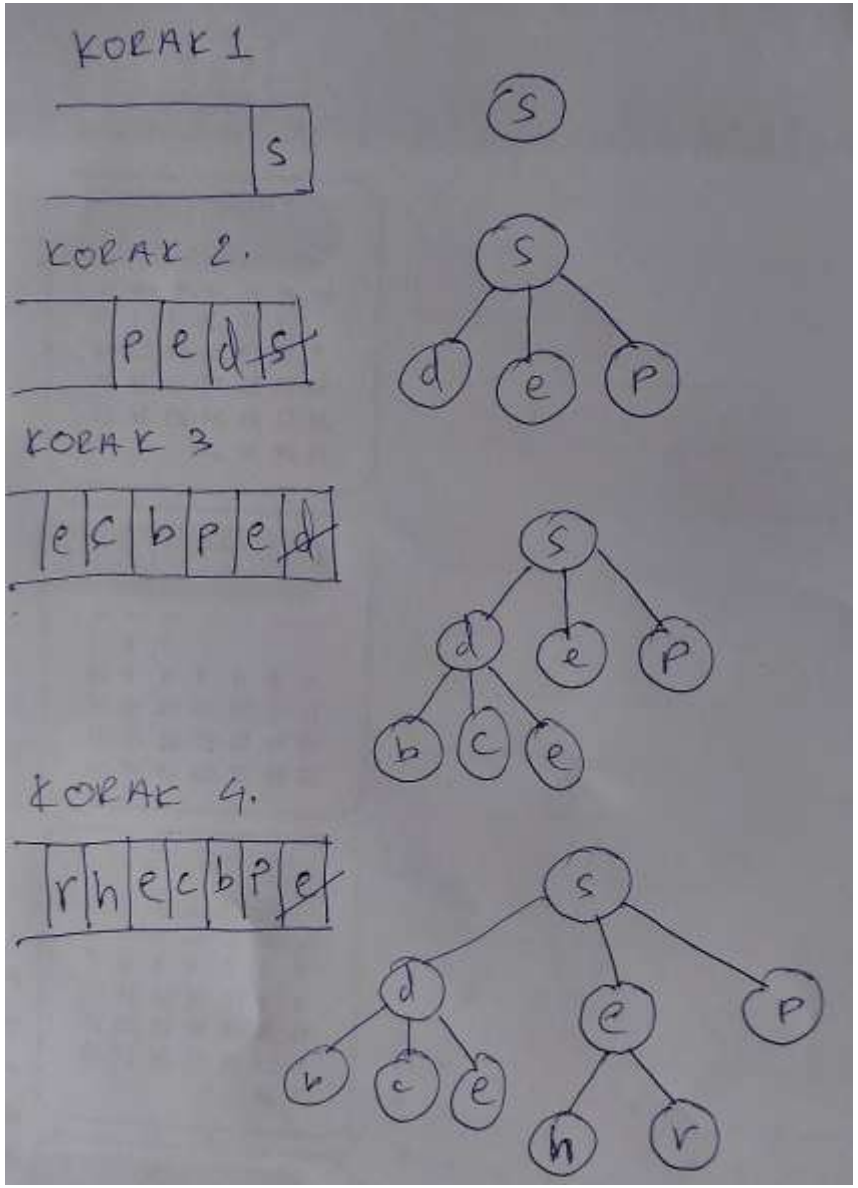
Implementacija: Struktura koju koristimo je: red tj. FIFO lista



Nivoi pretrage

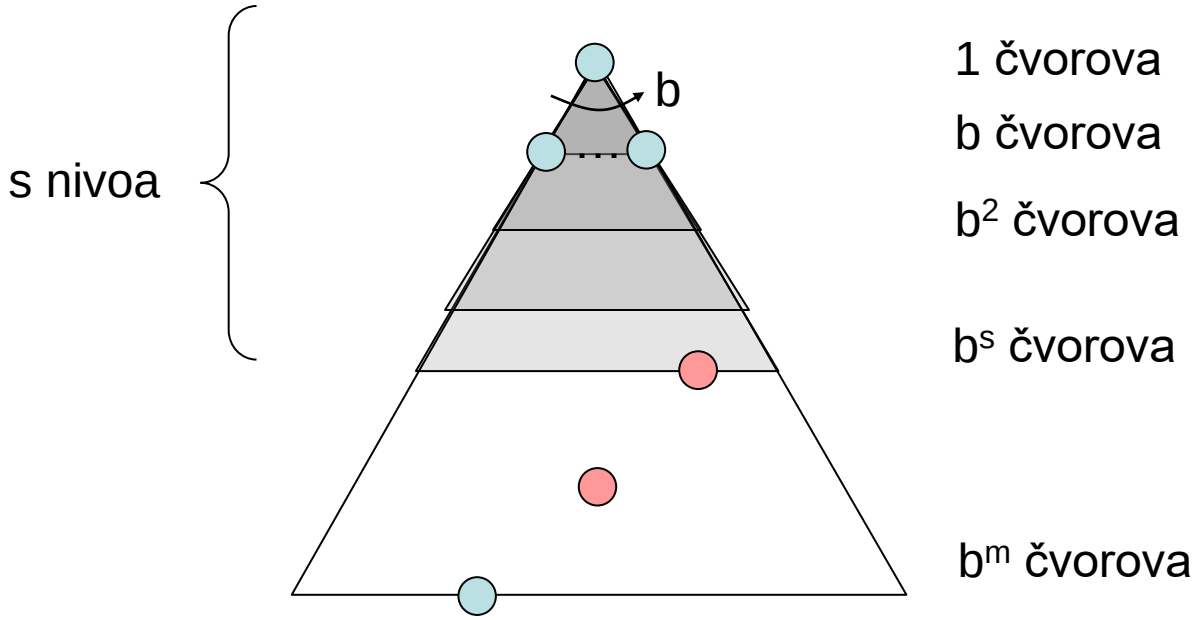


BFS – pojašnjenje

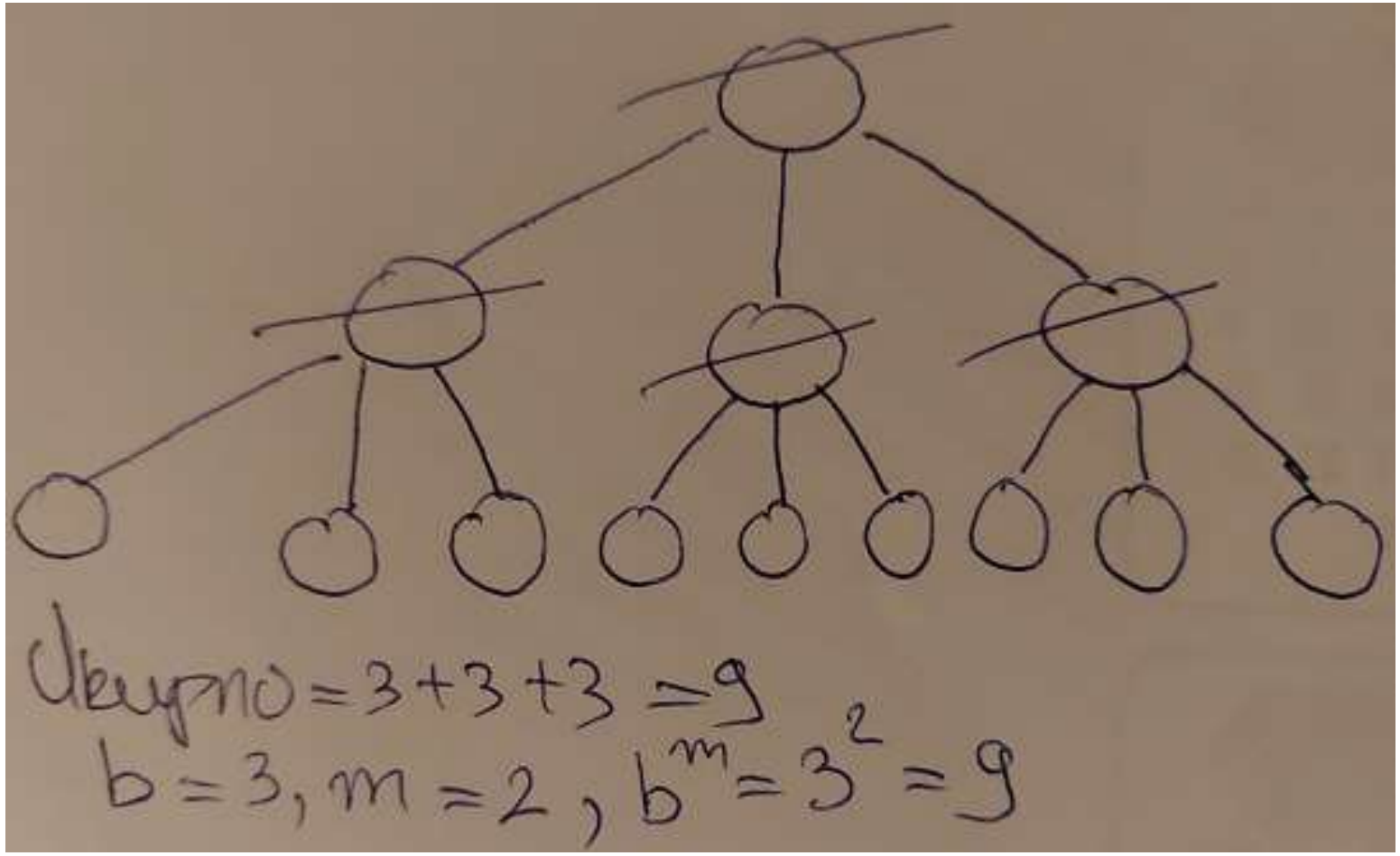


Karakteristike BFS algoritma

- Koje čvorove razvija BFS?
  - Sve čvorove na istom nivou tj. ako je s najbliže (najpliće) rešenje BFS će razivti sve čvorove iznad njega dok ne dođe do njega.
  - Vremenska složenost $O(b^s)$
- Koliko prostora zauzima u FIFO?
  - Otprilike onliko koliko ima na nivou do kog je stigao tj. $O(b^s)$ – (broj čvorova iznad za veliku vrednost s je zanemarljiv).
- Da li je kompletan?
  - Da, ako rešenje postoji na nekom konačnom nivou, BFS će ga pronaći!
- Da li je optimalan?
  - Da ako su cene svih akcija iste (npr. 1) – više o cenama kasnije.



Prostorna složenost - pojašnjenje

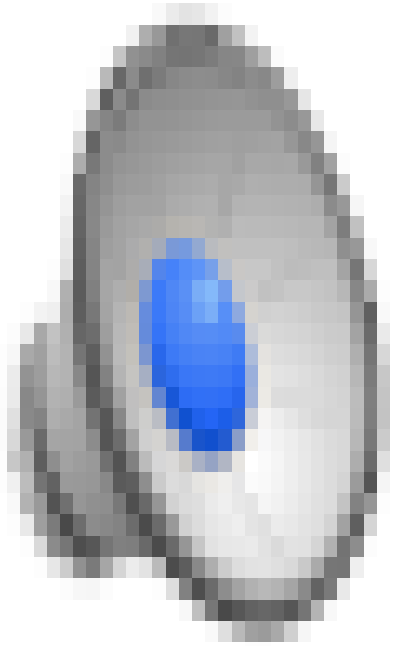


DFS vs BFS

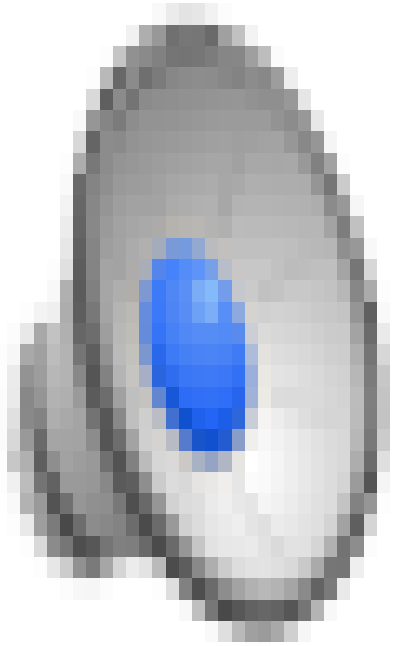
- Kada je BFS bolji od DFS?

- Kada je DFS bolji od BFS?

Demo Maze Water DFS/BFS (deo 1)



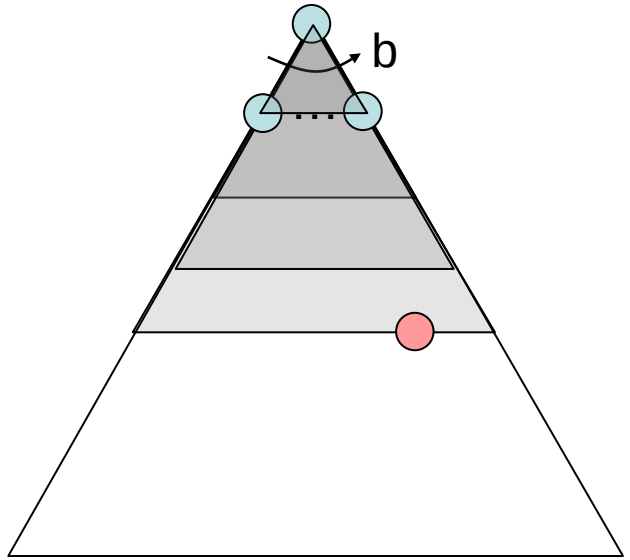
Demo Maze Water DFS/BFS (deo 2)



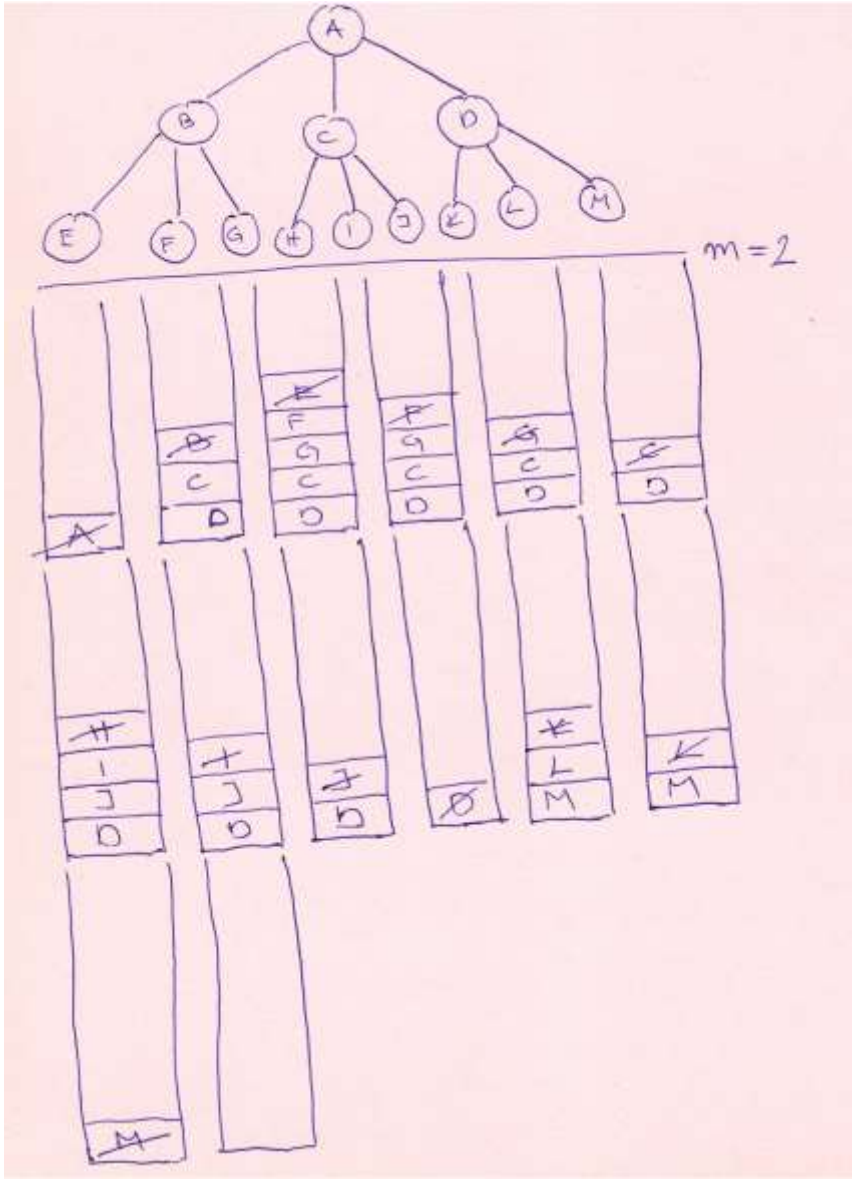
Iterativno Produbljivanje – Iterativno Prvo u Dubninu

- Ideja: spojiti malo zauzeće memorije DFS sa optimalnošću BFS.
  - Stratujemo DFS do dubine 1. Ako ne pronađemo rešenje…
  - Stratujemo DFS do dubine 2. Ako ne pronađemo rešenje…
  - Stratujemo DFS do dubine 3. Ako ne pronađemo rešenje…
- Da li gubimo vreme na ponavljanjima?
  - Generalno najviše razvijanja biće na jako dalekim nivoima tako da ponavljanje iznad nije strašno!
  - Vremenska složenost (*s* je nivo na kome je rešenje):

$1 + (1+b) + (1+b+b^2) + \ldots + (1+b+\ldots+b^s) = (s+1) + sb + (s-1)b^2 + .. + b^s = O(b^s)$

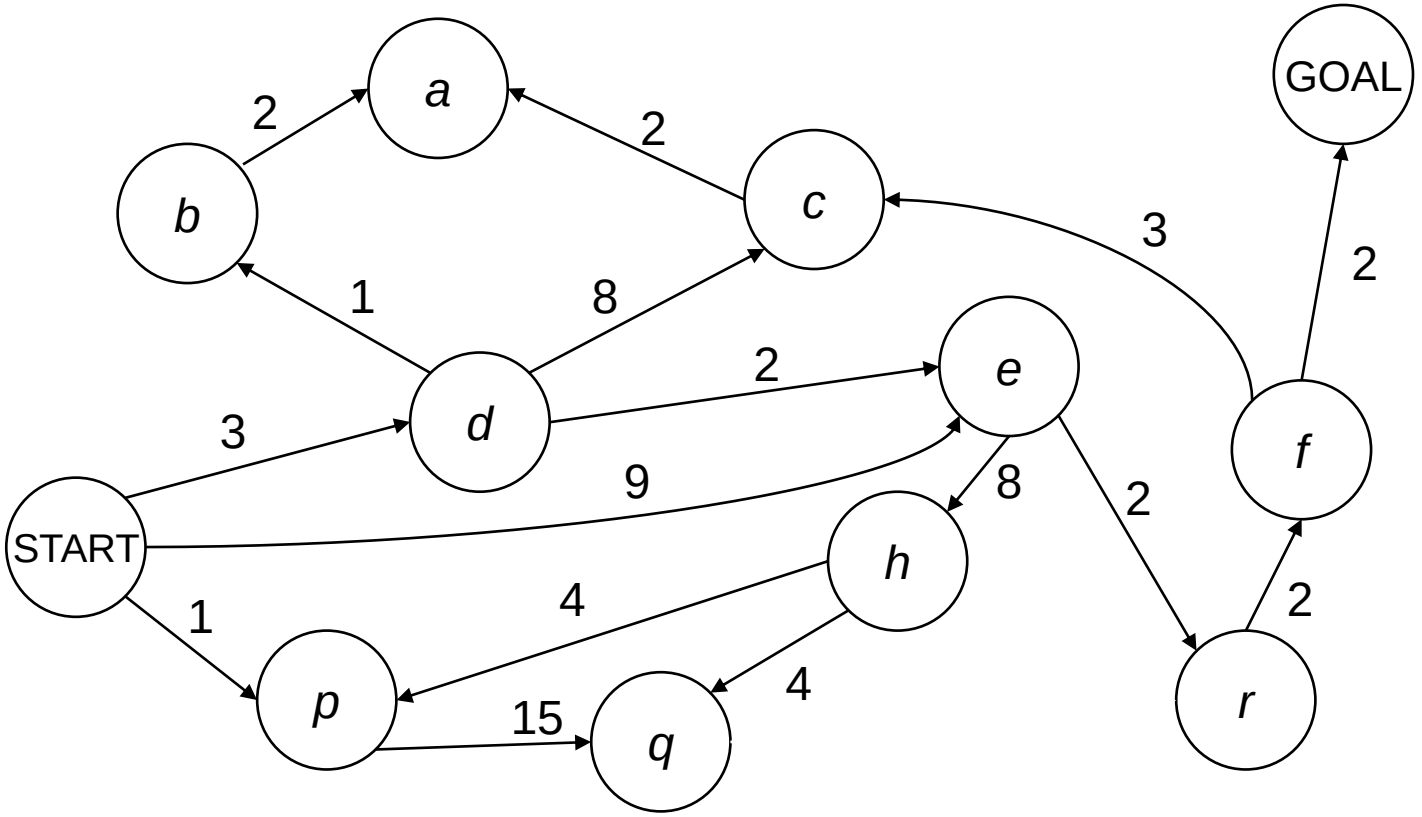


Prostorna složenost - pojašnjenje



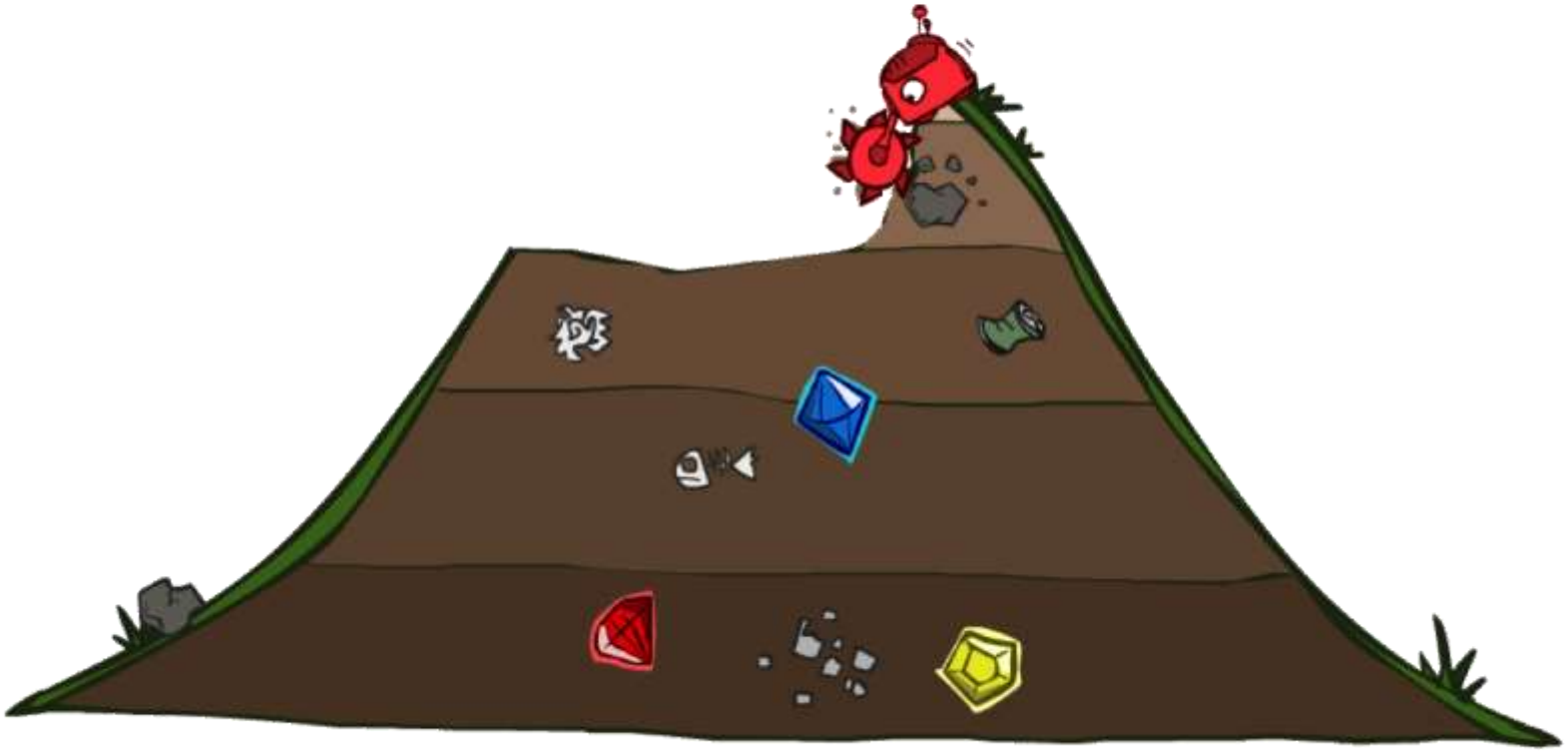
- Dat je primer pokretanja itreativnog DFS za zadatu makismalnu dubinu m=2.
- Iako je nacrtano celo stablo, ono se se ne razija celo odjednom.
- Ispod stabla prikazano je stanje steka u svakom trenutku.
- Praćenjem stanja steka možete videti kako je stablo razvijano.
- U svakom trenutku na steku nije bilo više od O(bm) čvorova, gde je b=3, a m=2.
- Ali je algoritam obišao sve čvorove do zadate dubine m=2.
- To znači da imamo prednosti:
  - BFS (obišli smo sve čvorove pre pelaska na sledeći nivo)
  - DFS (memorijsko zauzeće nije prelazilo O(bm).

Pretraga Sa Cenama
Cost-Sensitive Search



BFS pronalazi najkraći put u smislu broja akcija.
BFS ne nalazi najkraći put u smislu najmanje cene. Sada ćemo prikazati algoritam koji radi po sličnom principu, ali pronalazi „najjeftniji“ put.

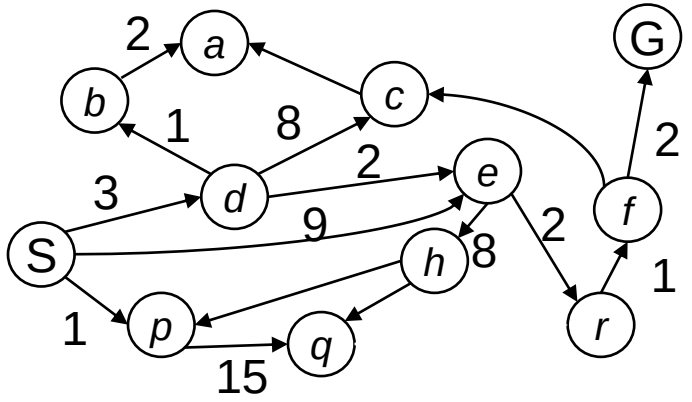
Algoritam: Uniform Cost Search, UCS

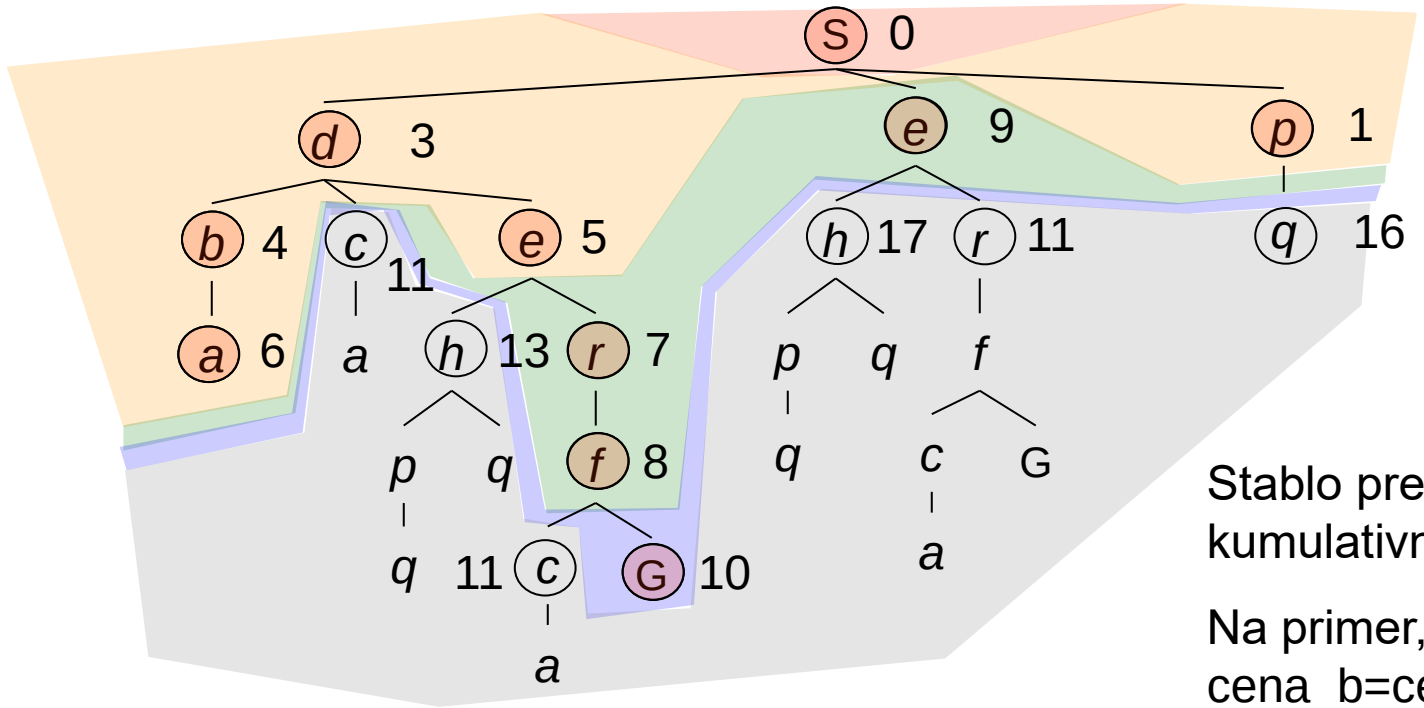


Uniform Cost Search

Strategija: razvijamo čvor koji ima najmanju cenu:

Struktura je red sa prioritetom (prioritet: kumulativna cena tj. cena tog čvora + cene svih čvorova na putu do njega)

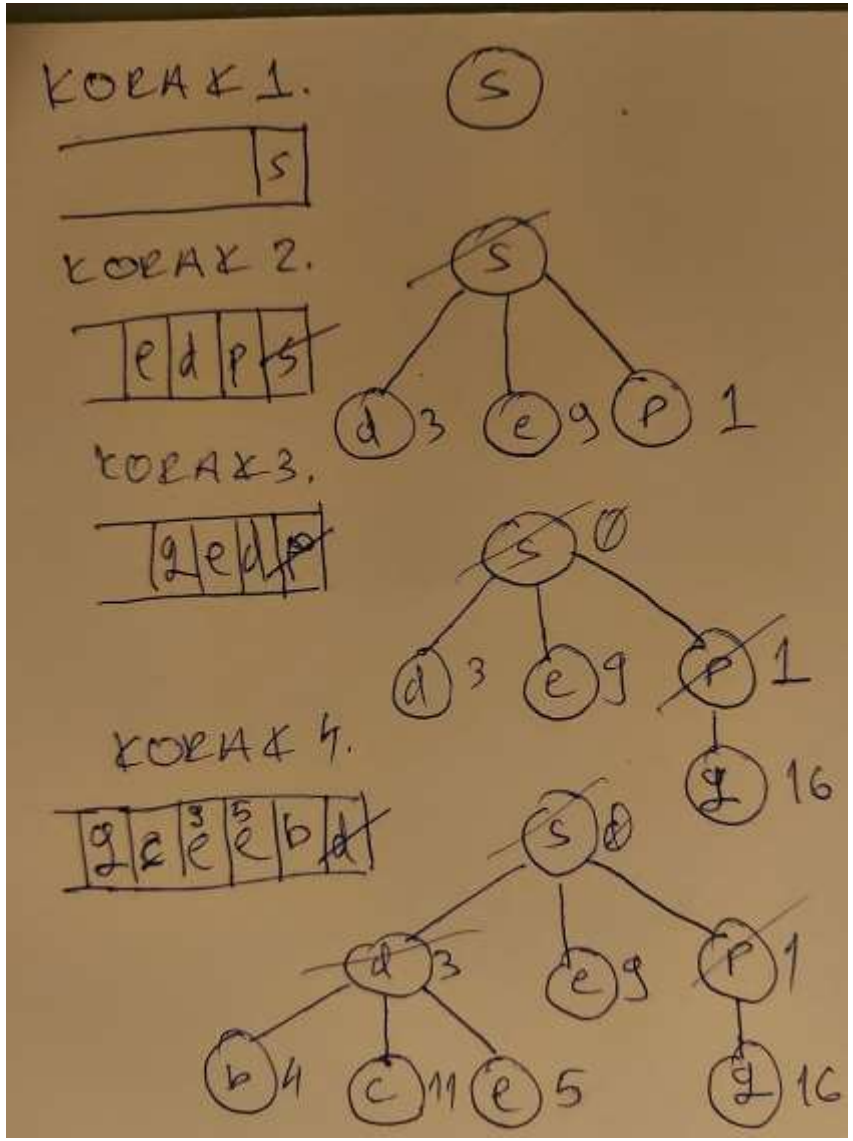




Stablo pretraživanja sadrži kumulativne cene pored svakog čvora.

Na primer, za čvor b imamo cena_b=cena(S,d)+cena(d,b)=3+1=4

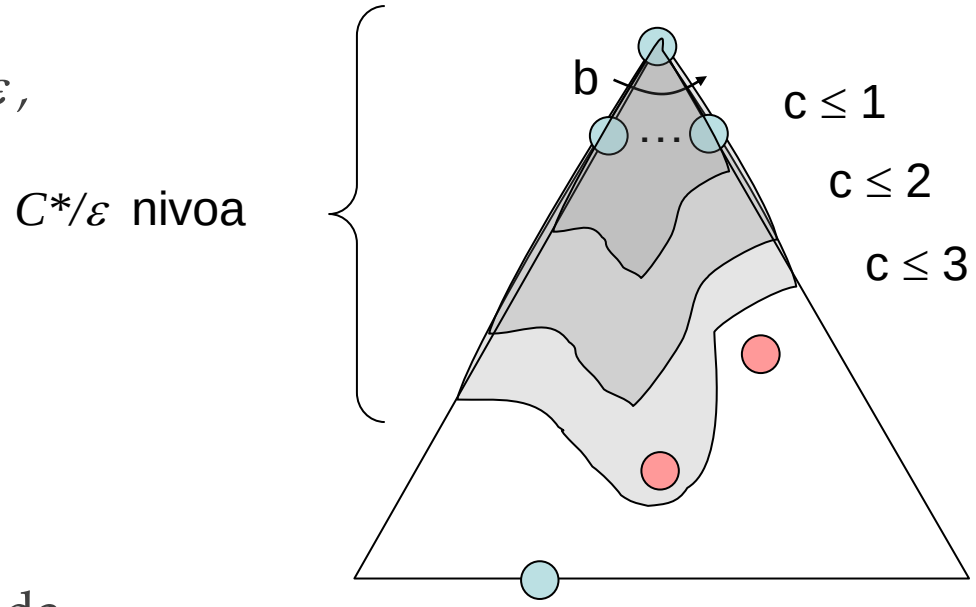
USC - pojašnjenje



- Struktura koja se koristi je red prioriteta.
- Čvor koji ima najmanju cenu je prvi koji je na redu za obradu.
- U kontekstu ovog kursa, način na koji je implementiran red piroriteta nema značaja.

Karakteristike Uniform Cost Search (UCS) algoritma

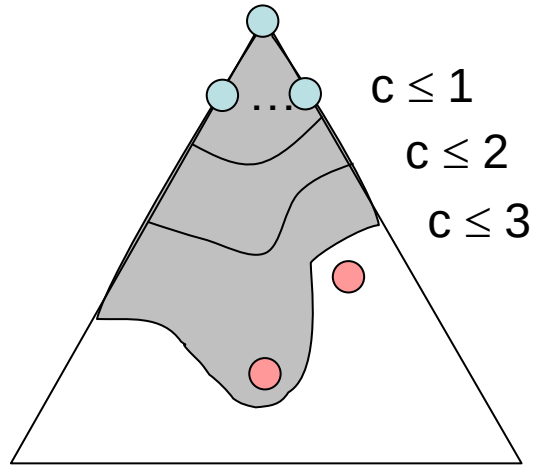
- Koje čvorove UCS razvija?
  - Razvija sve čvorove sa cenom manjom od najjeftinijeg rešenja!
  - Ako to rečenje ima cenu C^* i cene akcija su najmanje ε, onda UCS stiže dubine otprilike C^*/ε
  - Vremenska kompleksnost je $O(b^{C^*/\varepsilon})$
- Koja je prostorna složenost?
  - Otprilike poslednji sloj je u redu prioriteta pa $O(b^{C^*/\varepsilon})$
- Da li je kompletan?
  - Da, uz uslov da bar jedno rešenje ima konačnu cenu i da nemamo negativne cene!
- Da li je optimalan?
  - Da! (Dokaz je povezan sa dokazom za optimalnost A*)

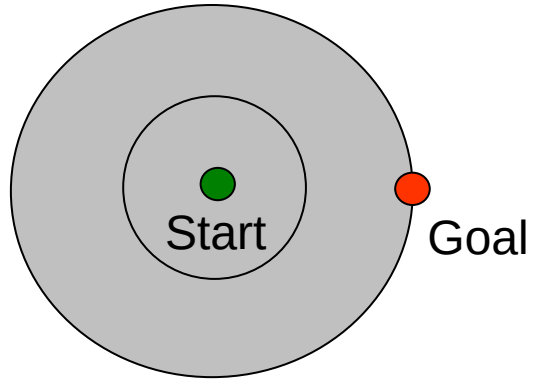


Mane UCS algoritma

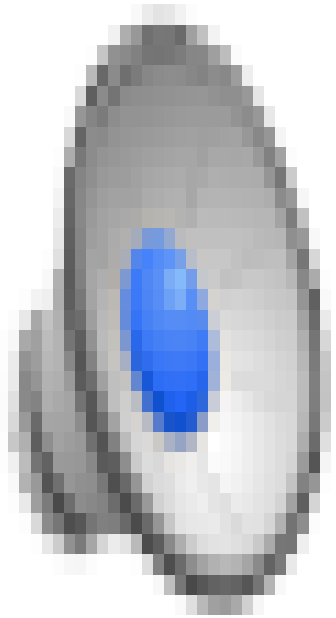
- Dobre strane: UCS je kompletan i optimalan!

- Loše strane:
  - Gleda samo cenu tj. razvija u svim pravcima,
  - Ne uzima u obzir informacije o cilju.

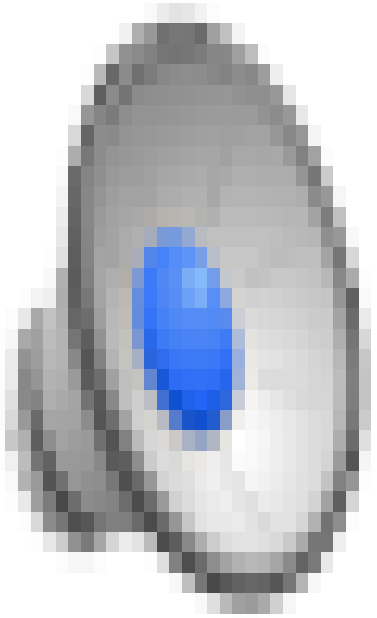




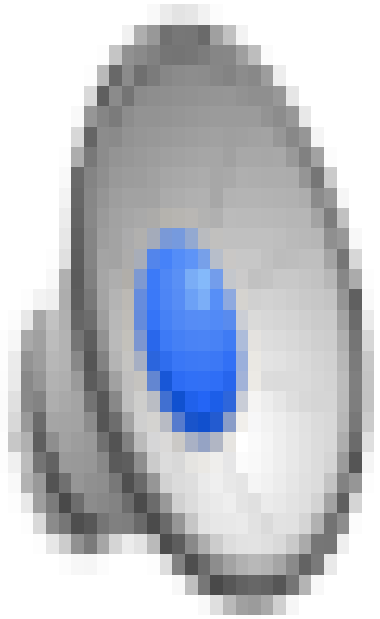
Demo UCS – voda iste dubine



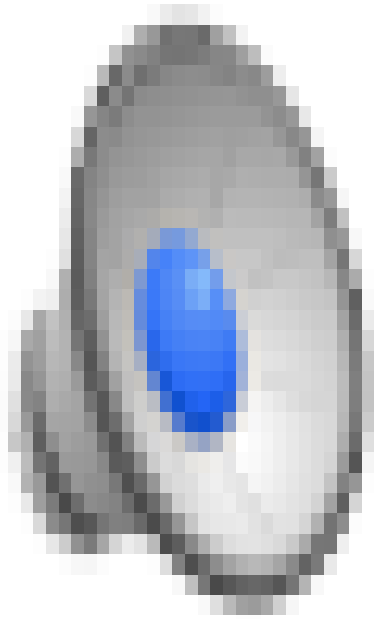
Demo duboka (veća cena) i plitka voda, Koji je algoritam u pitanju: DFS, BFS, ili UCS? (deo 1)



Demo duboka (veća cena) i plitka voda, Koji je algoritam u pitanju: DFS, BFS, ili UCS? (deo 2)

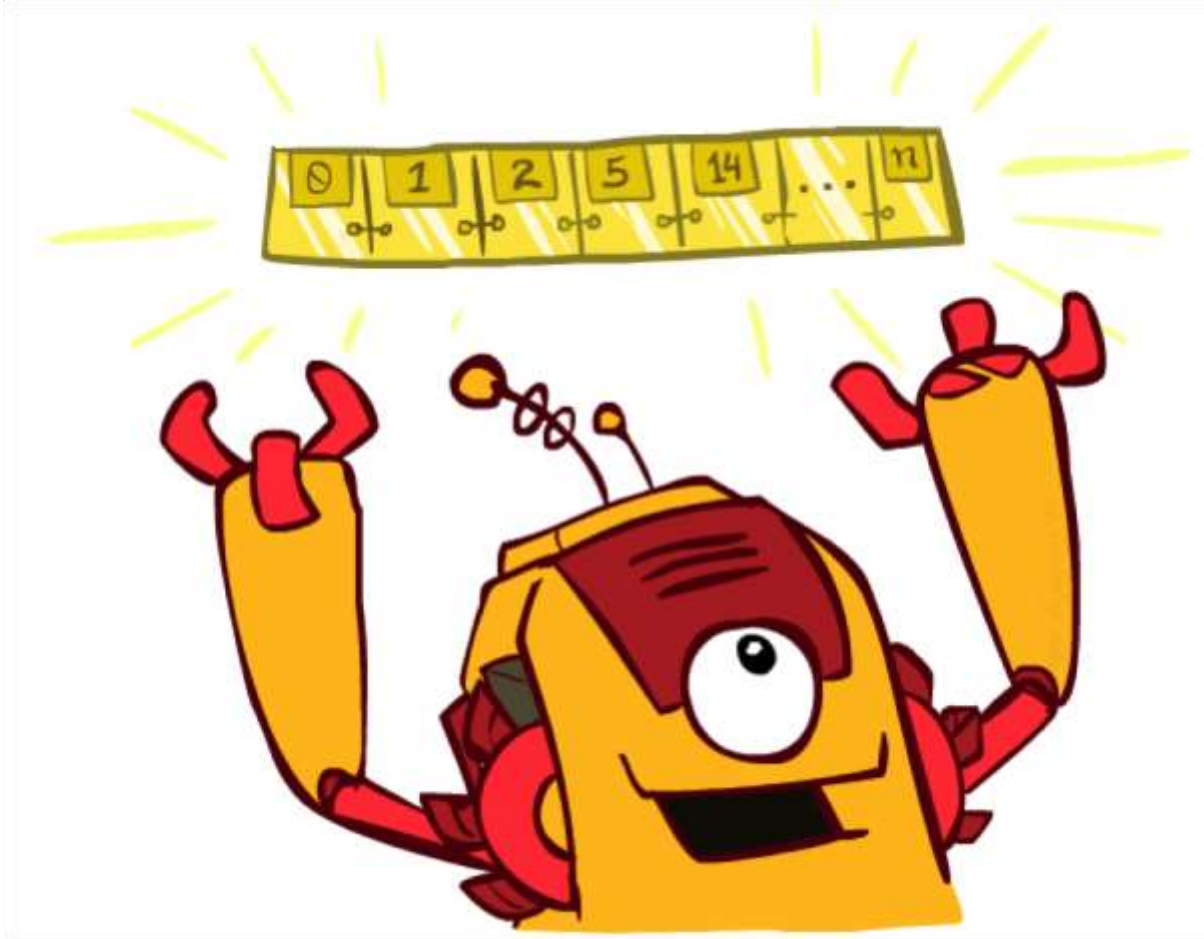


Demo duboka (veća cena) i plitka voda, Koji je algoritam u pitanju: DFS, BFS, ili UCS? (deo 3)



Jedna Struktura

- Svi do sada prikazani algoritmi funkcionišu po istom principu.
- Jedina razlika je u strukturi koja se koristi.
  - Konceptualno, svaki algoritam koristi red prioriteta.
  - Naravno za BFS i DFS je bolje koristiti gotovu implementaciju steka i reda koju nudi jezik koji koristite.



Pretraga i Modeli Sveta

- Pretraga radi sa modelima sveta tj. problema
  - Agent ne isprobava sve akcije u svetu već planira.
  - Planira u simulaciji tj. „u svojoj glavi“
  - To znači da kvalitet pretrage zavisi od kvaliteta modela sveta…