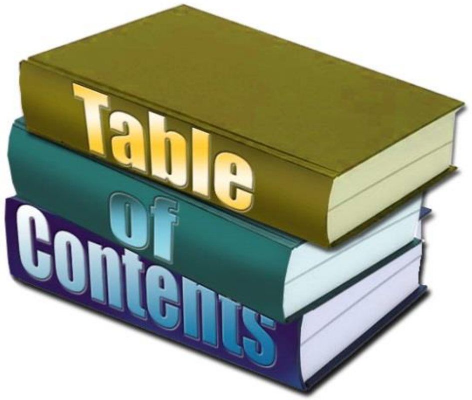
OSNOVE UMETNE INTELIGENCE 2018/19

lokalno preiskovanje

grafi AND/OR

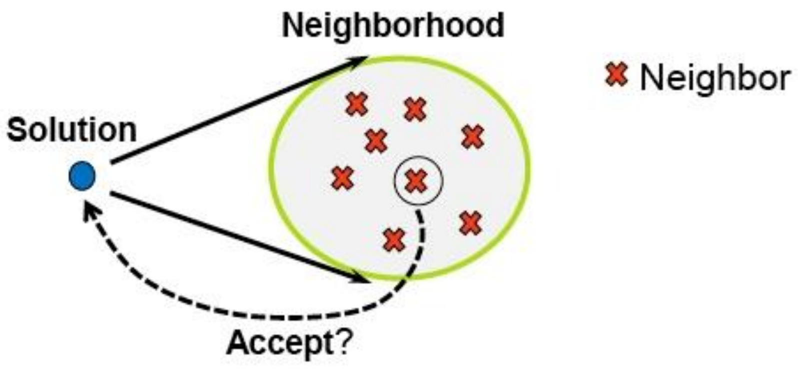
Pregled



- neinformirani preiskovalni algoritmi
- informirani preiskovalni algoritmi
- **lokalni preiskovalni algoritmi in optimizacijski problemi**
  - plezanje na hrib
  - simulirano ohlajanje
  - lokalno iskanje v snopu
- **preiskovanje grafov AND/OR**
  - predstavitev problemov z grafi AND/OR
  - algoritem AO*
  - preiskovanje v nedeterminističnem okolju
  - preiskovanje brez informacije o stanju

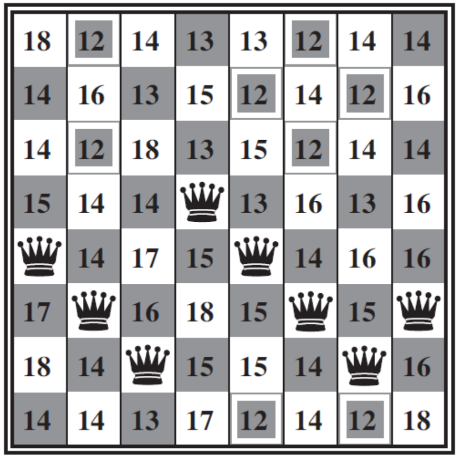
Lokalni preiskovalni algoritmi

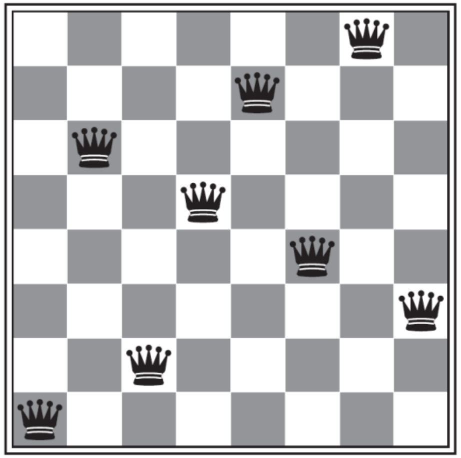
- namesto sistematičnega preiskovanja možnih poti od začetka do cilja izvajajo **<u>iterativno</u> ocenjevanje in spreminjanje podanih stanj**
  - izberi eno trenutno stanje (ali več njih)
  - poišči sosednja stanja od trenutnega, pri tem ne ohranjaj poti
  - ponavljaj do ustavitvenega pogoja
- koristni v primerih:
  - če nas zanima samo kakovost rešitve (stanja) in ne tudi pot do cilja (primer: pri igri 8 ploščic je pot pomembna, pri problemu 8 kraljic pa ne)
  - za reševanje optimizacijskih problemov, kjer je podana **kriterijska funkcija** za oceno kakovosti rešitve
- prednosti:
  - majhna poraba prostora
  - v praksi najdejo dober približek rešitve v prostorih, ki so s sistematičnimi preiskovalnimi algoritmi neobvladljivi



Plezanje na hrib

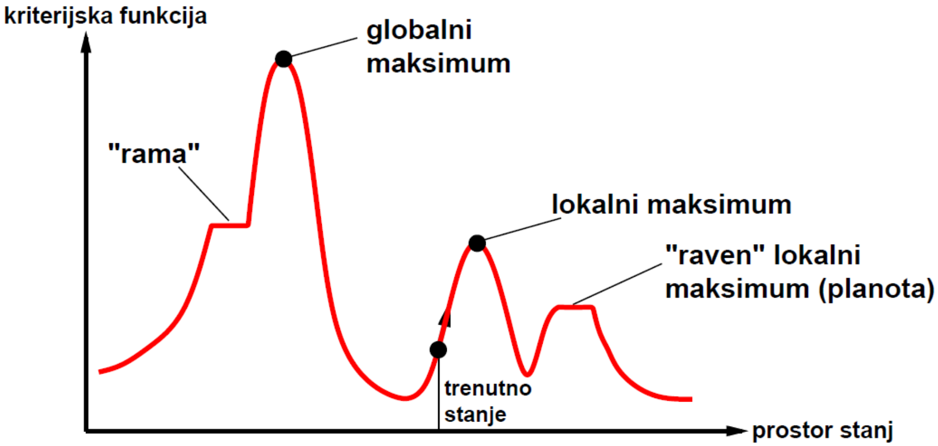
- angl. *hill-climbing search* (ali: *greedy local search*)
- premikaj se po prostoru stanj v smeri najboljše izboljšave kriterijske funkcije
- primer problema 8 kraljic:
  - kot "soseda" trenutnega stanja definiramo stanje, kjer 1 kraljico premaknemo na drugo polje znotraj istega stolpca
  - skupaj: 8×7=56 sosednih stanj
  - kriterijska funkcija: število kraljic, ki se ne napadajo
  - lokalno iskanje lahko "obtiči" v lokalnem optimumu (primer na spodnji sliki: h=1, vendar ima vsak sosed stanja višjo vrednost funkcije)
  - v 14% lokalno iskanje najde rešitev v 4 korakih, v 86% obtiči v lokalnem maksimumu po 3 korakih (vseh stanj je 17 milijonov)

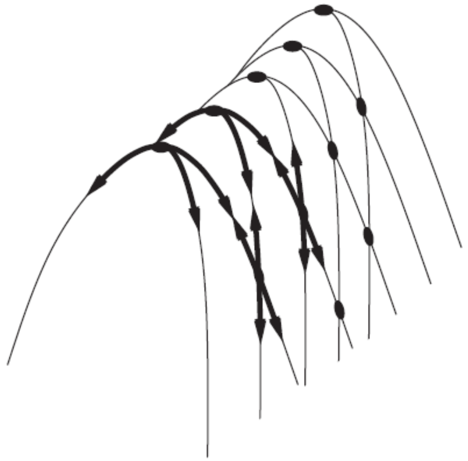




Lokalni preiskovalni algoritmi

- preiskujejo prostor stanj z namenom najti globalni maksimum glede na vrednost kriterijske funkcije
  - optimalni algoritem: najde globalni maksimum
- težave:
  - lokalni maksimumi ("vrhovi")
  - področja, kjer ima kriterijska funkcija konstantno vrednost (rame, planote)
  - grebeni (za plezanje navzgor je najprej potreben sestop po pobočju grebena)





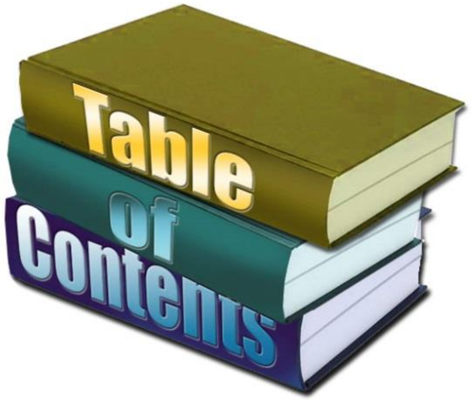
Reševanje iz lokalnih maksimumov

- **koraki vstran**: če ima naslednje stanje isto vrednost kriterijske funkcije, dovolimo premik v to stanje (korak "vstran", angl. *sideways move)*
  - upamo, da smo na "rami" in ne na "planoti"
  - smiselno je omejiti število dovoljenih korakov vstran
  - primer: pri problemu 8 kraljic verjetnost uspeha naraste s 14% na 94%

- **stohastično** plezanje na hrib: iz množice boljših stanj, izberemo naključno stanje za nadaljevanje (z večjo verjetnostjo boljša stanja)

- **naključni ponovni zagon**: večkrat poženi plezanje na hrib iz naključnih začetnih stanj, dokler ne najdeš rešitve
  - če je verjetnost uspeha enega zagona p, je v povprečju potrebnih $1/p$ zagonov
  - za problem 8 kraljic:
    - $p = 0.14 \Rightarrow$ potrebnih je 7 zagonov (skupaj približno 22 korakov)
    - če dovolimo tudi korake vstran ($p = 0{,}94$), je potrebnih $1/0.94 \approx 1{,}06$ zagonov

Pregled

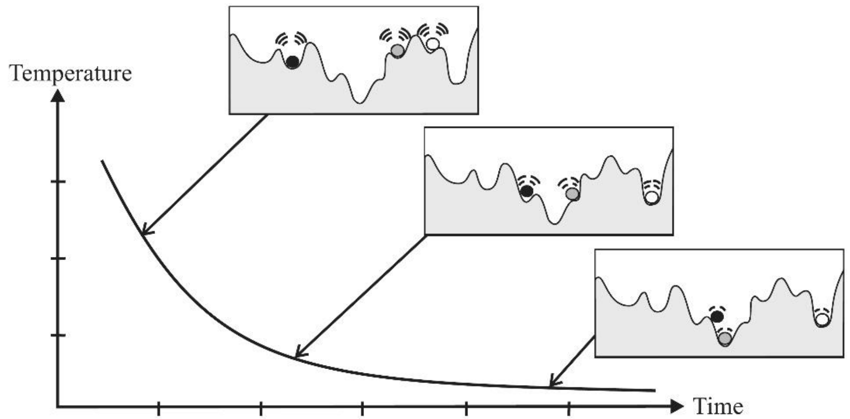


- neinformirani preiskovalni algoritmi
- informirani preiskovalni algoritmi
- **lokalni preiskovalni algoritmi in optimizacijski problemi**
  - plezanje na hrib
  - simulirano ohlajanje
  - lokalno iskanje v snopu
- **preiskovanje grafov AND/OR**
  - predstavitev problemov z grafi AND/OR
  - algoritem AO*
  - preiskovanje v nedeterminističnem okolju
  - preiskovanje brez informacije o stanju

Simulirano ohlajanje

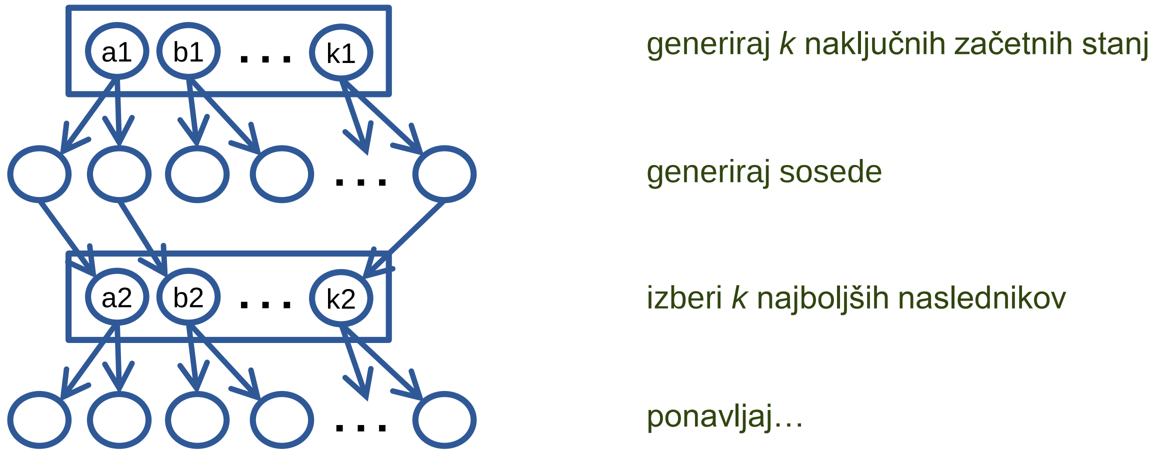
- angl. *simulated annealing*
- optimizacijski algoritem, ki izvira iz fizikalnih lastnosti v metalurgiji (ko je jeklo tekoče, so molekule v njem bolj gibljive; ko se ohlaja, se strjuje in molekule se umirjajo)
- analogija:
  - generiramo naključne sosede trenutnega stanja
  - če najdemo boljše stanje, ga vedno izberemo
  - če najdemo slabše stanje, ga izberemo z določeno verjetnostjo
  - verjetnost izbire neoptimalnega stanja s časom pada (nižanje temperature)
- analogija: zibanje igralne površine, da žogice skočijo iz lokalnih optimumov (na sliki: iščemo lokalne minimume)

for $t \leftarrow 1$ **to** ∞ **do**
  $T \leftarrow schedule[t]$
  if $T = 0$ **then return** *current*
  next ← a randomly selected successor of *current*
  $\Delta E \leftarrow$ VALUE[*next*] – VALUE[*current*]
  if $\Delta E > 0$ **then** *current* ← *next*
  else *current* ← *next* only with probability $e^{\Delta E/T}$



Lokalno iskanje v snopu

- algoritem:
  - v spominu hrani *k* aktualnih stanj namesto enega
  - izberi *k* optimalnih stanj od sosedov aktualnih stanj
  - ponavljaj do ustavitvenega pogoja
- ni enako kot *k* vzporednih iskanj, ker ocenjujemo kakovost cele generacije iskanj (ne neodvisnih vzporednih iskanj)
- problemi:
  - celoten snop k iskanj obtiči v lokalnih maksimumih
  - rešitev: stohastično iskanje v snopu: izberi naključne naslednike z verjetnostjo, ki je sorazmerna njihovi kakovosti

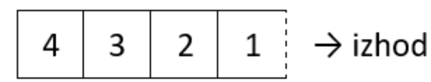


Primer izpitne naloge

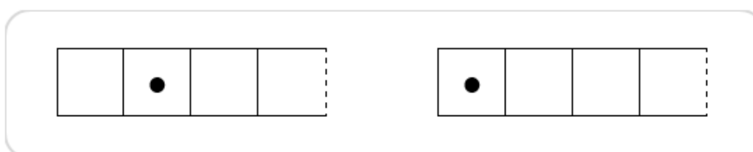
- 3. izpit, 7. 9. 2018

3. NALOGA (25t):

Rešujemo problem iskanja izhoda iz hodnika, ki ga sestavljajo štiri sobe (prikazano na desni sliki). V eni izmed sob se nahaja robot, ki se lahko na vsakem koraku lahko premakne za 1 sobo v levo (L) ali v desno (D).



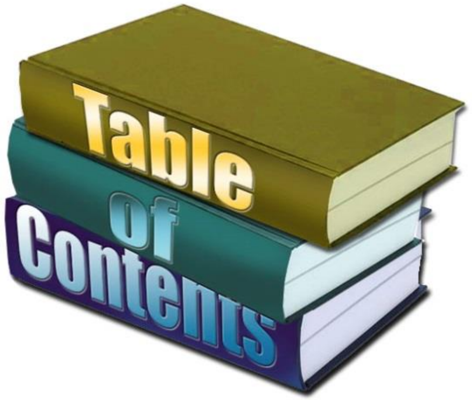
Problem želimo rešiti s preiskovanjem v snopu, ki na vsakem koraku hrani 2 aktualni stanji (k=2). Preiskovanje začnemo s stanjema, ki sta prikazani na spodnji sliki (ti dve stanji smo izbrali naključno, pika prikazuje lokacijo robota). Kot kriterijsko funkcijo najdene rešitve uporabljamo razdaljo od izhoda (vrednost kriterijske funkcije za robota v vsaki posamezni sobi je prikazana na zgornji skici).



Naloge:

a) **(10t) Nariši zaporedje stanj pri preiskovanju, ki ga nadaljuj vse dokler ne najdemo končnega stanja (robot se premakne skozi desno (črtkano) stranico labirinta).**
b) **(10t) Denimo, da namesto "navadnega" iskanja v snopu uporabljamo stohastično iskanje v snopu. Za generirane rešitve v 2. (nasledniki začetnega vozlišča) in 3. iteraciji preiskovanja izračunaj verjetnosti, da bo sosed izbran za naslednjo iteracijo.**
c) **(5t) V katero družino algoritmov spada iskanje v snopu? Naštej še tri druge algoritme iz iste družine, ki jih poznaš.**

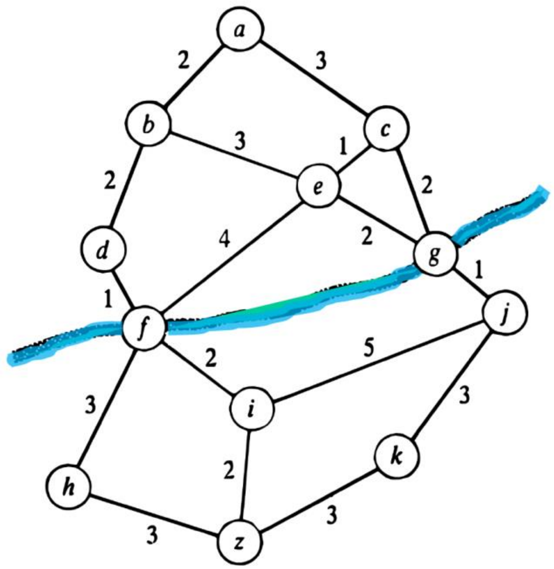
Pregled

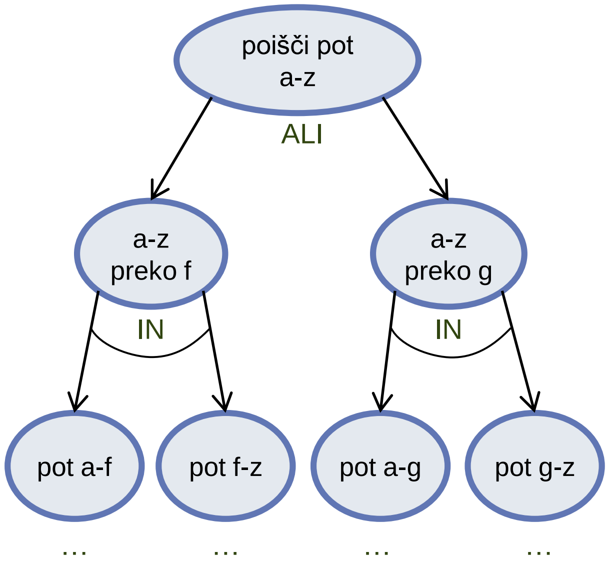


- neinformirani preiskovalni algoritmi
- informirani preiskovalni algoritmi
- lokalni preiskovalni algoritmi in optimizacijski problemi
  - plezanje na hrib
  - simulirano ohlajanje
  - lokalno iskanje v snopu
  - genetski algoritmi
- **preiskovanje grafov AND/OR**
  - predstavitev problemov z grafi AND/OR
  - algoritem AO*
  - preiskovanje v nedeterminističnem okolju
  - preiskovanje brez informacije o stanju

Grafi AND/OR

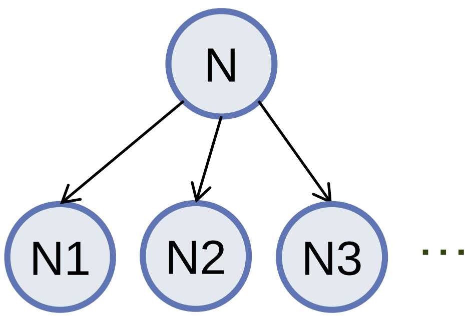
- pomagajo reševati probleme z dekompozicijo na manjše probleme
- uporabnost:
  - poenostavitev reševanja, princip *deli in vladaj*
  - iskanje rešitev v nedeterminističnih okoljih
  - igre med dvema nasprotnikoma s popolno informacijo (šah, dama)
  - ekspertno reševanje problemov
- primer:
  - zemljevid z reko, mosta v vozliščih f in g
  - dekompozicija v manjša problema: poišči pot $a - z$ preko f ALI pot $a - z$ preko g



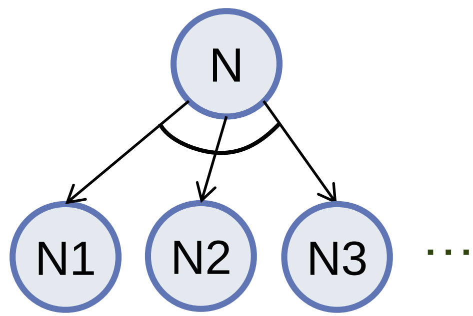


Grafi AND/OR

- vozlišča dveh tipov
  - vozlišče OR: za rešitev vozlišča N reši N1 ali N2 ali N3 ali …
  - vozlišče AND: za rešitev vozlišča N reši N1 in N2 in N2 in …



vozlišče tipa OR
(disjunkcija)

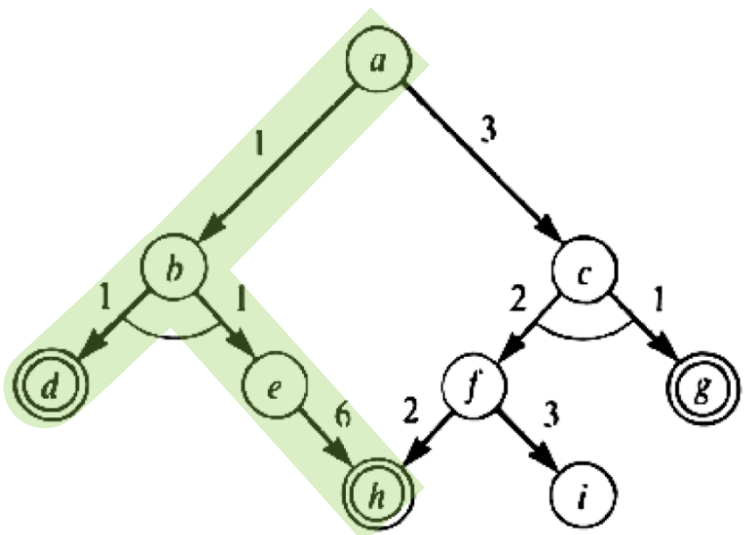


vozlišče tipa AND
(konjunkcija)

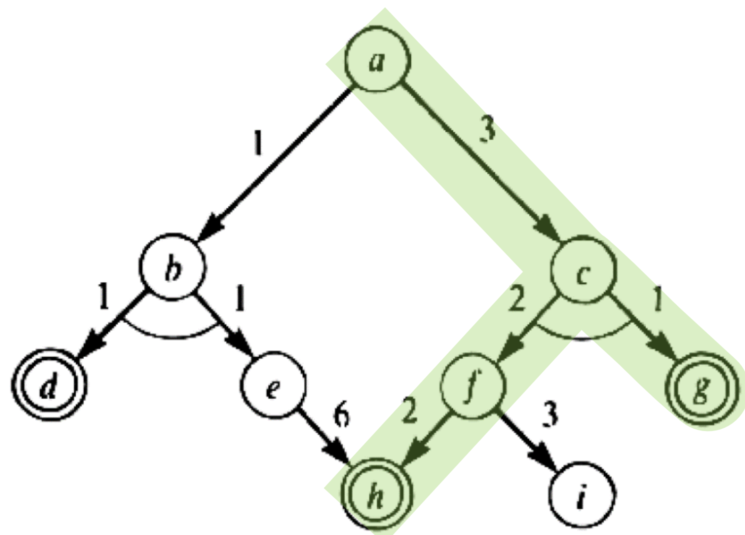
- uporaba neinformiranih metod za preiskovanje
  - iskanje v globino
  - iskanje v globino z omejitvijo globine
  - iskanje v širino
  - iterativno poglabljanje

Rešitev grafa AND/OR

- problem je predstavljen z začetnim vozliščem, grafom, cilji
- rešitve problema so cela drevesa
- cena rešitve – vsota cen povezav v drevesu
- graf ima lahko različne rešitve z različnimi cenami



cena = 9

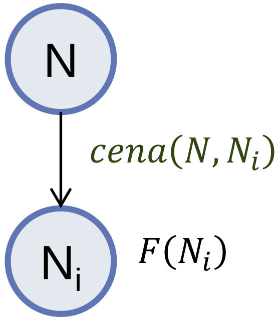


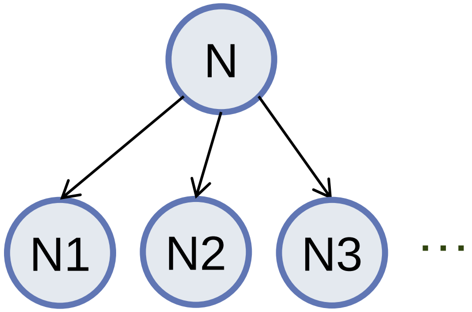
cena = 8

Preiskovanje grafa AND/OR

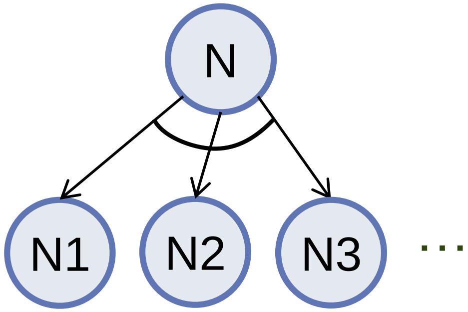
- informirana metoda – **algoritem AO***
  - posplošitev A* na grafe AND/OR
  - tudi za AO* velja, da je popoln, če hevristika nikoli ne precenjuje dejanske cene do cilja
  - cena vozlišča – cena optimalnega rešitvenega drevesa

- **definirajmo**:
  - vsako vozlišče N ima:
    - lokalno (dinamično) hevristično oceno $H(N)$
    - lokalno (dinamično) vrednost kriterijske funkcije $F(N)$:
      $F(N_i) = G(N_i) + H(N_i) = cena(N, N_i) + H(N_i)$
  - dinamična hevristična ocena $H(N)$ je odvisna od vozlišča:
    - za liste: $H(N) = h(N)$
    - za notranja vozlišča: izračun $H(N)$ glede na vrsto vozlišča



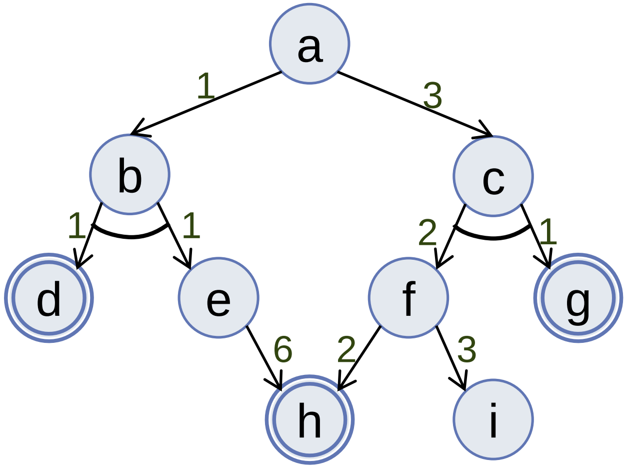


$$H(N) = \min_i (cena(N, N_i) + H(N_i))$$

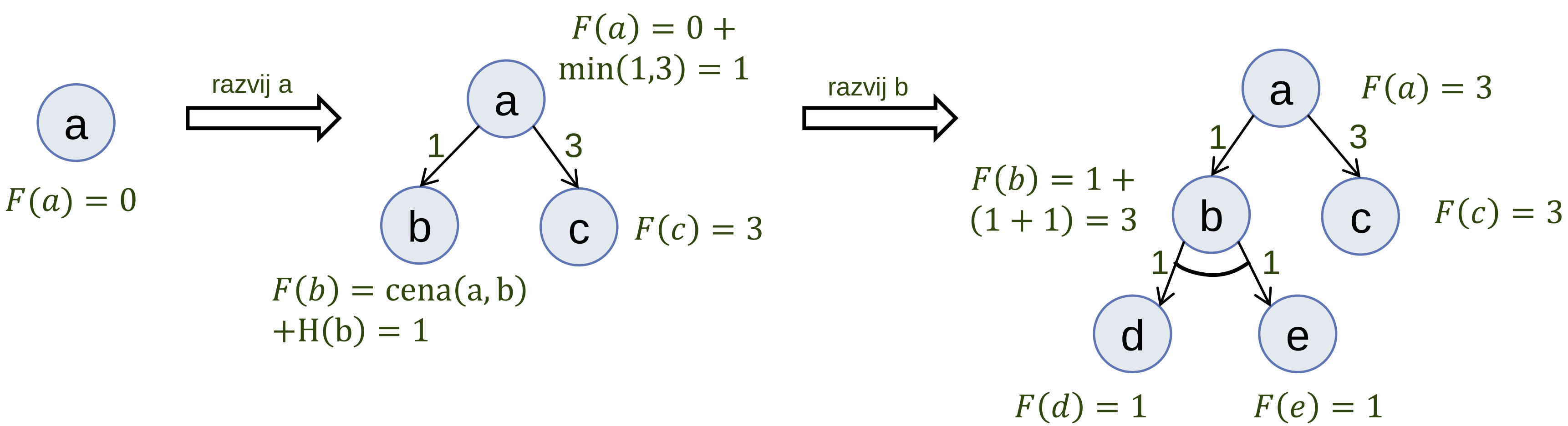


$$H(N) = \sum_i (cena(N, N_i) + H(N_i))$$

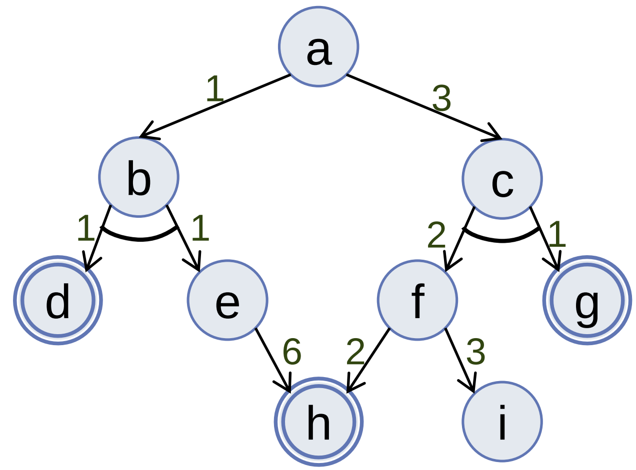
Preiskovanje grafa AND/OR



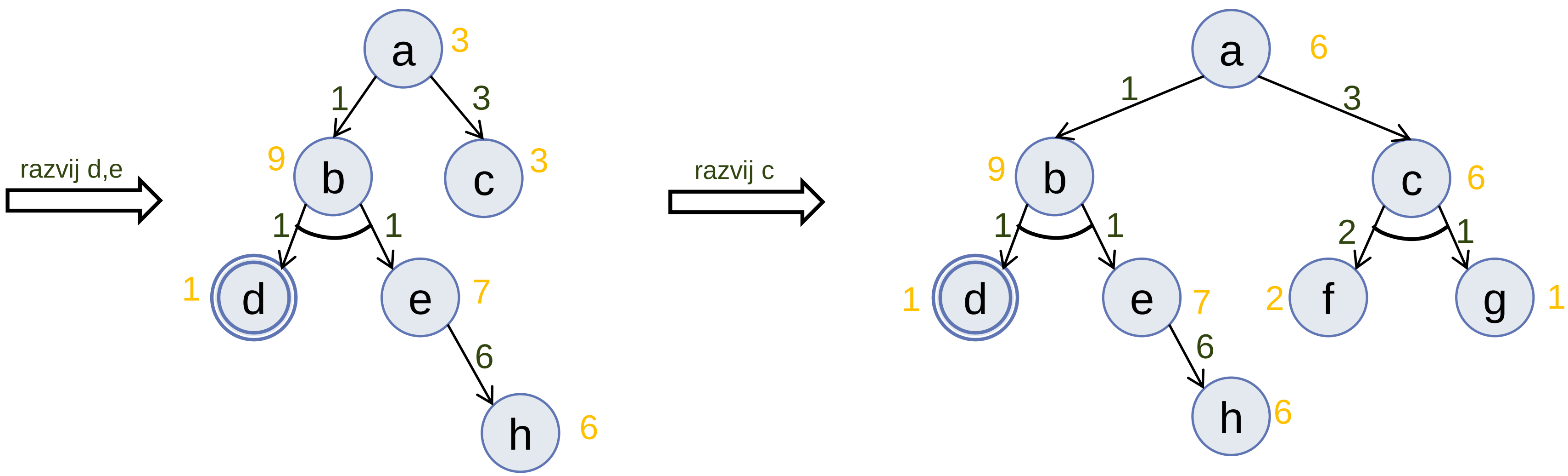
- primer (Bratko, 2001)
- $h = 0$ za vsa vozlišča
- pripisane so vrednosti $F(N_i) = cena(N, N_i) + H(N_i)$



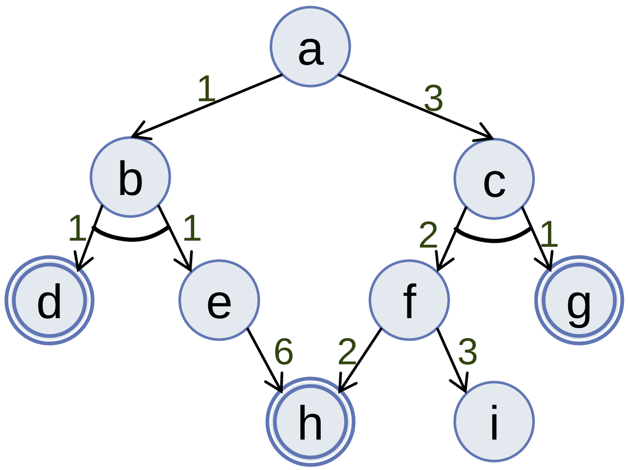
Preiskovanje grafa AND/OR



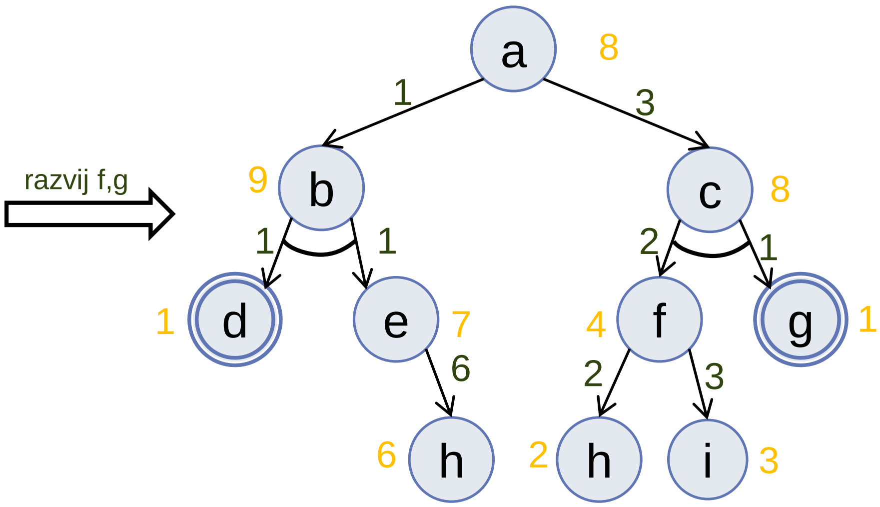
- primer (Bratko, 2001)
- $h = 0$ za vsa vozlišča
- pripisane so vrednosti $F(N_i) = cena(N, N_i) + H(N_i)$

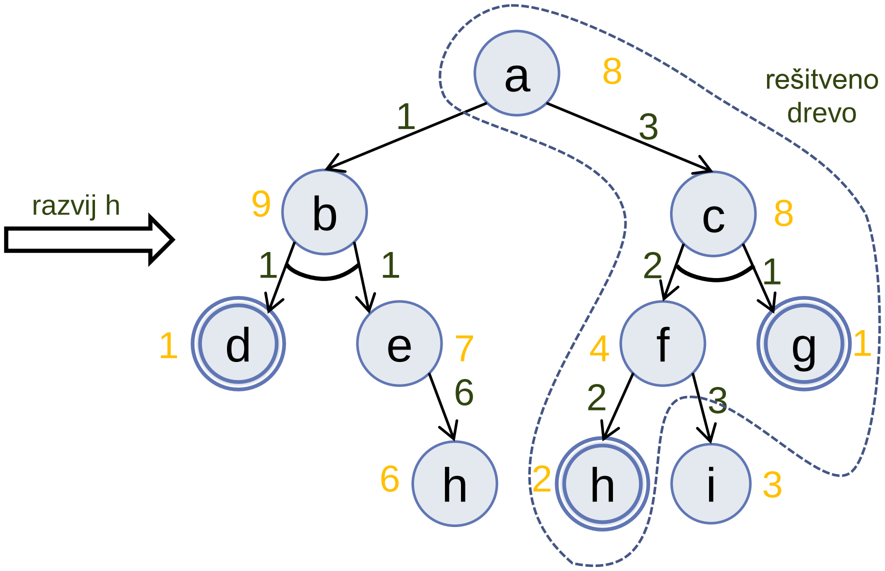


Preiskovanje grafa AND/OR



- primer (Bratko, 2001)
- $h = 0$ za vsa vozlišča
- pripisane so vrednosti $F(N_i) = cena(N, N_i) + H(N_i)$





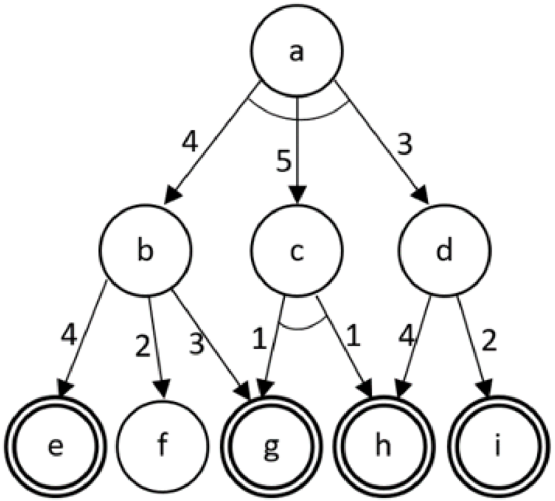
Primer izpitne naloge

- 3. izpit, 7. 9. 2018

1. NALOGA (25t):

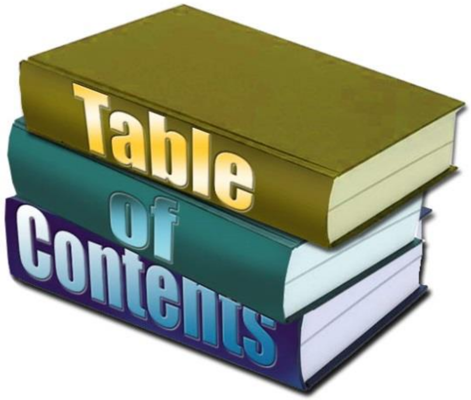
Podan je AND/OR graf na sliki. Hevristične ocene posameznih vozlišč so podane v spodnji tabeli. Naslednike vozlišč generiramo po abecednem vrstnem redu, razvijamo pa jih glede na vrednost dinamične hevristične ocene. Pri vozliščih tipa AND vedno razvijemo vse naslednike.

N	a	b	c	d	e	f	g	h	i
h(n)	5	4	2	7	4	1	6	6	7



a) **(15t) Simuliraj algoritem AO* in zapiši dobljeno rešitveno drevo.**
b) **(4t) Ali je rešitveno drevo iz prejšnje točke optimalno? Če ni, nariši optimalno rešitveno drevo.**
c) **(6t) Kako bi morali popraviti hevristične ocene vozlišč, da bi bila rešitev optimalna?**

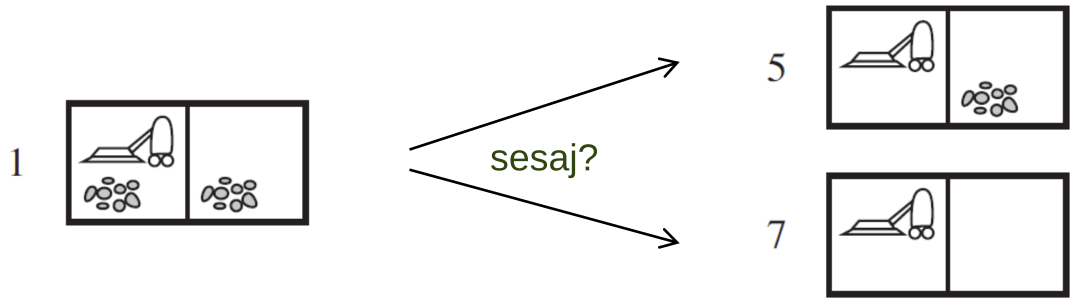
Pregled



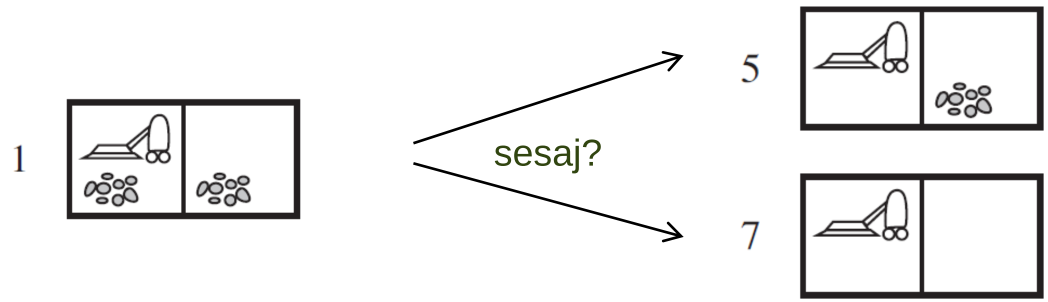
- **neinformirani preiskovalni algoritmi**
- **informirani preiskovalni algoritmi**
- **lokalni preiskovalni algoritmi in optimizacijski problemi**
  - plezanje na hrib
  - simulirano ohlajanje
  - lokalno iskanje v snopu
  - genetski algoritmi
- **preiskovanje grafov AND/OR**
  - predstavitev problemov z grafi AND/OR
  - algoritem AO*
  - preiskovanje v nedeterminističnem okolju
  - preiskovanje brez informacije o stanju

Preiskovanje v nedeterminističnem okolju

- do sedaj smo se posvetili **determinističnim** in transparentnim okoljem
- kaj pa, če so **akcije nedeterministične** (ista akcija lahko obrodi različna ciljna stanja)?
- primer:
    - sesalec lahko ob sesanju včasih posesa tudi sosednji prostor
    - sesalec lahko včasih ob sesanju čistega prostora tudi umaže trenutni prostor (okvara?)
- potrebna je **redefinicija prehodne funkcije**:
    - **deterministična** (do sedaj):
    $$rezultat(trenutno\_stanje, akcija) = novo\_stanje$$
    - **nedeterministična**:
    $$rezultat(trenutno\_stanje, akcija) = \{novo\_stanje1, novo\_stanje2, \dots\}$$



Preiskovanje v nedeterminističnem okolju



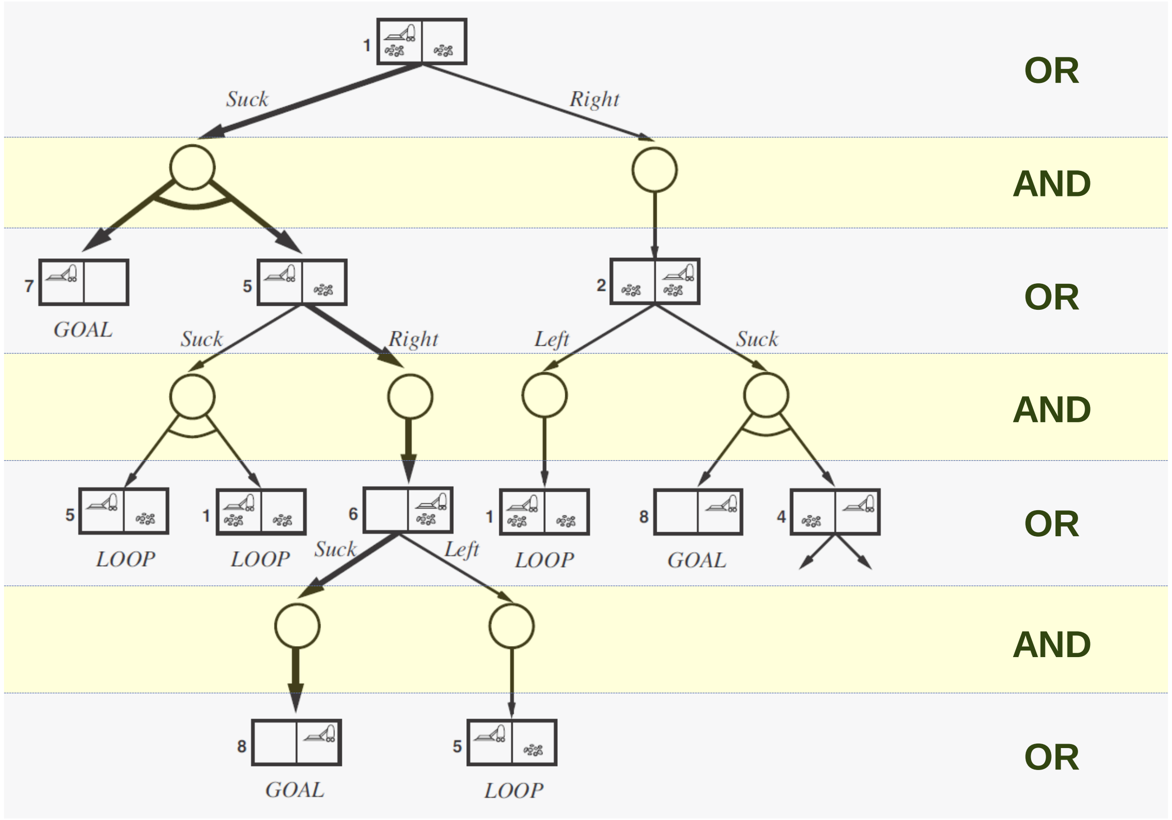
- zaporedje rešitev v prostoru z nedeterminističnimi akcijami mora upoštevati vse poti v prostoru stanj glede na možne rezultate akcij
- primer rešitve za zgornji primer:

```
sesaj
if stanje==5
    desno
    sesaj
else Ø /* CILJ */
```

- zgornje pomeni, da rešitve problemov niso več **poti**, temveč **drevesa**

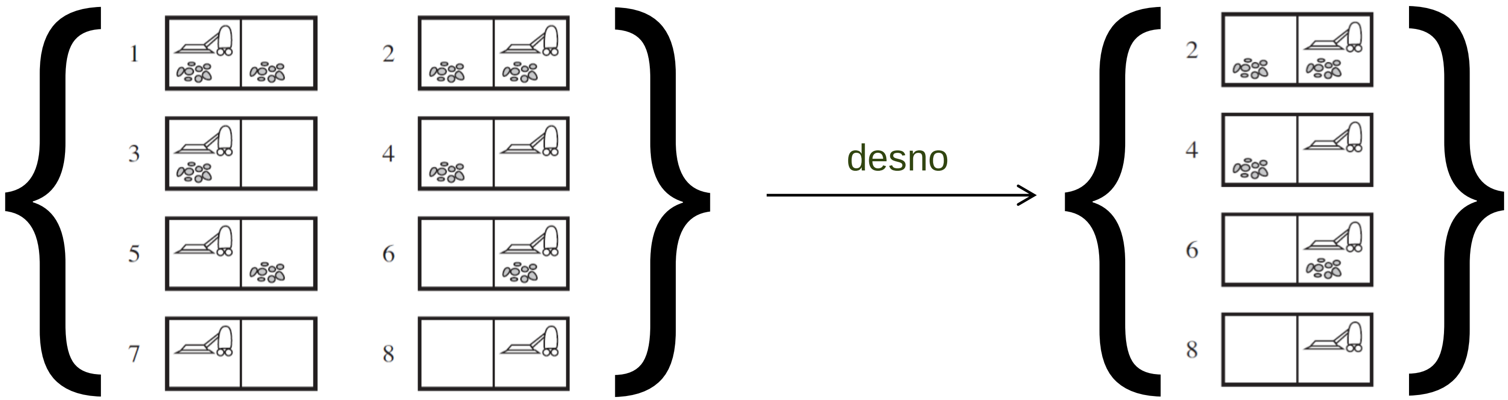
Preiskovanje v nedeterminističnem okolju

- predstavitev rešitve z drevesom AND/OR:
  - vozlišča OR: predstavljajo možne akcije, med katerimi robot lahko izbira v danem stanju
  - vozlišče AND: predstavljajo možna stanja, ki so rezultat nedeterminističnih akcij
  - v drevesu si izmenično sledijo OR in AND nivoji
- primer (pozor, drugačna predstavitev vozlišč)

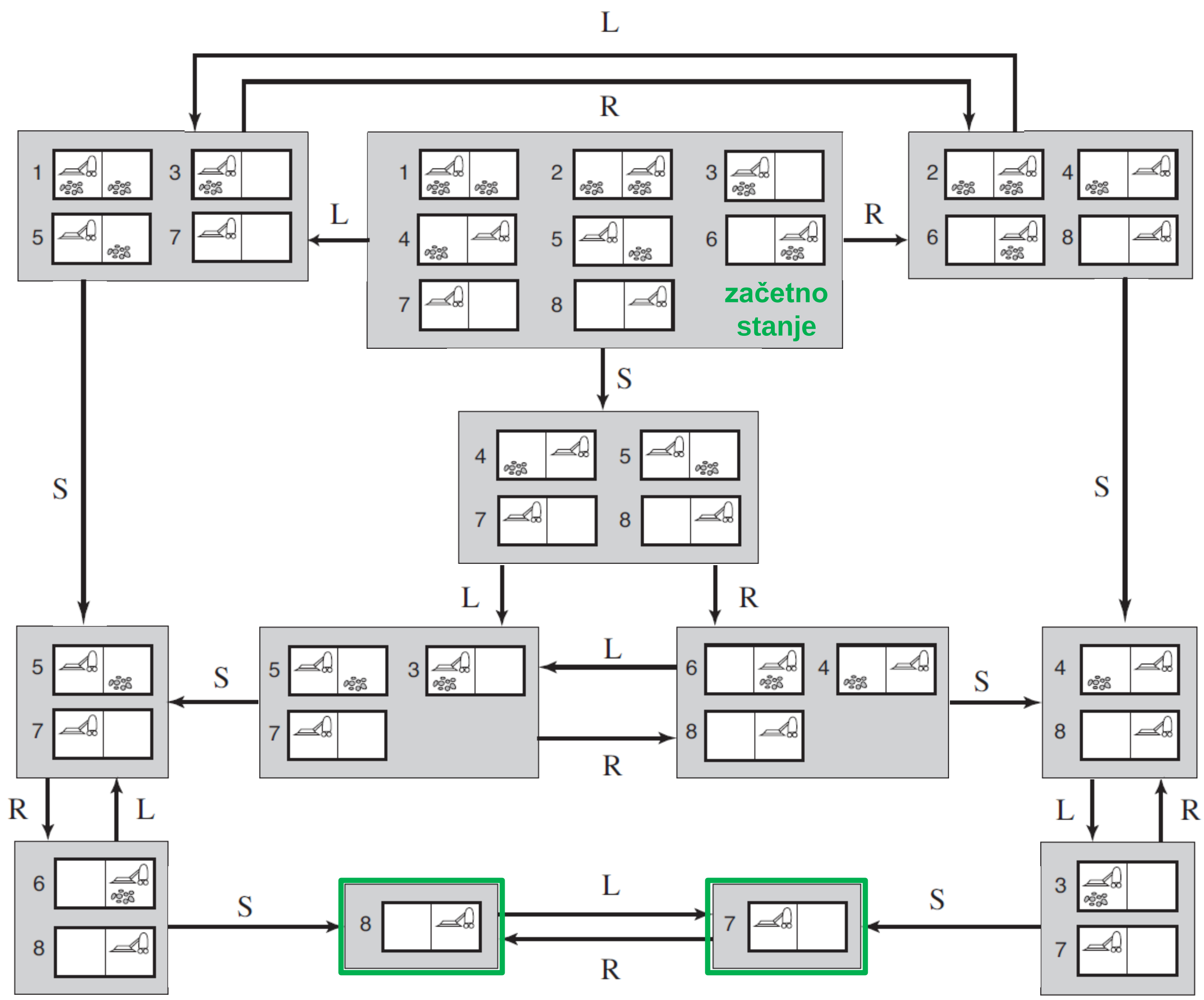


Preiskovanje brez informacije o stanju

- okolja smo razdelili na **transparentna** (angl. *observable*, agent lahko zazna popolno informacijo) in **netransparentna**
- kaj če imamo opravka z netransparentnim okoljem, v katerem agent ne ve, v katerem stanju se nahaja (npr. agent brez senzorjev)?
  - primeri, prednosti?
- preiskovanje
  - izvajamo preiskovanje prostora **verjetnih** stanj (angl. *belief* states) in ne prostora **dejanskih** stanj
  - izvajamo s postopkom omejevanja možnosti kandidatnih stanj (angl. *coercion*) ob izvedbi določenih akcij



Primer

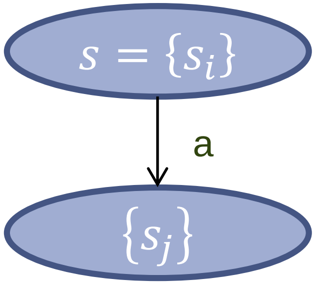


L
R
1
3
5
7
L
1
2
3
4
5
6
7
8
začetno stanje
R
2
4
6
8
S
S
S
4
5
7
8
L
R
5
7
S
5
3
7
L
R
6
4
8
S
4
8
R
L
L
R
6
8
S
8
L
R
7
S
3
7

Preiskovanje brez informacije o stanju

Definicija problema preiskovanja brez informacije o stanju:

- **verjetna stanja** (angl. *belief states*): prostor verjetnih stanj je sestavljen iz potenčne množice vseh možnih dejanskih (fizičnih stanj)
- **začetno stanje**: običajno je to množica vseh možnih dejanskih stanj
- **akcije**: če za stanje $s = \{s_1, s_2\}$ velja $akcije(s_1) \neq akcije(s_2)$, se je potrebno odločiti za strategijo:
  - $akcije(s) = \bigcup_{s_i \in s} akcije(s_i)$ - preprosto, vendar akcija $s_k \in akcije(s_1) \backslash akcije(s_2)$ lahko pripelje do neveljavnega stanja
  - $akcije(s) = \bigcap_{s_i \in s} akcije(s_i)$ - bolj varno, razvito stanje vsebuje samo stanja, ki so možen rezultat vseh akcij
- **prehodna funkcija**:
  - $rezultat(s, \mathrm{a}) = \{\mathrm{s_j} : \mathrm{s_j} = \mathrm{rezultat}(\mathrm{s_i}, \mathrm{a}), \mathrm{s_i} \in s\}$



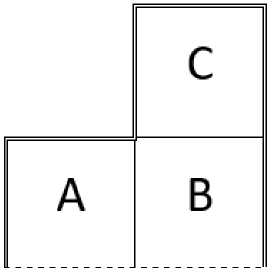
- **ciljno stanje**: verjetno stanje, v katerem vsa dejanska stanja izpolnjujejo ciljni predikat
- **cena poti?**

Primer izpitne naloge

- 2. izpit, 15. 2. 2018

<u>4. NALOGA:</u>

Podan je majhen labirint, sestavljen iz treh sob (A, B in C). V labirintu se nahaja robot, ki mora priti v sobo C. Vendar pa robot nima senzorja za zaznavo, v kateri sobi se nahaja, zato mora brez informacije o stanju poiskati akcije, ki ga bodo zagotovo pripeljale na cilj. Možne akcije, ki jih lahko izvede robot, so: U (up – premik gor), D (down – premik dol), L (left - premik levo) in R (right – premik desno). V kolikor se robot želi v neki sobi premakniti v smer stene (stene so označene z dvojno obrobo; npr. v sobi A so stene v smeri gor in levo), robot ostane na mestu. V kolikor se robot želi premakniti v smeri črtkanih sten (npr. v sobah A in B navzdol), pa pade iz labirinta (preide v nedovoljeno stanje).



a) **(16t) Nariši prostor verjetnih stanj (angl. belief states) in prehodov med njimi (glede na robotove akcije) za dani problem.**
b) **(9t) Navedi šest primerov rešitvenih poti, ki ne vsebujejo ciklov in ob vsaki akciji spremenijo stanje verjetja, po naraščajoči dolžini poti.**


Igranje iger, planiranje