# OSNOVE UMETNE INTELIGENCE
## 2022/23

*grafi AND/OR*

*algoritem AO**

*igranje iger*

# Pridobljeno znanje s prejšnjih predavanj

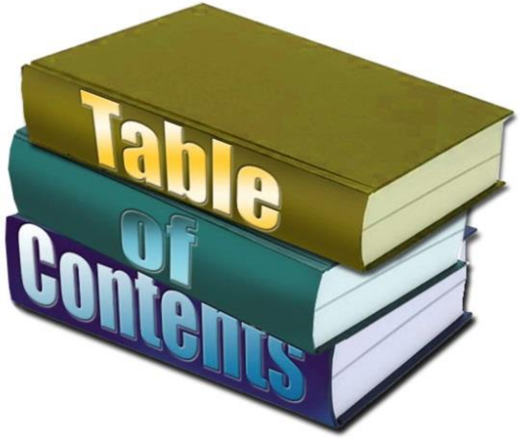
- **lokalni preiskovalni algoritmi**

  - **iterativno** ocenjujejo in spreminjajo **aktualno množico stanj**, koristni, kadar nas ne zanima pot do cilja, majhna poraba prostora

  - **algoritmi**:
    - **plezanje na hrib**: generiranje sosedov in premikanje v smeri najboljše izboljšave
    - **simulirano ohlajanje**: verjetnost izbire slabšega stanja se niža s temperaturo, način pobega iz lokalnega optimuma
    - **lokalno iskanje v snopu**: hranimo več stanj namesto enega, izbiramo najboljše sosede iz cele množice stanj

  - **izogibanje lokalnim optimumom**: koraki vstran, stohastično plezanje, naključni ponovni zagon

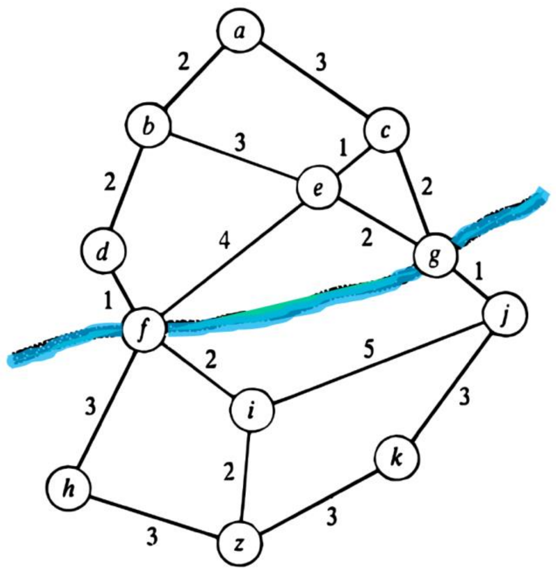
# Pregled

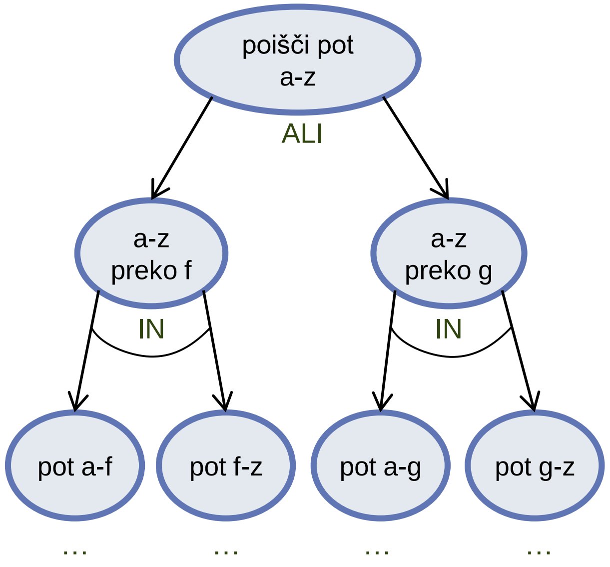



- preiskovanje
  - neinformirani preiskovalni algoritmi
  - informirani preiskovalni algoritmi
  - lokalni preiskovalni algoritmi in optimizacijski problemi
  - **preiskovanje grafov AND/OR**
    - predstavitev problemov z grafi AND/OR
    - algoritem AO*
    - preiskovanje v nedeterminističnem okolju
  - **preiskovanje brez informacije o stanju**
  - **igranje iger**
    - algoritem MINIMAX
    - rezanje alfa-beta

# Grafi AND/OR

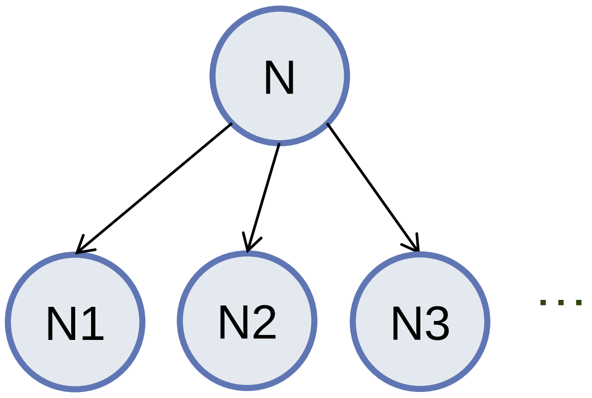
- pomagajo reševati probleme z dekompozicijo na manjše probleme
- uporabnost:
  - poenostavitev reševanja, princip *deli in vladaj*
  - iskanje rešitev v nedeterminističnih okoljih
  - igre med dvema nasprotnikoma s popolno informacijo (šah, dama)
  - ekspertno reševanje problemov
- primer:
  - zemljevid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



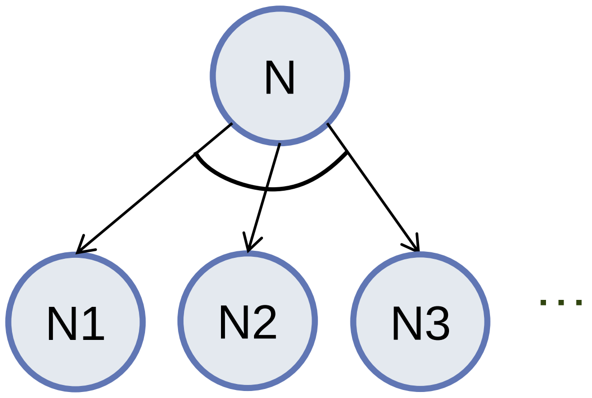



# Grafi AND/OR

- 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** N3 **in** …



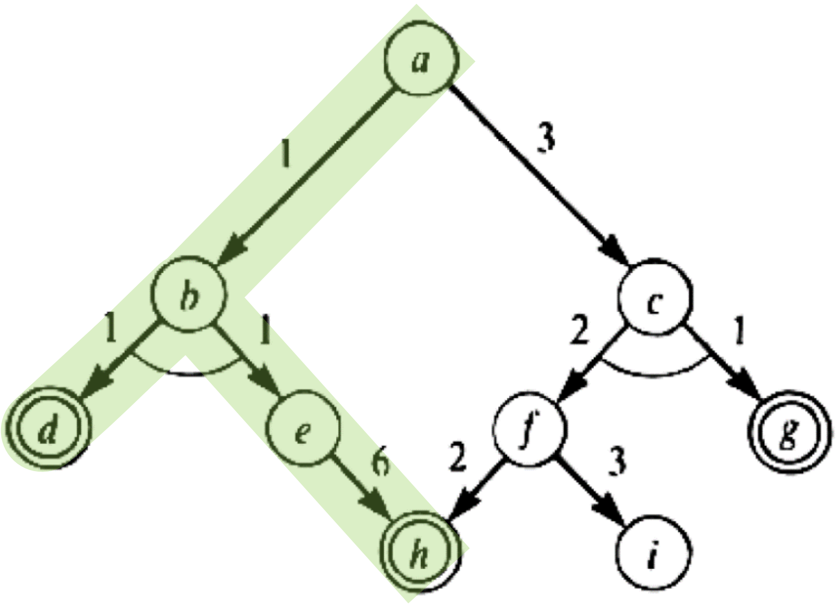

vozlišče tipa OR (disjunkcija)



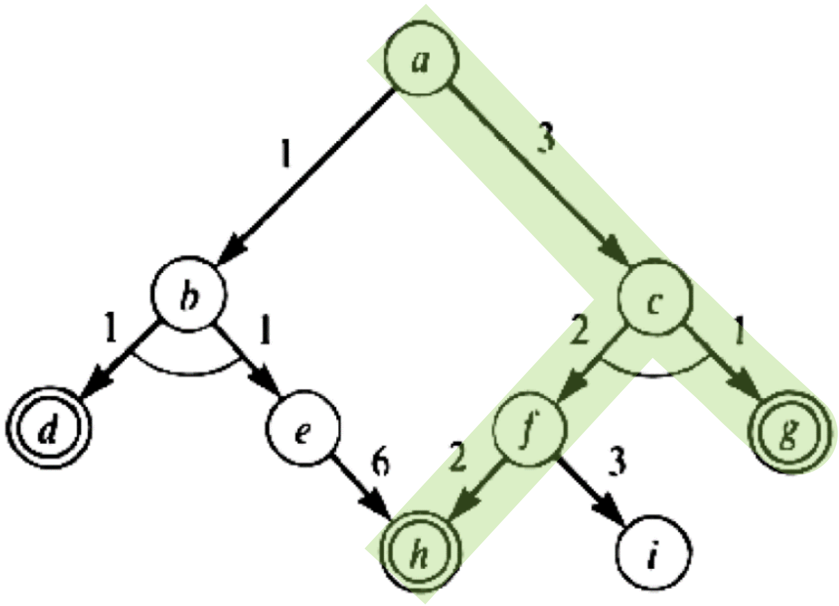

vozlišče tipa AND (konjunkcija)

# Rešitev grafa AND/OR

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



cena = 9

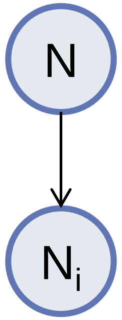


cena = 8

# Preiskovanje grafa AND/OR

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

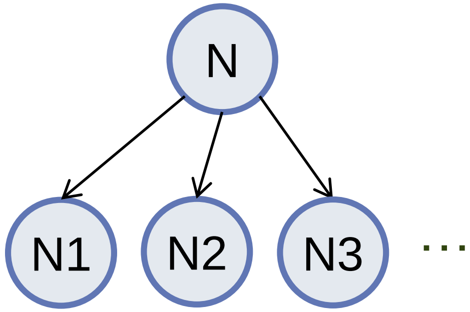
- **definirajmo**:
  - vsako vozlišče $N$ ima:
    - lokalno (dinamično) **hevristično oceno** $H(N)$
    - lokalno (dinamično) vrednost **kriterijske funkcije** $F(N)$:
      $F(N_i) = G(N_i) + H(N_i) = cena(N, N_i) + H(N_i)$
  - dinamična hevristična ocena $H(N)$ je **odvisna od tipa vozlišča**:
 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



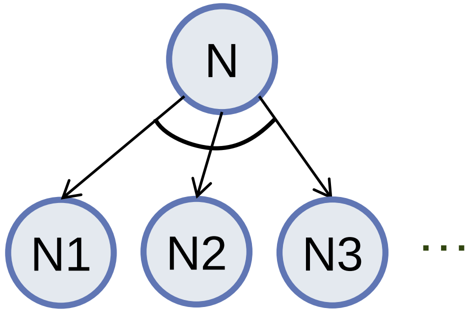

$G(N_i) = cena(N, N_i)$

$H(N_i)$

$F(N_i) = G(N_i) + H(N_i)$

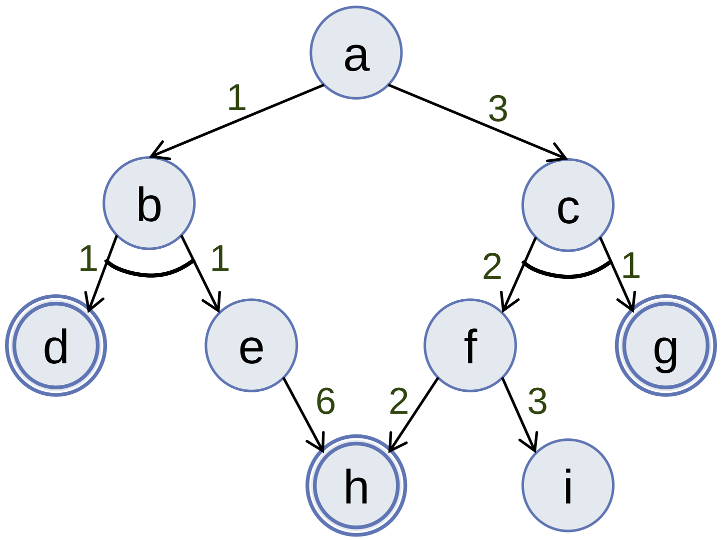



$$H(N) = \min_i (cena(N, N_i) + H(N_i))$$

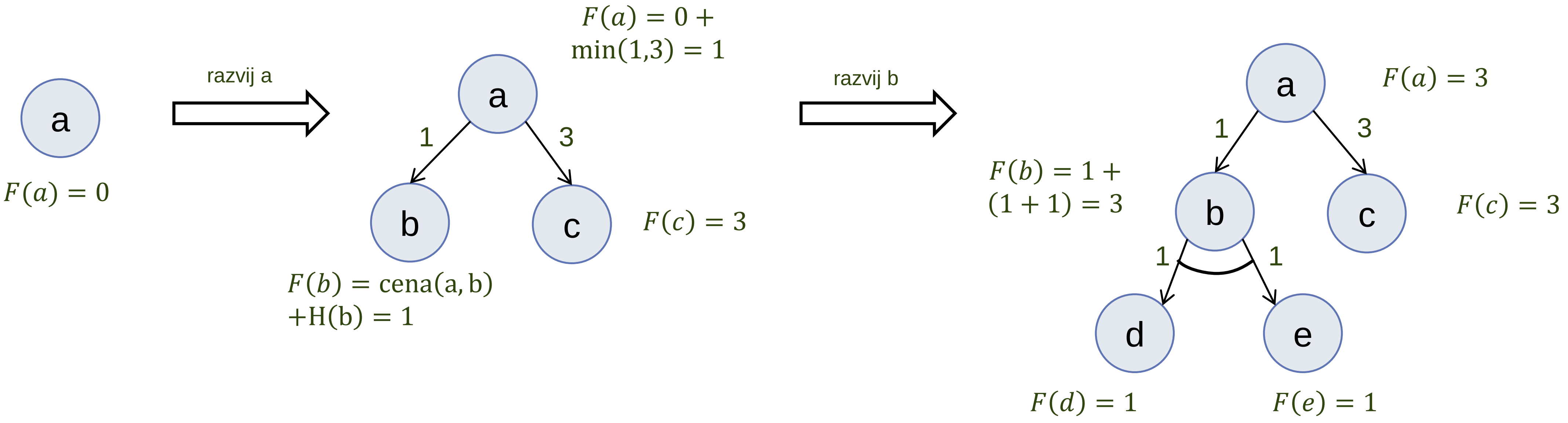



$$H(N) = \sum_i (cena(N, N_i) + H(N_i))$$

# Preiskovanje grafa AND/OR

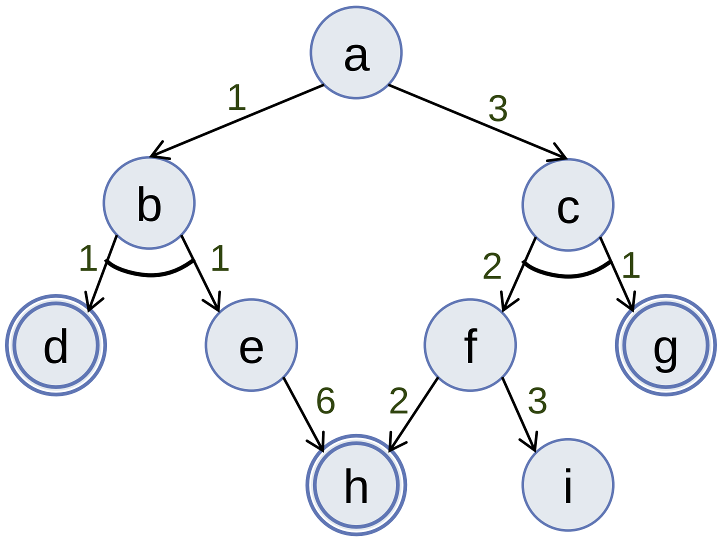


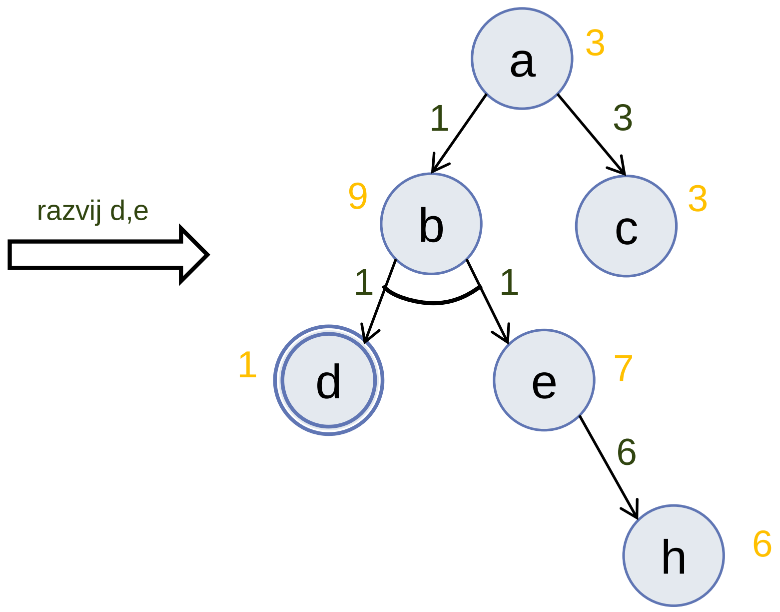

- primer (Bratko, 2001)
- $h = 0$ za vsa vozlišča
- pripisane so vrednosti $F(N_i) = cena(N, N_i) + H(N_i)$
- iztočnice:
  - **vozlišče OR**: razvijamo najbolj obetavno poddrevo, dokler njegova cena ne preseže cene alternativnega poddrevesa,
  - **vozlišče AND**: razvijemo vsa poddrevesa (od leve proti desni), razvijemo do konca preden gremo na naslednje poddrevo.

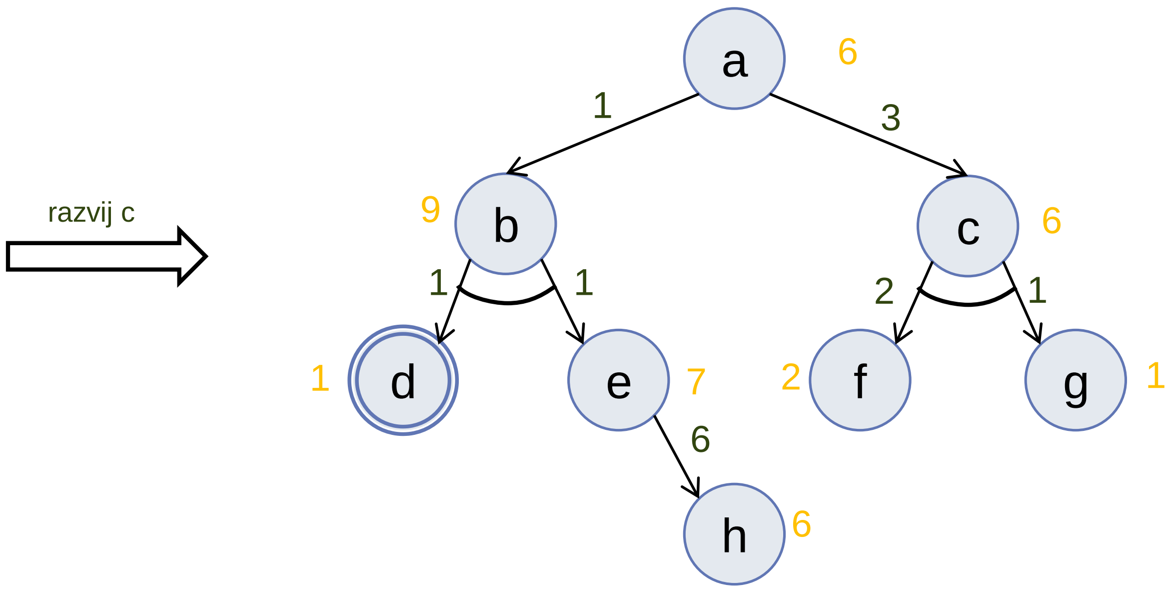


# Preiskovanje grafa AND/OR

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

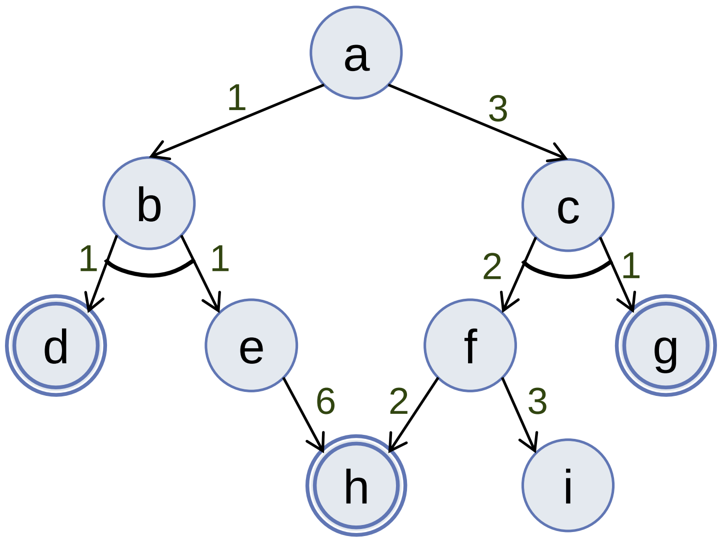


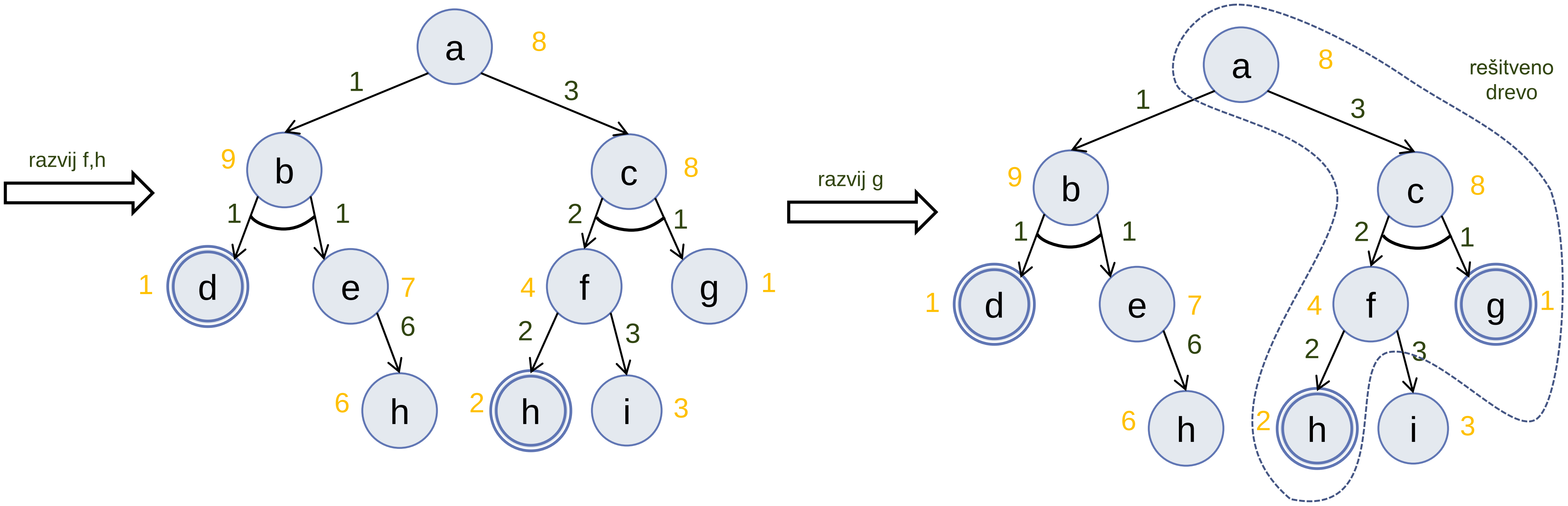



# Preiskovanje grafa AND/OR

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



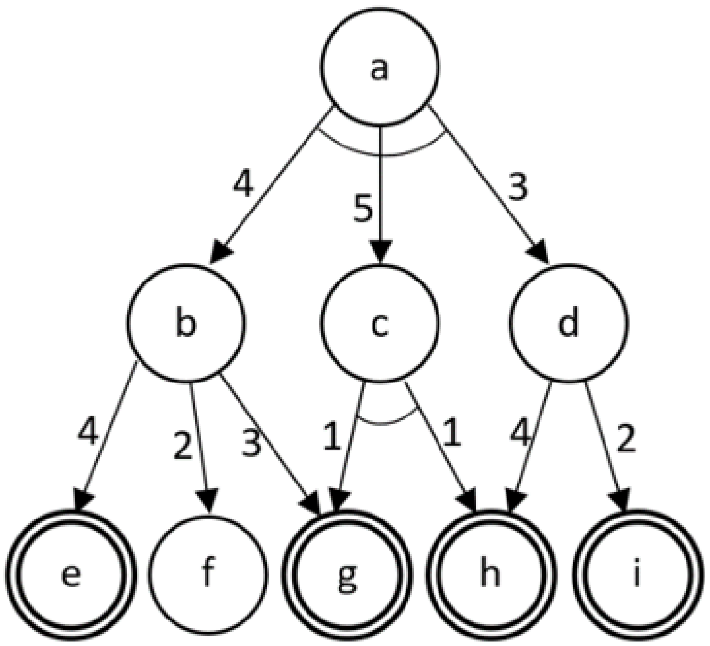
# Primer izpitne naloge

- 3. izpit, 7. 9. 2018

## 1. NALOGA (25t):

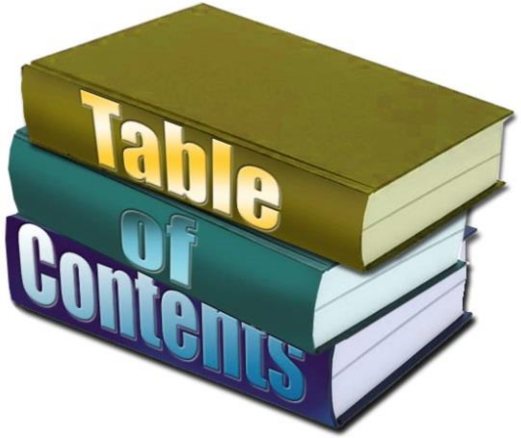
Podan je AND/OR graf na sliki. Hevristične ocene posameznih vozlišč so podane v spodnji tabeli. Naslednike vozlišč generiramo po abecednem vrstnem redu, razvijamo pa jih glede na vrednost hevristične ocene vozlišča. Pri vozliščih tipa AND vedno razvijemo vse naslednike.

| 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




- preiskovanje
  - neinformirani preiskovalni algoritmi
  - informirani preiskovalni algoritmi
  - lokalni preiskovalni algoritmi in optimizacijski problemi
  - **preiskovanje grafov AND/OR**
    - predstavitev problemov z grafi AND/OR
    - algoritem AO*
    - preiskovanje v nedeterminističnem okolju
  - **preiskovanje brez informacije o stanju**
  - **igranje iger**
    - algoritem MINIMAX
    - rezanje alfa-beta

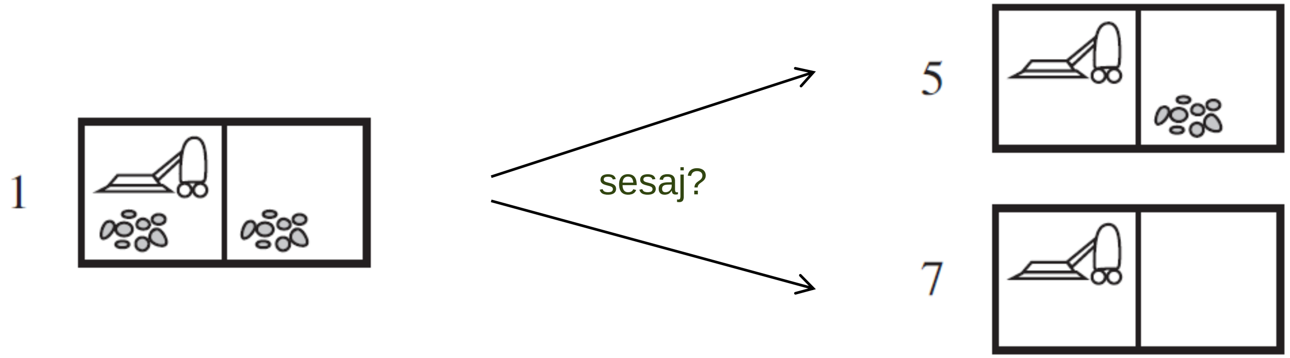
# Preiskovanje v nedeterminističnem okolju

- 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 umazanega prostora včasih posesa tudi sosednji prostor
  - sesalec lahko ob sesanju čistega prostora včasih tudi umaže trenutni prostor (okvara?)
- potrebna je **redefinicija prehodne funkcije**:
  - **deterministična** (do sedaj):

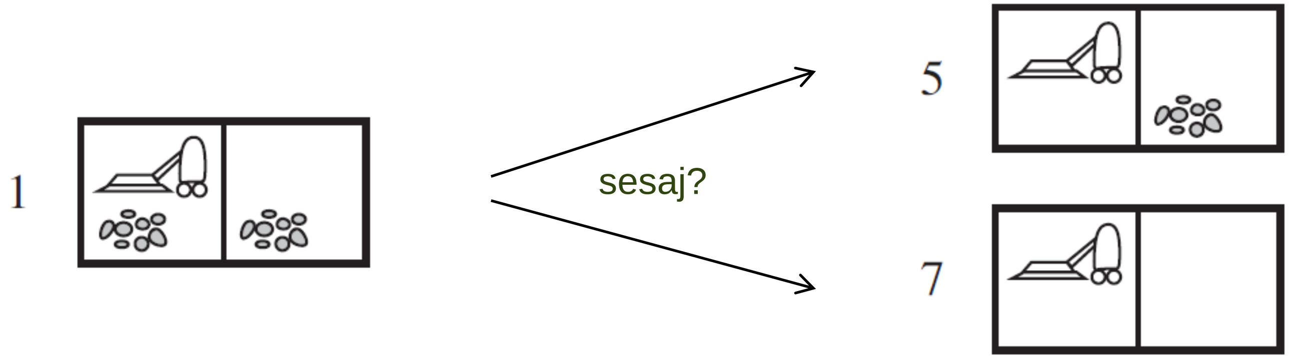
$$rezultat(trenutno\_stanje, akcija) = novo\_stanje$$

  - **nedeterministična**:

$$rezultat(trenutno\_stanje, akcija) = \{novo\_stanje1, novo\_stanje2, \dots\}$$



# Preiskovanje v nedeterminističnem okolju



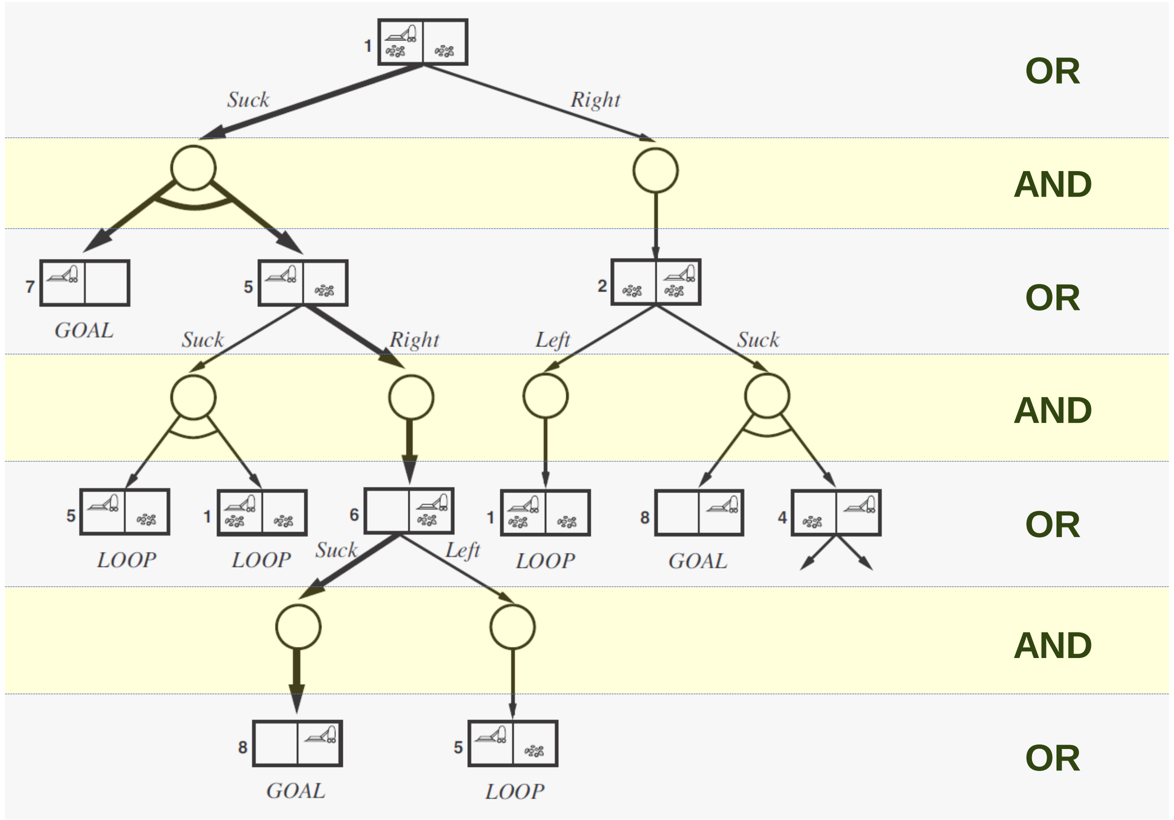

- zaporedje rešitev v prostoru z nedeterminističnimi akcijami mora upoštevati vse poti v prostoru stanj glede na možne rezultate akcij
- primer rešitve za zgornji primer:

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

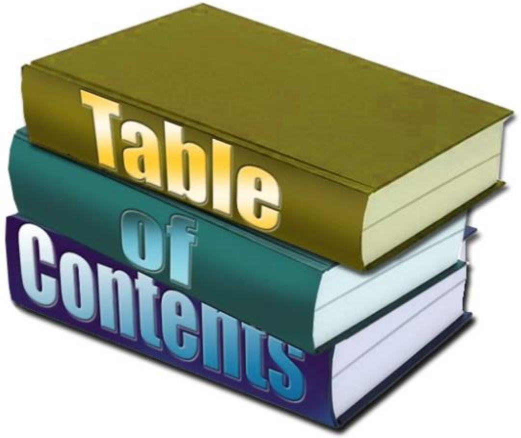
- zgornje pomeni, da rešitve problemov niso več **poti**, temveč **drevesa**

# 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 **vejanja v možna stanja**, ki so **rezultat** nedeterminističnih akcij
  - v drevesu si izmenično sledijo OR in AND nivoji
- primer (pozor, drugačna predstavitev vozlišč)



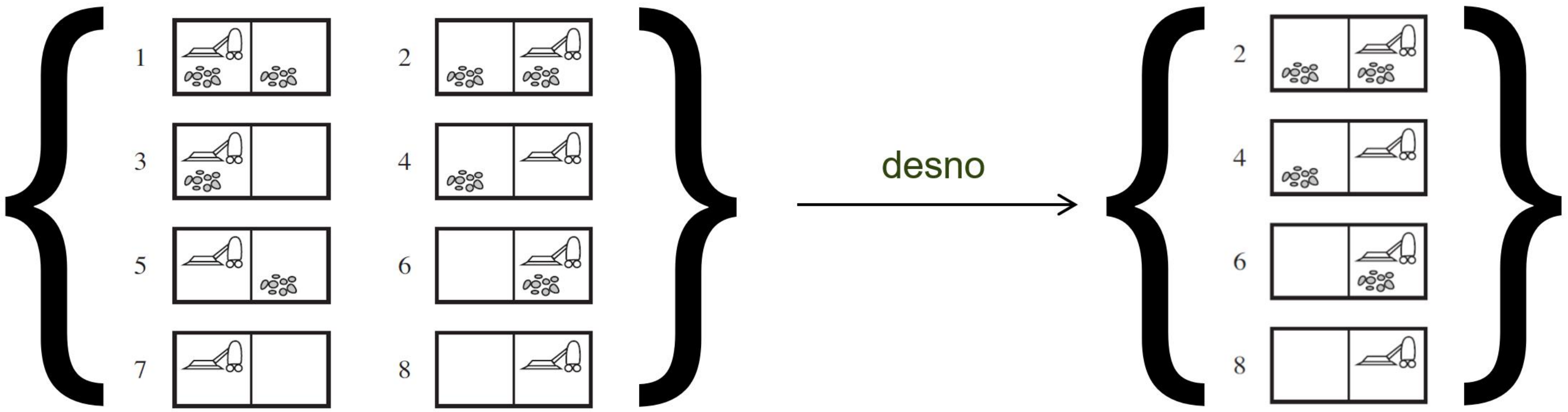
# Pregled



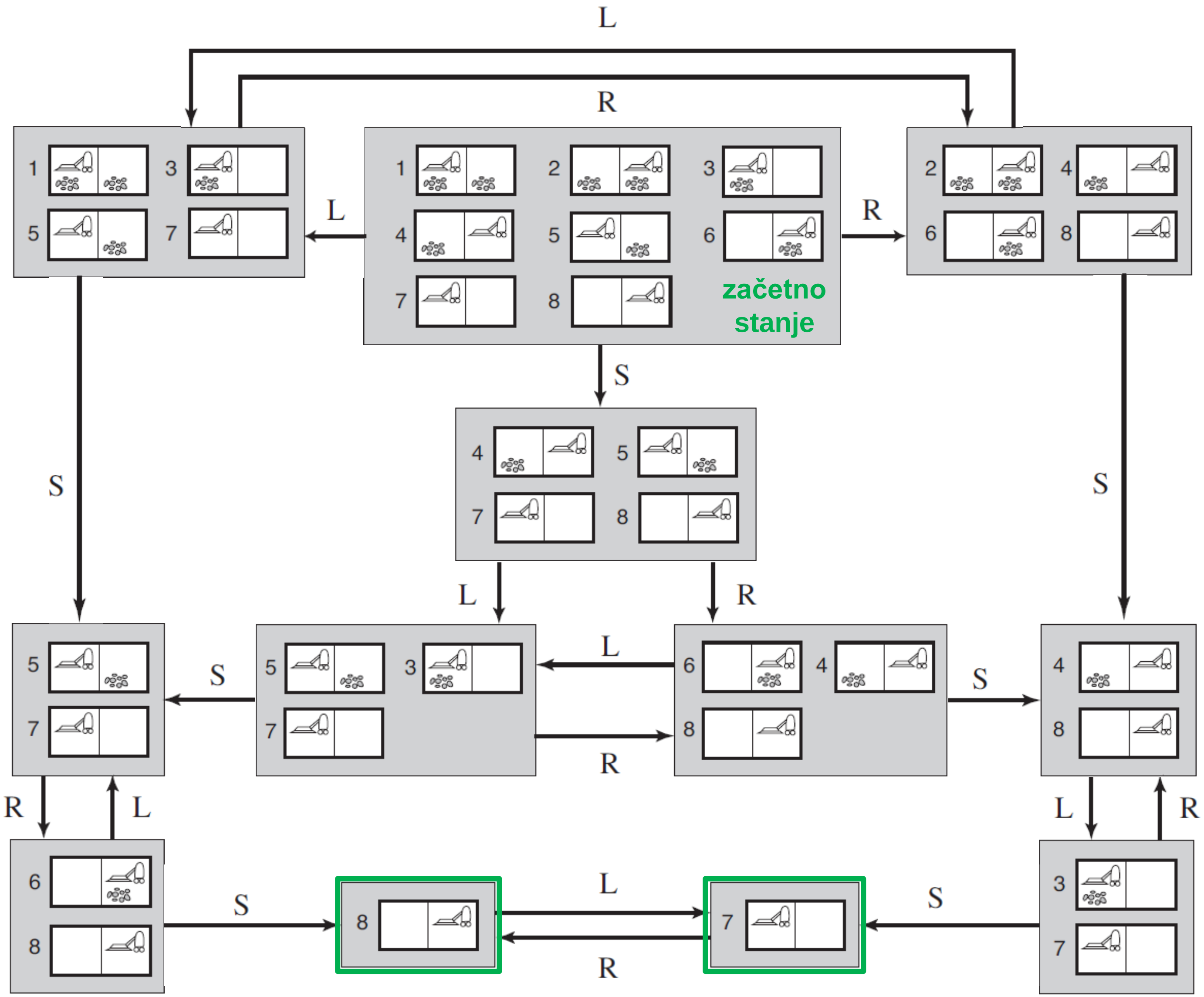

- preiskovanje
  - neinformirani preiskovalni algoritmi
  - informirani preiskovalni algoritmi
  - lokalni preiskovalni algoritmi in optimizacijski problemi
  - **preiskovanje grafov AND/OR**
    - predstavitev problemov z grafi AND/OR
    - algoritem AO*
    - preiskovanje v nedeterminističnem okolju
  - **preiskovanje brez informacije o stanju**
  - **igranje iger**
    - algoritem MINIMAX
    - rezanje alfa-beta

# Preiskovanje brez informacije o stanju

- okolja smo razdelili na **transparentna** (angl. *observable*, agent lahko zazna popolno informacijo) in **netransparentna** (brez informacije o stanju)
- 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



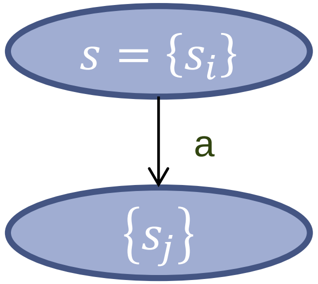
# Primer



# 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?**

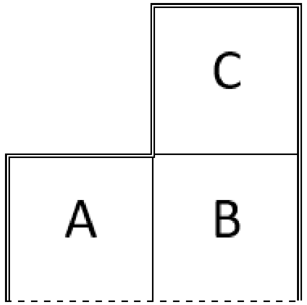


# Primer izpitne naloge

- 2. izpit, 15. 2. 2018

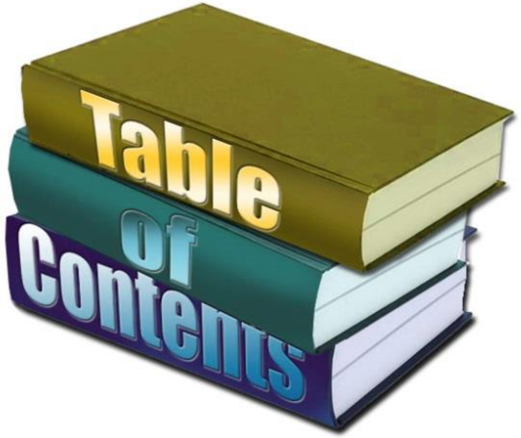
**4. NALOGA:**

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.**

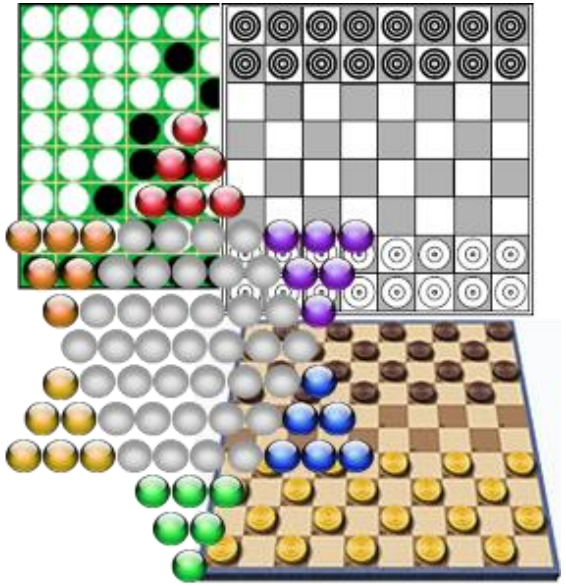
# Pregled




- preiskovanje
  - neinformirani preiskovalni algoritmi
  - informirani preiskovalni algoritmi
  - lokalni preiskovalni algoritmi in optimizacijski problemi
  - preiskovanje grafov AND/OR
    - predstavitev problemov z grafi AND/OR
    - algoritem AO*
    - preiskovanje v nedeterminističnem okolju
  - preiskovanje brez informacije o stanju
  - **igranje iger**
    - algoritem MINIMAX
    - rezanje alfa-beta

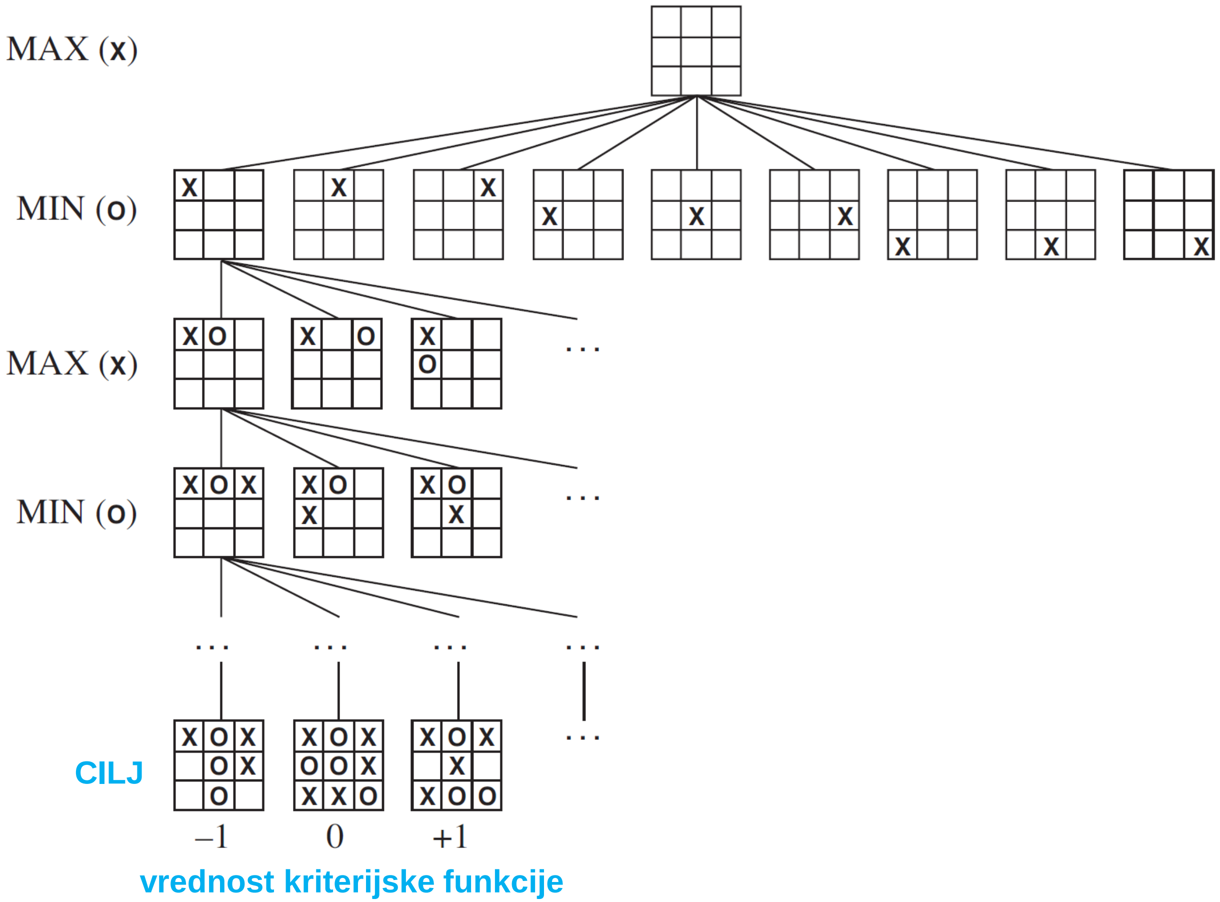
# Igranje iger

- preiskovanje prostora med **dvema nasprotnikoma** (angl. *adversarial search*)
- **več-agentno** tekmovalno okolje, kjer mora vsak agent upoštevati vpliv akcij drugega agenta na svojo uspešnost
- večina iger: deterministične, izmenične poteze, dva igralca, transparentne (s popolno informacijo)
  - primeri iger s **popolno informacijo**: šah, dama, go
  - primeri iger z **nepopolno informacijo**: potapljanje ladjic, poker, scrabble
- rešitev igre je **strategija**, ki za vsako možno potezo nasprotnika predvidi akcijo
- izziv:
  - iskanje rešitev je lahko kompleksno, velik prostor stanj
  - primer: šah ima faktor vejanja okoli 35, igra vsebuje okoli 50 potez vsakega igralca → to pomeni $35^{100}$ $(= 2{,}5 \cdot 10^{154})$ stanj



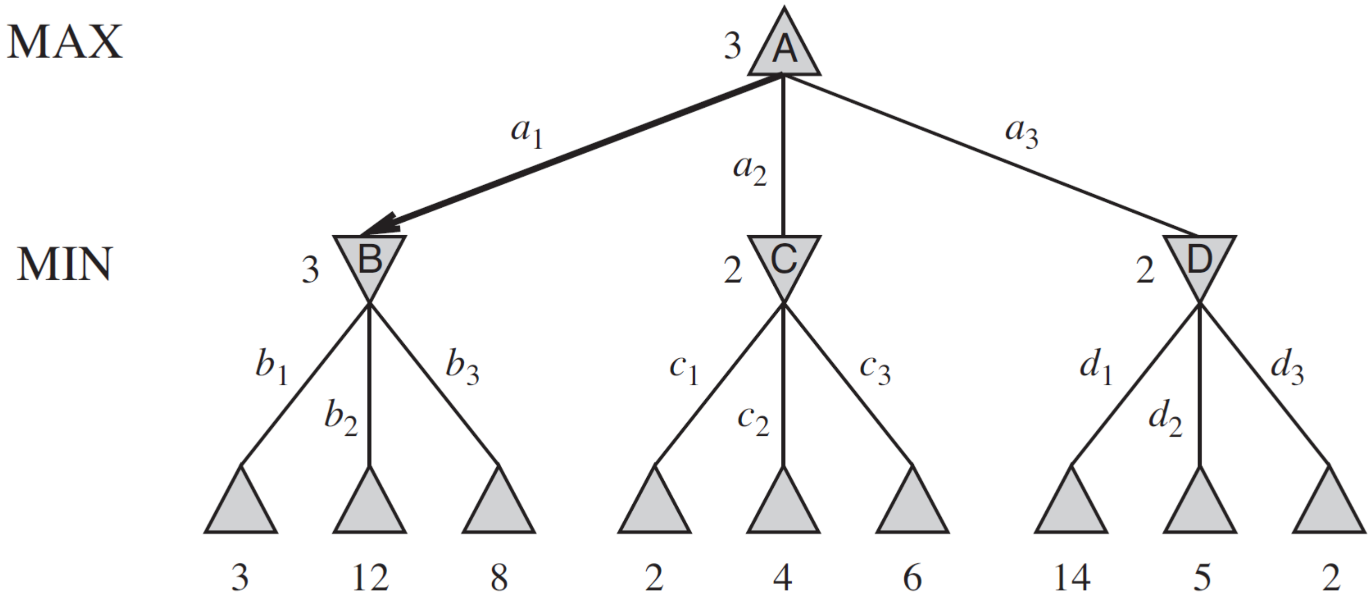
# Igranje iger

- potek igre predstavimo z **igralnim drevesom**, v katerem si poteze izmenjujeta igralca **MAX** in **MIN**
- ciljna stanja vrednotimo s **kriterijsko funkcijo** (pozitivne vrednosti so ugodne za MAX, negativne za MIN)
- dve možnosti:
  - igra s **konstantno vsoto kriterijske funkcije** (angl. zero-sum game, pozor na izraz): npr. pri šahu 1+0, 0+1, ½+½ (kompetitivna igra)
  - igra s **spremenljivo vsoto kriterijske funkcije** (angl. non-zero-sum game): kompetitivna ali nekompetitivna



# Igranje iger

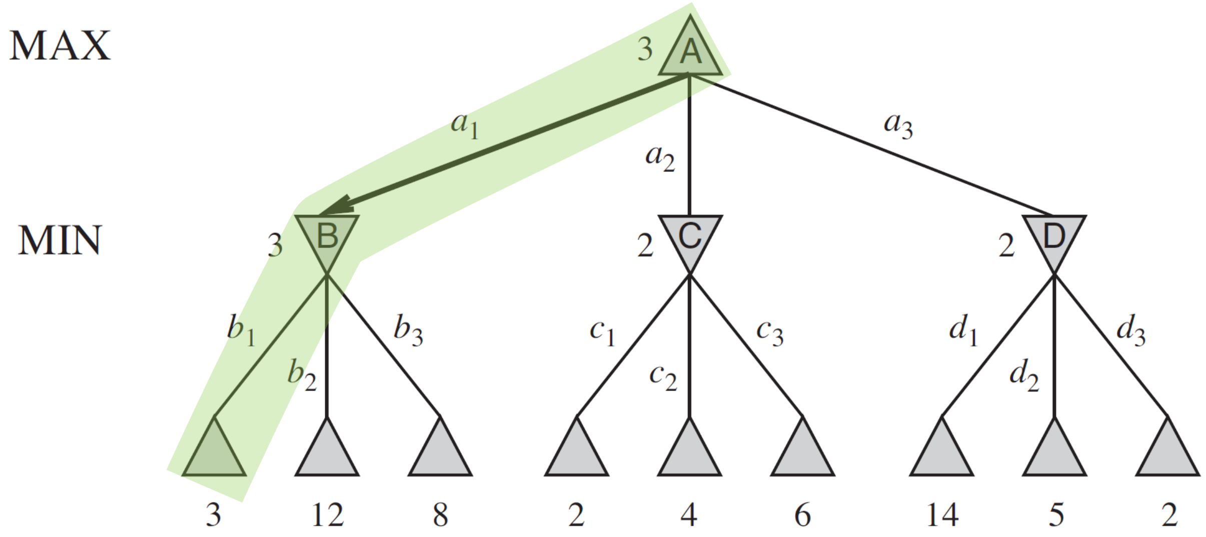
- celotna igralna drevesa so lahko velika (križci in krožci – $9! = 362.880$ ciljnih vozlišč, šah - $10^{40}$ ciljnih vozlišč)
- iskalno drevo vsebuje podmnožico vseh možnih stanj igralnega drevesa, ki razkriva dovolj informacije za izvedbo poteze
- ne zadošča iskanje končnega vozlišča, ker na pot vpliva nasprotni igralec (MIN)
- podobno predstavitvi z grafi AND/OR
  - OR: izbira poteze s strani igralca MAX
  - AND: predvideti je potrebno vse poteze nasprotnika MIN



# Igranje iger

- optimalno strategijo določa ***MINIMAX* vrednost vozlišča**, ki je enaka vrednosti kriterijske funkcije (za MAX), če **oba igralca igrata optimalno**
  - **MAX** preferira **zvišanje** vrednosti kriterijske funkcije (najboljša lastna poteza)
  - **MIN** preferira **znižanje** vrednosti kriterijske funkcije (najboljša protipoteza)
  - predpostavimo, da MIN igra optimalno

$$MINIMAX(v) = \begin{cases} kriterijska\_funkcija(v) & če\ je\ v\ končno\ stanje \\ \max_{a \in akcija(v)} MINIMAX(rezultat(v,a)) & če\ je\ igralec\ MAX \\ \min_{a \in akcija(v)} MINIMAX(rezultat(v,a)) & če\ je\ igralec\ MIN \end{cases}$$



# Algoritem *MINIMAX*

- **popolnost algoritma:**
  - da, če je prostor stanj končen (ta je definiran s pravili igre)
- **optimalnost algoritma**: da, če nasprotnik igra optimalno strategijo
  - kaj, če ne?
- **časovna zahtevnost**: $O(b^m)$
- **prostorska zahtevnost**: $O(bm)$ ali $O(m)$
  - od česa je zgornje odvisno?

- ali je potrebno preiskati celoten prostor stanj?
  - rezanje drevesa (alfa-beta rezanje)

# Igranje iger, planiranje, razporejanje opravil