

OSNOVE UMETNE INTELIGENCE

2021/22

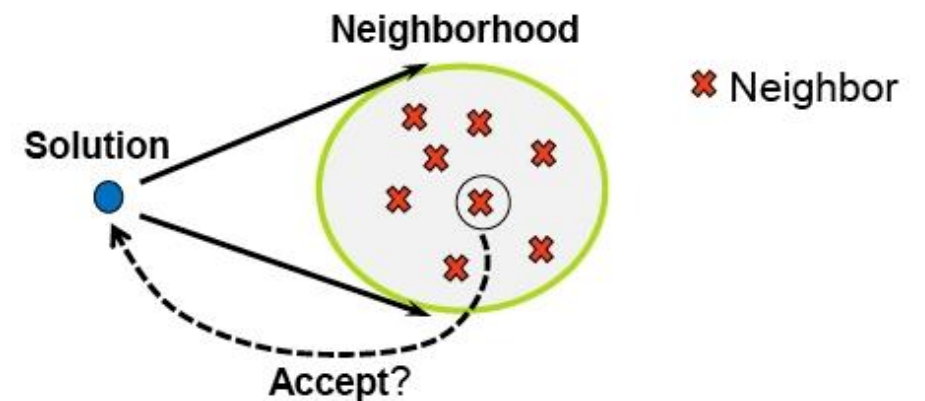
*lokalno preiskovanje
preiskovanje grafov AND/OR, AO**

Pridobljeno znanje s prejšnjih predavanj

- **informirano (hevristično) preiskovanje**
 - **požrešno iskanje**
 - vedno razvijemo najbolj obetavno vozlišče glede na hevristično oceno
 - vozlišča urejena v prioritetni vrsti
 - $f(n) = h(n)$
 - nepopoln, neoptimalen, možnost ciklanja
 - **A***
 - $f(n) = g(n) + h(n)$
 - vozlišča urejena v prioritetni vrsti
 - popoln in optimalen, če je hevristika dopustna (ne precenjuje cene do cilja)
 - **IDA***
 - podobno iterativnem poglobljanju, vendar povečujemo mejo funkcije $f(n)$
 - neučinkovito, če prevladujejo vozlišča z raznolikimi vrednostmi funkcije $f(n)$
 - IDA* razvija vozlišča v prioritetnem vrstnem redu, če je hevristika monotona
 - **kakovost hevrističnih funkcij**
 - vpliva na efektivno vejanje in število generiranih vozlišč
 - želimo dopustne hevristike s čim višjimi vrednostmi in s sprejemljivim časom izračuna
 - hevristike pridobimo: s poenostavljanjem problema, z vzorci podproblemov, z uteževanjem kriterijev

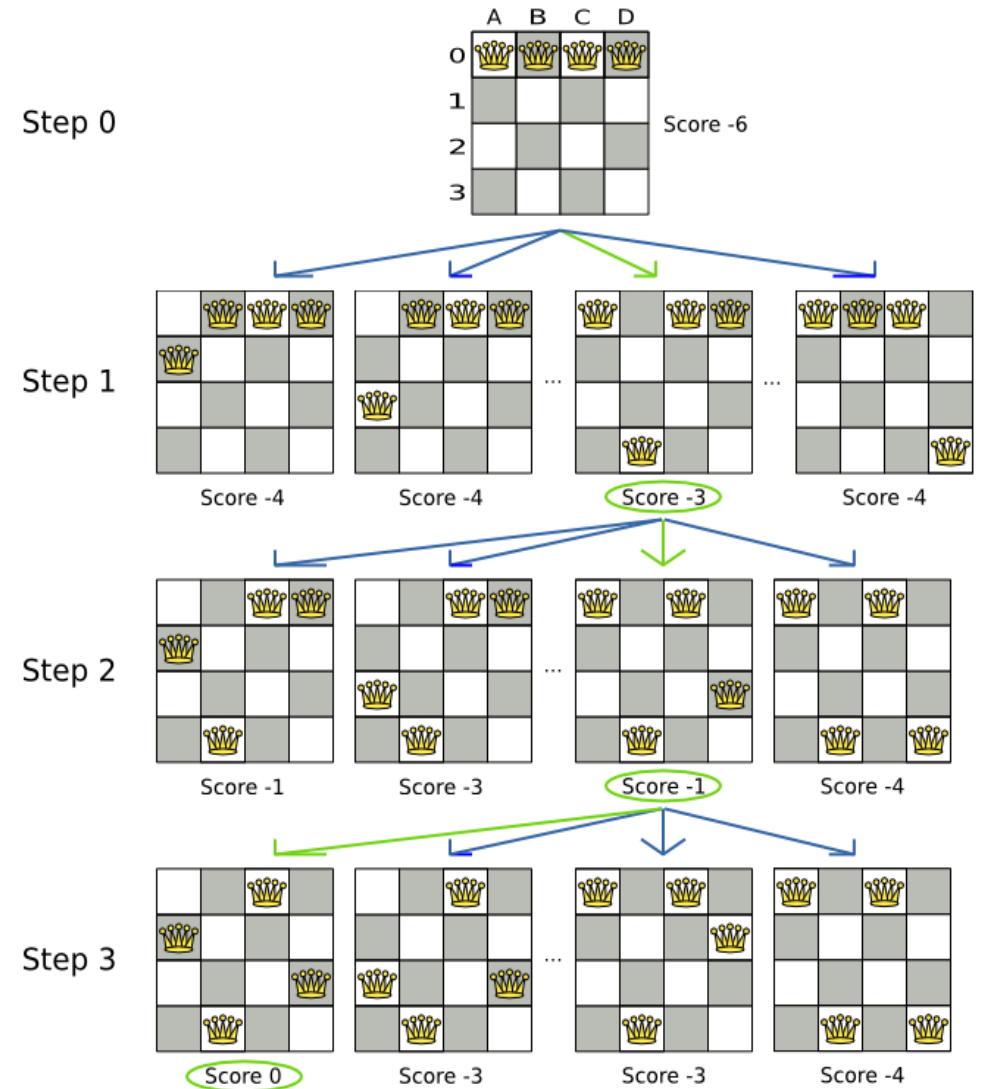
Lokalni preiskovalni algoritmi

- namesto sistematičnega preiskovanja možnih poti od začetka do cilja izvajajo **iterativno ocenjevanje** in spreminjanje podanih stanj
 - izberi **začetno** množico stanj (eno ali več njih)
 - poišči sosednja stanja od trenutnega, pri tem **ni nujno ohranjati poti**
 - ponavljaj do **ustavitvenega pogoja**
- koristni v primerih:
 - če nas primarno zanima samo kakovost rešitve (stanja) in ne nujno 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



Lokalni preiskovalni algoritmi

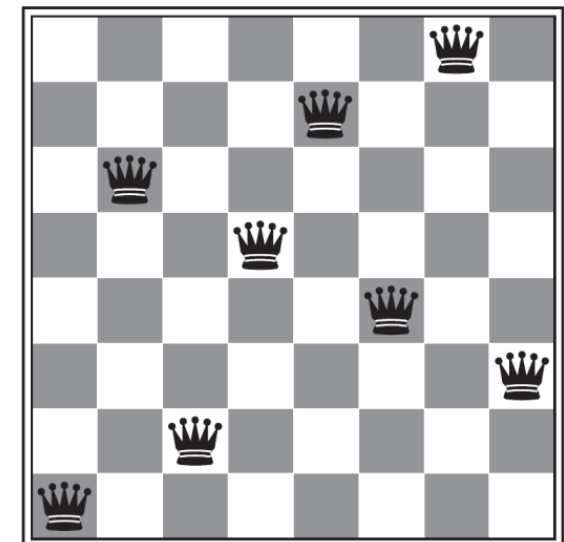
- **primer:** 4 kraljice na šahovnici
- **kriterijska funkcija:** maksimiziramo – (minus) število kraljic, ki se medsebojno napadajo
- če želimo ohraniti zaporedje korakov, ki vodijo do rešitve, je najdena **iskalna pot** pomembna, sicer pa ne; najdena iskalna pot za desni problem:
 - B0 na B3
 - D0 na D2
 - A0 na A1



Plezanje na hrib

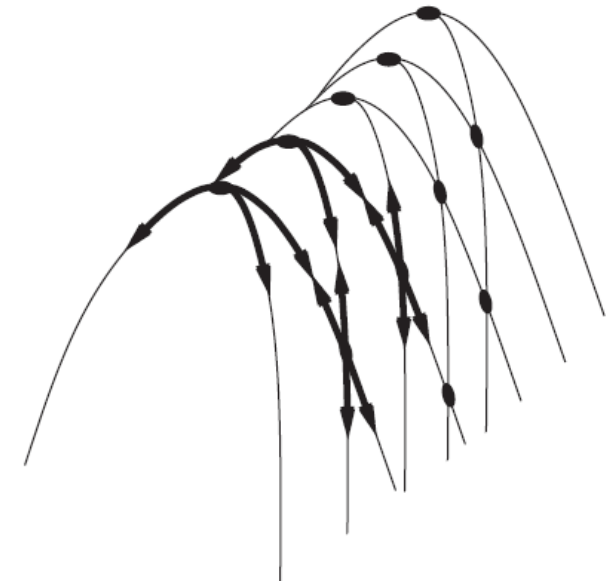
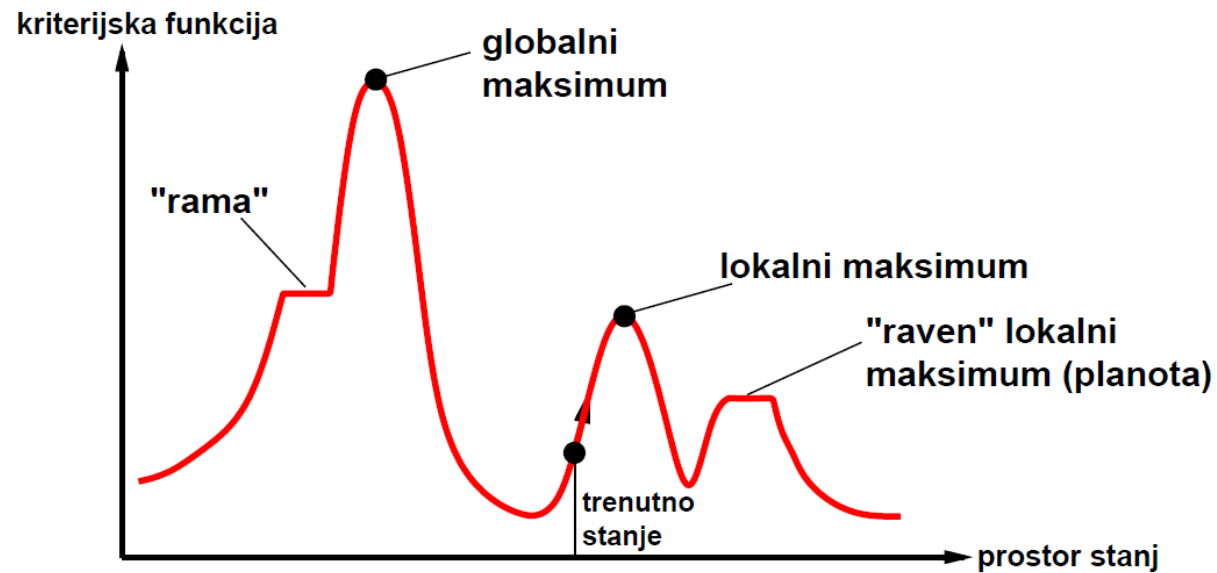
- angl. *hill-climbing search* (ali: *greedy local search*)
- premikaj se po prostoru stanj v smeri najboljše izboljšave kriterijske funkcije
- primer problema 8 kraljic:
 - kot "sosed" trenutnega stanja definiramo stanje, kjer 1 kraljico premaknemo na drugo polje znotraj istega stolpca
 - skupaj: $8 \times 7 = 56$ sosednih stanj
 - kriterijska funkcija: minimiziramo število kraljic, ki se napadajo (na zgornji sliki je prikazana vrednost kriterijske funkcije, če kraljico iz vsakega stolpca premaknemo na izbrano mesto znotraj stolpca)
 - 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)

18	12	14	13	13	12	14	14
14	16	13	15	12	14	12	16
14	12	18	13	15	12	14	14
15	14	14	♔	13	16	13	16
♔	14	17	15	♔	14	16	16
17	♔	16	18	15	♔	15	♔
18	14	♔	15	15	14	♔	16
14	14	13	17	12	14	12	18



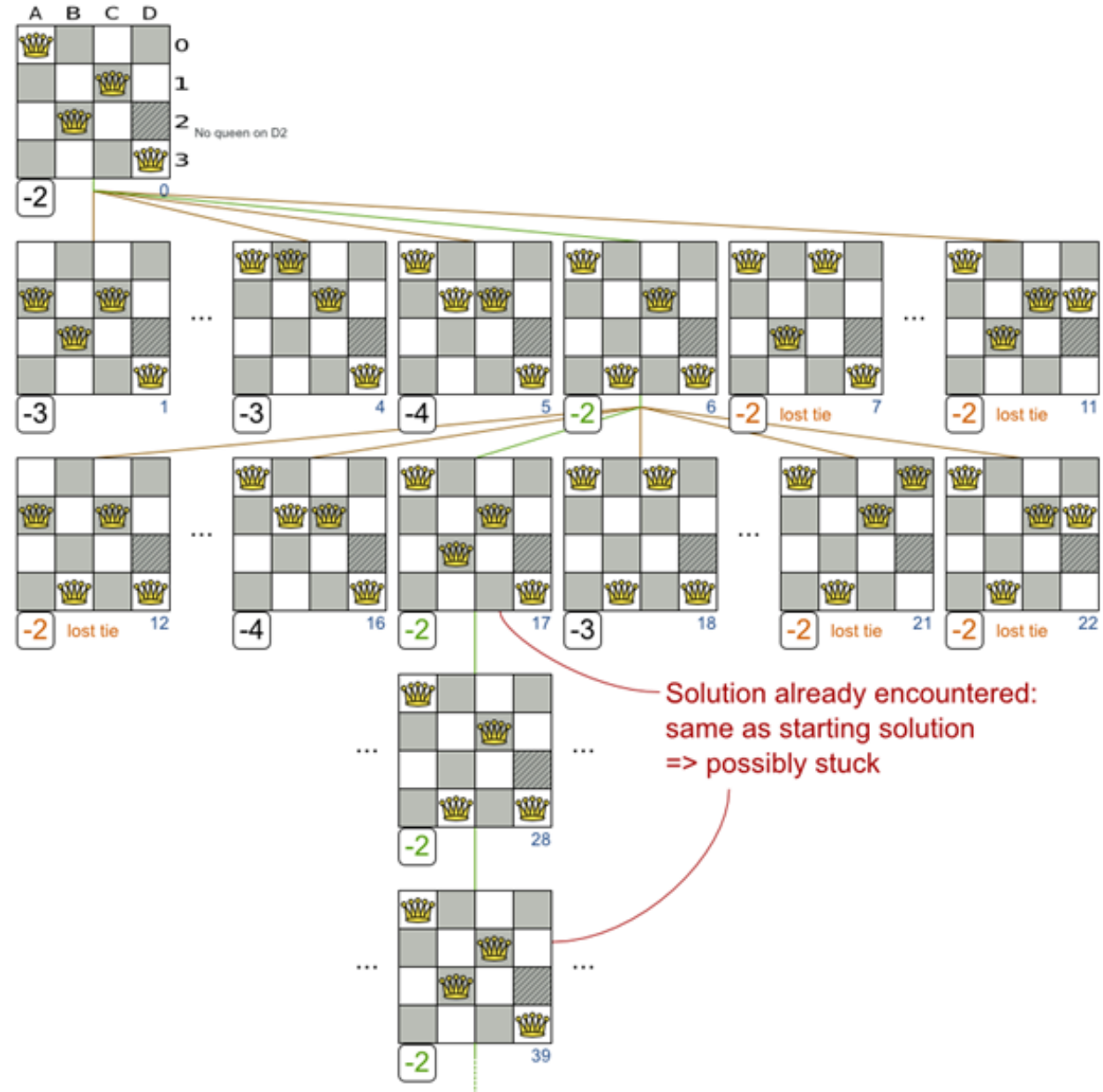
Lokalni preiskovalni algoritmi

- 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)



Plezanje na hrib

- možnost obtičanja v lokalnem maksimumu (najden cikel pri preiskovanju)



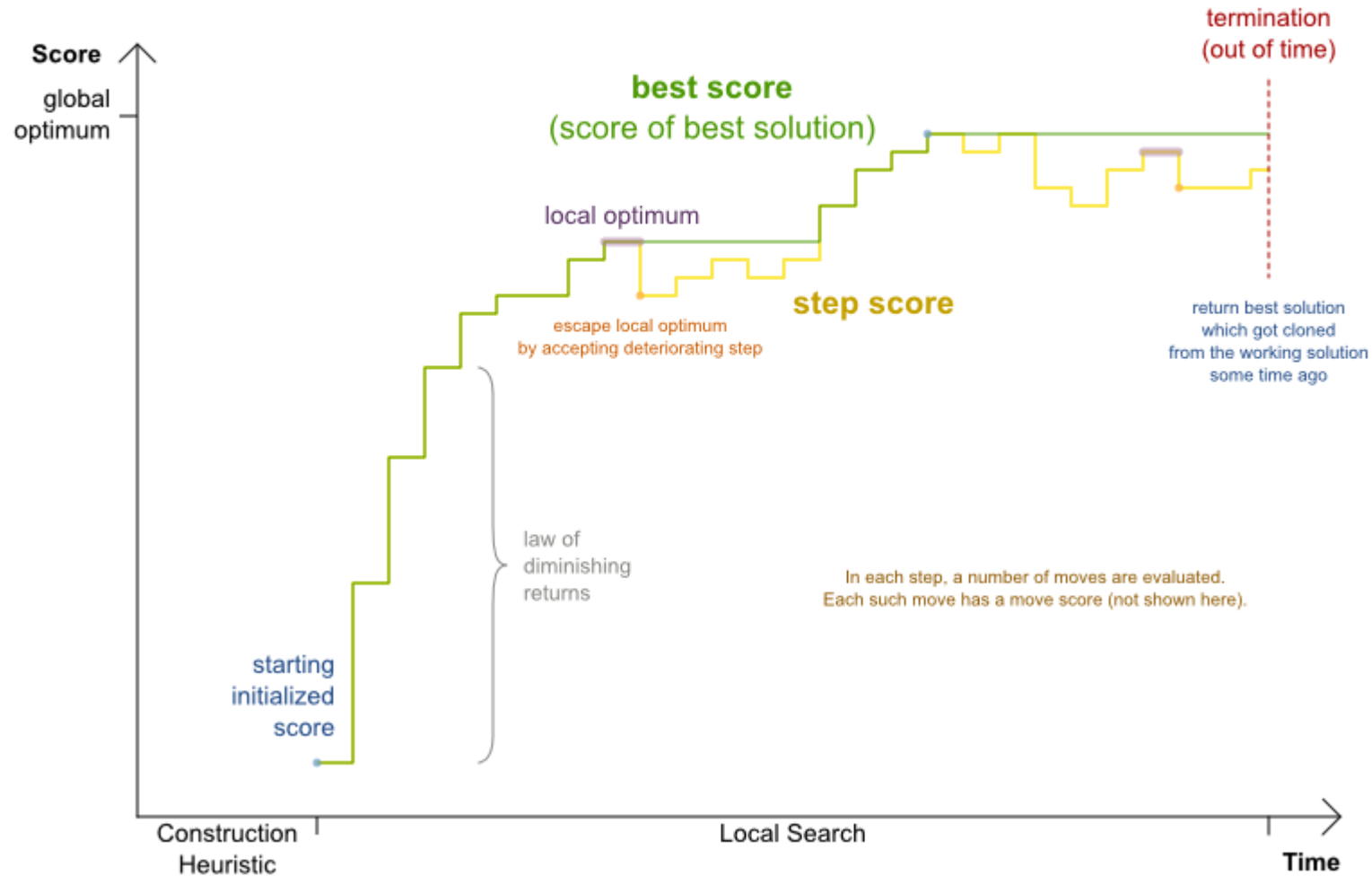
Reševanje iz lokalnih maksimumov

- **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 lahko z določeno verjetnostjo izberemo tudi slabše stanje (pri čemer upoštevamo, da imajo boljša stanja večjo verjetnost izbora)
- **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

Reševanje iz lokalnih maksimumov

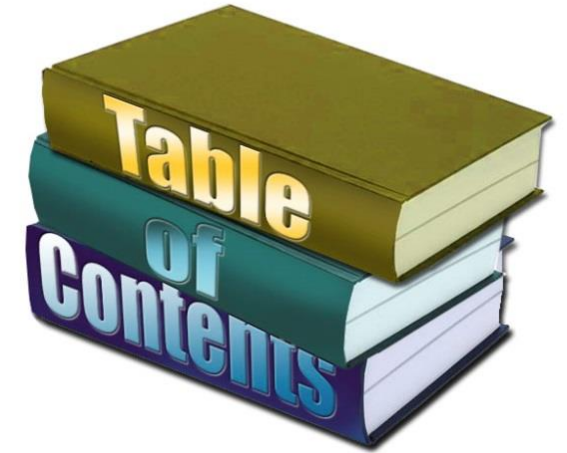
Local Search score over time

In 1 Local Search run, do not confuse starting initialized score, best score, step score and move score.



Pregled

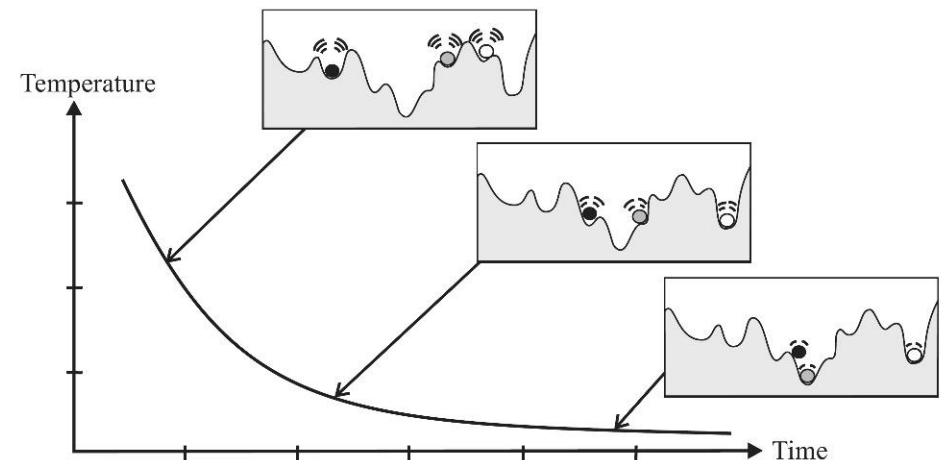
- **preiskovanje prostora stanj**
 - neinformirani preiskovalni algoritmi
 - informirani preiskovalni algoritmi
- **lokalni preiskovalni algoritmi**
 - plezanje na hrib (hill-climbing)
 - simulirano ohlajanje
 - lokalno iskanje v snopu
- **preiskovanje grafov AND/OR**
 - predstavitev problemov z grafi AND/OR
 - algoritem AO*



Simulirano ohlajanje

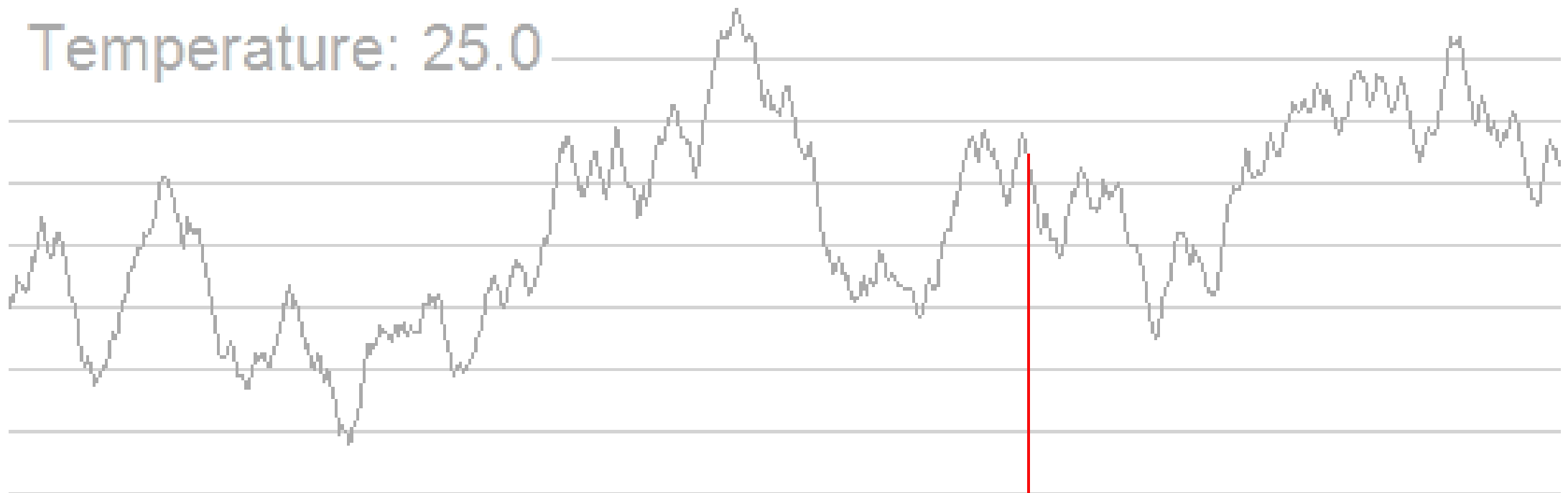
- angl. *simulated annealing*
- optimizacijski algoritem, ki izvira iz fizikalnih lastnosti v metalurgiji (ko je jeklo tekoče, so molekule v njem bolj gibljive; ko se ohlaja, se strjuje in molekule se umirjajo)
- analogija:
 - generiramo **naključne sosede** trenutnega stanja
 - če najdemo **boljše stanje, 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  $\infty$  do  
   $T \leftarrow \text{schedule}[t]$   
  if  $T = 0$  then return current  
   $\text{next} \leftarrow$  a randomly selected successor of current  
   $\Delta E \leftarrow \text{VALUE}[\text{next}] - \text{VALUE}[\text{current}]$   
  if  $\Delta E > 0$  then  $\text{current} \leftarrow \text{next}$   
  else  $\text{current} \leftarrow \text{next}$  only with probability  $e^{-\Delta E/T}$ 
```



Simulirano ohlajanje - demo

- iskanje globalnega maksimuma



- [druga simulacija](#)

Simulirano ohlajanje

- za vsako slabše vozlišče naključno odločimo, ali ga izberemo kot naslednika
- če je vozlišče boljše, ga vedno izberemo za naslednika
- sčasoma (s padcem temperature) simulirano ohlajanje preide v navadno plezanje na hrib

Temperature decreases for each step

Step 0

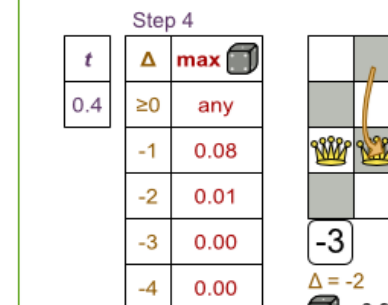
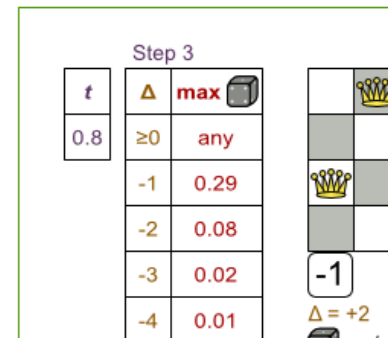
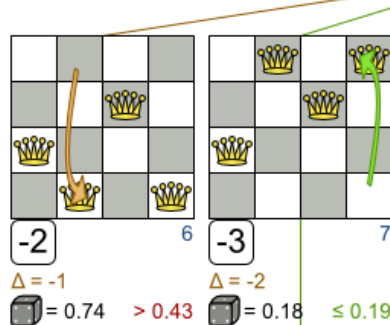
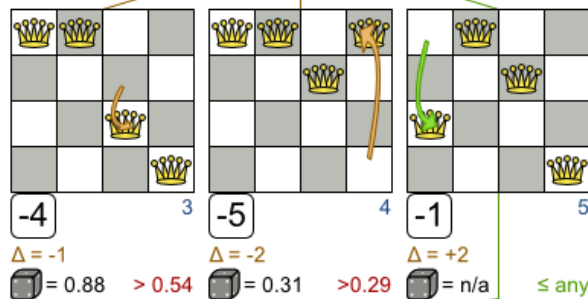
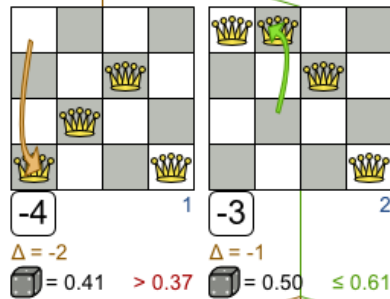
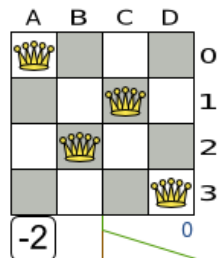
t	Δ	max
2.0	≥ 0	any
	-1	0.61
	-2	0.37
	-3	0.22
	-4	0.14

Step 1

t	Δ	max
1.6	≥ 0	any
	-1	0.54
	-2	0.29
	-3	0.15
	-4	0.08

Step 2

t	Δ	max
1.2	≥ 0	any
	-1	0.43
	-2	0.19
	-3	0.08
	-4	0.04



for $t \leftarrow 1$ to ∞ do

$T \leftarrow \text{schedule}[t]$

if $T = 0$ then return *current*

next \leftarrow a randomly selected successor of *current*

$\Delta E \leftarrow \text{VALUE}[\textit{next}] - \text{VALUE}[\textit{current}]$

if $\Delta E > 0$ then *current* \leftarrow *next*

else *current* \leftarrow *next* only with probability $e^{\Delta E/T}$

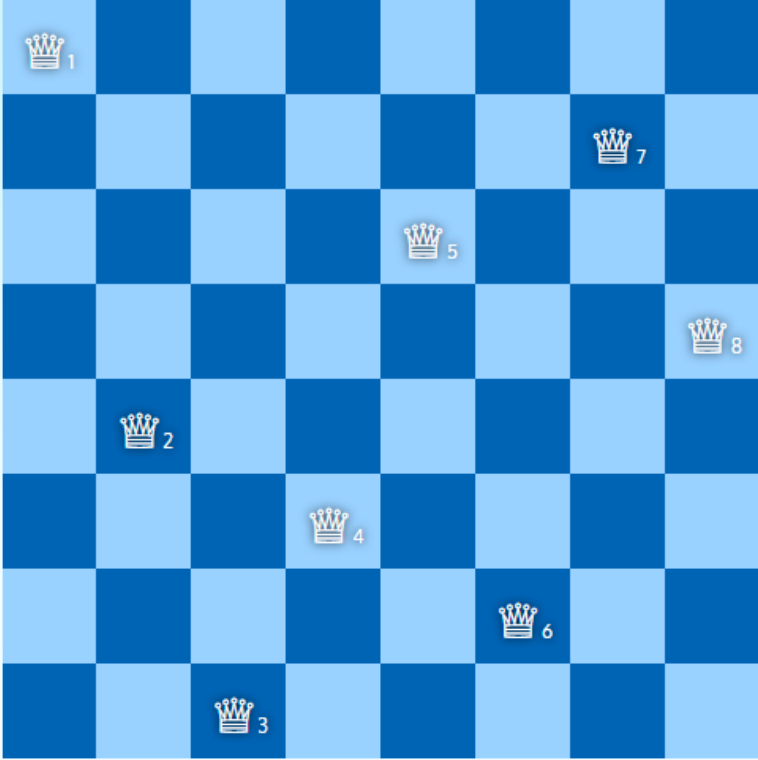
$$\text{max } \text{img alt="dice icon"} = e^{\Delta/t}$$

Demo

- <https://haseebq.com/n-queens-visualizer/>

N-Queens Visualizer

by Haseeb Qureshi



Brute force permutations

Random permutations

Backtracking

Simulated annealing

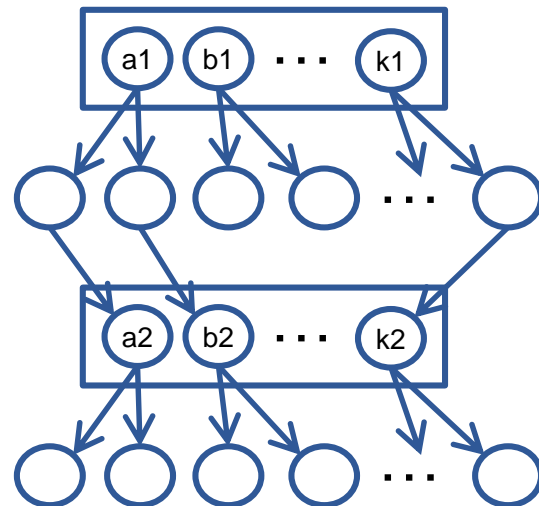
Iterative repair

Iterations:
981

1 ms 100 ms 200 ms 300 ms 400 ms 500 ms

Lokalno iskanje v snopu

- algoritem:
 - v spominu hrani **k aktualnih stanj** namesto enega
 - izberi **k najboljših stanj med sosedi** množice 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



generiraj k naključnih začetnih stanj

generiraj sosede

izberi k najboljših naslednikov

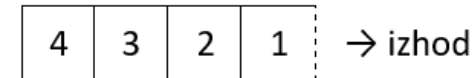
ponavljaj...

Primer izpitne naloge

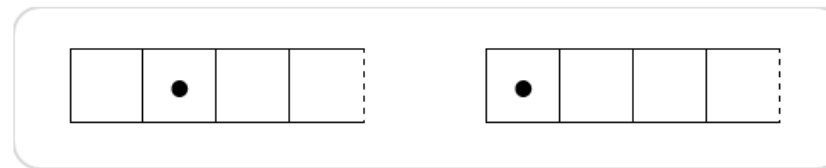
- 3. izpit, 7. 9. 2018

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 do izhoda (vrednost kriterijske funkcije za robota v vsaki posamezni sobi je prikazana na zgornji skici).

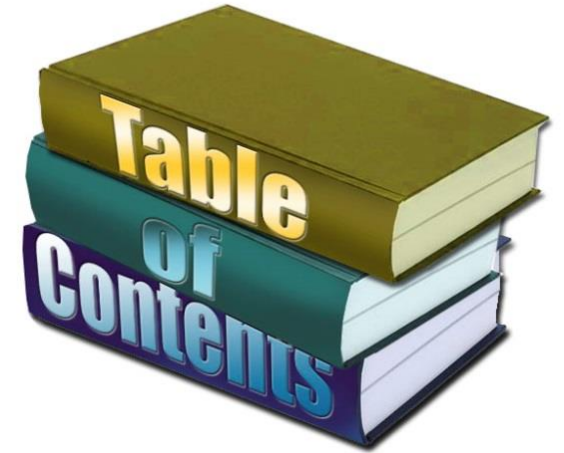


Naloge:

- (10t) Nariši zaporedje stanj pri preiskovanju, ki ga nadaljuj vse dokler ne najdemo končnega stanja (robot se premakne skozi desno (črtkano) stranico labirinta).
- (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.
- (5t) V katero družino algoritmov spada iskanje v snopu? Naštej še tri druge algoritme iz iste družine, ki jih poznaš.

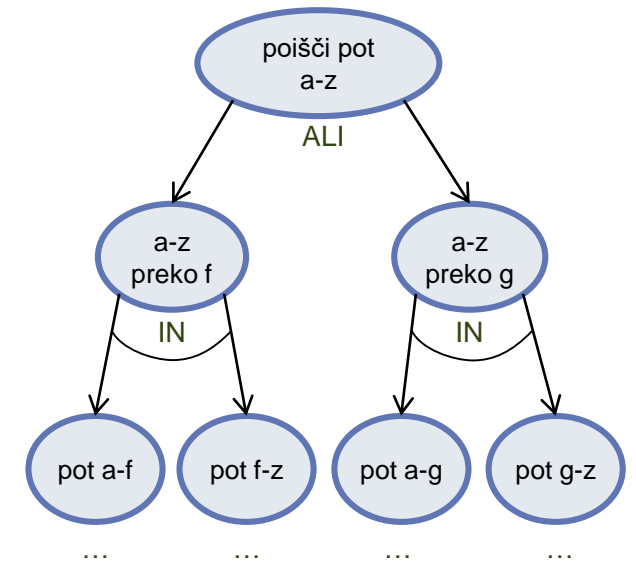
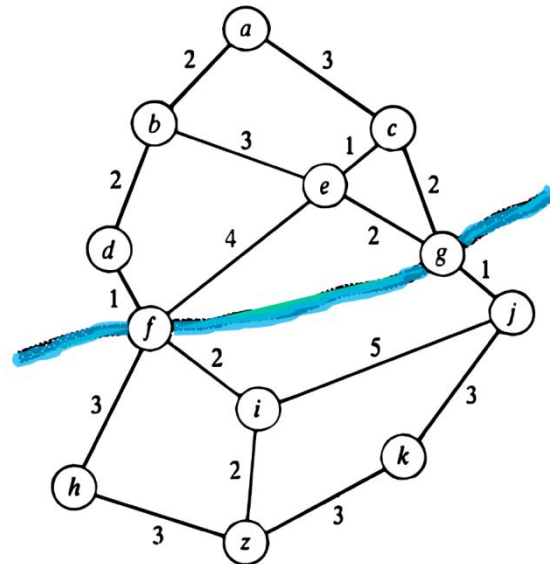
Pregled

- **preiskovanje prostora stanj**
 - neinformirani preiskovalni algoritmi
 - informirani preiskovalni algoritmi
- **lokalni preiskovalni algoritmi**
 - plezanje na hrib (hill-climbing)
 - simulirano ohlajanje
 - lokalno iskanje v snopu
- **preiskovanje grafov AND/OR**
 - predstavitev problemov z grafi AND/OR
 - algoritem AO*



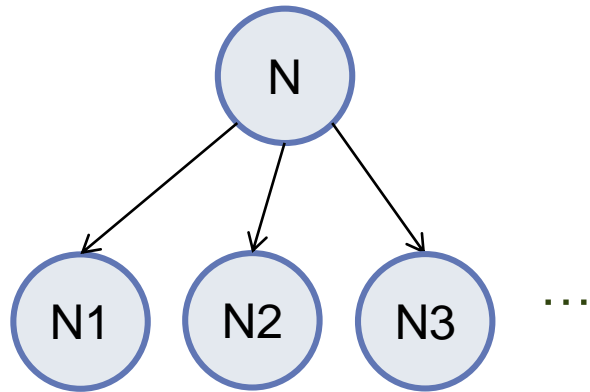
Grafi AND/OR

- 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

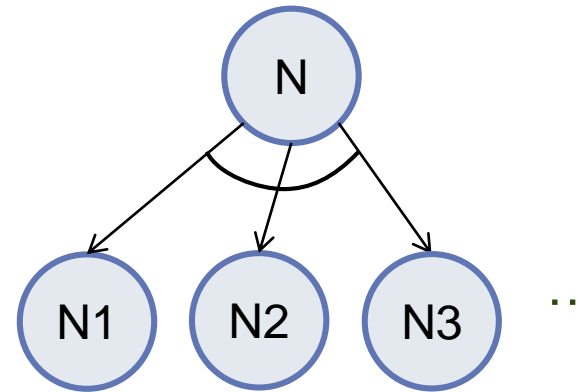


Grafi AND/OR

- 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** ...



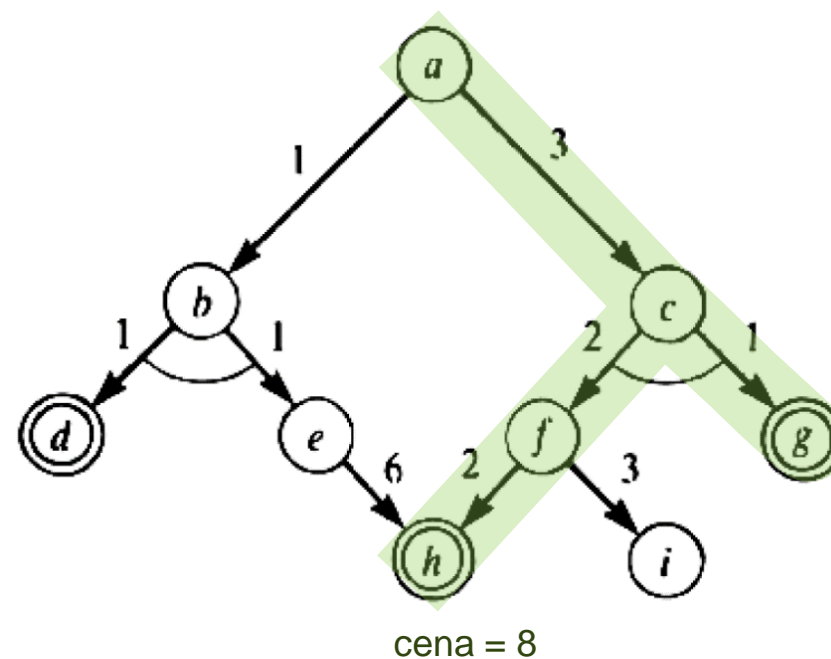
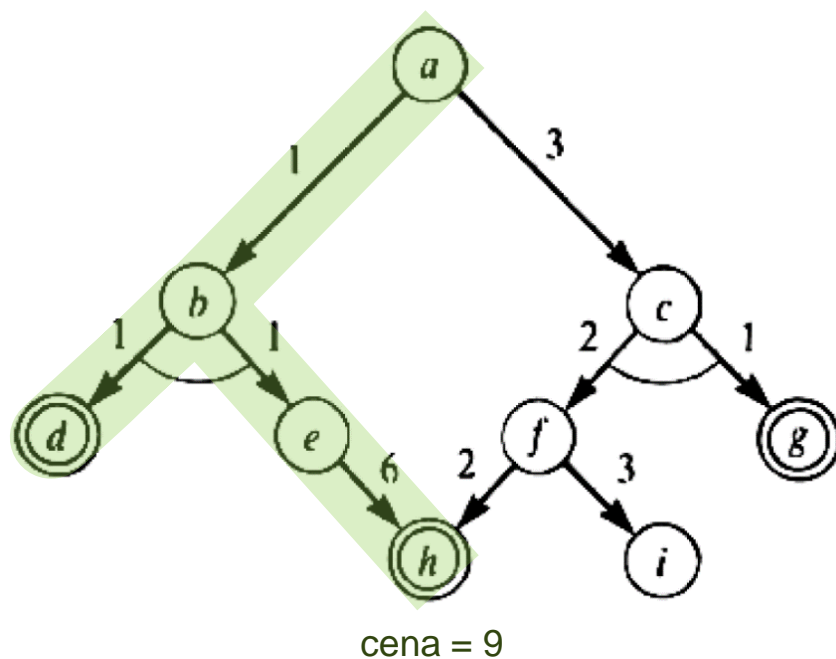
vozlišče tipa OR
(disjunkcija)



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 drevesu
- graf ima lahko različne rešitve z različnimi **cenami**

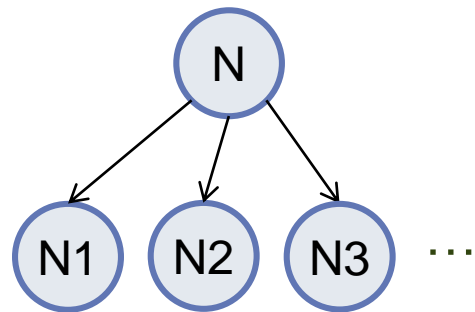
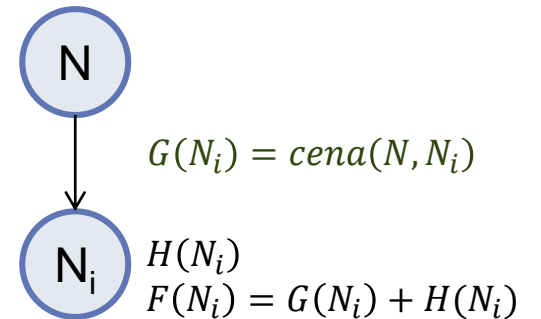


Preiskovanje grafa AND/OR

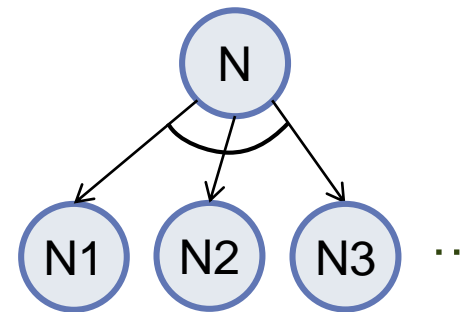
- informirana metoda – **algoritem AO***
 - posplošitev A* na grafe AND/OR
 - tudi za AO* velja, da je **popoln in optimalen**, če heuristika nikoli ne precenjuje dejanske cene do cilja
 - cena vozlišča – cena optimalnega rešitvenega drevesa

- definirajmo:**

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



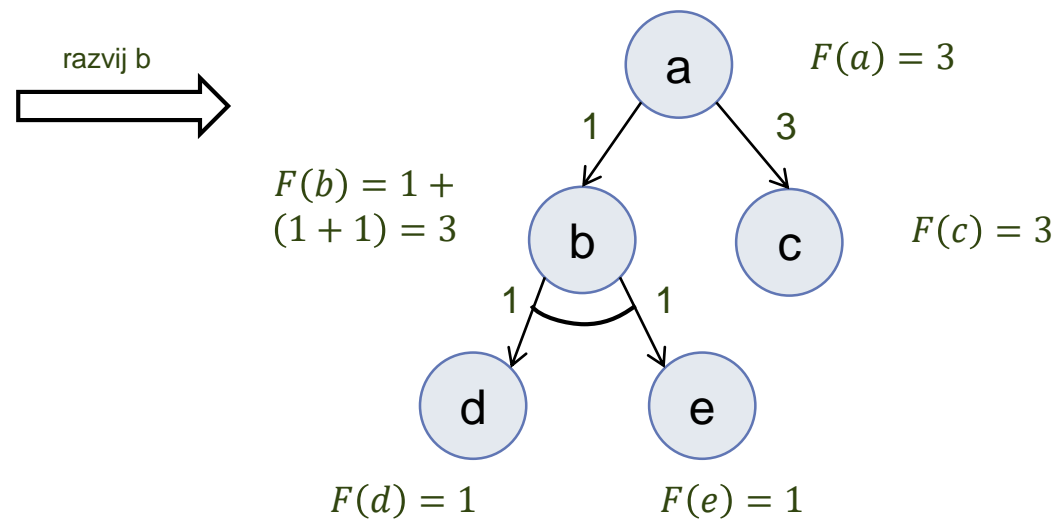
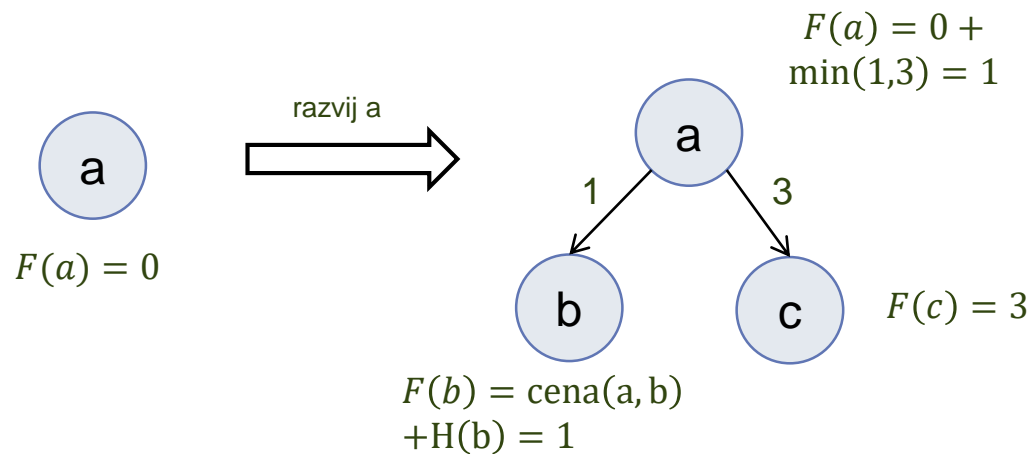
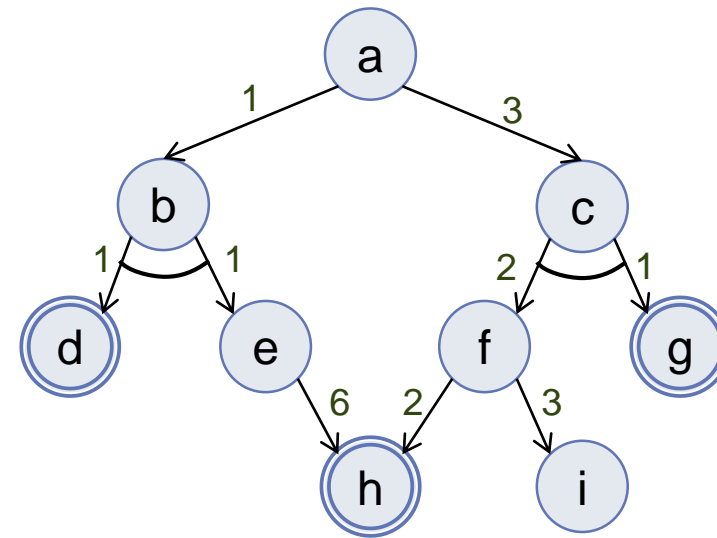
$$H(N) = \min_i (\text{cena}(N, N_i) + H(N_i))$$



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

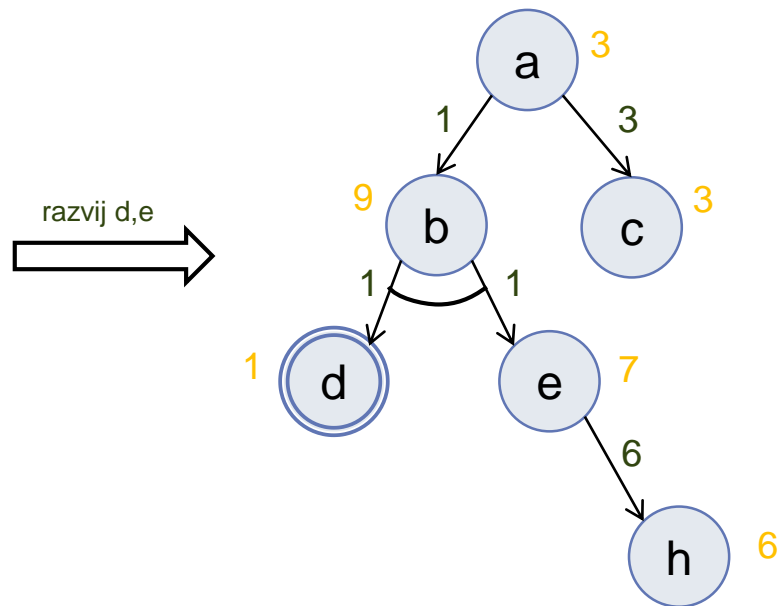
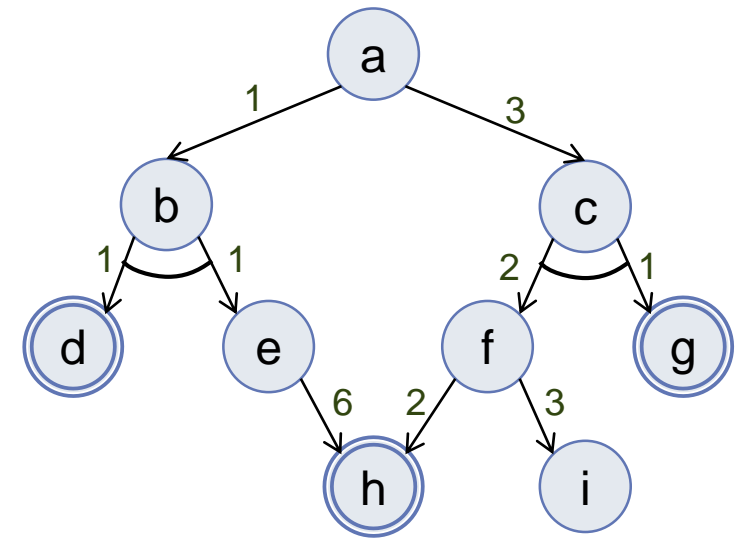
Preiskovanje grafa AND/OR

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

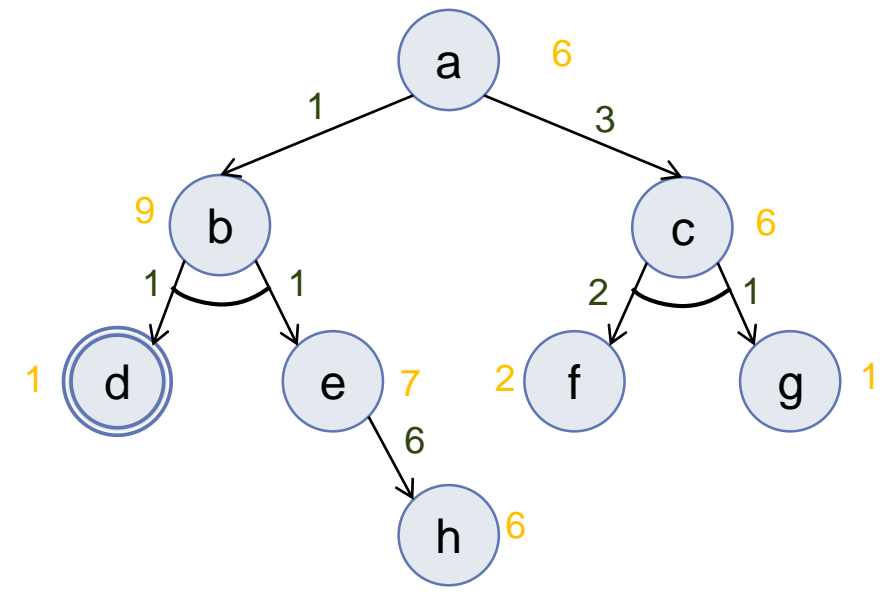


Preiskovanje grafa AND/OR

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

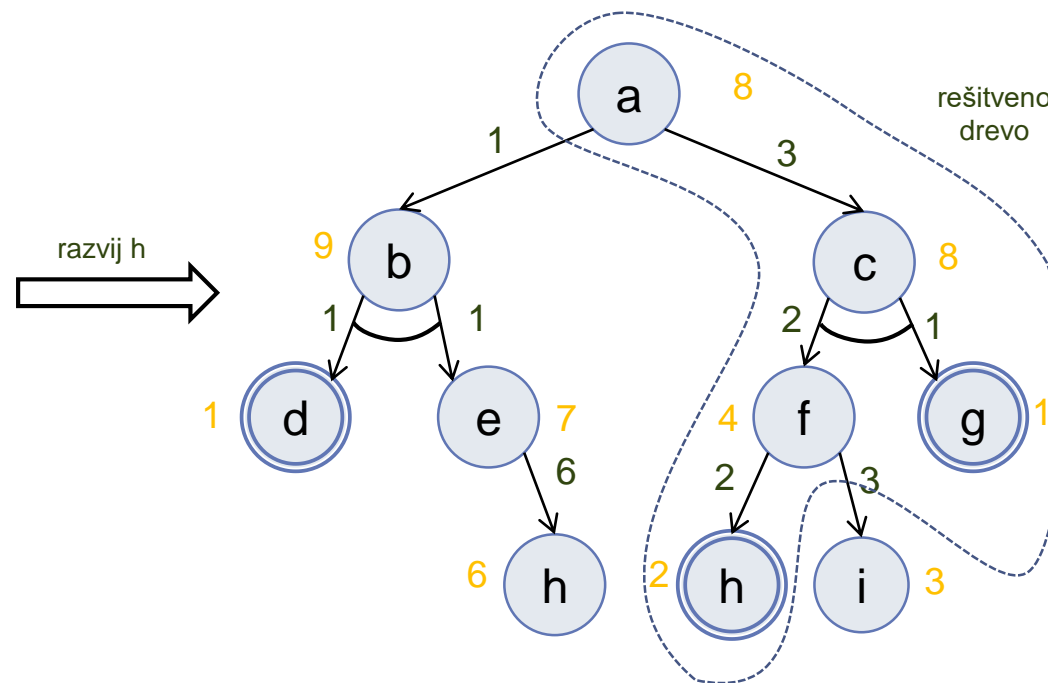
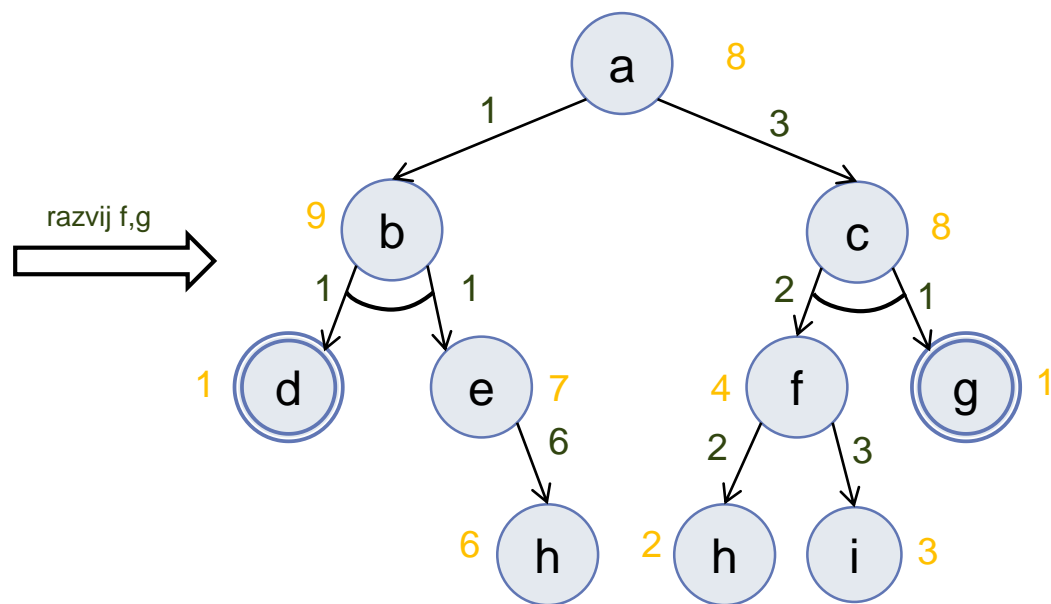
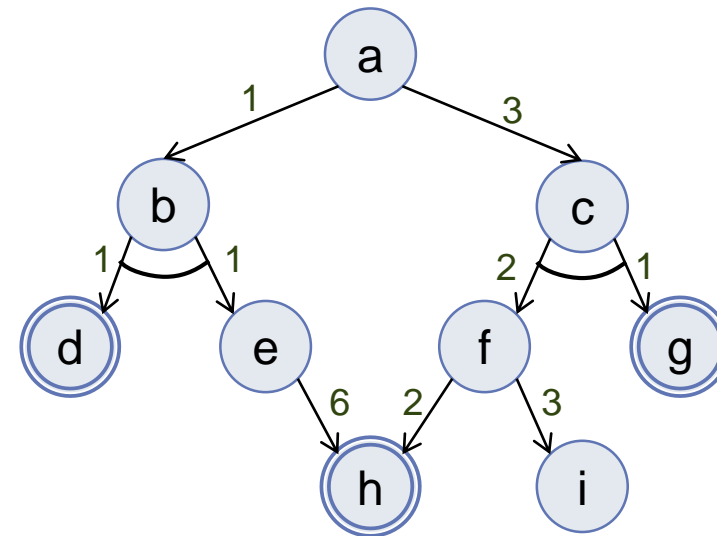


razvij c



Preiskovanje grafa AND/OR

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



Primer izpitne naloge

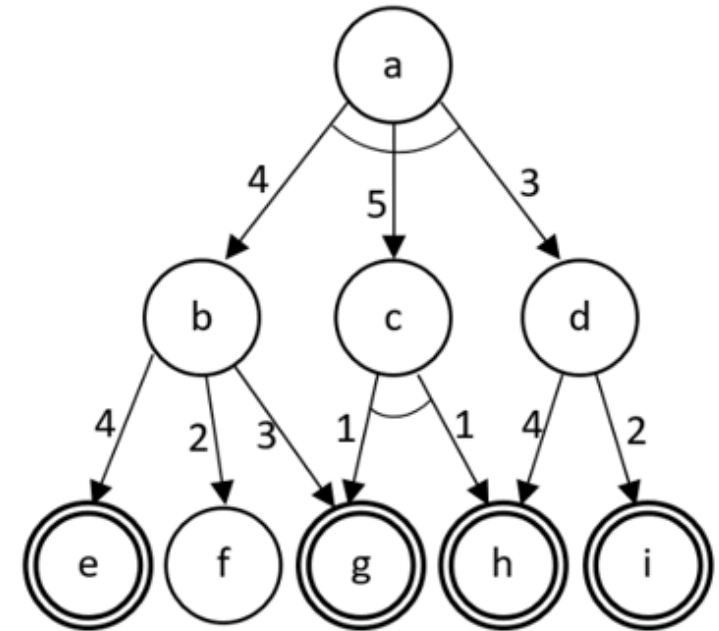
- 3. izpit, 7. 9. 2018

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

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





**Nedeterministična okolja,
okolja brez informacije o stanju, igranje iger**