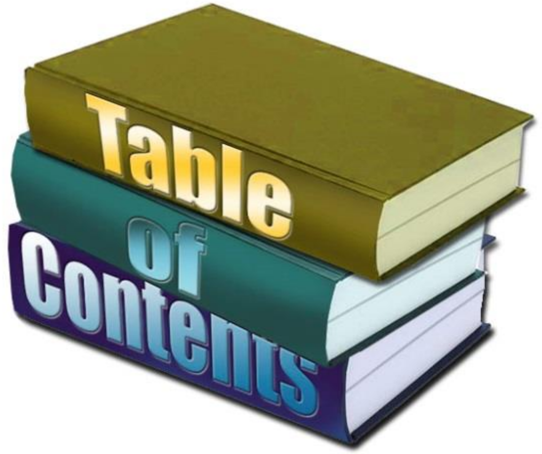
OSNOVE UMETNE INTELIGENCE
2021/22

informirani preiskovalni algoritmi

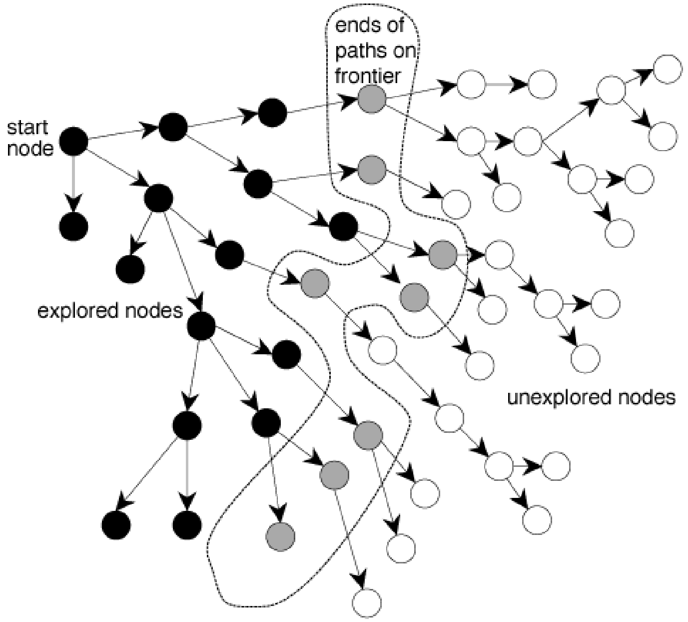
Pridobljeno znanje s prejšnjih predavanj

- **preiskovanje**
  - **iskanje v širino**
    - strategija: razvij najbolj plitvo še nerazvito vozlišče (FIFO)
    - možnosti za **detekcijo ciljnega vozlišča** (ob generiranju, ob razvijanju)
    - pomen **preprečevanja ciklov** in zaznavanje **že obiskanih** vozlišč
    - popoln, optimalen, časovna zahtevnost $O(b^d)$, prostorska zahtevnost $O(b^d)$
  - **iskanje v globino**
    - strategija: najprej razvij najglobje še nerazvito vozlišče (LIFO)
    - nepopoln, neoptimalen, časovna zahtevnost $O(b^{max})$, prostorska zahtevnost $O(bm)$
  - **iterativno poglabljanje**
    - povečevanje omejitve globine
    - popoln, optimalen, časovna zahtevnost $O(b^d)$, prostorska zahtevnost $O(bd)$
    - kombinira prednosti iskanja v širino in iskanja v globino
  - **dvosmerno iskanje**
    - potrebno poznavanje končnega stanja
    - časovna zahtevnost $O(b^{d/2})$
  - **cenovno-optimalno iskanje**
    - razvija vozlišče, ki ima najmanjšo skupno ceno dosedanje poti – $g(n)$
    - fronta urejena kot prioritetna vrsta
    - popoln, optimalen, časovna in prostorska zahtevnost $O\left(b^{1+\lfloor C^*/\epsilon \rfloor}\right)$
    - potrebna detekcija ciljnega vozlišča šele ob njegovem razvijanju

Pregled



- preiskovanje prostora stanj
  - neinformirani preiskovalni algoritmi
    - iskanje v širino
    - iskanje v globino
    - iterativno poglabljanje
    - dvosmerno iskanje
    - cenovno – optimalno iskanje
  - informirani preiskovalni algoritmi
    - hevristično preiskovanje (primer)
    - požrešno preiskovanje
    - A*
    - IDA*
    - kakovost hevrističnih funkcij



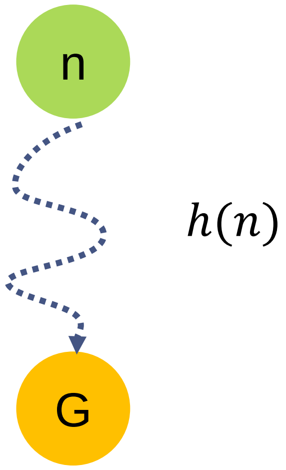
Hevristično preiskovanje

- potreba po usmerjanju iskanja z motivacijo, da hitreje/lažje najde optimalno rešitev
- ideja: uporabimo **oceno vozlišč** (stanj), ki ocenjujejo obetavnost za doseganje do cilja
- **hevristika** (ali hevristična ocena, ugibanje) je **ocenitvena funkcija za obetavnost vozlišča** (najcenejše poti iz vozlišča do najbližjega cilja)

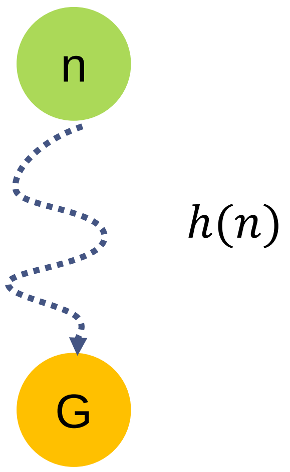
$$h: vozlišče \rightarrow \mathbb{R}$$

- izberemo in razvijemo vozlišče glede na **najboljšo vrednost hevristike**
  - nizek h nakazuje **bolj obetavno** vozlišče, višji h pa **manj obetavno** (težji problem)
  - fronta hrani vozlišča, urejena v **prioritetni vrsti po obetavnosti**

- primeri hevrističnih preiskovalnih algoritmov:
  - A*
  - IDA* (*iterative deepening A**)
  - RBFS (*recursive best-first search*)
  - iskanje v snopu (*beam search*)
  - plezanje na hrib (*hill climbing*)



Požrešno iskanje

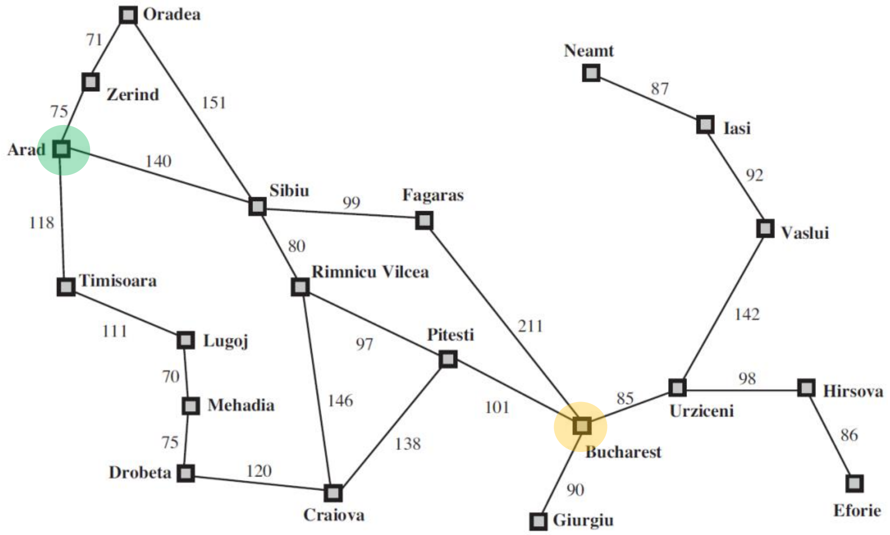


- angl. *greedy best-first search*
- vedno razvijemo najbolj obetavno vozlišče **glede na hevristično oceno**
- vrednotenje vozlišča: $f(n) = h(n)$

- primer: iskanje optimalne poti z upoštevanjem najkrajše zračne razdalje
- zračne razdalje do cilja (Bukarešta) lahko uporabimo kot hevristične ocene:

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

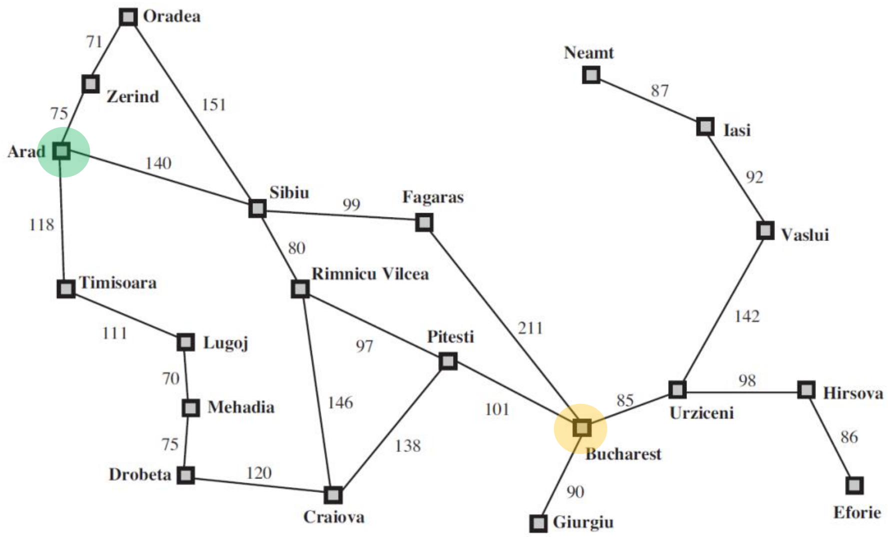
Požrešno iskanje: primer



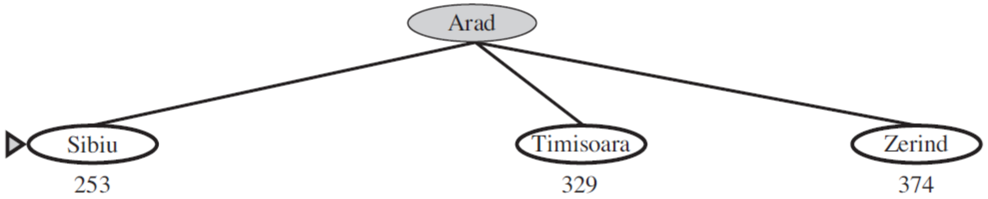
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

hevristična ocena

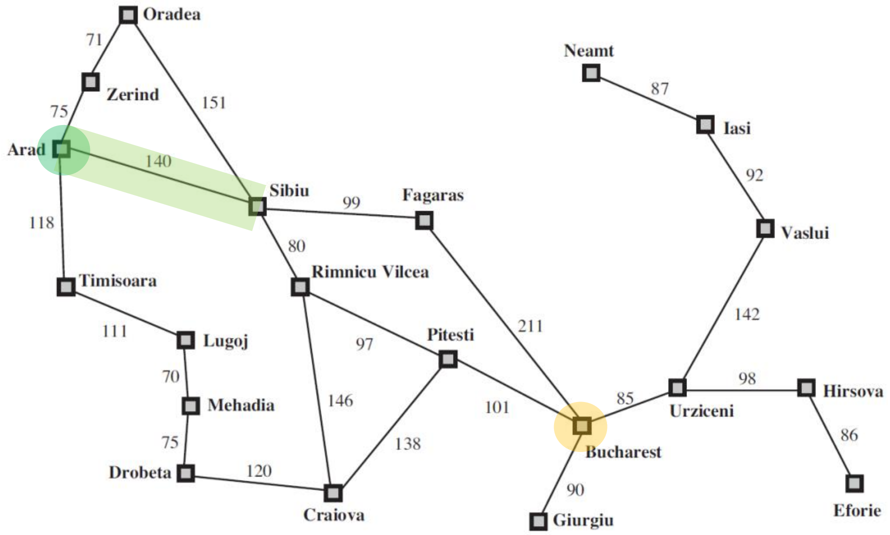
Požrešno iskanje: primer



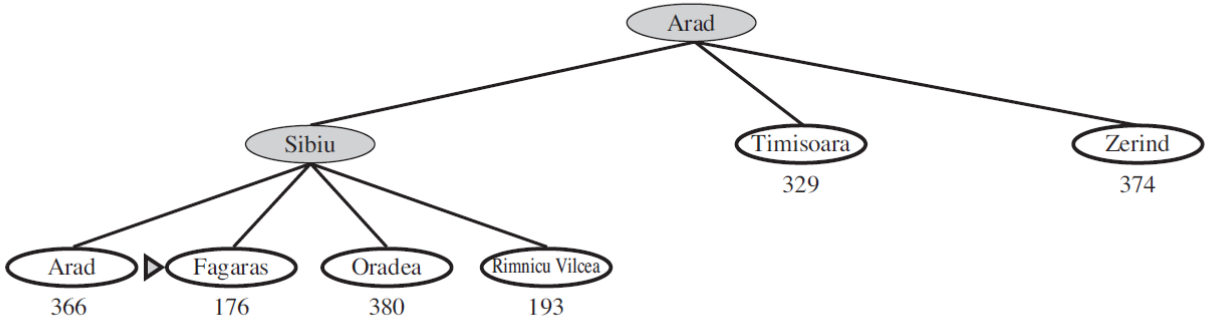
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374



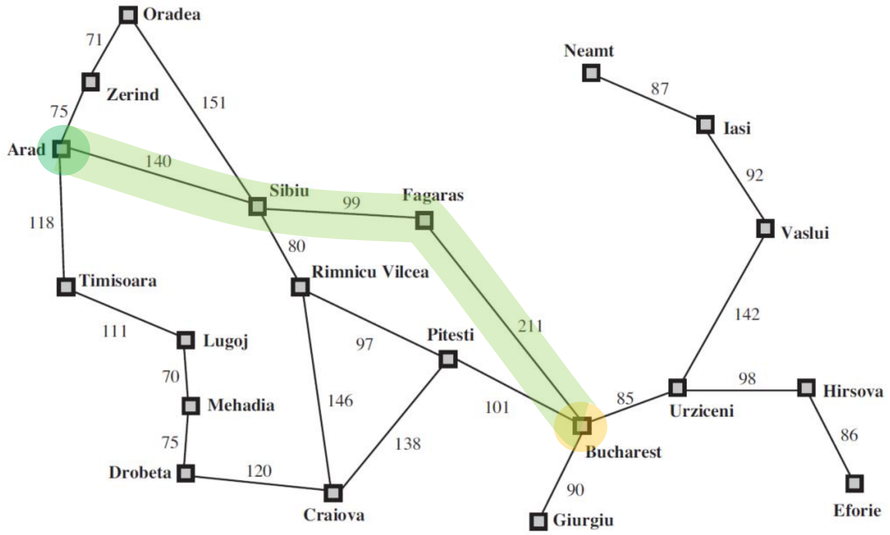
Požrešno iskanje: primer



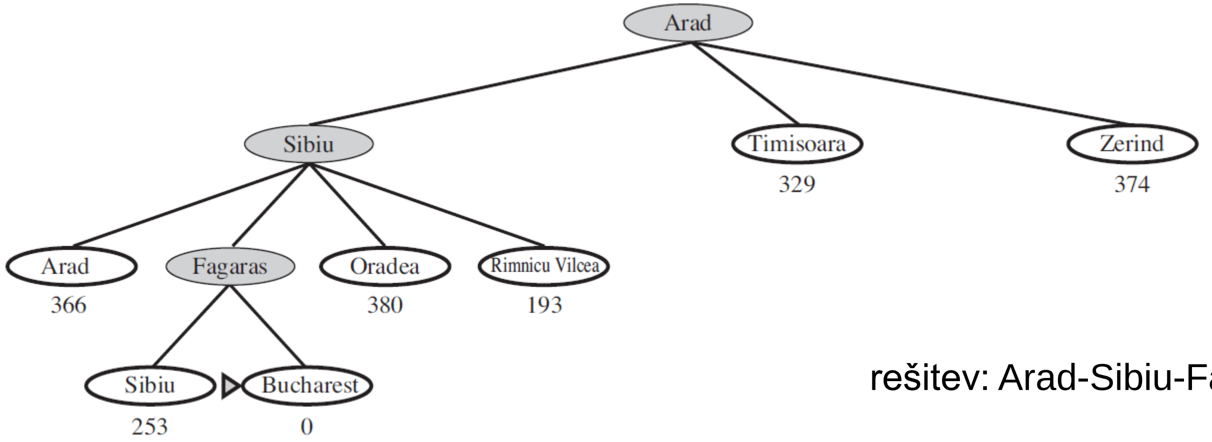
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



Požrešno iskanje: primer



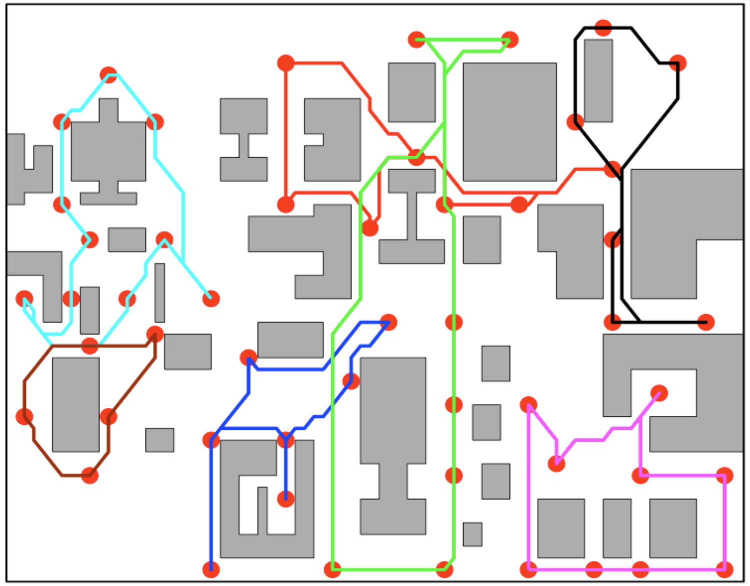
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

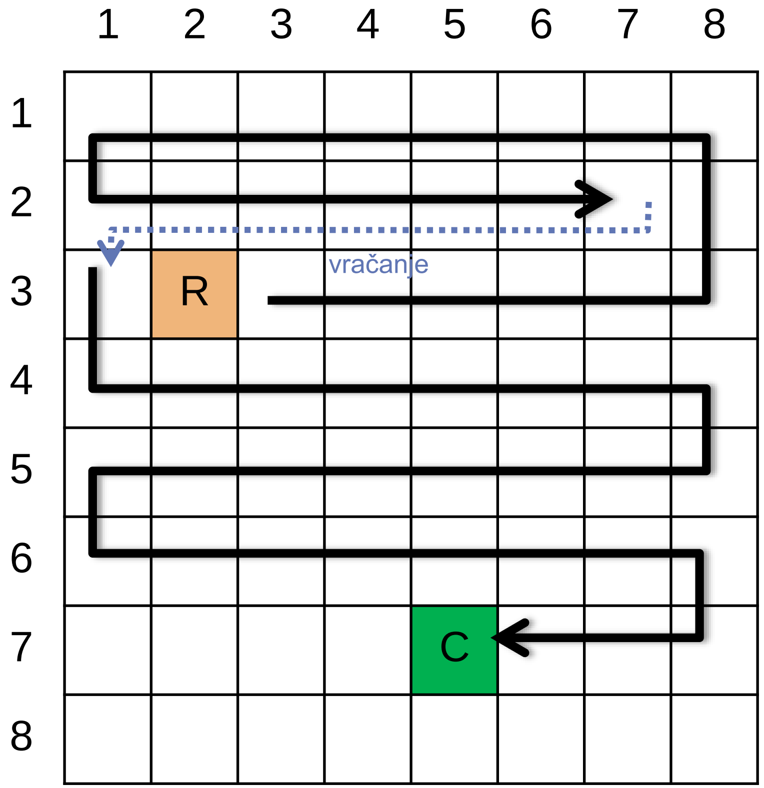


rešitev: Arad-Sibiu-Fagaras-Bucharest, cena=450

Požrešno iskanje: primer

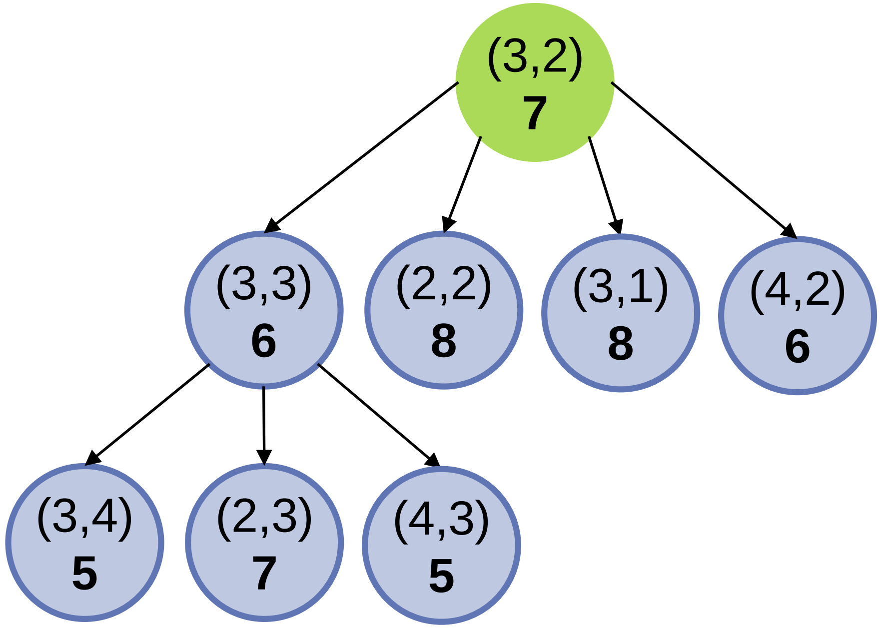
- primer 2: robotovo iskanje ciljne lokacije (Bratko, OUI, 2016/17)
- R možni premiki: 1 →, 2 ↑, 3 ←, 4 ↓
- dolžina rešitve pri **preiskovanju v globino** je: 45
- optimalna dolžina rešitve: 7



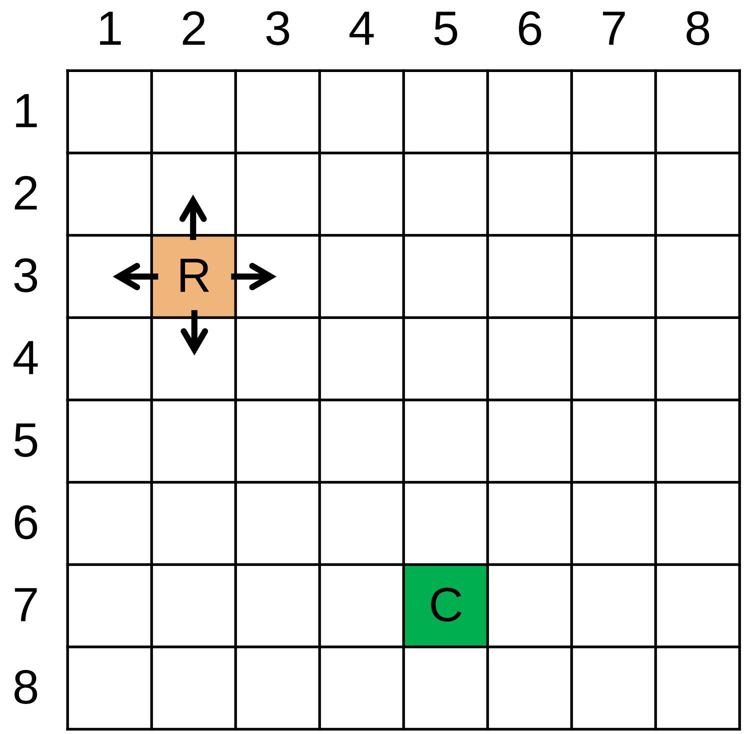


Požrešno iskanje: primer

- ideja: uporabimo hevristiko, ki ocenjuje obetavnost koordinate za doseganje do cilja - npr. manhattanska razdalja
- izberemo in razvijemo vozlišče glede na najmanjšo ocenjeno oceno (hevristiko)



itd.



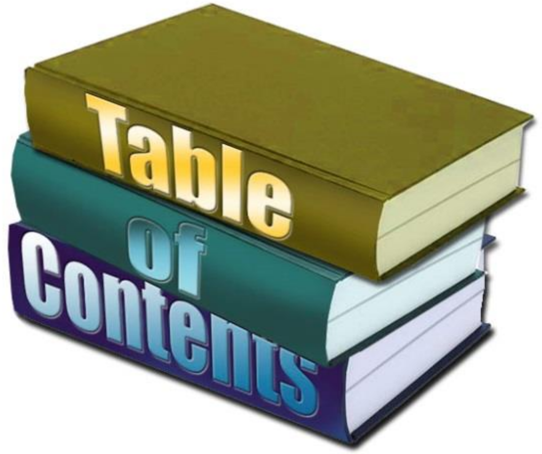
Učinkovitost požrešnega iskanja

- **POPOLNOST** (angl. *completeness*):
  - Ne.
  - Možnost ciklanja v lokalnih delih grafa (primer: pri iskanju poti Iasi->Fagaras se lahko zaciklamo med Iasi⇔Neamt).

- **OPTIMALNOST** (angl. *optimality*):
  - Ne.
  - Ali obstaja v našem primeru iskanja najkrajše poti bolj optimalna pot do cilja?

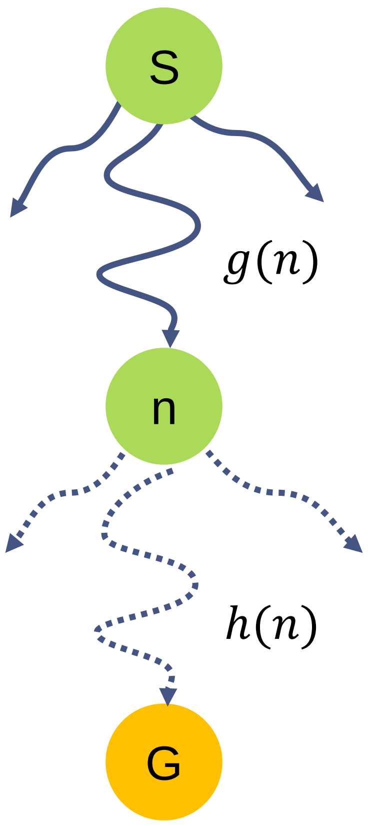
- **ČASOVNA in PROSTORSKA ZAHTEVNOST**:
  - $O(b^m)$, kjer je m največja globina drevesa
  - vsa vozlišča moramo hraniti v spominu, ker so kandidati za razvijanje nadaljnje poti
  - pomembnost ustrezne hevristične ocene (!)

Pregled



- preiskovanje prostora stanj
  - neinformirani preiskovalni algoritmi
    - iskanje v širino
    - iskanje v globino
    - iterativno poglabljanje
    - dvosmerno iskanje
    - cenovno – optimalno iskanje
  - informirani preiskovalni algoritmi
    - hevristično preiskovanje (primer)
    - požrešno preiskovanje
    - A*
    - IDA*
    - kakovost hevrističnih funkcij

Algoritem A*



- ideja: izboljšajmo hevristično funkcijo, ker so od nje očitno odvisni uspešnost iskanja in poraba časa/prostora
- vozlišča vrednotimo glede na ceno najboljše poti skozi vozlišče n:

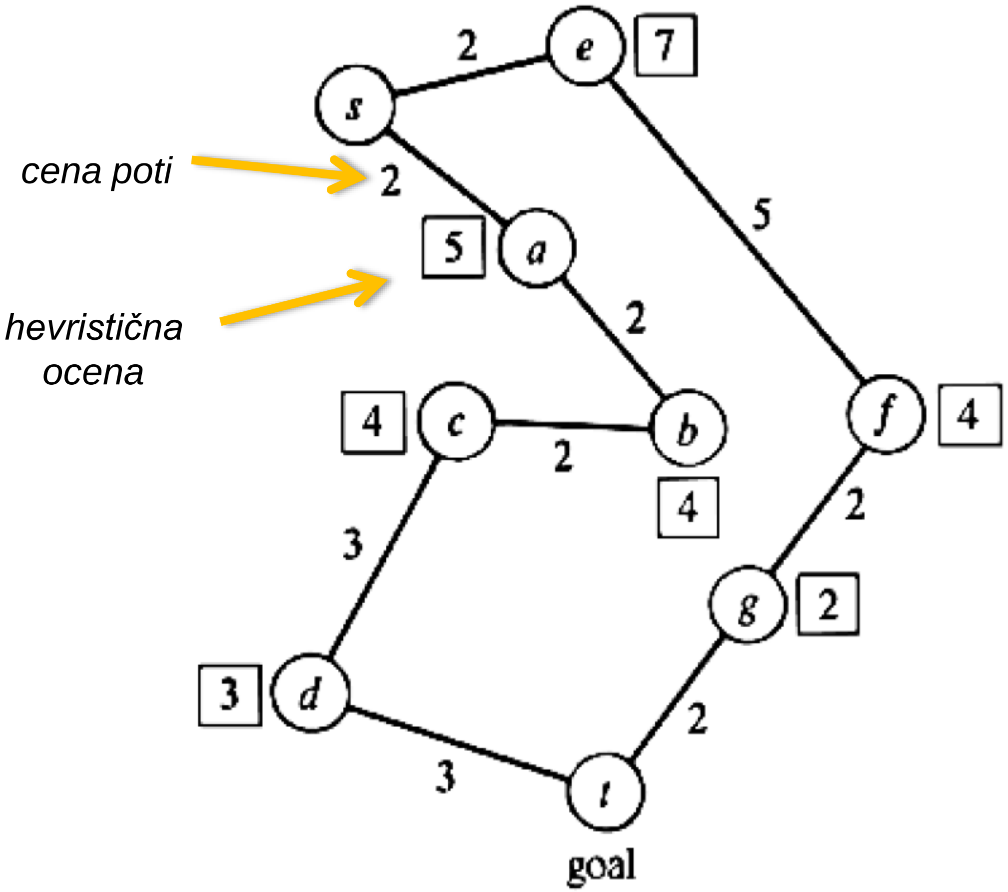
  $\boldsymbol{f(n) = g(n) + h(n)}$

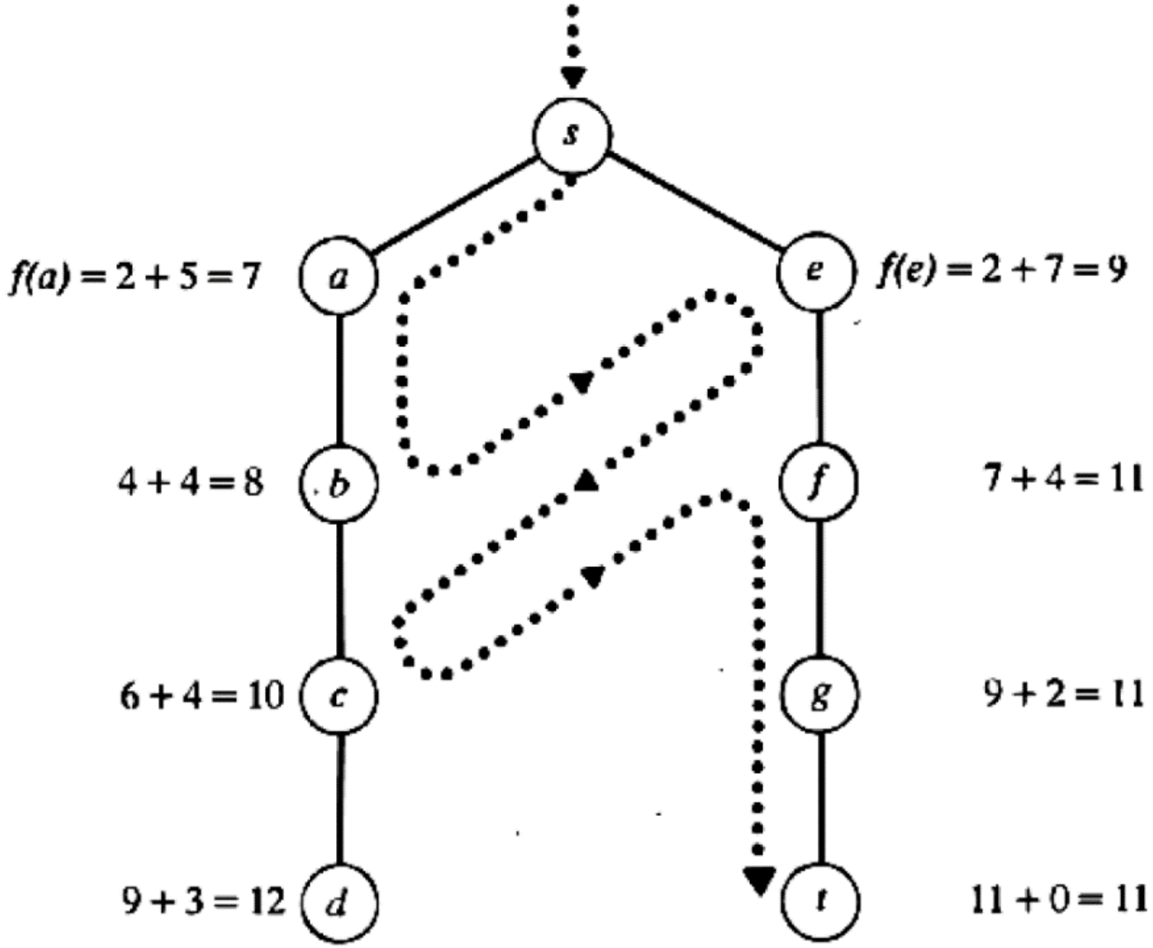
  $\mathrm{g}(n)$ – cena poti do n (znano)

  $h(n)$ – cena od n do najbližjega cilja (ocena)
- vozlišča v fronti hranimo v **prioritetni vrsti**, ki je urejena naraščajoče glede na funkcijo $f(n)$
- pri preiskovanju lahko **ponovno generiramo vozlišče $\boldsymbol{n}$, ki je že med razvitimi vozlišči** $(\boldsymbol{n'})$
  - če je $\mathrm{g}(n') < g(n)$, smo našli boljšo pot do n, vozlišče dodamo v fronto
  - če je $\mathrm{g}(n') \geq g(n)$, lahko ponovno generirano vozlišče n ignoriramo

Algoritem A*: primer

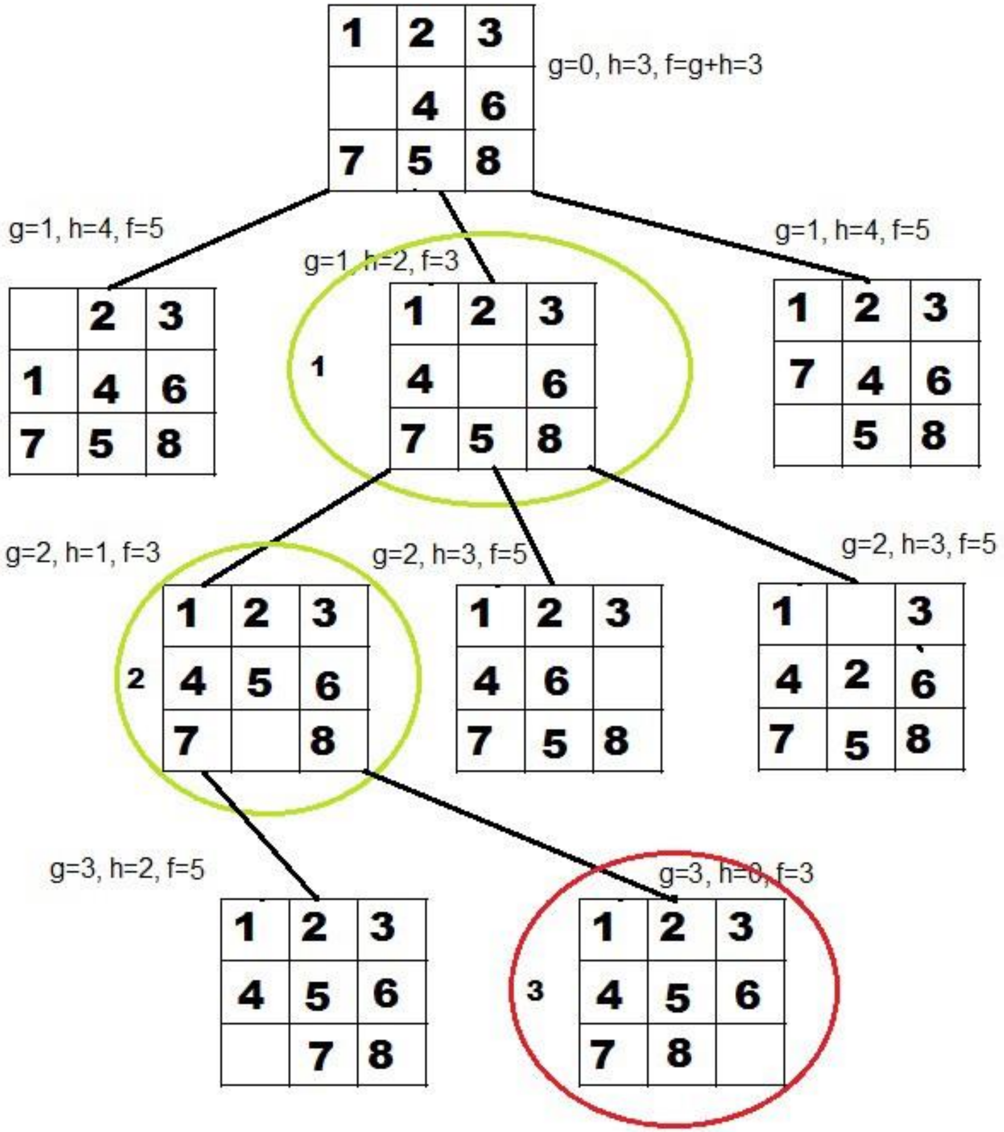
- primer 1: preiskovanje manjšega grafa





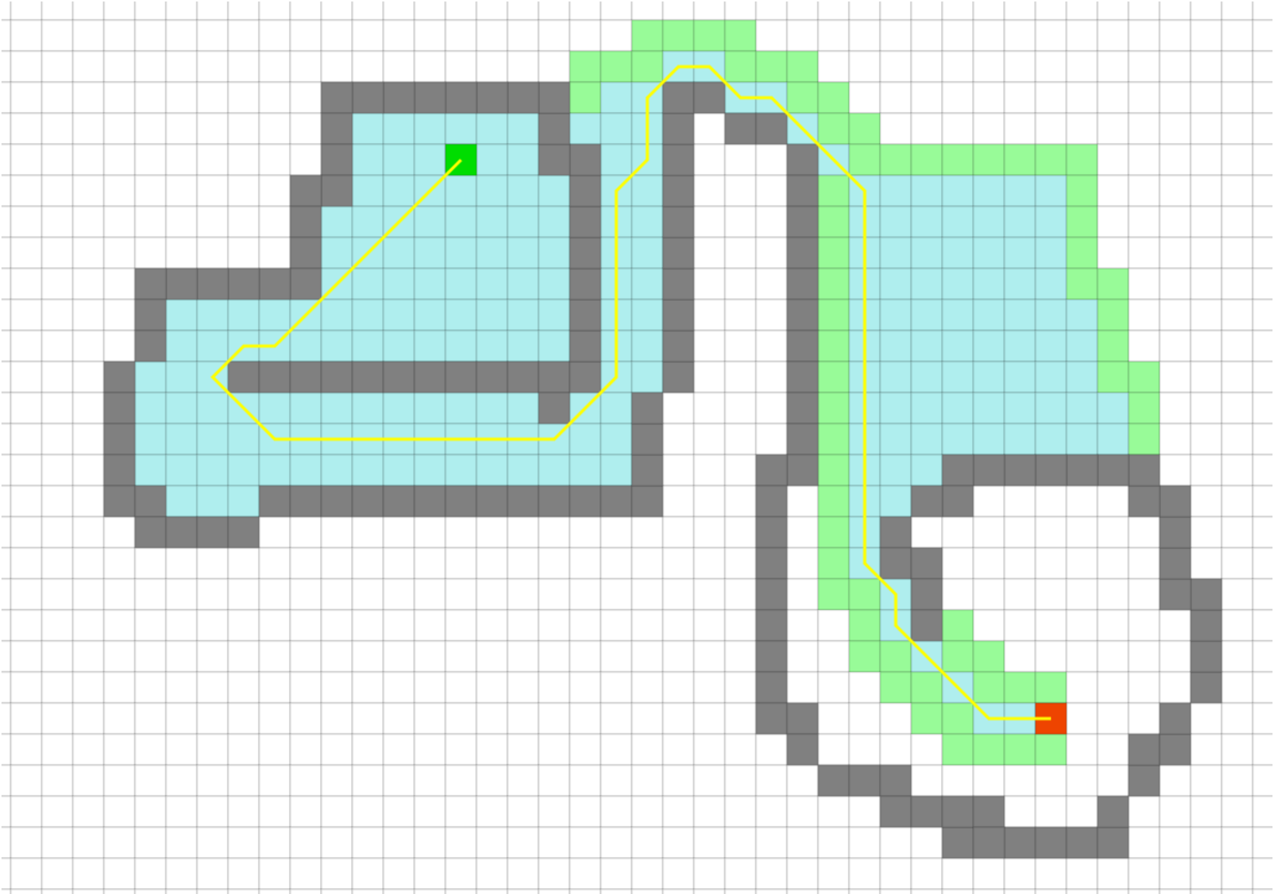
Algoritem A*: primer

- primer 2: igra 8 ploščic
- hevristika: koliko ploščic ni na pravem mestu

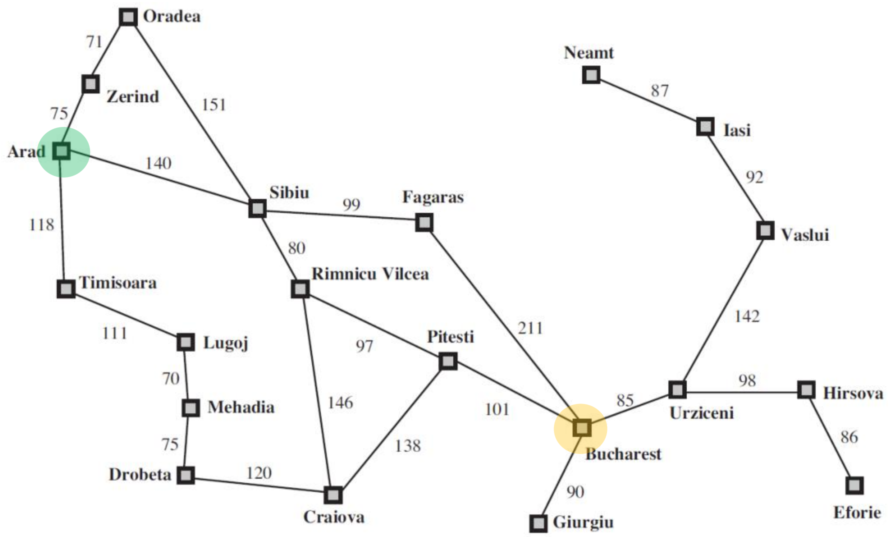


Algoritem A*: primer

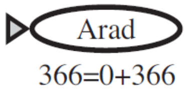
- primer 3: https://qiao.github.io/PathFinding.js/visual/



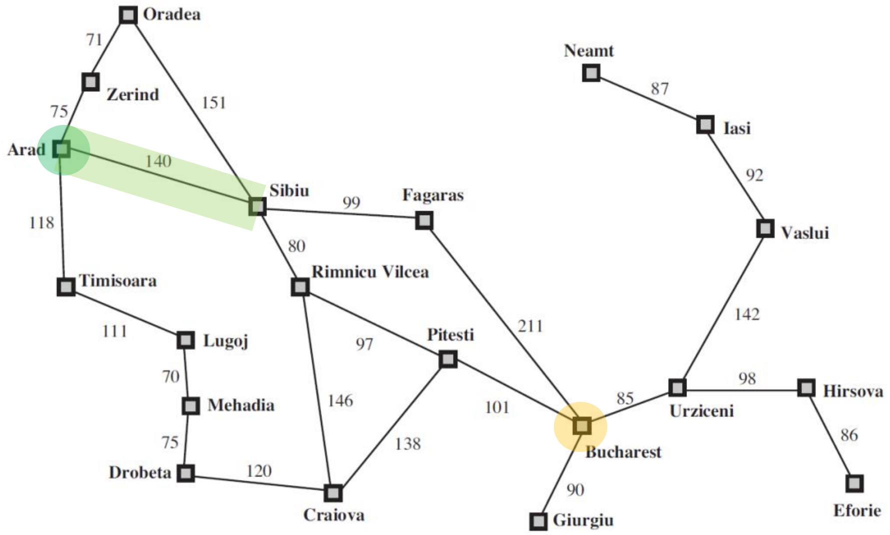
Algoritem A*: primer



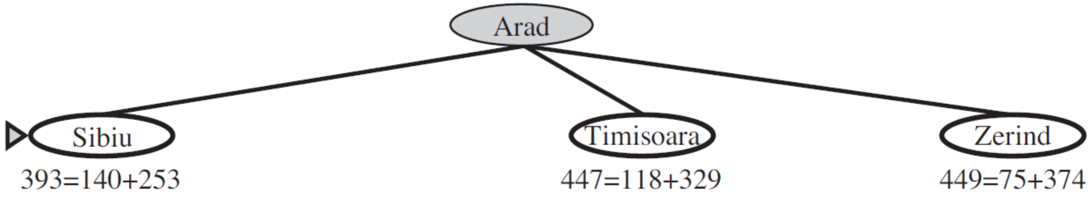
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



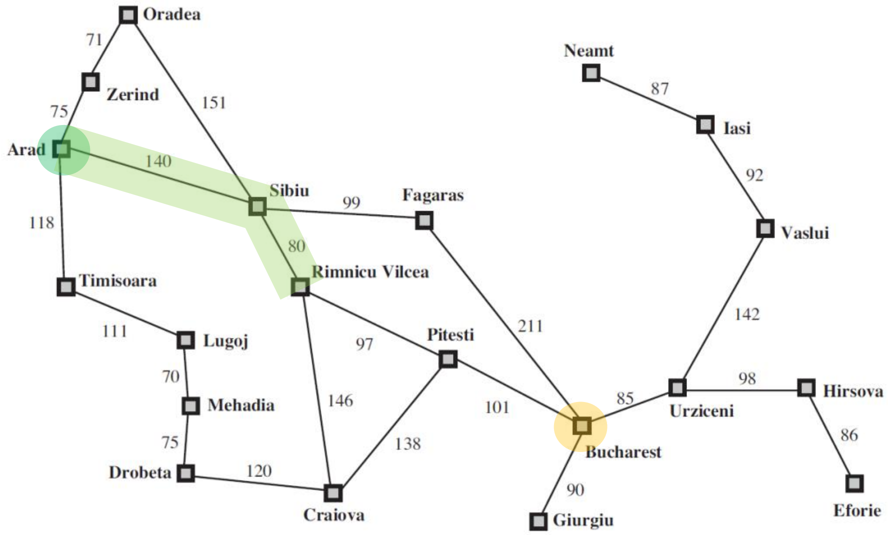
Algoritem A*: primer



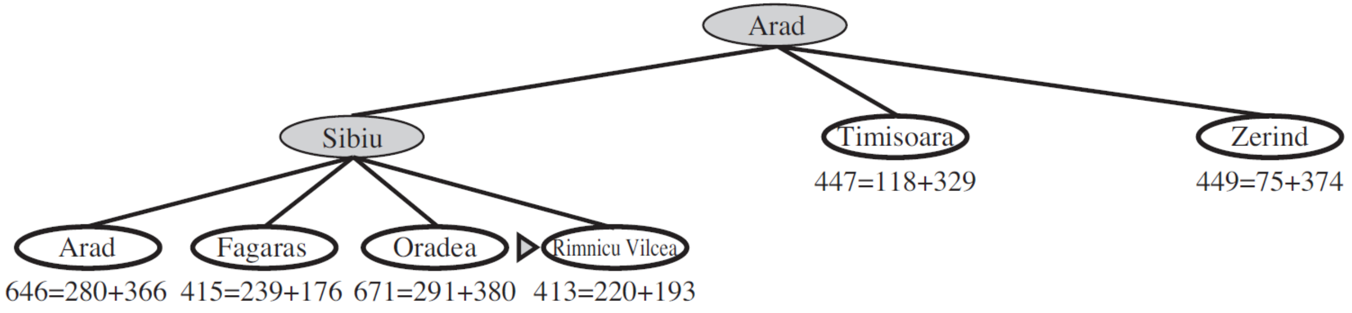
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374



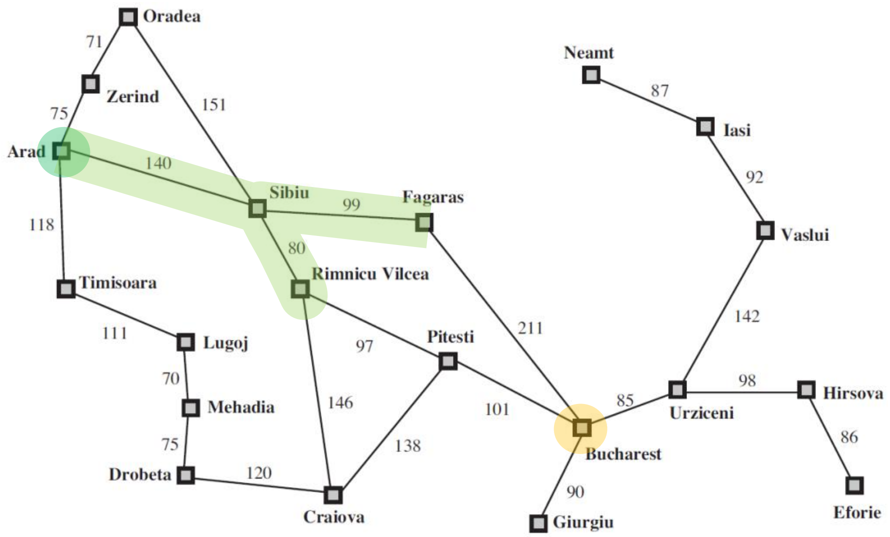
Algoritem A*: primer



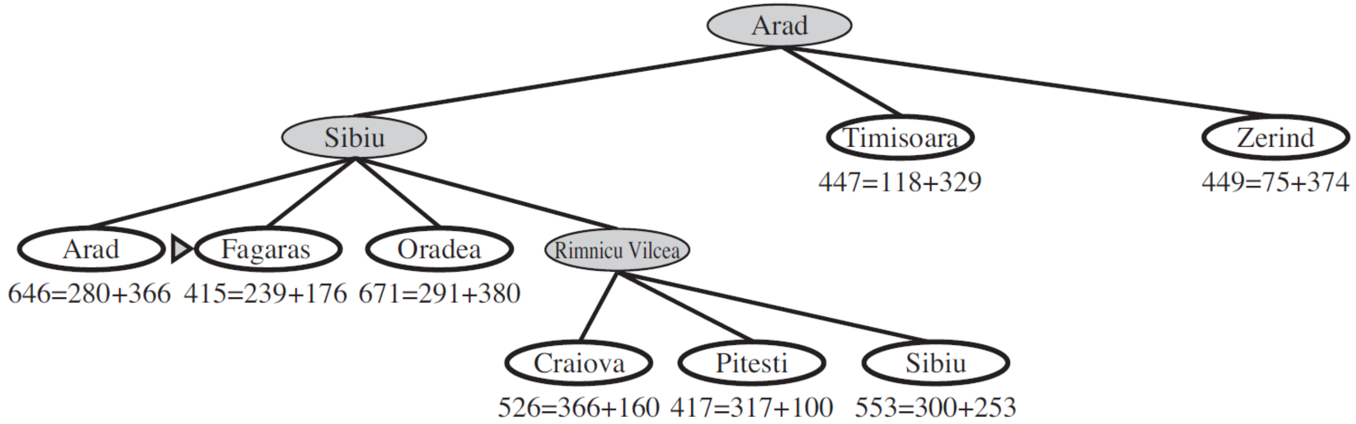
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374



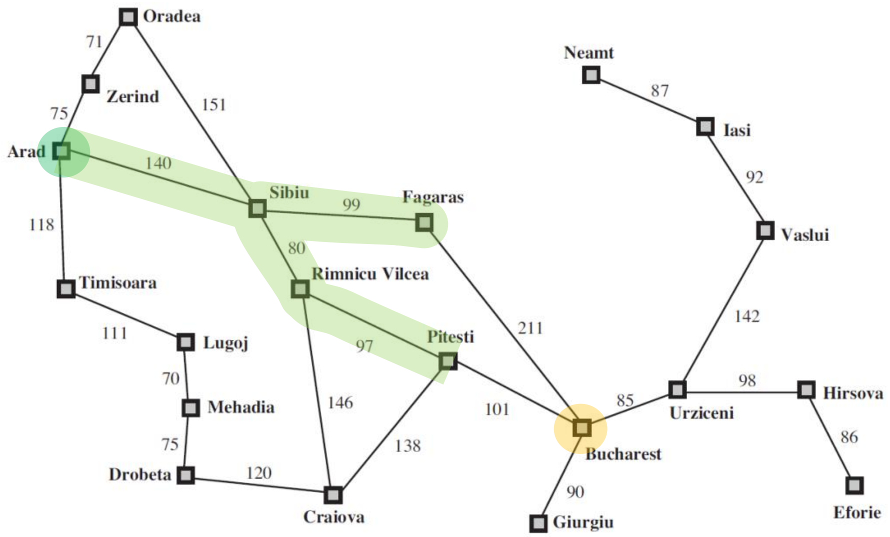
Algoritem A*: primer



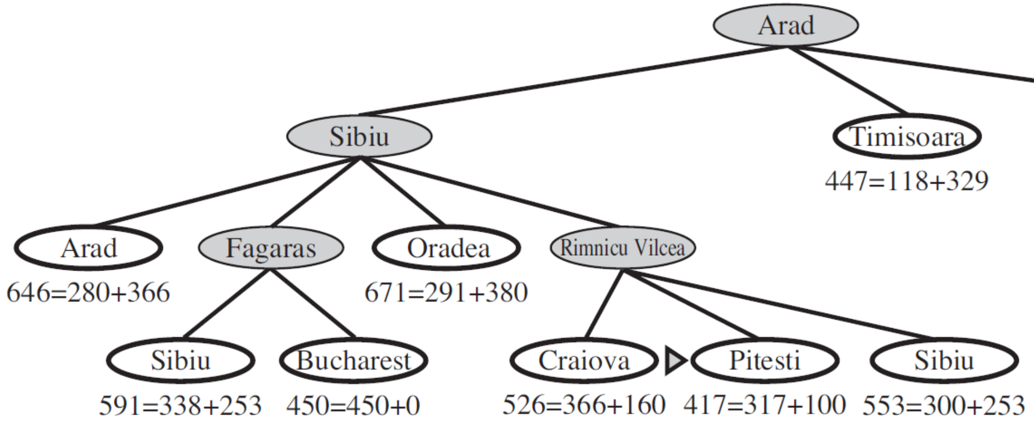
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



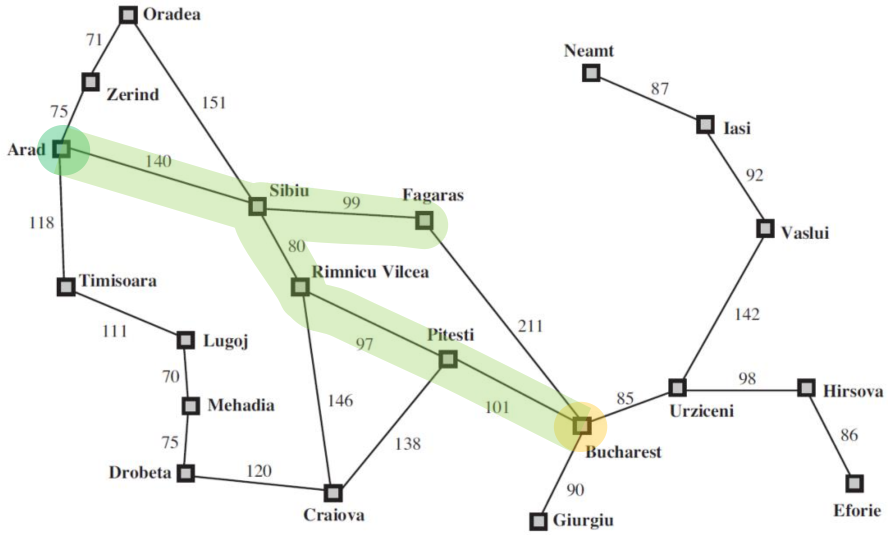
Algoritem A*: primer



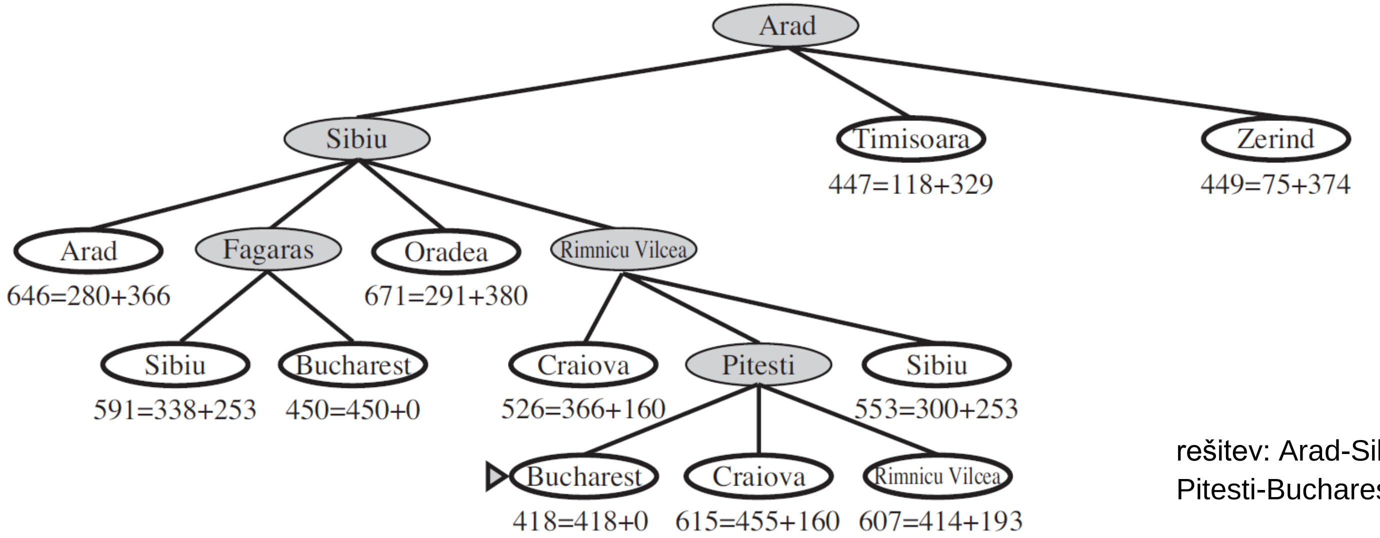
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



Algoritem A*: primer



Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

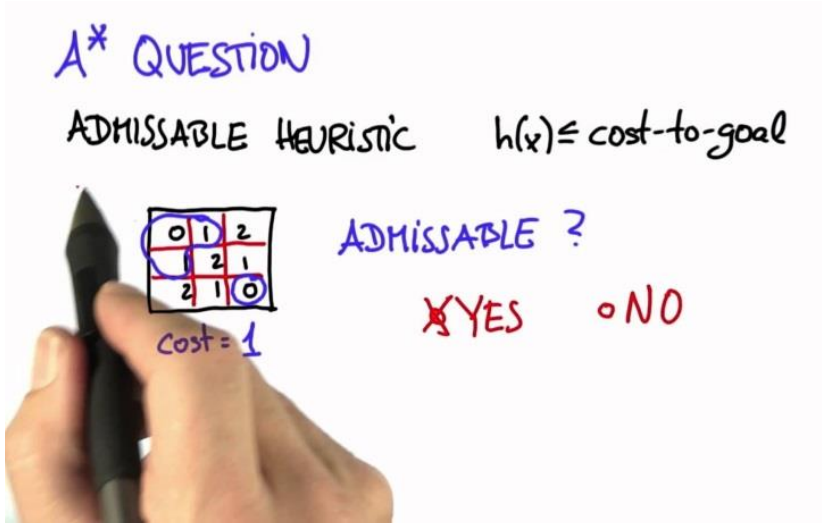


rešitev: Arad-Sibiu-Riminicu Vilcea-Pitesti-Bucharest, cena=418

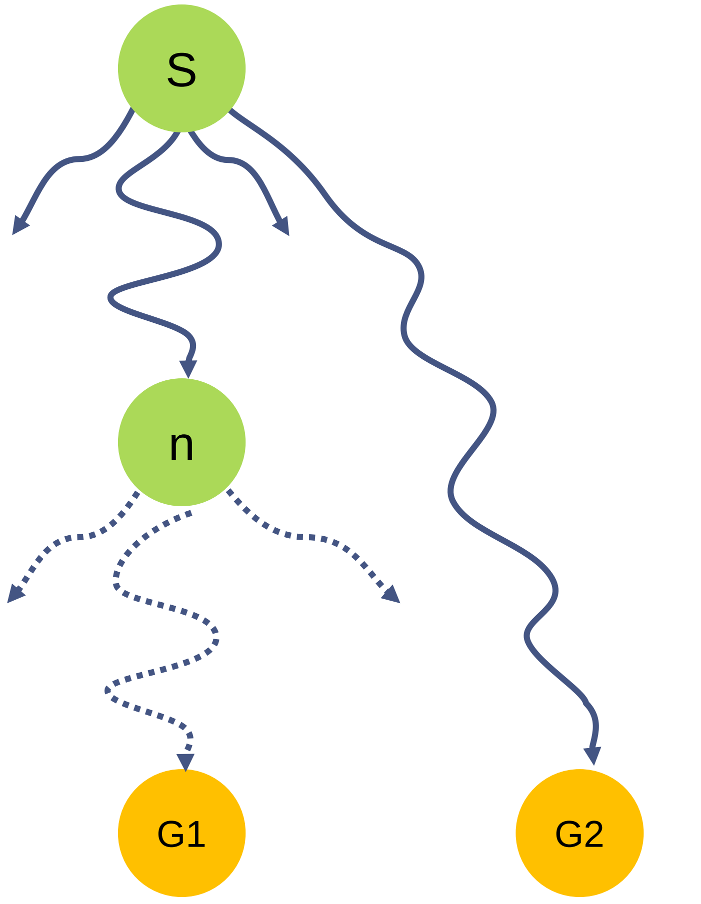
Popolnost in optimalnost A*

- algoritem A* je **popoln** in **optimalen**, če ustreza pogoju **dopustnosti** (angl. *admissibility*)
- za hevristiko $h(n)$ pravimo, da je **dopustna**, če nikoli **ne precenjuje cene do cilja**
  - formalno: hevristika $h(n)$ je dopustna, če za vsako vozlišče n velja $h(n) <= h^*(n)$, kjer je $h^*(n)$ dejanska cena optimalne poti do cilja za vozlišče n
  - zgornje pomeni, da je hevristika $h(n)$ "**optimistična**" (= predvideva, da je do cilja manj, kot dejansko je)
  - posledično tudi $f(n)$ ne precenjuje cene do cilja, saj je $g(n)$ znan, velja pa $f(n) = g(n) + h(n)$

  - ali je lahko $h(n) = 0$
    (to je tudi optimistična cenilka)?
    Da, vendar… ... ?
  - idealno velja $h(n) = h^*(n)$



Skica dokaza optimalnosti A*

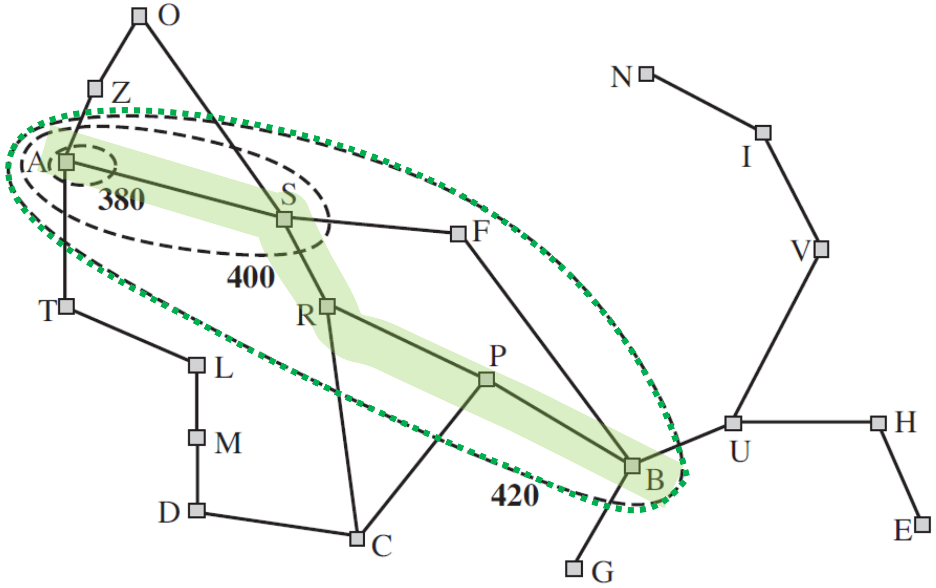


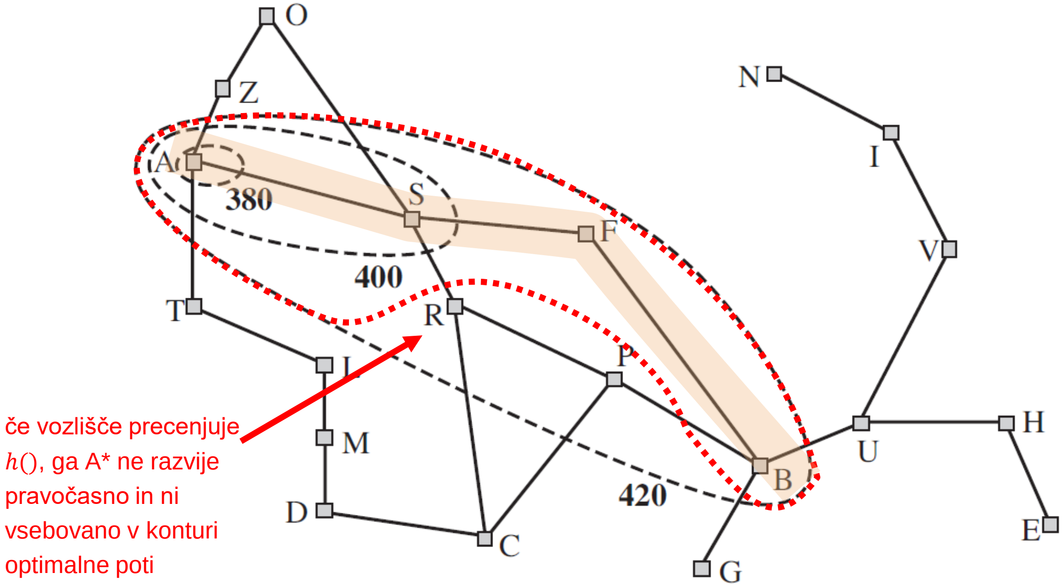
- $G1$ je optimalen cilj, $G2$ neoptimalen
- denimo, da je $h(n)$ dopustna cenilka

- velja:
  - $f(G2) = g(G2) + h(G2) = g(G2) + 0 = g(G2)$, ker je $G2$ cilj
  - $f(G1) = g(G1)$, ker je $G1$ tudi cilj
  - velja $g(G2) > g(G1)$, ker je $G1$ optimalen cilj, $G2$ pa neoptimalen
  - če velja $g(G2) > g(G1)$, potem velja tudi $\boldsymbol{f(G2)} > \boldsymbol{g(G1)}$, ker $g(G2) = f(G2)$ (prva alinea)
  - n naj bo neko vmesno vozlišče na poti do optimalnega cilja $G1$
  - ker je $h(n)$ dopustna cenilka (ne precenjuje cene do optimalnega cilja $G1$), mora veljati $f(G2) \geq f(n)$ (sicer bi bil $G2$ bolj optimalen cilj)
  - ker velja $f(G2) \geq f(n)$, algoritem A* nikoli ne bo izbral cilja $G2$ za razvijanje (kot končni cilj), torej bo A* vrnil kot rešitev optimalni cilj $G1$

Skica dokaza optimalnosti A*

- še drugačen premislek o optimalnosti
- algoritem A* razvija vozlišča glede na naraščajočo oceno vozlišč $f(n)$. Pri tem povečuje "raziskanost" prostora v obliki **reliefnih kontur** (vsaka kontura predstavlja večjo vrednost $f(n)$)
- če hevristika ne bi bila dopustna (in bi v nekem vozlišču precenjevala ceno do cilja), to vozlišče ne bi bilo vsebovano v konturi, ki predstavlja vrednost optimalne poti do cilja
- primer prikazuje:
  - levo: optimalna pot
  - desno: A* bi zaobšel vozlišče R, ki je na optimalni poti, če bi imel preveliko vrednost $f(n)$





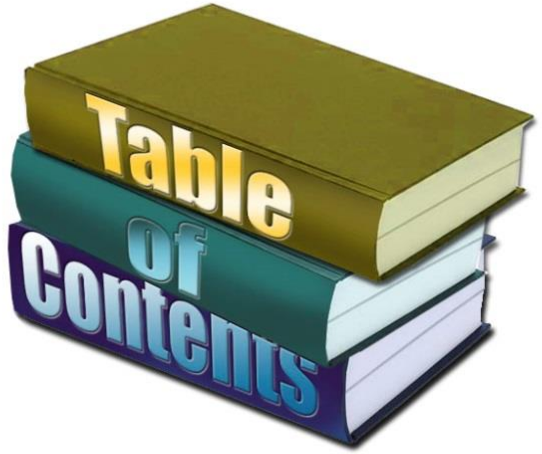
Učinkovitost algoritma A*

- **POPOLNOST** in **OPTIMALNOST**:
  - Da, če je hevristika **dopustna**.

- **ČASOVNA ZAHTEVNOST**:
  - odvisni sta od kakovosti hevristike $h(n)$
    (boljša hevristika – manjša poraba časa in prostora)
  - definirajmo:
    - h^* naj bo dejanska cena do optimalne rešitve
    - relativna napaka hevristike $\epsilon = (h^* - h)/h^*$
  - zahtevnost je eksponentna glede na funkcijo relativne napake in globino rešitve: $\mathrm{O}\left(\mathrm{b}^{f(\epsilon)\cdot d}\right)$

- **PROSTORSKA ZAHTEVNOST**:
  - večji problem kot časovna zahtevnost, ker mora A* hraniti vsa vozlišča v spominu
  - nepraktično za velike probleme
  - boljše alternative glede porabe prostora: algoritem IDA* (*iterative deepening A**), RBFS (*recursive best-first search*), MA* (*memory-bounded A**), SMA* (*simplified A**), LRTA* (*learning real-time A**)

Pregled



- preiskovanje prostora stanj
  - neinformirani preiskovalni algoritmi
    - iskanje v širino
    - iskanje v globino
    - iterativno poglabljanje
    - dvosmerno iskanje
    - cenovno – optimalno iskanje
  - informirani preiskovalni algoritmi
    - hevristično preiskovanje (primer)
    - požrešno preiskovanje
    - A*
    - IDA*
    - kakovost hevrističnih funkcij

Algoritem IDA*

- iterative-deepening A*
- deluje analogno kot **iterativno poglabljanje** (izvaja **preiskovanje v globino** z različnimi mejami), vendar za mejo ne uporabljamo globine, temveč vrednost funkcije $f(n)$
  - za mejo na začetku izberemo vrednost $f(n)$ začetnega vozlišča
  - na vsaki iteraciji razvijemo vsa vozlišča z $f(n) \leq$ mejni vrednosti
  - za naslednjo iteracijo izberemo mejo, ki je najmanjši $f(n)$ še nerazvitih vozlišč

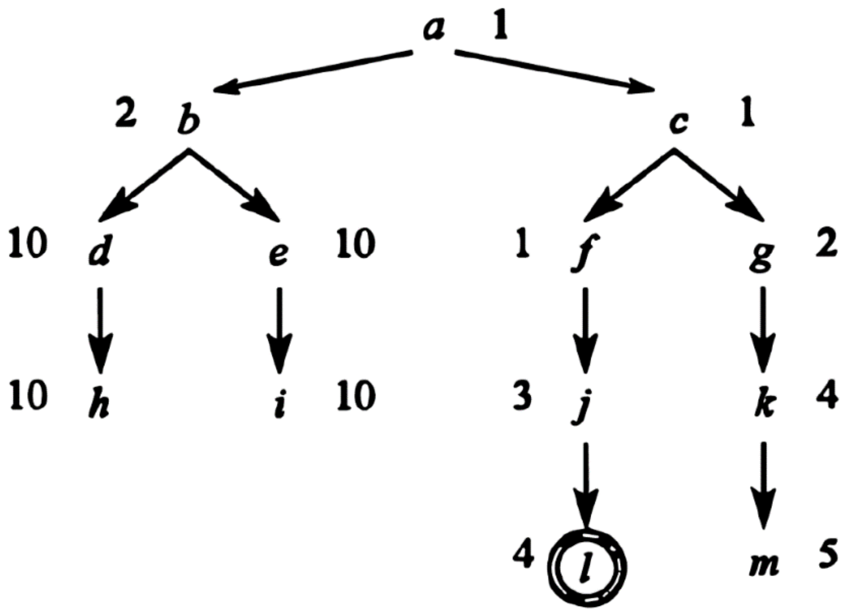
```
procedure ida_star(root)
  bound := h(root)
  path := [root]
  loop
    t := search(path, 0, bound)
    if t = FOUND then return (path, bound)
    if t = ∞ then return NOT_FOUND
    bound := t
  end loop
end procedure
```

```
function search(path, g, bound)
  node := path.last
  f := g + h(node)
  if f > bound then return f
  if is_goal(node) then return FOUND
  min := ∞
  for succ in successors(node) do
    if succ not in path then
      path.push(succ)
      t := search(path, g + cost(node, succ), bound)
      if t = FOUND then return FOUND
      if t < min then min := t
      path.pop()
    end if
  end for
  return min
end function
```

Psevdokoda: https://en.wikipedia.org/wiki/Iterative_deepening_A*

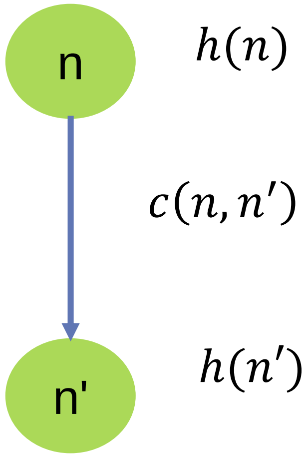
Algoritem IDA*

- primer:
  - podane so vrednost **f(n)** ($= g(n) + h(n)$) vozlišč
  - simuliraj preiskovanje z IDA*
- generirana vozlišča
  - 1. iteracija, meja=1: a/1, b/2, c/1, f/1, j/3, g/2
  - 2. iteracija, meja=2: a/1, b/2, d/10, e/10, c/1, f/1, j/3, g/2, k/4
  - 3. iteracija, meja=3: a/1, b/2, d/10, e/10, c/1, f/1, j/3, l/4, g/2, k/4
  - 4. iteracija, meja=4: a/1, b/2, d/10, e/10, c/1, f/1, j/3, **l/4**

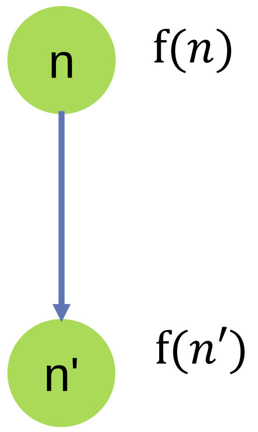


Učinkovitost algoritma IDA*

- redundanca: ponovno generiranje velikega števila vozlišč
- neučinkovit, če prevladujejo vozlišča z zelo raznolikimi vrednostmi funkcije $f(n)$
- vendar: v spominu hrani samo trenutno pot (podobno kot pri iskanju v globino) in ne vseh vozlišč kot A*
- želimo si, da je IDA* optimalen glede na: <u>ceno</u> najdene rešitve, <u>porabljen prostor</u> in <u>porabljen čas</u>:
  - to je možno, če IDA* razvije najmanjše potrebno število vozlišč, čemur rečemo, da jih razvija v *prioritetnem vrstnem redu*
  - IDA* razvija vozlišča v prioritetnem vrstnem redu, če je hevristična ocena $h(n)$ **monotona** ali **konsistentna** (monotone/consistent). To je res, kadar za vsaki povezani vozlišči n in n' velja (trikotniška neenakost): $h(n) \leq c(n, n') + h(n')$ in $h(g) = 0$ za vsako končno vozlišče g
  - če je hevristika monotona/konsistentna, je tudi dopustna (!)

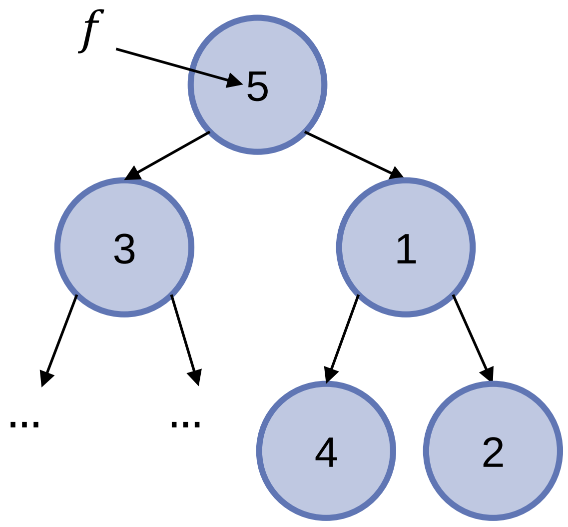


Učinkovitost algoritma IDA*



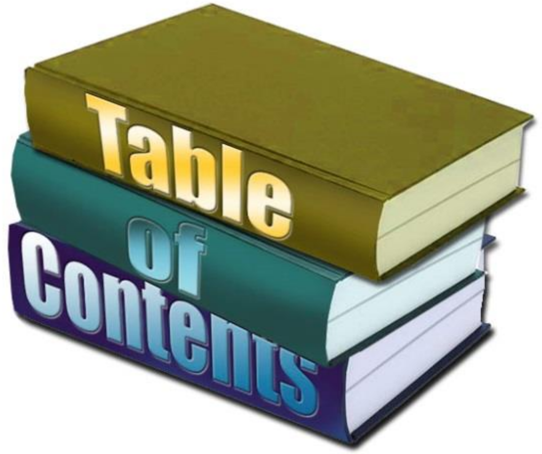
- poenostavitev: za zagotovitev razvijanja vozlišč v prioritetnem vrstnem redu ni nujno potrebno, da velja monotonost za hevristiko h, temveč **že zadošča, da je monotona cenilna funkcija $\boldsymbol{f}$ $(= \boldsymbol{g} + \boldsymbol{h})$**
- cenilna funkcija f je monotona, če za vsak par povezanih vozlišč n in n' velja $f(n) <= f(n')$

- primer nemonotone funkcije f:
  - meja za f je enaka 5, vendar pa vozlišče z $f = 3$ razvijemo pred vozlišči z $f = 1$ in $f = 2$



- premislek – izziv (objavi odgovor na forumu)!
  - Ali dopustna hevristična funkcija h zagotavlja, da je cenilna funkcija f monotona?
  - Ali za monotone cenilne funkcije f velja, da zadoščajo pogoju iz izreka o dopustnosti hevristik h?

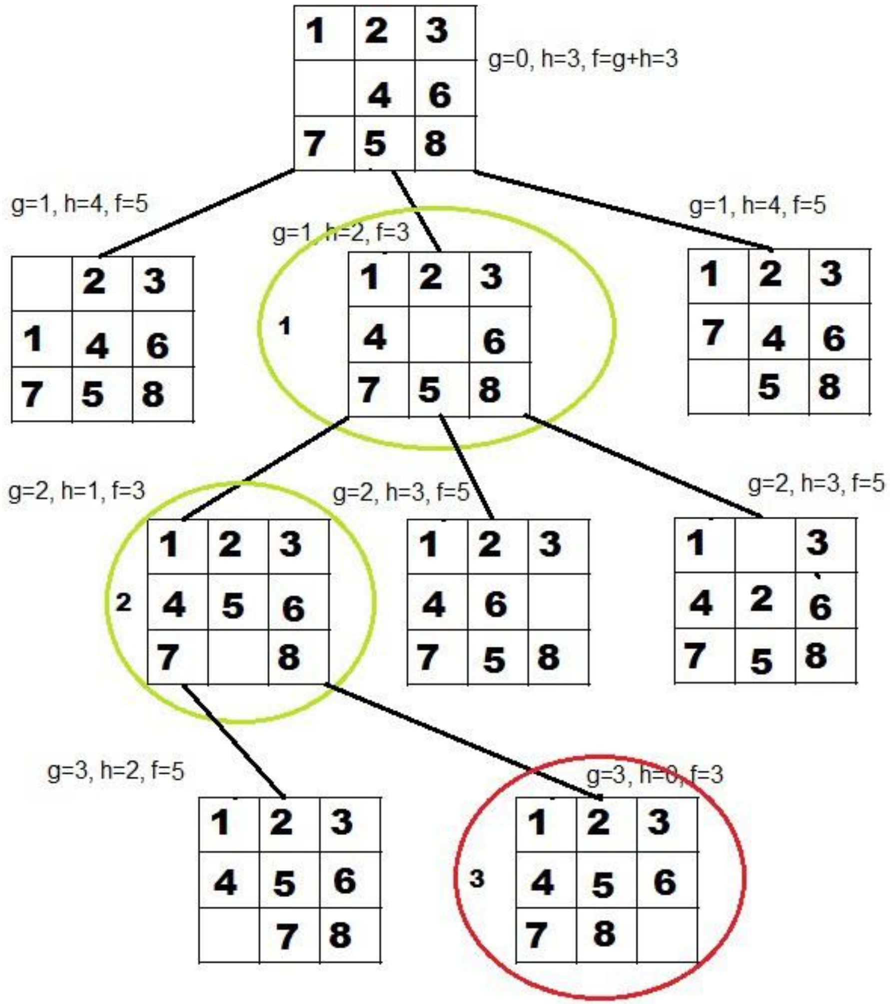
Pregled



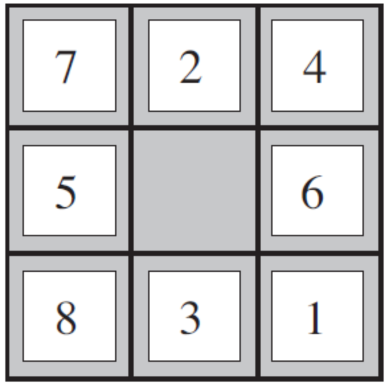
- preiskovanje prostora stanj
  - neinformirani preiskovalni algoritmi
    - iskanje v širino
    - iskanje v globino
    - iterativno poglabljanje
    - dvosmerno iskanje
    - cenovno – optimalno iskanje
  - informirani preiskovalni algoritmi
    - hevristično preiskovanje (primer)
    - požrešno preiskovanje
    - A*
    - IDA*
    - kakovost hevrističnih funkcij

Kakovost hevrističnih funkcij

- primer: igra 8 ploščic
- povprečna dolžina rešitve je 22 korakov
- povprečni faktor vejanja je 3
  (2 potezi možni v vogalu, 3 ob robu, 4 v sredini)
- izčrpno preiskovanje bi torej pregledalo prostor
  3^{22} stanj (za 3x3); na srečo je dosegljivih
  le približno 9!/2 = 181,440 stanj
- hevristična funkcija lahko učinkovito zmanjša
  prostorsko in časovno ceno iskanja
- pri iskanju ustrezne hevristike si pomagamo
  s poznavanjem problema



Kakovost hevrističnih funkcij

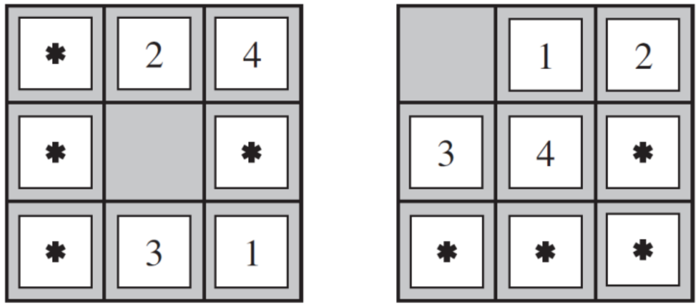


- primeri hevrističnih funkcij:
  - h_1 – število ploščic, ki niso na pravem mestu (za primer na desni: $\mathrm{h}_1 = 8$)
  - h_2 – vsota manhattanskih razdalj ploščic do pravega mesta (za primer na desni: $h_2 = 3+1+2+2+2+3+3+2 = 18$)
- kakovost h lahko ocenimo:
  - s številom generiranih vozlišč
  - z efektivnim faktorjem vejanja (koliko vozlišč N je algoritem generiral, da je na globini d našel rešitev)

	število generiranih vozlišč			*efektivni faktor vejanja*		
Globina	**IDS**	**A*(h_1)**	**A*(h_2)**	**IDS**	**A*(h_1)**	**A*(h_2)**
2	10	6	3	2,45	1,79	1,79
4	112	13	12	2,87	1,48	1,45
6	680	20	18	2,73	1,34	1,30
8	6384	39	25	2,80	1,33	1,24
10	47127	93	39	2,79	1,38	1,22
12	3644035	227	73	2,78	1,42	1,24
14	?	539	113	?	1,44	1,23
16	?	1301	211	?	1,45	1,25
18	?	3056	363	?	1,46	1,26
20	?	7276	676	?	1,47	1,27
22	?	18094	1219	?	1,48	1,28
24	?	39135	1641	?	1,48	1,26

Kako do idej za hevristike?

- želimo imeti dopustne hevristike:
  - s čim večjimi vrednostmi
  - s sprejemljivo ceno (časom) izračuna
- v prejšnjem primeru je h_2 boljša od h_1 (ker $h_2(n) \geq h_1(n)$ za vsak n, pravimo, da h_2 dominira h_1)
- pridobivanje hevristik:
  - iz poenostavljenega (relaksiranega) problema:
    - "ploščico lahko prestavimo na poljubno (tudi neprazno) polje"
    - "ploščico lahko prestavimo na poljubno (tudi nesosednje) prazno polje"
    - "ploščico lahko prestavimo na sosednje (tudi neprazno) polje"
  - z vzorci podproblemov (osredotočimo se npr. samo na iskanje rešitve za del problema)
  - z izkušnjami in uteževanjem kriterijev (npr. oddaljenost od cilja, število sosednjih ploščic, ki ne mejijo na ciljno mesto ipd.)



Start State Goal State

lokalno preiskovanje