OSNOVE UMETNE INTELIGENCE

2022/23

neinformirani preiskovalni algoritmi (ID, BS, UCS)

informirani preiskovalni algoritmi (A)*

Pridobljeno znanje s prejšnjih predavanj

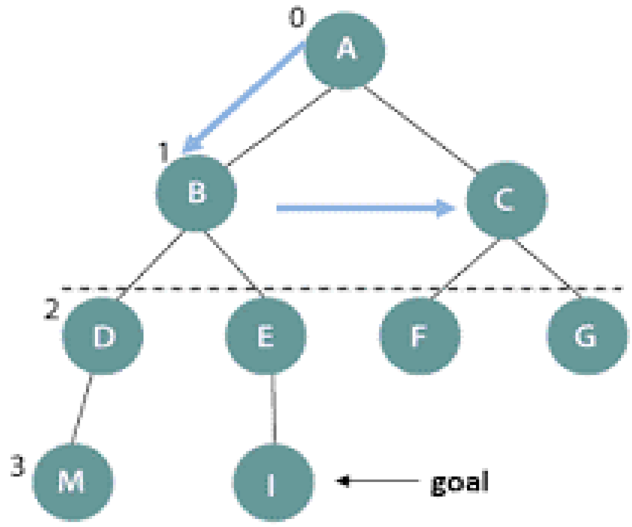
- **strojno učenje**
  - nenadzorovano učenje: metoda k-voditeljev

- **preiskovanje**
  - definicija in cilji področja
  - primeri umetnih in realnih problemov
  - **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 $O(bm)$

Iskanje v globino - izboljšave

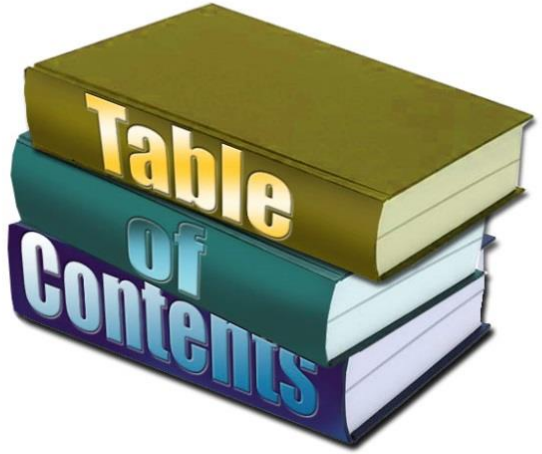
- iskanje s sestopanjem (*backtracking search*):
  - namesto vseh naslednikov generiramo samo enega po enega
  - ➔ prostorska zahtevnost $O(m)$

- iskanje z omejitvijo globine (*depth-limited search*):
  - vnaprej definiramo mejo globine l
  - vozlišča na globini l obravnavamo, kot da nimajo naslednikov
  - če izberemo $l < d$, je algoritem nepopoln (ne najde rešitve)
  - če izberemo $l > d$, je algoritem popoln, a neoptimalen
  - časovna zahtevnost je $O\left(b^l\right)$, prostorska pa $O(bl)$
  - pri določitvi l pomaga domensko znanje

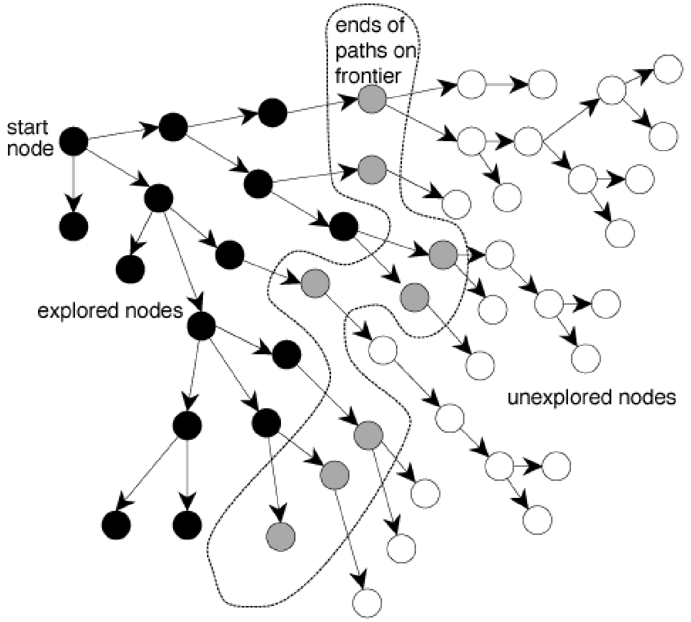


- iterativno poglabljanje (angl. I*terative Deepening Search (IDS)*)

Pregled

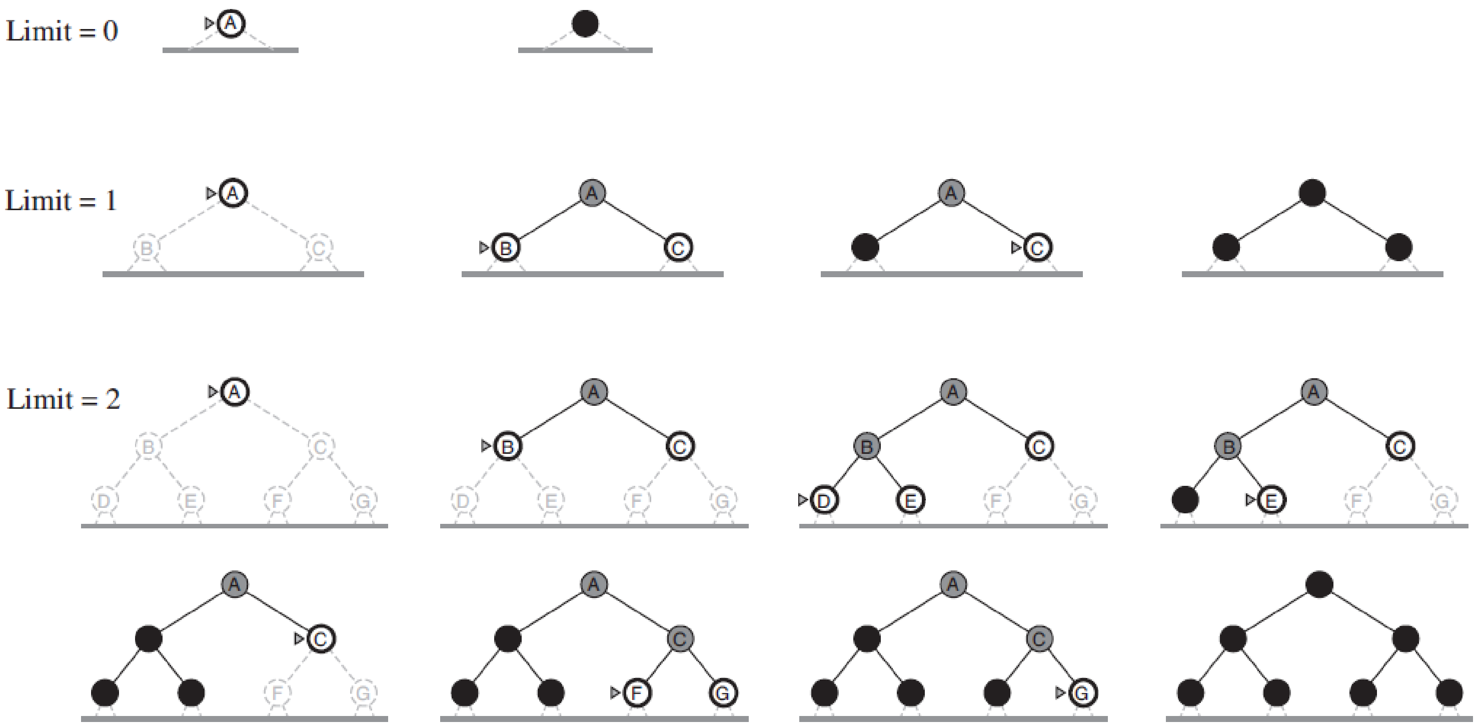


- preiskovanje prostora stanj
  - neinformirani preiskovalni algoritmi
    - iskanje v širino
    - iskanje v globino
    - iterativno poglabljanje
    - dvosmerno iskanje
    - cenovno – optimalno iskanje
  - informirani preiskovalni algoritmi
    - hevristično preiskovanje (primer)
    - požrešno preiskovanje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



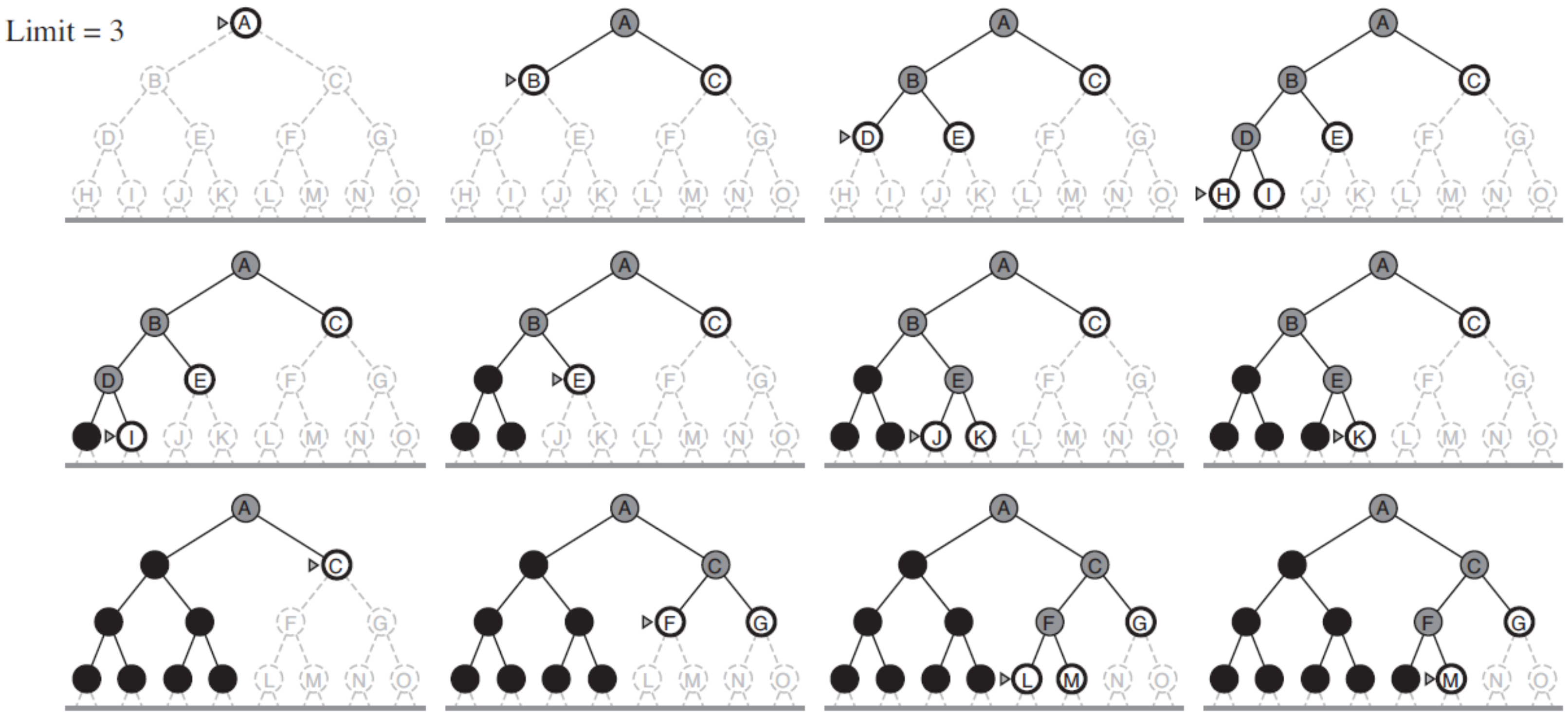
Iterativno poglabljanje

- problem globinsko omejenega iskanja v globino je nastavitev meje l
- rešitev: iterativno poglabljanje (*iterative deepening depth-first search*)
- strategija: začnimo z nizko mejo *limit* in jo povečujmo za 1, dokler ne najdemo rešitve. Na vsakem koraku poženimo iskanje v globino.



Iterativno poglabljanje

- problem globinsko omejenega iskanja v globino je nastavitev meje l
- rešitev: iterativno poglabljanje (*iterative deepening depth-first search*)
- strategija: začnimo z nizko mejo *limit* in jo povečujmo za 1, dokler ne najdemo rešitve. Na vsakem koraku poženimo iskanje v globino.



Učinkovitost iterativnega poglabljanja

- **POPOLNOST** (angl. *completeness*):
  - Da.

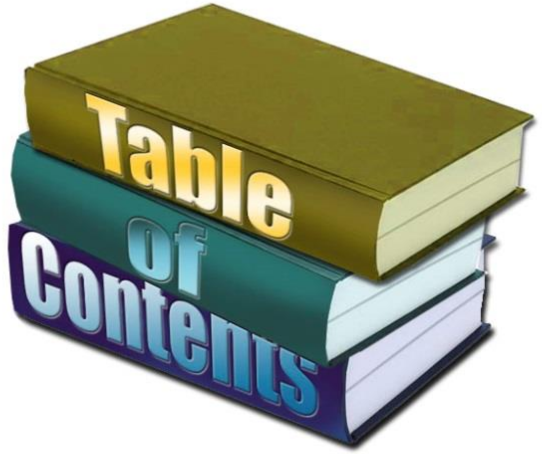
- **OPTIMALNOST** (angl. *optimality*):
  - Da (v kolikor iščemo najkrajšo rešitev).

- **ČASOVNA ZAHTEVNOST**:
  - v iteracijah se ponavlja generiranje istih vozlišč znova
  - asimptotično gledano, je generirano število vozlišč enako kot pri iskanju v širino:
    $$[\mathrm{b}] + [b + b^2] + \cdots [b + b^2 + \cdots + b^d]$$
    $$= db + (d-1)b^2 + \cdots + 2b^{d-1} + b^d = O\left(b^d\right)$$
  - kljub višji ceni pa je ta cena še vedno sprejemljiva, ker se največ vozlišč generira na zadnjem nivoju (npr. za $b = 10$, $d = 5$, velja: $N(IDS) = 123.450$, $N(BFS) = 111.110$)

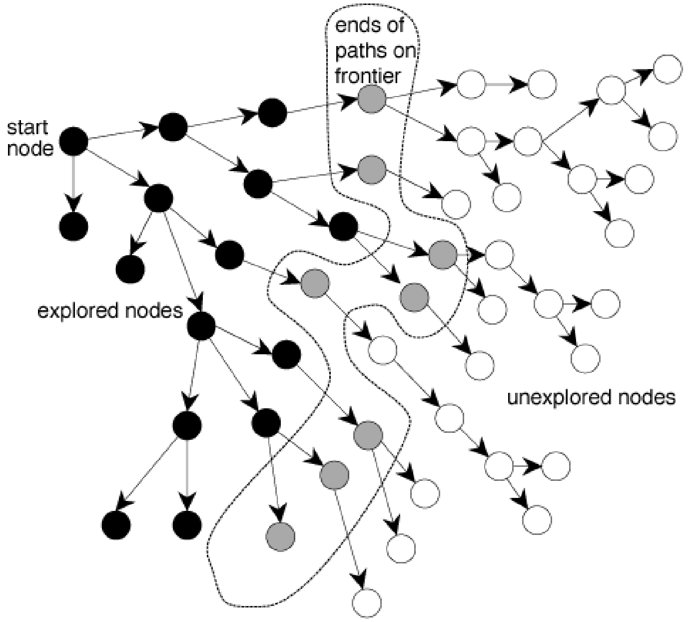
- **PROSTORSKA ZAHTEVNOST**:
  - hraniti mora samo $O(bd)$ razvitih vozlišč (linearna prostorska zahtevnost!)

- **metoda torej kombinira prednosti iskanja v širino (popolnost, optimalnost) in iskanja v globino (linearna prostorska zahtevnost)**

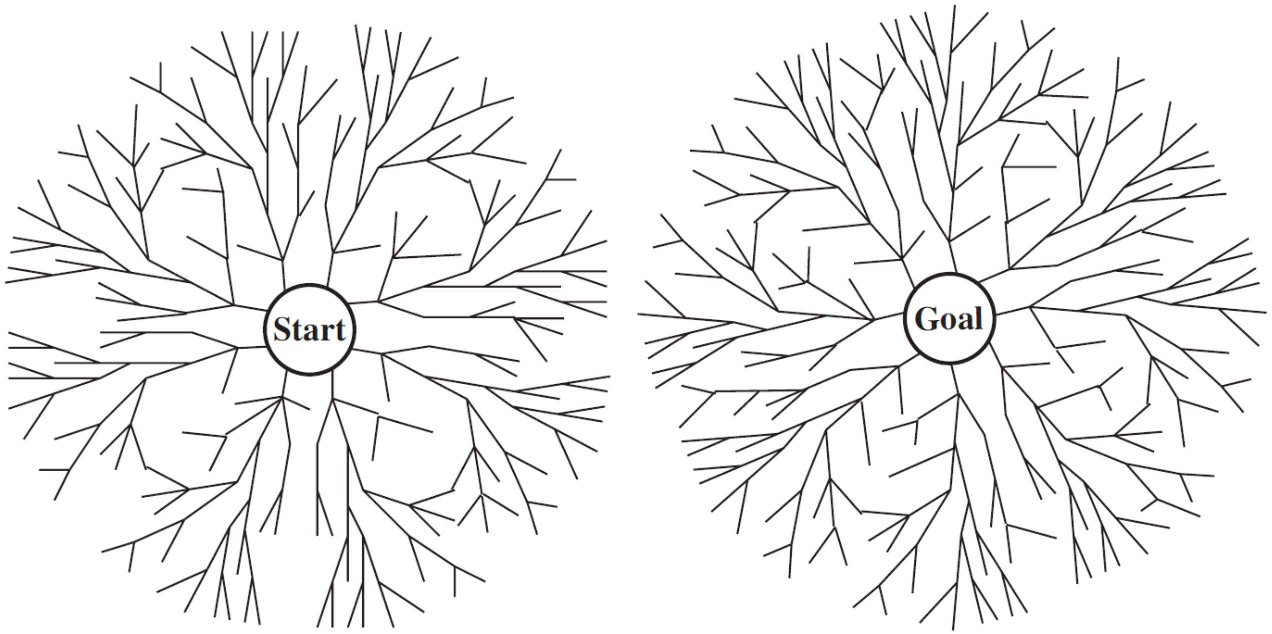
Pregled



- 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



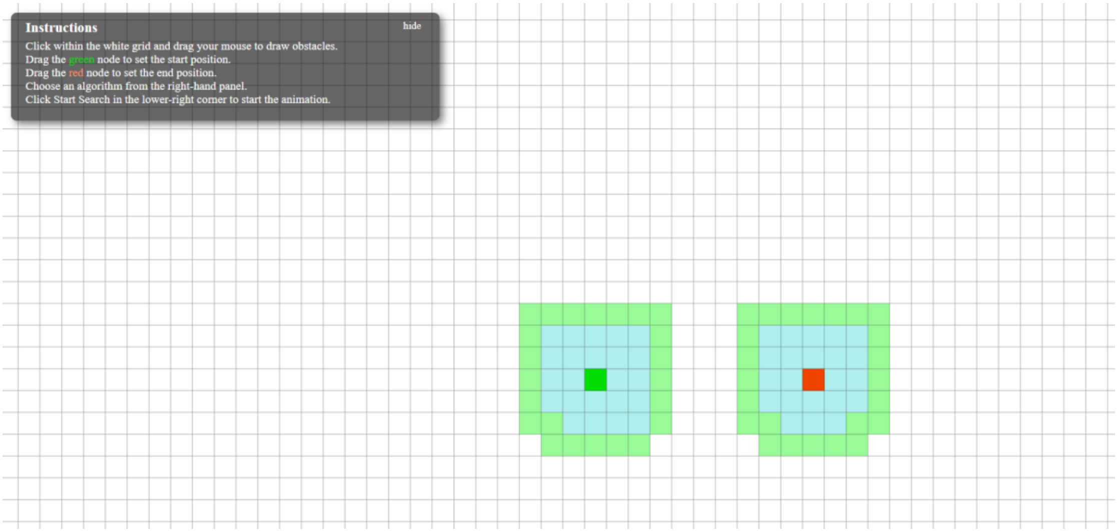
Dvosmerno iskanje



- ideja: pognati vzporedni iskanji od začetnega vozlišča proti cilju in vzvratno od cilja proti začetnemu vozlišču z upanjem, da se iskanji "srečata" na polovici poti
- motivacija: zaradi znižanja globine iskanja želimo doseči časovno zahtevnost $b^{d/2} + b^{d/2} = O(b^{d/2})$, kar je manj kot $O(b^d)$

Demo

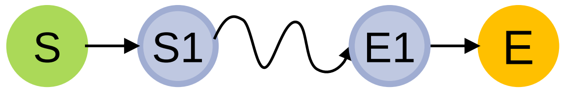
- PathFinding
  https://qiao.github.io/PathFinding.js/visual/



Implementacija dvosmernega iskanja

- za izvedbo vzvratnega iskanja morajo vozlišča imeti kazalec na predhodnika
- ciljno vozlišče mora biti znano
  - pri igri 8 ploščic je npr. znano, pri uganki Sudoku pa ne
- uporabimo lahko poljuben preiskovalni algoritem
  - če uporabimo iskanje v širino, najde algoritem optimalno rešitev
- cilj iskanja preverja, ali obstaja med frontama obeh iskanj presečišče

- problemski prostor lahko redefiniramo tako, da en korak iskanja v "dvosmernem" prostoru predstavlja dva koraka (od začetka proti cilju in od cilja proti začetku) v originalnem prostoru
  - če v originalnem prostoru velja



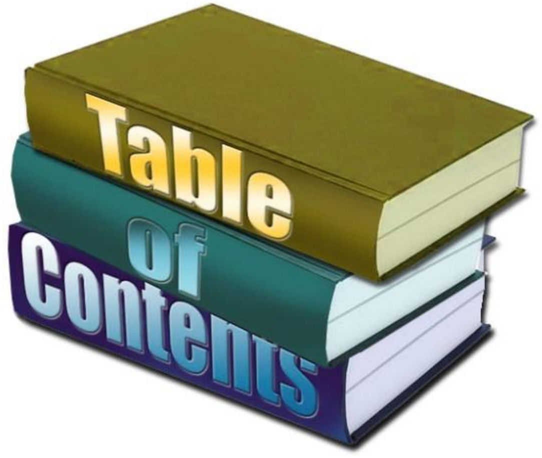
potem definiramo v novem prostoru novi vozlišči (S,E) in (S1, E1)

  - v "dvosmernem" prostoru velja (S,E) → (S1, E1), če obstaja v "enosmernem" prostoru povezava med S → S1 in med E1 → E
  - vozlišče (S,E) je v "dvosmernem" prostoru ciljno vozlišče, če velja E=S ali S → E

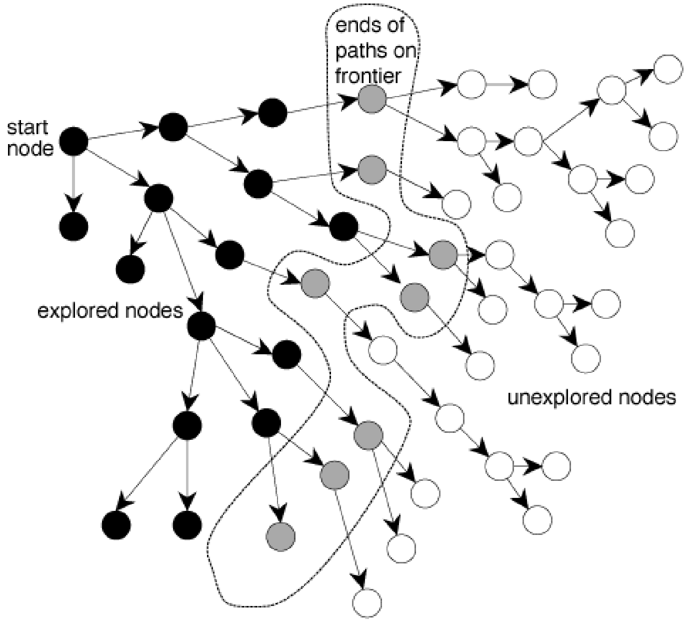
Primerjava časovnih zahtevnosti

Kriterij	Iskanje v širino	Iskanje v globino	Iskanje z omejitvijo globine	Iterativno poglabljanje	Dvosmerno iskanje
Popolnost	Da (če je b končen)	Ne	Ne	Da (če je b končen)	Da
Optimalnost	Da (če so cene enake)	Ne	Ne	Da (če so cene enake)	Da
Čas. zahtevnost	$O(b^d)$	$O(b^m)$	$O(b^l)$	$O(b^d)$	$O(b^{d/2})$
Prost. zahtevnost	$O(b^d)$	$O(bm)$	$O(bl)$	$O(bd)$	$O(b^{d/2})$

Pregled

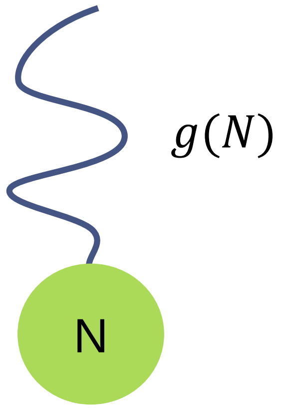


- preiskovanje prostora stanj
  - neinformirani preiskovalni algoritmi
    - iskanje v širino
    - iskanje v globino
    - iterativno poglabljanje
    - dvosmerno iskanje
    - cenovno – optimalno iskanje
  - informirani preiskovalni algoritmi
    - hevristično preiskovanje (primer)
    - požrešno preiskovanje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



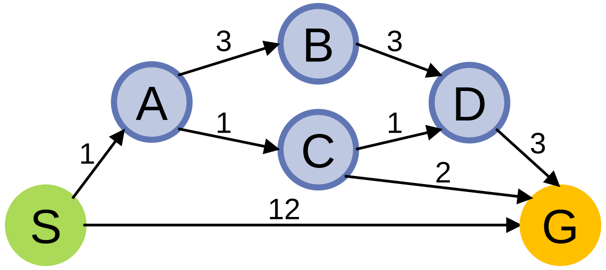
Cenovno-optimalno iskanje

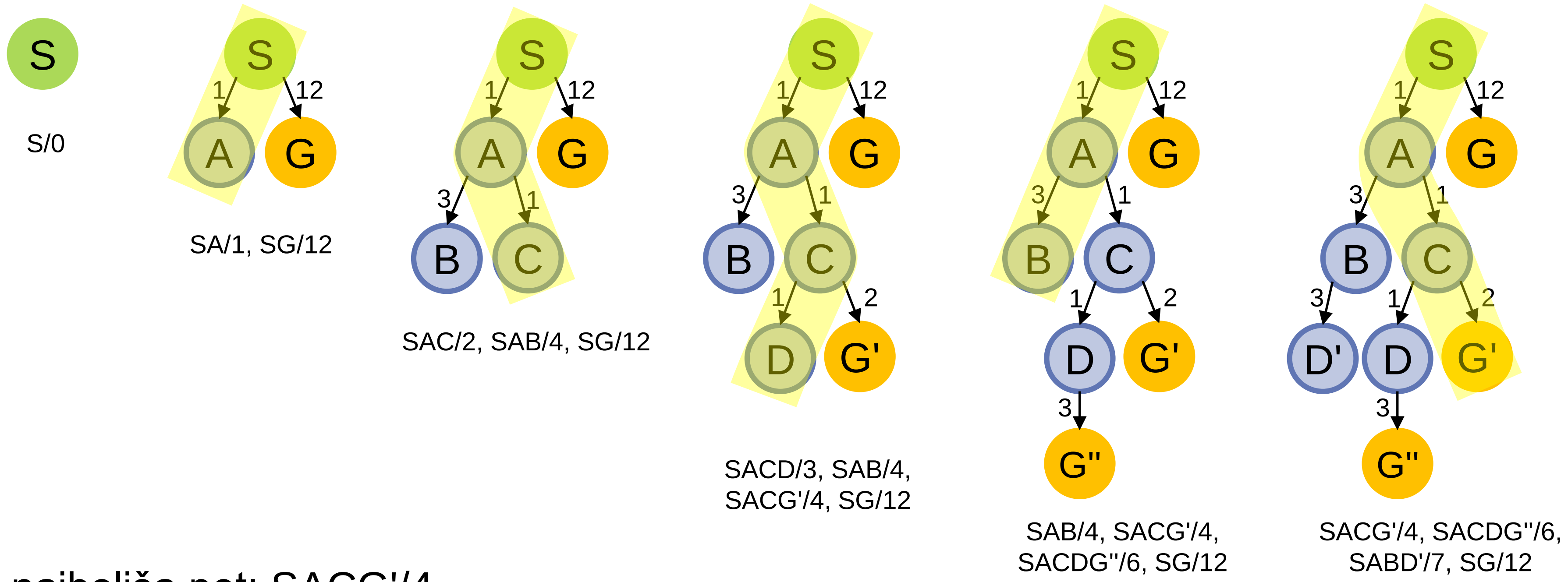
- angl. *uniform-cost search (best-first search with no heuristic)*
- posplošitev **iskanja v širino**
  - iskanje v širino je optimalno, če so cene vseh povezav enake 1
- če cene vseh povezav **niso enake**, je optimalno **razviti vozlišče, ki ima najmanjšo skupno ceno dosedanje poti** – $g(n)$
- fronta je urejena kot prioritetna vrsta
- test, ali je vozlišče ciljno, opravimo šele, ko je vozlišče na vrsti za razvijanje in ne ob generiranju vozlišča
  - zakaj?
  - ciljno vozlišče morda ni optimalno (obstaja boljša rešitev)
  - morda do najdenega optimalnega cilja vodi krajša pot



Cenovno-optimalno iskanje

- primer





- rešitev: najboljša pot: SACG'/4

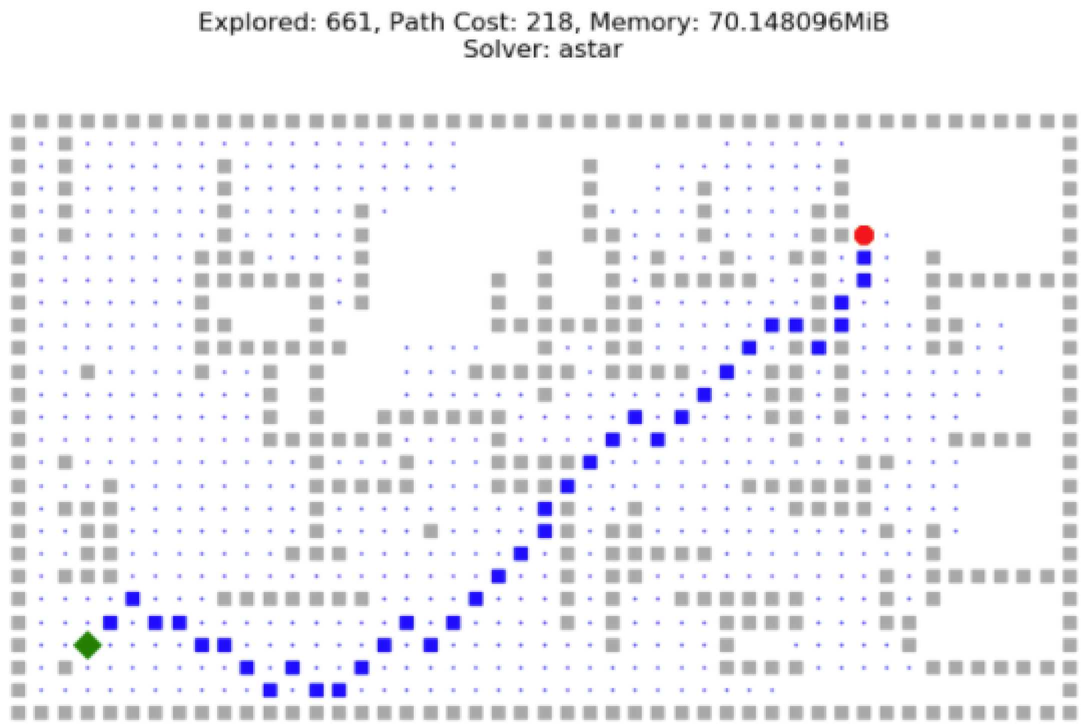
Učinkovitost iskanja

- za potrebe analize predpostavimo, da je prostor stanj drevo:
  - globina (*depth*) optimalne rešitve naj bo $\boldsymbol{d}$
  - stopnja vejanja (*branching factor*) naj bo $\boldsymbol{b}$, na nivoju $\boldsymbol{d}$ imamo torej b^d vozlišč
  - največja globina drevesa naj bo $\boldsymbol{max}$
  - $\boldsymbol{C}^*$ naj bo cena optimalne rešitve
  - $\boldsymbol{\epsilon}$ naj bo najmanjša cena povezave

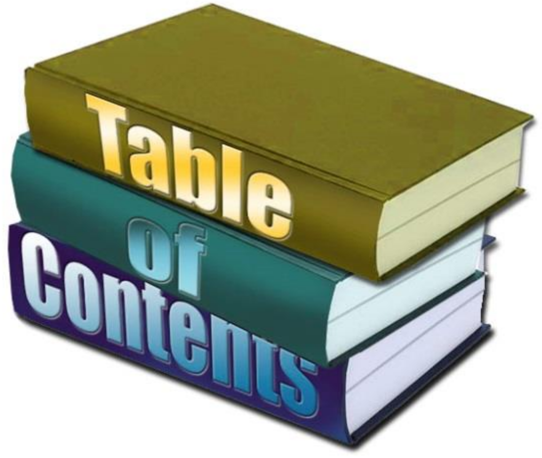
- **POPOLNOST** (angl. *completeness*):
  - Da, za cene povezav > 0.
- **OPTIMALNOST** (angl. *optimality*):
  - Da.
- **ČASOVNA in PROSTORSKA ZAHTEVNOST**:
  - odvisni sta od cen poti in ne samo od globine d in vejanja b
  - zahtevnost $O\left(b^{1+\lfloor C^*/\epsilon \rfloor}\right)$, kar je lahko veliko več kot $O\left(b^d\right)$
  - če so vse cene poti enake, se zahtevnost poenostavi v $O\left(b^{1+d}\right)$
    - zakaj $O\left(b^{d+1}\right)$ in ne $O\left(b^d\right)$?

Demo

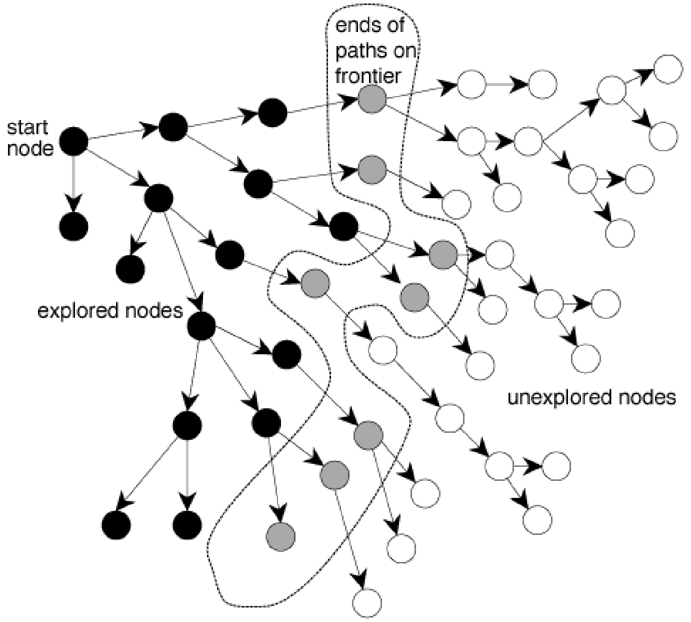
- Searching in AI https://medium.com/analytics-vidhya/searching-in-ai-e05973068c8e



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



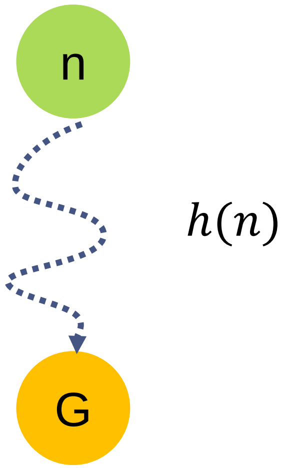
Hevristično preiskovanje

- 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)

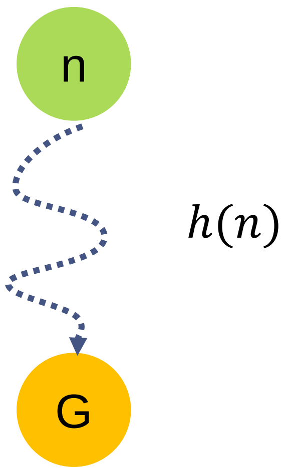
$$h: vozlišče \rightarrow \mathbb{R}$$

- 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

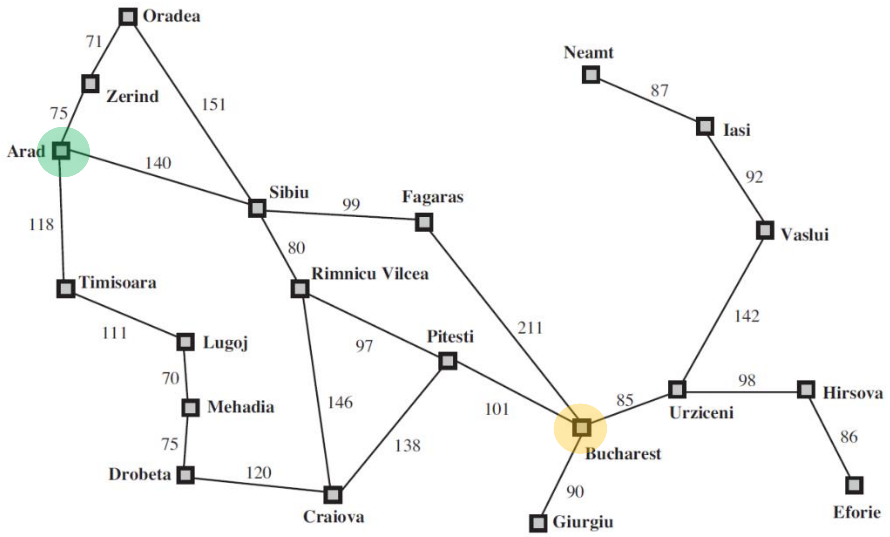


- angl. *greedy best-first search*
- vedno razvijemo najbolj obetavno vozlišče **glede na hevristično oceno**
- vrednotenje vozlišča: $f(n) = h(n)$

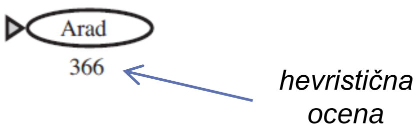
- primer: iskanje optimalne poti z upoštevanjem najkrajše zračne razdalje
- zračne razdalje do cilja (Bukarešta) lahko uporabimo kot hevristične ocene:

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

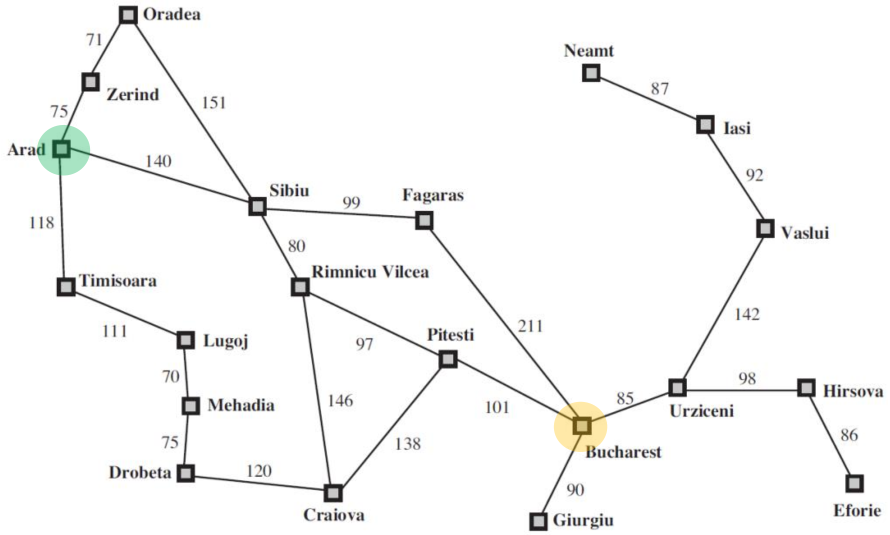
Požrešno iskanje: primer



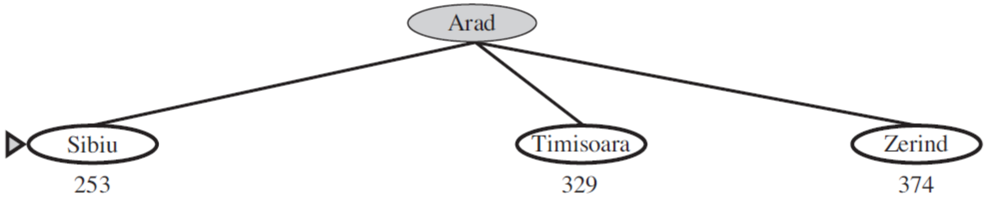
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



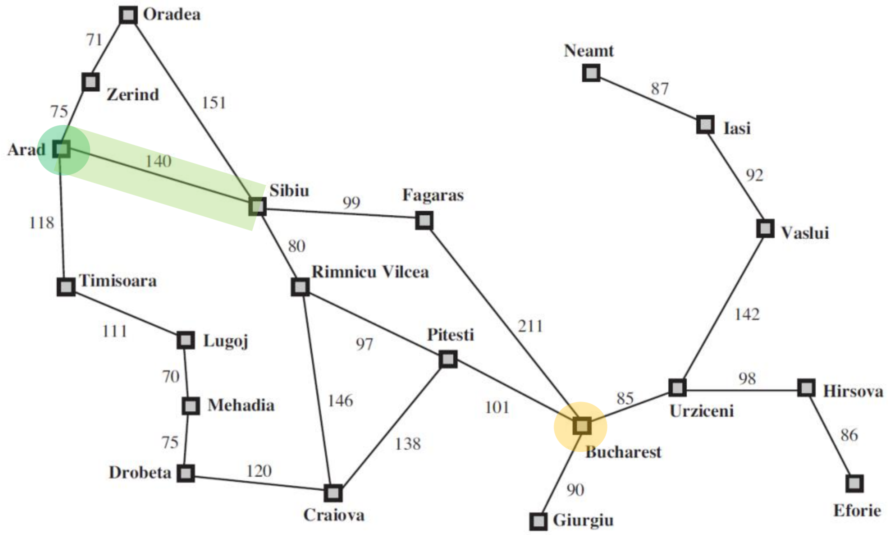
Požrešno iskanje: primer



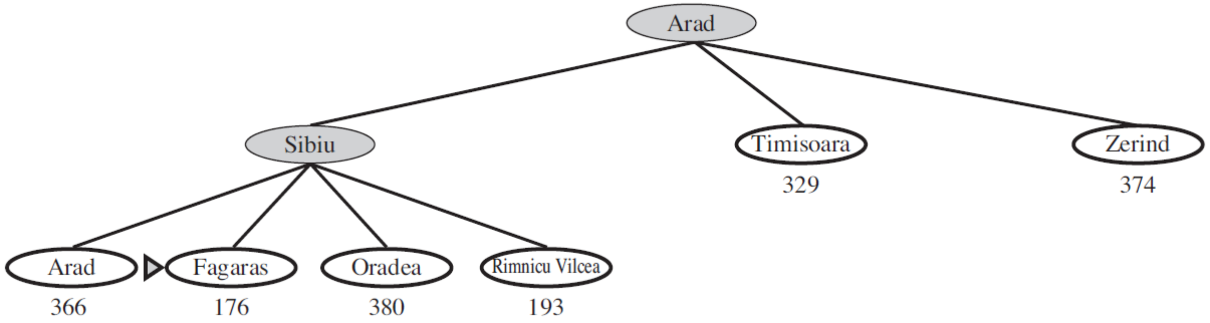
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374



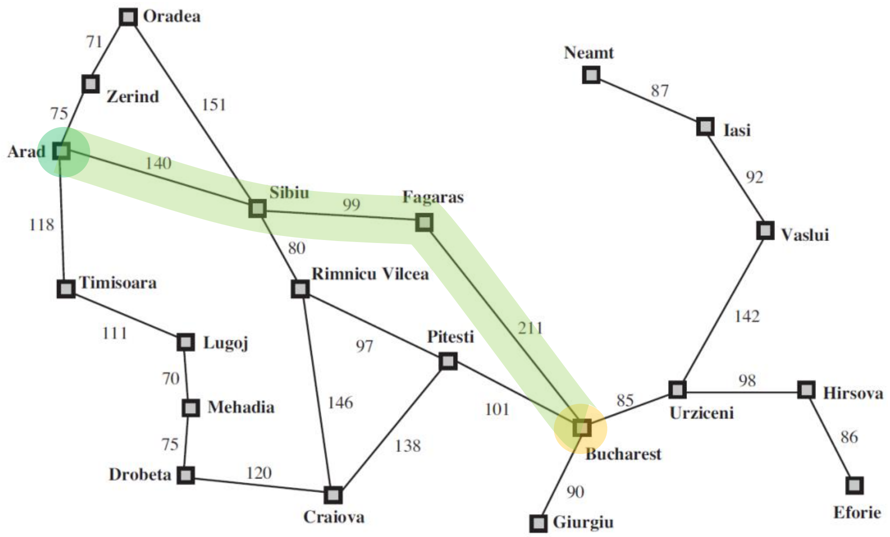
Požrešno iskanje: primer



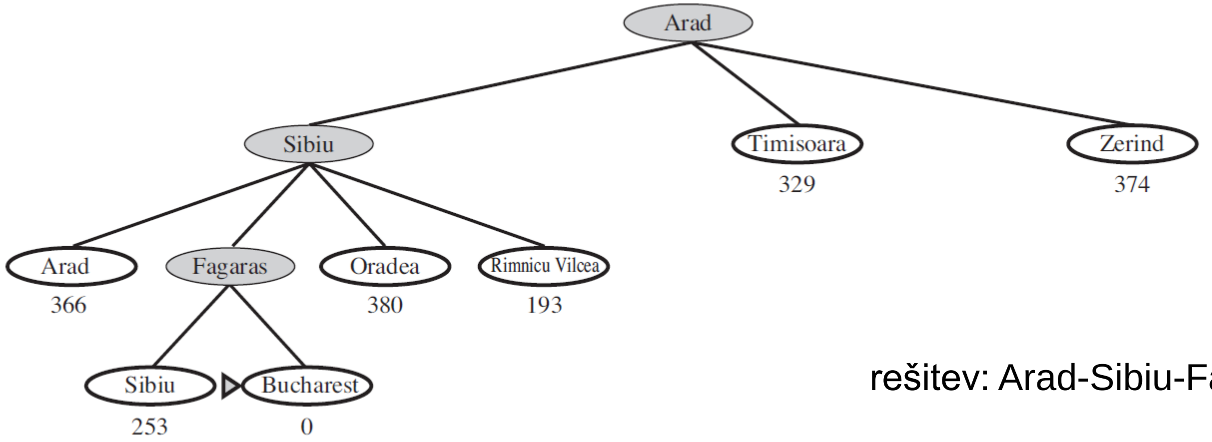
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



Požrešno iskanje: primer



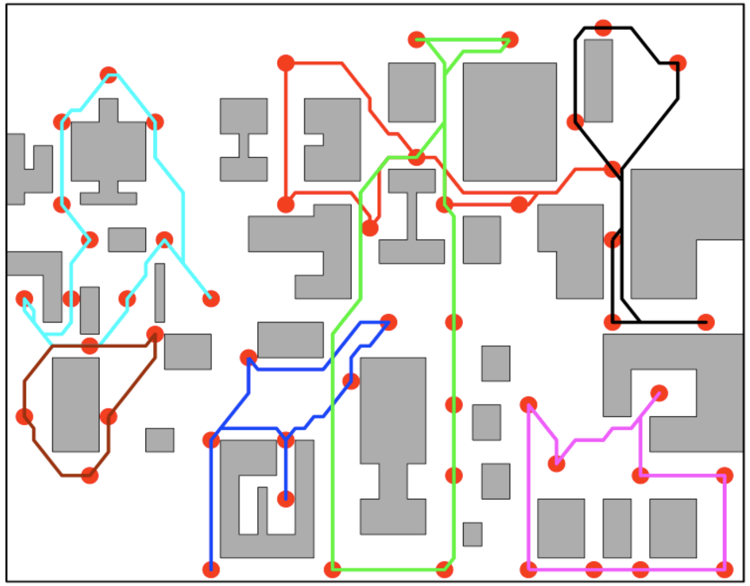
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

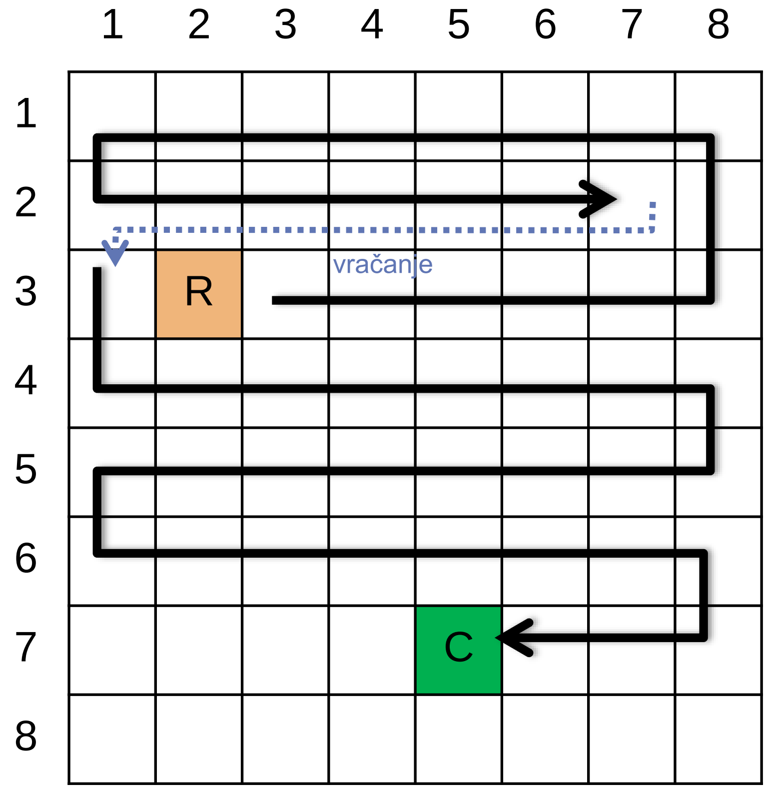


rešitev: Arad-Sibiu-Fagaras-Bucharest, cena=450

Požrešno iskanje: primer

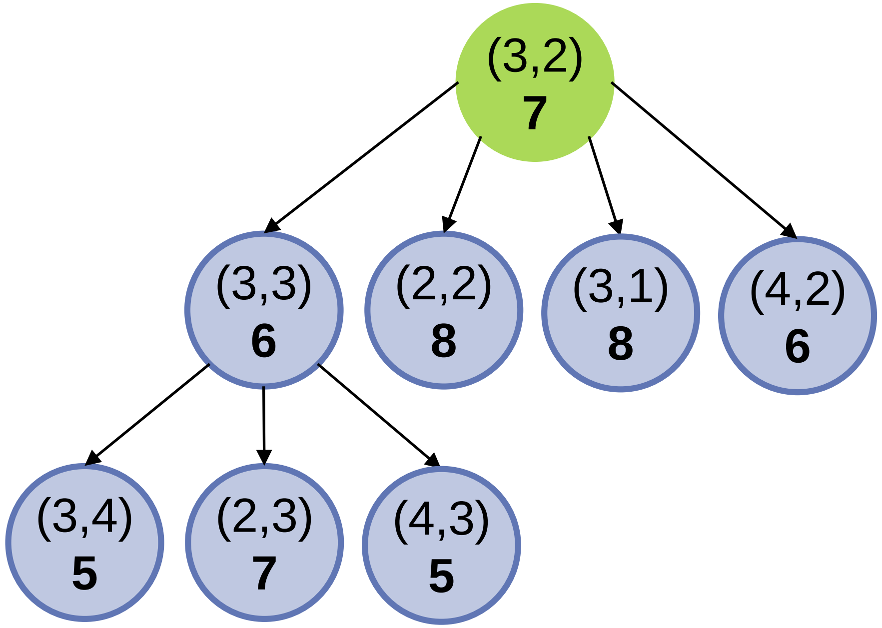
- primer 2: robotovo iskanje ciljne lokacije (Bratko, OUI, 2016/17)
- R možni premiki: 1 $\rightarrow$, 2 $\uparrow$, 3 $\leftarrow$, 4 $\downarrow$
- dolžina rešitve pri **preiskovanju v globino** je: 45
- optimalna dolžina rešitve: 7



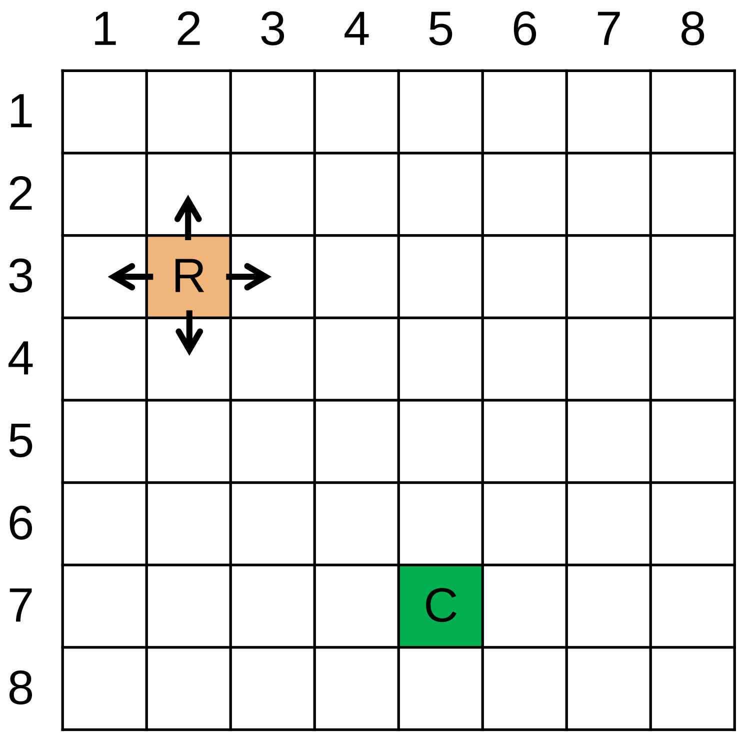


Požrešno iskanje: primer

- 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.



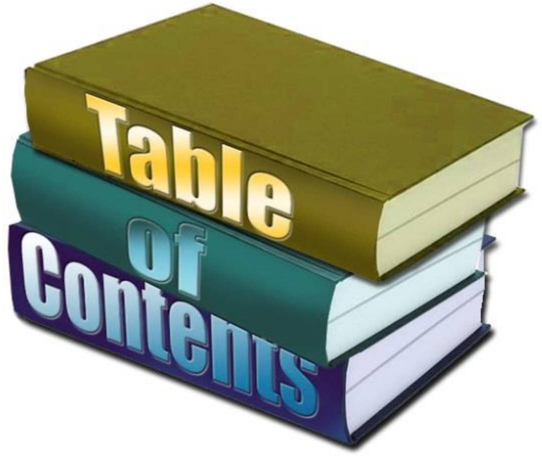
Učinkovitost požrešnega iskanja

- **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?

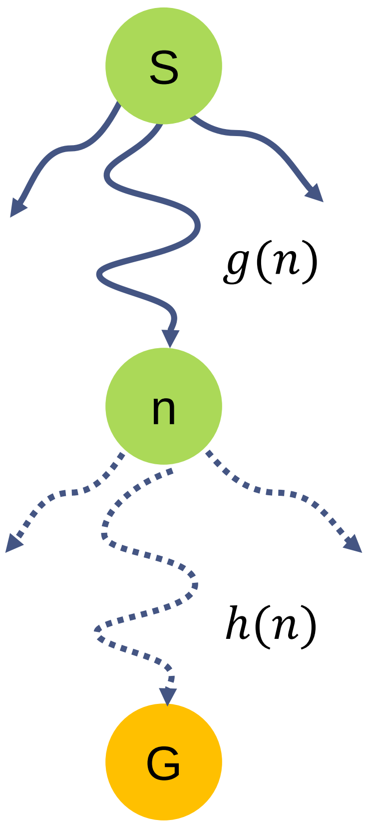
- **ČASOVNA in PROSTORSKA ZAHTEVNOST**:
  - $O(b^m)$, kjer je m največja globina drevesa
  - vsa vozlišča moramo hraniti v spominu, ker so kandidati za razvijanje nadaljnje poti
  - pomembnost ustrezne hevristične ocene (!)

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:

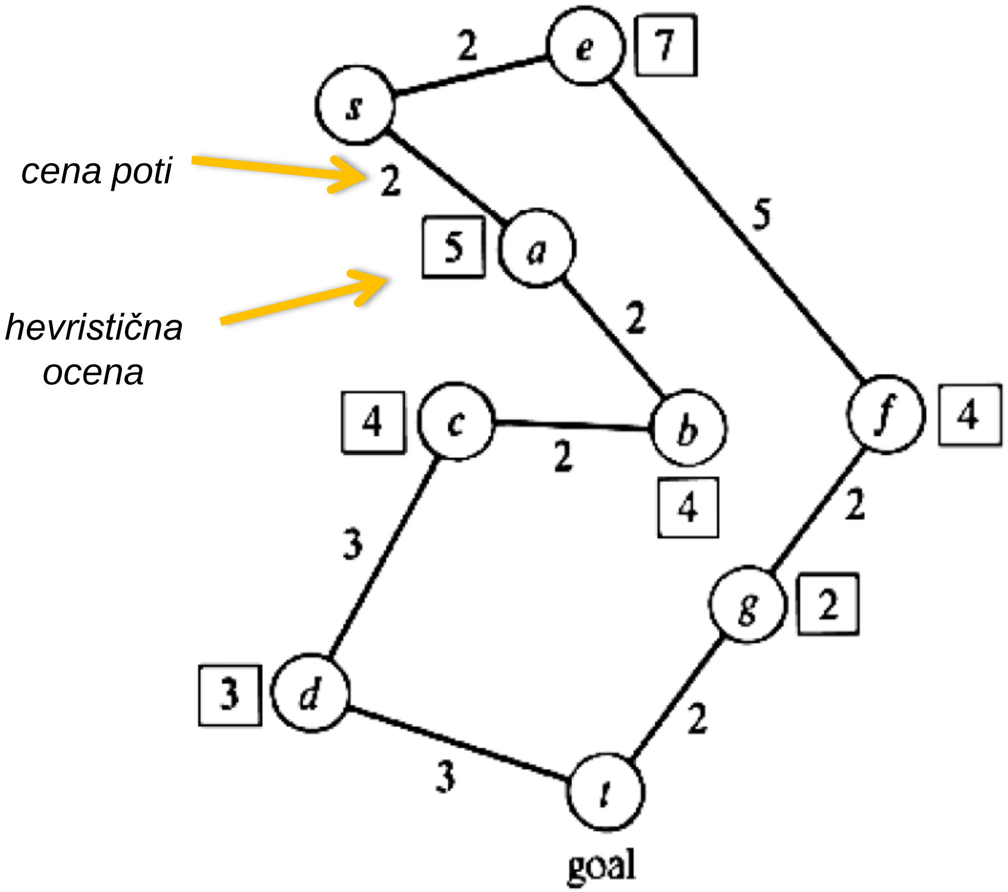
  $$\boldsymbol{f(n) = g(n) + h(n)}$$

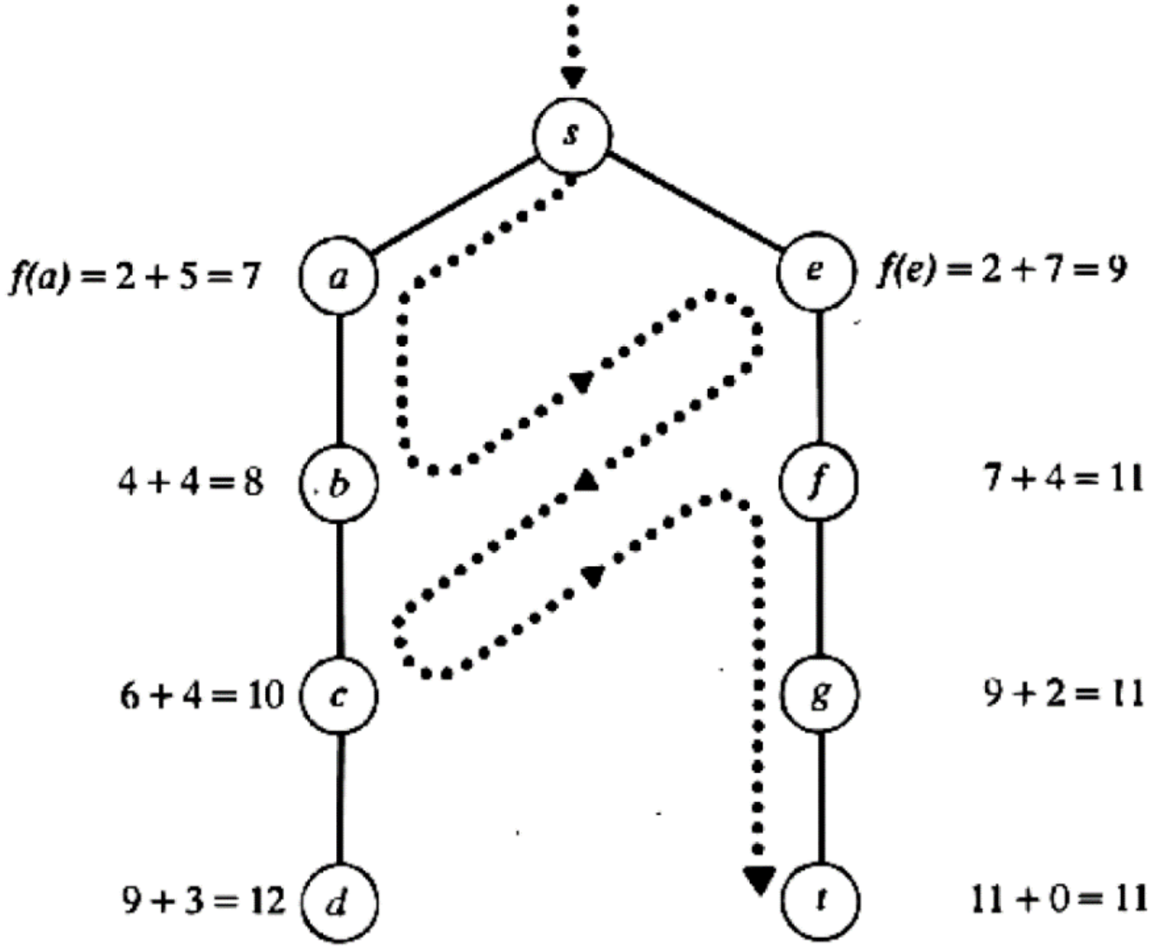
  $\mathrm{g}(n)$ – cena poti do n (znano)

  $h(n)$ – 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

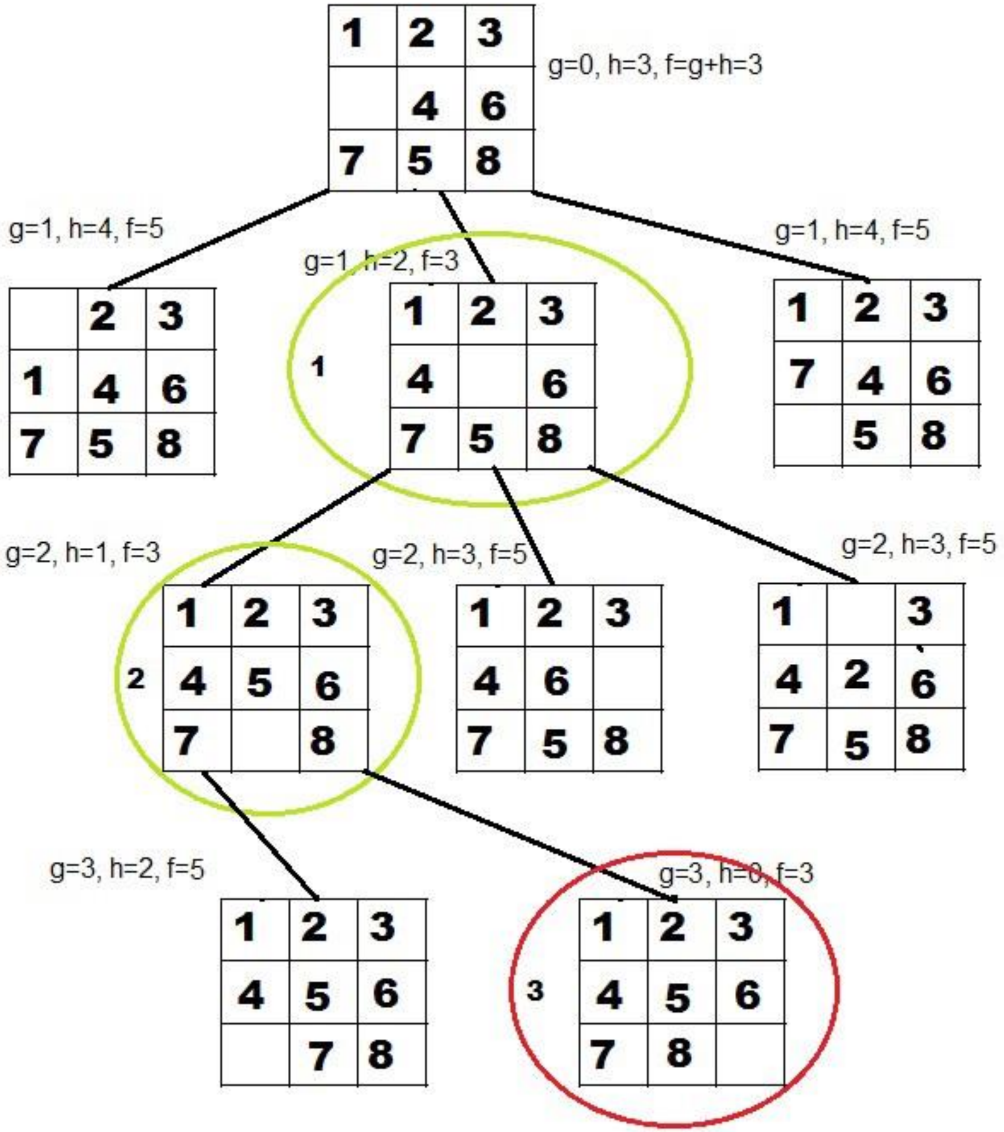
- primer 1: preiskovanje manjšega grafa





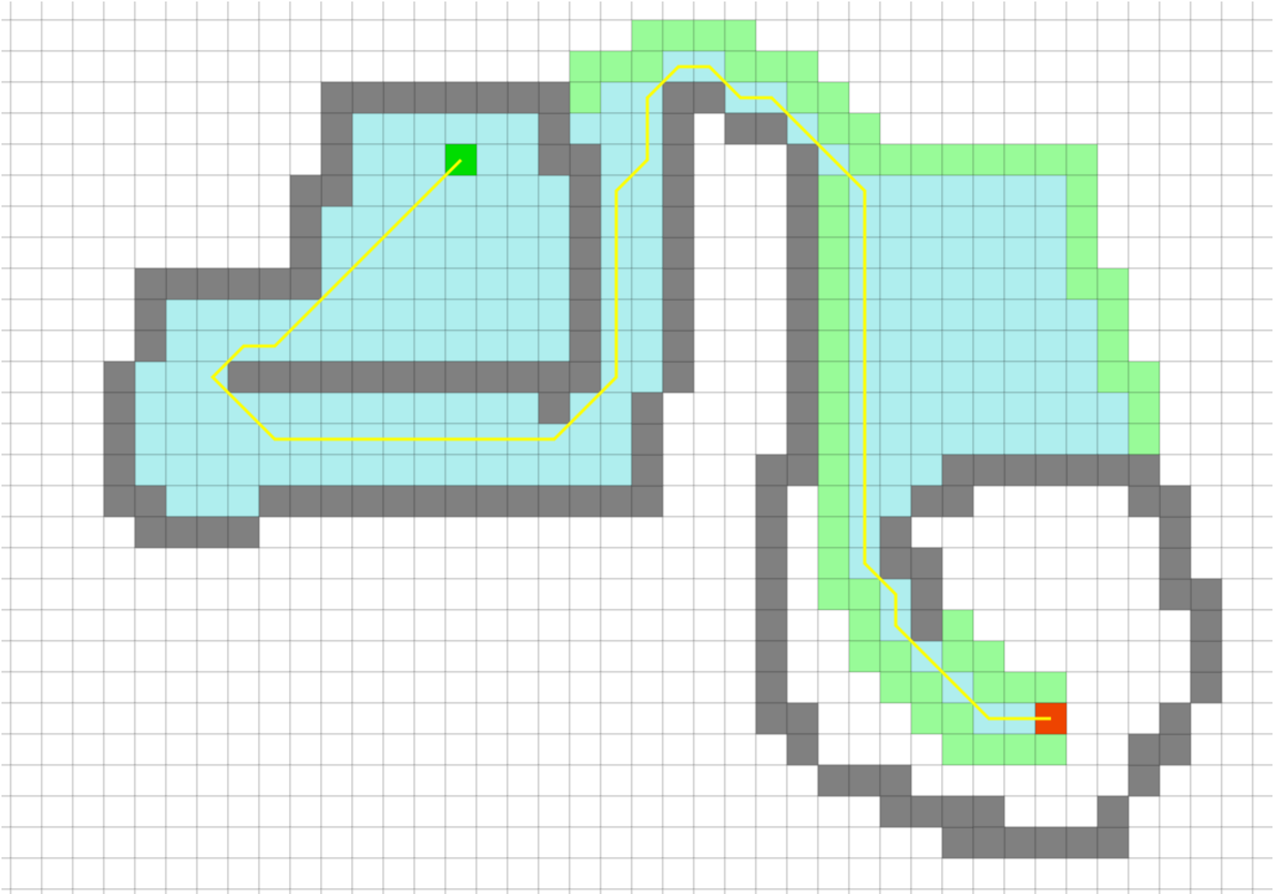
Algoritem A*: primer

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

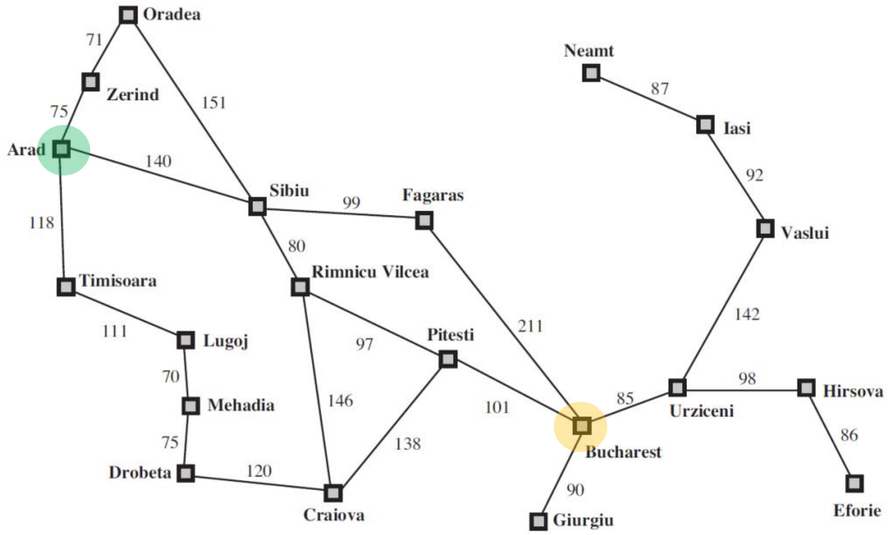


Algoritem A*: primer

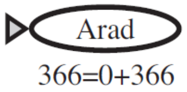
- primer 3: https://qiao.github.io/PathFinding.js/visual/



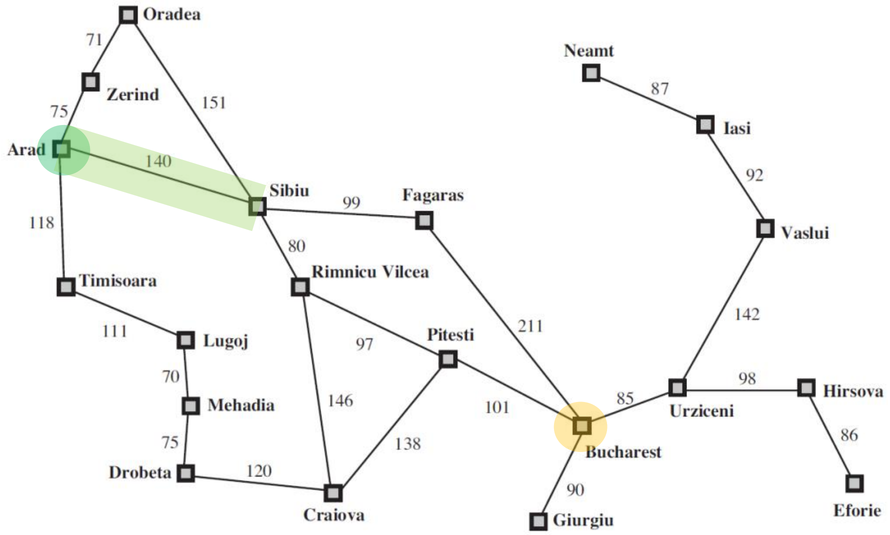
Algoritem A*: primer



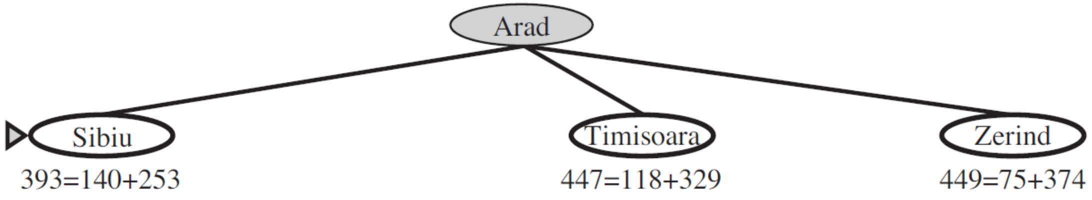
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



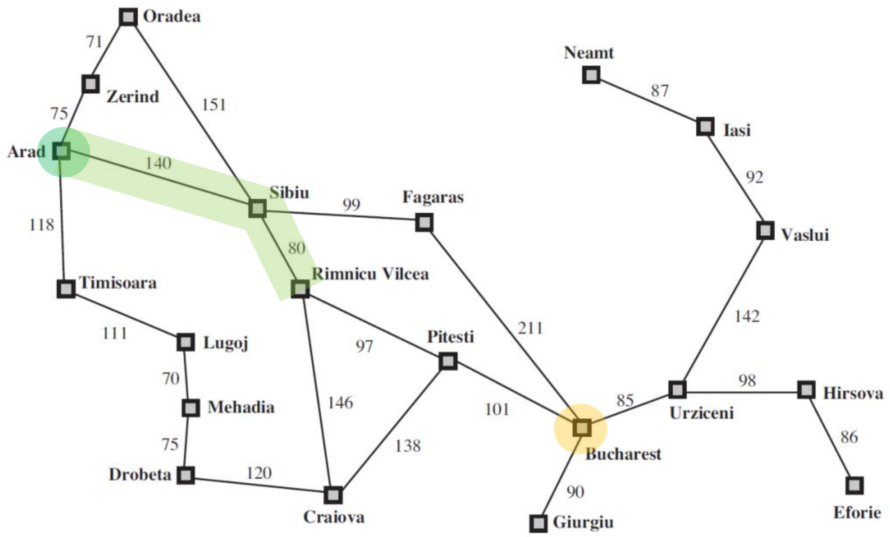
Algoritem A*: primer



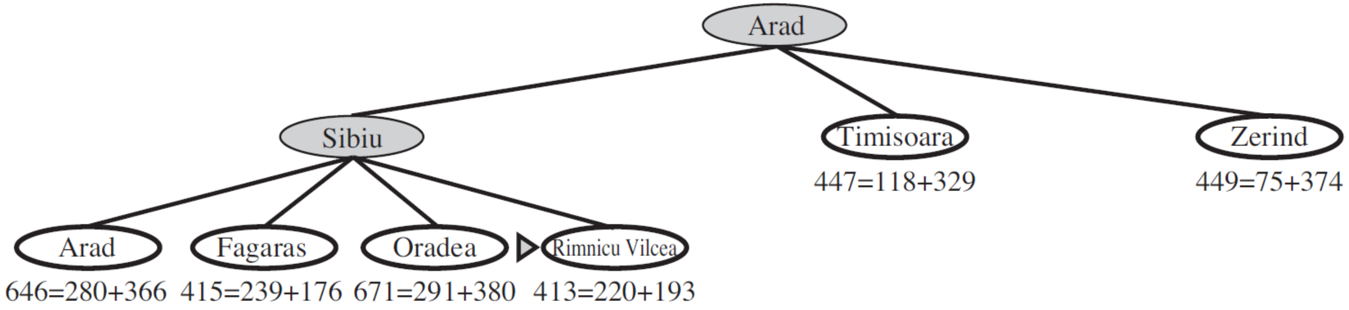
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374



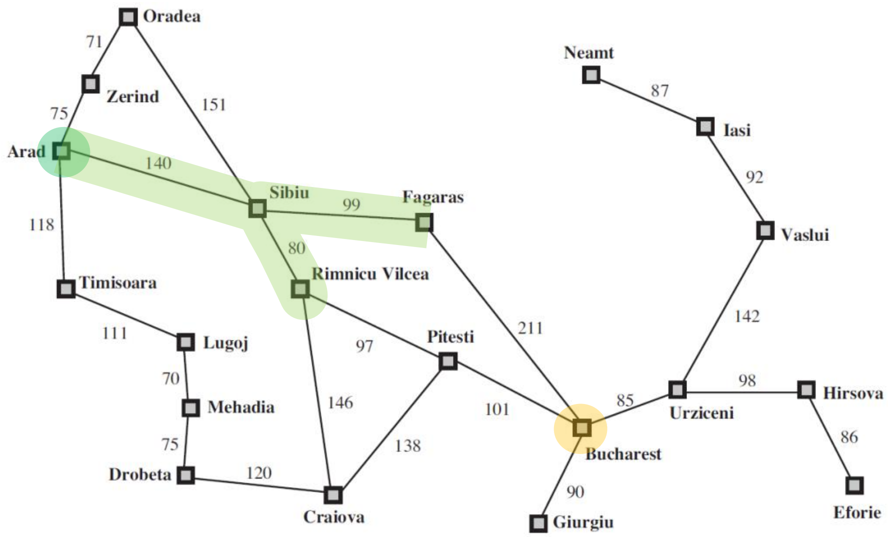
Algoritem A*: primer



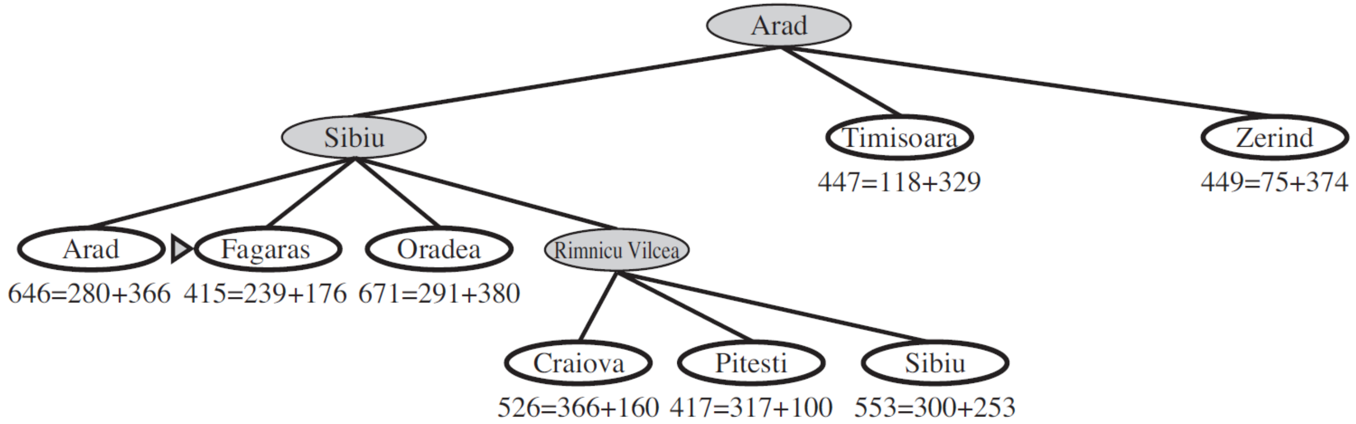
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



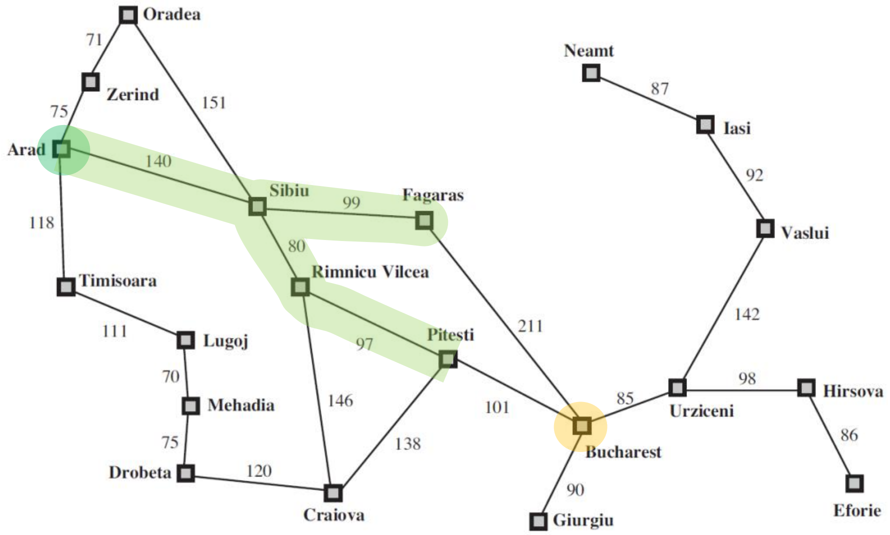
Algoritem A*: primer



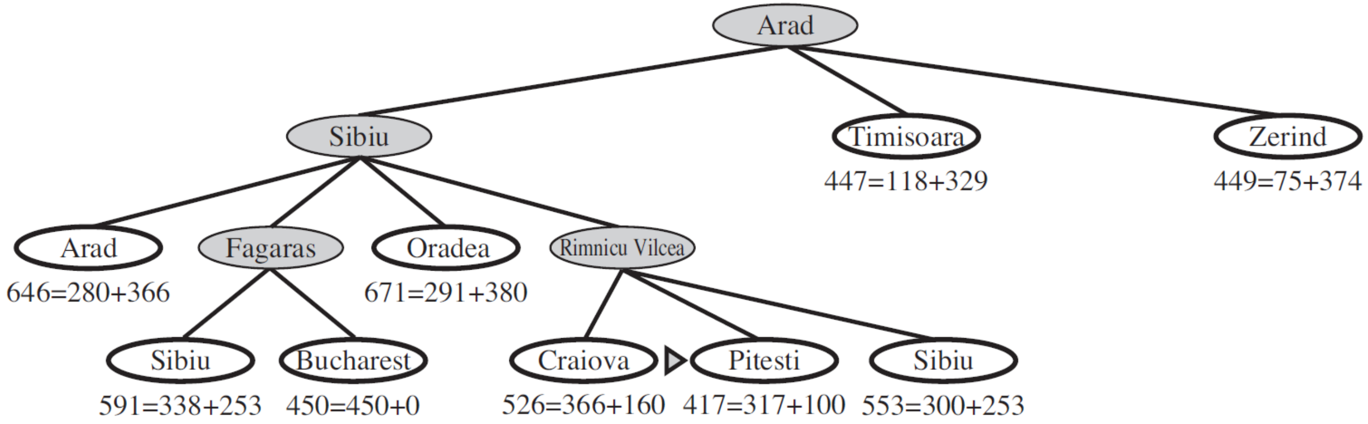
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374



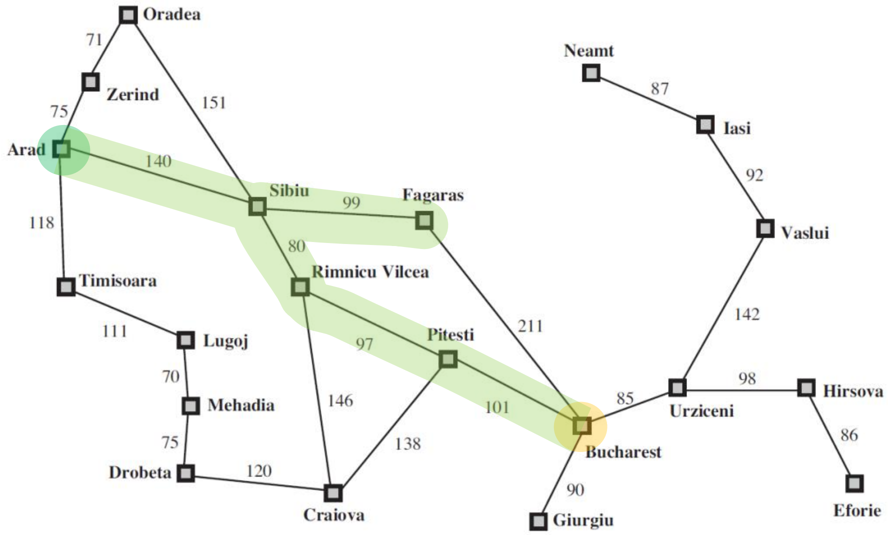
Algoritem A*: primer



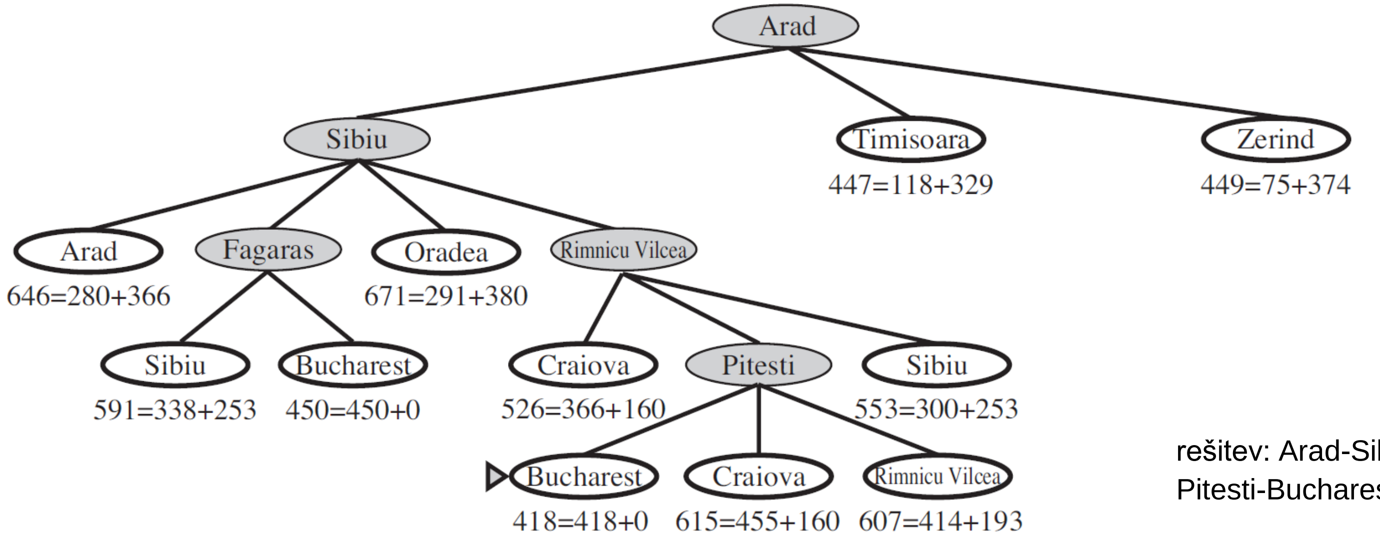
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374



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

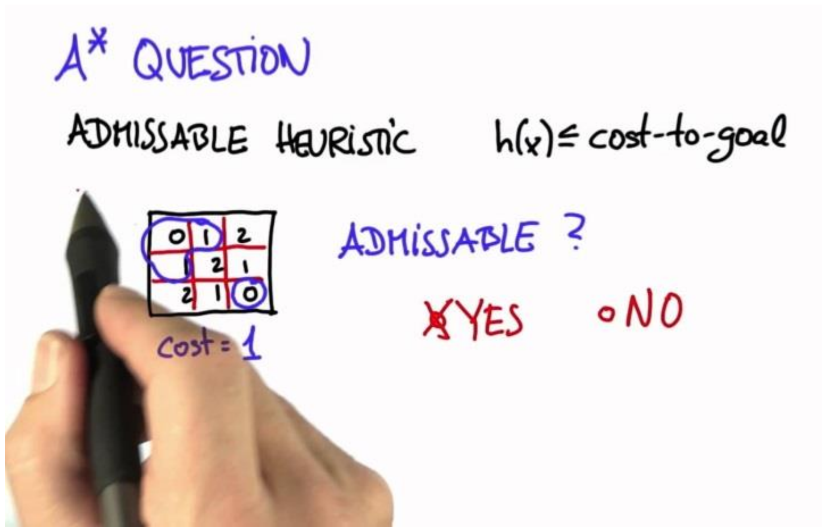


rešitev: Arad-Sibiu-Riminicu Vilcea-Pitesti-Bucharest, cena=418

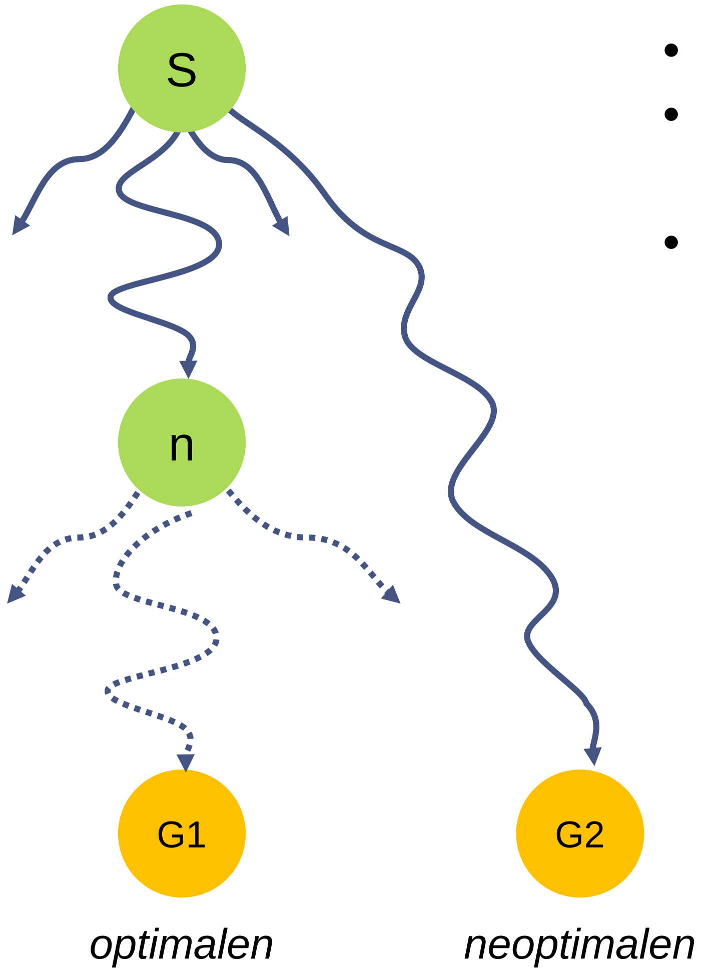
Popolnost in optimalnost A*

- 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*



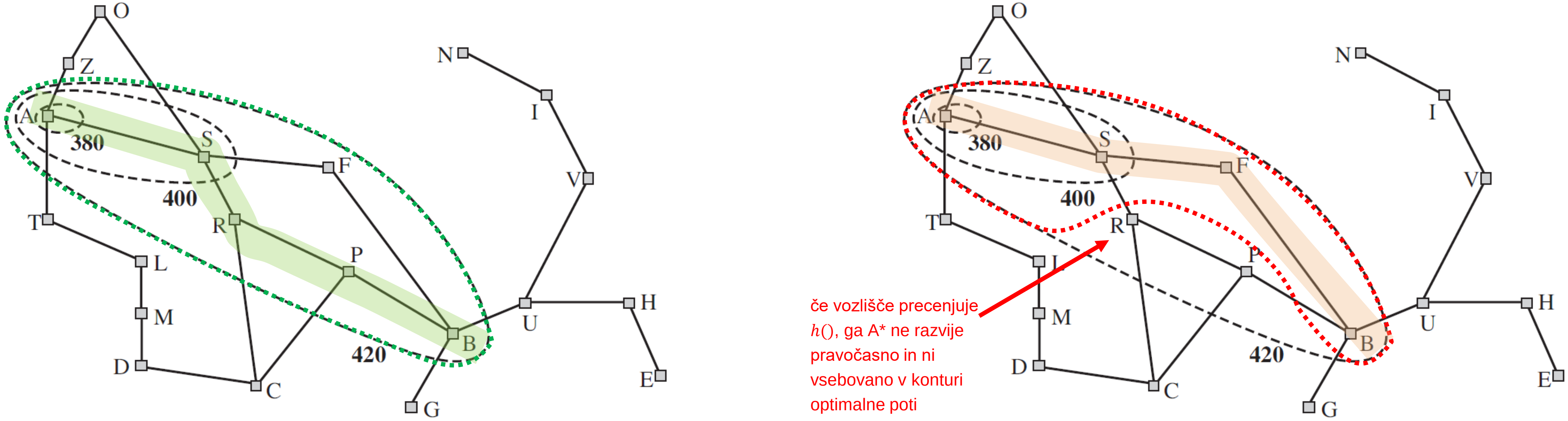
- denimo, da je $G1$ je optimalen cilj, $G2$ neoptimalen ($g(G1) < g(G2)$)
- denimo, da je $h(n)$ dopustna cenilka ($h(n) \leq h^*(n)$)

- 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
  - n naj bo neko vmesno vozlišče na poti do optimalnega cilja $G1$
  - ker je $h(n)$ dopustna cenilka, mora veljati:

    $g(n) < g(G1) < g(G2)$, ker je n na poti do G1 in ker je G1 optimalen
    $g(n) < f(G1) < f(G2)$, zaradi zgornjih enakosti
    $f(n) \leq f(G1) < f(G2)$, ker $h(n) \leq h^*(n) = \mathrm{g}(G1) - \mathrm{g}(n)$
  - zato 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**)


Informirano preiskovanje...