Umjetna inteligencija

3. Heuristično pretraživanje

prof. dr. sc. Bojana Dalbelo Bašić
doc. dr. sc. Jan Šnajder

Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Ak. god. 2012./2013.

v3.2

Motivacija

- Slijepi postupci raspolažu isključivo egzaktnim informacijama (početnim stanjem, operatorima i ispitnim predikatom)
- Ne koriste nikakvu dodatnu **informaciju o prirodi problema** koja bi mogla poboljšati učinkovitost pretraživanja
- Ako otprilike znamo u kojem se smjeru nalazi rješenje, zašto ne iskoristiti to znanje kako bismo ubrzali pretragu?

Heuristika

- **Heuristika** – iskustvena pravila o prirodi problema i osobinama cilja čija je svrha pretraživanje brže **usmjeriti** k cilju

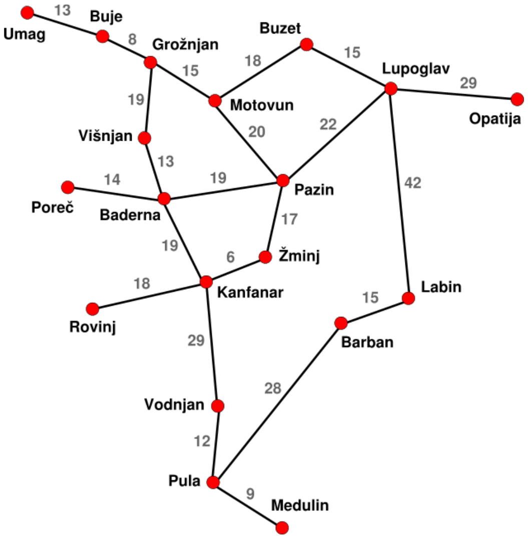
Heuristička funkcija

Heuristička funkcija $h : S \to \mathbb{R}^+$ pridjeljuje svakom stanju $s \in S$ procjenu udaljenosti od tog stanja do ciljnog stanja

Što je vrijednost $h(s)$ manja, to je čvor s bliži ciljnome stanju. Ako je s ciljno stanje, onda $h(s) = 0$

- Postupke pretraživanja koji koriste heuristiku kako bi suzili prostor pretraživanja nazivamo **heurističkim** ili **usmjerenim**

Primjer: Putovanje kroz Istru



Zračne udaljenosti do Buzeta:

Baderna	25
Barban	35
Buje	21
Grožnjan	17
Kanfanar	30
Labin	35
Lupoglav	13
Medulin	61
Motovun	12
Opatija	26
Pazin	17
Poreč	32
Pula	57
Rovinj	40
Umag	31
Višnjan	20
Vodnjan	47
Žminj	27

Primjer: Slagalica 3×3

početno stanje:

8		**7**
6	**5**	**4**
3	**2**	**1**

ciljno stanje:

1	**2**	**3**
4	**5**	**6**
7	**8**	

Heuristička funkcija?

- Broj pločica koje nisu na svome mjestu:

  $$\mathrm{h}_1\left(\begin{array}{|c|c|c|}\hline 8 & \blacksquare & 7 \\ \hline 6 & 5 & 4 \\ \hline 3 & 2 & 1 \\ \hline\end{array}\right) = 7$$

- Zbroj Manhattan-udaljenosti (L1) pločica od svoga mjesta:

  $$\mathrm{h}_2\left(\begin{array}{|c|c|c|}\hline 8 & \blacksquare & 7 \\ \hline 6 & 5 & 4 \\ \hline 3 & 2 & 1 \\ \hline\end{array}\right) = 21$$

Vrijedi $\mathrm{h}_2(s) \geq \mathrm{h}_1(s)$

Heuristično pretraživanje

Razmotrit ćemo:

1. Pretraživanje "najbolji prvi" (engl. *greedy best-first search*)
2. Algoritam A^*
3. Pretraživanje usponom na vrh (engl. *hill-climbing search*)

Pretraživanje "najbolji prvi"

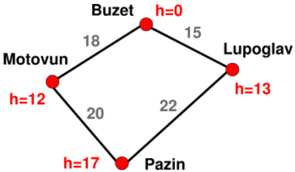
- Proširuje čvor s najboljom heurističkom vrijednošću

Pretraživanje "najbolji prvi"

function $\text{greedyBestFirstSearch}(s_0, \text{succ}, \text{goal}, h)$
  $open \leftarrow [\,\text{initial}(s_0)\,]$
  while $open \neq [\,]$ **do**
    $n \leftarrow \text{removeHead}(open)$
    if $\text{goal}(\text{state}(n))$ **then return** n
    for $m \in \text{expand}(n, \text{succ})$ **do**
      $\text{insertSortedBy}(\mathsf{f}, m, open)$
  return fail
where $\mathsf{f}(n) = h(\text{state}(n))$

Pohlepno pretraživanje

- Ovo je tzv. **pohlepan** (engl. *greedy*) algoritam "najbolji prvi"
- **Pohlepno pretraživanje**: odabire se onaj čvor koji se čini najbliži cilju, ne uzimajući u obzir ukupnu cijenu puta
- Odabrani put možda nije optimalan, no algoritam nema mogućnost oporavka od pogreške! Dakle, algoritam **nije optimalan**
- **Q:** Primjer?



- **Nije potpun** (osim ako koristimo listu posjećenih stanja)
- **Vremenska i prostorna složenost:** $\mathcal{O}(b^m)$

Algoritam A^*

- Algoritam "najbolji prvi" koji u obzir uzima i heuristiku i cijenu ostvarenog puta, tj. kombinira "najbolji prvi" i pretraživanje s jednolikom cijenom
- Kao i kod pretraživanja s jednolikom cijenom, pri proširenju čvora ažurira se cijena do tada ostvarenog puta:

function $\mathrm{expand}(n, \mathrm{succ})$
  return $\{\, (s, \mathrm{g}(n) + c) \mid (s, c) \in \mathrm{succ}(\mathrm{state}(n)) \,\}$

Ukupna cijena računa se na temelju:

$\mathrm{g(n)}$ – **stvarna cijena** puta od početnog čvora do čvora n

$\mathrm{h(s)}$ – **procjena cijene** puta od stanja s do cilja

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

Podsjetnik: opći algoritam pretraživanja

Opći algoritam pretraživanja

function $\mathrm{search}(s_0, \mathrm{succ}, \mathrm{goal})$
  $open \leftarrow [\,\mathrm{initial}(s_0)\,]$
  while $open \neq [\,]$ **do**
    $n \leftarrow \mathrm{removeHead}(open)$
    if $\mathrm{goal}(\mathrm{state}(n))$ **then return** n
    for $m \in \mathrm{expand}(n, \mathrm{succ})$ **do**
      $\mathrm{insert}(m, open)$
  return $fail$

Algoritam A^* – izvedba

Algoritam A^*

function $\mathrm{aStarSearch}(s_0, \mathrm{succ}, \mathrm{goal}, h)$
  $open \leftarrow [\,\mathrm{initial}(s_0)\,]$
  $closed \leftarrow \varnothing$
  while $open \neq [\,]$ **do**
    $n \leftarrow \mathrm{removeHead}(open)$
    if $\mathrm{goal}(\mathrm{state}(n))$ **then return** n
    $closed \leftarrow closed \cup \{\, n \,\}$
    for $m \in \mathrm{expand}(n)$ **do**
      if $\exists m' \in closed \cup open$ **such that** $\mathrm{state}(m') = \mathrm{state}(m)$ **then**
        if $\mathrm{g}(m') < \mathrm{g}(m)$ **then continue**
        else $\mathrm{remove}(m', closed \cup open)$
      $\mathrm{insertSortedBy}(\mathrm{f}, m, open)$
  return $fail$
where $f(n) = g(n) + h(\mathrm{state}(n))$

Algoritam A^* – primjer izvođenja

0. $open = [\,(Pula, 0)\,]$
   $closed = \varnothing$
1. $\text{expand}(Pula, 0) = \{(Vodnjan, 12), (Barban, 28), (Medulin, 9)\}$
   $open = [\,(Vodnjan, 12)^{59}, (Barban, 28)^{63}, (Medulin, 9)^{70}\,]$
   $closed = \{(Pula, 0)\}$
2. $\text{expand}(Vodnjan, 12) = \{(Kanfanar, 41), (Pula, 24)\}$
   $open = [\,(Barban, 28)^{63}, (Medulin, 9)^{70}, (Kanfanar, 41)^{71}\,]$
   $closed = \{(Pula, 0), (Vodnjan, 12)\}$
3. $\text{expand}(Barban, 28) = \{(Labin, 43), (Pula, 56)\}$
   $open = [\,(Medulin, 9)^{70}, (Kanfanar, 41)^{71}, (Labin, 43)^{78}\,]$
   $closed = \{(Barban, 28), (Pula, 0), (Vodnjan, 12)\}$
4. $\text{expand}(Medulin, 9) = \{(Pula, 18)\}$
   $open = [\,(Kanfanar, 41)^{71}, (Labin, 43)^{78}\,]$
   $closed = \{(Barban, 28), (Medulin, 9), (Pula, 0), (Vodnjan, 12)\}$

   $\vdots$

Algoritam A^* – svojstva

- **Vremenska i prostorna složenost:** $\mathcal{O}(\min(b^{d+1}, b|S|))$ (u praksi veći problem predstavlja prostorna složenost)
- **Potpunost:** da, jer u obzir uzima cijenu puta
- **Optimalnost:** da, ali pod uvjetom da je heuristika h optimistična:

Optimističnost heuristike

Heuristika h je **optimistična** ili **dopustiva** (engl. *optimistic, admissible*) akko nikad ne precjenuje, tj. nikad nije veća od prave cijene do cilja:

$$\forall s \in S.\ h(s) \leq h^*(s),$$

gdje je $h^*(s)$ prava cijena od stanja s do cilja

- Ako heuristika nije optimistična, može se dogoditi da pretraga zaobiđe optimalni put jer se on čini skupljim nego što zapravo jest
- **Q:** Jesu li heuristike u ranijim primjerima optimistične?

Primjer: Slagalica 3×3

početno stanje:

8		**7**
6	**5**	**4**
3	**2**	**1**

ciljno stanje:

1	**2**	**3**
4	**5**	**6**
7	**8**	

Koje su heuristike optimistične?

- $\mathrm{h}_1(s) =$ broj pločica koje nisu na svome mjestu
- $\mathrm{h}_2(s) =$ zbroj Manhattan-udaljenosti pločica od svoga mjesta
- $\mathrm{h}_3(s) = 0$
- $\mathrm{h}_4(s) = 1$
- $\mathrm{h}_5(s) = h^*(s)$
- $\mathrm{h}_6(s) = \min(2, h^*(s))$
- $\mathrm{h}_7(s) = \max(2, h^*(s))$

Konzistentna heuristika (1)

- Uz pretpostavku optimistične heuristike, vrijedi $\mathrm{f}(n) \leq C^*$ (funkcija cijene je odozgo ograničena)
- Duž staze u stablu pretraživanja, $\mathrm{f}(n)$ može općenito rasti i padati, a u ciljnom stanju vrijedi $\mathrm{f}(n) = \mathrm{g}(n) = C^*$
- Poželjno je da $\mathrm{f}(n)$ **monotono raste**:

$$\forall n_2 \in \mathrm{expand}(n_1) \implies \mathrm{f}(n_2) \geq \mathrm{f}(n_1)$$

(Za savršenu heuristiku h^* vrijedi $\mathrm{f}(n_1) = \mathrm{f}(n_2) = C^*$)

- Ako $\mathrm{f}(n)$ monotono raste, svaki čvor koji **prvi generiramo** za neko stanje bit će čvor s **najmanjom cijenom** za to stanje
- To znači, kod ponovljenih stanja, ne moramo provjeravati cijenu već ztvorenih čvorova (ona će sigurno biti manja ili jednaka)

Konzistentna heuristika (2)

- Ako $\mathrm{f}(n)$ monotono raste, onda $\forall n_2 \in \mathrm{expand}(n_1)$:

$$\mathrm{f}(n_1) \leq \mathrm{f}(n_2)$$

$$\mathrm{g}(n_1) + h(\underbrace{\mathrm{state}(n_1)}_{s_1}) \leq \mathrm{g}(n_2) + h(\underbrace{\mathrm{state}(n_2)}_{s_2})$$

$$\mathrm{g}(n_1) + h(s_1) \leq \underbrace{\mathrm{g}(n_1) + c}_{\mathrm{g}(n_2)} + h(s_2)$$

$$h(s_1) \leq h(s_2) + c$$

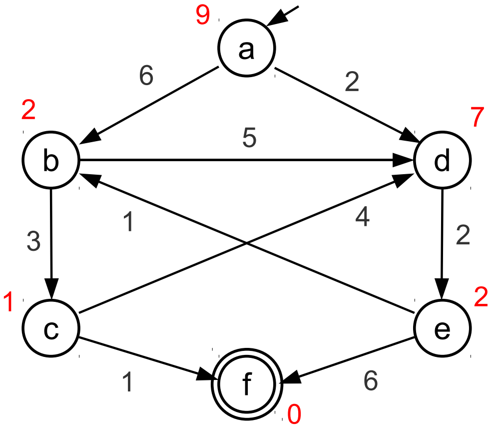
Konzistentnost heuristike

Heuristika h je **konzistentna** ili **monotona** akko:

$$\forall (s_2, c) \in \mathrm{succ}(s_1).\ h(s_1) \leq h(s_2) + c$$

- **NB:** Konzistentna heuristika je nužno optimistična, a u praksi su optimistične heuristike ujedno i konzistentne

Konzistentna heuristika – primjer



Q: Je li ova heuristika optimistična? Je li konzistentna?

Algoritam A^* – varijante

Sljedeće varijante algoritma A^* također su **optimalne**:

1. Varijanta bez liste *closed* (u praksi memorijski zahtjevnija zbog ponavljanja stanja)
2. Varijanta s listom *closed*, ali bez otvaranja jednom zatvorenih čvorova, uz uvjet da je heuristička funkcija **konzistentna**
3. S listom *closed* i bez otvaranja, ali uz **pathmax-korekciju**:

$$f(n) = \max(f(\text{parent}(n)), g(n) + h(\text{state}(n)))$$

Svojstvo dominacije

Dominacija

Neka su A_1^* i A_2^* dva optimalna algoritma s *optimističnim* heurističkim funkcijama h_1 i h_2. Algoritam A_1^* **dominira** nad algoritmom A_2^* akko:

$$\forall s \in S.\ h_1(s) \geq h_2(s)$$

Također kažemo da je algoritam A_1^* **obavješteniji** od algoritma A_2^*

- Obavješteniji algoritam općenito pretražuje manji prostor stanja od manje obaviještenog algoritma
- Npr. za slagalicu vrijedi: $h^*(s) \geq h_2(s) \geq h_1(s)$, tj. L1-heuristika daje obavješteniji algoritam od heuristike koja broji razmještene pločice
- Pritom u obzir treba uzeti i složenost izračunavanja heuristike!

Dobra heuristika

- Dobra heuristika je:
  1. optimistična
  2. što obavještenija
  3. jednostavno izračunljiva

Pesimistične heuristike?

- Ako ne trebamo baš optimalno rješenje, nego neko rješenje koje je dovoljno dobro, možemo koristiti heuristiku koja nije optimistična (heuristiku koja precjenjuje)
- Uporaba takve heuristike dodatno će smanjiti broj generiranih čvorova
- Radimo kompromis između kvalitete rješenja i složenosti pretraživanja

- Kako oblikovati dobru heuristiku za neki zadani problem?

Oblikovanje heuristike

1. Relaksakcija problema
   - stvarna cijena **relaksiranog problema** je optimistična heuristika izvornog problema
   - npr. relaksacija slagalice 3×3:
     pločice se mogu pomicati bilo kuda $\Rightarrow$ L1-udaljenost
   - dobivamo optimističnu i ujedno konzistentnu heuristiku (zašto?)
2. Kombiniranje optimističnih heuristika
   - Ako su $h_1, h_2, \ldots, h_n$ optimistične, možemo ih kombinirati u dominantnu heuristiku koja će također biti optimistična:

$$h(s) = \max\big(h_1(s), h_2(s), \ldots, h_n(s)\big)$$

3. Cijena rješavanja podproblema
   - baza uzoraka koja sadržava optimalne cijene pojedinih podproblema
4. Učenje heuristike
   - primjena metoda strojnog učenja. Npr. učimo koeficijente w_1 i w_2:
     $h(s) = w_1 x_1(s) + w_2 x_2(s)$, gdje su x_1 i x_2 značajke stanja

Laboratorijski zadatak: Pretraživanje visinske mape

Napišite program koji će korištenjem algoritama A* izračunati najjeftiniji put između dvije točke na zadanoj visinskoj mapi.

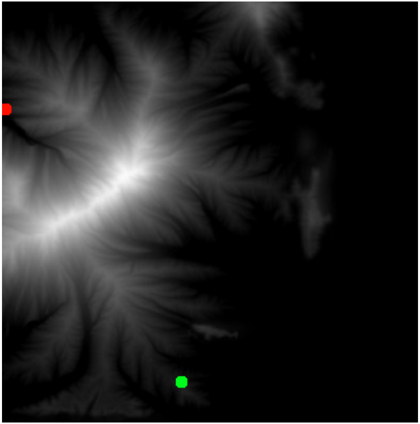
Neka $\Delta v = v(x_2, y_2) - v(x_1, y_1)$. Dozvoljen je pomak s polja (x_1, y_1) na polje (x_2, y_2) ukoliko je $|x_1 - x_2| \leq 1, |y_1 - y_2| \leq 1$ i $\Delta v \leq m$.

Cijena puta od polja (x_1, y_1) do polja (x_2, y_2) je

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} + \big(\frac{1}{2}\mathrm{sgn}(\Delta v) + 1\big) \cdot |\Delta v|$$

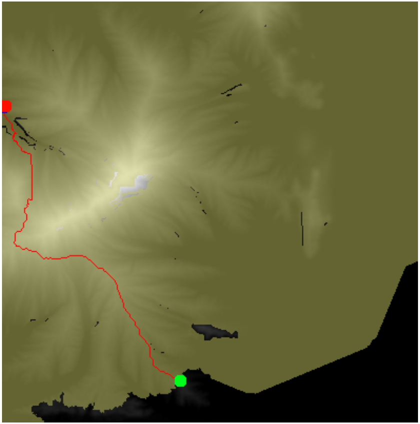
Mapu i parametre program treba učitati iz zadane tekstne datoteke. Program na generiranoj slici treba prikazati visinsku mapu, sve zatvorene čvorove, sve otvorene čvorove, pronađeni put, duljinu pronađenog puta i broj koraka algoritma. Potrebno je osmisliti barem tri različite heuristike i isprobati rad algoritama s tim heuristikama.

Pretraživanje visinske mape (1)



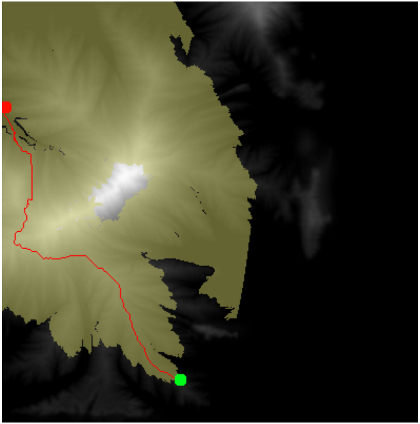
Visinska mapa
(svjetliji elementi su na većoj visini)
crveno: početno stanje, zeleno: ciljno stanje

Pretraživanje visinske mape (2)



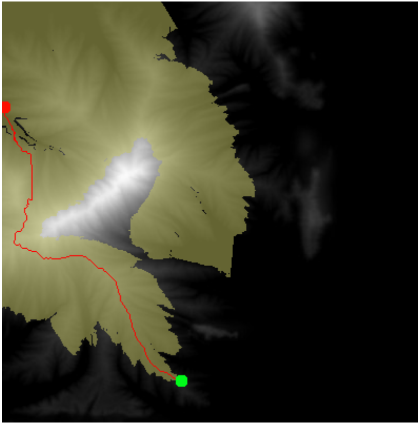
Pretraživanje s jednolikom cijenom
(crveno: pronađen put, žuto: posjećena stanja)
broj zatvorenih čvorova: 140580, dužina puta: 740,58

Pretraživanje visinske mape (3)



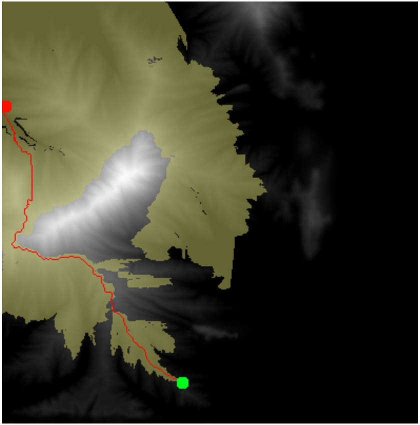
Algoritam A*
(heuristika: zračna udaljenost)
broj zatvorenih čvorova: 64507, dužina puta: 740,58

Pretraživanje visinske mape (4)



Algoritam A*
(heuristika: zračna udaljenost + $\frac{1}{2}\Delta v$)
broj zatvorenih čvorova: 56403, dužina puta: 740,58

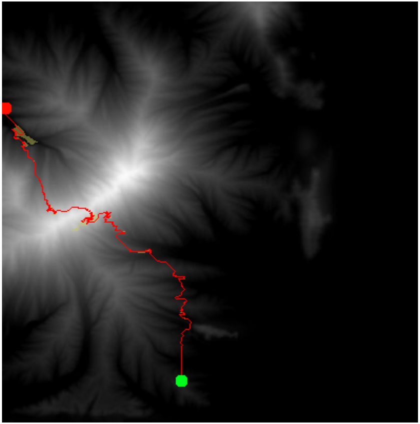
Pretraživanje visinske mape (5)



Algoritam A*
(heuristika: zračna udaljenost + Δv)
broj zatvorenih čvorova: 52099, dužina puta: 755,16

Ova heuristika nije optimistična!

Pretraživanje visinske mape (6)



Pohlepno pretraživanje “najbolji prvi”
(uz uporabu liste posjećenih stanja)
broj zatvorenih čvorova: 822, dužina puta: 1428,54

Algoritam uspona na vrh

- Kao pohlepno pretraživanje "najbolji prvi", s tom razlikom da se generirani čvorovi uopće ne pohranjuju u memoriji

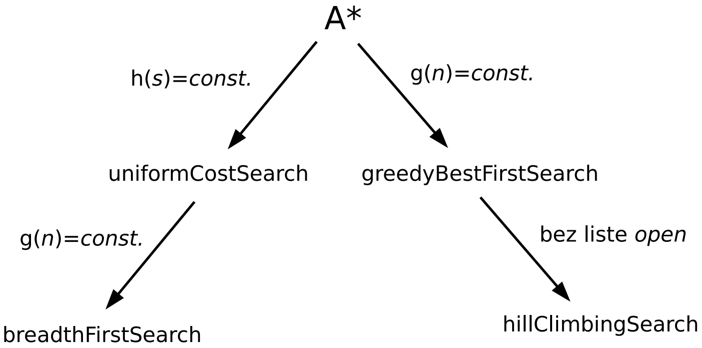
Algoritam uspona na vrh

function $\mathrm{hillClimbingSearch}(s_0, \mathrm{succ}, h)$
  $n \leftarrow \mathrm{initial}(s_0)$
  loop do
    $M \leftarrow \mathrm{expand}(n, \mathrm{succ})$
    if $M = \varnothing$ **then return** n
    $m \leftarrow \mathrm{minimumBy}(f, M)$
    if $f(n) < f(m)$ **then return** n
    $n \leftarrow m$
where $f(n) = h(\mathrm{state}(n))$

Algoritam uspona na vrh – svojstva

- **Nije potpun** i **nije optimalan**
- Lako zaglavljuje u tzv. **lokalnim optimumima**
- Učinkovitost uvelike ovisi o izboru heurističke funkcije
- Uobičajeno se koristi tehnika **slučajnog ponovnog starta** (engl. *random-restart*)
- **Vremenska složenost:** $\mathcal{O}(m)$
- **Prostorna složenost:** $\mathcal{O}(1)$

Odnosi između algoritama



- A^* dominira nad algoritmima uniformCostSearch i breadthFirstSearch

Sažetak

- Heuristička funkcija **usmjerava pretraživanje** i tako ga ubrzava
- Heuristička funkcija definira **procjenu udaljenosti** trenutnog stanja od ciljnog stanja. Što je ona manja, to smo bliže cilju
- Heuristika treba biti **optimistična**, može biti **konzistentna**, a poželjno je da bude što **obavještenija** jer to ubrzava pretraživanje
- Algoritami "najbolji prvi" i "uspon na vrh" su **pohlepni** i zato nisu optimalni
- Algoritam A^* je **potpun i optimalan**

Sljedeća tema: Igranje igara