Pretraga. A*

Danijela Simić

March 8, 2016

Literatura

- Danijela Simić – *www.matf.bg.ac.rs/∼danijela*
  danijela@matf.bg.ac.rs

Literatura

- Danijela Simić – *www.matf.bg.ac.rs/∼danijela*
  danijela@matf.bg.ac.rs
- Predrag Janicic i Mladen Nikolić – *Veštačka inteligencija*

Literatura

- Danijela Simić – *www.matf.bg.ac.rs/∼danijela* *danijela@matf.bg.ac.rs*
- Predrag Janicic i Mladen Nikolić – *Veštačka inteligencija*
- Stuart Russel, Peter Norvig – *Artificial Intelligence – A Modern Approach*

Literatura

- Danijela Simić – *www.matf.bg.ac.rs/∼danijela*
  danijela@matf.bg.ac.rs
- Predrag Janicic i Mladen Nikolić – *Veštačka inteligencija*
- Stuart Russel, Peter Norvig – *Artificial Intelligence – A Modern Approach*
- 3 testa

Pretraga

- *Nalaženje niza akcija kojima se ostvaruje cilj kada to ne može biti ostvareno pojedinačnim akcijama.*

Pretraga

- *Nalaženje niza akcija kojima se ostvaruje cilj kada to ne može biti ostvareno pojedinačnim akcijama.*
- Problem se često predstavlja *grafom*, dok se rešavanje problema često predstavlja *stablom*.

Pretraga

- *Nalaženje niza akcija kojima se ostvaruje cilj kada to ne može biti ostvareno pojedinačnim akcijama.*
- Problem se često predstavlja *grafom*, dok se rešavanje problema često predstavlja *stablom*.
- Primene: prilikom planiranja obilaska pri putovanju, dizajniranje čipova, rutiranje u računarskim mrežama, u industriji igara, ...

- Da bi nešto bilo problem pretrage, mora imati određenu strukturu:

- Da bi nešto bilo problem pretrage, mora imati određenu strukturu:
  1. **skup mogućih stanja**

- Da bi nešto bilo problem pretrage, mora imati određenu strukturu:
  1. **skup mogućih stanja**
  2. **polazno stanje**

- Da bi nešto bilo problem pretrage, mora imati određenu strukturu:
  1. **skup mogućih stanja**
  2. **polazno stanje**
  3. **skup mogućih akcija**
     $akcija(stanje_1) = \{a_1, a_2, a_3, \ldots\}$

- Da bi nešto bilo problem pretrage, mora imati određenu strukturu:
  1. **skup mogućih stanja**
  2. **polazno stanje**
  3. **skup mogućih akcija**
     $akcija(stanje_1) = \{a_1, a_2, a_3, \ldots\}$
  4. **funkcija prelaska**
     $prelazak(stanje_1, a_k) = stanje_p$

- Da bi nešto bilo problem pretrage, mora imati određenu strukturu:
  1. **skup mogućih stanja**
  2. **polazno stanje**
  3. **skup mogućih akcija**
     $akcija(stanje_1) = \{a_1, a_2, a_3, \ldots\}$
  4. **funkcija prelaska**
     $prelazak(stanje_1, a_k) = stanje_p$
  5. **test cilja**

- Da bi nešto bilo problem pretrage, mora imati određenu strukturu:
  1. **skup mogućih stanja**
  2. **polazno stanje**
  3. **skup mogućih akcija**
     $akcija(stanje_1) = \{a_1, a_2, a_3, \ldots\}$
  4. **funkcija prelaska**
     $prelazak(stanje_1, a_k) = stanje_p$
  5. **test cilja**
  6. **cena akcije**

Bitne osobine algoritama pretrage

- potpunost – pronalazimo rešenje!

Pretraga može biti *neiformisana* i *informisana* pretraga.

Bitne osobine algoritama pretrage

- potpunost – pronalazimo rešenje!
- optimalnost – pronalazi najjeftinije rešenje!

Pretraga može biti *neiformisana* i *informisana* pretraga.

Bitne osobine algoritama pretrage

- potpunost – pronalazimo rešenje!
- optimalnost – pronalazi najjeftinije rešenje!
- vremenska složenost – posmatramo najgori i prosečni slučaj

Pretraga može biti *neiformisana* i *informisana* pretraga.

Bitne osobine algoritama pretrage

- potpunost – pronalazimo rešenje!
- optimalnost – pronalazi najjeftinije rešenje!
- vremenska složenost – posmatramo najgori i prosečni slučaj
- prostorna složenost – koliko memorije je potrebno; posmatramo najgori i prosečni slučaj

Pretraga može biti *neiformisana* i *informisana* pretraga.

DFS – pretraga u dubinu

Ulaz: Graf G, polazni čvor, ciljni čvor

Izlaz: Put od polaznog do ciljnog čvora u grafu G (ako takav put postoji)

1. Inicijalno, stek *put* i skup *posećenih čvorova* sadrže samo polazni čvor.
2. Izvršavaj dok stek *put* nije prazan:
    - Uzmi čvor sa vrha steka *put*.
    - Ako je n ciljni čvor, izvesti u uspehu i vrati put konstruisan na osniovu sadržaja steka *put*.
    - Ako n nema potomaka koji nisu posećeni, izbaci n sa steka *put*.
    - U suprotnom izaberi prvog takvog potomka m i dodaj ga na vrh steka *put* i u skup posećenih čvorova.
3. Izvestiti da traženi put ne postoji.

Vremenska složenost – proporcionalna zbiru čvorova i grana grafa koji se pretražuje.

Prostorna složenost – proporcionalna broju čvorova.

BFS – pretraga u širinu

Ulaz: Graf G, polazni čvor, ciljni čvor
Izlaz: Put od polaznog do ciljnog čvora u grafu G (ako takav put postoji)

1. Red S sadrži samo polazni čvor.
2. Izvršavaj dok red S nije prazan:
   - Uzmi čvor n sa početka reda S, dodaj ga u listu posećenih čvorova i obriši ga iz reda.
   - Ako je n ciljni čvor, izvestiti o uspehu i vratiti put od polaznog do ciljnog čvora (idući unazad od ciljnog čvora)
   - Za svaki od potomaka m čvora n za koji nije definisan roditelj, zapatiti n kao roditelja i dodaj ga na kraj reda S.
3. Izvestiti da traženi put ne postoji.

Vremenska složenost – proporcionalna zbiru čvorova i grana grafa koji se pretražuje.
Prostorna složenost – proporcionalna broju čvorova.

Dejikstring algoritam

Ulaz: Graf G, polazni čvor, ciljni čvor
Izlaz: Najkraći put od polaznog do ciljnog čvora u grafu G (ako takav put postoji)

1. Skup Q inicijalno sadrži sve čvorove grafa
2. Izvršavaj sve dok je skup Q neprazan:
   - Izaberi iz Q čvor n sa najmanjim ustanovljenim rastojanjem od polaznog čvora i obriši ga iz Q
   - Ako je n ciljni čvor, konstruiši put od polaznog do ciljnog čvora (idući unazad od ciljnog čvora) i izvesti o uspehu
   - Za svaki čvor m iz Q koji je direktno dostupan iz n, proveri da li je ustanovljeno rastojanje od polaznog čvora do m veće od rastojanja od polaznog čvora do m preko čvora n i ako jeste promeni informaciju o roditelju čvora m na čvor n i upamti novo rastojanje.
3. Izvesti da traženi put ne postoji (Q je prazan skup i uspeh nije prijavljen)

Pohlepna pretraga

- U svakom korako bira se onaj čvor koji ima najbolju ocenu.

Pohlepna pretraga

- U svakom korako bira se onaj čvor koji ima najbolju ocenu.
- Biraju se *lokalno* optimalne akcije

Pohlepna pretraga

- U svakom korako bira se onaj čvor koji ima najbolju ocenu.
- Biraju se *lokalno* optimalne akcije
- Ne može da proceni dugoročni kvalitet izabranih akcija

Prvo najbolji

- U svakom korako bira onaj čvor iz liste otvorenih koji ima najbolju ocenu

Prvo najbolji

- U svakom korako bira onaj čvor iz liste otvorenih koji ima najbolju ocenu
- Ima svojstvo potpunosti

Prvo najbolji

- U svakom korako bira onaj čvor iz liste otvorenih koji ima najbolju ocenu
- Ima svojstvo potpunosti
- ako nam je ocena *dubina* – pretraga u sirinu

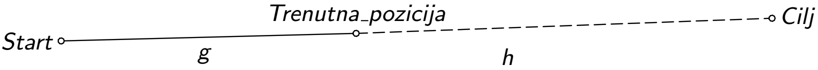
Prvo najbolji

- U svakom korako bira onaj čvor iz liste otvorenih koji ima najbolju ocenu
- Ima svojstvo potpunosti
- ako nam je ocena *dubina* – pretraga u sirinu
- ako nam je ocena *cena predjenog puta* – Dejikstrin algoritam

A^*

- Posmatra se:

$$f(n) = g(n) + h(n)$$

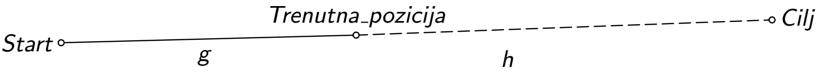


A^*

- Posmatra se:

$$f(n) = g(n) + h(n)$$

- $g(n)$ – cena *od polaznog* čvora do čvora n

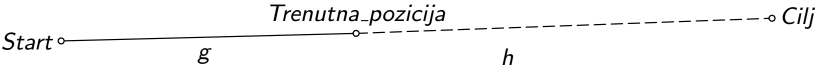


$A*$

- Posmatra se:

$$f(n) = g(n) + h(n)$$

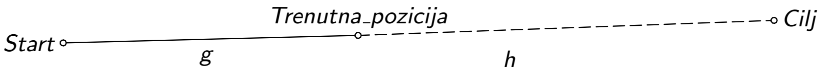
- $g(n)$ – cena *od polaznog* čvora do čvora n
- $h(n)$ – *PROCENA* koliko ima do cilja;
  Nema nikakve veze sa dužinom pređenog puta, bitan je cilj
  To je **heuristika** i zadaje se unapred, za svaki čvor se procenjuje koliko je udaljen od cilja



$A*$

- Da bi rešenje bilo **optimalno** mora da važi:

$$0 \leq h(n) < h*(n)$$

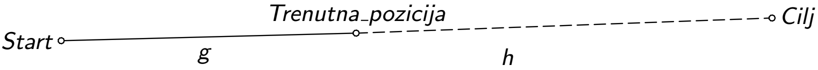


$A*$

- Da bi rešenje bilo **optimalno** mora da važi:

$$0 \leq h(n) < h*(n)$$

- *konzistentna*: $c(m,n) + h(m) \geq h(n)$

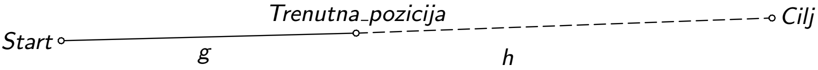


$A*$

- Da bi rešenje bilo **optimalno** mora da važi:

$$0 \leq h(n) < h*(n)$$

- *konzistentna*: $c(m, n) + h(m) \geq h(n)$
- minimizovanjem g postižemo da *put bude kratak*

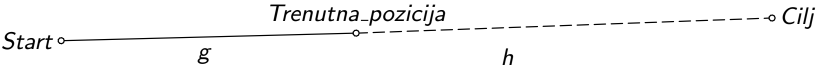


$A*$

- Da bi rešenje bilo **optimalno** mora da važi:

$$0 \leq h(n) < h*(n)$$

- *konzistentna*: $c(m, n) + h(m) \geq h(n)$
- minimizovanjem g postižemo da *put bude kratak*
- minimizovanjem h postižemo da smo *usresređeni na cilj* (ne lutamo ka nečemu što nije cilj)

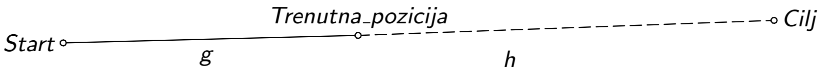


$A*$

- Da bi rešenje bilo **optimalno** mora da važi:

$$0 \leq h(n) < h*(n)$$

- *konzistentna*: $c(m, n) + h(m) \geq h(n)$
- minimizovanjem g postižemo da *put bude kratak*
- minimizovanjem h postižemo da smo *usresređeni na cilj* (ne lutamo ka nečemu što nije cilj)
- Primer: kada je h = 0 (prvo najbolji)

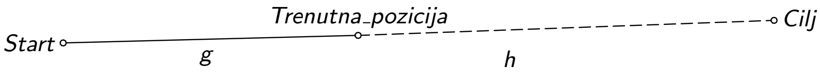


$A*$

- Da bi rešenje bilo **optimalno** mora da važi:

$$0 \leq h(n) < h*(n)$$

- *konzistentna*: $c(m,n) + h(m) \geq h(n)$
- minimizovanjem g postižemo da *put bude kratak*
- minimizovanjem h postižemo da smo *usresređeni na cilj* (ne lutamo ka nečemu što nije cilj)
- Primer: kada je h = 0 (prvo najbolji)
- Primer: kada je g = 0 (pohlepna pretraga)



$A*$

- Da bi rešenje bilo **optimalno** mora da važi:

$$0 \leq h(n) < h*(n)$$

- *konzistentna*: $c(m,n) + h(m) \geq h(n)$
- minimizovanjem g postižemo da *put bude kratak*
- minimizovanjem h postižemo da smo *usresređeni na cilj* (ne lutamo ka nečemu što nije cilj)
- Primer: kada je h = 0 (prvo najbolji)
- Primer: kada je g = 0 (pohlepna pretraga)
- Primer: kada imamo prepreku