Hľadanie najskôr najlepší

function HĽADANIE-NAJSKÔR-NAJLEPŠÍ(*problém*, VYHODNOCOVACIA-FUNKCIA) **returns** *riešenie* alebo neúspech

ZARAĎOVACIA-FUNKCIA ← funkcia, ktorá zaradí uzly (1. arg) podľa hodnôt, ktoré vráti VYHODNOCOVACIA-FUNKCIA po aplikácii na ne do frontu (2. arg) tak, aby ostalo zachované usporiadanie

return VŠEOBECNÉ-HĽADANIE(*problém*, ZARAĎOVACIA-FUNKCIA)

- vyhodnocovacia funkcia vyjadruje iba odhad, takže na rozvitie sa vyberá uzol, ktorý sa iba zdá byť najlepším

1

hľadanie najlepší najskôr

Treba pripomenúť, že poradie stavov v

OKRAJi definuje stratégia hľadania

```
hľadanie-najskôr-najlepší(začiatočný-s)
   OKRAJ <- vytvor-front(začiatočný-s,
h(začiatočný-s))
   while (not(prázdny(OKRAJ)))
      uzol ← vyber(OKRAJ)
      s ← stav(uzol)
      if cieľový-test(s) then return s alebo cesta do
      uzol
      for each stav s´ v nasledovníky(s)
         OKRAJ <- zaraď-do-frontu(s´, h(s´))
   return failure
```

2

hľadanie najlepší najskôr

- Využíva opis stavov na odhadnutie ako „dobrý“ bude vyhľadávací uzol
- vyhodnocovacia funkcia f zobrazuje každý uzol N stromu hľadania na reálne číslo f(N) ≥ 0
  [Zvyčajne, f(N) je očakávaná cena, takže čím menšie f(N), tým je uzol N sľubnejší]
- hľadanie najlepší najskôr usporadúva OKRAJ podľa stúpajúcej hodnoty f [poradie uzlov s rovnakou hodnotou f je ľubovoľné]

3

hľadanie najlepší najskôr

- Využíva popis stavov na odhadnutie ako „dobrý“ bude vyhľadávací uzol
- vyhodnocovacia funkcia f zobrazuje každý uzol N stromu hľadania na reálne číslo f(N) ≥ 0
  [Zvyčajne, f(N) je očakávaná cena, takže čím menšie f(N), tým je uzol N sľubnejší]
- hľadanie najlepší najskôr usporadúva OKRAJ podľa stúpajúcej hodnoty f [poradie uzlov s rovnakou hodnotou f je ľubovoľné]

“Najlepší” nepredstavuje kvalitu generovanej cesty.
Vo všeobecnosti best-first prehľadávanie negeneruje optimálne cesty.

4

Ako vyrobiť f?

- f(N) odhaduje:
  - Buď cenu cesty riešenia cez N
    Potom f(N) = g(N) + h(N), kde
    - g(N) je cena cesty z počiatočného uzla do N
    - h(N) je odhad ceny cesty z N do cieľového bodu
  - Alebo cenu cesty z N do cieľového bodu
    - Potom f(N) = h(N) → lačné hľadanie
- Nie sú tu avšak žiadne limity na F. Ľubovoľná funkcia vášho výberu je akceptovateľná. Pomôže to však vyhľadávaniu?

5

Ako vyrobiť f?

- f(N) odhaduje:
  - Buď cenu riešenej cesty cez N
    Potom f(N) = g(N) + h(N), kde
    - g(N) je cena cesty z počiatočného uzla do N
    - h(N) je odhad ceny cesty z N do cieľového bodu
  - Alebo cena cesty z N do cieľového bodu
    - Potom f(N) = h(N) → lačné (nenásytné) hľadanie

Heuristická funkcia

- Nie sú tu avšak žiadne limity na F. Ľubovoľná funkcia vášho výberu je akceptovateľná. Pomôže to však vyhľadávaniu?

6

Informované hľadanie

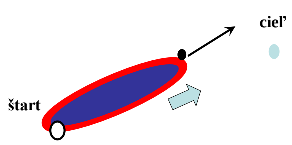
- Vyhodnocovacia funkcia musí zahŕňať aj nejaký odhad ceny cesty z daného uzla do najbližšieho cieľového uzla.
- Dva prístupy:
  - rozvitie uzla, ktorý sa zdá byť podľa vyhodnocovacej funkcie najbližšie k cieľu
  - rozvitie uzla na najlacnejšej ceste riešenia

Lačné hľadanie

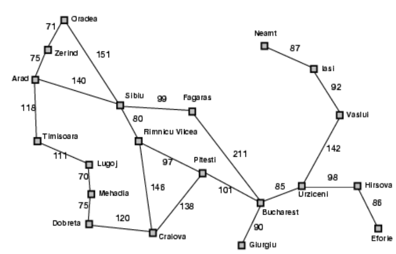
- minimalizácia odhadovanej ceny dosiahnutia cieľa
- h(u) = odhad ceny najlacnejšej cesty z uzla u do cieľového uzla
- h(c) = 0, ak c je cieľový uzol
- Hodnotí sa cena cesty do cieľa – čím lacnejšie, tým lepšie
- Pripomína hľadanie do hĺbky
- Nie je úplné, neoptimalizuje riešenie

function LAČNÉ-HĽADANIE(*problém*, *h*) **returns** *riešenie* alebo neúspech
return HĽADANIE-NAJSKÔR-NAJLEPŠÍ(*problém*, *h*)

Lačné hľadanie



príklad: Rumunsko, cestná sieť

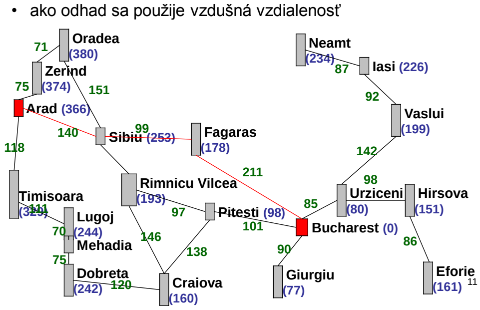


Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Heuristika: príklad Rumunsko

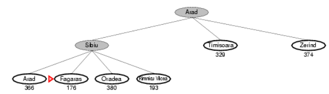
- cestovanie: h(n) = odhad-vzdialenosti(n, cieľ)
- ako odhad sa použije vzdušná vzdialenosť



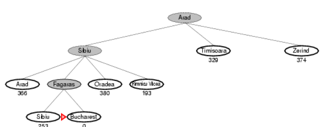
lačné hľadanie najlepší najskôr

Iačné hľadanie najlepší najskôr

Iačné hľadanie najlepší najskôr



Iačné hľadanie najlepší najskôr



Heuristická funkcia

- Heuristická funkcia h(N) ≥ 0 odhaduje cenu cesty zo STAVu(N) do cieľového stavu
- Jej hodnota je nezávislá na **aktuálnom prehľadávanom strome.** Závisí iba na STAVe(N) a cieľovom teste GOAL?
- Príklad:

5		8
4	2	1
7	3	6

STAV(N)

1	2	3
4	5	6
7	8	

Cieľový stav

$h_1(N)$ = počet zle umiestnených očíslovaných doštičiek = 6

[Prečo je to odhad vzdialenosti ku cieľu?]

Iné príklady

5		8
4	2	1
7	3	6

STAV(N)

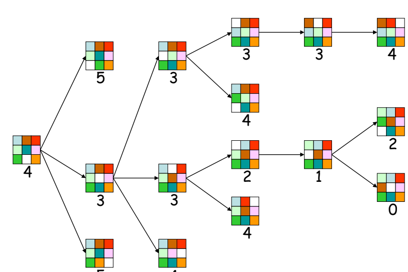
1	2	3
4	5	6
7	8	

Cieľový stav

- $h_1(N)$ = počet zle umiestnených očíslovaných doštičiek = 6
- $h_2(N)$ = súčet (manhattanských) vzdialeností každej očíslovanej doštičky do jej cieľovej pozície
  = 2 + 3 + 0 + 1 + 3 + 0 + 3 + 1 = 13
- $h_3(N)$ = súčet inverzií permutácií
  $= n_5 + n_8 + n_4 + n_2 + n_1 + n_7 + n_3 + n_6$
  = 4 + 6 + 3 + 1 + 0 + 2 + 0 + 0
  = 16
  - Ak doštička obsahujúca číslo i sa nachádza pred n doštičkami obsahujúcimi čísla menšie než i, tak dochádza k inverzii rádu n = označíme n_i.

8-hlavolam

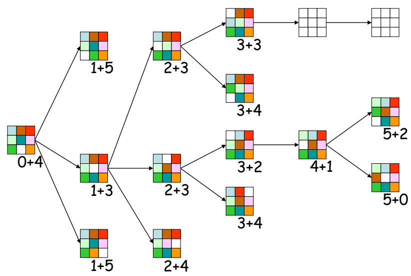
f(N) = h(N) = počet zle umiestnených ofarbených doštičiek



Biela je prázdna

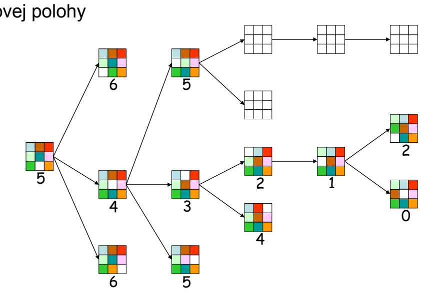
8-hlavolam

f(N) = g(N) + h(N)
kde h(N) = počet zle umiestnených ofarbených doštičiek

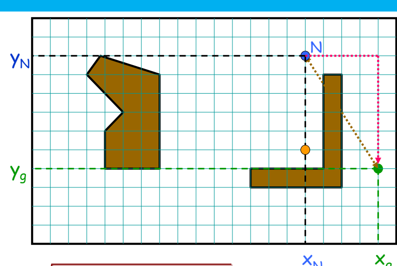


8-hlavolam

f(N) = h(N) = Σ vzdialeností ofarbených doštičiek do ich cieľovej polohy



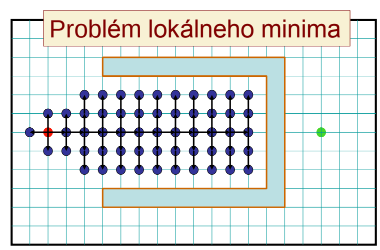
Navigácia robota



$h_1(N) = \sqrt{(x_N-x_g)^2+(y_N-y_g)^2}$ (L_1 alebo euklidovská vzd.)

$h_2(N) = |x_N-x_g| + |y_N-y_g|$ (L_2 alebo manhattanská vzd.)

Best-First $\not\rightarrow$ Efektívnosť



f(N) = h(N) = priama vzdialenosť do cieľa

Môžeme niečo dokázať?

- Ak stavový priestor je nekonečný, vo všeobecnosti hľadanie nie je úplné
- Ak stavový priestor je konečný a nezrušíme uzly, ktoré znovu navštívili stavy, vo všeobecnosti hľadanie nie je úplné
- Ak stavový priestor je konečný a zrušili sme uzly, ktoré znovu navštevujú stavy, hľadanie je úplné, ale vo všeobecnosti nie je optimálne

Prípustná heuristika

- Nech h*(N) je cena optimálnej cesty z N do cieľového uzla
- Heuristická funkcia h(N) je prípustná ak:

  $0 \le h(N) \le h^*(N)$

- Prípustná heuristická funkcia je vždy optimistická!

Prípustná heuristika

- Nech h*(N) je cena optimálnej cesty z N do cieľového uzla
- Heuristická funkcia h(N) je prípustná ak:

$$0 \le h(N) \le h^*(N)$$

- Prípustná heuristická funkcia je vždy optimistická!

G je cieľový uzol ➔ h(G) = 0

Heuristiky pre 8-hlavolam

5		8
4	2	1
7	3	6

STAV(N)

1	2	3
4	5	6
7	8	

cieľový stav

- $h_1(N)$ = počet zle uložených doštičiek = 6
je ???

Heuristiky pre 8-hlavolam

5		8
4	2	1
7	3	6

STAV(N)

1	2	3
4	5	6
7	8	

cieľový stav

- $h_1(N)$ = počet zle uložených doštičiek =
1=1+2=1+3=1+4=0+5=1+6=1+7=0+8=1=6
je prípustná
- $h_2(N)$ = súčet (manhattanských) vzdialeností každej doštičky do jej cieľa
= 1=3+2=1+3=3+4=0+5=2+6=1+7=0 = 10
je ???

Heuristiky pre 8-hlavolam

5		8
4	2	1
7	3	6

STAV(N)

1	2	3
4	5	6
7	8	

cieľový stav

- $h_1(N)$ = počet zle uložených doštičiek = 6
je prípustná
- $h_2(N)$ = súčet (manhattanských) vzdialeností každej doštičky do jej cieľa
= 1=3+2=1+3=3+4=0+5=2+6=1+7=0 = 10
je prípustná
- $h_3(N)$ = súčet inverzií permutácií
= 4 + 6 + 3 + 1 + 0 + 2 + 0 + 0 = 16
je ???

prípustnosť heuristík pre 8-hlavolam

5		8
4	2	1
7	3	6

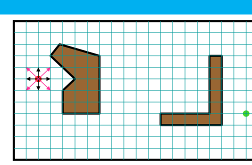
STAV(N)

1	2	3
4	5	6
7	8	

cieľový stav

- $h_1(N)$ = počet zle uložených doštičiek = 6
je prípustná
- $h_2(N)$ = súčet (manhattanských) vzdialeností každej doštičky do jej cieľa
= 1=3+2=1+3=3+4=0+5=2+6=1+7=0 = 10
je prípustná
- $h_3(N)$ = súčet inverzií permutácií
= 4 + 6 + 3 + 1 + 0 + 2 + 0 + 0 = 16
nie je prípustná

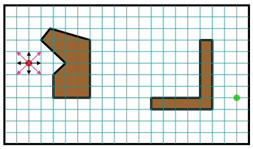
Navigácia robota



Cena horizontálneho alebo vertikálneho kroku = 1
Cena diagonálneho kroku = $\sqrt{2}$

$h_1(N) = \sqrt{(x_N-x_g)^2+(y_N-y_g)^2}$ Je prípustná

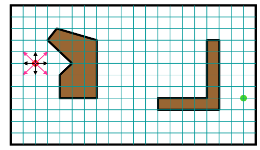
Navigácia robota



Cena horizontálneho alebo vertikálneho kroku = 1
Cena diagonálneho kroku = $\sqrt{2}$

$h_2(N) = |x_N - x_g| + |y_N - y_g|$ je ???

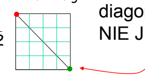
Navigácia robota



Cena horizontálneho alebo vertikálneho kroku = 1
Cena diagonálneho kroku = $\sqrt{2}$

$h_2(N) = |x_N - x_g| + |y_N - y_g|$ Je prípustná ak pohyb po diagonálach je zakázaný, inak NIE JE prípustná

$h^*(I) = 4\sqrt{2}$
$h_2(I) = 8$



Ako vytvoriť prípustnú h?

- Prípustná heuristika môže obyčajne chápaná ako cena optimálneho riešenia voľnejšieho (všeobecnejšieho) problému (takého, ktorý vznikne odstránením niektorých ohraničení v pôvodnom probléme)
- Príklad navigácie robota:
  - Manhattanská vzdialenosť zodpovedá tomu, že sa odstránili prekážky *(budovy atď., ulice a triedy tvoria úplnú mriežku)*
  - Euklidovská vzdialenosť zodpovedá tomu, že sa odstránili aj prekážky aj ohraničenie, že robot sa môže pohybovať iba po mriežke

príklad uvoľnenia (relaxovania) problému

- cena optimálneho riešenia voľnejšieho problému je prípustná heuristika pre pôvodný problém
- ak sa pravidlá 8-hlavolamu zvoľnia tak, že doštička sa môže presunúť hocikde, tak $h_1(n)$ určuje najkratšie riešenie
- ak sa pravidlá 8-hlavolamu zvoľnia tak, že doštička sa môže presunúť na hociktoré susedné políčko, tak $h_2(n)$ určuje najkratšie riešenie

A* hľadanie (najpopulárnejší algoritmus umelej inteligencie)

***f*(*u*) = *g*(*u*) + *h*(*u*)**

g(*u*) = *cena cesty do uzla u*
h(*u*) = *odhad ceny cesty z uzla u k riešeniu*
f(*u*) = odhad ceny najlacnejšej cesty riešenia vedúcej cez uzol *u*

- Pre všetky hrany: $c(u,u') \geq \varepsilon > 0$
- Použije sa algoritmus najlepší najskôr

➔ Hľadanie najskôr najlepší s vyhodnocovacou funkciou *f* a prípustnou heuristikou *h* sa nazýva **A* hľadanie**.

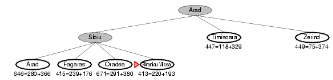
function A*-HĽADANIE(*problém, g, h*) **returns** *riešenie* alebo neúspech
return HĽADANIE-NAJSKÔR-NAJLEPŠÍ(*problém, g+h*)

príklad A* hľadania

príklad A* hľadania

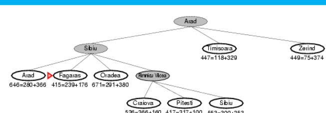
37

príklad A* hľadania



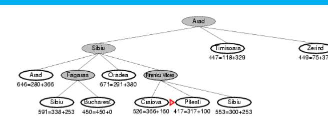
38

príklad A* hľadania



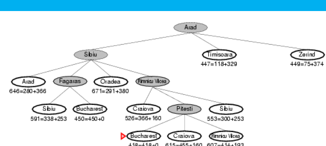
39

príklad A* hľadania



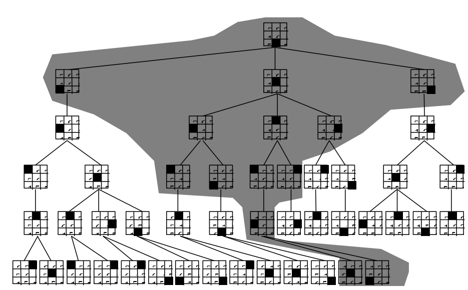
40

príklad A* hľadania



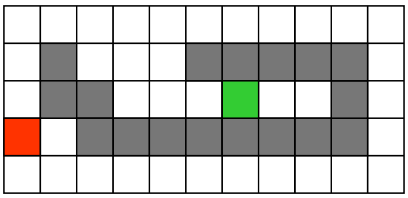
41

Porovnanie A* hľadania s hľadaním do šírky



42

Navigácia robota



Navigácia robota

f(N) = h(N), kde h(N) = manhattanská vzdialenosť do cieľa (nie A*)

8	7	6	5	4	3	2	3	4	5	6
7		5	4	3						5
6			3	2	1	0	1	2		4
7	6									5
8	7	6	5	4	3	2	3	4	5	6

Navigácia robota

f(N) = h(N), kde h(N) = manhattanská vzdialenosť do cieľa (nie A*) – navštívené stavy

8	7	6	5	4	3	2	3	4	5	6
7		5	4	3						5
6			3	2	1	0	1	2		4
7	6									5
8	7	6	5	4	3	2	3	4	5	6

Navigácia robota

f(N) = g(N)+h(N), kde h(N) = manhattanská vzdialenosť do cieľa (A*) – navštívené stavy

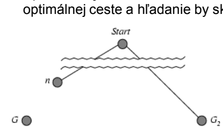
8+3	7+4	6+3	5+6	4+7	3+8	2+9	3+10	4	5	6
7+2		5+6	4+7	3+8						5
6+1			3	2+9	1+10	0+11	1	2		4
7+0	6+1									5
8+1	7+2	6+3	5+4	4+5	3+6	2+7	3+8	4	5	6

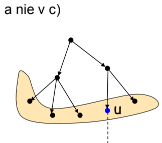
tvrdenie o A*

A* je úplný a prípustný

dôkaz prípustnosti (optimálnosti) A^*

- nech C* je cena optimálnej cesty.
- Predpokladajme: hľadanie skončí v cieľovom stave c (h(c)=0), pre ktorý f(c) = g(c) > C* (to je ale suboptimálne riešenie! – vedieme dôkaz sporom)
- uvažujme uzol *u* na okraji (čiže nie je cieľový), ležiaci na optimálnej ceste (existuje určite? áno, inak by sa už dosiahol cieľ na optimálnej ceste a hľadanie by skončilo tam a nie v c)

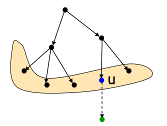


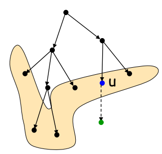


- legenda: uzol u = n, cieľ c = G_2

dôkaz prípustnosti (optimálnosti) A*

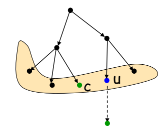
- uvažujme uzol *u* na okraji (čiže nie je cieľový), ležiaci na optimálnej ceste (existuje určite? áno, inak by sa už dosiahol cieľ na optimálnej ceste a hľadanie by skončilo tam a nie v c)
- každé rozvinutie nejakého uzla zvyšuje dĺžku nejakej cesty, takže skôr či neskôr musí prísť na rozvíjanie uzla u. Iba ak by sa už skôr našlo riešenie na inej ceste.



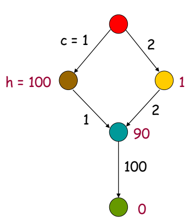


dôkaz prípustnosti (optimálnosti) A*

- C* ≥ f(u) lebo h je prípustná
- f(u) ≥ f(c) lebo u sa nevybralo dosiaľ na rozvitie
- C* ≥ f(c) priamo z predchádzajúcich
- C* ≥ g(c) lebo h(c) = 0 a teda f(c) = g(c)
- C* ≥ g(c) je spor s predpokladom g(c) > C* .
- nie je pravda, že A* vyberie neoptimálny cieľ. Naopak, A* vyberie optimálny cieľ, t.j. cieľ na najlepšej ceste riešenia.

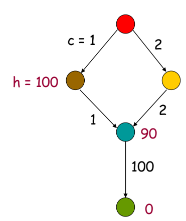


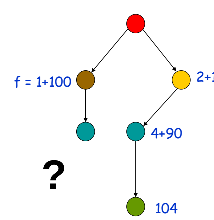
čo robiť s opätovne navštívenými stavmi?



heuristika je zjavne prípustná

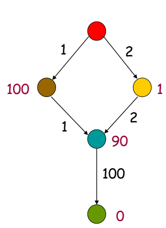
čo robiť so opätovne navštívenými stavmi?

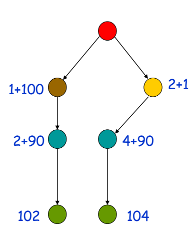




ak zahodíme tento nový uzol, tak algoritmus rozvinie v ďalšom kroku cieľový uzol a vráti neoptimálne riešenie

čo robiť s opätovne navštívenými stavmi?

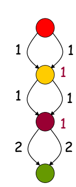


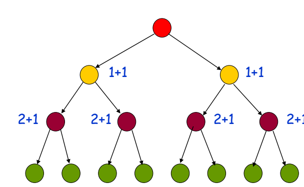


ak namiesto toho nezahadzujeme uzly s opätovne navštívenými stavmi, hľadanie skončí s optimálnym riešením.

ale ...

ak nezahadzujeme uzly so znovunavštívenými stavmi, tak veľkosť stromu hľadania môže byť exponenciálna v závislosti od počtu navštívených stavov





odhadzovanie uzlov, reprezentujúcich znovunavštívené stavy

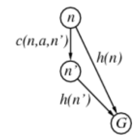
- odhadzovanie uzlov, reprezentujúcich znovunavštívené stavy, neškodí, ak cena novej cesty do toho uzla nie je menšia než cena doterajšej cesty
  - čiže napr možno odhodiť uzol, ktorý znovunavštevuje stav, navštívený niektorým jeho predchodcom
- pre veľkú triedu prípustných heuristík, tzv konzistentných heuristík existuje omnoho efektívnejší spôsob spracovania znovunavštívených stavov

55

konzistentná heuristika

- heuristika je konzistentná, ak pre každý uzol *n* a pre každý jeho nasledovník *n'* generovaný aplikovaním operátora *a*,
- $h(n) \le c(n,a,n') + h(n')$
- ak *h* je konzistentná, platí
-

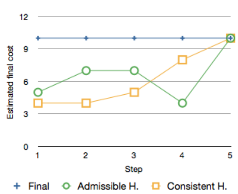
$$f(n') = g(n') + h(n') = g(n) + c(n,a,n') + h(n') \ge g(n) + h(n) = f(n)$$



- t.j. *f(n)* je neklesajúca pozdĺž ľubovoľnej cesty.
-
- Veta: Ak je *h(n)* konzistentná, A* je optimálny.
-

56

prípustná a konzistentná heuristika

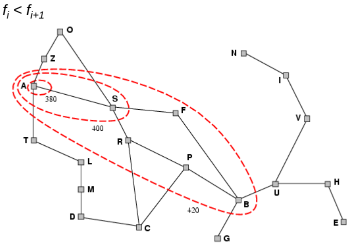


- na obrázku je f(n)
- heuristika:
  - konzistentná ⇒ prípustná
  - prípustná ¬⇒ konzistentná

57

prípustnosť A*

- A* rozvíja uzly v poradí podľa zvyšujúcej sa hodnoty *f*
- postupne pridáva obrysy uzlov "*f*-obrysy"
- obrys *i* má všetky uzly s hodnotou vyhodnocovacej funkcie $f = f_i$, kde $f_i < f_{i+1}$
-



58

úplnosť A*

- A* rozvíja uzly v poradí zvyšujúcej sa hodnoty vyhodnocovacej funkcie f *(najprv porozvíja všetky uzly s menším ohodnotením a až potom prejde na uzly s najbližším vyšším ohodnotením)*
- skôr či neskôr, ale po konečnom počte krokov, musí dôjsť na rozvitie cieľového uzla.
- alebo nie? áno, ak nie je v grafe nekonečne veľa uzlov s f(u) < C*.
- čo je možné vtedy, ak
  - sú uzly s nekonečným faktorom vetvenia (počtom aplikovateľných operátorov)
  - (alebo) existuje cesta c s konečnou cenou ale nekonečným počtom uzlov pozdĺž nej*.
  - * = Zenónov paradox dichotómie

59

Zenonov paradox dichotómie

- Zenonov paradox dichotómie: „To, čo sa pohybuje, musí najprv dospieť do polovice cesty skôr, než dospeje do konca. Ale aby to dospelo do polovice cesty, musí to najprv dospieť do polovice tejto polovice, a tak ďalej. Takže pohyb sa vlastne ani nemôže začať."
- Tento paradox poukazuje na to, že prejsť konečnú dĺžku znamená prejsť nekonečne veľa bodov, a tieto dve veci sa nám intuitívne zdajú v rozpore. Tento rozpor sa však výrazne zmierni (alebo dokonca celkom zanikne) ak si uvedomíme, že podobne ako vzdialenosť môžeme deliť aj čas. Ak sa niečo pohybuje stále rovnakou rýchlosťou, tak to do polovice cesty dospeje za polovicu času, do štvrtiny cesty to dospeje za štvrtinu času, a tak ďalej. Čím kratší úsek, tým kratší čas potrebný na jeho prekonanie*.
- * = časopis .týždeň, 3/2008.

60

záležitosť s časovým ohraničením

- Ak problém nemá riešenie, A* sa vykonáva donekonečna, ak stavový priestor je nekonečný alebo ak sa stavy môžu znovu navštevovať ľubovoľný počet ráz.
- Ak aj problém má riešenie, jeho nájdenie si môže vyžadovať obrovský čas. Pričom my nevieme odhadnúť vopred, koľko času bude treba, nevieme dokonca vopred, či vôbec má riešenie.
- V praxi treba čas vykonávania vopred ohraničiť. Pokiaľ A* nenájde riešenie do stanoveného limitu, hľadanie sa skončí. V takom prípade ale nevieme nič: či problém nemá riešenie, alebo má riešenie, lebo bolo treba viac trpezlivosti.
- Pri malých problémoch to nemusí vadiť. pri väčších je meta-problém: ako nastaviť limit?

dominujúca heuristika

- Nech h_1 a h_2 sú prípustné heuristiky.
  Ak $h_2(n) \geq h_1(n)$ pre všetky n
- tak h_2 dominuje nad h_1

- dôsledok: h_2 je lepšia na hľadanie
-
- Príklad. Typické náklady hľadania (priemerný počet rozvitých uzlov):
-
- *d=12* IDS = 3,644,035 uzlov
  $A^*(h_1)$ = 227 uzlov
  $A^*(h_2)$ = 73 uzlov
- *d=24* IDS = príliš veľa uzlov
  $A^*(h_1)$ = 39,135 uzlov
  $A^*(h_2)$ = 1,641 uzlov
-

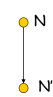
tvrdenie o A*

Nech h je konzistentná heuristika. Vždy keď A* rozvinie uzol, tak už našiel optimálnu cestu do stavu reprezentovaného týmto uzlom

dôkaz (1/2)

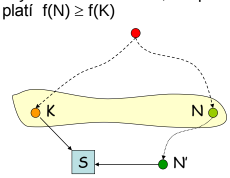
1) uvažujme uzol N a jeho potomka N'
keďže h je konzistentná: $h(N) \leq c(N,N')+h(N')$

$f(N) = g(N)+h(N) \leq g(N)+c(N,N')+h(N') = f(N')$
takže f je neklesajúca pozdĺž každej cesty



dôkaz (2/2)

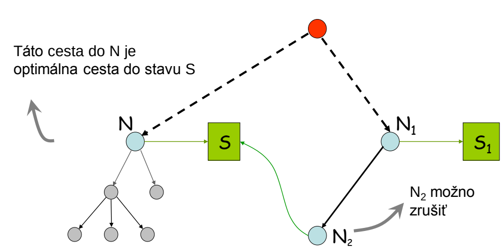
2) ak sa uzol K vyberie na rozvinutie, tak pre ľubovoľný iný uzol N na OKRAJi platí $f(N) \geq f(K)$



ak uzol N leží na inej ceste do stavu S reprezentovaného uzlom K, tak cena tejto inej cesty nie je menšia než cena cesty do K:
$f(N') \geq f(N) \geq f(K)$
a platí $h(N') = h(K)$ lebo sa ohodnocuje ten istý stav
takže $g(N') \geq g(K)$

(t.j. ak sa vybral na rozvitie K, tak cena cesty do neho nie je väčšia, než ľubovoľná iná cesta)

dôsledok tvrdenia



znovunavštívené stavy pri hľadaní s konzistentnou heuristikou

- keď sa uzol rozvije, ulož jeho stav do UZAVRETÉ
- keď sa vygeneruje nový uzol N:
  - ak STAV(N) je v UZAVRETÉ, zruš N
  - ak existuje uzol N' na okraji taký, že STAV(N') = STAV(N), zruš ten z nich – N alebo N' – s vyšším ohodnotením funkciou f

67

Je A* s nejakou konzistentnou heuristikou to jediné, čo potrebujeme?

Nie !

Jestvujú veľmi hlúpe ale konzistentné heuristické funkcie

68

napr: $h \equiv 0$

- je konzistentná (a teda prípustná) !
- A* s $h \equiv 0$ je hľadanie s rovnomernou cenou
- hľadanie do šírky a s rovnomernou cenou sú zvláštne prípady A*

69

Zvláštne prípady

- Ak nás zaujíma iba to, aby sme sa do cieľa dostali hocijakou cestou, trebárs aj nie najlepšou, tak môžeme definovať *g* tak, že vždy bude jej hodnotou 0. Vtedy sa vyberie vždy uzol, ktorý sa javí byť najbližšie k cieľu (ide o lačné hľadanie).
- Ak chceme nájsť cestu, ktorá obsahuje najmenší počet uzlov, tak definujeme cenu prechodu z uzla do jeho nasledovníka ako konštantu, najčastejšie 1.
- Ak heuristiku vôbec nepoužijeme (*h* sa definuje tak, že jej hodnota je vždy 0), tak sa hľadanie riadi stratégiou rovnomernej ceny.
- Ak navyše definujeme cenu prechodu z uzla do jeho nasledovníka ako 1, tak ide o hľadanie do šírky.

70

presnosť heuristiky

nech h_1 a h_2 sú dve konzistentné heuristiky také, že h_2 dominuje nad h_1 (t.j. pre všetky uzly N platí:

$h_1(N) \leq h_2(N)$)

hovoríme aj, že h_2 je presnejšia (informovanejšia) než h_1

5		8
4	2	1
7	3	6

STAV(N)

1	2	3
4	5	6
7	8	

cieľový stav

- $h_1(N)$ = počet zle položených doštičiek
- $h_2(N)$ = súčet vzdialeností každej doštičky do jej cieľového miesta
- h_2 je presnejšia než h_1

71

tvrdenie o A*

- nech h_2 je presnejšia než h_1
- nech A_1* je A* s použitím h_1 a nech A_2* je A* s použitím h_2
- vždy ak existuje riešenie, tak všetky uzly rozvinuté A_2*, možno s výnimkou niektorých uzlov takých, že
  $f_1(N) = f_2(N) = C^*$ (cena optimálneho riešenia)
  sú rozvinuté aj A_1*

72

Cyklicky sa prehlbujúce hľadanie algoritmom A* (IDA*)

- IDA* je úplný a prípustný.
- Tak ako hľadanie do hĺbky, vyžaduje pamäť v rozsahu úmernom dĺžke najdlhšej cesty, ktorú prezerá.

function IDA*(*problém*) **returns** *riešenie* alebo neúspech
  static: *f-hranica*, súčasná hranica určená funkciou *f*-CENA
    koreň, uzol

  koreň ← VYTVOR-UZOL(ZAČIATOČNÝ-STAV[problém])
  f-hranica ← *f*-CENA(*koreň*)
  loop do
    riešenie, *f-hranica* ← HDH-OBRYS(*koreň*, *f-hranica*)
    if *riešenie* nie je null **then return** *riešenie*
    if *f-hranica* = ∞ **then return** neúspech
  end

Cyklicky sa prehlbujúce hľadanie algoritmom A* (IDA*)

function HDH-OBRYS(*uzol*, *f-hranica*)

  returns *riešenie* a nová hranica určená funkciou *f*-CENA
  static: *ďalšia-f*, hranica určená funkciou *f*-CENA pre ďalší obrys oblasti hľadania, na začiatku ∞

  if *f*-CENA(*uzol*) > *f-hranica* **then return** null, *f*-CENA(*uzol*)
  if CIEĽOVÝ-TEST[*problém*] aplikovaný na STAV(*uzol*) je úspešný
      then return VYBER-RIEŠENIE (*uzol*), *f-hranica*

  for each uzol *u* **in** NASLEDOVNÍKY(*uzol*) **do**
    riešenie, *nová-f* ← HDH-OBRYS(*u*, *f-hranica*)
    if *riešenie* nie je null **then return** *riešenie*, *f-hranica*
    ďalšia-f ← MIN(*ďalšia-f*, *nová-f*)
  end
  return null, *ďalšia-f*

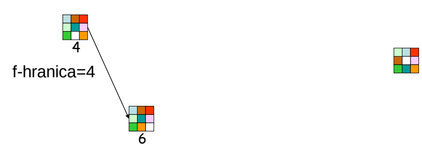
IDA* - príklad

Algoritmus IDA*:

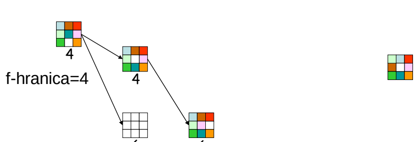
1. Inicializuj f-hranica na f-cena(koreň)
2. Opakuj:
   a. Prehľadaj do hĺbky všetky uzly spĺňajúce podmienku f-cena(N) ≤ f-hranica
   b. Priraď f-hranica najnižšiu hodnotu zo všetkých nerozvitých uzlov (listov)

IDA* - príklad

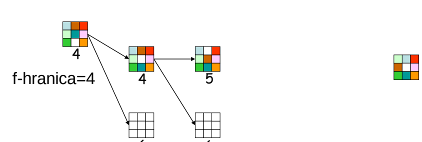
f(N) = g(N) + h(N)
h(N) = počet nesprávne umiestnených políčok



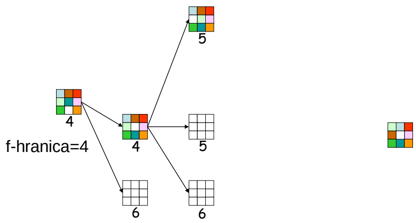
IDA* - príklad



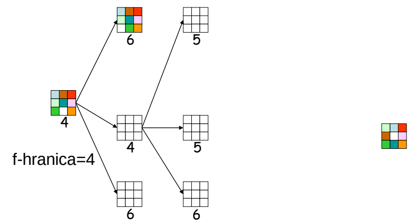
IDA* - príklad



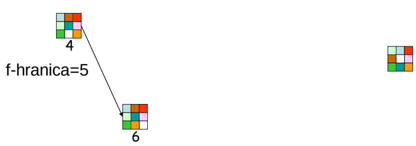
IDA* - príklad



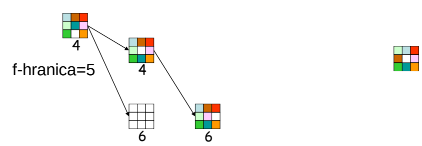
IDA* - príklad



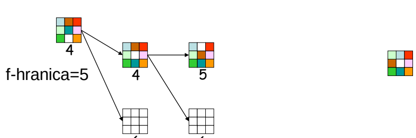
IDA* - príklad



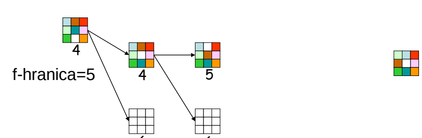
IDA* - príklad



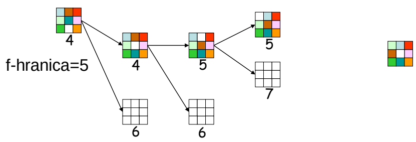
IDA* - príklad



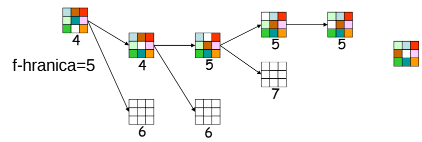
IDA* - príklad



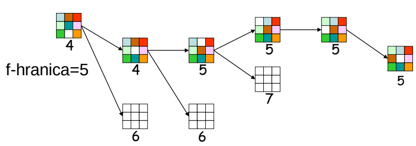
IDA* - príklad



IDA* - príklad



IDA* - príklad



účinný faktor vetvenia

- používa sa meranie účinnosti heuristiky
- nech n je celkový počet uzlov rozvitých stratégiou A* pre daný problém a nech d je hĺbka riešenia
- účinný faktor vetvenia b* sa definuje
  $n = 1 + b^* + (b^*)^2 + ... + (b^*)^d$

výsledky experimentov

(pozri R&N)

- 8-hlavolam s:
  - h_1 = počet zle položených doštičiek
  - h_2 = súčet vzdialeností doštičiek od ich cieľovej polohy
- náhodné generovanie mnohých inštancií problému
- priemerné účinné faktory vetvenia (počet rozvinutých uzlov):

d	IDS	A_1*	A_2*
2	2.45	1.79	1.79
6	2.73	1.34	1.30
12	2.78 (3,644,035)	1.42 (227)	1.24 (73)
16	--	1.45	1.25
20	--	1.47	1.27
24	--	1.48 (39,135)	1.26 (1,641)

výhody/nevýhody IDA*

- výhody:
  - úplný a optimálny
  - potrebuje menej pamäti než A*
  - nestráca čas usporadúvaním okraja
- nevýhody:
  - nedá sa vyhnúť znovunavštíveniu stavov, ktoré nie sú na súčasnej ceste
  - slabé využívanie pamäti

kedy použiť hľadanie?

1) stavový priestor je malý a
- iný spôsob riešenia problému nepoznáme alebo
- vyvíjať efektívnejší spôsob nestojí za to

2) stavový priestor je veľký a
- iný spôsob riešenia problému nepoznáme ad
- existujú dobré heuristiky