I007 Osnove umjetne inteligencije

Tema: Pretraživanje prostora stanja.

3. 3. 2021.

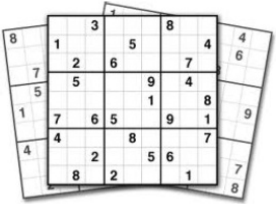
predavač: Darija Marković

asistent: Darija Marković

1. Pretraživanje prostora stanja
    - Slijepo pretraživanje

- mnogo se analitičkih problema može riješiti pretraživanjem prostora stanja
- krenuvši od početnog stanja problema, pokušavamo pronaći ciljno stanje
- slijed akcija koje nas vode do ciljnog stanja predstavljaju rješenje problema
- problem predstavlja velik broj stanja te velik broj mogućih izbora
- pretraživanje zato mora biti sustavno


E I-290/Eisenhower Expy E
Exit Wacker Dr/Franklin St
W Harrison St
0.2 mi
MENU
1.1 mi



- neka je S skup stanja (prostor stanja)
- problem se sastoji od početnog stanja, prijelaza između stanja i ciljnog (ciljnih) stanja

Problem pretraživanja

problem = $(s_0, \mathrm{succ}, \mathrm{goal})$

1. $s_0 \in S$ je **početno stanje**
2. $\mathrm{succ} : S \to \wp(S)$ je **funkcija sljedbenika** koja definira prijelaze između stanja
3. $\mathrm{goal} : S \to \{\top, \bot\}$ je **ispitni predikat** istinit samo za ciljna stanja

- funkcija sljedbenika može se definirati implicitno pomoću skupa operatora (različitim operatorima prelazi se u različita stanja) početno stanje:

8		7
6	5	4
3	2	1

ciljno stanje:

1	2	3
4	5	6
7	8	

Koji potezi vode do rješenja?

$$\mathit{problem} = (s_0, \mathrm{succ}, \mathrm{goal})$$

$$s_0 = \begin{bmatrix} 8 & \blacksquare & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$$

$$\mathrm{succ}\left(\begin{bmatrix} 8 & \blacksquare & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}\right) = \left\{ \begin{bmatrix} \blacksquare & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}, \begin{bmatrix} 8 & 7 & \blacksquare \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}, \begin{bmatrix} 8 & 5 & 7 \\ 6 & \blacksquare & 4 \\ 3 & 2 & 1 \end{bmatrix} \right\}$$

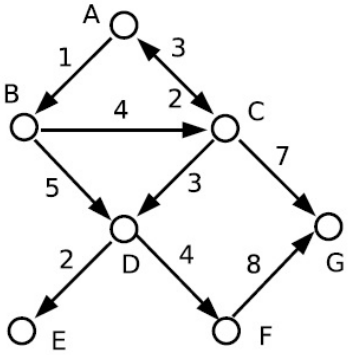
$$\vdots$$

$$\mathrm{goal}\left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & \blacksquare \end{bmatrix}\right) = \top$$

$$\mathrm{goal}\left(\begin{bmatrix} 8 & \blacksquare & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}\right) = \bot$$

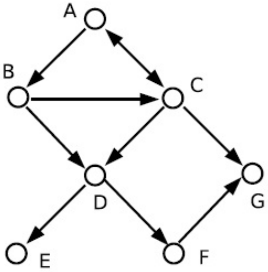
$$\mathrm{goal}\left(\begin{bmatrix} \blacksquare & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}\right) = \bot$$

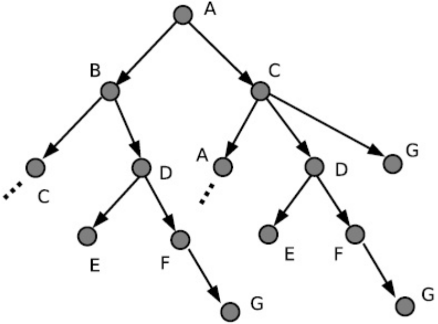
$$\vdots$$



- prostor pretraživanje prostora stanja svodi se na pretraživanje usmjerenog grafa (digrafa)
- vrhovi grafa = stanja; lukovi = prijelazi između stanja
- graf može biti zadan eksplicitno ili implicitno
- graf može imati cikluse
- ako definiramo cijene prijelaza, onda je to usmjeren težinski graf (težinski digraf)

- pretraživanjem usmjerenog grafa postepeno gradimo stablo pretraživanja
- stablo gradimo tako da pojedine čvorove proširujemo: pomoću funkcije sljedbenika (odnosno operatora) generiramo sve sljedbenike nekog čvora
- otvoreni čvorovi ili fronta: čvorovi koji su generirani, ali još nisu prošireni
- zatvoreni čvorovi: čvorovi koji su već prošireni
- redoslijed kojim proširujemo čvorove određuje strategiju pretraživanja





- stablo pretraživanja nastaje pretraživanjem prostora stanja
- stablo pretraživanja može biti beskonačno čak i onda kada je prostor stanja konačan (prostor stanja ima cikluse $\Rightarrow$ stablo pretraživanja je beskonačno)

- čvor n je podatkovna struktura koja sačinjava stablo pretraživanja
- čvor pohranjuje stanje, ali i još neke dodatne podatke:

Podatkovna struktura čvora

$$n = (s, d)$$

$s-$ stanje $d-$ dubina čvora u stablu

$\mathrm{state}(n) = s, \mathrm{depth}(n) = d$

$\mathrm{initial}(s_0) = (s_0, 0)$

Opći algoritam pretraživanja

function search(s_0, succ, 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
      insert(m, $open$)
  return $fail$

- $\mathrm{removeHead}(l)$: skida prvi element neprazne liste l
- $\mathrm{expand}(n, \mathrm{succ})$: proširuje čvor n uporabom funkcije sljedbenika succ
- $\mathrm{insert}(n, l)$: umeće čvor n u listu l

- proširivanje čvora treba ažurirati sve komponente čvora:

Proširivanje čvora

function $\mathrm{expand}(n, \mathrm{succ})$
  return $\{\ (s, \mathrm{depth}(n) + 1) \mid s \in \mathrm{succ}(\mathrm{state}(n))\ \}$

- funkcija će biti složenija kada u čvor budemo pohranjivali dodatne podatke (npr. pokazivač na roditeljski čvor)

Karakteristike problema:

- $|S|$ – broj stanja
- b – faktor grananja stabla pretraživanja
- d – dubina optimalnog rješenja u stablu pretraživanja
- m – maksimalna dubina stabla pretraživanja (moguće ∞)

Svojstva algoritama:

1. Potpunost (engl. completeness) – algoritam je potpun akko pronalazi rješenje uvijek kada ono postoji
2. Optimalnost (engl. optimality, admissibility) – algoritam je optimalan akko pronalazi optimalno rješenje (ono s najmanjom cijenom)
3. Vremenska složenost (broj generiranih čvorova)
4. Prostorna složenost (broj pohranjenih čvorova)

Dvije osnovne vrste strategija pretraživanja:

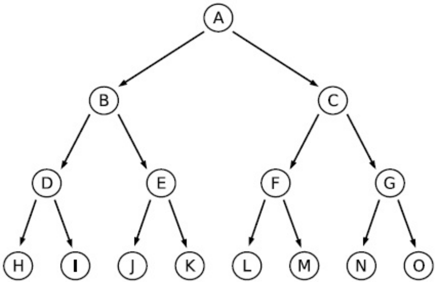
- Slijepo pretraživanje (engl. blind, uninformed search)
- Usmjereno pretraživanje (engl. directed, informed, heuristic search)

Slijepo pretraživanje

1. Pretraživanje u širinu (engl. breadth-first search, BFS)
2. Pretraživanje s jednolikom cijenom (engl. uniform-cost search)
3. Pretraživanje u dubinu (engl. depth-first search, DFS)
4. Ograničeno pretraživanje u dubinu
5. Iterativno pretraživanje u dubinu
6. Dvosmjerno pretraživanje

Pretraživanje u širinu

- jednostavna slijepa strategija pretraživanja
- nakon proširenja korijenskog čvora, proširuju se sva njegova djeca, zatim sva njihova djeca, itd.
- općenito, čvorovi na dubini d proširuju se tek nakon što se prošire svi čvorovi na razini $d - 1$, tj. pretražujemo razinu po razinu



A, B, C, D, E, F, G, H, . . .

Ovakvu strategiju ostvarit ćemo ako proširene čvorove uvijek dodajemo na kraj liste otvorenih čvorova

Pretraživanje u širinu

function $\mathrm{breadthFirstSearch}(s_0, \mathrm{succ}, \mathrm{goal})$
  $open \leftarrow [\mathrm{initial}(s_0)\,]$
  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
      $\mathrm{insertBack}(m, open)$
  return $fail$

- lista otvorenih čvorova zapravo je red (engl. queue)

Pretraživanje u širinu – svojstva

- pretraživanje u širinu je potpuno i optimalno
- u svakom koraku proširuje se najplići čvor, pa je strategija optimalna (uz pretpostavku da je cijena prijelaza konstantna)
- vremenska složenost:

$$1 + b + b^2 + b^3 + \cdots + b^d + (b^{d+1} - b) = \mathcal{O}(b^{d+1})$$

  (na zadnjoj razini generiraju se sljedbenici svih čvorova osim ciljnog)
- prostorna složenost: $\mathcal{O}(b^{d+1})$
- eksponencijalna složenost (pogotovo prostorna) glavni je nedostatak pretraživanja u širinu
- primjenjivo samo na male probleme

Cijene prijelaza

- ako operacije (prijelazi izmedu stanja) nisu jednake cijene, funkciju sljedećeg stanja modificiramo tako da ona za svakog sljedbenika vraća i cijenu prijelaza:

$$\mathrm{succ} : S \to \wp(S \times \mathbb{R}^+)$$

- u čvoru više ne pohranjujem dubinu nego ukupnu cijenu puta do tog čvora:

$$n = (s; c); \quad g(n) = c$$

- funkciju proširenja čvora moramo također modificirati tako da ažurira cijenu puta do čvora:

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

Pretraživanje s jednolikom cijenom

- kao i pretraživanje u širinu, no u obzir uzimamo cijenu prijelaza

Pretraživanje s jednolikom cijenom

function $\mathrm{uniformCostSearch}(s_0, \mathrm{succ}, \mathrm{goal})$
  $open \leftarrow [\mathrm{initial}(s_0)\,]$
  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
      $\mathrm{insertSortedBy}(g, m, open)$
  return $fail$

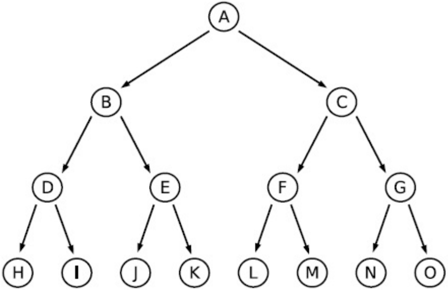
- $\mathrm{insertSortedBy}(f, n, l)$ – umeće čvor n u listu l sortiranu uzlazno prema vrijednosti $f(n)$
- lista $open$ funkcionira kao prioritetni red

Pretraživanje s jednolikom cijenom $-$ svojstva

- algoritam je potpun i optimalan
- ako je C^* optimalna cijena do cilja, a ε minimalna cijena prijelaza, dubina stabla do ciljnog čvora je $d = \lfloor C^*/\varepsilon \rfloor$
- prostorna i vremenska složenost: $\mathcal{O}(b^{1+\lfloor C^*/\varepsilon \rfloor})$

Pretraživanje u dubinu

- pretraživanje u dubinu uvijek prvo proširuje najdublji čvor u stablu pretraživanja
- postupak se vraća na pliće razine tek kada dosegne listove (stanja koja nemaju sljedbenika)



A, B, D, H, I, E, J, K, C, . . .

Pretraživanje u dubinu − izvedba

- strategiju pretraživanja u dubinu ostvarit ćemo ako proširene čvorove dodajemo na početak liste $open$

Pretraživanje u dubinu

function $\mathrm{depthFirstSearch}(s_0, \mathrm{succ}, \mathrm{goal})$
  $open \leftarrow [\mathrm{initial}(s_0)\,]$
  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
      $\mathrm{insertFront}(\,m, open)$
  return $fail$

- lista otvorenih čvorova zapravo je stog

Pretraživanje u dubinu − svojstva

- pretraživanje u dubinu manje je memorijski zahtjevno
- prostorna složenost: $\mathcal{O}(bm)$, gdje je m maksimalna dubina stabla
- vremenska složenost: $\mathcal{O}(b^m)$
  (nepovoljno, ako $m \gg d$)
- potpunost: ne, jer može zaglaviti u beskonačnoj petlji
- optimalnost: ne, jer ne pretražuje razinu po razinu
- pretraživanje u dubinu treba izbjegavati kod stabla pretraživanja čija je maksimalna dubina velika ili beskonačna

Pretraživanje u dubinu − rekurzivna izvedba

- listu $open$ možemo izbjeći:

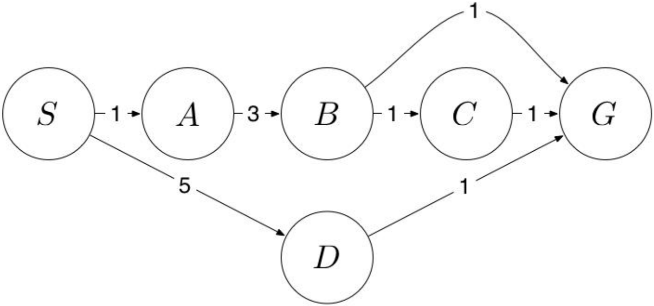
Pretraživanje u dubinu (rekurzivna izvedba)

function $\mathrm{depthFirstSearch}(s, \mathrm{succ}, \mathrm{goal})$
  if $\mathrm{goal}(s)$ then return s
  for $m \in \mathrm{succ}(s)$ do
    $r \leftarrow \mathrm{depthFirstSearch}(m, \mathrm{succ}, \mathrm{goal})$
    if $r \neq fail$ then return r
  return $fail$

- umjesto eksplicitne liste $open$ koristi se sistemski stog
- prostorna složenost je $\mathcal{O}(m)$

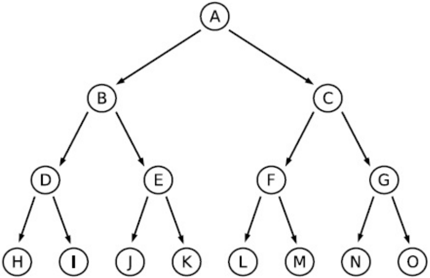
Primjer 1.

Za strategije pretraživanja u širinu, pretraživanja s jednolikom cijenom i pretraživanja u dubinu razmotrit ćemo redosljed posjećivanja čvrova, prateći frontu (listu otvorenih čvorova). U pozicijama gdje imamo nekoliko izbora za proširivanje, dat ćemo prednost izboru s manjom abecednom vrijednošću, npr. $S \to X \to A$ proširujemo prije $S \to X \to B$ i slično $S \to A \to Z$ proširujemo prije $S \to B \to A$.



Ograničeno pretraživanje u dubinu

- pretražuje u dubinu, ali ne dublje od zadane granice



$k = 0$: A
$k = 1$: A, B, C
$k = 2$: A, B, D, E, C, F, G

Ograničeno pretraživanje u dubinu − izvedba

- čvor proširujemo samo ako se u stablu pretraživanja nalazi iznad dubinskog ograničenja k:

Ograničeno pretraživanje u dubinu

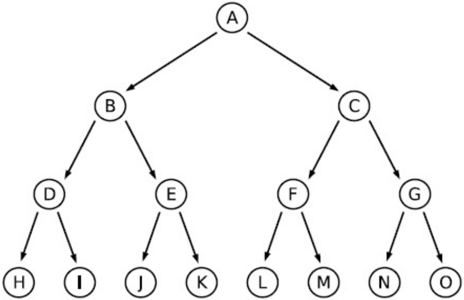
function $\mathrm{depthLimitedSearch}(s_0, \mathrm{succ}, \mathrm{goal}, k)$
    $open \leftarrow [\mathrm{initial}(s_0)\ ]$
    while $open \neq [\ ]$ do
        $n \leftarrow \mathrm{removeHead}(open)$
        if $\mathrm{goal}(\mathrm{state}(n))$ then return n
        if $\mathrm{depth}(n) < k$ then
            for $m \in \mathrm{expand}(n, \mathrm{succ})$ do
                $\mathrm{insertFront}(m, open)$
    return $fail$

Ograničeno pretraživanje u dubinu − svojstva

- prostorna složenost: $\mathcal{O}(bk)$, gdje je k dubinska granica
- vremenska složenost: $\mathcal{O}(b^k)$
- potpunost: da, ali samo ako $d \leq k$
- optimalnost: ne, jer ne pretražuje razinu po razinu
- algoritam je uporabiv ako znamo dubinu rješenja d
  (možemo postaviti $k = |S|$)

Iterativno pretraživanje u dubinu

- izbjegava problem izbora optimalne dubinske granice isprobavajući sve moguće vrijednosti krenuvši od dubine 0
- kombinira prednosti pretraživanja u dubinu i pretraživanja u širinu



A, A, B, C, A, B, D, E, C, F, G, A, B, D, H, . . .

Iterativno pretraživanje u dubinu − izvedba

Iterativno pretraživanje u dubinu

function $\mathrm{iterativeDeepeningSearch}(s_0, \mathrm{succ}, \mathrm{goal}, k)$
    for $k \leftarrow 1$ to ∞ do
        $result \leftarrow \mathrm{depthLimitedSearch}(s_0, \mathrm{succ}, \mathrm{goal}, k)$
        if $result \neq fail$ then return $result$
    return $fail$

Iterativno pretraživanje u dubini − svojstva

- strategija se na prvi pogled čini neučinkovitom: više puta proširujemo iste čvorove
- u većini slučajeva to ne predstavlja problem: većina čvorova stabla nalazi se na dubljim razinama, pa ponavljanje proširivanja čvorova na višim razinama nije problematično
- vremenska složenost: $\mathcal{O}(b^d)$
- prostorna složenost: $\mathcal{O}(bd)$
- potpunost: da, jer koristi dubinsko ograničenje
- optimalnost: da, jer pretražuje razinu po razinu
- Iterativno pretraživanje u dubinu preporučena je strategija za probleme s velikim prostorom stanja i nepoznatom dubinom rješenja

Dvosmjerno pretraživanje

- istovremeno se pretražuje od početnog stanja prema ciljnom stanju i od ciljnog stanja prema početnom stanju
- pretraživanje se zaustavlja najkasnije onda kada se dvije fronte susretnu na polovici puta
- npr. ako se u oba smjera koristi pretraživanje u širinu, onda su prostorna i vremenska složenost $\mathcal{O}(2b^{d/2}) = \mathcal{O}(b^{d/2})$
  Značajna ušteda!
- nedostatak: postupak je primjenjiv samo ako problem (1) ima malen broj eksplicitno definiranih ciljnih stanja i (2) svi operatori imaju inverze

Usporedba algoritama slijepog pretraživanja

Algoritam	Vrijeme	Prostor	Potpunost	Opt.
U širinu	$\mathcal{O}(b^{d+1})$	$\mathcal{O}(b^{d+1})$	Da	Da
Jednol. cijena	$\mathcal{O}(b^{1+\lfloor C^*/\varepsilon \rfloor})$	$\mathcal{O}(b^{1+\lfloor C^*/\varepsilon \rfloor})$	Da	Da
U dubinu	$\mathcal{O}(b^m)$	$\mathcal{O}(bm)$	Ne	Ne
Ogr. u dubinu	$\mathcal{O}(b^k)$	$\mathcal{O}(bk)$	Da, ako $d \leq k$	Ne
Iter. u dubinu	$\mathcal{O}(b^d)$	$\mathcal{O}(bd)$	Da	Da
Dvosmjerno	$\mathcal{O}(b^{\frac{d}{2}})$	$\mathcal{O}(b^{\frac{d}{2}})$	Da	Da

b – faktor grananja, d – dubina optimalnog rješenja,
m – maksimalna dubina stabla $(m \geq d)$, k – dubinsko ograničenje

- svi su algoritmi eksponencijalne vremenske složenosti!
- pretraživanje u dubinu (i njegove varijante) bolje su prostorne složenosti od pretraživanja u širinu