Mesterséges intelligencia előadássorozat

Az előadás diái az AIMA könyvre épülve (http://aima.cs.berkeley.edu) készültek a University of California, Berkeley mesterséges intelligencia kurzusának anyagainak felhasználásával (http://ai.berkeley.edu).

These slides are based on the AIMA book (http://aima.cs.berkeley.edu) and were adapted from the AI course material of University of California, Berkeley (http://ai.berkeley.edu).

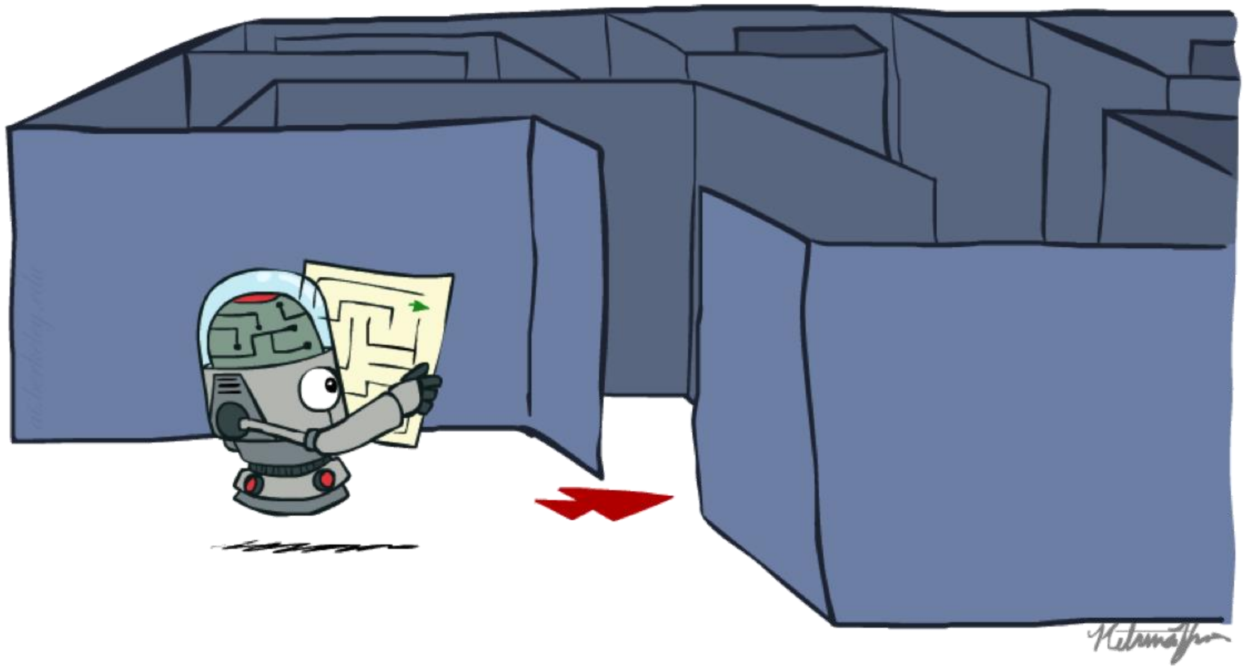
Mesterséges intelligencia

Problémamegoldás kereséssel

Előadó: Dr. Hullám Gábor

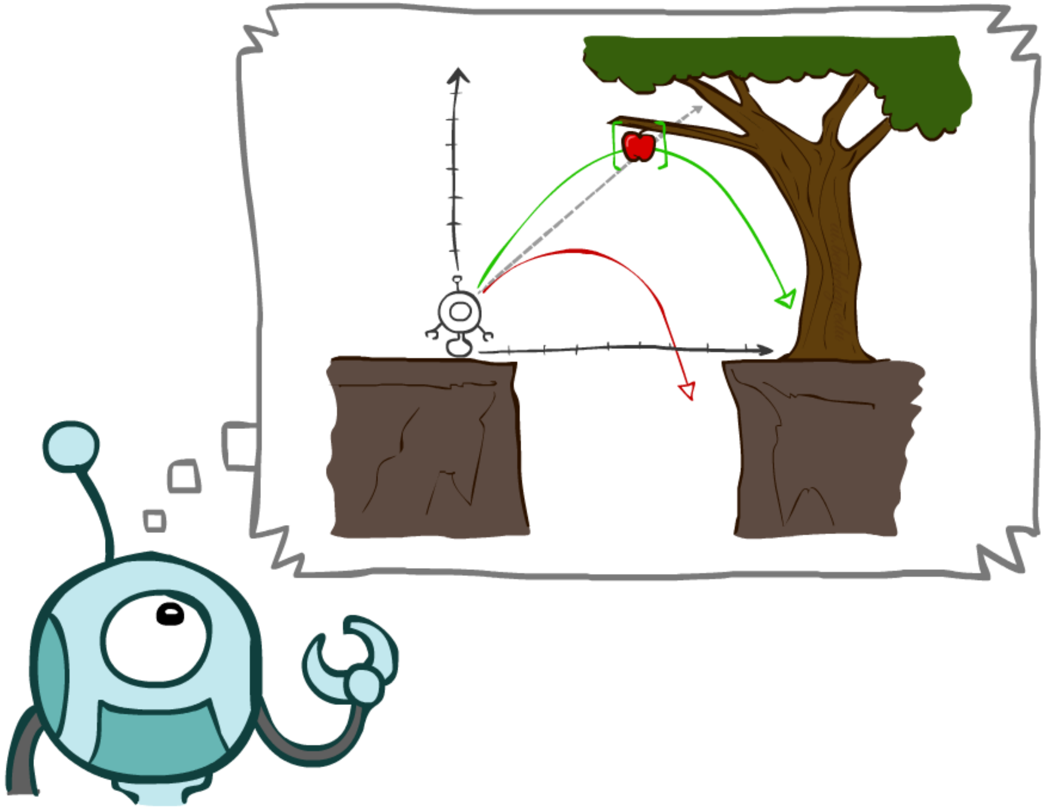
Problémamegoldás kereséssel

Nem informált keresés

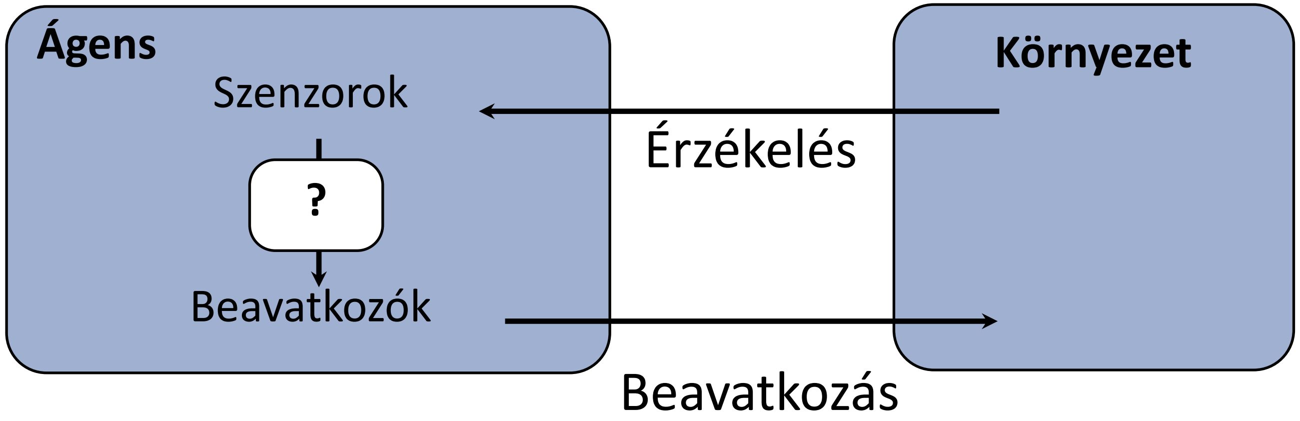


Tartalom

- Ágensek és a tervkészítés
- Keresési problémák
- Nem informált keresési módszerek
  - Mélységi keresés (Depth-First Search)
  - Szélességi keresés (Breadth-First Search)
  - Egyenletes költségű keresés (Uniform-Cost Search)



Ágens és a környezet



- Az ágens ***szenzorokon*** keresztül érzékeli a környezetet és
- ***beavatkozó szerv***eken keresztül cselekszik, azaz befolyásolja a környezetet

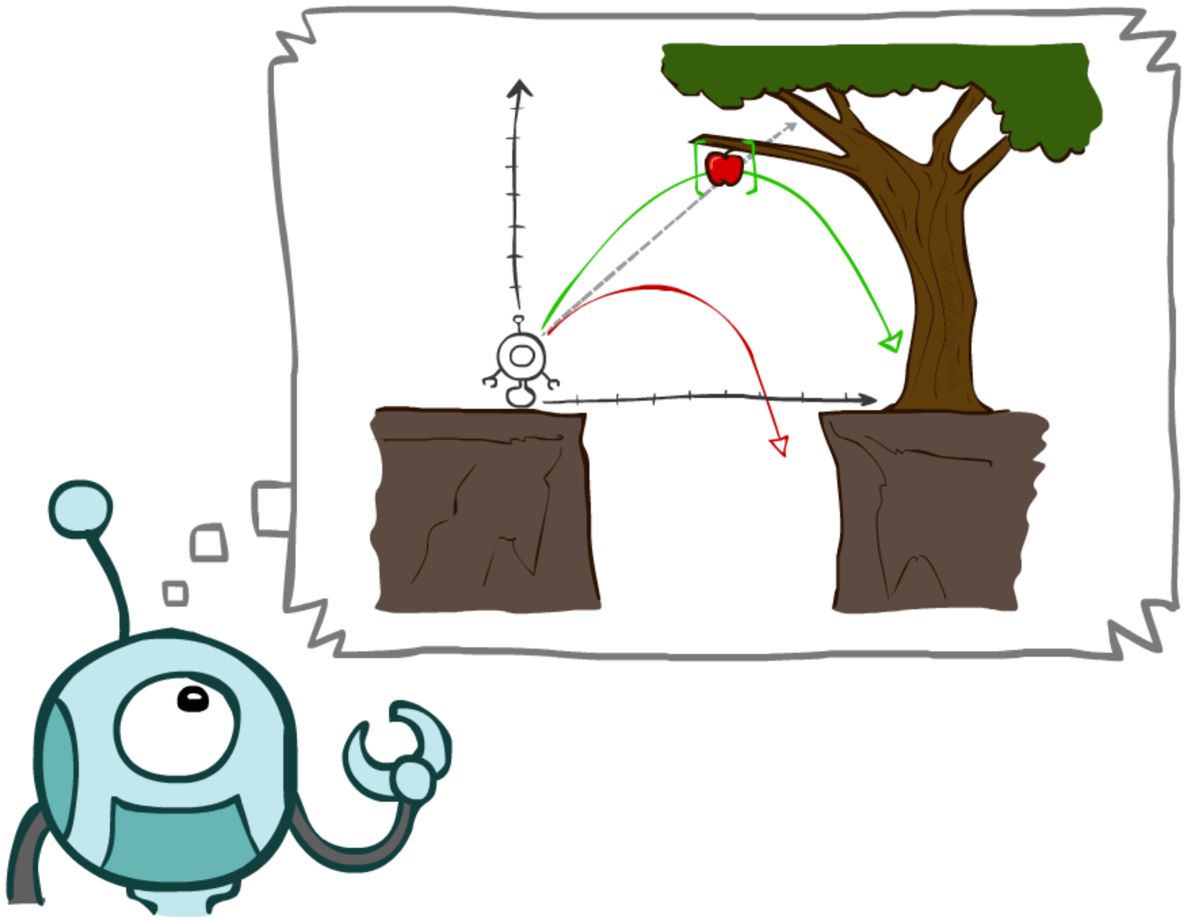
Racionalitás

- A ***racionális ágens*** úgy választ a lehetséges cselekvések közül, hogy általuk maximalizálja a ***várható hasznosságot***
  - Egyszerű eset: az ágens rendelkezik céllal és ismeri a költségeket
    - Például a cél a lehető legalacsonyabb költséggel elérni a célállapotot
  - Komplex eset: az ágens rendelkezik egy hasznosság függvénnyel, ismeri a várható jutalmakat, stb.
    - Az ágens olyan cselekvéseket hajt végre, melyek maximalizálják a teljes jutalmat adott idő alatt (például az elérhető nyereséget)

Ágensek tervezése

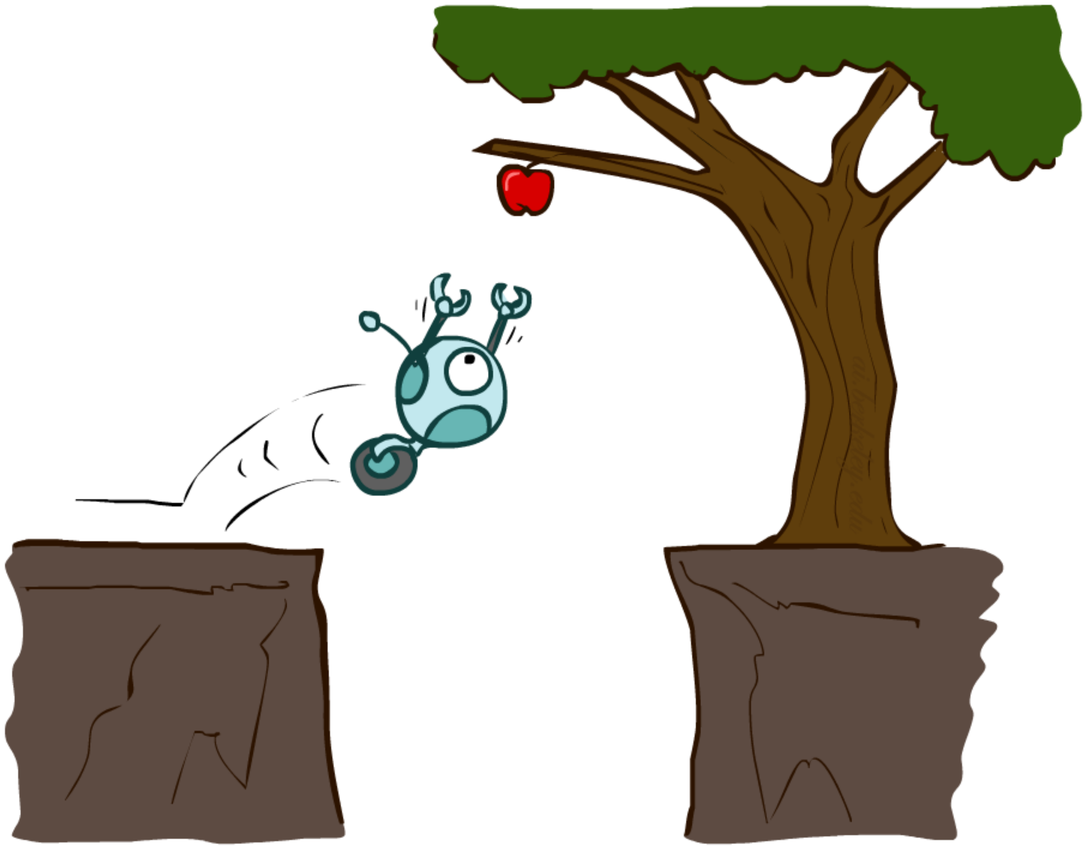
- **A környezet nagy mértékben befolyásolja az alkalmazandó ágens kialakítását, elvárt képességeit**
- **A környezet lehet…**
  - ***Teljesen/részlegesen megfigyelhető***=> az ágensnek szüksége lehet ***memóriára*** (belső állapot nyilvántartására)
  - ***Diszkrét/folytonos*** => az ágens lehet, hogy nem képes az összes lehetséges állapot megkülönböztetésére
  - ***Sztochasztikus/determinisztikus*** => az ágensnek lehet, hogy több lehetséges ***forgatókönyvet*** kell kidolgoznia
  - ***Egyedüli ágens/ Több ágens*** => lehet, hogy az ágensnek ***véletlenszerűen*** kell viselkednie

Tervkészítő ágensek



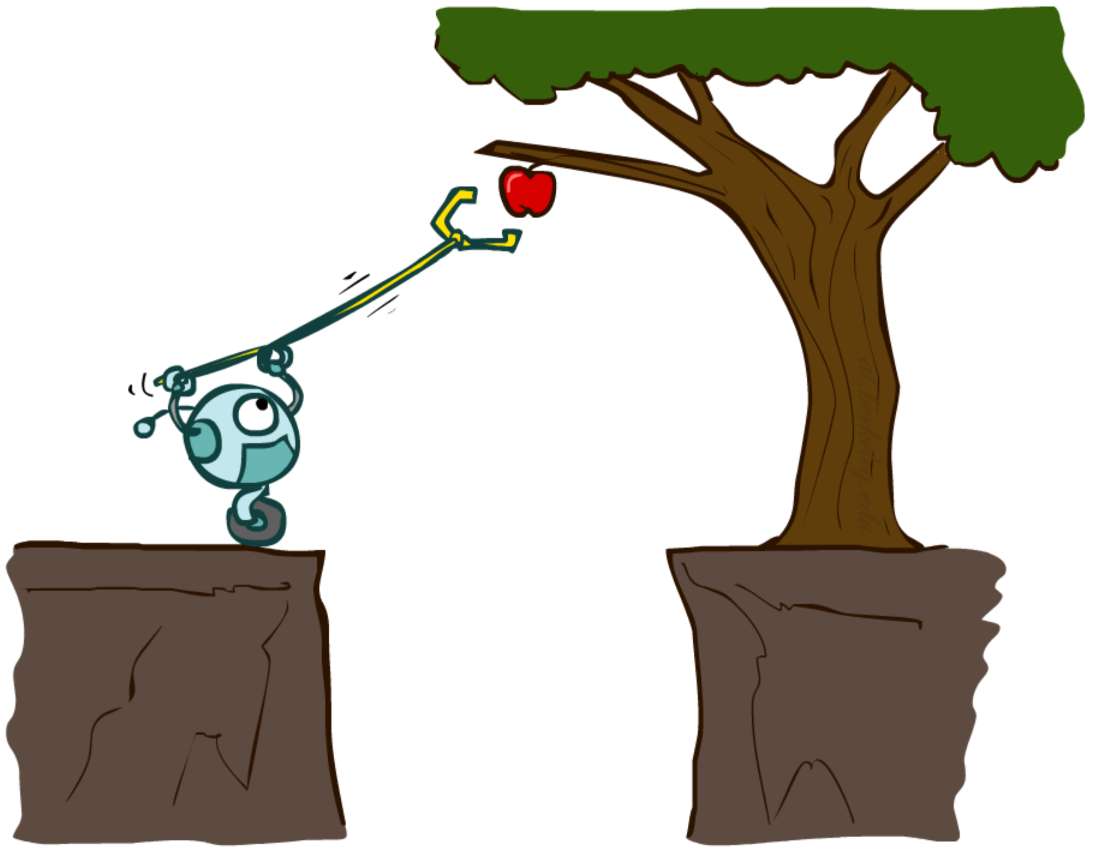
Reflex Ágens

- Reflex ágens:
  - Az aktuális érzékelés (esetleg memória) alapján választ cselekvést
  - A környező világ állapotát nyilvántarthatja a memóriájában vagy egy belső modell segítségével
  - Nem veszi figyelembe a cselekvések jövőbeli következményeit
  - A világot úgy tekinti, ahogy van
- Lehet-e egy reflex ágens racionális?



Tervkészítő ágensek

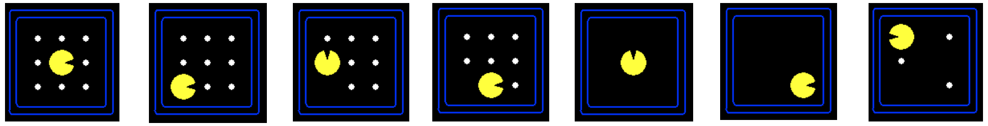
- Tervkészítő ágensek:
  - “Mi lenne ha?” kérdések feltevése
  - **Döntést** a cselekvések lehetséges következményeinek függvényében hoz.
  - Ehhez rendelkeznie kell egy **modellel**, ami leírja, hogy a világ hogyan változik a cselekvések hatására
  - Meg kell határoznia egy **célt** (tesztelhető célfüggvény)
  - Figyelembe veszi, hogy a világ milyen lehet / milyenné válhat



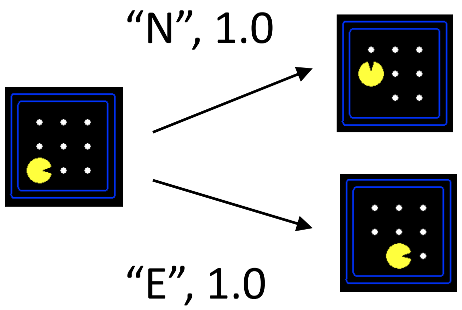
Keresési problémák

Keresési problémák

- Egy keresési probléma az alábbi elemeket tartalmazza:
  - Állapottér



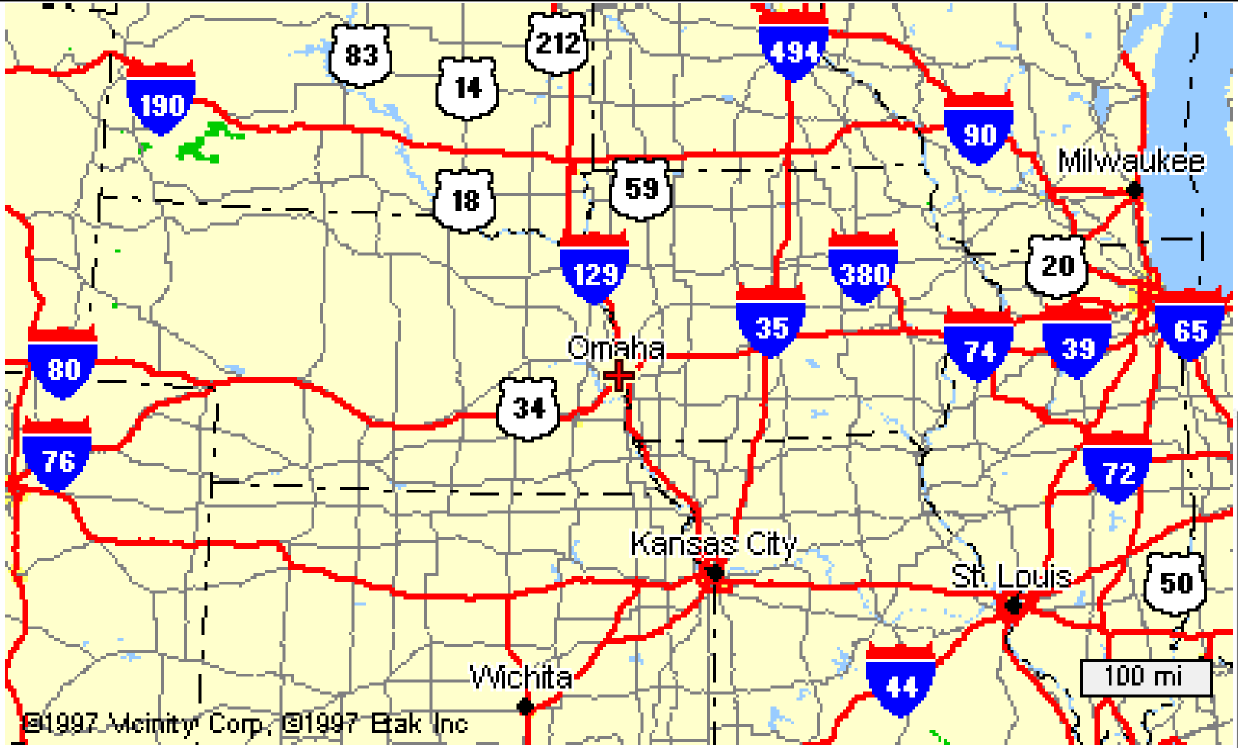
  - Állapotátmenet-függvény (cselekvések, költségek)



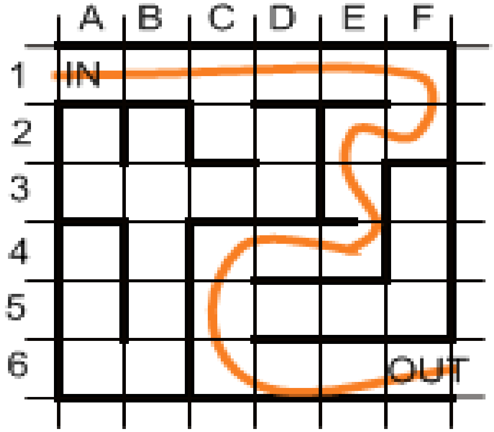
  - Kiindulási állapot és egy célállapot teszt

- A megoldás a cselekvések egy sorozata (egy terv) , amely a kiindulási állapotból eljuttat egy célállapotba (állapottranszformációs lépések révén)

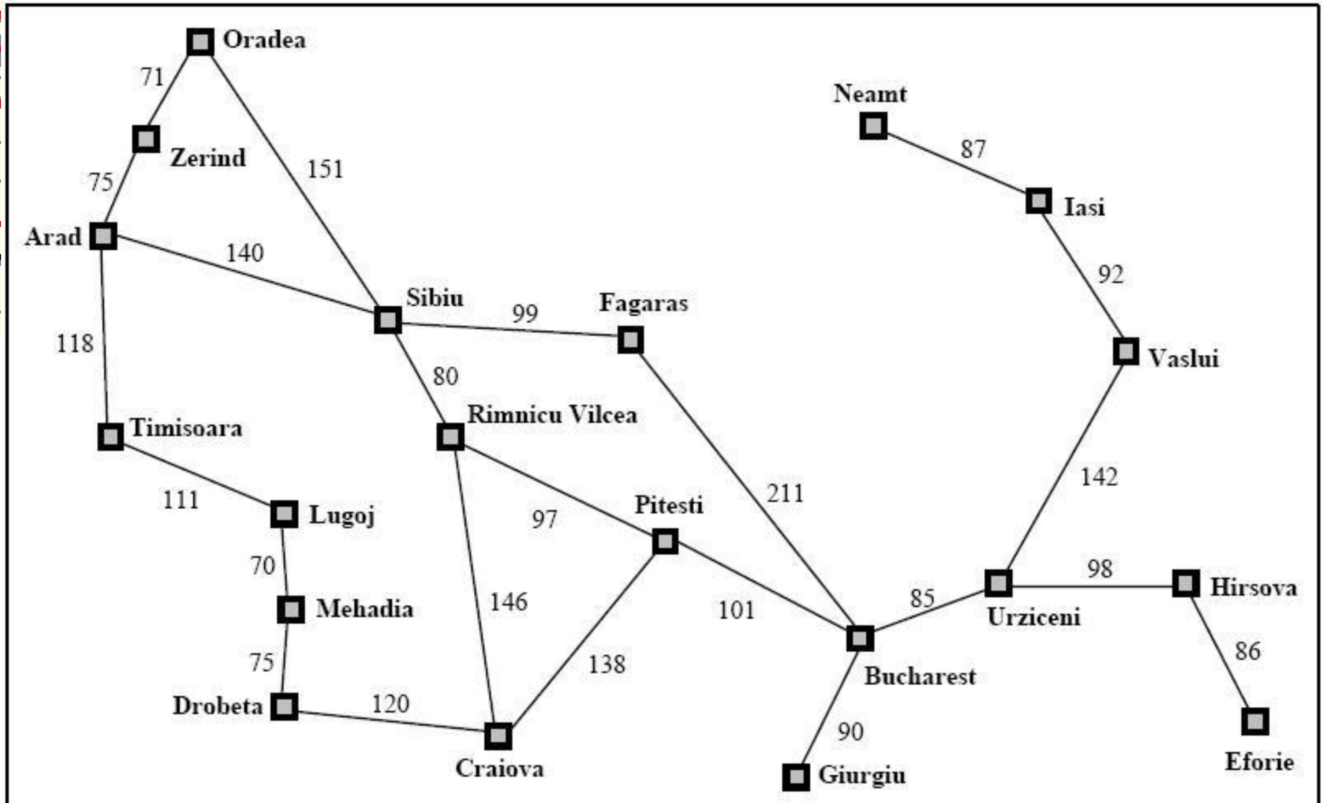
Fizikai tér



83
14
212
494
190
90
Milwaukee
18
59
129
380
20
35
74
39
65
Omaha
80
34
76
72
Kansas City
St. Louis
50
Wichita
44
100 mi
©1997 Mcinity Corp, ©1997 Etak Inc

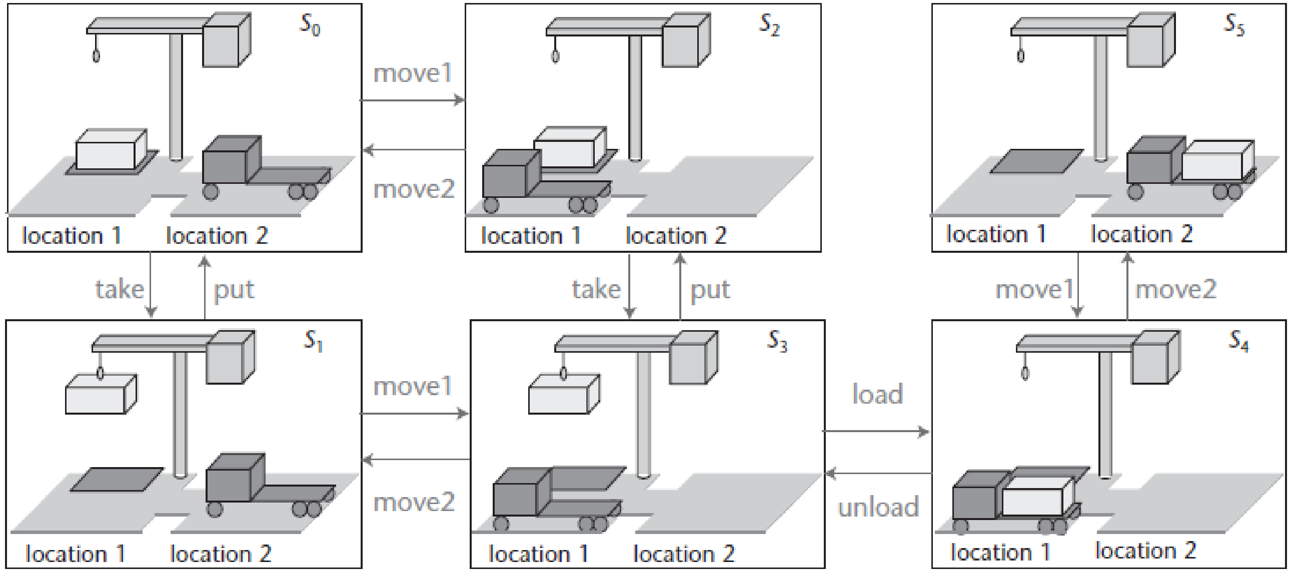


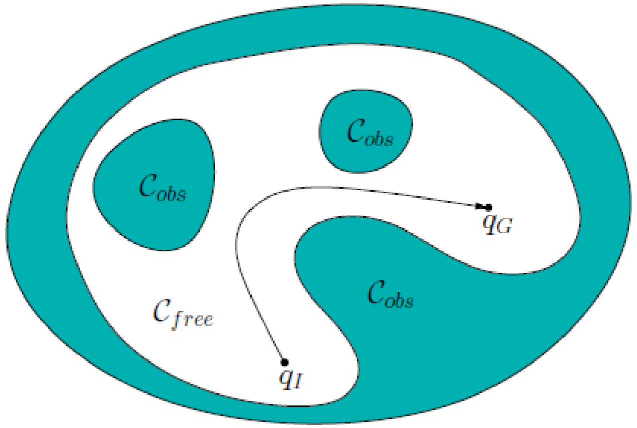
A
B
C
D
E
F
1
2
3
4
5
6
IN
OUT

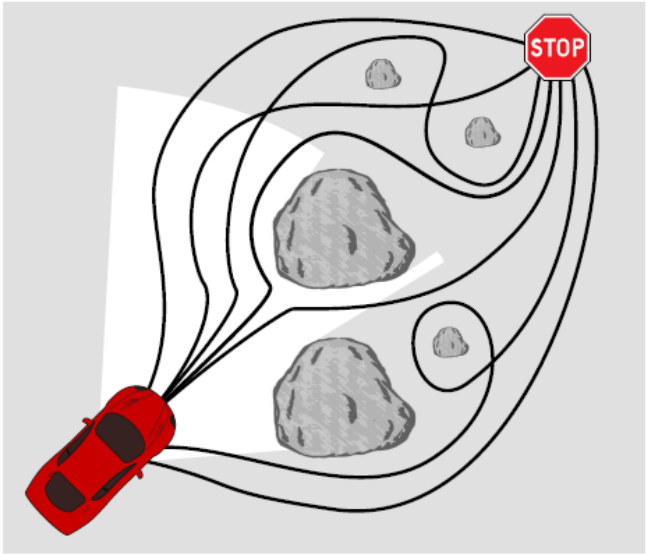


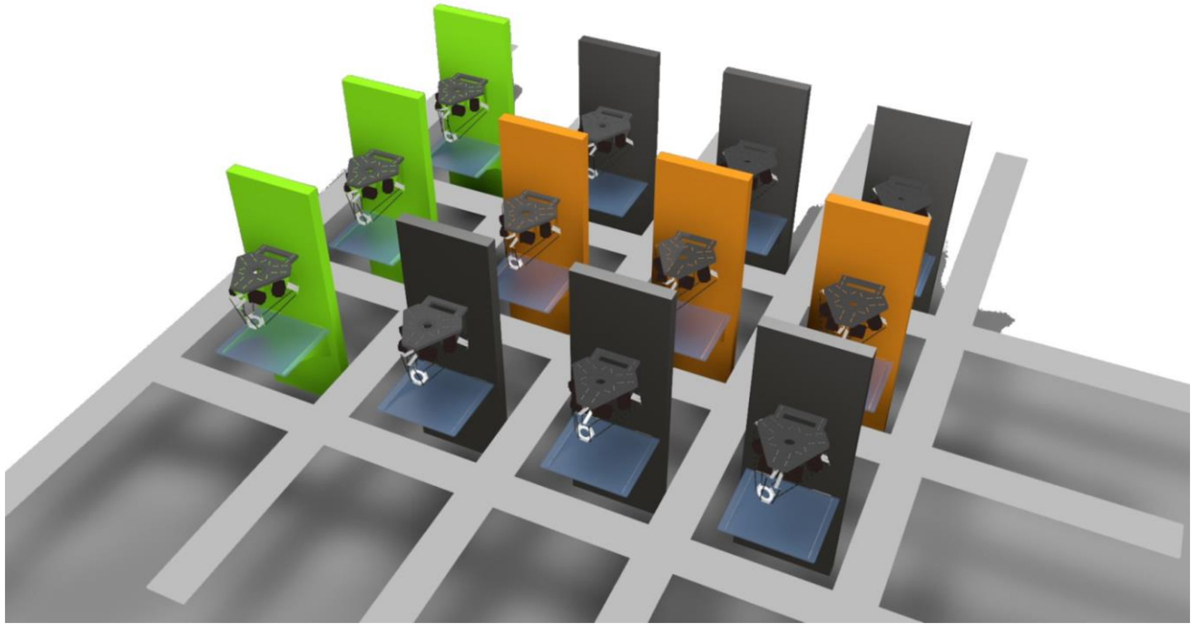
Oradea
71
Zerind
151
75
Arad
140
118
Sibiu
99
Fagaras
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
211
70
Mehadia
146
101
75
138
Drobeta
120
Craiova
Bucharest
90
Giurgiu
85
Urziceni
98
Hirsova
86
Eforie
142
Vaslui
92
Iasi
87
Neamt

Konfigurációs tér

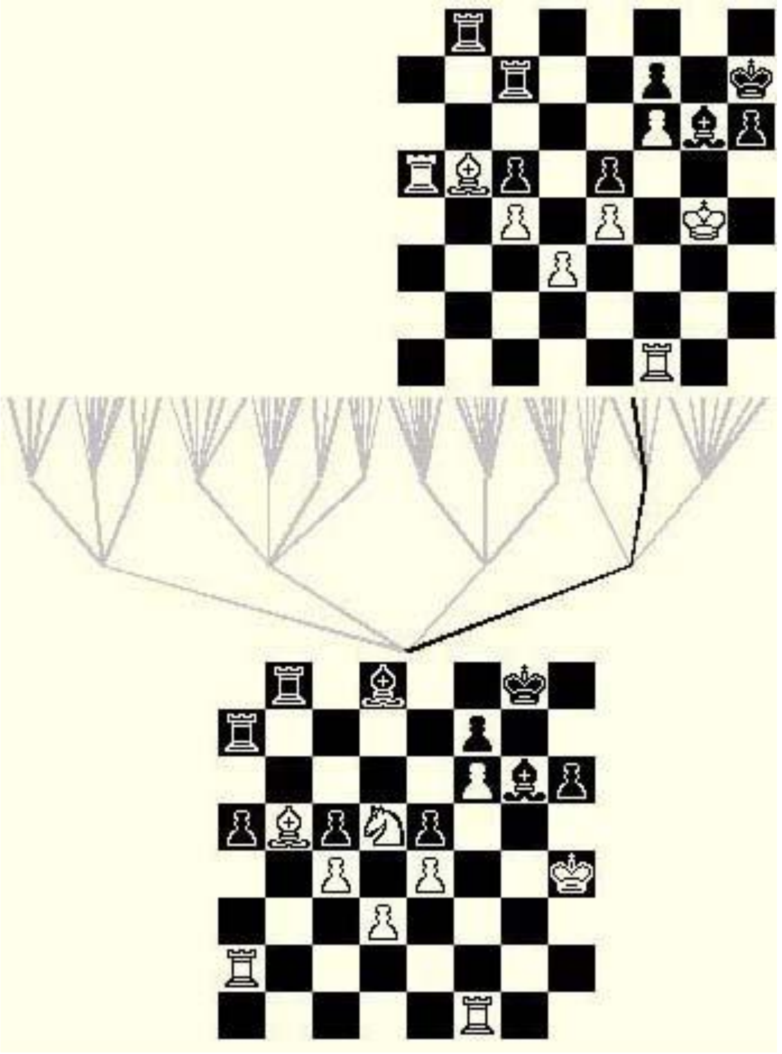


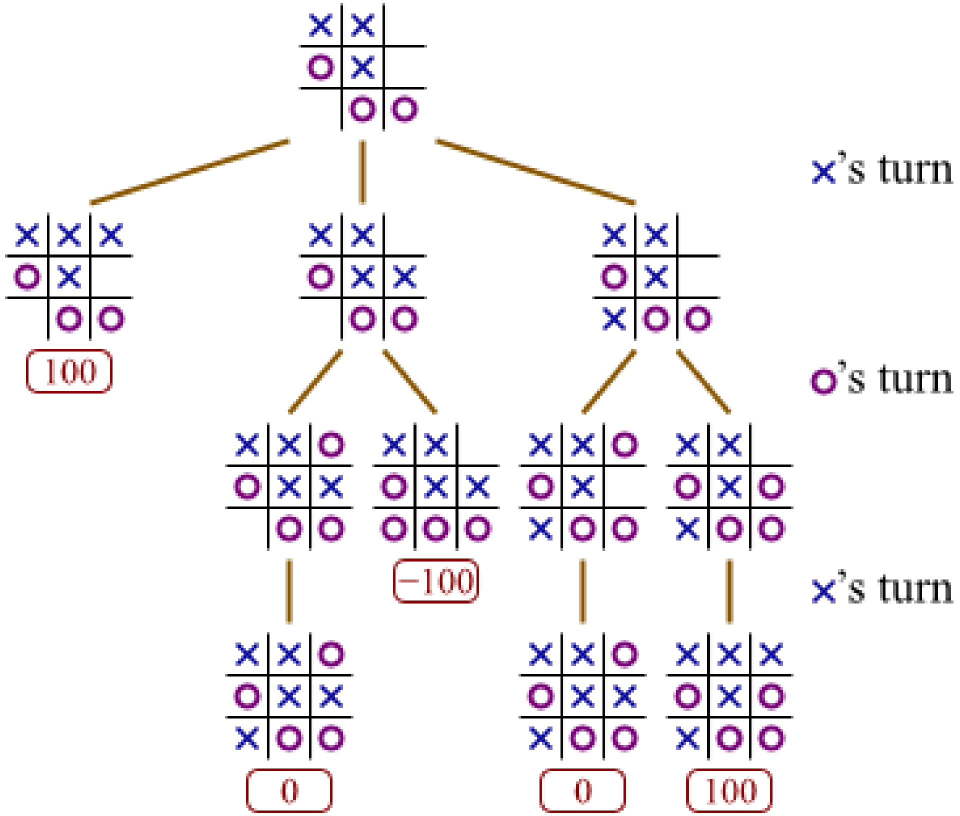


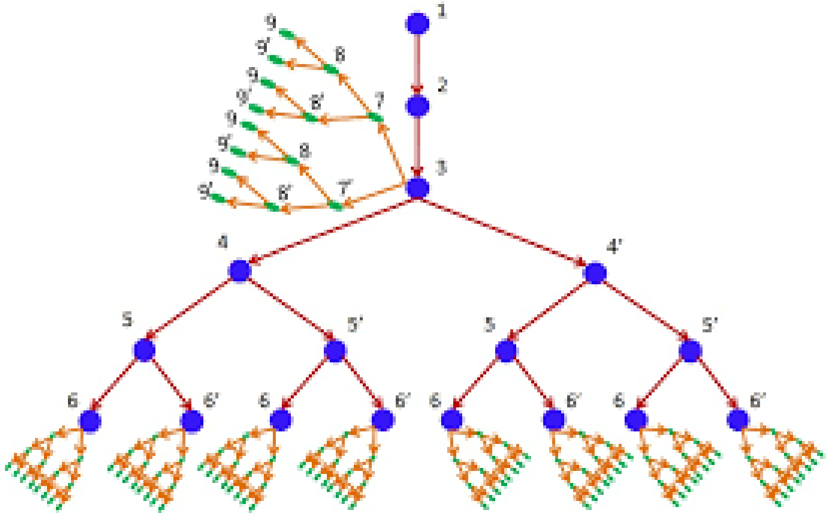




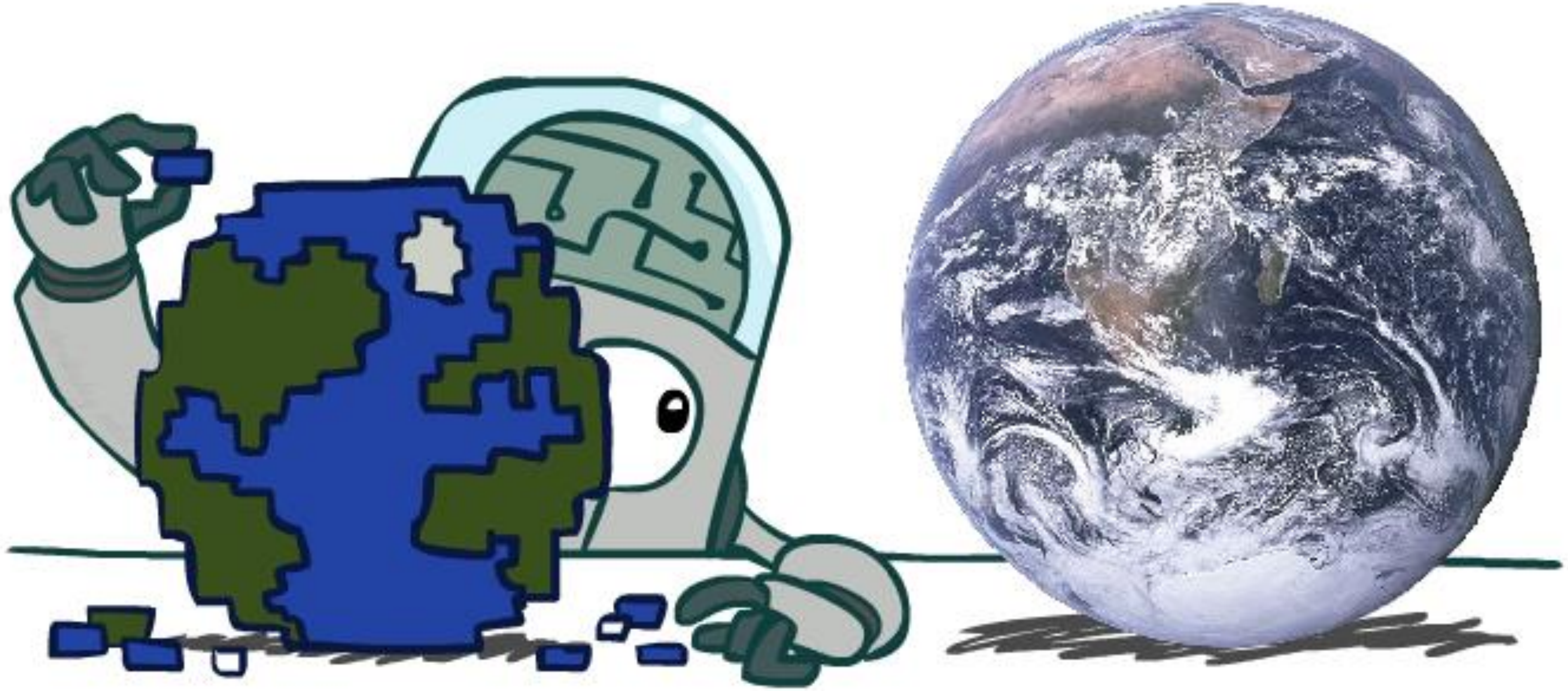
Absztrakt tér



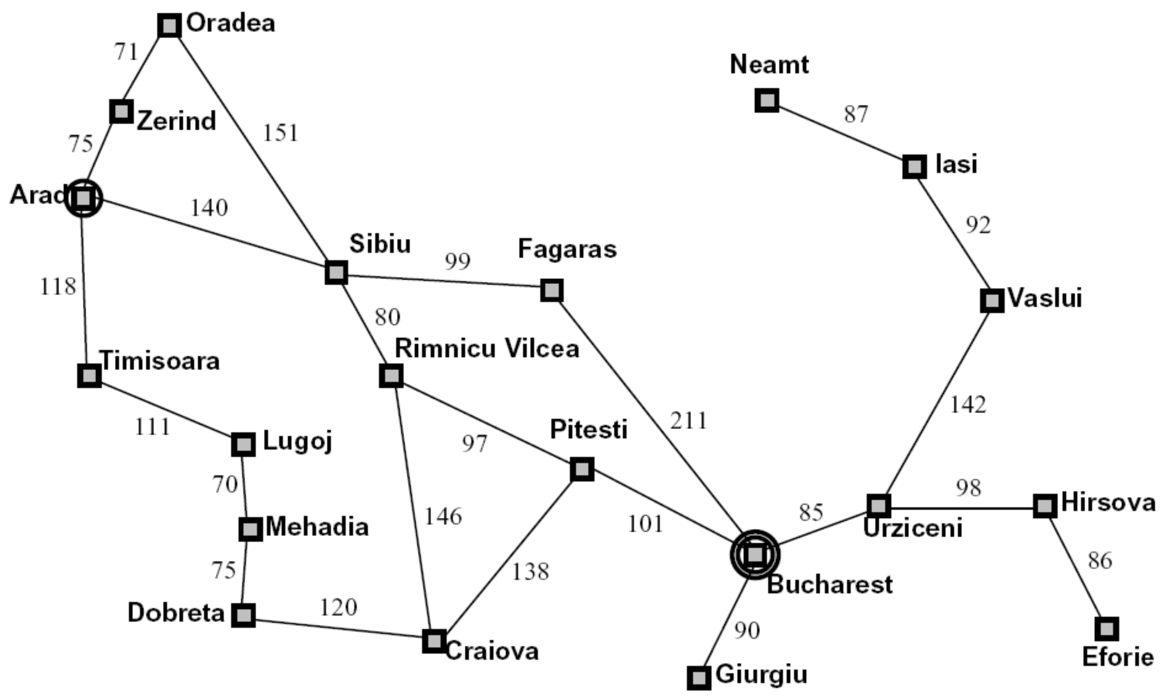




A keresési problémák modellek



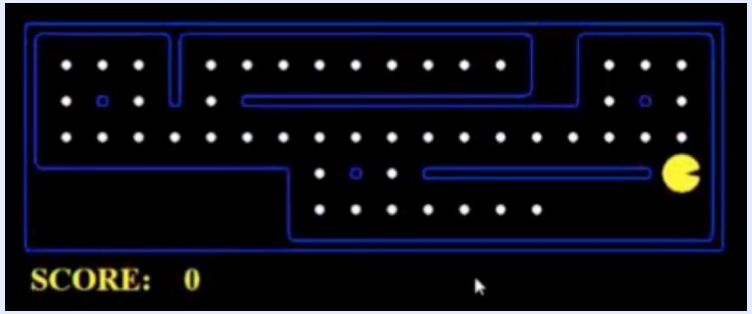
Példa: Útvonaltervezés



- Állapottér:
  - Városok
- Állapotátmenet-függvény:
  - Utak: Utazz a szomszédos városba adott költséggel (= távolság)
- Kiindulási állapot:
  - Arad
- Célállapot tesztelése:
  - Állapot == Bukarest?
- Megoldás?

Mit tartalmaz az állapottér?

A világ állapota a környezet minden részletét tartalmazza



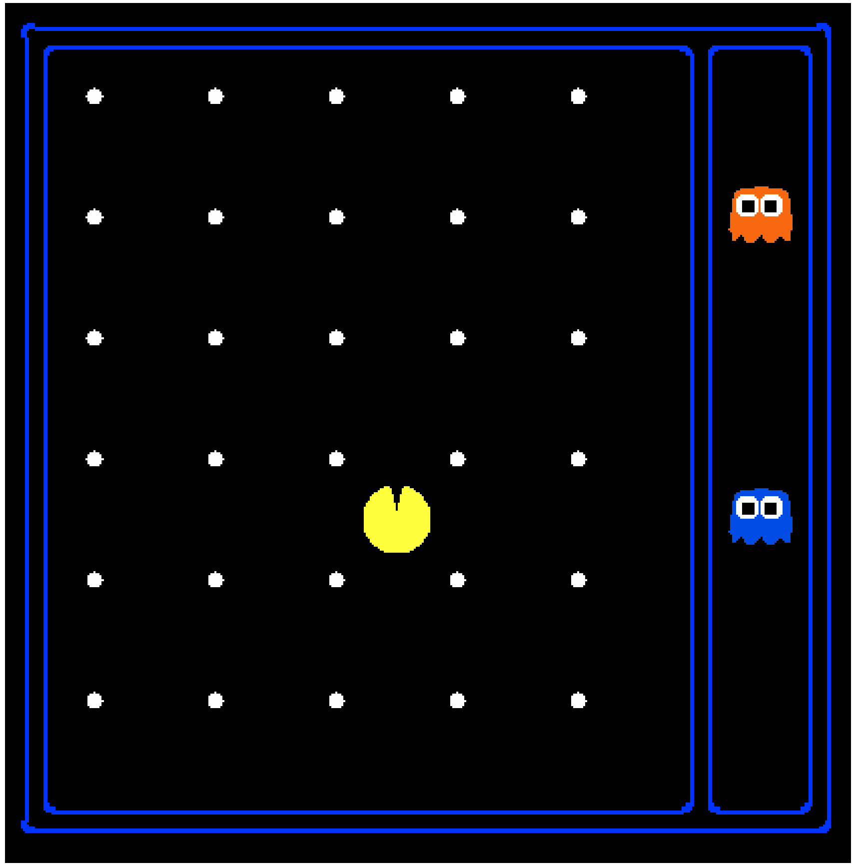
A keresési tér csak azokat a részleteket tartalmazza, melyekre szükség van a tervezéshez (absztrakció)

- 1. Probléma: útvonal
  - Állapotok: (x,y) helyzet
  - Cselekvések: ÉDKNY (mozgás)
  - Állapotátmenet függvény: helyzet frissítése
  - Célteszt: (x,y)=Vége
- 2. Probléma: „pontok megevése”
  - Állapotok: {(x,y), pontok - boolean}
  - Cselekvések: ÉDKNY (mozgás)
  - Állapotátmenet függvény: helyzet frissítése, (ha még ott van) pont jelenlét változó átállítása
  - Célteszt: nincs több pont

Állapotér nagysága

- Világállapotok:
  - Ágens pozíciók: 120
  - Pontok (élelem) száma: 30
  - Szellemek pozíciója: 12
  - Ágens iránya: ÉDKNY

- Nagyságrendek
  - Összes lehetséges állapot: $120 \text{x} (2^{30}) \text{x} (12^{2}) \text{x} 4$
  - Lehetséges útvonaltervezési állapotok: 120
  - Lehetséges pont (élelem) állapotok: $120 \text{x} (2^{30})$



Állapotér nagysága

Pl. a sakk – elég egyszerű szerkezetű állapottér (64 mező, 32 bábú)

1. lépés után (világos nyitólépése): **20** lehetséges állás
2. lépés után (sötét válaszlépése): **400** lehetséges állás
3. lépés után: **8.902**
4.lépés után: **197.281**
5. lépés után: **4.865.609**
stb.

A (jó) sakkozó nem egyforma intenzitással vizsgálja az egyes lehetőségeket!

Ágensek tervezése

- **A környezet nagy mértékben befolyásolja az alkalmazandó ágens kialakítását, elvárt képességeit**
- **A környezet lehet…**
  - ***Teljesen/részlegesen megfigyelhető***=> az ágensnek szüksége lehet ***memóriára*** (belső állapot nyilvántartására)
  - ***Diszkrét/folytonos*** => az ágens lehet, hogy nem képes az összes lehetséges állapot megkülönböztetésére
  - ***Sztochasztikus/determinisztikus*** => az ágensnek lehet, hogy több lehetséges ***forgatókönyvet*** kell kidolgoznia
  - ***Egyedüli ágens/ Több ágens*** => lehet, hogy az ágensnek ***véletlenszerűen*** kell viselkednie

Melyik a jó keresési eljárás?

- Elért eredmény szerint…
  - Megtaláltuk-e a legjobb célállapotot?
  - Találtunk-e legalább egy jó célállapotot?
  - Megtaláltuk-e az összes lehetséges célállapotot?

- Megtalálás költsége szerint…
  - A lehető legkisebb idő/tárhely
  - Melyik számít, számít-e egyáltalán?

- Az egzisztencia bizonyítás – létezik megoldás, de a gyakorlati részletek nem ismertek/nem érdekesek

Melyik a jó keresési eljárás?

1. Teljesség (*completeness*) - ha van megoldás, biztosan megtalálja

2. Időigény (*time complexity*) – mennyi idő a megoldás megtalálása?

3. Tárigény (*space complexity*) – mennyi tárhely kell

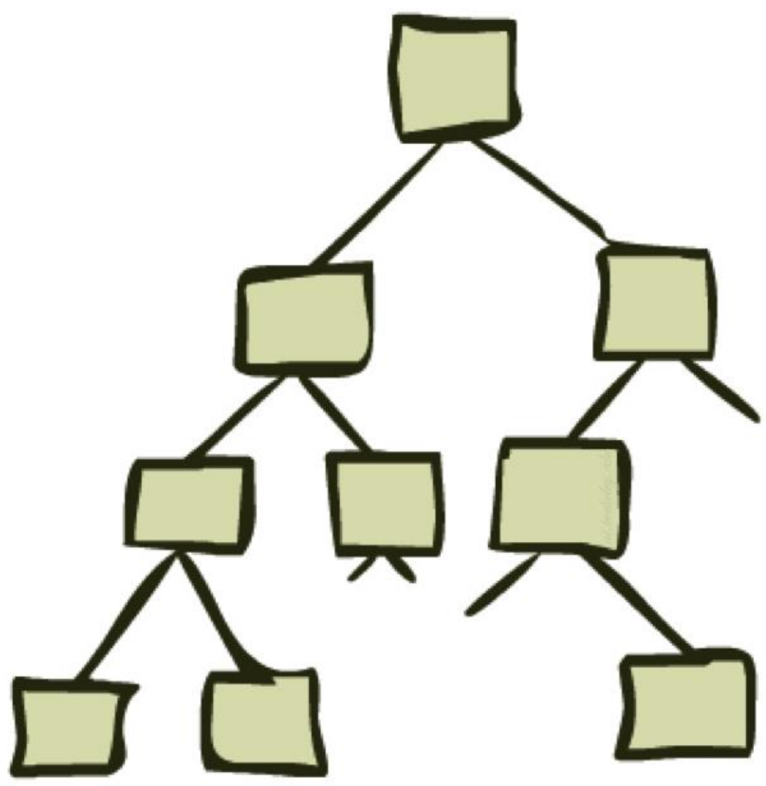
4. Optimalitás (*optimality*) – ha több megoldás van, megtaláljuk-e a legjobbat? (mi a legjobb? – sokszor izgalmas kérdés!)

Mesterséges intelligencia

Problémamegoldás kereséssel

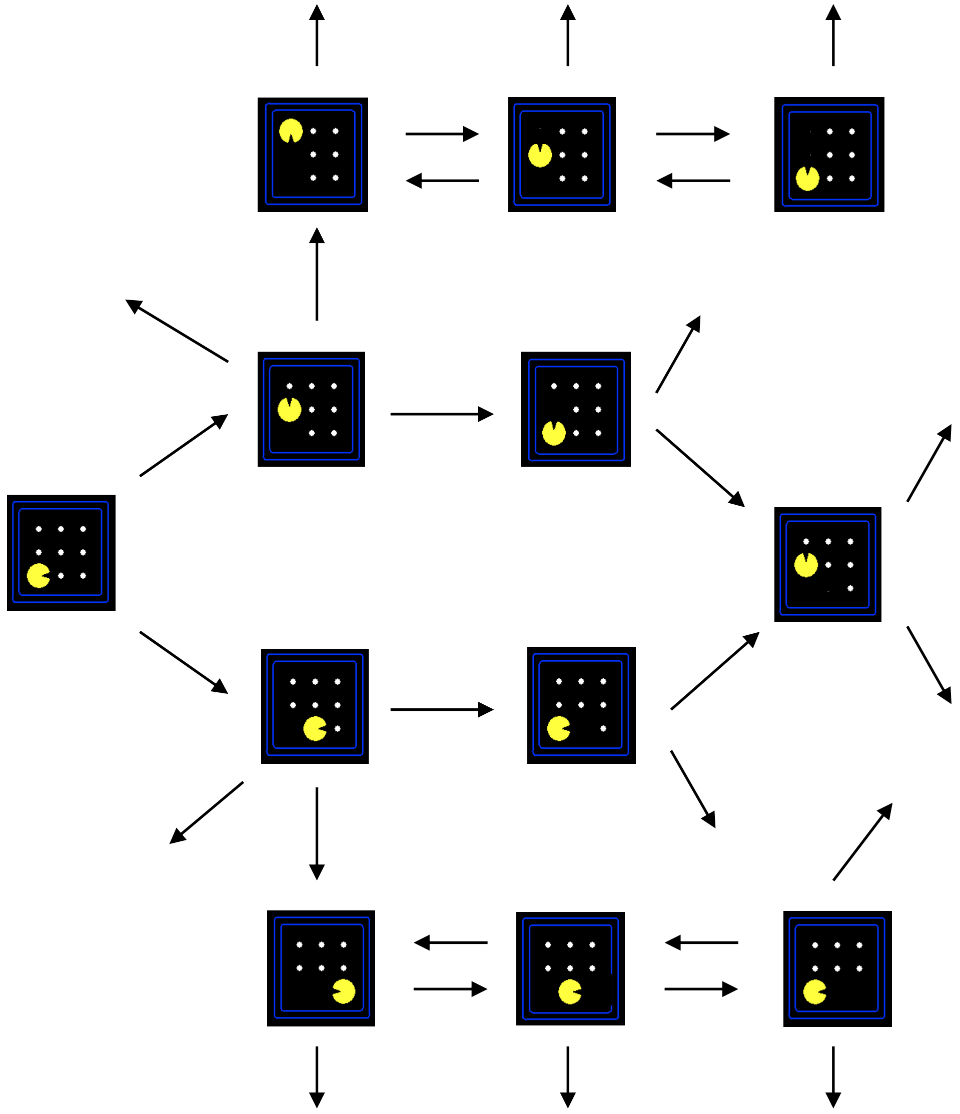
Állapottér-reprezentáció
Egyszerű keresési stratégiák

Állapottérgráfok és keresési fák



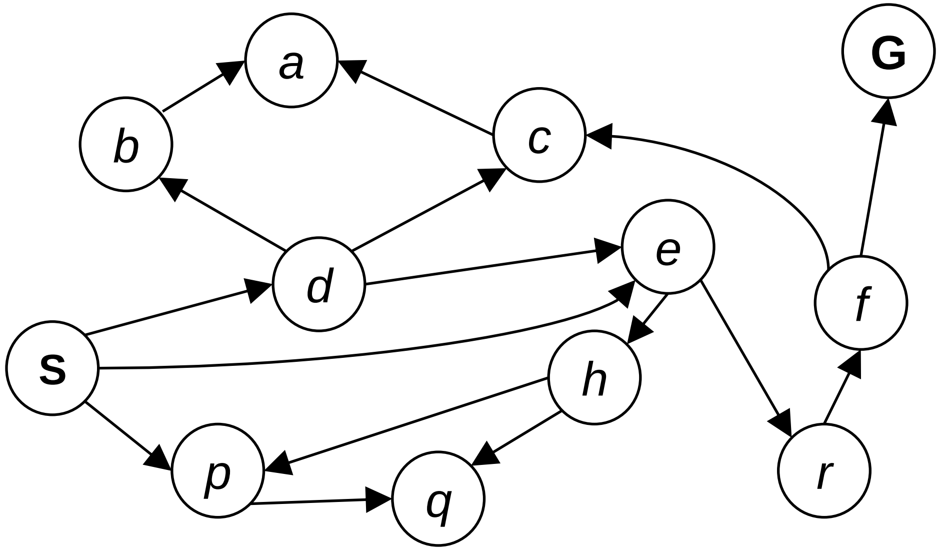
Állapottérgráfok

- Állapottérgráf: A keresési probléma matematikai reprezentációja
  - A csomópontok a világ állapotának absztrakt reprezentációi
  - Az élek az állapotátmenetet jelölik (cselekvések következményeit)
  - A célteszt a célállapot(ok) elérését vizsgálja
- Az állapottérgráfban minden állapot egyszer fordul elő
- Ritkán lehet ezt a gráfot teljes egészében felépíteni a memóriában (tárhelykorlátok miatt)



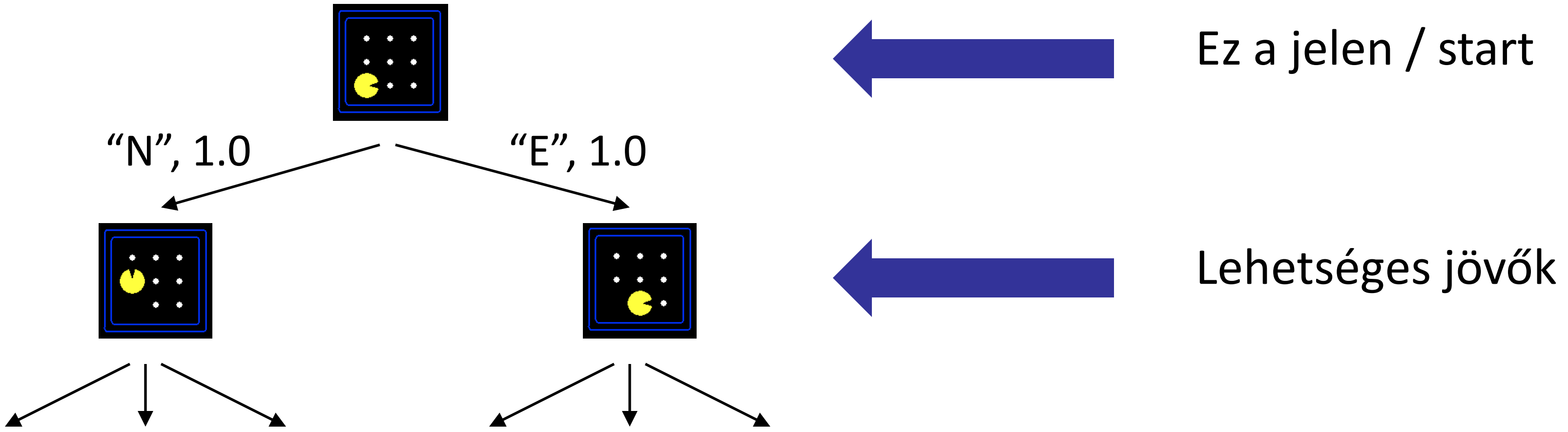
Állapottérgráfok

- Állapottérgráf: A keresési probléma matematikai reprezentációja
  - A csomópontok a világ állapotának absztrakt reprezentációi
  - Az élek az állapotátmenetet jelölik (cselekvések következményeit)
  - A célteszt a célállapot(ok) elérését vizsgálja
- Az állapottérgráfban minden állapot egyszer fordul elő
- Ritkán lehet ezt a gráfot teljes egészében felépíteni a memóriában (tárhelykorlátok miatt)



Kis állapottérgráf egy kis méretű keresési problémához

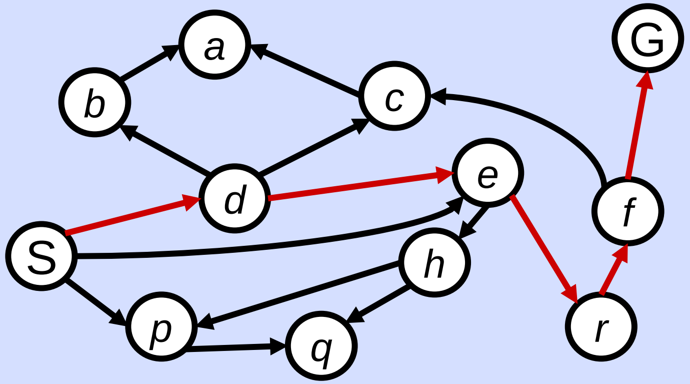
Keresési fák



- A keresési fa:
  - Egy "Mi lenne ha" fa tervekből és azok következményeiből
  - A kezdőállapot a gyökércsomópont
  - A gyermekcsomópontok a követő állapotoknak felelnek meg
  - A csomópontok állapotokat jelölnek és olyan tervekhez tartoznak, melyek elérik/tartalmazzák azokat az állapotokat.
  - Legtöbb probléma esetén nem lehet ténylegesen felépíteni a teljes keresési fát

Állapottérgráfok és keresési fák

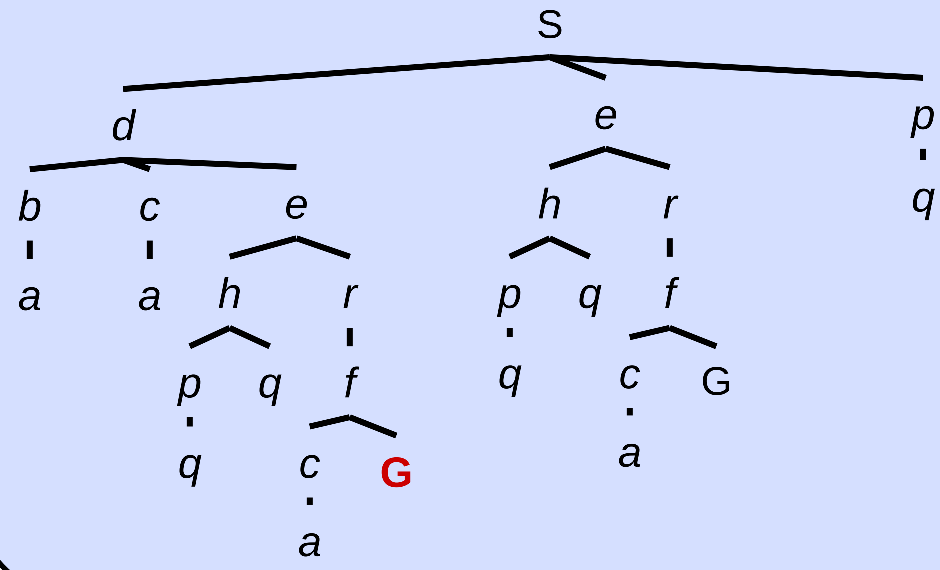
Állapottérgráf



Minden egyes csomópont a keresési fában egy útvonalnak felel meg az állapottér-gráfban.

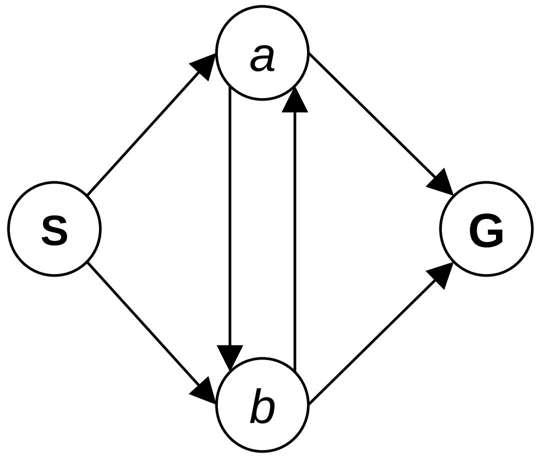
Mindkettőt igény szerint építjük, olyan kis mértékben, amennyire csak lehetséges

Keresési fa



Kvíz: állapottérgráf vs. keresési fa

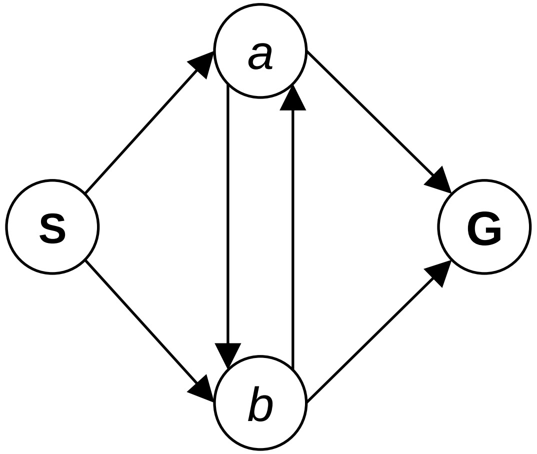
Tekintsük egy 4 állapotú gráfot:



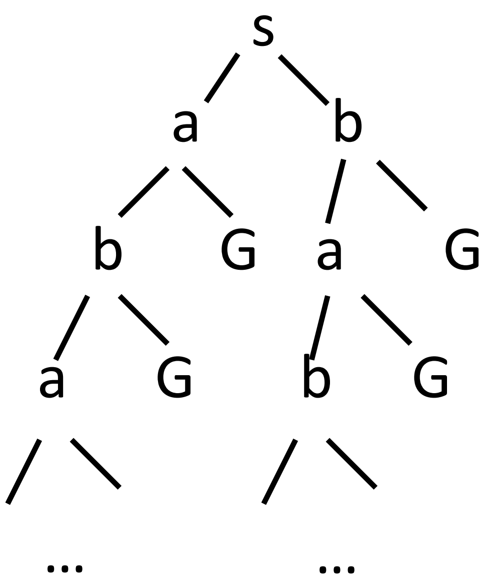
Mekkora az ehhez tartozó keresési fa (S-ből indulva)?

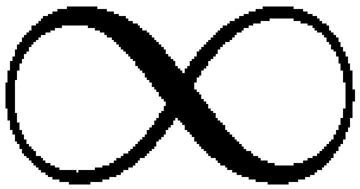
Kvíz: állapottérgráf vs. keresési fa

Tekintsük egy 4 állapotú gráfot:



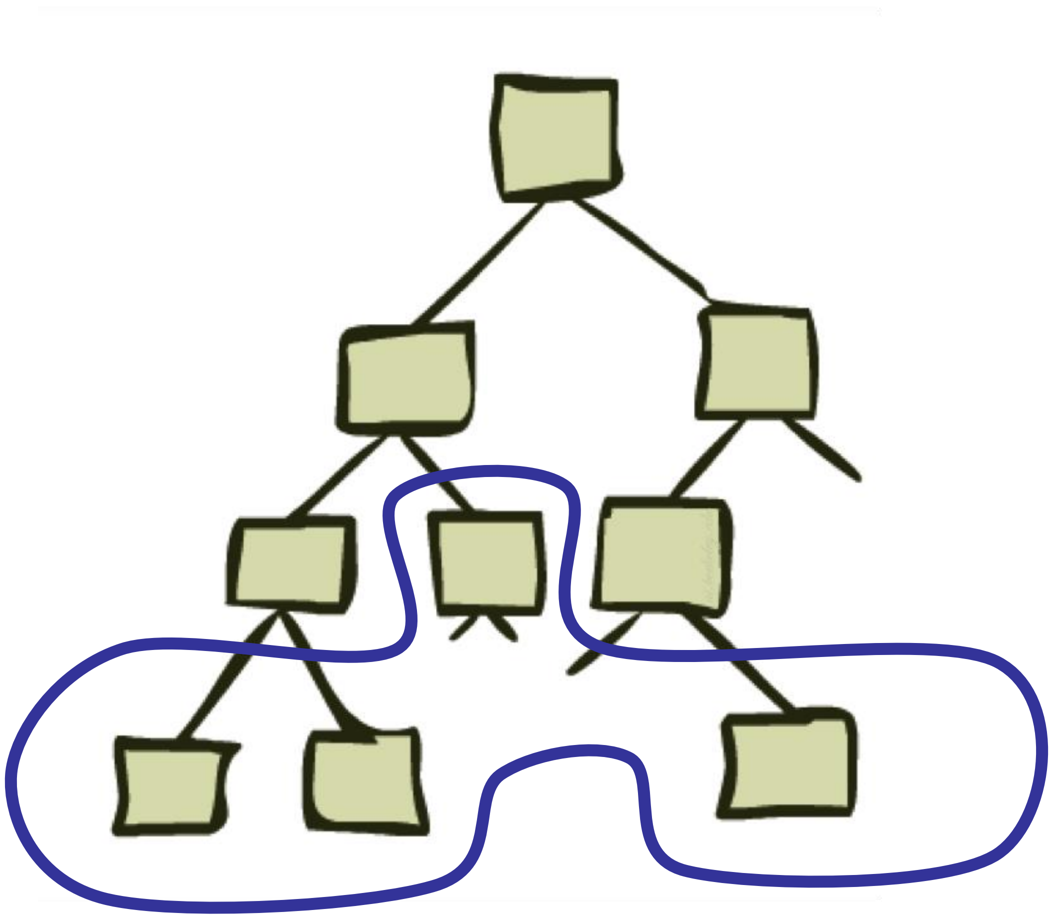
Mekkora az ehhez tartozó keresési fa (S-ből indulva)?



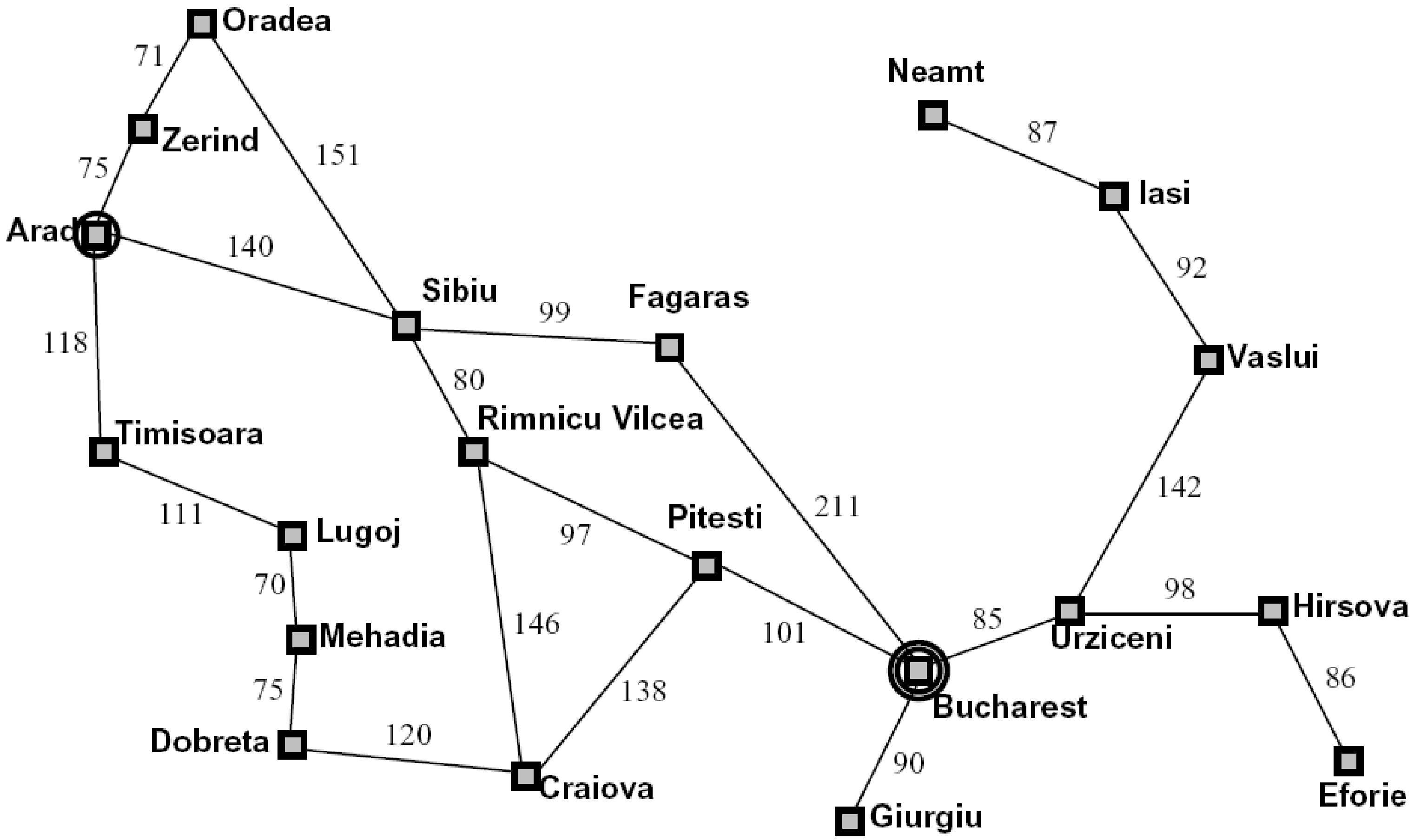


Általában: sok lehet az ismétlődő elem a keresési fában

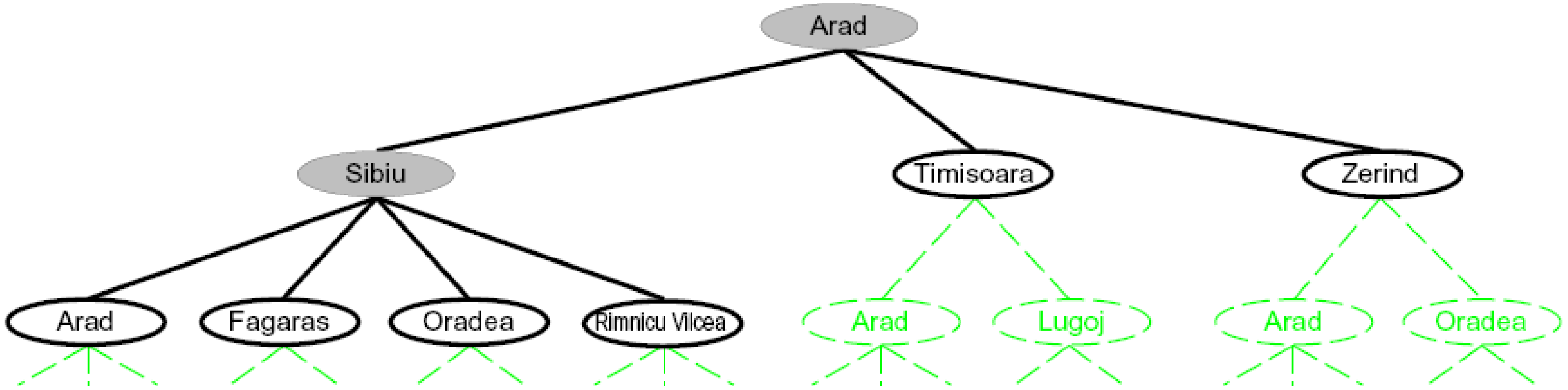
Keresési fa



Keresési példa: útvonaltervezés



Keresés keresési fával



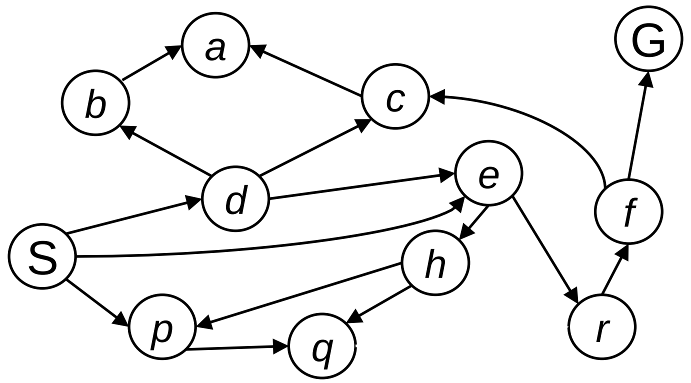
- Keresés:
  - Lehetséges tervek (fa csomópontok) kifejtése
  - „Perem” létrehozása a figyelembe vehető résztervek nyilvántartására
  - Minél kevesebb csomópont kifejtése

Általános fa alapú keresés

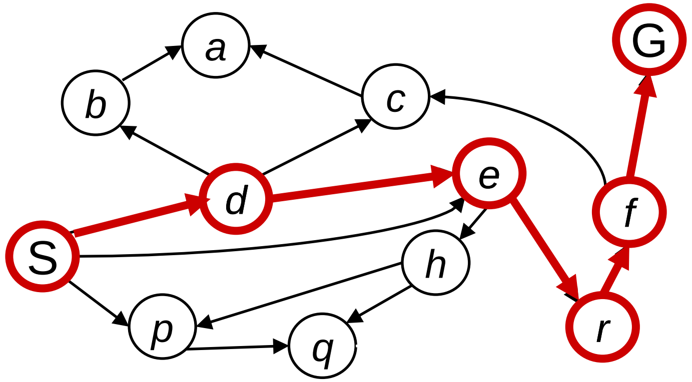
```
function FA-KERESÉS (probléma, stratégia) returns megoldás vagy kudarc
        Keresési fa inicializálása a probléma kiinduló állapotával
        loop do
                If nincs jelölt a kifejtéshez then return kudarc
                Kifejtendő levél csomópont kijelölése a stratégia alapján
                If a csomópont tartalmaz egy célállapotot then return megoldás
                else csomópont kifejtése és az eredmény csomópontok hozzáadása a keresési fához
        end
```

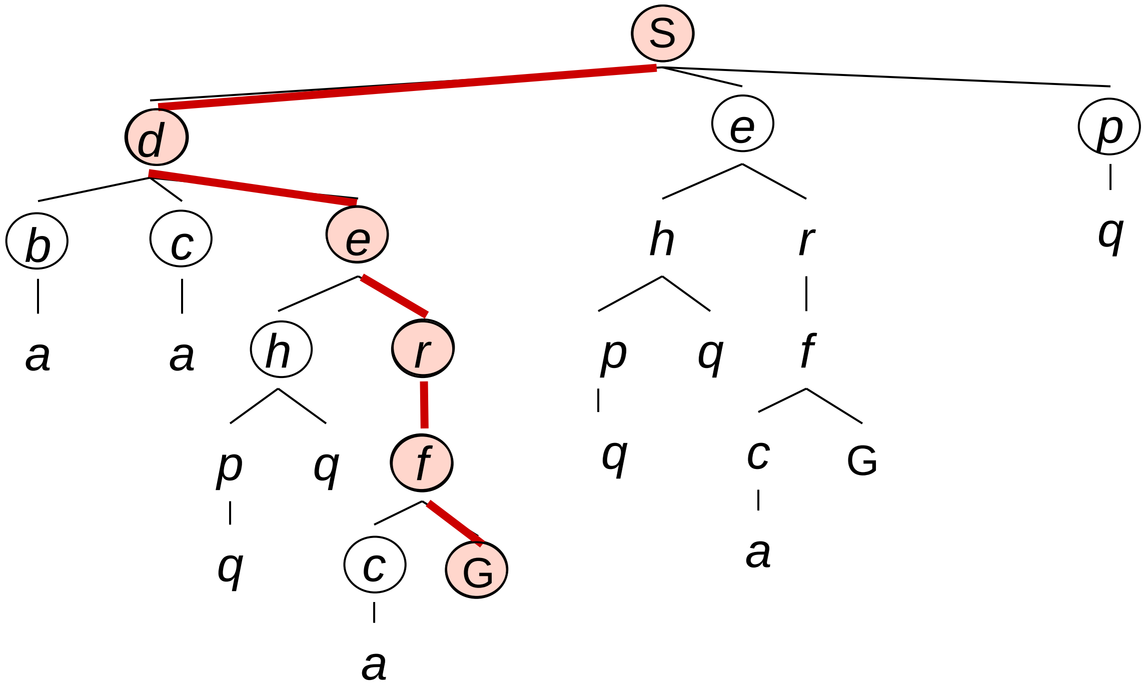
- Lényeges elemek:
  - Perem
  - Kifejtés
  - Felfedezési stratégia
- Központi kérdés: melyik perembeli csomópontot fejtsük ki?

Példa: Fa keresés



Példa: Fa keresés





~~s~~

~~s → d~~

s → e

s → p

s → d → b

s → d → c

~~s → d → e~~

s → d → e → h

~~s → d → e → r~~

~~s → d → e → r → f~~

s → d → e → r → f → c

~~s → d → e → r → f → G~~

Keresési stratégiák – avagy miből gazdálkodhatunk?

- Mire vagyunk képesek? (saját modell)
- Milyen körülöttünk a környezet? (cél-, távolsági modellek)

Nem informált keresések (**gyenge** vagy **vak** keresések)

a. tudjuk: hogy néz ki a célállapot

b. egyáltalán nem: milyen költségű az aktuálisból a célállapotba vezető út

Informált keresések (heurisztikus keresések)

a. tudjuk: hogy néz ki a célállapot

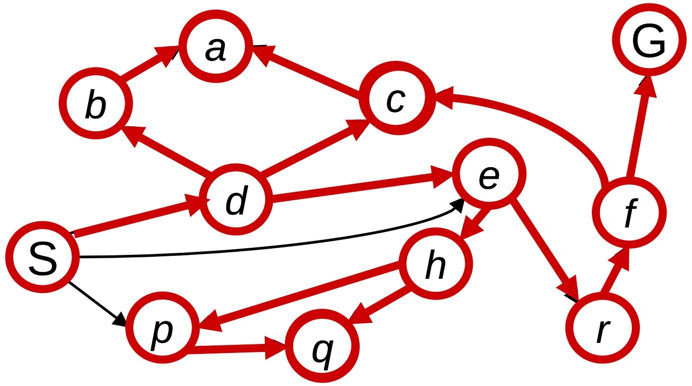
b. (jó) becslésünk van arra, hogy: milyen költségű lehet az aktuális állapotból a célállapotba vezető út

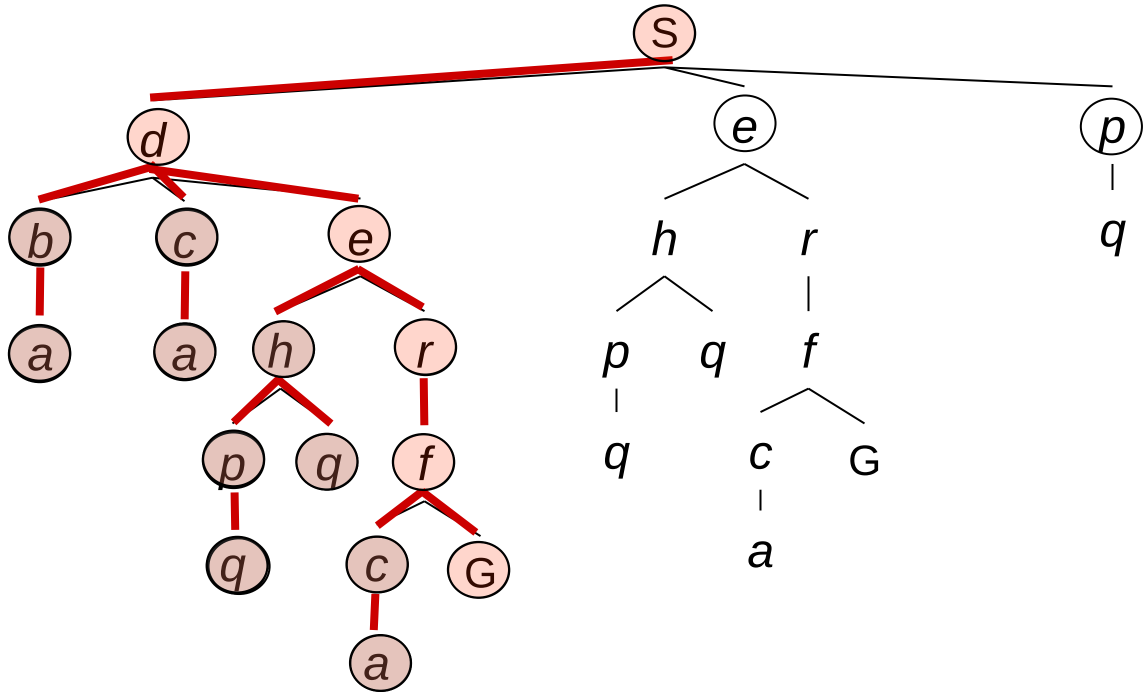
Mélységi keresés

Mélységi keresés

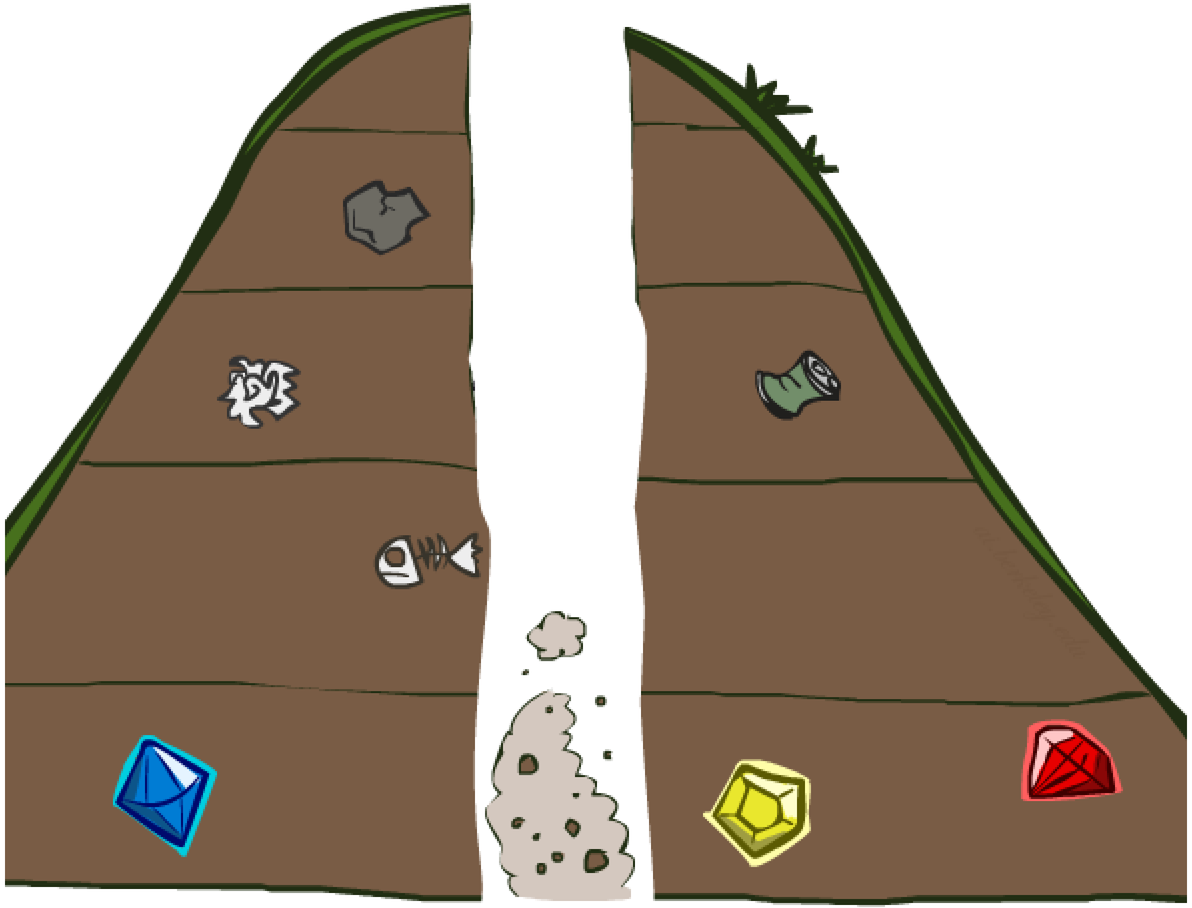
Stratégia: legmélyebb csomópont kifejtése

Megvalósítás: A perem egy LIFO stack





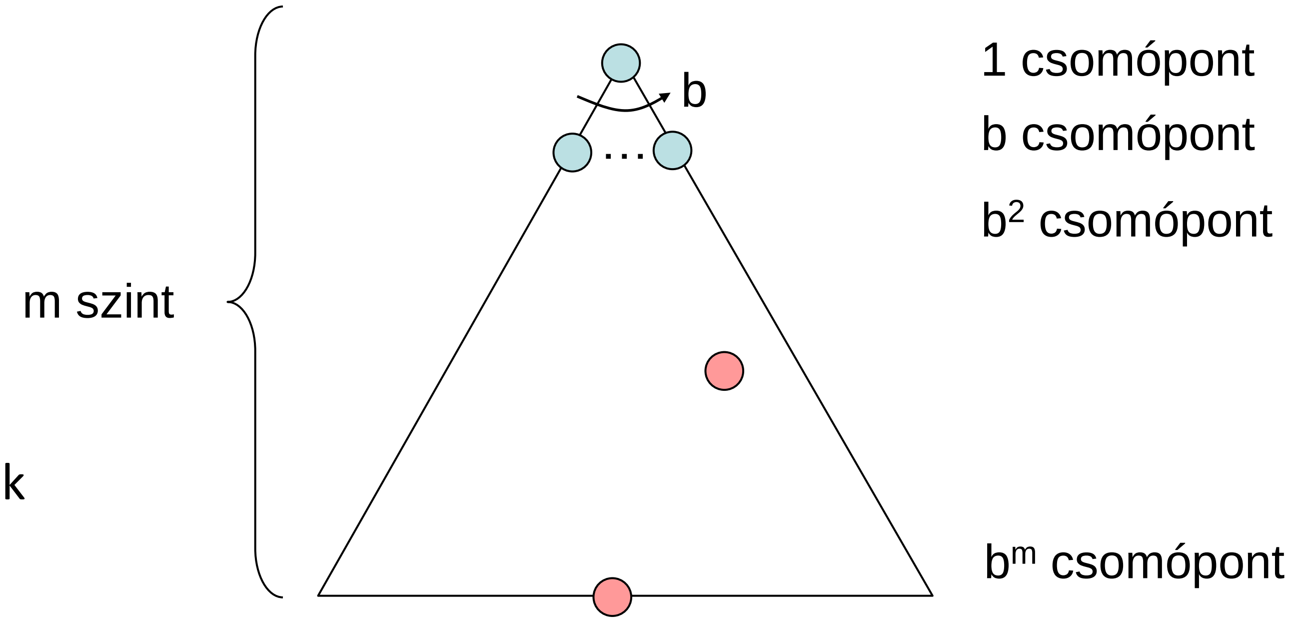
Keresési algoritmusok tulajdonságai



Keresési algoritmusok tulajdonságai

- Teljesség: Garantáltan megtalál-e egy megoldást, ha létezik egy?
- Optimalitás: Garantáltan megtalálja-e a legkisebb költségű utat?
- Időkomplexitás
- Tárkomplexitás

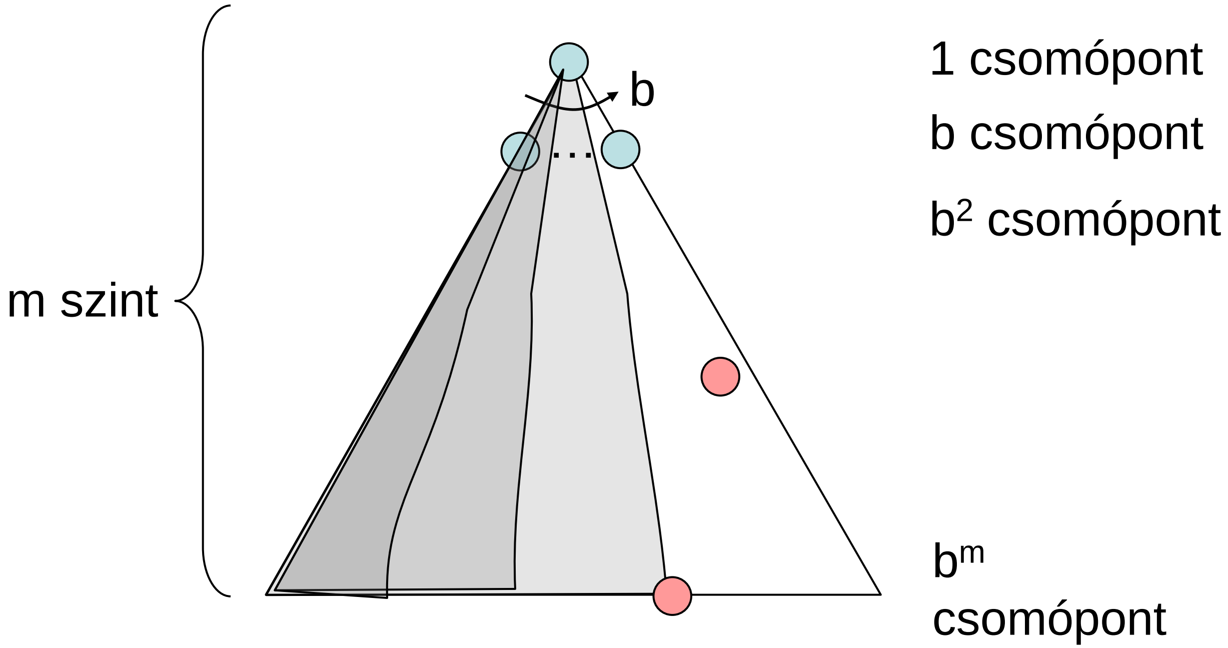
- Keresési fa váza:
  - **b** az elágazási faktor
  - **m** a maximum mélység
  - adott mélységben elérhető megoldások



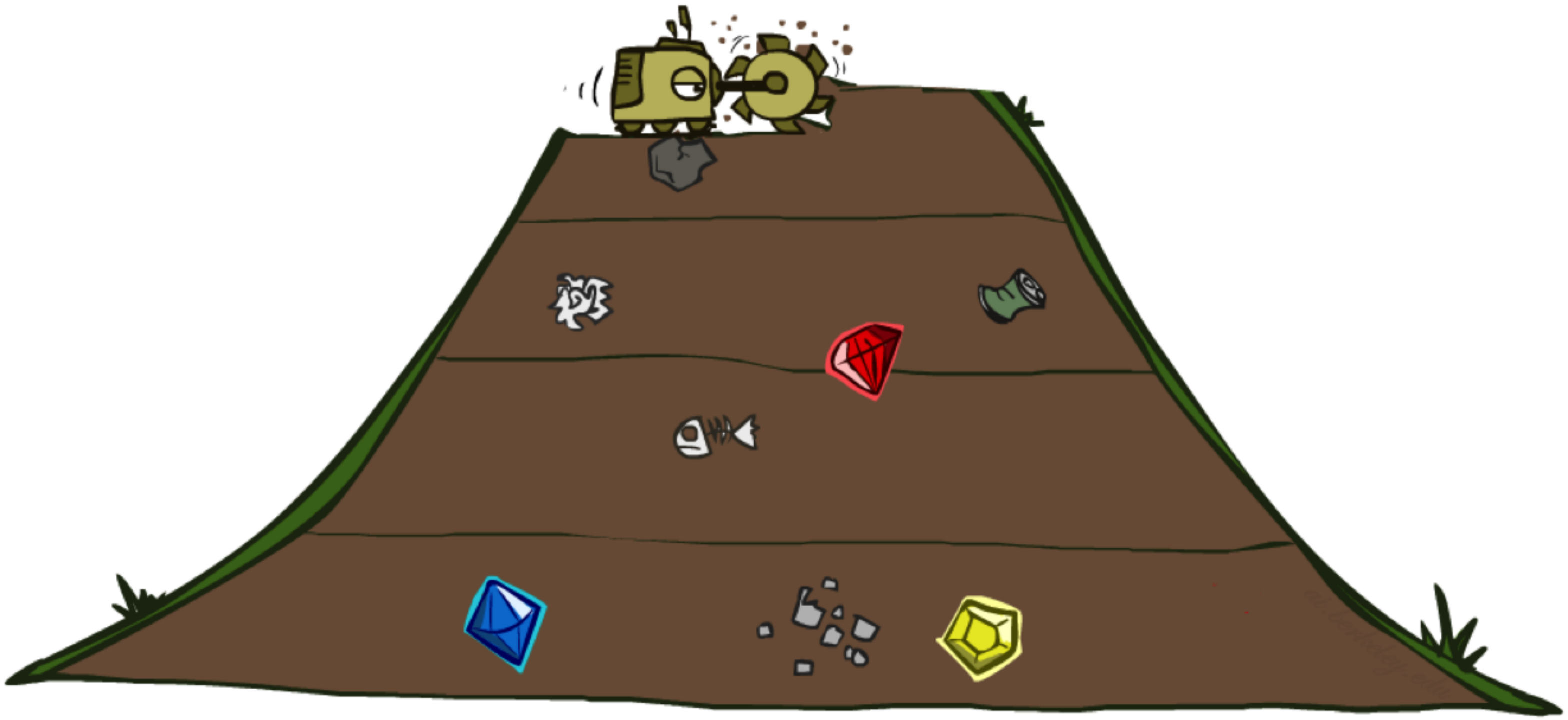
- A teljes fában lévő összes csomópont:
  - $1 + b + b^2 + \ldots. b^m = O(b^m)$

Mélységi keresés (DFS) jellemzői

- Milyen csomópontokat fejt ki a mélységi keresés?
  - A keresési fa egy bal oldali prefix részét fejti ki
  - A teljes fát feldolgozhatja
  - Ha **m** véges, $O(b^m)$ időt vesz igénybe
- Mennyi tárhelyet igényel a perem?
  - Csak a gyökércsomóponthoz vezető úton szerepelő elágazások számítanak: O(bm)
- Teljes-e?
  - **m** lehetne végtelen, így csak akkor lehet teljes, ha körök kialakulását megakadályozzuk
- Optimális-e?
  - Nem, megtalálja a legbaloldalibb megoldást, függetlenül a mélységtől és a költségtől



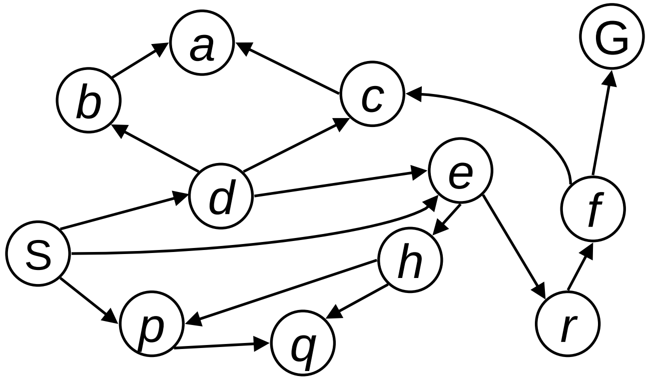
Szélességi keresés



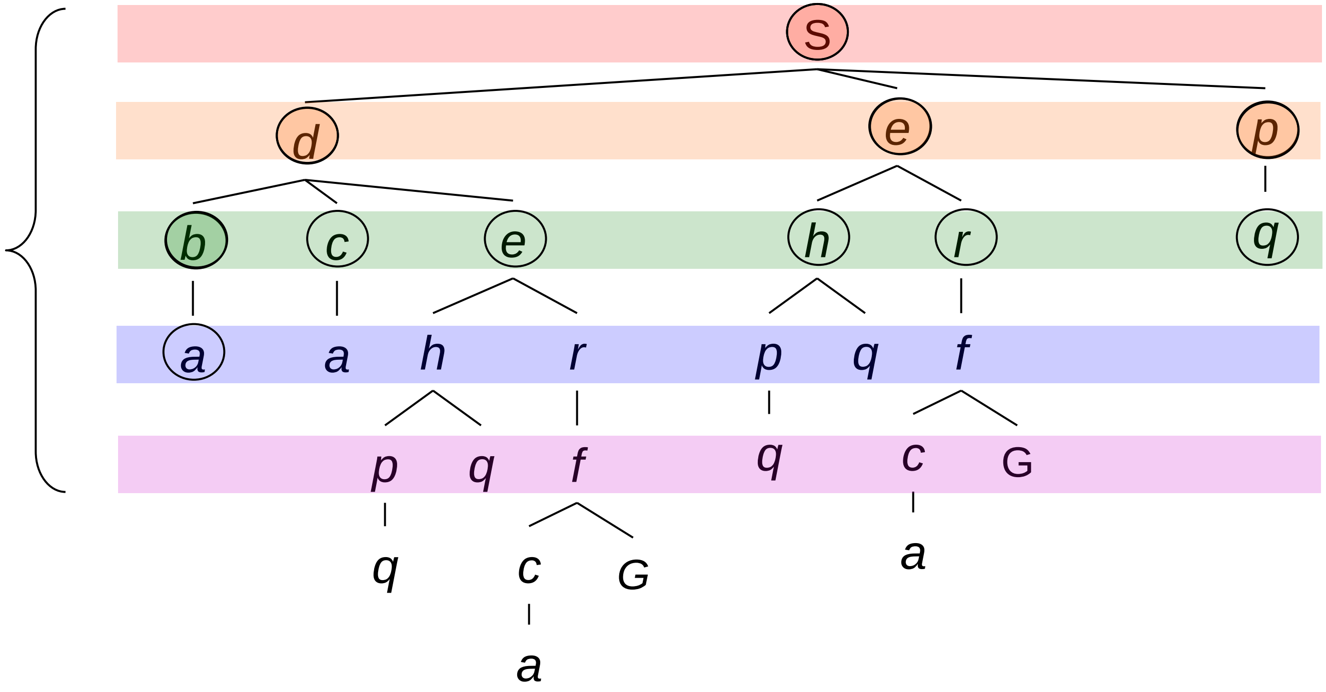
Szélességi keresés

Stratégia: a legkevésbé mélyen lévő csomópont kifejtése

Megvalósítás: A perem egy FIFO sor

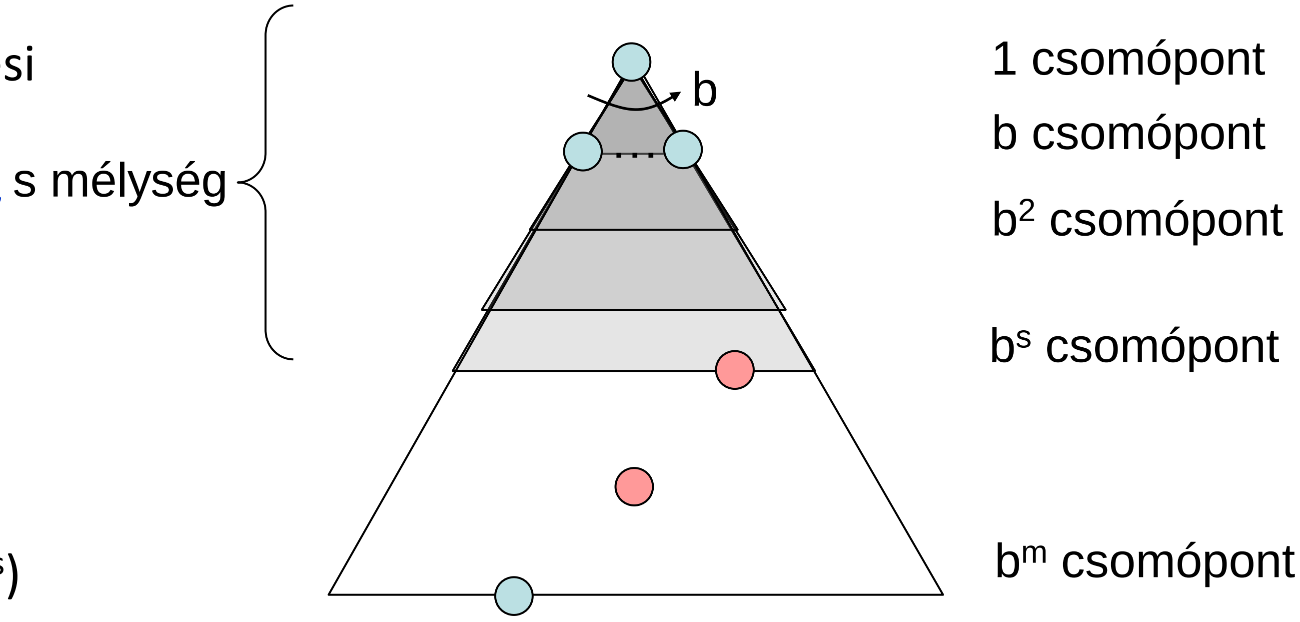


Keresési szintek

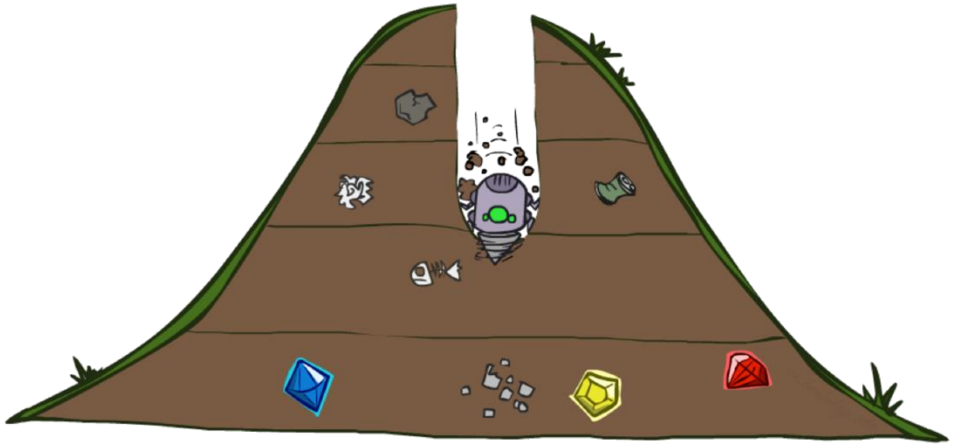


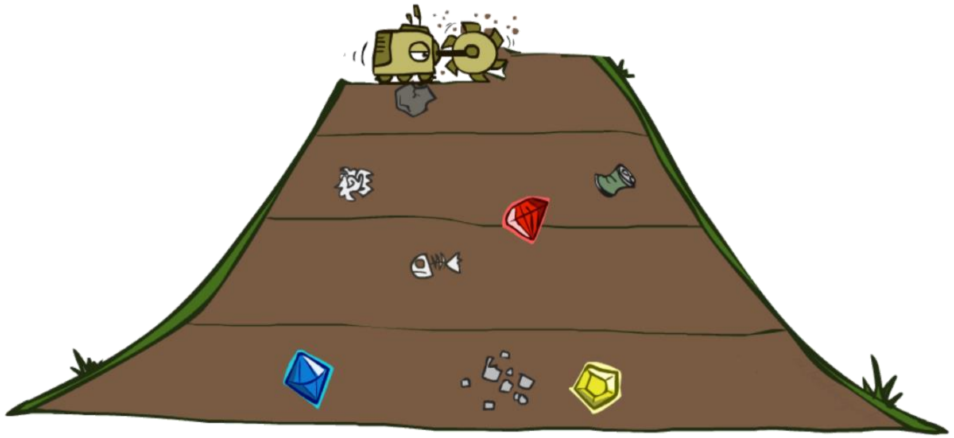
Szélességi keresés (BFS) tulajdonságai

- Milyen csomópontokat fejt ki a szélességi keresés?
  - Feldolgozza az összes csomópontot a keresési fában legsekélyebben fekvő megoldásig
  - Jelölje a legsekélyebb megoldás mélységét ***s***
  - A keresés ideje ekkor: $O(b^s)$
- Mekkora tárhelyet igényel a perem nyilván tartása?
  - Közelítőleg a legutolsó szinttel arányos: $O(b^s)$
- Teljes-e?
  - ***s*** véges, ha létezik megoldás, tehát igen!
- Optimális-e?
  - Csak akkor, ha minden egységesen 1 költségű



Melyik a jobb: a mélységi vagy a szélességi keresés?





Melyik a jobb: a mélységi vagy a szélességi keresés?

- Mikor múlja felül szélességi keresés a mélységit?

- Mikor múlja felül mélységi keresés a szélességit?

Mélységkorlátozott keresés

- Az utak maximális mélységére egy vágási korlátot ad.

- A mélységi levágásnál a keresés visszalép. A megoldást, amennyiben létezik és a mélységkorlátnál sekélyebben fekszik, garantáltan megtaláljuk.

- De semmi garancia nincs arra, hogy a legkisebb költségű (itt: legrövidebb út) megoldást találjuk meg.

- Amennyiben túl kis mélységkorlátot választunk, akkor a mélységkorlátozott keresés még csak teljes sem lesz.

Románia jelen egyszerűsített térképe: 20 város. Ha létezik egy megoldás, az maximálisan 19 lépés hosszú lehet.
Minden város bármelyik városból legfeljebb 9 lépésben elérhető:
az állapottér **átmérője** = jobb mélységkorlát.

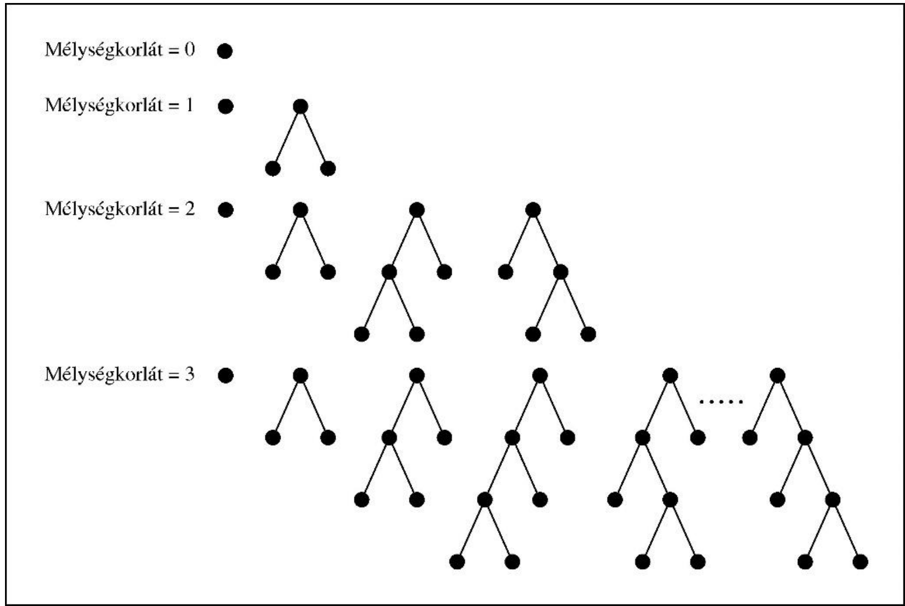
Iteratívan mélyülő keresés

Slate és Atkin (1977): CHESS 4.5 sakkprogram.
- mélységkorlátozott keresés - egy jó mélységkorlát megválasztása?

A legtöbb esetben azonban mindaddig nem tudunk jó mélységkorlátot adni, amíg meg nem oldottuk a problémát.

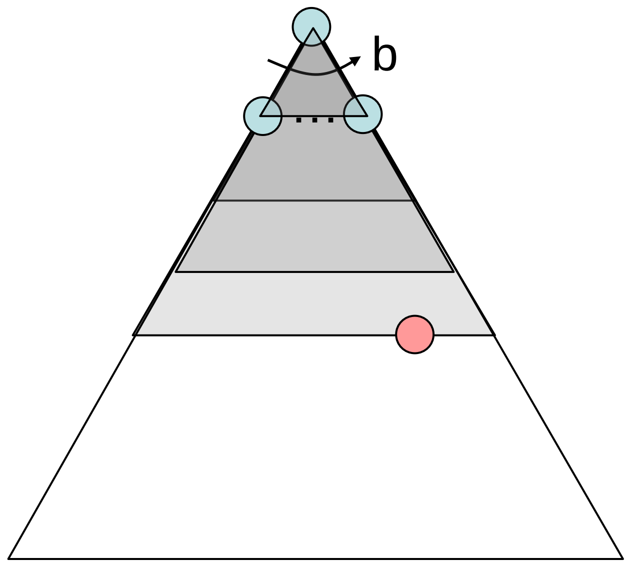
A legjobb mélységkorlát kiválasztása:

- kipróbálja az összes lehetséges mélységkorlátot: először 0, majd 1, majd 2, stb.
- mélységkorláttal végez mélységkorlátozott keresést.



Iteratívan mélyülő keresés

- Ötlet: egyesítsük a mélységi keresés relatíve alacsony tárhelyigényét a szélességi keresés relatíve alacsony időigényével
  (legsekélyebben fekvő megoldás megtalálása)
  - Futtassuk a mélységi keresést 1-es mélységi korláttal. Ha nincs megoldás …
  - Futtassuk a mélységi keresést 2-es mélységi korláttal. Ha nincs megoldás …
  - Futtassuk a mélységi keresést 3-es mélységi korláttal. Ha nincs megoldás …



- Nem pazarlóan redundáns ez?
  - A legtöbb „munka” a legalsó keresési szinten van, tehát nem olyan rossz a helyzet!

Iteratívan mélyülő keresés gyakorlatilag ötvözi a szélességi és mélységi keresés előnyös tulajdonságait

- A szélességi kereséshez hasonlóan **optimális** és **teljes**, de csak
- a mélységi keresés **szerény** memória igényével rendelkezik.
- De bizonyos állapotokat az algoritmus többször is kifejt!

Tékozló? - egy exponenciális keresési fában majdnem az összes csomópont a legmélyebb szinten található

d mélységben a megoldás, *b* elágazási tényező
szélességi keresésnél a kifejtések száma:

$1 + b + b^2 + ... + b^{d-2} + b^{d-1} + b^d$

az iteratívan mélyülő keresésnél a kifejtések teljes száma:

$(d+1)1 + (d)b + (d-1)b^2 + ... + 3b^{d-2} + 2b^{d-1} + 1b^d$

Iteratívan mélyülő keresés

- szélességi keresésnél a kifejtések száma:

$1 + b + b^2 + ... + b^{d-2} + b^{d-1} + b^d$

pl. b =10 és d = 5 esetén ez a szám: 1 + 10 + 100 + ... + 100000 = 111111

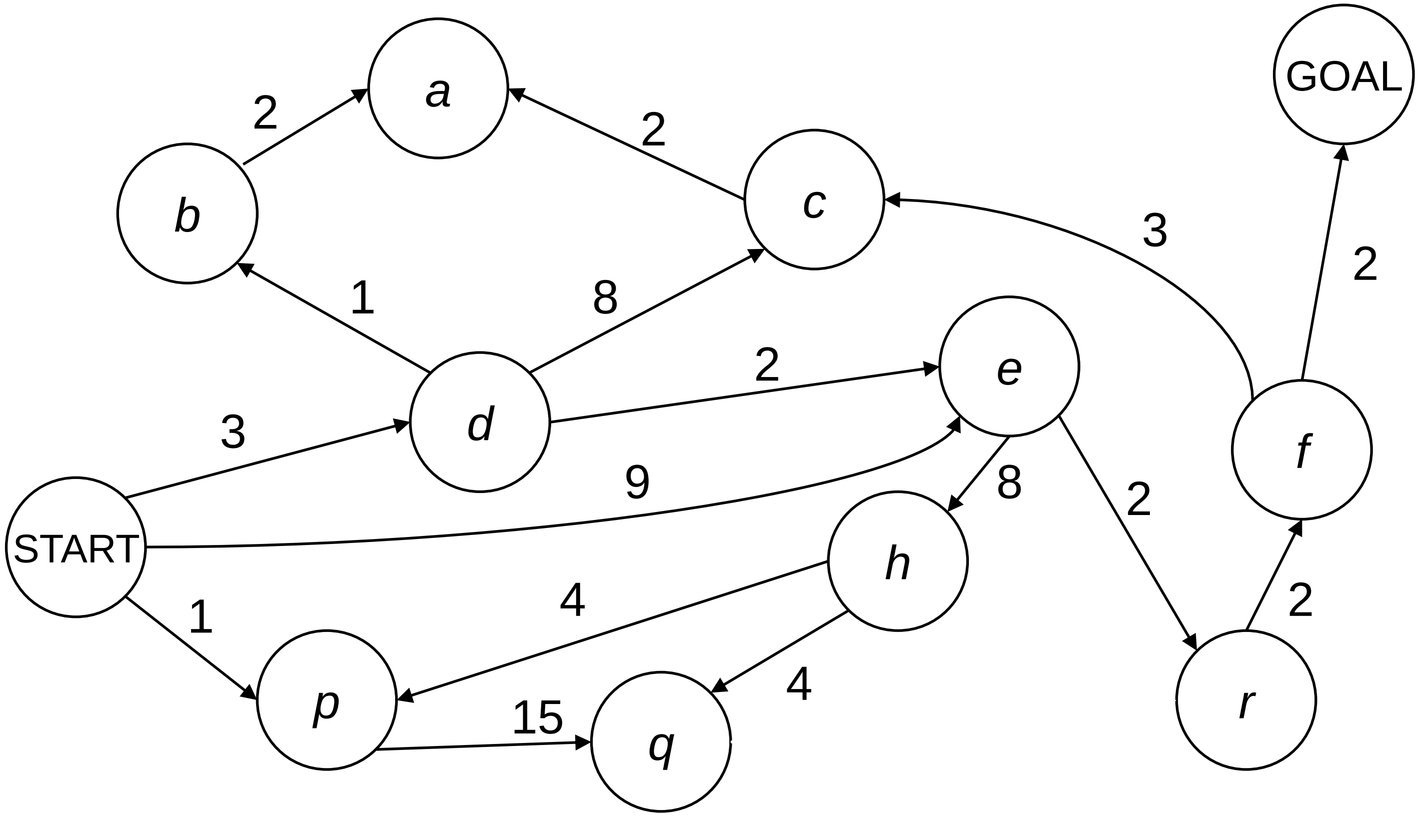
- az iteratívan mélyülő keresésnél a kifejtések teljes száma:

$(d+1)1 + (d)b + (d-1)b^2 + ... + 3b^{d-2} + 2b^{d-1} + 1b^d$

b = 10 és d = 5 esetén: 6 + 50 +400 + ... + **100.000** = **123.456** **(+23,5%)**

- minél nagyobb az elágazási tényező, annál kisebb a többletmunka (max. b = 2, kb. **200%**, kb. *kétszerese* a szélességinek!)

Költségérzékeny keresés



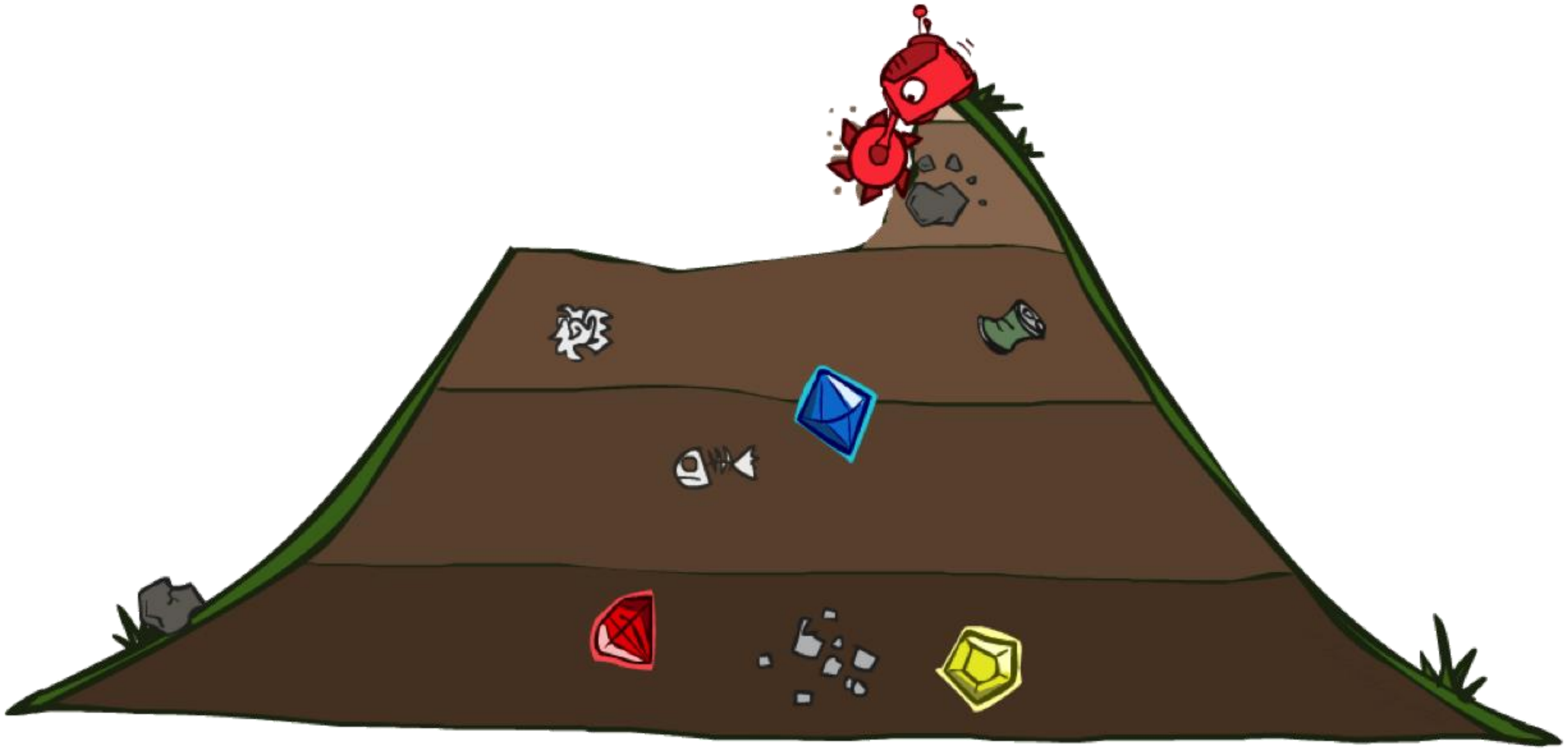
A szélességi keresés megtalálja a cselekvések száma tekintetében legrövidebb utat.
De nem találja meg a legkisebb költségű utat.
Tekintsünk egy hasonló algoritmust, ami viszont megtalálja a legkisebb költségű utat.

Mesterséges intelligencia

Problémamegoldás kereséssel

Költségérzékeny keresés

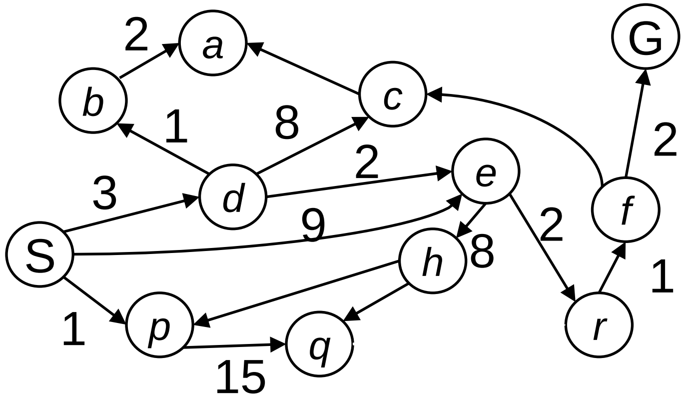
Egyenletes költségű keresés

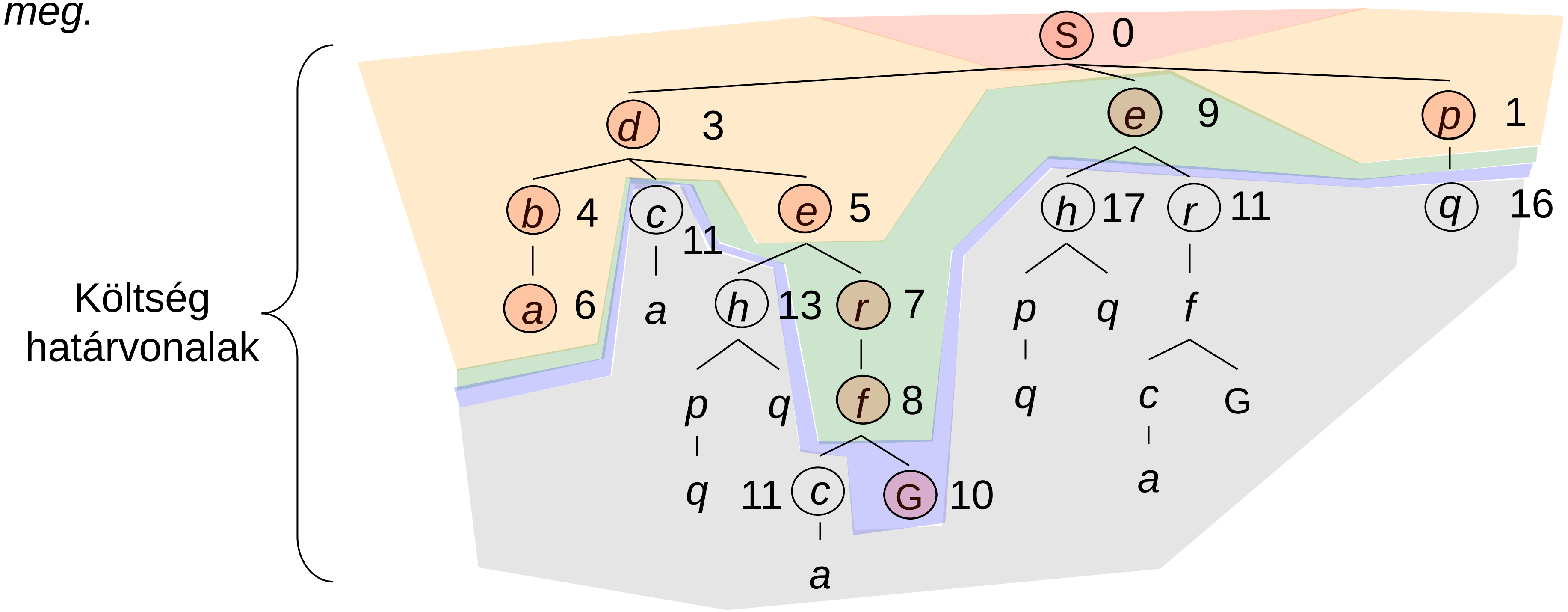


Egyenletes költségű keresés (UCS)

Stratégia: a legkisebb költségű csomópont kifejtése

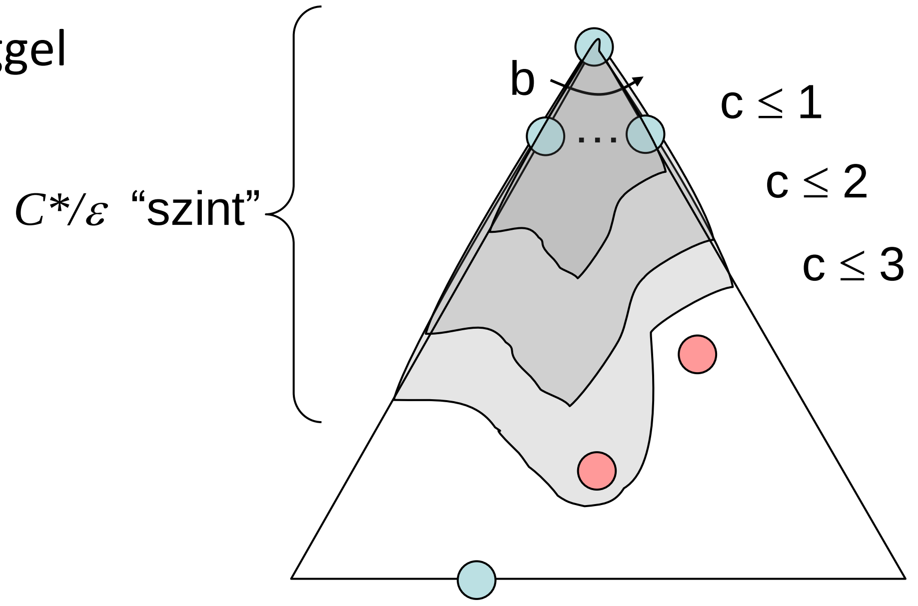
A perem egy prioritásos sor, ahol a prioritás a kumulált költségnek felel meg.





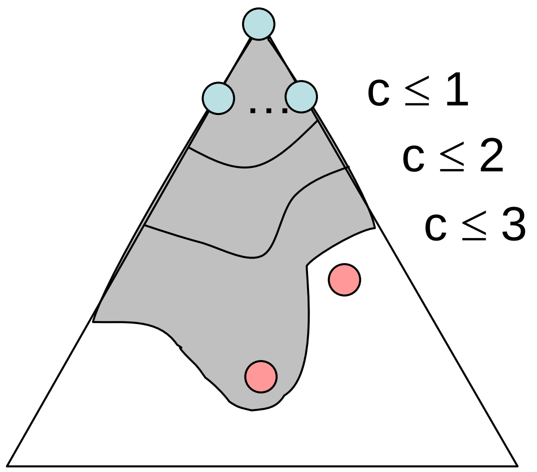
Egyenletes költségű keresés jellemzői

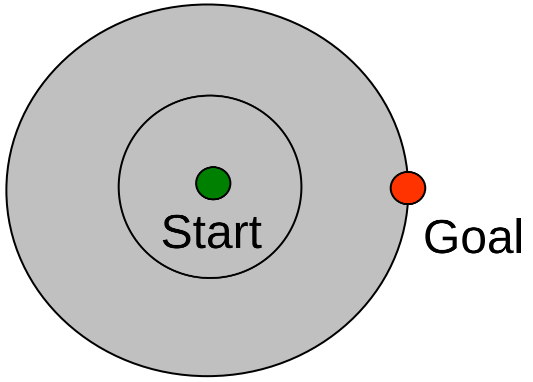
- Melyik csomópontokat fejti ki az egyenletes költségű keresés?
  - Kifejti az összes olyan csomópontot, ami kevesebb költséggel elérhető, mint a legkisebb költségű megoldás
  - Ha a megoldás C^* költségű és az élek legalább ε költségűek, akkor az effektív mélység közelítőleg: C^*/ε
  - A szükséges idő O(b$^{C^*/\varepsilon}$)
    (az effektív mélységben exponenciális)
- Mennyi tárhelyet igényel a perem?
  - Közelítőleg az utolsó szinttel arányos O(b$^{C^*/\varepsilon}$)
- Teljes-e?
  - Feltéve, hogy a legjobb megoldás véges költségű és a minimum élköltség pozitív, igen!
- Optimális-e?
  - Igen! (Bizonyítás később)



Egyenletes költségű keresés problémái

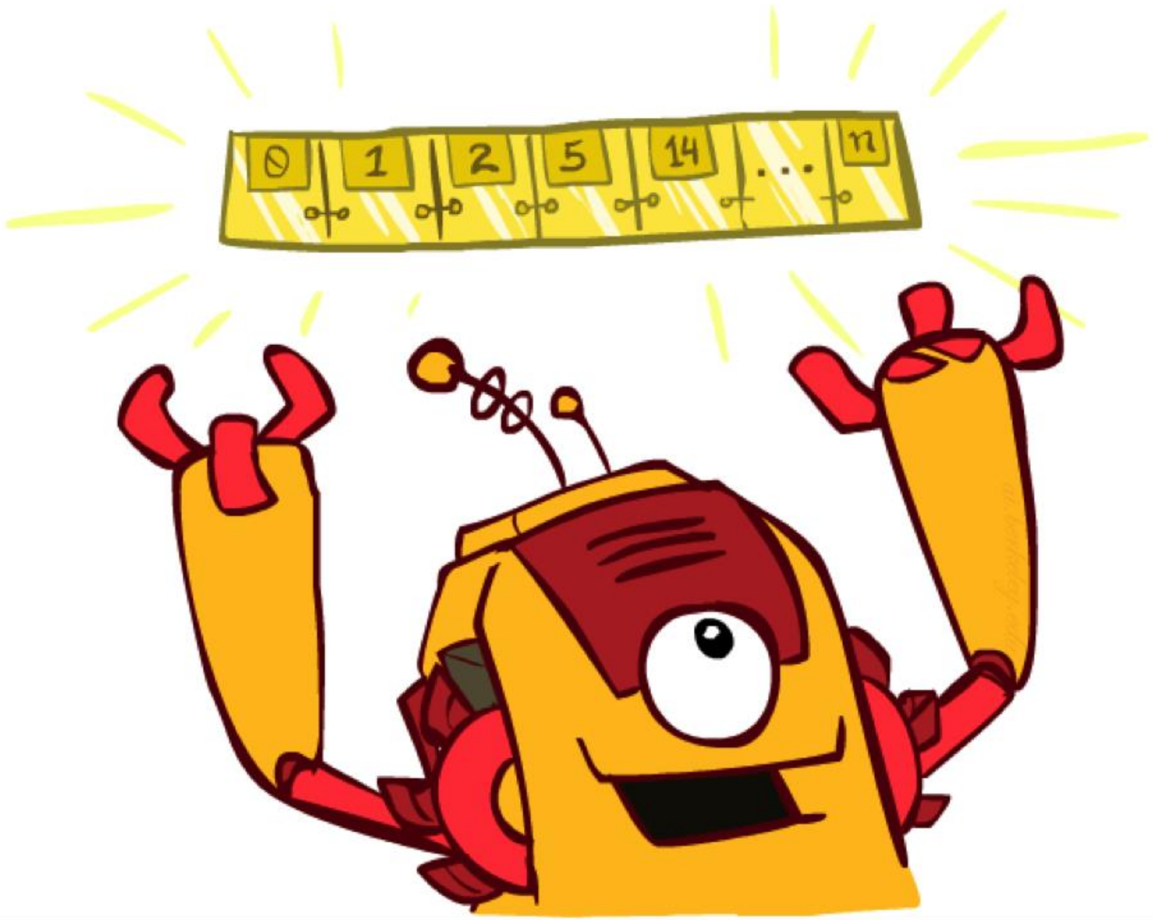
- Egyre növekvő költségű kontúrokat fejt ki
- A jó hír: teljes és optimális!
- A rossz hír:
  - Minden „irányban” feltár megoldási opciókat
  - Nem rendelkezik információval a célállapot helyzetéről





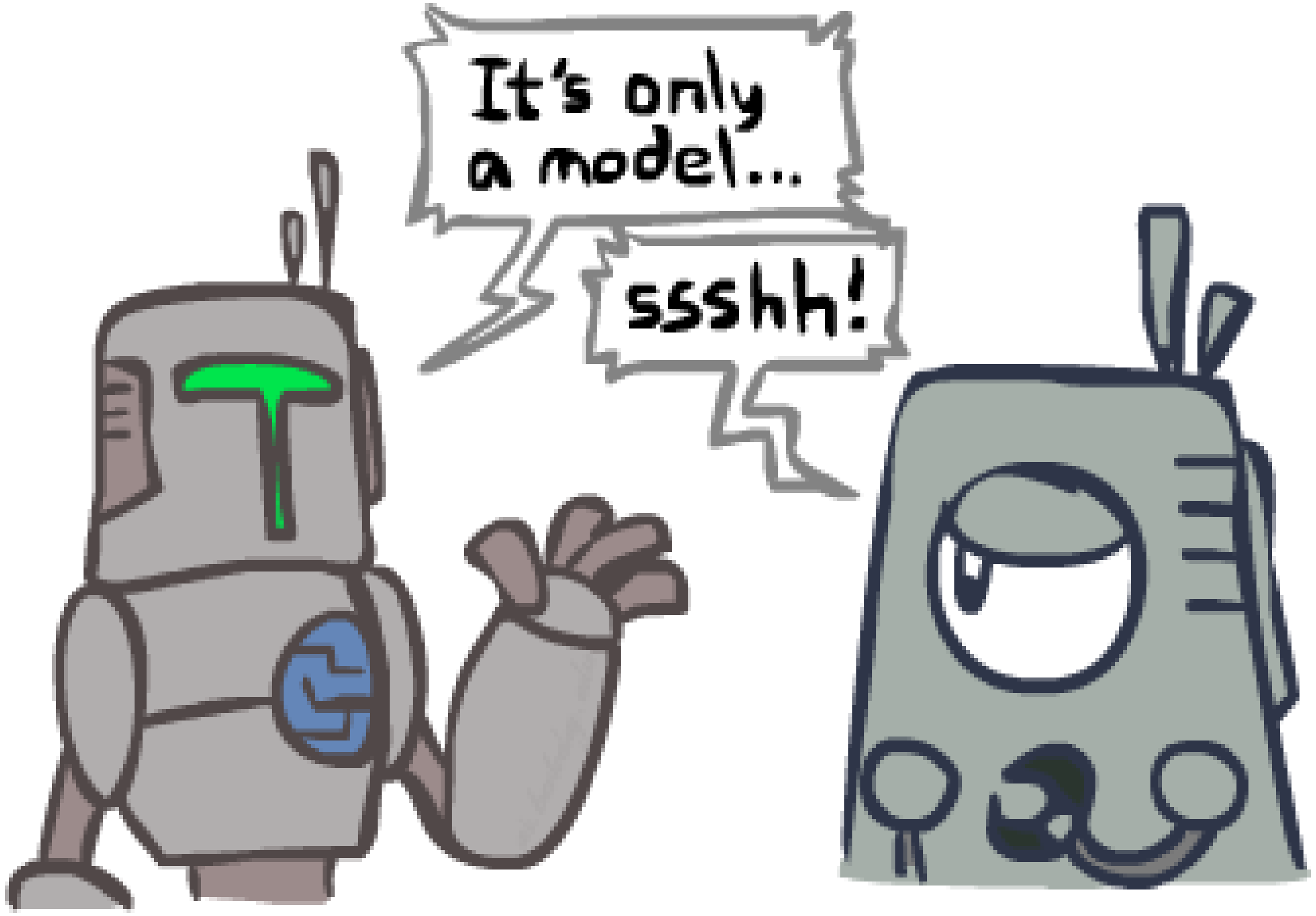
Közös strukturális elem: a sor (queue)

- A bemutatott keresési algoritmusok a perem nyilvántartási stratégiát leszámítva azonosak
  - Lényegében minden perem egy prioritásos sor (vagyis prioritással rendelkező csomópontok gyűjteménye)
  - Gyakorlatban, a mélységi és a szélességi keresés esetén megspórolható a prioritásos sor log(n) számítási igénye, ha (rendre) vermet (stack) és egy egyszerű sort (queue) alkalmazunk
  - Tehát létrehozható egy keretrendszer a keresési algoritmusok megvalósítására, amiben csak a sor megvalósítása változik

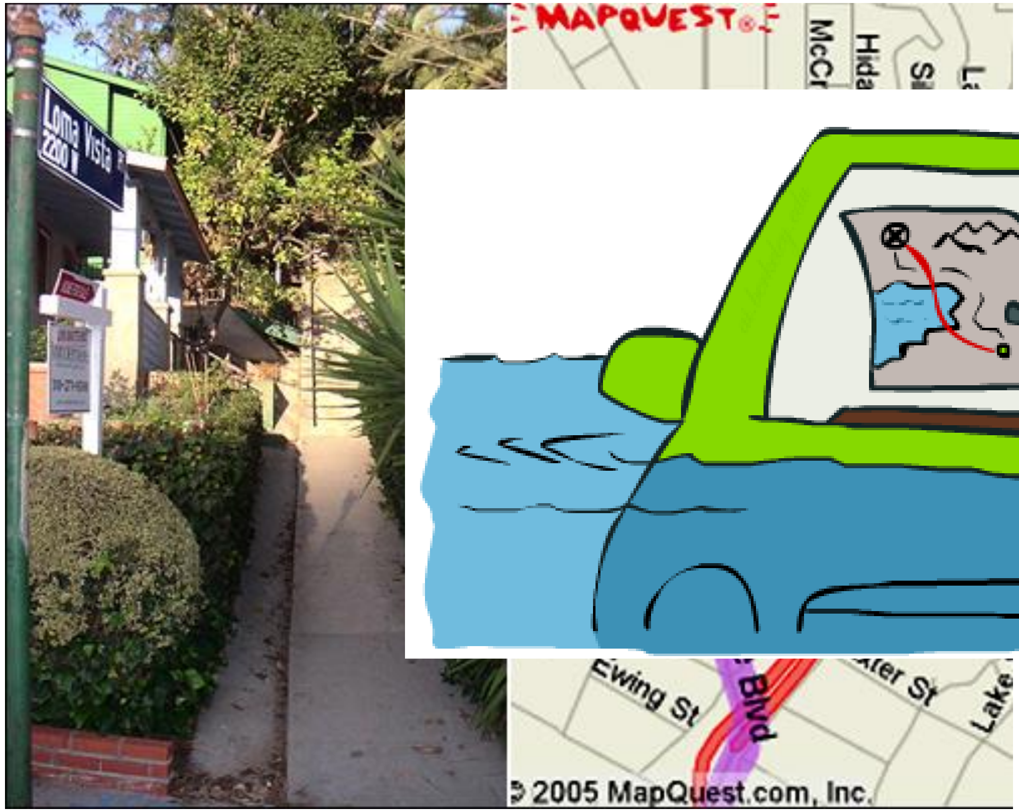


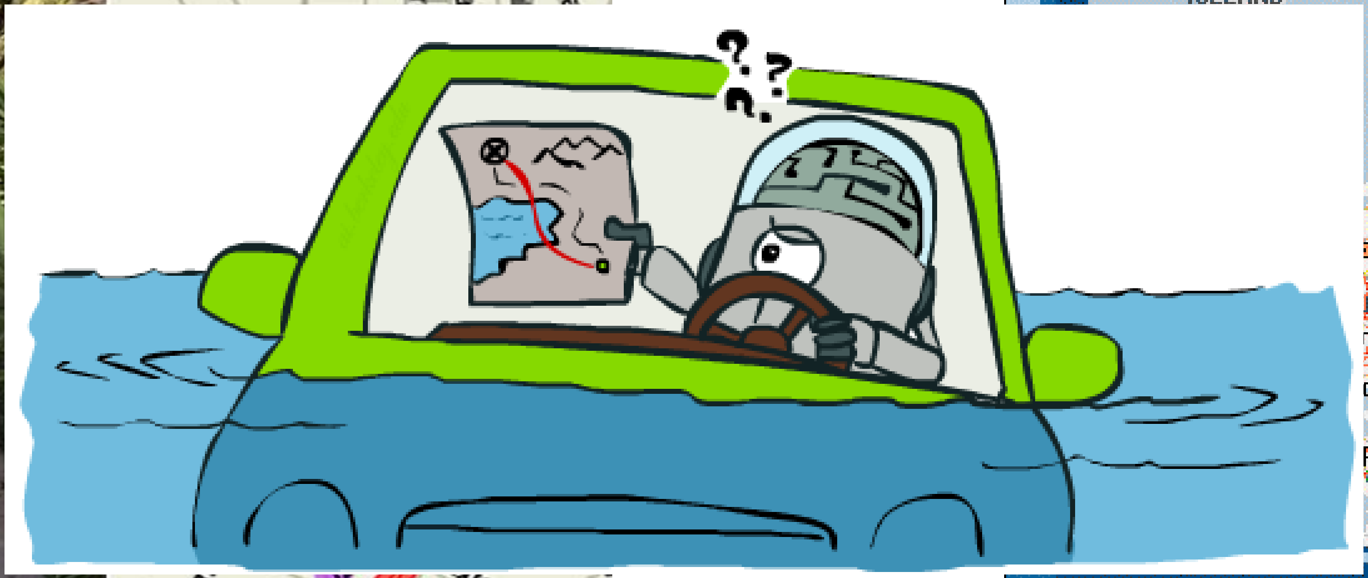
Keresés és a világ modelljei

- A keresés a világ egy modellje alapján történik
  - Az ágens nem próbálja ki az összes lehetséges tervet a valós világban
  - A tervezés „szimulációban történik”
  - A keresés csak annyira lehet jó, mint az alapjául szolgáló modellek

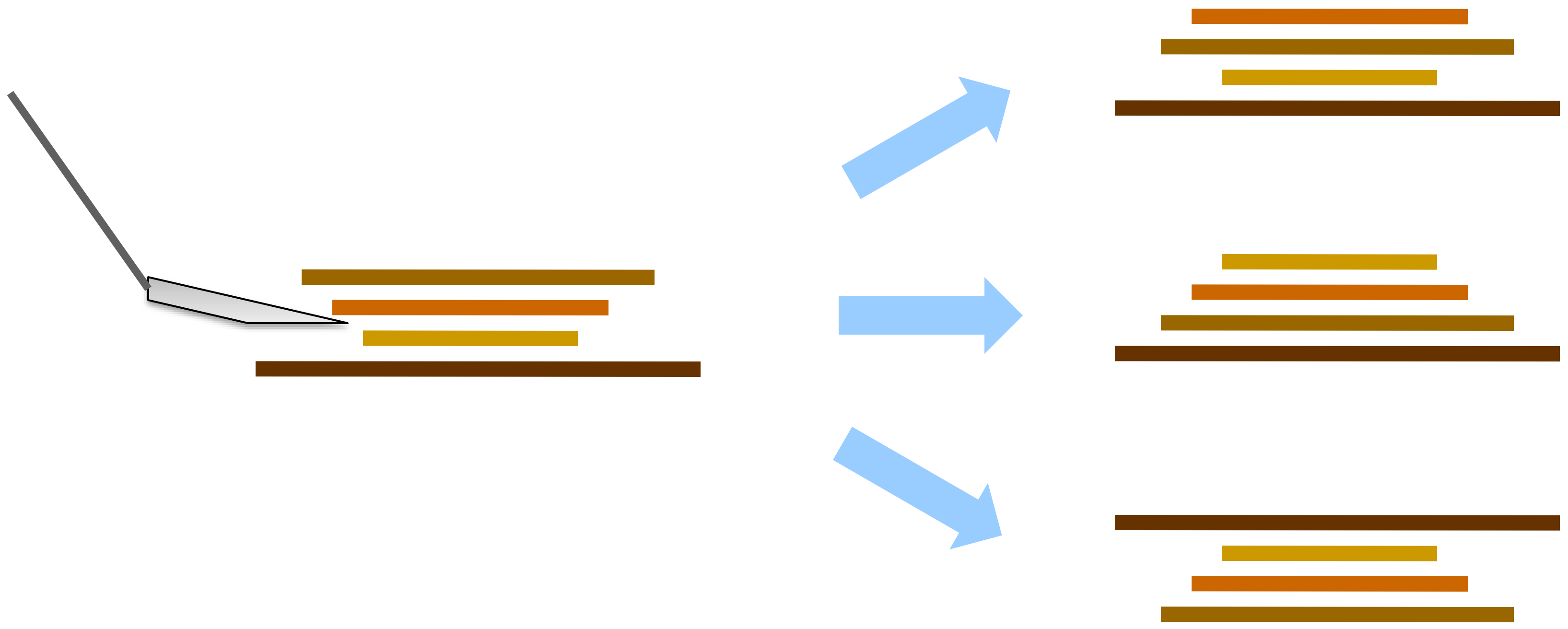


„Félre mehet-e” így a keresés?





Példa: Palacsinta probléma



Költség: A megfordított palacsinták száma

Példa: Palacsinta probléma

BOUNDS FOR SORTING BY PREFIX REVERSAL

William H. GATES

Microsoft, Albuquerque, New Mexico

Christos H. PAPADIMITRIOU*†

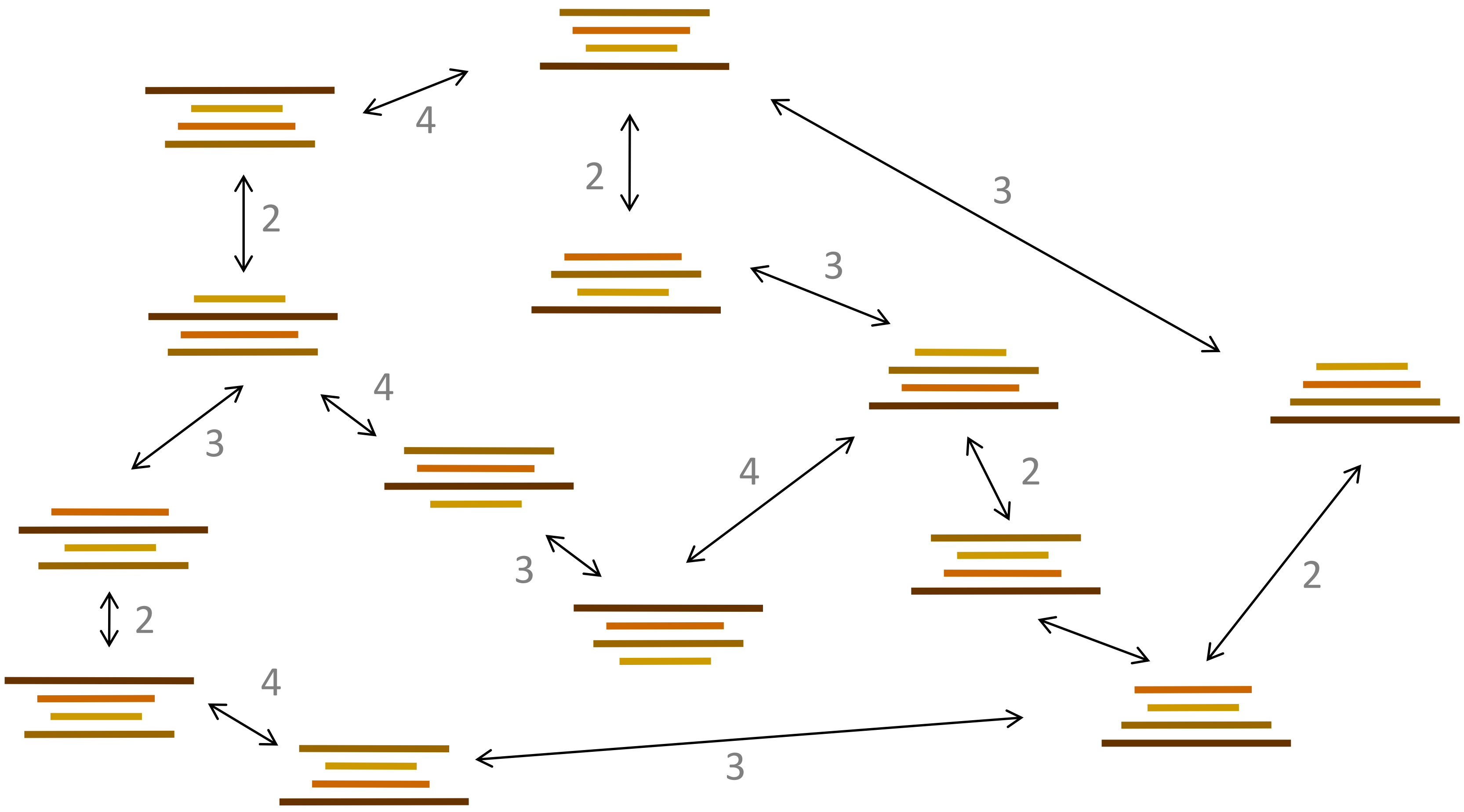
Department of Electrical Engineering, University of California, Berkeley, CA 94720, U.S.A.

Received 18 January 1978
Revised 28 August 1978

For a permutation σ of the integers from 1 to n, let $f(\sigma)$ be the smallest number of prefix reversals that will transform σ to the identity permutation, and let $f(n)$ be the largest such $f(\sigma)$ for all σ in (the symmetric group) S_n. We show that $f(n) \leq (5n+5)/3$, and that $f(n) \geq 17n/16$ for n a multiple of 16. If, furthermore, each integer is required to participate in an even number of reversed prefixes, the corresponding function $g(n)$ is shown to obey $3n/2 - 1 \leq g(n) \leq 2n+3$.

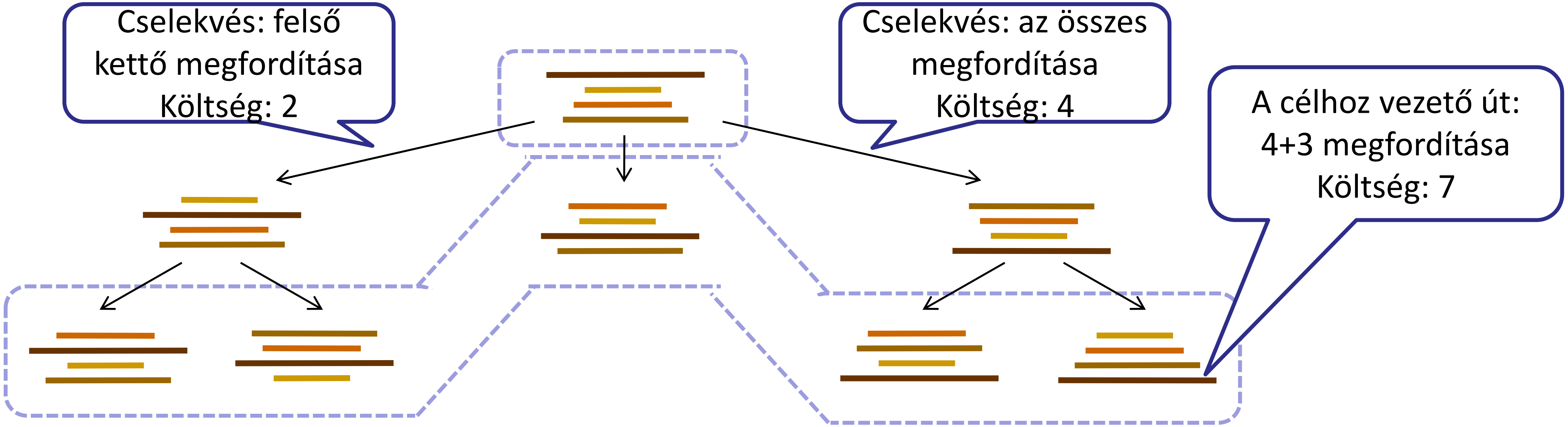
Példa: Palacsinta probléma

Állapottérgráf, ahol a súlyok a költségek



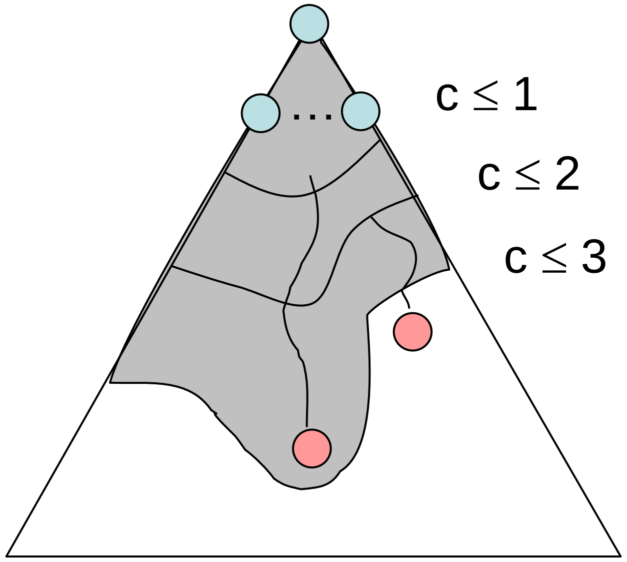
Általános fa keresés

function FA-KERESÉS (*probléma*, *stratégia*) **returns** *megoldás* vagy *kudarc*
    Keresési fa inicializálása a *probléma kiinduló állapotával*
    loop do
        If nincs jelölt a kifejtéshez **then return** *kudarc*
        Kifejtendő levél csomópont kijelölése a *stratégia* alapján
        If a csomópont tartalmaz egy *célállapotot* **then return** *megoldás*
        else csomópont kifejtése és az eredmény csomópontok hozzáadása a keresési fához
    end



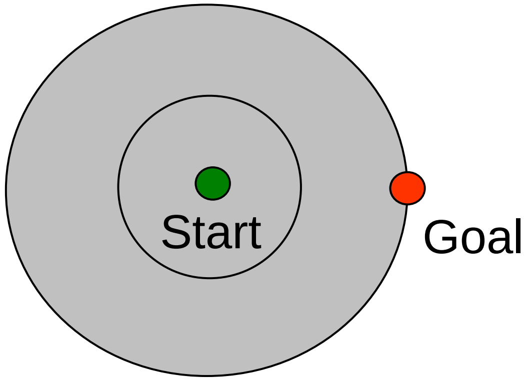
Egyenletes költségű keresés

- Stratégia: legkisebb költségű út kifejtése



- Teljes és optimális!

- A problémа:
  - Minden „irányban” feltár megoldási opciókat
  - Nem rendelkezik információval a célállapot helyzetéről



Kétirányú keresés

- egyszerre előrefelé a kiinduló állapotból, illetve hátrafelé a cél állapotból

- a keresés akkor fejeződik be, ha a két keresés valahol találkozik

$$O(2 \times b^{d/2})=O(b^{d/2})$$

Például: $b = 10$, $d = 6$:

a szélességi keresés = **1.111.111** csomópont,

a kétirányú keresés mindkét irányban 3 mélységnél ér célba = **2.222** csomópontot generál.

Kétirányú keresés

Elméletben nagyon jó, az implementálás nem triviális.
- <u>célállapotból hátrafelé keresni? </u>Az *n* csomópont **előd csomópontjai** azon csomópontok, amelyek követő csomópontja *n*.

- **A hátrafelé keresés** a cél csomópontból indulva az előd csomópontok egymást követő generálását jelenti. Megfordítható-e a folyamat?

- Ha az összes operátor reverzibilis, akkor az előd és követő halmazok azonosak.
- Néhány probléma esetén azonban az elődök meghatározása nagyon nehéz.

Mi van, ha nagyon sok cél állapot létezik?

a cél állapotok egy ***explicit*** listája

a cél állapotok egy ***leírása***

A neminformált keresési stratégiák összehasonlítása

b - elágazási tényező,
m - a keresési fa maximális mélysége,
d - a megoldás mélysége,
l - a mélység korlát.

Jellem-ző	Szélességi keresés	Egyenletes költségű keresés	Mélységi keresés	Mélység korlátozott keresés	Iteratívan mélyülő keresés	Kétirányú keresés
Idő-igény	b^d	b^d	b^m	b^l	b^d	$b^{d/2}$
Tár-igény	b^d	b^d	bm	bl	bd	$b^{d/2}$
Opt.?	Igen (ha…)	Igen	Nem	Nem	Igen	Igen
Teljes?	Igen	Igen	Nem	Igen, ha $l \geq d$	Igen	Igen

Informált keresés