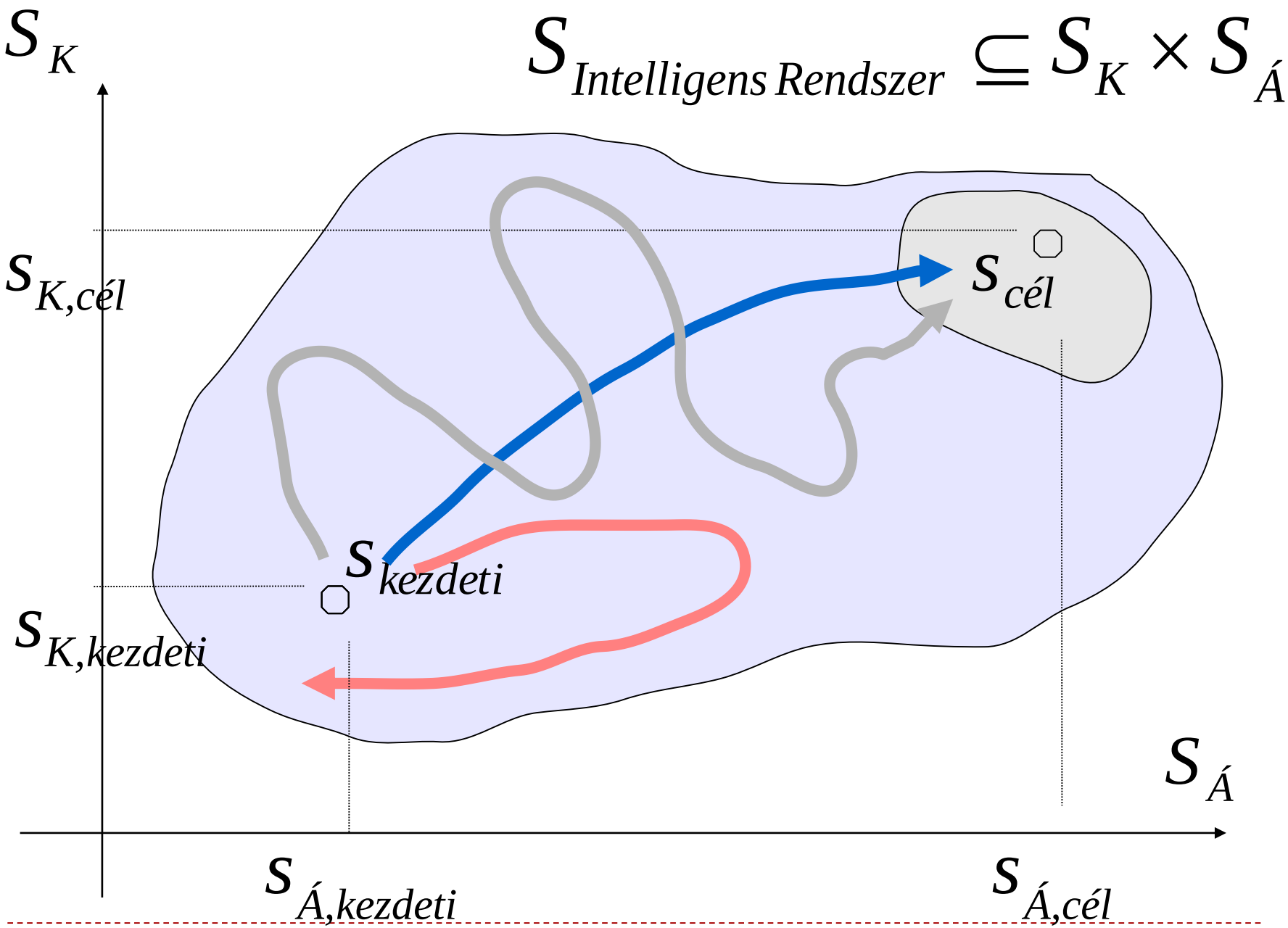

Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs rendszerek Tanszék

# Mesterséges intelligencia – VIMIAC10

## Problémamegoldás kereséssel

Előadó: Hullám Gábor

Előadás anyaga: Pataki Béla
Dobrowiecki Tadeusz



$S_K$
$S_{Intelligens\ Rendszer} \subseteq S_K \times S_Á$
$S_{K,cél}$
$S_{cél}$
$S_{kezdeti}$
$S_{K,kezdeti}$
$S_Á$
$S_{Á,kezdeti}$
$S_{Á,cél}$

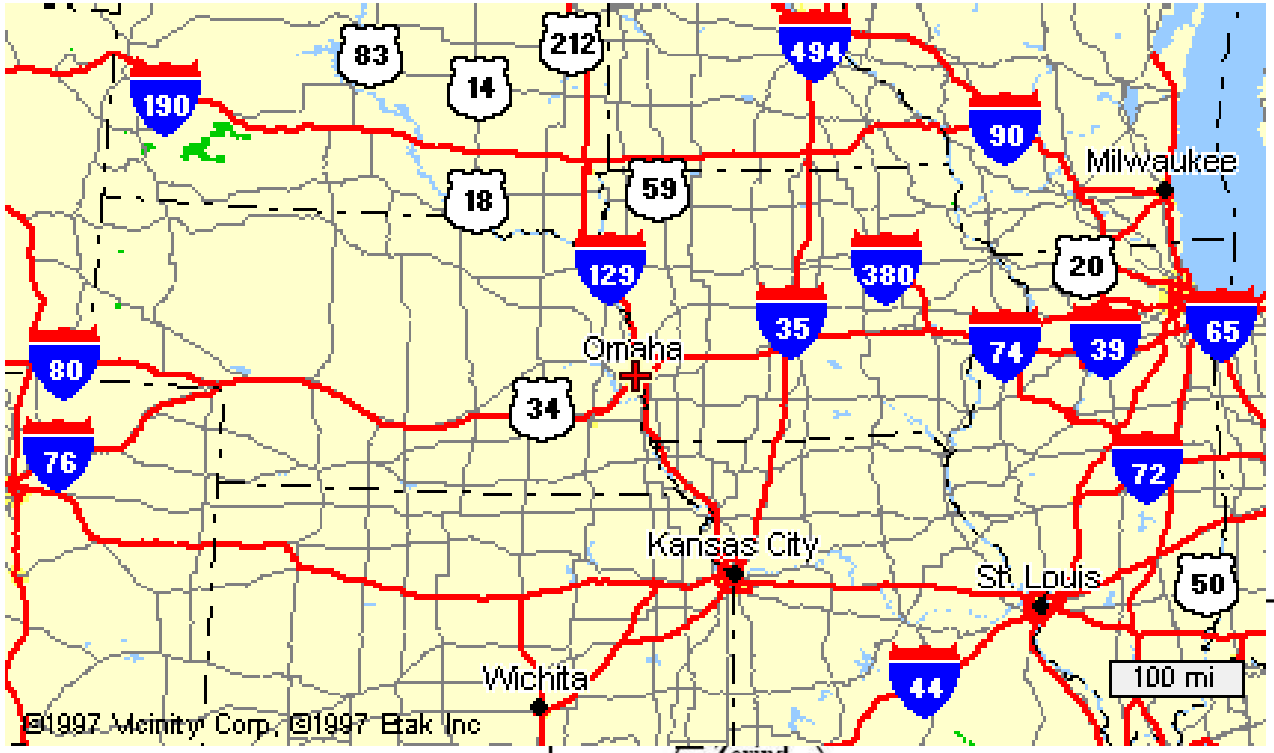
**Keressük meg azt a jó trajektóriát** (utat), ami a kezdeti állapotból a célállapotba visz, a lehető legkisebb költséggel!

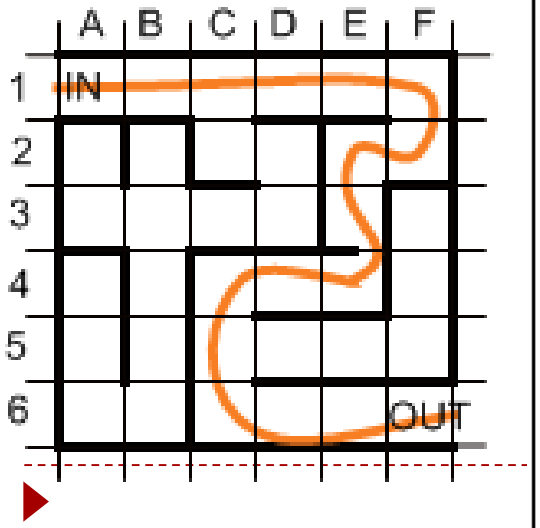
A feladat könnyűnek tűnik, de van ha

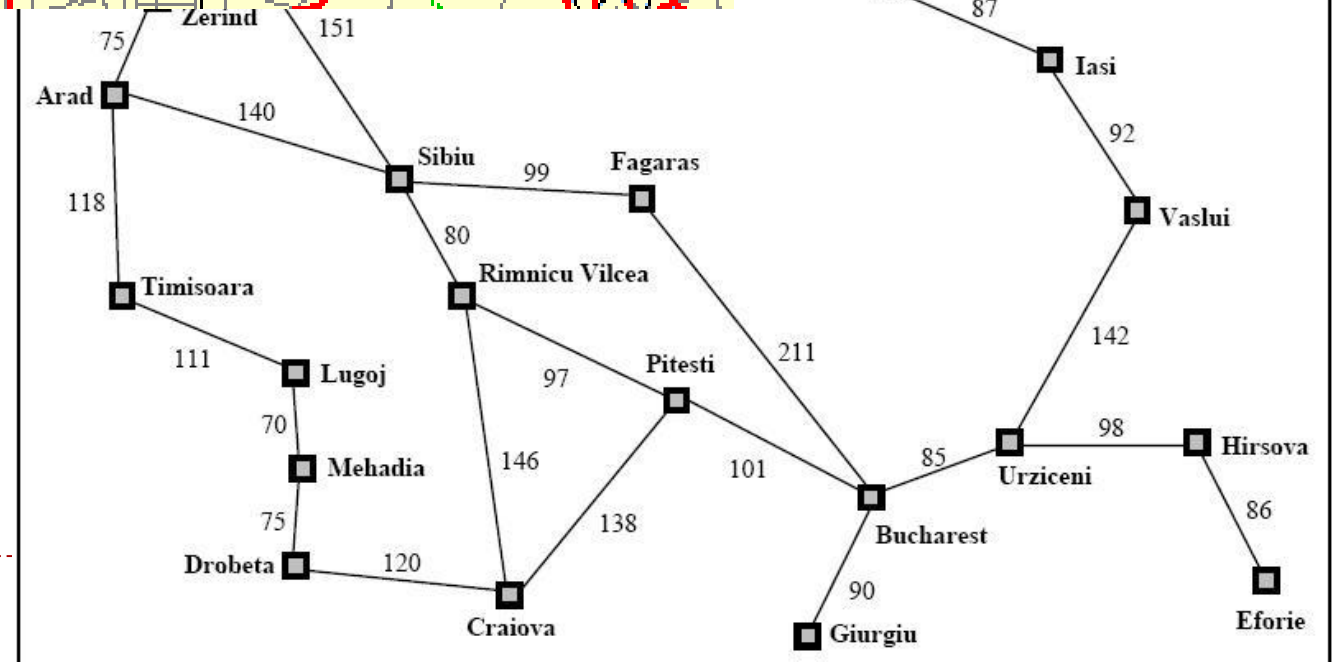
- nagyon sok állapot van,
- sokféle átmenet lehet egyikből a másikba, és
- nem látjuk át „felülről” az állapotteret.

Példa:
„Search and Rescue”….

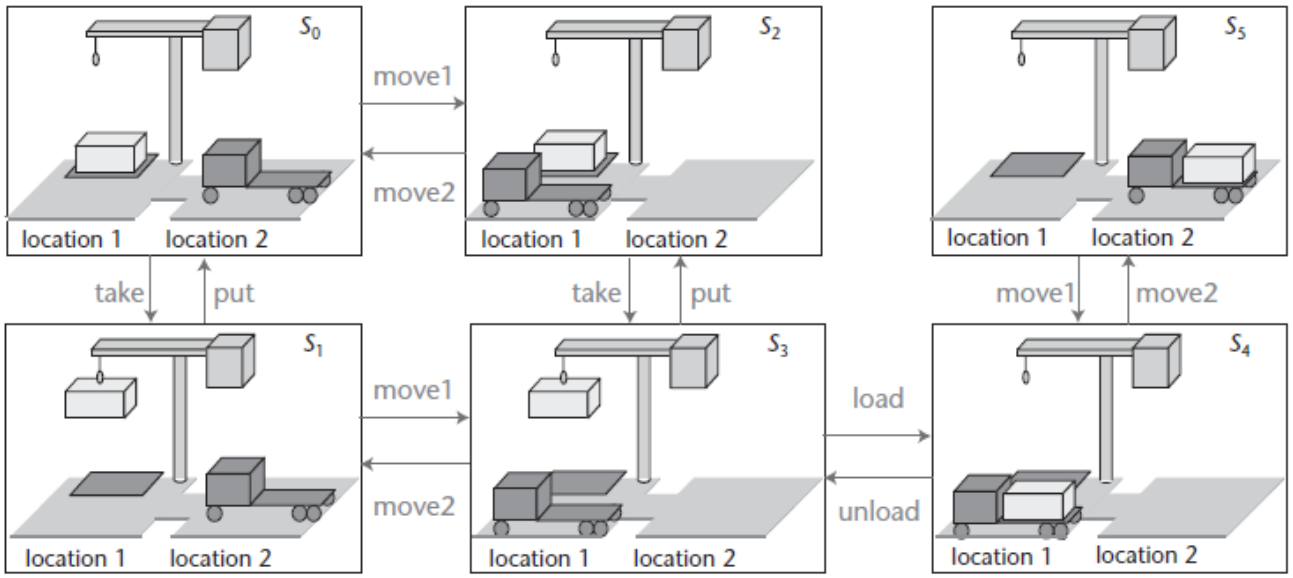
# Fizikai tér

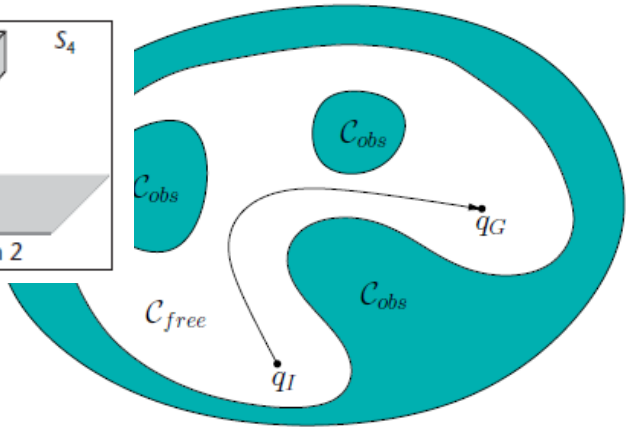

83
14
212
494
190
90
Milwaukee
18
59
129
380
20
35
74
39
65
80
Omaha
34
76
72
Kansas City
St. Louis
50
44
100 mi
Wichita
©1997 Vicinity Corp. ©1997 Etak Inc

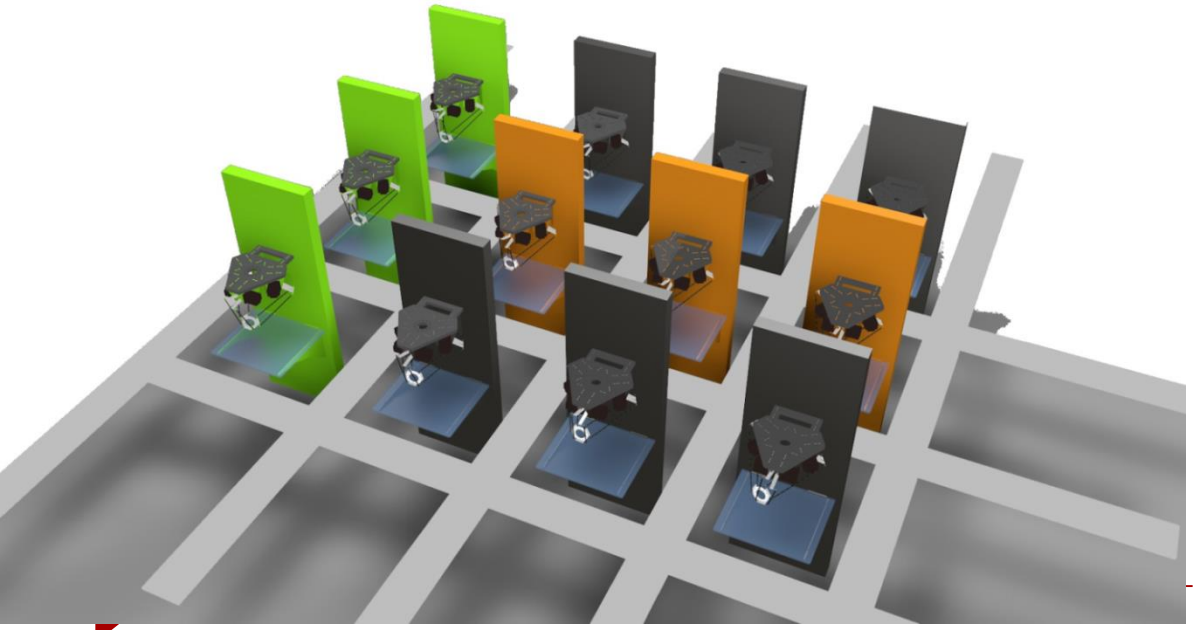

A B C D E F
1 IN
2
3
4
5
6 OUT

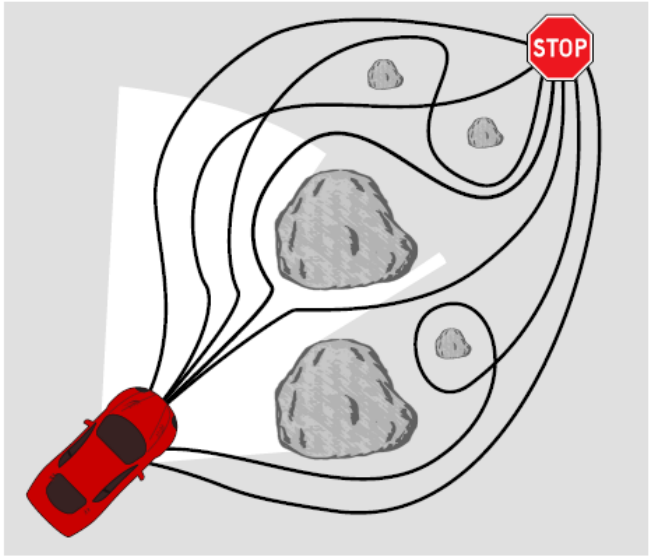

Neamt
87
Iasi
92
Vaslui
142
Zerind
75
151
Arad
140
Sibiu
99
Fagaras
118
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
211
70
Mehadia
146
101
85
Urziceni
98
Hirsova
75
138
Bucharest
86
Drobeta
120
90
Craiova
Giurgiu
Eforie

# Konfigurációs tér

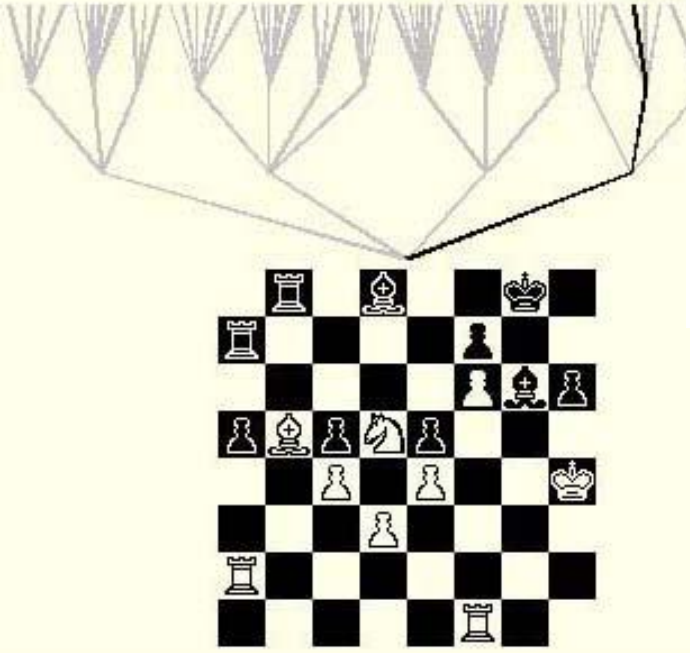


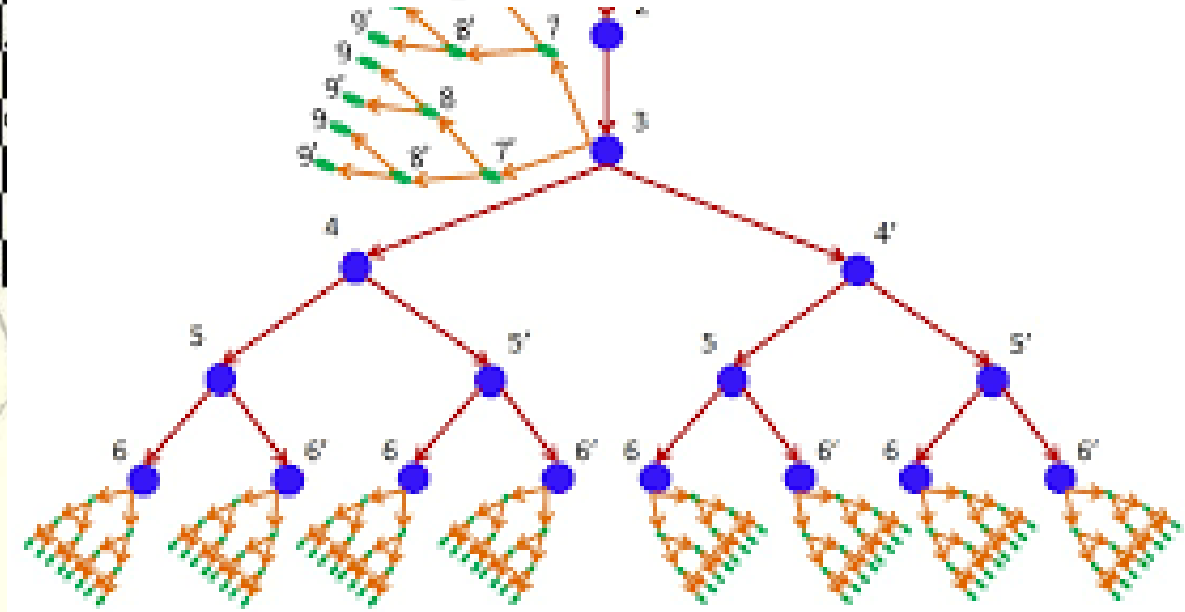



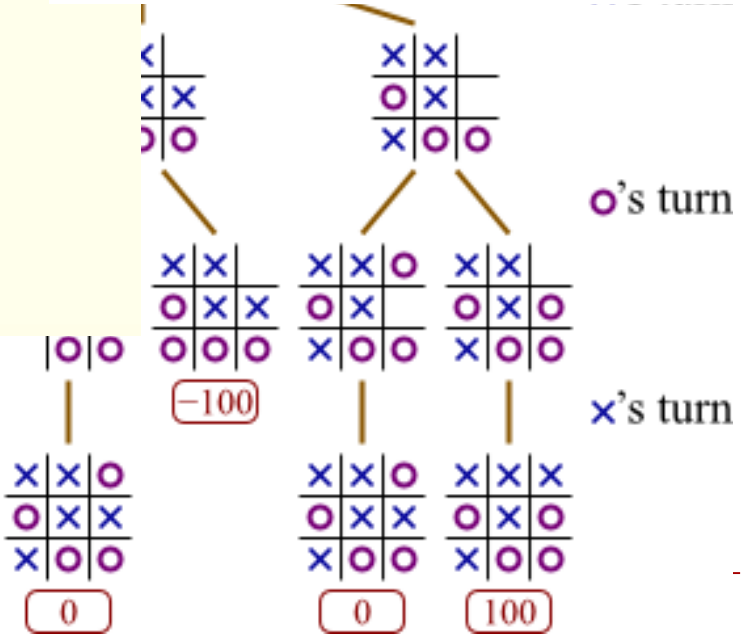


# Absztrakt tér



A graph-theory algorithm for rapid protein side-chain prediction





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!

# 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?

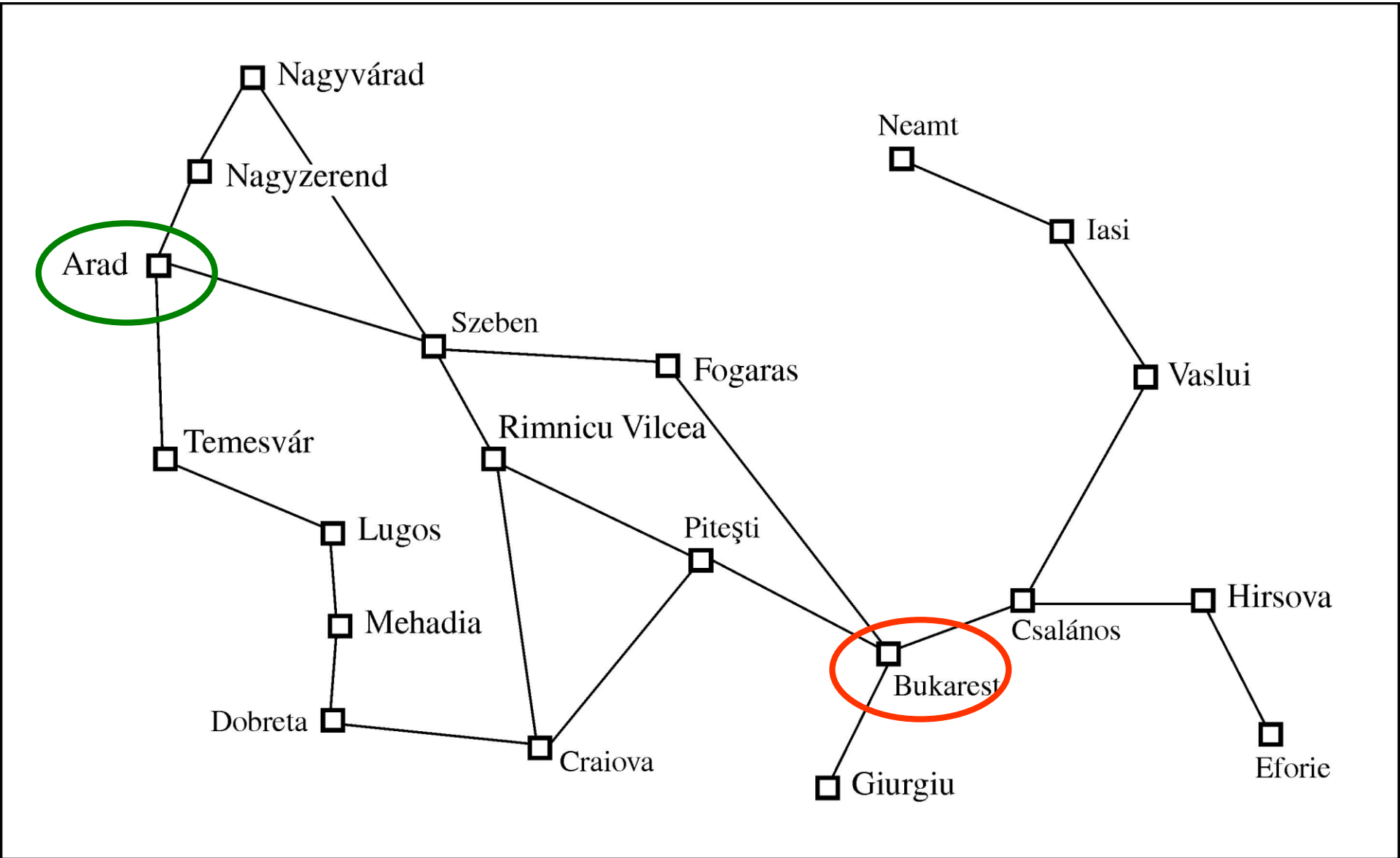
1. Teljesség (*completeness*) - ha van megoldás, biztosan megtalálja

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

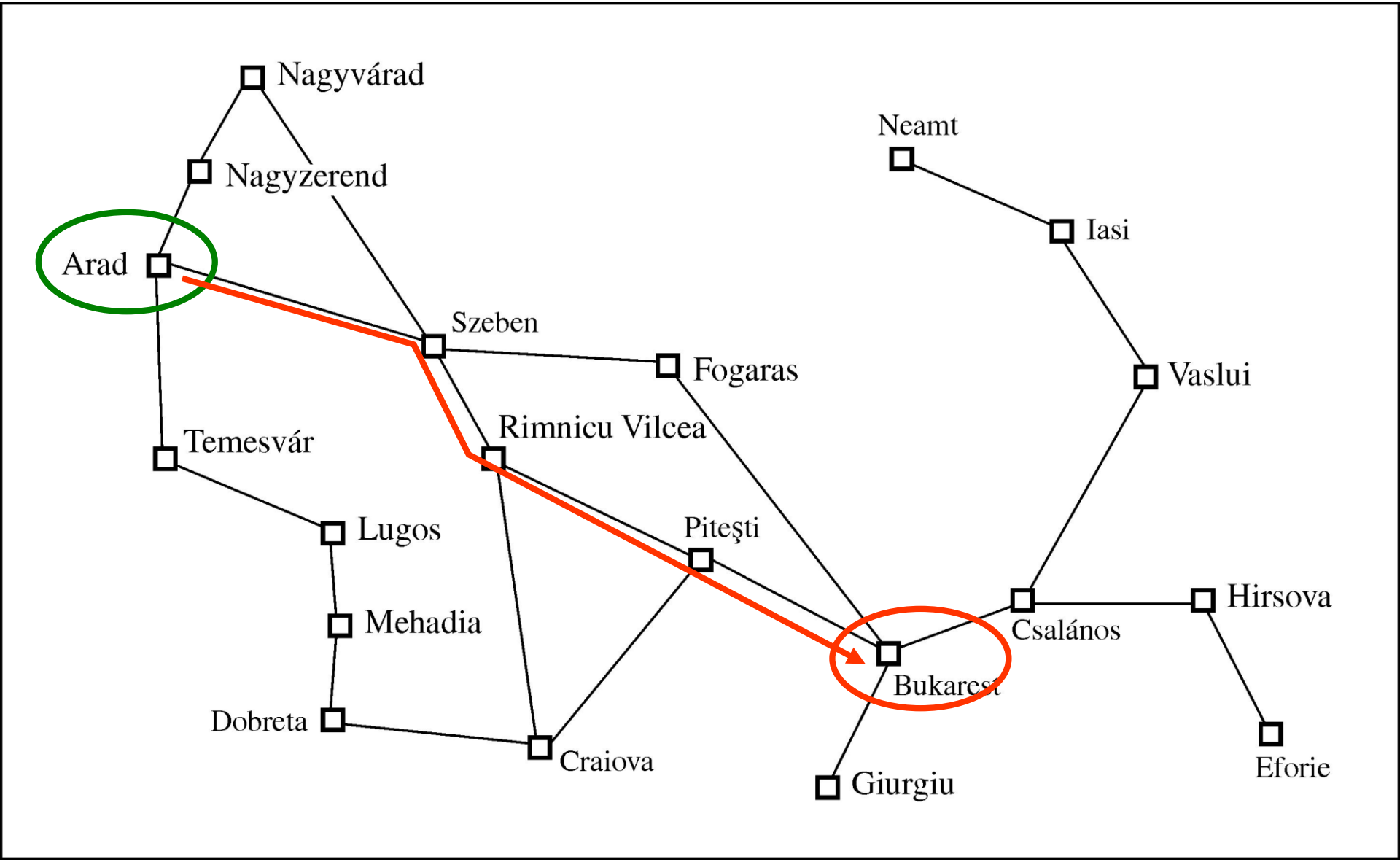
3. Tárigény (*space complexity*) – mennyi tárhely kell

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

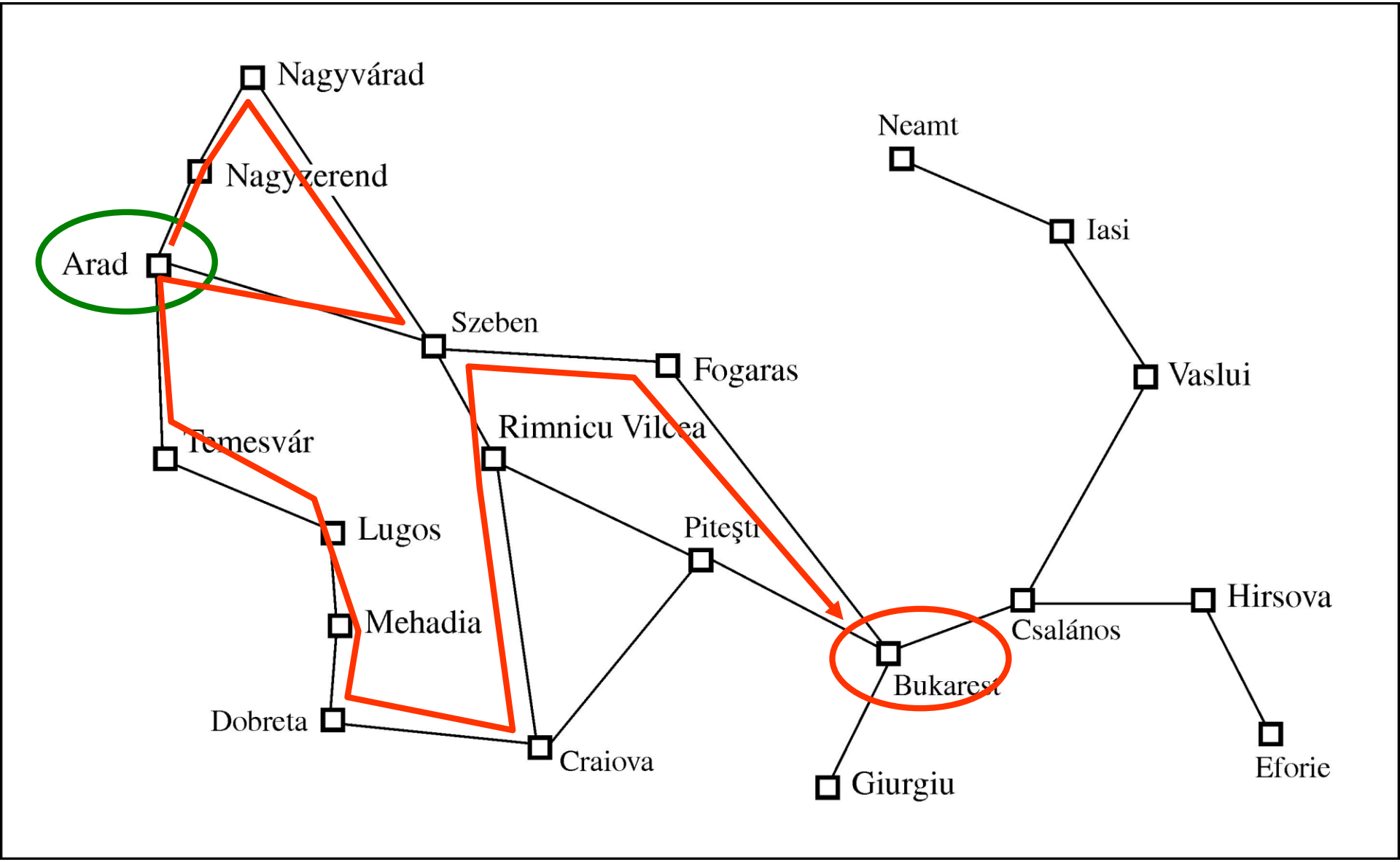
# A Russel-Norvig könyv példája: el akarunk jutni Aradról Bukarestbe...



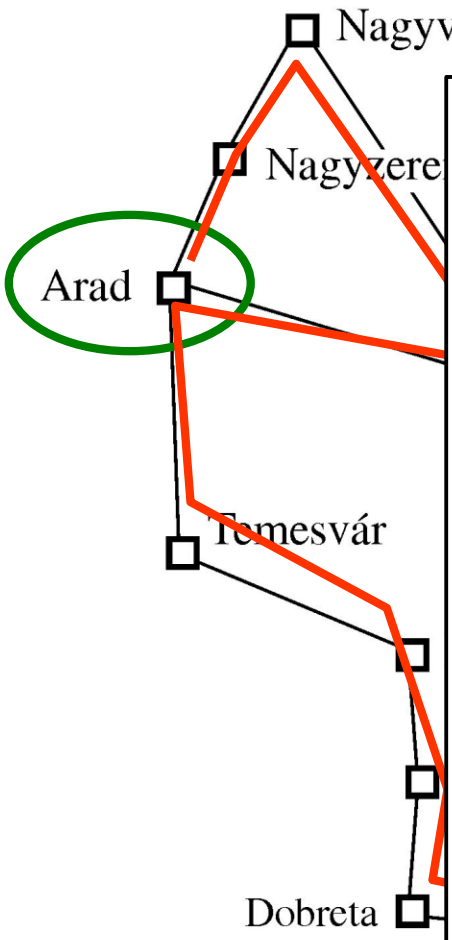
# Milyen utat találunk meg? Így?



# Milyen utat találunk meg? Vagy úgy?



# A probléma formalizálása




**Kezdeti állapot** = Arad
**Célállapot** = Bukarest

**Operátorok**:

IF (X és (X $\Rightarrow$ Y)) THEN Y

**Útköltség** = k + táv(X,Y) + …

**Célállapot teszt**: Y = Célállapot?

**Adatbázis:**

| | |
|---|---|
| (Arad $\Rightarrow$ Nagyzerénd) | táv(Arad,Nagyzerénd) = 75 |
| (Arad $\Rightarrow$ Temesvár) | táv(Arad,Temesvár) = 118 |
| … | ... |

# Absztrakció módja = a „jól definiált” probléma megalkotása

Probléma = információk gyűjteménye:
állapotok + legális cselekvések

Formálisan:

(1) **kiinduló állapot** (tudnunk kell, hogy abban az állapotban vagyunk)

(2) **lehetséges cselekvések** halmaza

- **operátor** = egy-egy cselekvés leírása, tipikusan

  HA egy ilyen állapotban van,
  AKKOR majd olyan állapotban lesz
  a cselekvés alkalmazása után

- „**követő állapot**” függvény = egy cselekvés egy adott állapotban való alkalmazásának hatására az ágens mely állapotba kerül?

# A „jól definiált” probléma

Formálisan:

(1) **kiinduló állapot**
(2) **lehetséges cselekvések** halmaza

Ezek együtt (implicit módon megadva): **a probléma állapottere**

- azon állapotok halmaza, amelyek a kiinduló állapotból valamilyen cselekvés sorozattal elérhetők.

- Nehézség: az állapottér explicite ritkán adható meg (pl.: sakk)
- Állapottérben **út:** állapotok között vezető cselekvéssorozat

(3) **célteszt**: el kell tudnunk dönteni, hogy az adott állapot egy célállapot-e.

*a. a lehetséges célállapotok egy **explicit** halmaza*
*- egyszerűen megnézzük, hogy elértük-e ezek egyikét*

*b. a cél valamilyen **absztrakt tulajdonsággal** van definiálva,*
*pl. a sakkban az ún. „sakk-matt” helyzet*

(4) **útköltség** függvény: az úthoz hozzárendel egy költséget

A keresési algoritmus kimenete - egy **megoldás**:
a kiinduló állapotból egy olyan állapotba vezető út, amely teljesíti a cél tesztet.

## A problémamegoldó hatékonyság mérése

- Mi a megoldás megtalálásához szükséges (idő/ tár) **keresési költség?**
- A keresés **összköltsége**: **az útköltség + a keresési költség (!)**

# 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

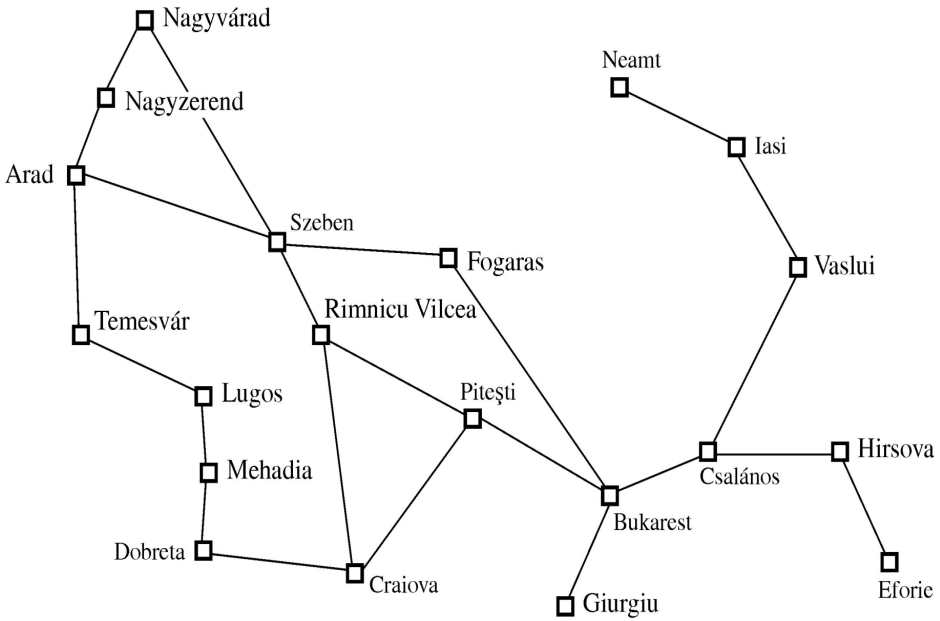
# Keresési stratégiák – avagy miből gazdálkodhatunk?

**Nem informált** keresés: nincs különbség számomra, merre indulok...

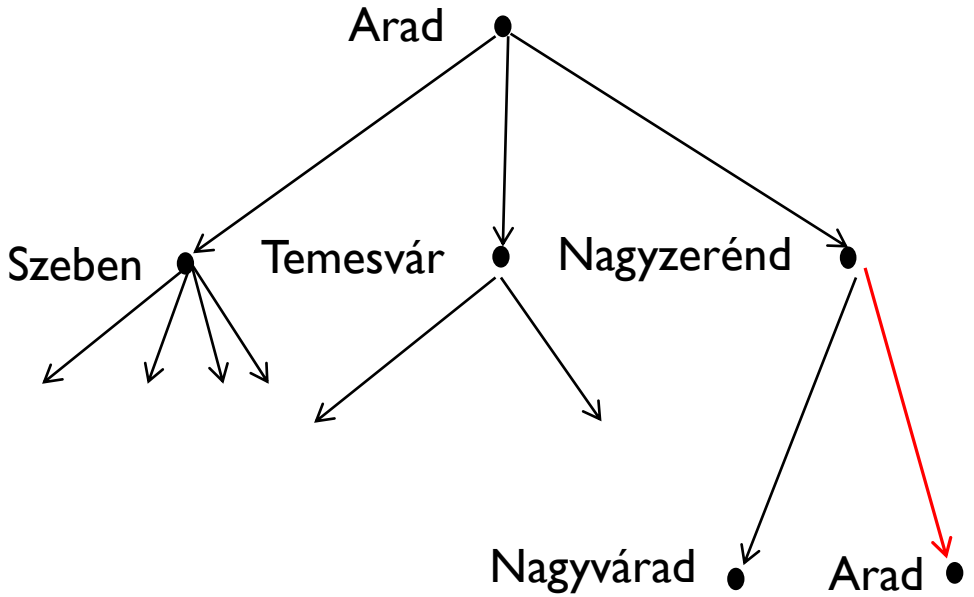
**Informált, okosabb ágens**: olyan irányba lép, ami a célállapot irányában fekszik ...

A vak (nem informált) keresési stratégiákat a **csomópontok kifejtési sorrendje különbözteti meg**. Ez a különbség óriási jelentőséggel bírhat!

**...** útkereső probléma: Aradból kiindulva 3 cselekvés 3 állapotba vezet:
Szebenbe, Temesvárra és Nagyzeréndre ...



A keresés folyamatát az ún. **keresési fával** ragadjuk meg.




A <u>gráf csomópontjainak adatszerkezete</u> (pl.):

1. az állapottérnek a csomóponthoz tartozó állapota
2. szülő csomópont
3. a csomópontot generáló operátor
4. a gyökértől eddig a csomópontig vezető út csomópontjainak száma (mélység – *depth*)
5. a csomópontig tartó útköltség

<u>Fontos fogalmak</u>:

- kifejtés (*expand*)
- (átl.) elágazási tényező – *b* (*branching factor*)
- mélység – *d* (*depth*)
- hullámfront (*frontier*)

**Szélességi keresés**

szisztematikus feltárás széles fronton, a visszalépésre nincs szükség (nincs mihez)

**Példa** (Russel-Norvig könyv): $b$=10, 10.000 csp/perc feldolgozási sebesség (100 μsec), 1000 B/csp tárigény

(pl. a sebesség nyilván elavult, de ha 3 nagyságrenddel gyorsabb a program futása, akkor se vonzó 3,5 évet várni)

| Mélység | Csomópontok | Időigény | Tárigény |
|---|---|---|---|
| 2 | 1100 | 0,11 sec | 1 MB |
| 4 | 111100 | 11 sec | 106 MB |
| 6 | $10^7$ | 19 perc | 10 GB |
| 8 | $10^9$ | 31 óra | 1 TB |
| 10 | $10^{11}$ | 129 nap | 101 TB |
| 12 | $10^{13}$ | 35 év | 10 PB |
| 14 | $10^{15}$ | 3523 év | 1 EB |

# Egyenletes költségű keresés (Dijkstra)

Szélességi keresés módosítása:

- a csomópontoknál az **eddig megtalált útszakasz útköltségét** is tároltuk **g(*n*)** (startállapottól eddig, az *n*-dik csomópontig)

- a hullámfront g(*n*) útköltség függvénnyel mért legkisebb költségű csomópontját fejti ki először, nem pedig a legkisebb mélységű csomópontot.

- A szélességi keresés is egyenletes költségű keresés, amelyben:

**g(*n*) = Mélység(*n*)**,

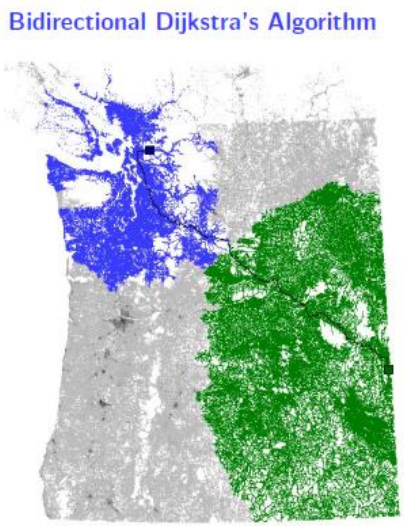
vagyis a költség a csomópont mélysége.

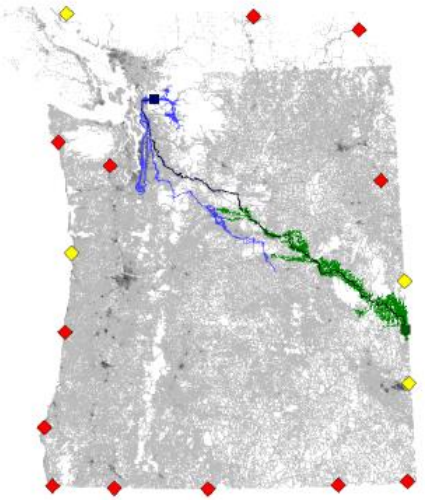
# Egyenletes költségű keresés (Dijkstra)

Az egyenletes költségű keresés a legolcsóbb megoldást találja meg, feltéve, ha az **útköltség egy út bejárása során nem csökkenhet**:

$$\mathbf{g(Követő(}n\mathbf{))} \geq \mathbf{g(}n\mathbf{)}$$

minden egyes *n* csomópontra.

## Mélységi keresés

- haladás előre, egy ág mentén, de a visszalépés lehetőségével

- (pl. sakk: d2-d4-el nyitunk, és kizárólag a sötét d7-d5 válaszát bontjuk ki, annak is egy alváltozatát, al-alváltozatát stb.)

## 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.

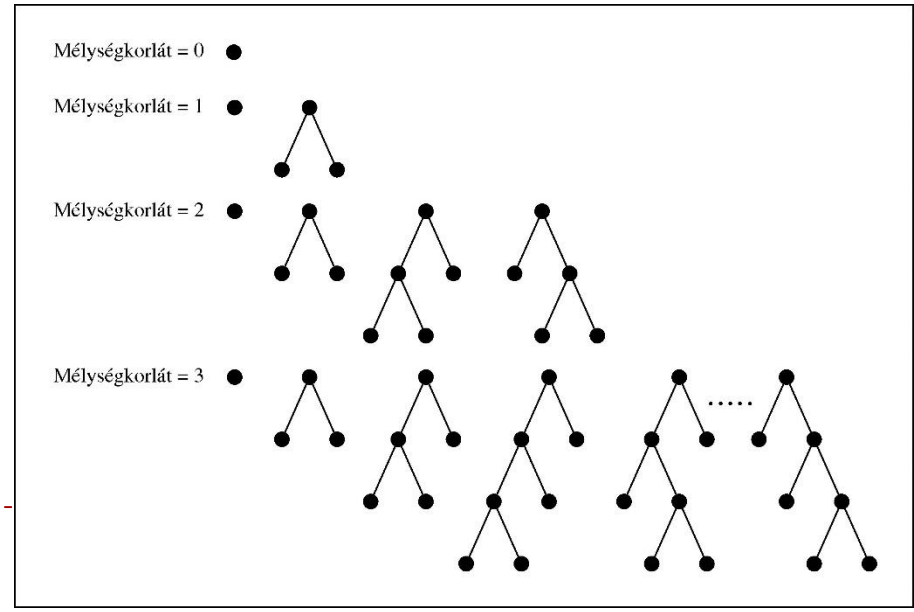
# Iteratívan mélyülő keresés

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 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:

$1 + b + b^2 + \ldots + b^{d-2} + b^{d-1} + b^d$

az iteratívan mélyülő keresésnél a kifejtések teljes száma:

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

# Iteratívan mélyülő keresés

- szélességi keresésnél a kifejtések száma:

$1 + b + b^2 + \ldots + b^{d-2} + b^{d-1} + b^d$

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

- az iteratívan mélyülő keresésnél a kifejtések teljes száma:

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

$b$ = 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!)

## 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.
- célállapotból hátrafelé keresni? 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***

- Például a sakkban mik a sakk-matt célállapotok (nem egy célállapot van!) megelőző állapotai?

- Hatékony módszer kell arra, hogy: egy frissen generált csomópont megjelenik-e már a másik fél keresési fájában.
- El kell tudni dönteni, hogy az egyes félrészekben milyen keresésre fog sor kerülni.

- $O(b^{d/2})$ komplexitás feltételezi, hogy a két hullámfront metszésének megállapítása konstans idő alatt elvégezhető(?)

- Hogy a két keresés találkozzon, legalább az egyik keresés összes csomópontját memóriában kell tartani = tár $O(b^{d/2})$.

# A neminformált keresési stratégiák összehasonlítása

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

| Jellemző | 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 |

# A neminformált keresési stratégiák összehasonlítása

$b$ - elágazási tényező, $d$ - a megoldás mélysége,
$m$ - a keresési fa maximális 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 |