Alkalmazott mesterséges intelligencia (AMI)

http://www.mit.bme.hu/oktatas/targyak/vimibb01

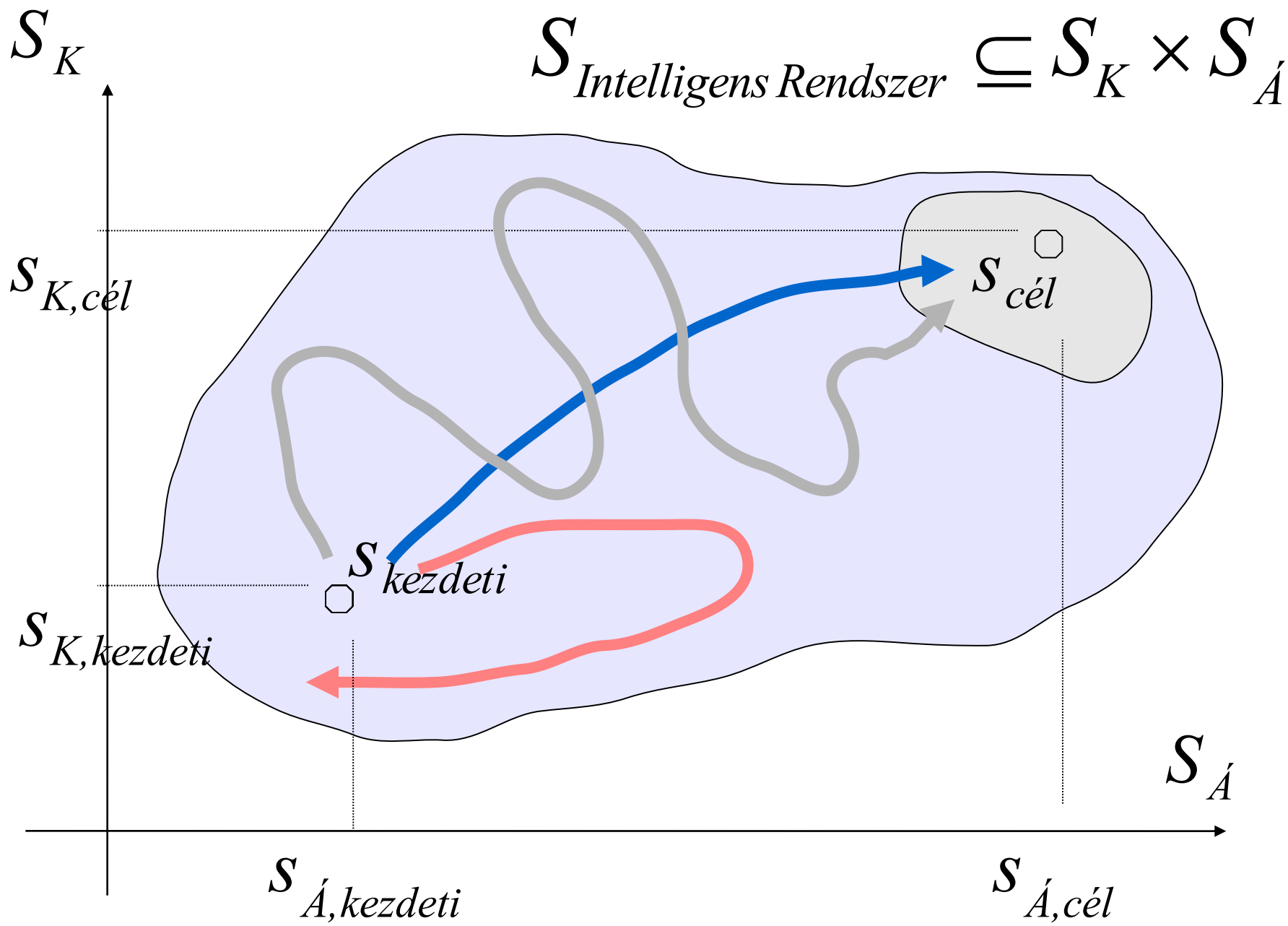
2 ea.- (2019 ősz)

Problémamegoldás kereséssel – vakon

http://mialmanach.mit.bme.hu/aima/ch03s03
3. fejezet 3.4 alfejezet

Előadó: Pataki Béla

BME I.E. 414, 463-26-79
pataki@mit.bme.hu,
http://www.mit.bme.hu/general/staff/pataki


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

(Próbáljuk meg egy részletes térképen megkeresni az autóval legrövidebb utat Sásdról Győrtelekre úgy, hogy mindig csak azt látjuk, hogy az adott helységnek melyek a közvetlen, autóúton elérhető szomszédai.)

Mindig nagyon fontos, hogy minél pontosabban meg tudjuk fogalmazni, mi a célunk, mit tekintünk jónak!

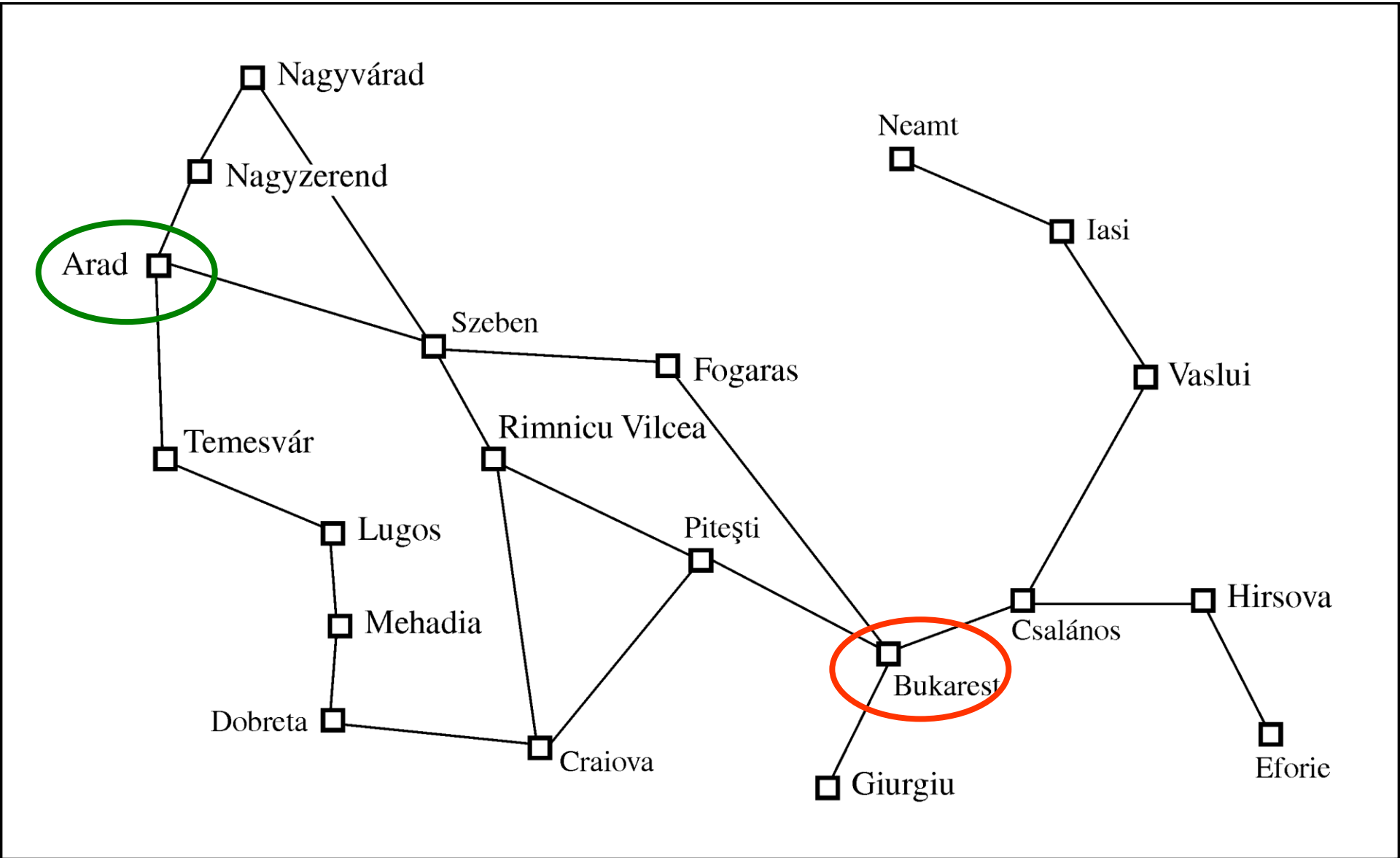
Itt, a megoldási út keresésnél melyik a jó eljárás?

Részben az elért eredmény, részben annak megtalálási költsége minősíti. A szokásos értékelési szempontok:

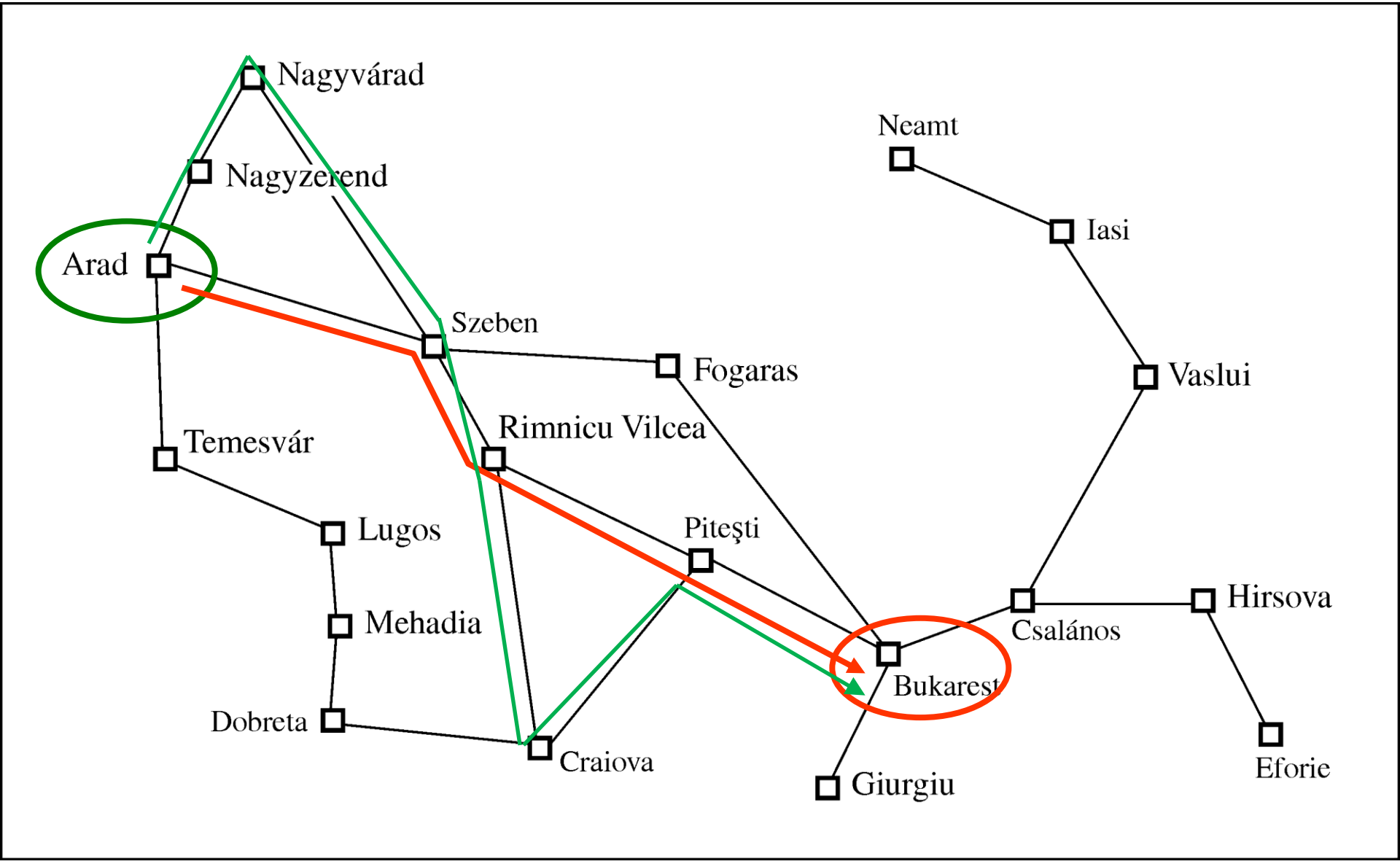
1. Teljesség (*completeness*) - ha van megoldás, biztosan megtalálja
2. Időigény (*time complexity*) – mennyi idő a megoldás megtalálása?
3. Tárigény (*space complexity*) – mennyi tárhely kell
4. Optimalitás (*optimality*) – ha több megoldás van, megtaláljuk-e a legjobbat? (megint, mi a legjobb? – sokszor izgalmas kérdés!)

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

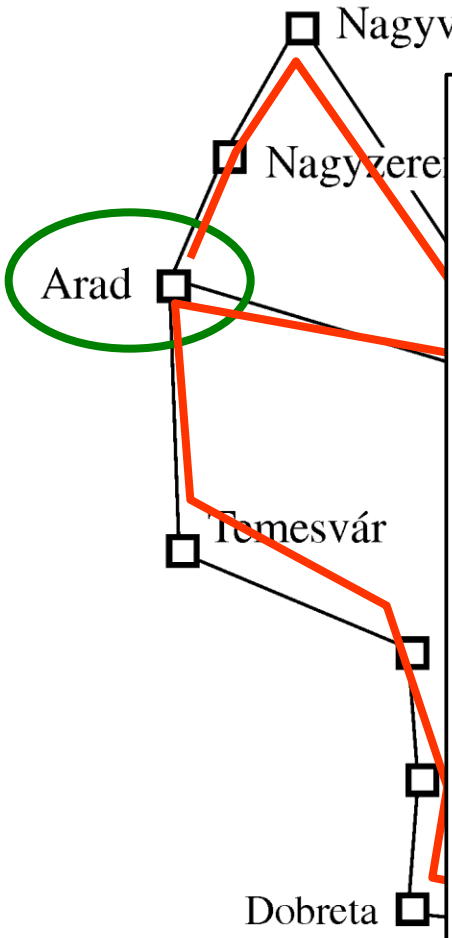
(FONTOS: a keresés, nem csak térképen keresést jelent, csak ez az egyik legszemléletesebben bemutatható probléma!)



Milyen utat találunk meg?



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
… ...

Keresési stratégiák – 1

Nem informált keresések (un. **gyenge**, vagy **vak** keresések)

a. tudjuk: hogy néz ki a start- és a célállapot(ok), ismerjük az állapotok közti lehetséges átmeneteket

b. egyáltalán nincs infónk arról: milyen költségű az aktuálisból a célállapotba vezető út, merrefele kell elindulnunk

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

Múlt óra végén vázlatosan:

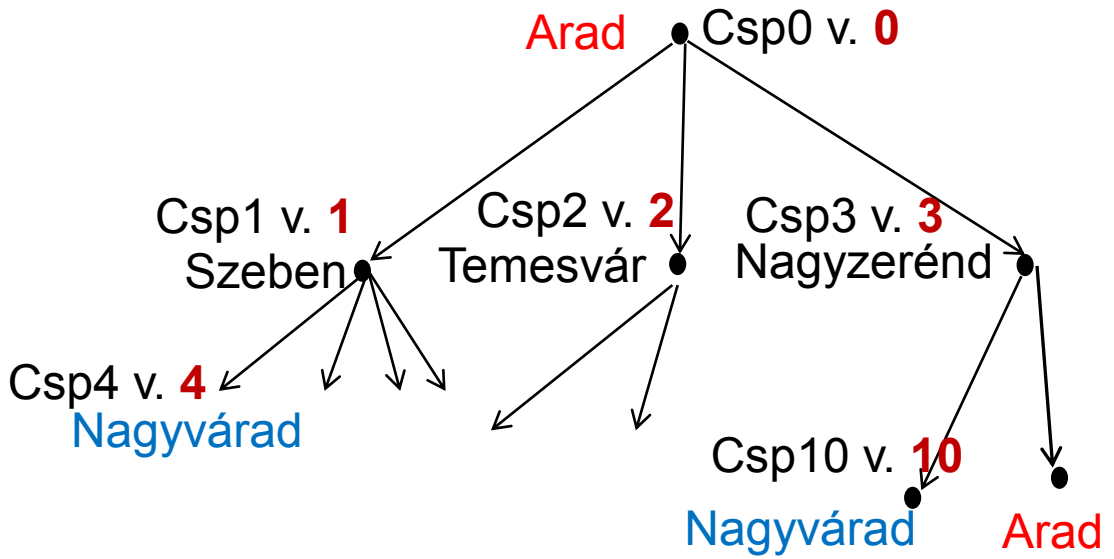
1. Szélességi keresés (width first search)
2. Mélységi keresés (depth first search)

Most szisztematikusan megvizsgáljuk ezeket.

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

A gráf csomópontjainak adatszerkezete (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 $g(n)$)
6. Stb.



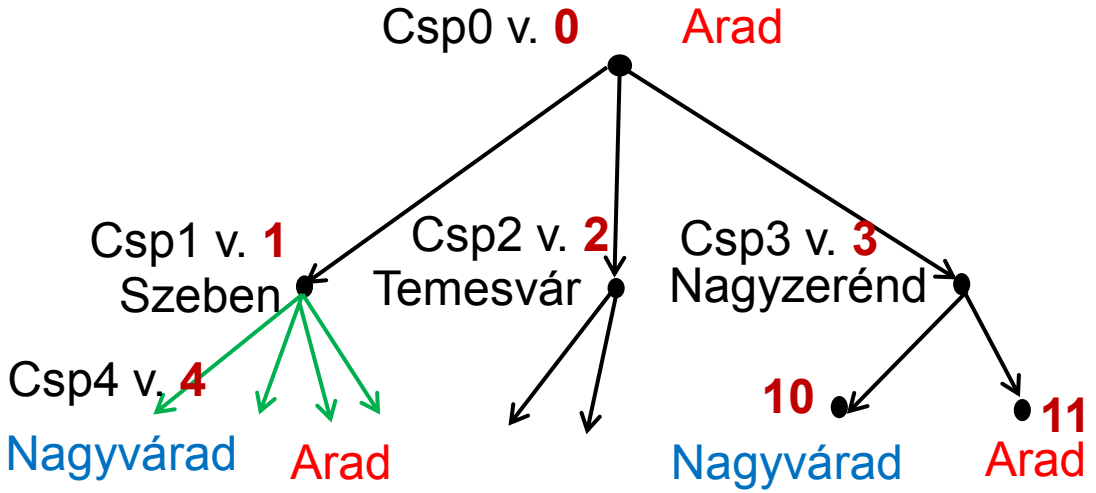
Egyetlen „Nagyvárad” vagy „Arad” állapot van, de több csomóponthoz is tartozhat ez az állapot.

Fontos fogalmak:

- kifejtés (*expand*) – általában ez időigényes!
- (átlagos) elágazási tényező – *b* (*branching factor*)
- mélység – *d* (*depth*)
- hullámfront (*frontier*)

Szélességi keresés

Két listát kezelünk, openList (oL) és closedList (cL):



0. Iniciálás:
- mindkettő üres oL={},cL={}
- oL={ Csp0 }

1. Vesszük az oL első elemét, megvizsgáljuk, hogy elértük-e a Célt. Ha nem, kifejtjük, majd a keletkező gyermek csomópontokat felvesszük oL ***végére***. A vizsgált első elemet betesszük cL-be.

oL={Csp1, Csp2, Csp3}, cL={Csp0}

2. Vesszük az oL első elemét, megvizsgáljuk, hogy elértük-e a Célt. Ha nem, kifejtjük, majd a keletkező gyermek csomópontokat felvesszük oL ***végére***. A vizsgált első elemet betesszük cL-be.

oL={Csp1, Csp2, Csp3}, cL={Csp0}

3. Veszzük az oL első elemét, megvizsgáljuk, hogy elértük-e a Célt. Ha nem, kifejtjük, majd a keletkező gyermek csomópontokat felvesszük oL ***végére***. A megvizsgált első elemet beteszi cL-be.

oL={Csp2, Csp3, Csp4, Csp5, Csp6, Csp7}, cL={Csp0, Csp1}

…. ezt ismételjük, amíg a sor első csomópontjának állapota nem lesz a Cél!

+ minden lépésnél a Cél vizsgálat után meg kell vizsgálni, hogy nem jött-e létre hurok!

Egyenletes költségű keresés (Dijkstra)

A szélességi keresés módosítása:

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

-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:

$$\mathbf{g(}n\mathbf{) = Mélység(}n\mathbf{)},$$

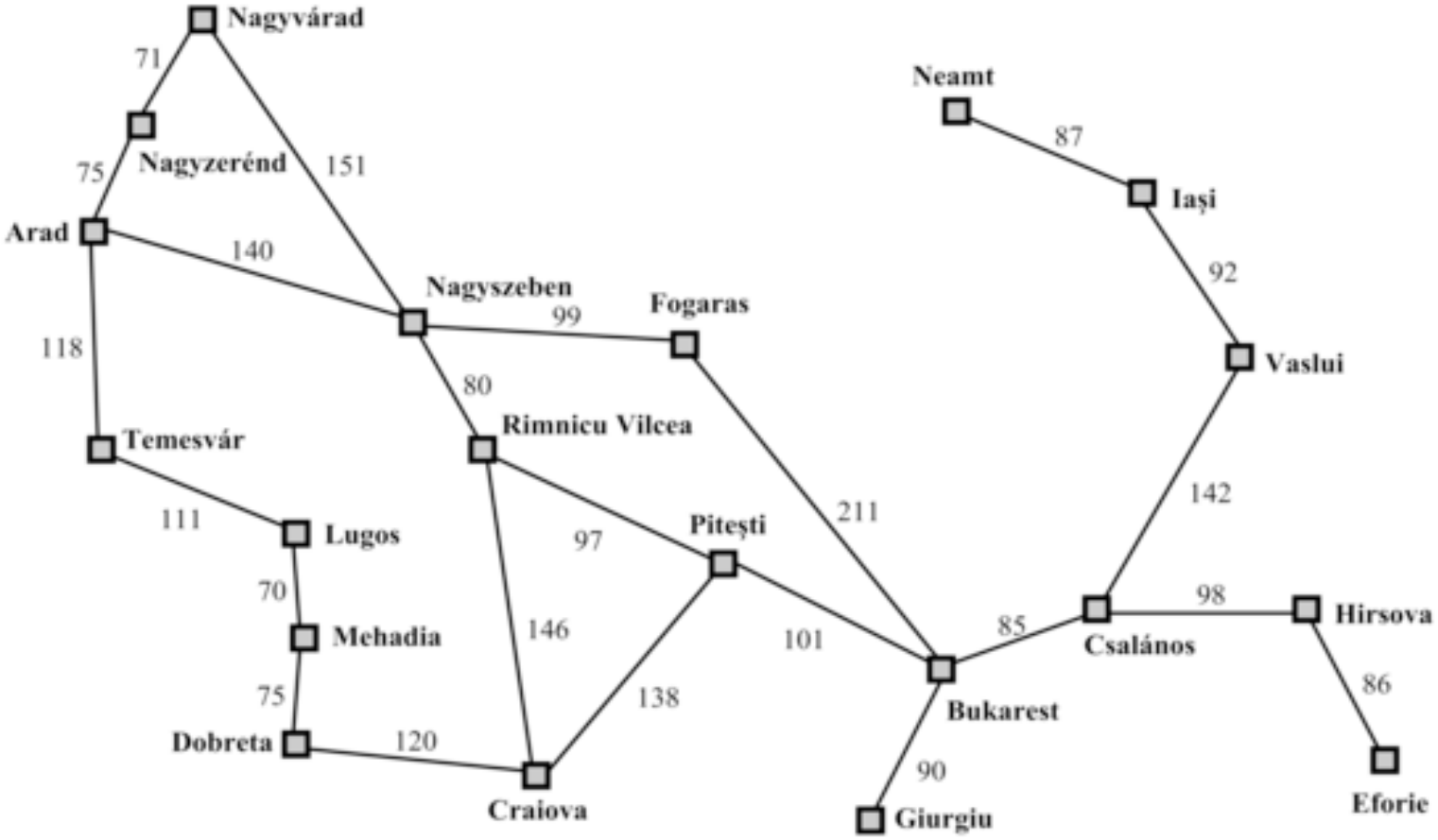
vagyis a csomópont mélysége.

Az egyenletes költségű keresés a legolcsóbb megoldást találja meg, feltéve ha:

az út költség egy út bejárása során nem csökkenhet:

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

minden egyes *n* csomópontra.

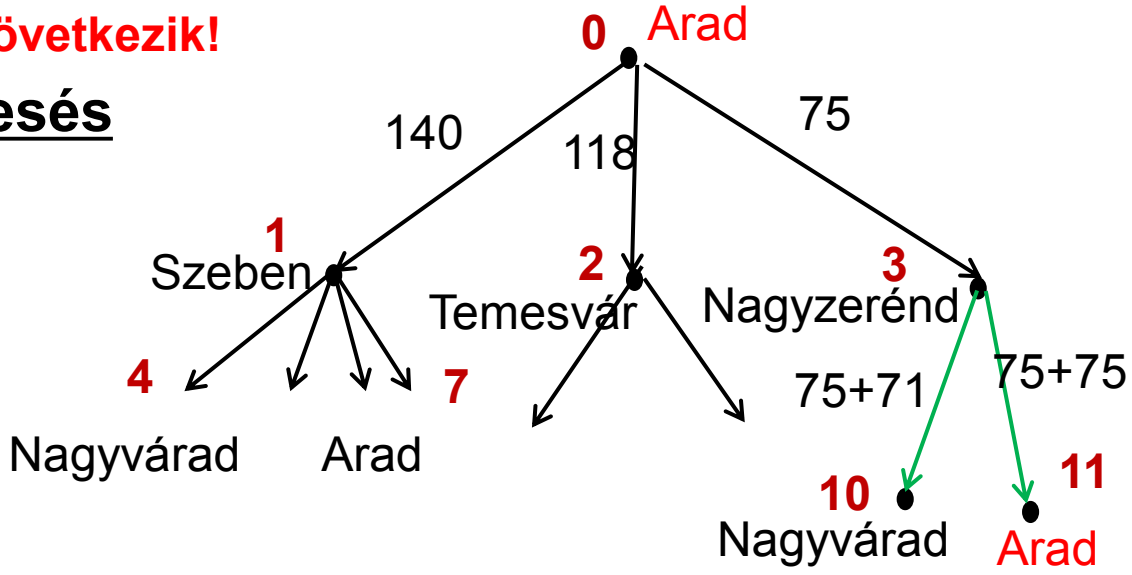

Nagyvárad
71
Nagyzerénd
75
151
Arad
140
Nagyszeben
99
Fogaras
118
80
Rimnicu Vilcea
Temesvár
111
Lugos
97
Piteşti
211
70
Mehadia
146
101
75
138
Dobreta
120
Craiova
Neamt
87
Iaşi
92
Vaslui
142
98
Hirsova
85
Csalános
86
Bukarest
90
Giurgiu
Eforie

Kvíz következik!

Egyenletes költségű keresés

0. Iniciálás:
- mindkettő üres oL={},cL={}
- oL={Csp0}

1. Vesszük … oL-re, a ***megtett úthossz növekvő sorrendjében***.



2. Vesszük az oL első elemét, megvizsgáljuk, hogy elértük-e a Célt. Ha nem, kifejtjük, majd a keletkező gyermek csomópontokat felvesszük oL-re, a ***megtett úthossz növekvő sorrendjében***. A vizsgált első elemet betesszük cL-be.
oL={Csp3, Csp2, Csp1}, cL={Csp0}

3. Vesszük az oL első elemét, megvizsgáljuk, hogy elértük-e a Célt. Ha nem, kifejtjük, majd a keletkező gyermek csomópontokat felvesszük oL-re de ***a megtett úthosszak, g(n) sorrendje szerint***! A megvizsgált első elemet betesszük cL-be.
oL={Csp2, Csp3, Csp10, Csp11}, cL={Csp0, Csp3}
118 140 146 150

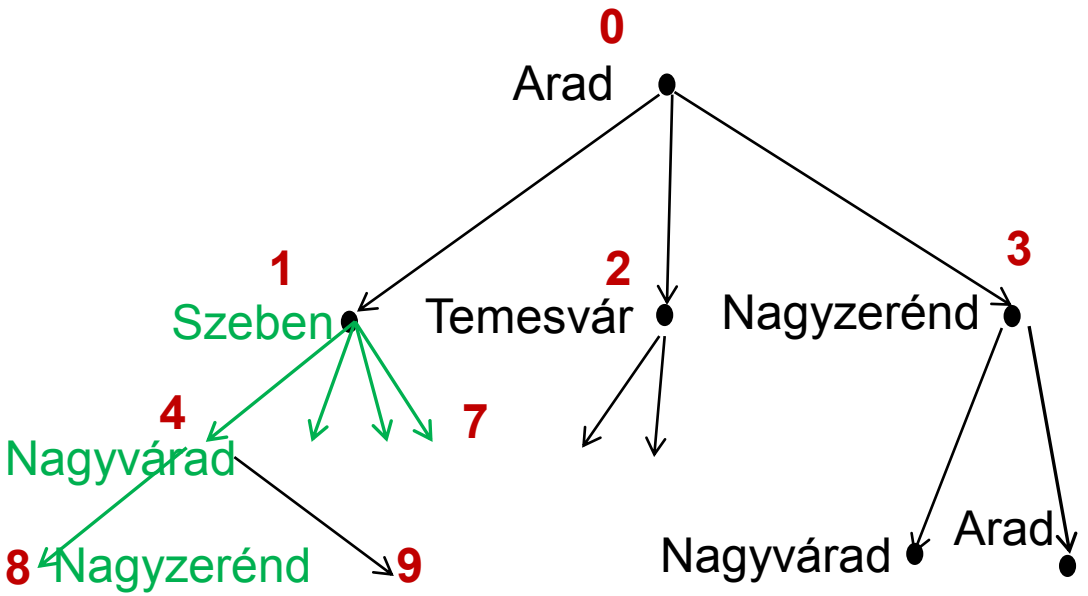
\+ Az oL első elemét vizsgálva a Cél vizsgálat után érdemes megvizsgálni , hogy nem jutottunk-e olyan állapotba (városba), amiben már voltunk (a cL listán szerepel valamelyik csomóponthoz rendelt állapotként). Hurkok kizárása!

Kvíz

Mi változna, ha egyenletes költségű keresésnél nem akkor ellenőriznénk, hogy a Célt elértük-e, amikor elővesszük a csomópontot az open listából, hanem akkor,
amikor létrejön mint a kifejtett csomópont gyermek-csomópontja, mielőtt betesszük az open listába?

A. Semmi, mindig ugyanazt az eredményt kapnánk
B. Elveszítenénk a „teljesség” tulajdonságot
C. Lényegesen megnövekedne a keresés időigénye
D. Lehet, hogy nem optimális megoldásra jutunk

Mélységi keresés



Két listát kezelünk, openL és closedL:

0. Iniciálás:

- mindkettő üres oL={},cL={}
- oL={Csp0}

1. Veszi az oL első elemét, megvizsgálja, hogy elértük-e a Célt. Ha nem, kifejti, majd a keletkező gyermek csomópontokat felveszi oL-re. A vizsgált első elemet beteszi cL-be.

oL={Csp1, Csp2, Csp3}, cL={Csp0}

2. Veszi az oL első elemét, megvizsgálja, hogy elértük-e a Célt. Ha nem, kifejti, majd a keletkező gyermek csomópontokat felveszi oL ***elejére***. A megvizsgált első elemet beteszi cL-be.

oL={Csp4, Csp5, Csp6, Csp7, Csp2, Csp3, }, cL={Csp0, Csp1}

…. ezt ismételjük, amíg a sor első csomópontjának állapota nem lesz a Cél, vagy nem jutunk zsákutcába!

+ a Cél vizsgálat után érdemes megvizsgálni, hogy nem jutottunk-e olyan városba, amiben már voltunk (a cL listán szerepel), Hurkok kizárása!

Mélységkorlátozott keresés

az utak maximális mélységére egy vágási korlátot ad. Ha ezt elértük – visszalépünk, azt az ágat már nem folytatjuk.

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, de ezt tipikusan még sokkal nehezebb kideríteni.)

A mélységi korlát elérésénél 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.
Sőt amennyiben túl kis mélységkorlátot választunk, akkor a mélységkorlátozott keresés még csak teljes sem lesz.

A híd (logikai feladvány)
Egy ködös este 4 hallgató megrekedt a Borzalmas Titkok Szigetén, ahonnan a szárazföldre csak egy rozoga híd vezetett át. Az éjszakát a szigeten életveszélyes volna eltölteni, a rothadó híd egyszerre csak 2 személyt bír el, és sötétben nem járható. Viszont a növekvő sötétséghez csak egy zseblámpájuk volt, melynek eleme még 17 perc világításra volt elég.

Dani, a maratonfutó, 1 perc alatt menne át a túlsó oldalra. Cili, a biciklista, 2 perc alatt végig tudná futni a hidat. A jó tornász Bercel 5 perc alatt kocogná végig. Viszont Andrásnak, az erős dohányosnak, 10 percre lenne szüksége.

Sikerül-e mindannyiuknak megmenekülni a szigetről? Ha nem, miért nem, ha igen, hogyan?

Megoldás: a táblán (mélységkorlátozott kereséssel)

Iteratívan mélyülő keresés

Pár percen belül kvíz!!!

(Slate és Atkin (1977): CHESS 4.5 sakkprogram)

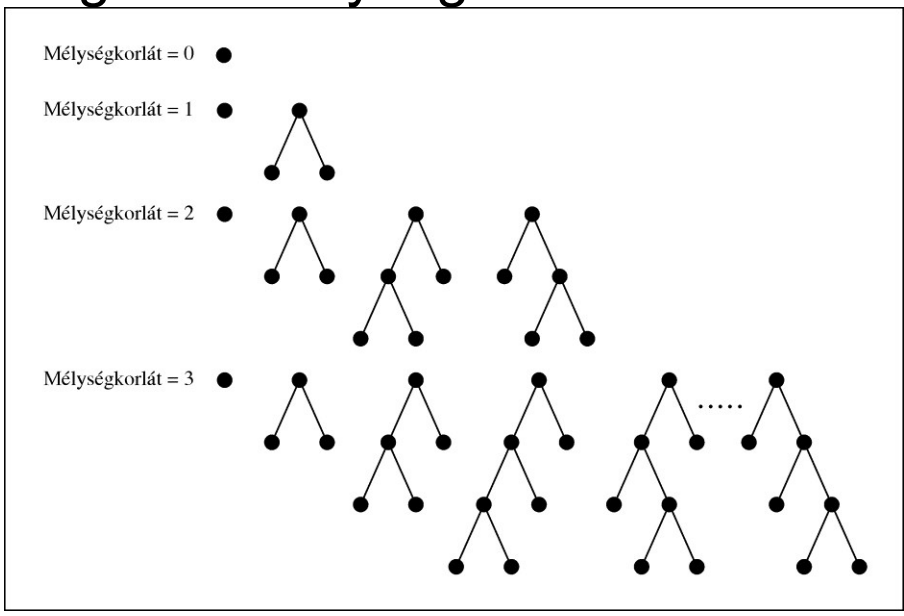
Mélységkorlátozott keresésnél: **hogyan válasszunk jó mélységkorlátot**?

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

Lehetőség: a legjobb mélységkorlát mechanikus kiválasztása:

- kipróbáljuk az összes lehetséges mélységkorlátot: először 0, majd 1, majd 2, stb.
- sorban ezekkel a mélységkorlátokkal végzünk mélységkorlátozott keresést.

Probléma: a nem túl mélyen fekvő állapotokat nagyon sokszor kifejtjük! (időigényes!)



Kvíz

Iteratívan mélyülő keresést végzünk, és a keresési fában minden csomópontunknak 4 gyermek-csomópontja van. Ha 10 mélységben találjuk meg a megoldást (a lehető leghosszabb lépésszám után), akkor hányszor több csomópontkifejtést végeztünk, mint ha egyszerű mélységkorlátozott keresést végeztünk volna 10 mélységkorláttal?

A. Nagyságrendileg másfélszer több csomópontkifejtést
B. Nagyságrendileg tízszer több csomópontkifejtést
C. Nagyságrendileg ezerszer több csomópontkifejtést
D. Nagyságrendileg tízezerszer több csomópontkifejtést

Az 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 memóriaigény**ével rendelkezik.

De bizonyos állapotokat az algoritmus többször is kifejt!

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

d mélységben a megoldás, *b* elágazási tényező: szélességi/mélységi 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 = **111.111**

az iteratívan mélyülő keresésnél a kifejtések teljes száma:

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

b = 10 és *d* = 5 esetén: $6 \cdot 1 + 5 \cdot 10 + 4 \cdot 100 + ... +$ **1· 100.000** = **123.456**

(csak +11,1% „felesleges” növekedés)

Minél nagyobb az elágazási tényező, annál kisebb a többletmunka!

(pl. b = 2 elég rossz, de csak kb. **200%**, kb. *kétszerese* a szélességi/mélységinek!)

Kétirányú keresés

Pár percen belül kvíz!!!

- 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/mélységi keresés = **1.111.111** csomópont,

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

Elméletben nagyon jó, a megvalósítás nem triviális.

Célállapotból hátrafelé keresni? Az *n* csomópont **előd csomópontjai** azon csomópontok, amelyek követő csomópontja lehet *n*.

- A hátrafelé keresés a cél csomópontból indulva az előd csomópontok egymást követő generálását jelenti. Megfordítható-e a folyamat? (Összerakható-e Csernobil a romokból?)
- Pl. sakk, nagyon sok célállapot, melyikből induljunk visszafele?

Kvíz

Optimális, teljes keresést akarunk végezni, nem ismerünk korrekt mélységkorlátot. A legrosszabb esetet feltételezve melyik lesz a legkisebb tárigényű megfelelő módszer?
(pl. b=9, d=12)

A. Mélységi keresés
B. Szélességi keresés
C. Iteratívan mélyülő keresés
D. Kétirányú keresés

A neminformált keresési stratégiák összehasonlítása

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

Jellemző	SzK	EgyKK	MK	MKK	IMK	KK
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.

Krit.	SzK	EgyKK	MK	MKK	IMK	KK
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

Az exponenciális tár-, illetve időigény komoly problémát jelent!
Jobb módszerek kellenek!