

Mesterséges Intelligencia MI

Problémamegoldás kereséssel – a problémára vonatkozó információ felhasználása (heurisztika)

<http://mialmanach.mit.bme.hu/aima/ch04>



Pataki Béla

BME I.E. 414, 463-26-79

pataki@mit.bme.hu,

<http://www.mit.bme.hu/general/staff/pataki>

A neminformált keresési stratégiák – általános (nem problémaszpecifikus) módszerek

b - elágazási tényező,

m - a keresési fa maximális mélysége,

d - a megoldás mélysége,

l - a mélység korlát.

Jellem- ző	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!**

Keresési stratégiák – 2

Heurisztikus, vagy más néven **informált keresés**

Büntessük a hátra (céltól elfele) keresést! De ugyanaz, és könnyebb díjazni az előrekeresést a célállapot irányába.

Ehhez kell valami elképzelés, hogy a cél:

- ***merrefelé és,***
- ***nagyjából milyen messzire fekszik.***

Ez az információ az un. **heurisztika**, **heurisztikus függvény** $h(n)$, amit:

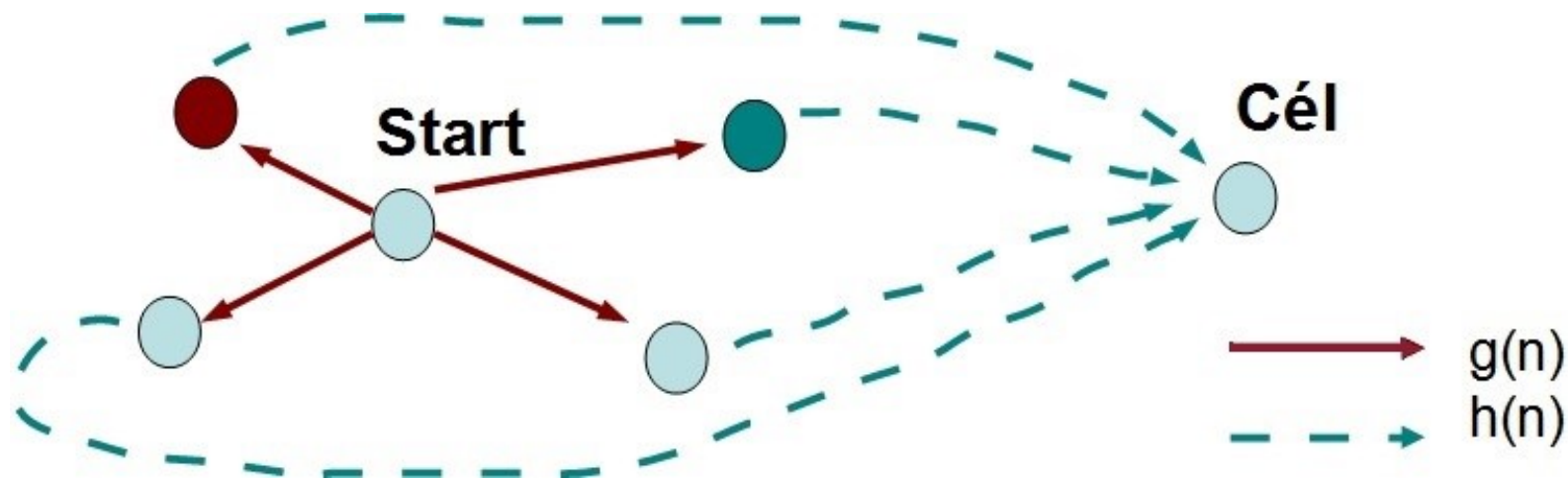
- a probléma minden n állapotára ki kell tudnunk számítani
- kifejezi a célig előrehaladás becsült költségét
- ha pontos, akkor elvben fölöslegessé teszi a keresést (ha nagyon pontatlan, akkor viszont semmit sem segít)

Ez minden problémára más, problémaszpecifikus!

Büntessük a hátra (céltól elfele) keresést!

Ilyen keresés az ún. **heurisztikus**, vagy más néven **informált keresés**. Az ún. **heurisztika**, **heurisztikus függvény**, $h(n)$ becslést ad arra, hogy mekkora a hátralévő út legkisebb költsége. (Ezáltal becslést ad a teljes út legjobb költségére – $f(n)$ -re – is.)

$$f(n) = g(n) + h(n)$$



Jó heurisztikus függvénnnyel a tár- és időigény jelentősen csökkenthető, a csökkentés az adott problémától és a h függvény minőségétől függ.

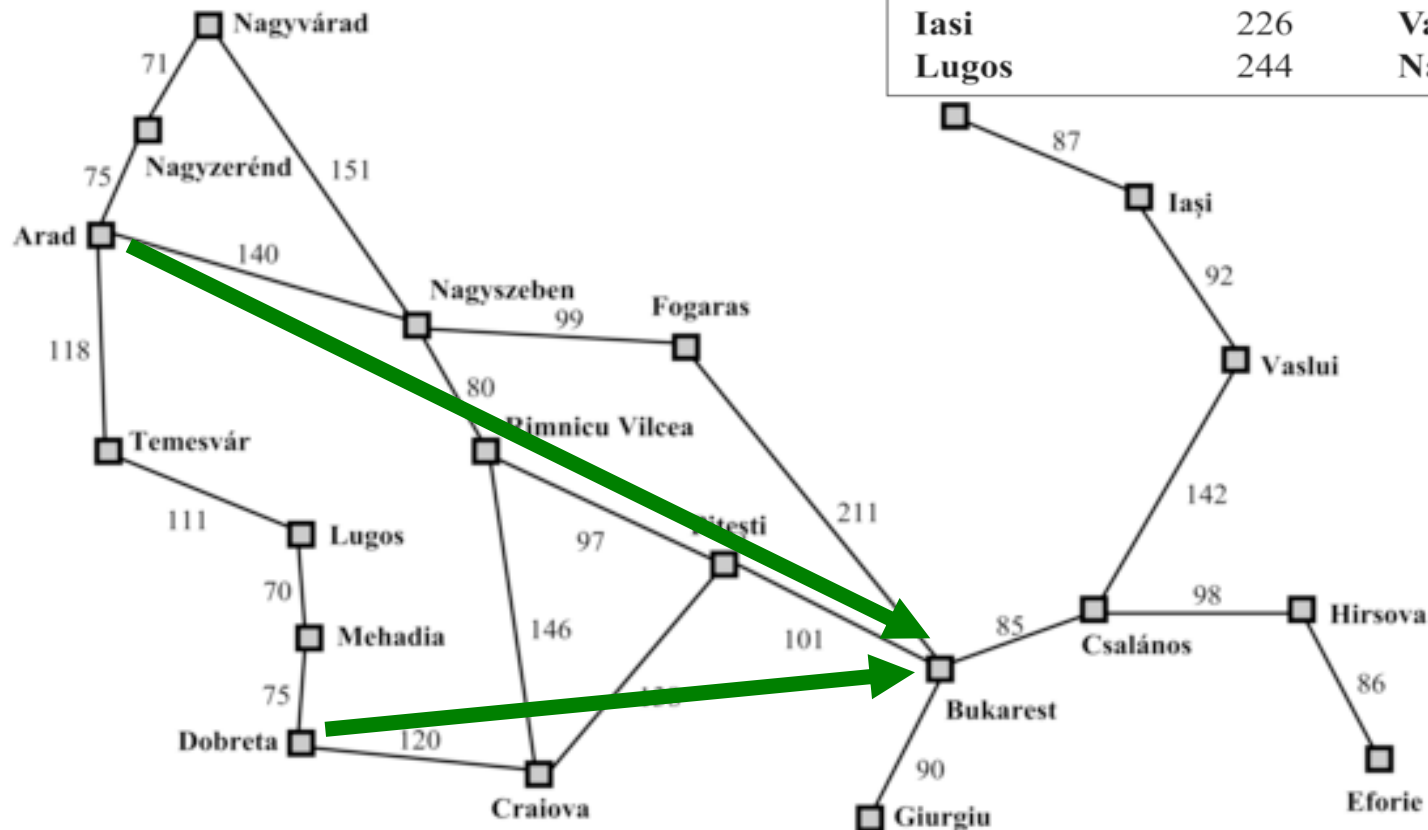
Tökéletes információ (heurisztika) $\Rightarrow$ előretartás elágazások nélkül (hiszen minden csomópontra pontosan tudjuk, hogy mibe kerül ezen az úton menni) $\Rightarrow$ lineáris tárigény = lineáris időigény!

Legyen az útkeresési feladatunkban a heurisztikus függvény a légvonalban mért távolság (h_{LMT})

Az igényeket teljesíti?

Mi mondható a hibájáról?

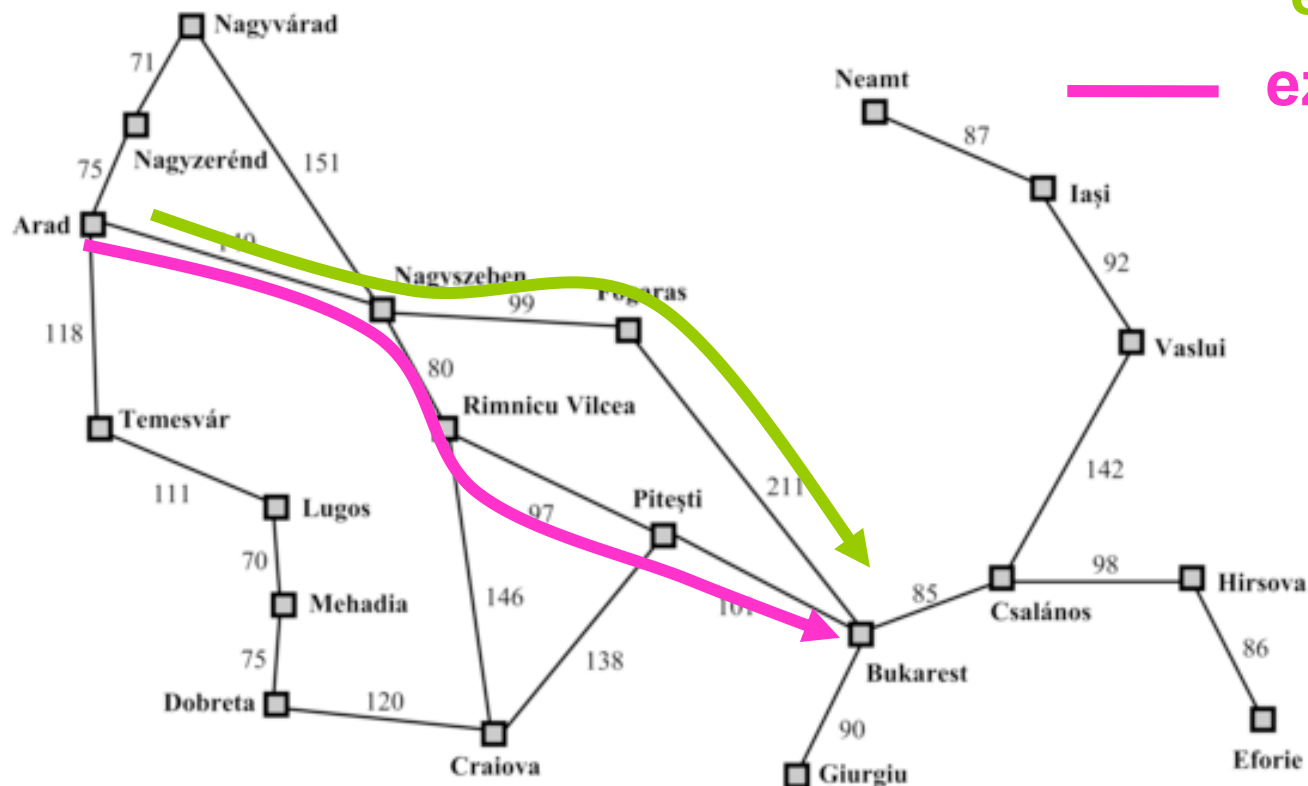
Arad	366	Mehadia	241
Bukarest	0	Neamt	234
Craiova	160	Nagyvárad	380
Dobreta	242	Pitești	100
Eforie	161	Rimnicu Vilcea	193
Fogaras	176	Nagyszeben	253
Giurgiu	77	Temesvár	329
Hirsova	151	Csalános	80
Iasi	226	Vaslui	199
Lugos	244	Nagyzerénd	374



Mohó keresés:

a célhoz legközelebbinek tűnő csomópont először (a legjobbnak becsültet először)

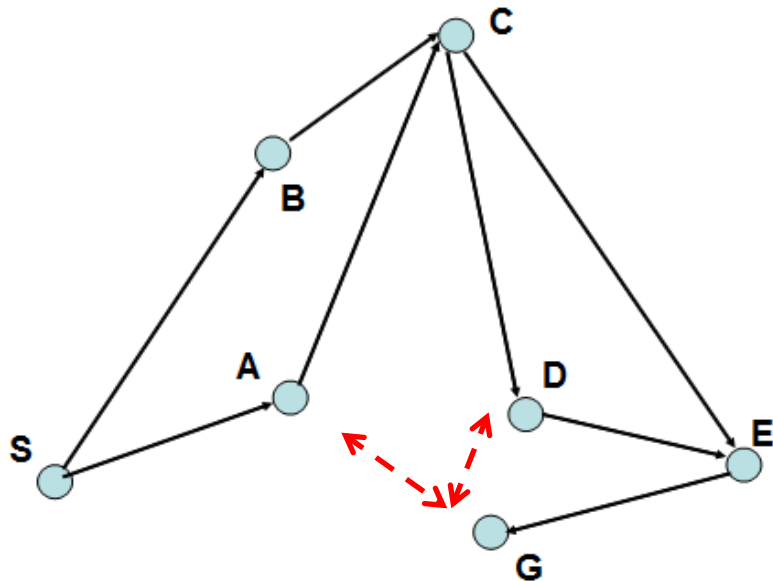
Stratégia: a következő lépésben azt a csp-t fejti ki, amelyhez rendelt probléma-állapotot a legközelebbinek ítéli a célállapothoz (legkisebb az n -dik csp-hoz rendelt $h(n)$ heurisztikus érték).



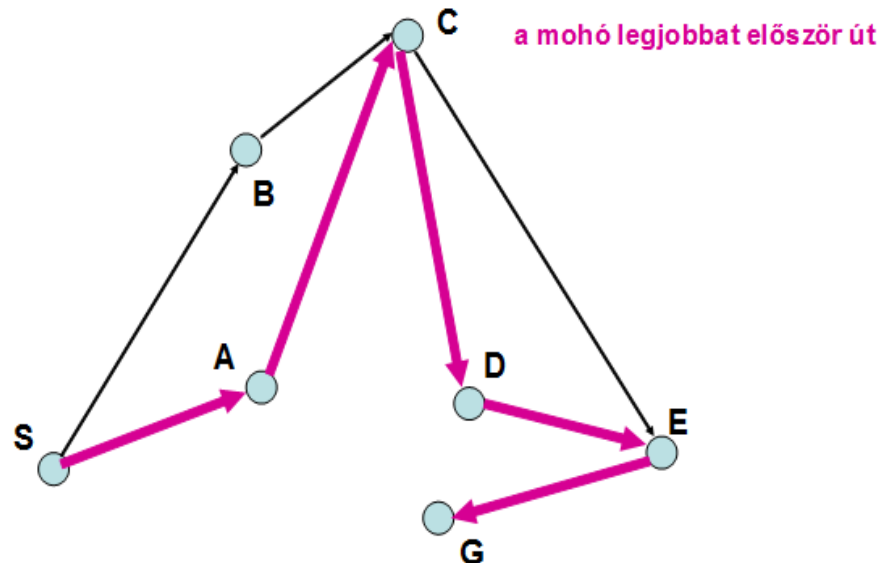
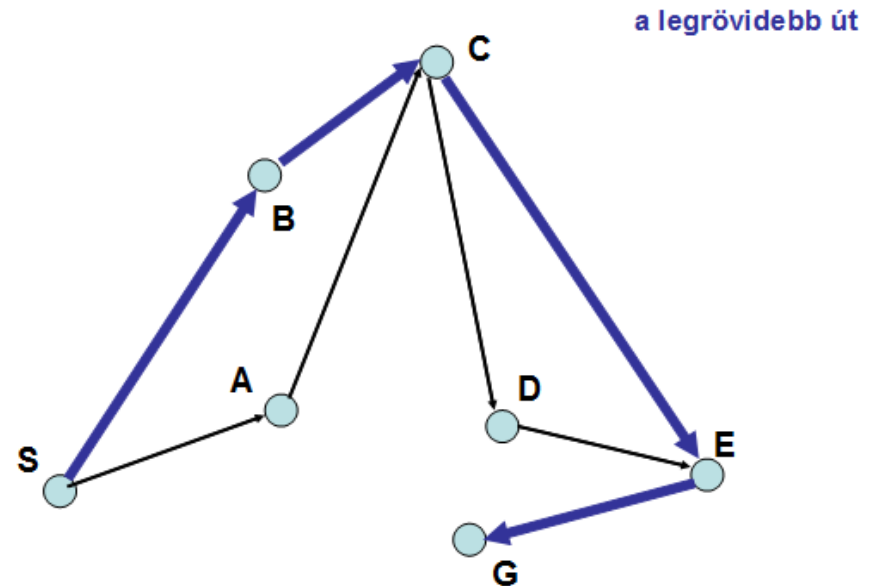
— ezt kaptuk

— ezt kellene kapni

Mohó algoritmus általában **gyorsan** megtalálja a megoldást, de **nem mindig az optimális** megoldást találja meg. A mohó keresés érzékeny a hibás kezdő lépésekre is.



Probléma: A és D légvonalban nagyon közel van a G célhoz, de az úthálózaton nem



Mohó keresés

mélységi keresésre hasonlít, egyetlen út végigkövetését preferálja a célíg, zsákutcából visszalép.

Ua. a problémák, mint a mélységi keresésnél: nem optimális, nem teljes (elindul egy végtelen úton, és ez esetben nem tér vissza új lehetőséget kipróbálni).

Az összes csomópontot a memóriában tartja: ezért a legrosszabb (worst-case) idő- és tárigény: $O(b^m)$ – ha m a problémátér mélysége

Persze jó heurisztikus függvénnel a tár- és időigény jelentősen csökkenthető, a csökkentés az adott problémától és a h függvény minőségétől függ.

Szóval a mohó keresés ígéretes, egyszerű, sokszor akár használják is, de mégsem igazán jó.

A* keresés: a teljes útköltség minimalizálása

(de facto standard az útkeresésben)

mohó: $\min \{h(n)\}$ a célhoz vezető útköltség becslőjét minimalizálja
nem teljes (nem optimális)

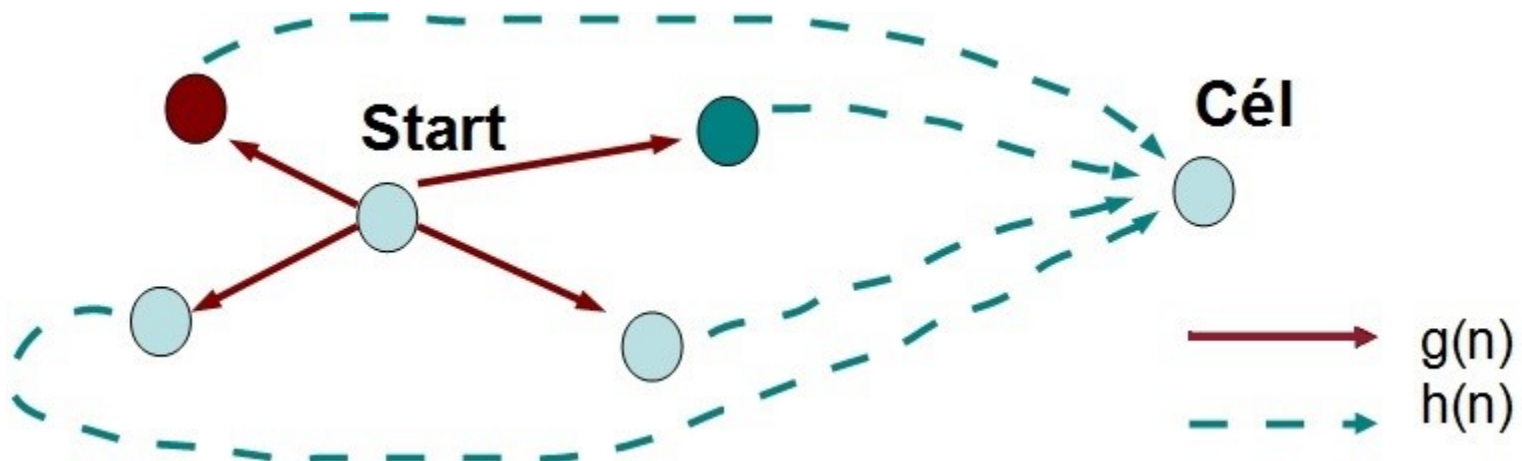
egyenletes költségű: $\min \{g(n)\}$ a megtett út költségét minimalizálja
optimális (egyben **teljes** is), de nagyon **rossz hatékonyságú**

a két stratégia ötvözése: $\min \{f(n)\} = \min \{h(n) + g(n)\}$

$g(n)$: a kiinduló cs-ponttól az n cs-pontig számított út **tényleges** költsége

$h(n)$: az n cs-ponttól a célba vezető legolcsóbb költségű út **becsült** értéke

$f(n)$: a kiinduló csp-tól az adott n csp-on át a célba vezető legolcsóbb költségű út **becsült** értéke



Elfogadható heurisztika = Ha a h függvény soha nem becsüli felül a cél eléréséhez szükséges költséget (**optimista**: a cél közelebbinek legalább is nem távolabbinak tűnik, mint amilyen)

(Optimista heurisztika haszna: lásd Lázár Ervin A hazudós egér c. meséje - a Farkasnak és a Kosnak adott heurisztika)

Ha h elfogadható, akkor $f(n)$ sohasem becsüli túl az n -dik csomóponton át vezető legjobb megoldás valódi költségét

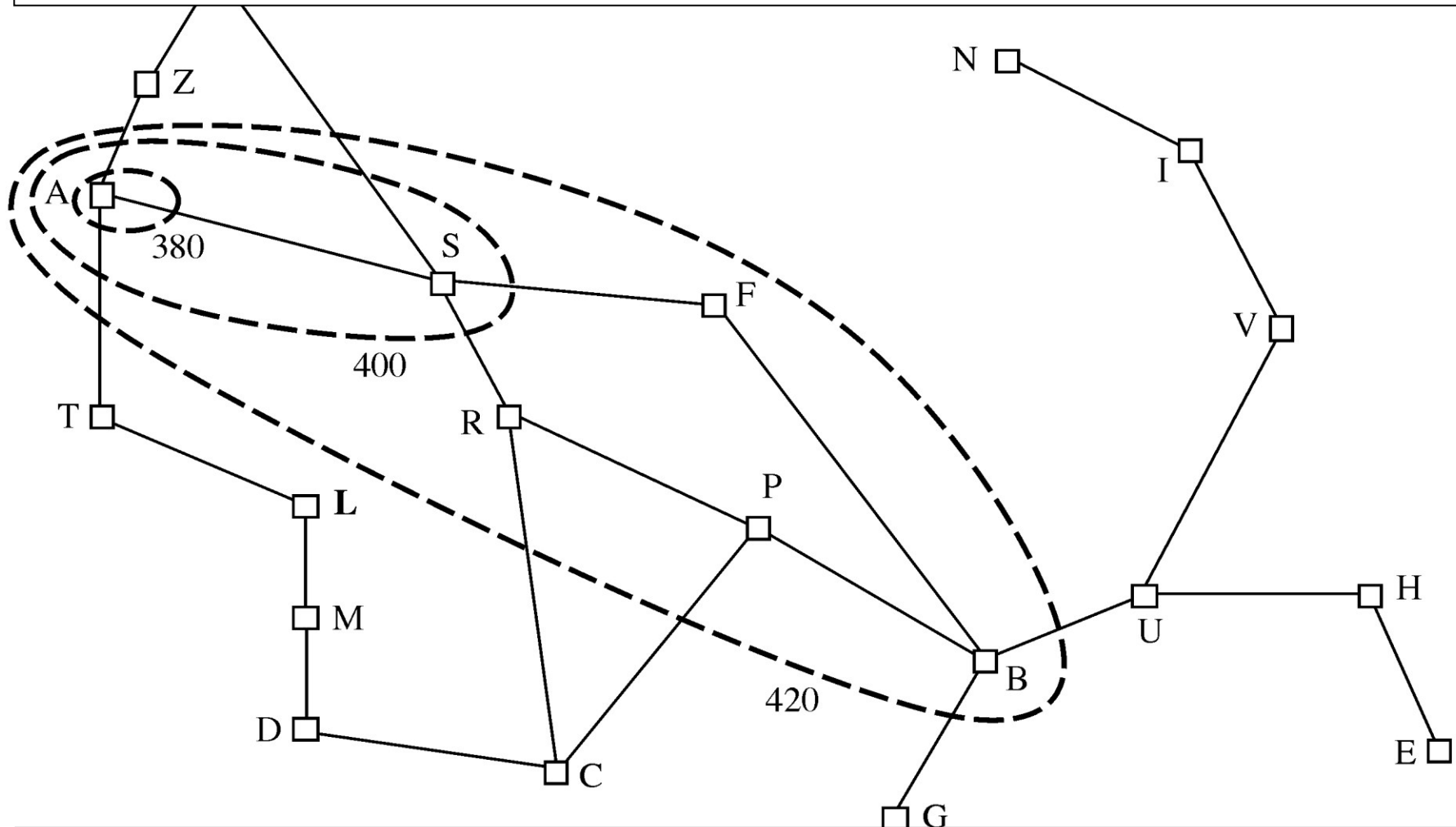
A* keresés:
az f függvényt alkalmazó legjobbat-először keresés, ahol h elfogadható

Ha a gyökérből nézve egyetlen út f értéke sem csökken – a heurisztika **monoton nő**. Egy heurisztikus függvény aka. monoton, ha teljesíti a háromszög egyenlőtlenséget (**konzisztens heurisztika**, h_{LMT} ilyen).

Monotonná tehető, a trükk: ha az n' gyermekcsp-ra $f(n')$ kisebb lenne, mint a szülőé (n csp) , akkor a szülő $f(n)$ értékét használhatjuk n' -re is: $n \Rightarrow n' \quad f(n') = \max(f(n), g(n') + h(n'))$

Ha f a gyökérből kiinduló utak mentén soha sem csökken $\Rightarrow$ **határvonalak** rajzolhatók

Egyenletes költségű keresés: olyan A^* keresés lenne, amelynél $h = 0$ minden csomópontra, ezért a vizsgálandó csp-sávok a kiinduló csp köré húzott többé-kevésbé koncentrikus „körök”. Pontosabb heurisztikus függvény esetén: az A^* keresésnél sávok a **cél-állapot felé nyúlnak, fókuszálódnak az optimális út körül.**



Az A^* teljes és optimális

$$f(G_2) = g(G_2) > g(G) = f^* \geq f(n)$$

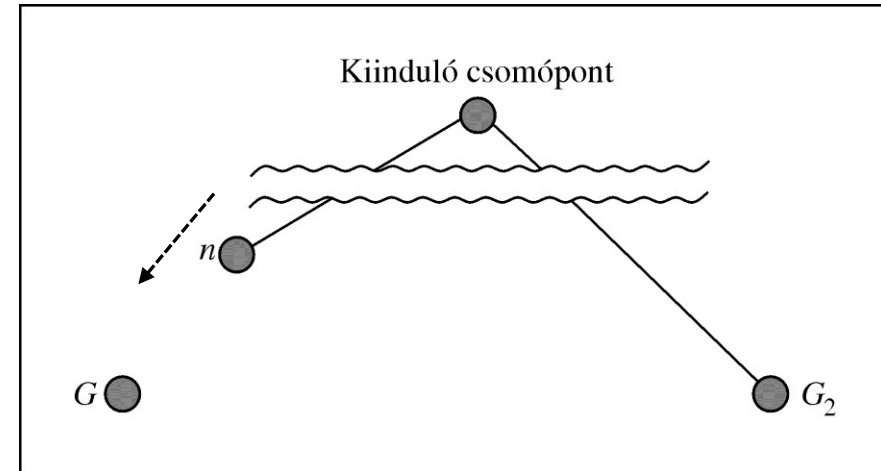
G : opt. cél $f^* = g(G)$ kltsg, $g(G_2)$ szubopt. cél, a G fele vezető úton n nem lett kifejtve G_2 előtt

Ha f^* az **optimális** megoldási út költsége, akkor:

- A^* kifejti az összes $f(n) < f^*$ csp-t
- ezek után egy cél-csp kiválasztása előtt még kifejthet néhány csp-t a „cél-határvonalon”, amelyekre $f(n) = f^*$

Az ilyen típusú optimális algoritmusok közül az A^* keresés bármely adott heurisztikus függvény mellett **optimális hatékonyságú**. Egyetlen optimális algoritmus sem fejt ki **garantáltan kevesebb** csomópontot az A^* keresés által kifejtett csomópontnál.

Pár percen belül kvíz!!!



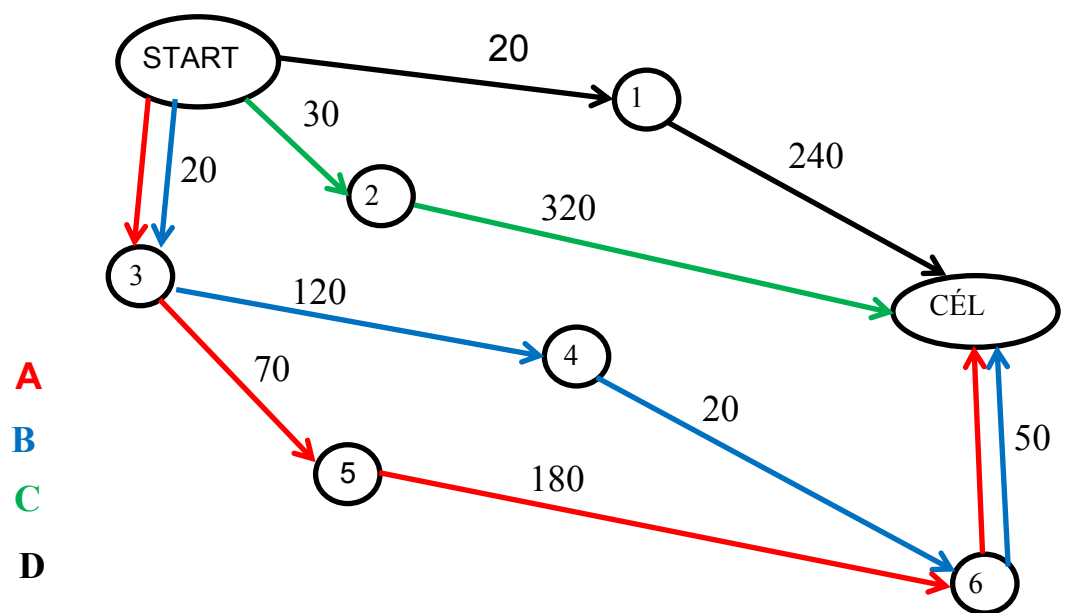
A^* teljessége - pontosan: az A^* algoritmus **lokálisan véges gráfokon** – (véges elágazási tényező) teljes, ha létezik δ pozitív konstans, amelynél semelyik operátor költsége sem kisebb.

(ki akarjuk kerülni a véges határértékkel rendelkező végtelen számú infinitezimális mennyiségekkel kapcsolatos problémákat, azonban a probléma formalizálása rajtunk múlik, és vessen magára, aki ilyenek formalizálná a problémát)

Kvíz

Az alábbi keresési feladatban A* keresést végzünk. Az élek mentén az állapotok „távolságát” (az állapotátmenethez tartozó tényleges költséget) tüntettük fel. Ahol két párhuzamos eltérő színű élt lát, az csak azt jelenti, hogy két potenciális megoldásnak is része az az állapotátmenet. A használt heurisztika:

START→CÉL	1→CÉL	2→CÉL	3→CÉL	4→CÉL	5→CÉL	6→CÉL	CÉL→CÉL
150	120	270	160	66	200	49	0



Melyik útvonalat találja meg az algoritmus? (Az A, B, C, D válaszokat az ábrán színnel jelöltük.)

Az A* algoritmus komplexitása

(**teljes, optimális** és az összes ilyen közül **optimálisan hatékony**)

Buktató: a legtöbb esetben a csp-tok száma a keresési tér cél-határvonalán belül a megoldás hosszának még mindig **exponenciális** függvénye.

Exponenciális - **kivéve**, ha a heurisztikus függvény hibája legfeljebb az aktuális útköltség logaritmusával nő:

$$|h(n) - h^*(n)| \leq O(\log(h^*(n))), \quad (h^*(n) \text{ a valódi költség})$$

Majdnem minden, gyakorlati heurisztikus függvény esetén a hiba legalább arányos az út költséggel - exponenciális növekedés.

Egy jól megválasztott heurisztikus függvény ettől függetlenül a nem informált keresési algoritmushoz képest jelentős megtakarítást eredményezhet!

A* algoritmus nagy problémája: az összes legenerált csomópontot eltárolja, lényegesen hamarabb felemészti a memóriát, mintsem kifutna az időből.

Heurisztikus függvények létrehozása

Kvíz következik!

Nagyon jól bemutatatható egy demóproblémán, 8-as tili-toli logikai játék:

- kb. 20 lépésben lehet átlagosan megoldani,
- $b \sim 3$,
- kimerítő keresés $\Rightarrow$ kb. $3^{20} = 3.5 \times 10^9$ állapot

Elfogadható heurisztikus függvény kell!

5	4	
6	1	8
7	3	2

Kiindulóállapot

1	2	3
8		4
7	6	5

Célállapot

Heurisztikus függvények létrehozása (legyen sok és jó)

8-as tili-toli: kb. 20 lépés, $b \sim 3$, kimerítő keresés = kb. $3^{20} = 3.5 \times 10^9$ állapot

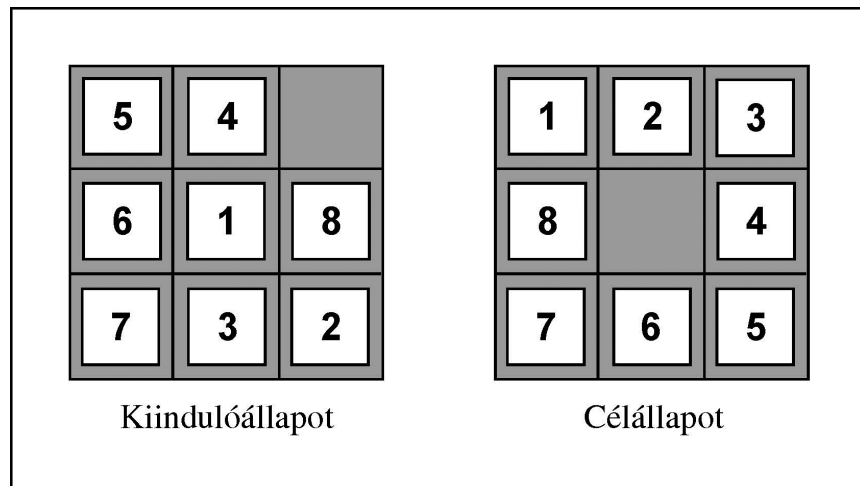
gyorsan és a legrövidebb megoldás?

- elfogadható heurisztikus függvény kell!

$h1$ = **a rossz helyen lévő lapkák száma.**

elfogadható: minden rossz helyen lévő lapkát legalább egyszer mozgatni kell

$h2$ = **a lapkák céltól mért (vízsz. és függ.) távolságainak összege: háztömb- vagy Manhattan-távolság**



Kvíz A $h1$ és $h2$ heurisztika esetén melyik állítás igaz?

- A. A $h1$ mindig pontosabb becslés a $h2$ -nél
- B. A $h2$ mindig pontosabb becslés a $h1$ -nél
- C. Időnként az egyik, időnként a másik pontosabb becslés
- D. Csak a $h1$ heurisztika elfogadható, mert $h2$ nem minden esetben optimista

Heurisztikus függvények létrehozása (legyen sok és jó)

8-as tili-toli: kb. 20 lépés, $b \sim 3$, kimerítő keresés = kb. $3^{20} = 3.5 \times 10^9$ állapot

gyorsan és a legrövidebb megoldás?

- elfogadható heurisztikus függvény kell!

$h1$ = a rossz helyen lévő lapkák száma.

elfogadható: minden rossz helyen lévő lapkát legalább egyszer mozgatni kell

$h2$ = a lapkák céltól mért (vízsz. és függ.)

távolságainak összege: háztömb- vagy Manhattan-távolság

elfogadható: minden egyes mozgatással egy lapkát csak egy (vízszintes és függőleges) lépéssel lehet közelebb vinni a célhoz.

.....

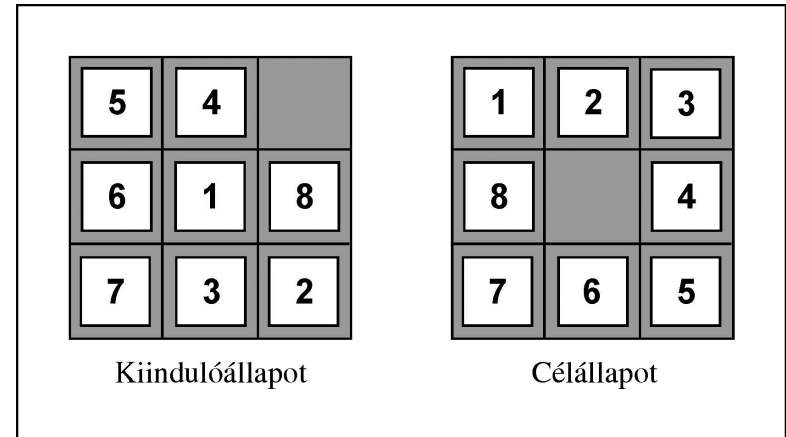
$$h1 = 1 + 1 + 1 + 1 + 1 + 1 + 1 = 7$$

$$h2 = 2 + 3 + 3 + 2 + 4 + 2 + 0 + 2 = 18$$

$$h3 = \dots = \dots ?$$

Jó lesz pl.:

**$h3 = (h1 + h2)/2$?
stb.**



Heurisztikus függvény pontossága és hatékonysága

A heurisztikus függvények minősítése (általában nem könnyű feladat a minősítés, a megoldások összehasonlítása – **skalár mérőszám kell!**): b^* **effektív elágazási tényező**

ha A^* által kifejtett összes csomópont száma N , a megoldás mélysége d , akkor b^* annak a d mélységű kiegyensúlyozott fának az elágazási tényezője, amely N csomópontot tartalmazna:

$$N = 1 + b^* + (b^*)^2 + \dots + (b^*)^d$$

(pl. 5 mélységben fekvő megoldás 52 cs-pont kifejtésével:

az effektív elágazási tényező 1.91)

$$1 + 1^1 + 1^2 + 1^3 + 1^4 + 1^5 = 6 < 52 < 63 = 1 + 2 + 2^2 + 2^3 + 2^4 + 2^5$$

Egy adott heurisztikus függvény által generált fa effektív elágazási tényezője általában nagyjából állandó egy adott problémaosztály számos egyedére.

A b^* kis számú problémahalmazon végzett kísérleti mérése jó becslés.

Egy jól megtervezett heurisztikus függvény effektív elágazási tényezője **1 körüli érték**. (Ideális lenne az 1.)

Heurisztikus függvény: pontosság és hatékonyság

a ***h1***, ***h2*** heurisztikus függvények tesztelése:

vajon a ***h2*** mindig jobb-e, mint a ***h1***? **Igen** (miért?).

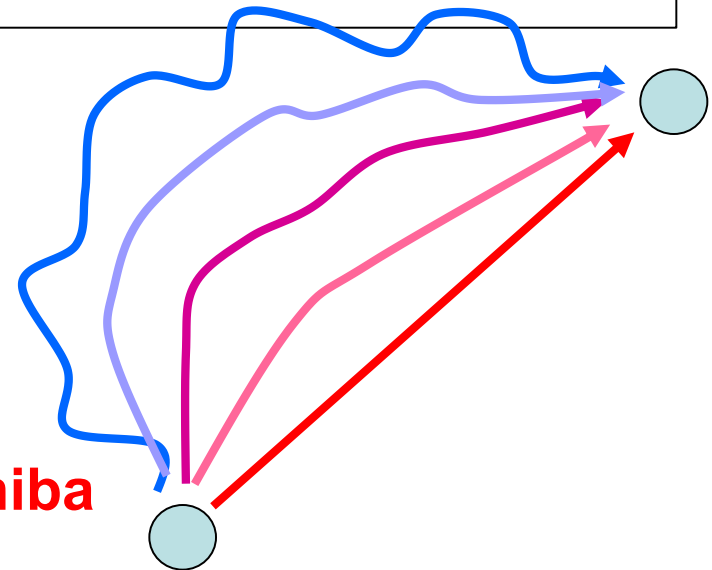
minden n csp-ra ***h2*** (n) $\geq$ ***h1*** (n):

h2 **dominálja** ***h1***-et, **dominancia $\Rightarrow$ hatékonyság**

a ***h2***-t használó A^* kevesebb csp-t fog kifejtetni, mint a ***h1***-et használó!

Mindig jobb, ha nagyobb értékeket adó heurisztikus függvényeket alkalmazunk, amíg nem becsüljük túl a valódi költséget.

-  **Tényleges**
-  **Elfogadható, kis hiba**
-  **Elfogadható, közepes hiba**
-  **Elfogadható, durva hiba**
-  **Elfogadható, nagyon durva hiba**



h1 és ***h2*** tesztelése (8-as kirakójáték) : 100-100 random problémapéldány, 2, 4, ..., 24 mélységű megoldással, ***A****, illetve a neminformált iteratívan mélyülő keresés

	Keres. ktg.			b*		
<i>d</i>	IMK	<i>A*(h1)</i>	<i>A*(h2)</i>	IMK	<i>A*(h1)</i>	<i>A*(h2)</i>
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	364404	227	73	2.78	1.42	1.24
14	3473941	539	113	2.83	1.44	1.23
16	-	1301	211	-	1.45	1.25
18	-	3056	363	-	1.46	1.26
20	-	7276	676	-	1.47	1.27
22	-	18094	1219	-	1.48	1.28
24	-	39135	1641	-	1.48	1.26

(Jó) Heurisztikus függvények kitalálása

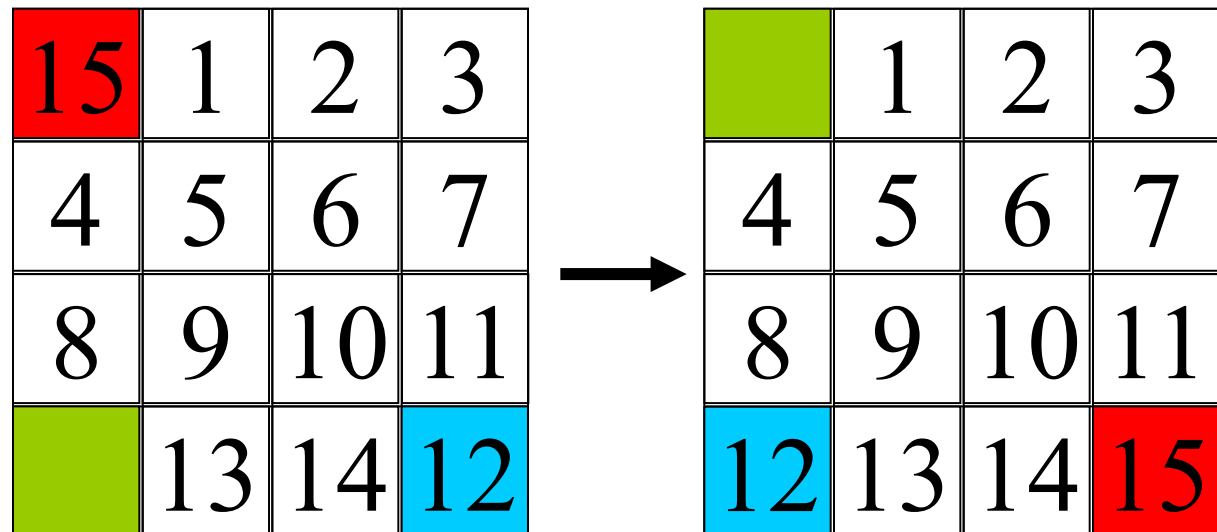
Szabály: Egy lapka az **A-ról a B-re mozgatható**, ha **A és B szomszédok** és **a B mező üres**.

Ha egy vagy több feltételt törölünk, akkor ún. relaxált problémákat hozhatunk létre (lazítunk - nem minden szabályt tartunk be)

- (a) Egy lapka az A-ról a B-re mozgatható, **ha A és B szomszédosak** (h2 betartja, h1 nem)
- (b) Egy lapka az A-ról a B-re mozgatható, **ha a B mező üres** (sem h1 sem h2 nem törődik vele)
- (c) Egy lapka az A-ról a B-re mozgatható (mindkettő betartja)

csak (a): $6 + 3 = 9$ lépés

csak (c): $1 + 1 = 2$ lépés



(Jó) Heurisztikus függvények kitalálása

Relaxált probléma: olyan probléma, amelyben az operátorokra kevesebb megkötést teszünk, mint az eredeti problémában.

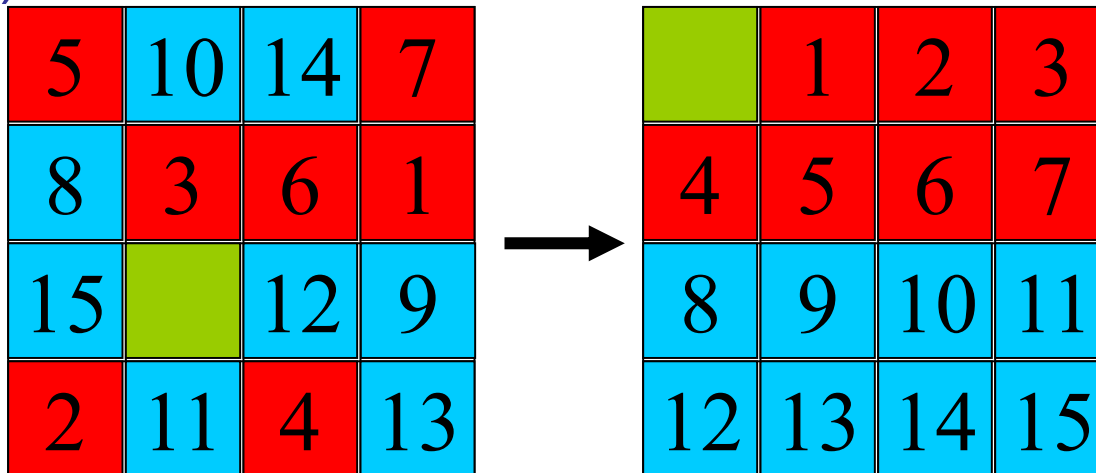
Gyakori, hogy a relaxált probléma pontos megoldásának költsége jó heurisztikus függvény az eredeti problémára.

Gyakori, hogy – a relaxált probléma egyszerűsége miatt – az optimális megoldás a relaxált feladatra analitikusan is meghatározható.

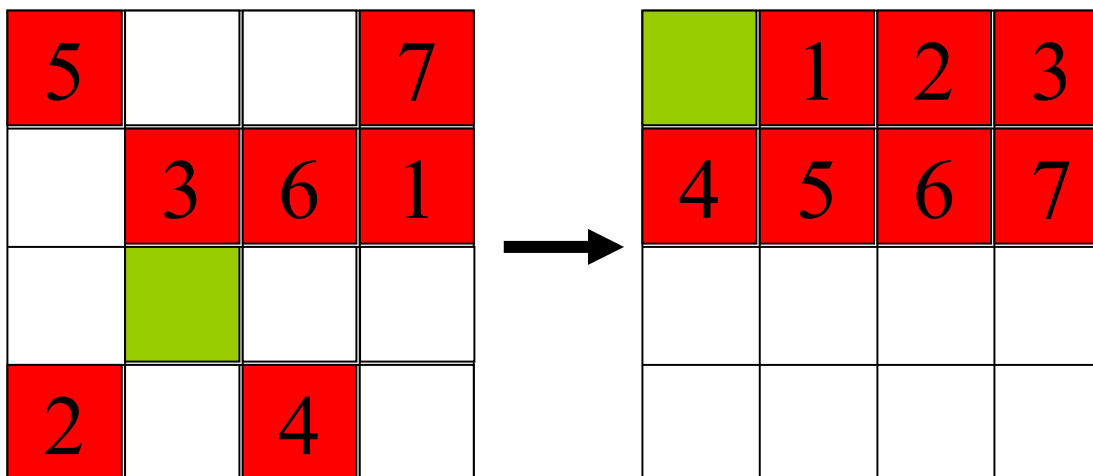
A relaxált probléma pontos megoldása mindig egy elfogadható heurisztika a kevésbé relaxált problémára.

(Jó) Heurisztikus függvények kitalálása

h3: (Disjoint) Pattern Database Heuristics



Piros lapkák megoldása = 20 lépés

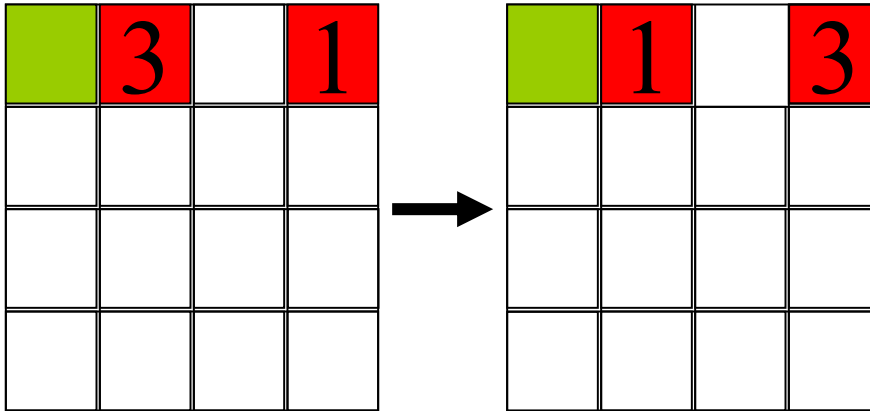


Kék lapkák megoldása = 25 lépés

Teljes heurisztika: $h3 = 20 + 25 = 45$ lépés

(Jó) Heurisztikus függvények kitalálása

h4: Linear Conflict Heuristics



$$h2 = 2 + 2 = 4$$

de a lapkák egymással szemben nem tolhatók

$$h4 = h2 + 2 \text{ (kikerülés)}$$

$h4 = h2 + 2 \times$ (sorbeli lineáris konfliktus, minden sorra
+ oszlopbeli lineáris konfliktus, minden oszlopra)
(h2: a lapkák céltól mért (vízsz. és függ.) távolságainak összege)

(Jó) Heurisztikus függvények kitalálása

h5: Gaschnig's Heuristics

(b) Egy lapka az A-ról a B-re mozgatható,

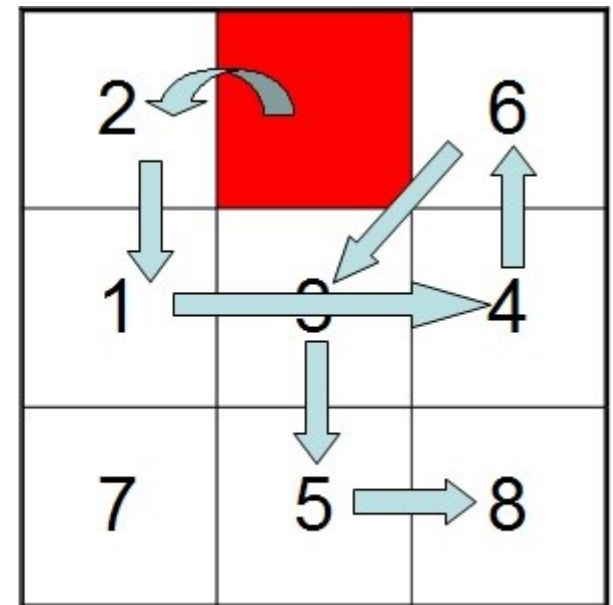
ha a B mező üres.

elfogadható, jó becslés

2		6
1	3	4
7	5	8



1	2	3
4	5	6
7	8	



A „lyuk” mozgása (pl. a 3 átlós mozgása szabálytalan lenne)

(Jó) Heurisztikus függvények kitalálása

Nehéz felismerni a „nyilvánvalóan legjobb” heurisztikus függvényt.

Ha egy problémához adottak a $h_1, \dots, h_m$ elfogadható heurisztikák és semelyik sem dominálja a másikat, melyiket kellene választanunk?

Nem kell választanunk. A lehető legjobb:

$$h(n) = \max\{ h_1(n), \dots, h_m(n) \}$$

mindig azt a függvényt használja, amelyik az adott csomópontra a **legpontosabb**. Mivel mind elfogadhatóak, ezért h is **elfogadható** lesz. h **dominálja** az összes heurisztikus függvényt.

Probléma: az elemezhetőség, hiszen a konkrét h mindig változik! (...mikor h_3 adja az eredő h -t, mikor h_7 stb.)

Ha a heurisztikus függvény olyan összetett, hogy értékének egy csp-ra történő meghatározása sok időt vesz igénybe, akkor baj van.

Egy jó heurisztikus függvény: pontos és hatékonyan számítható

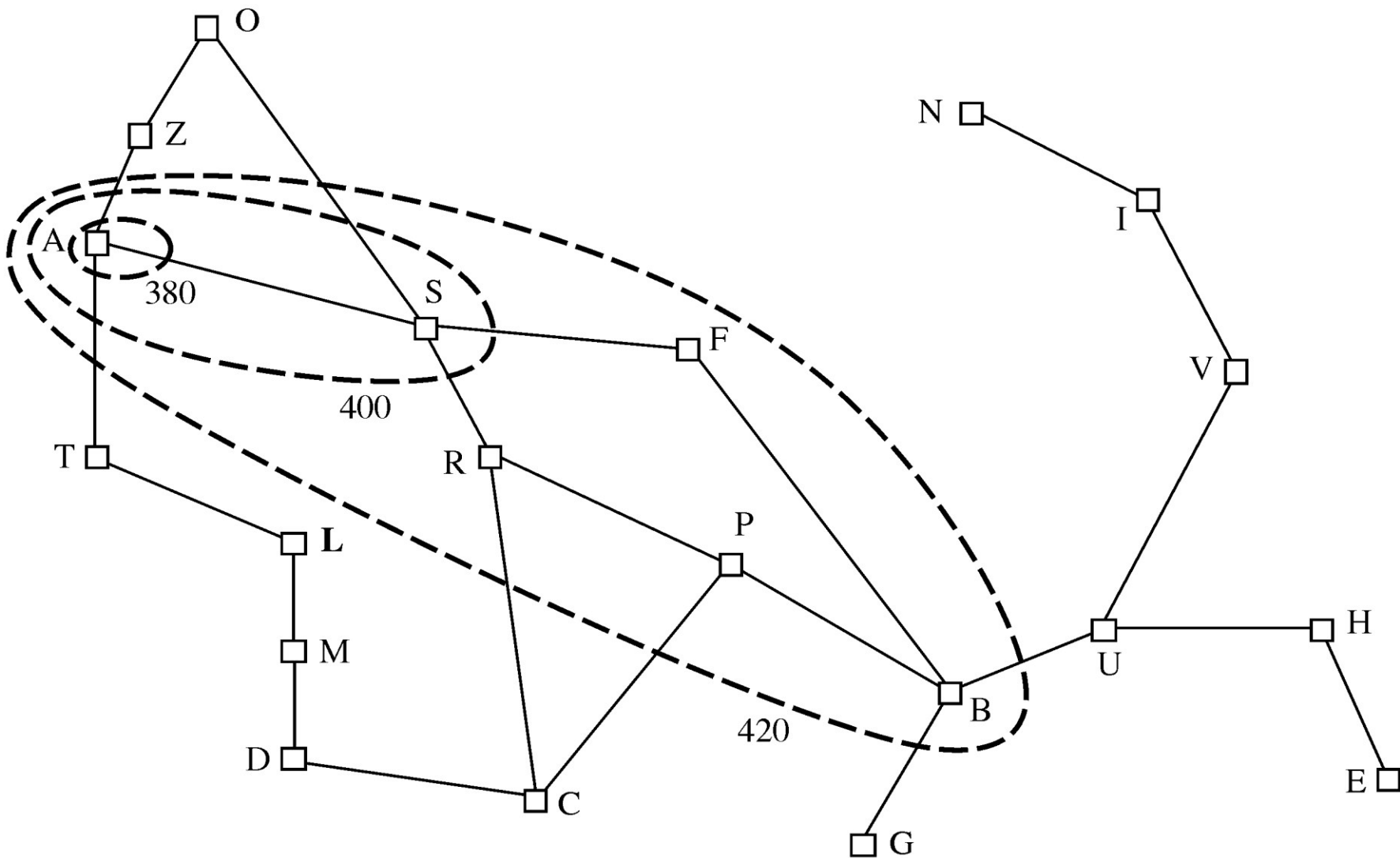
Iteratívan Mélyülő A* keresés (IMA*)

Minden egyes iteráció egy mélységi keresés, mélységkorlát helyett egy ***f***-költség korláttal.

Minden egyes iteráció kifejtí az összes - az adott ***f***-költség határvonalon belül fekvő - csomópontot, átnézve a határvonalon, hogy megtalálja, hol fekszik a következő határvonal.

Az IMA* ugyanazokkal a kitételekkel **teljes** és **optimális**, mint az A* algoritmus, de a **mélységi keresés jellege** miatt csak a leghosszabb felderített út hosszával arányos memóriát igényel.

Iteratívan Mélyülő A* keresés (IMA*)



Iteratívan Mélyülő A* keresés (IMA*)

Az IMA algoritmus időigénye erősen függ attól, hogy a heurisztikus függvény hány különböző értéket vehet fel.*

Pl. a 8-as kirakójáték:

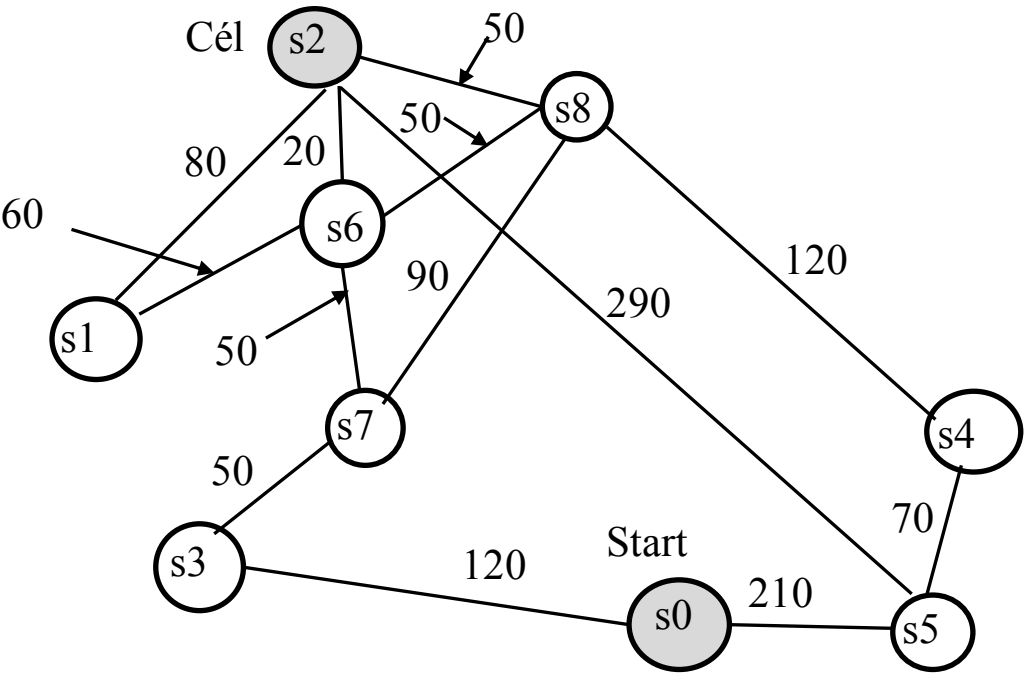
Manhattan távolság csak néhány egész értéket vesz fel.
 f tipikusan csak 2-3-szor növekszik bármely megoldási út mentén.

Ezért az IMA* algoritmus csak 2-3 iterációt végez, és az időhatékonysága az A*-hoz hasonló. Az IMA* utolsó iterációja általában durván ugyanannyi csomópontot fejt ki, mint az A* (időkomplexitás!).

Az IMA* algoritmus összetettebb problématerületekkel nehezebben boldogul (ha heurisztikus függvény értéke minden állapotra más és más – sok iteráció kell). Pl. térképnél szinte minden egyes lépésben újra iterál.

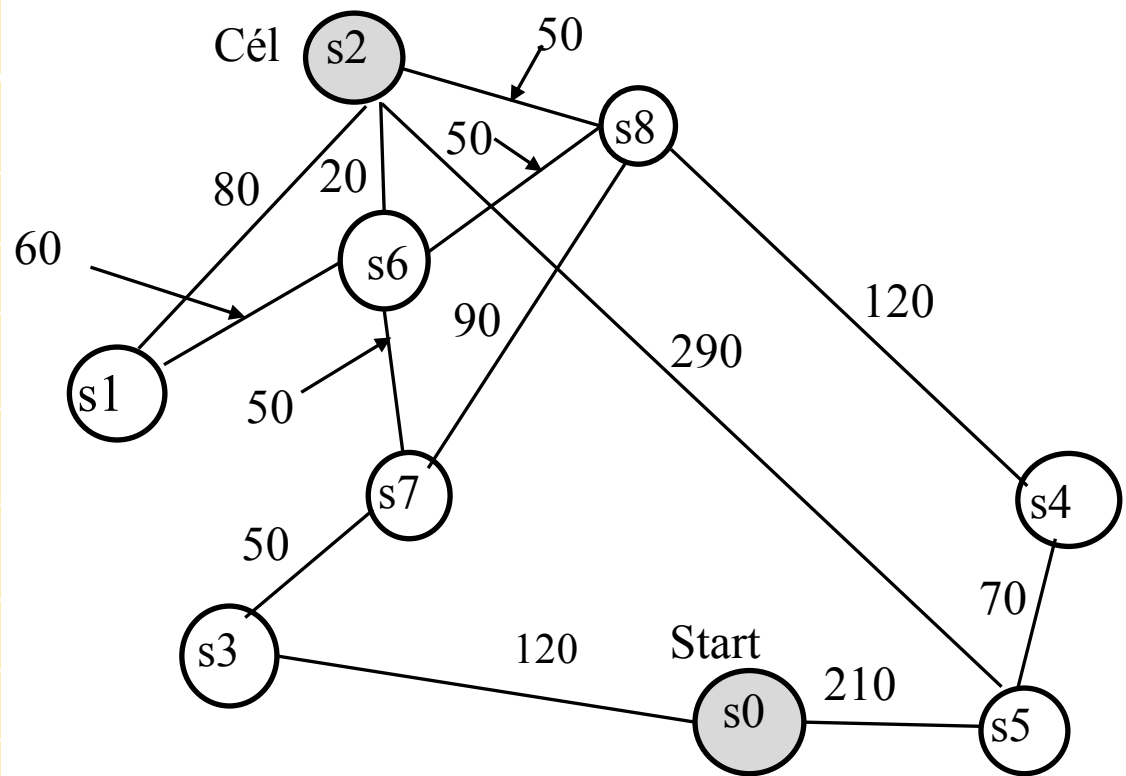
Tavalyi zh példa: Az alábbi problémát A* kereséssel oldjuk meg, nem lépünk vissza abba az állapotba, ahonnan érkezünk. A mellékelt táblázat mutatja a heurisztikánk egyes állapotokhoz tartozó értékeit:

Állapot	$h(s_n)$
s0	210
s1	70
s2	0
s3	110
s4	155
s5	285
s6	18
s7	65
s8	50



Az ágens két listát épít, az elsőben azok a csomópontok szerepelnek, amiket már kifejtett, a másodikban azok, amelyekhez már eljutott, de még nem fejtette ki ezeket. Mindegyik listaelem 5 mezőből épül fel: (szülőcsomópont, aktuális csomópont, állapot, eddig megtett út költsége, az akt. csomópont heurisztika értéke) például a gyökércsomópontra: (-,cs0,s0,0,210).

Állapot	h(sn)
s0	210
s1	70
s2	0
s3	110
s4	155
s5	285
s6	18
s7	65
s8	50



Az ágens két listát épít, az elsőben azok a csomópontok szerepelnek, amiket már kifejtett, a másodikban azok, amelyekhez már eljutott, de még nem fejtette ki ezeket. Mindegyik listaelem 5 mezőből épül fel:

(szülőcsomópont, aktuális csomópont, állapot, eddig megtett út költsége, az akt. csomópont heurisztika értéke), például a gyökércsomópontra:
(-,cs0,s0,0,210).

A második lépés után a következők vannak a két listán

Lista1={(-,cs0,s0,0,210), (cs0,cs1,s3,120,110)}

Lista2={(cs0,cs2,s5,210,285), (cs1,cs3,s7,170,65)}

Adja meg a következő lépés után kialakuló listát!

