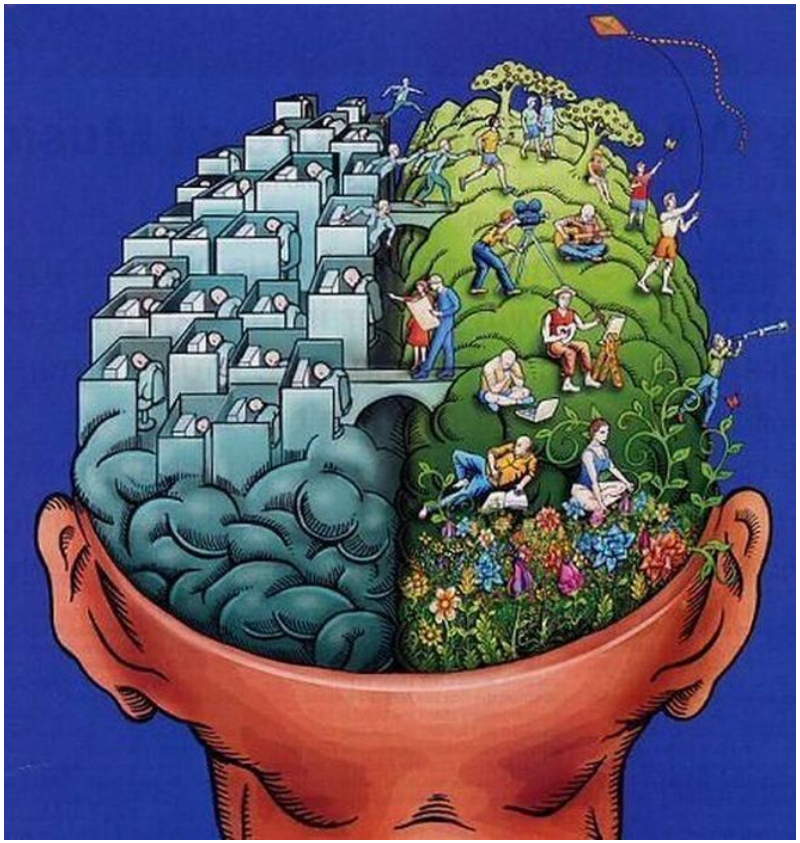
Mesterséges Intelligencia MI

Problémamegoldás
kereséssel -
ha segítenek
útjelzések



Dobrowiecki Tadeusz
Eredics Péter, és mások

BME I.E. 437, 463-28-99
dobrowiecki@mit.bme.hu,
http://www.mit.bme.hu/general/staff/tade

Büntessük a hátrakeresést!

De ugyanaz és könnyebb **díjazni az előrekeresést** a célállapot irányába.

Ehhez kell valami elképzelés, hogy a cél:

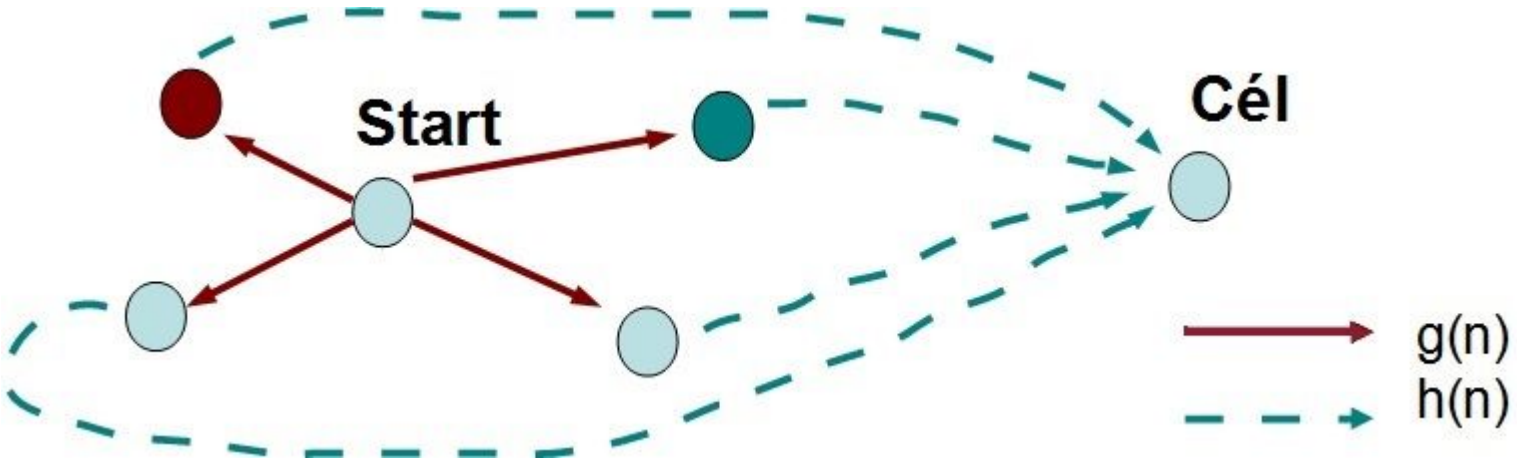
- merrefelé és,
- nagyjából milyen messzire fekszik.

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

- a probléma minden állapotára tudni kell kiszámítani
- kifejezi az előrehaladás becsült költségét
- ha pontos, akkor elvben fölöslegessé teszi a keresést

(ha nagyon pontatlan, akkor semmit sem segít)

Ilyen keresés az un. **heurisztikus**, másképpen **informált keresés**.

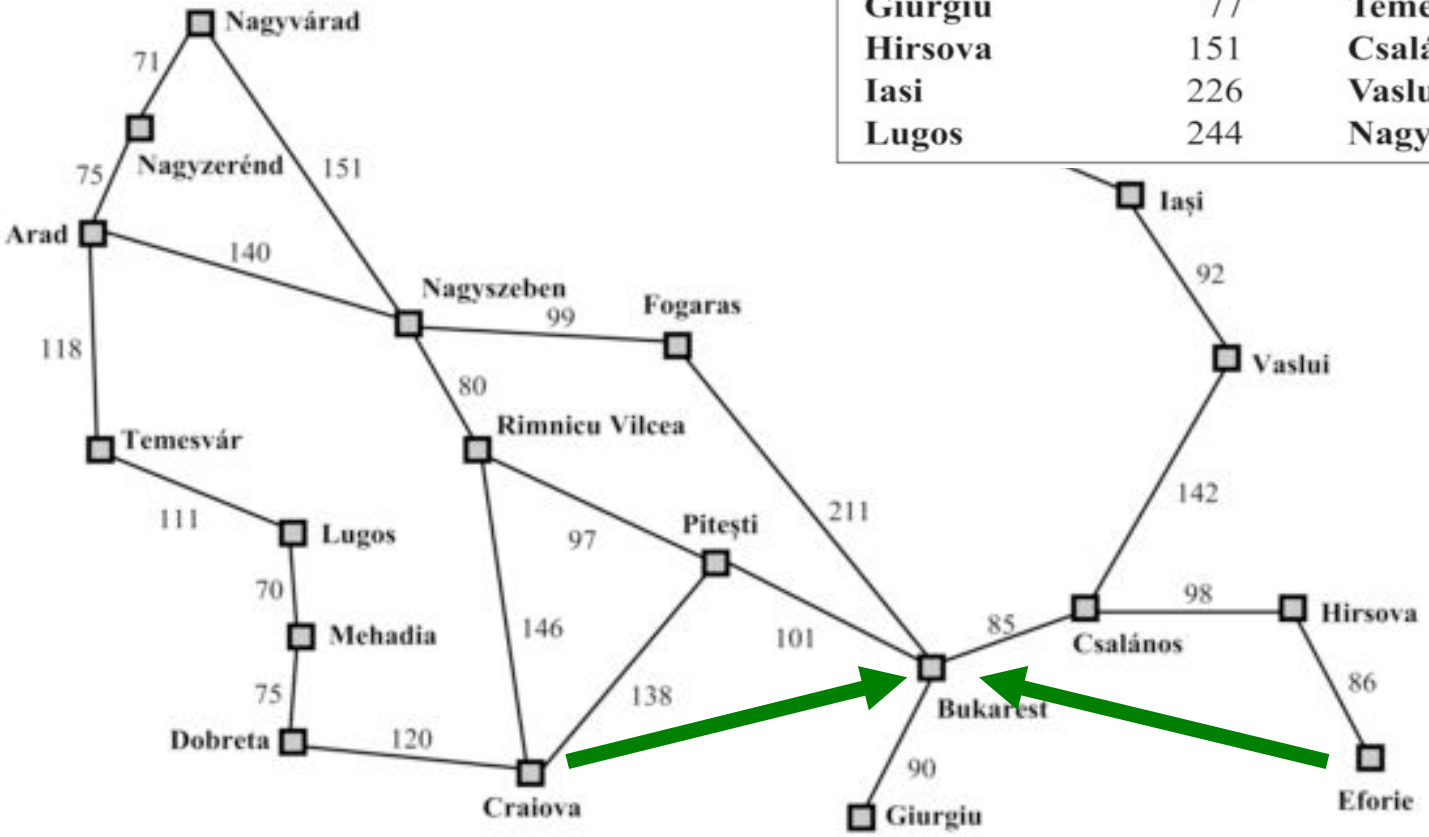


Legyen a heurisztikus függvény a légvonalban mért távolság (h_{LMT})

Az igényeket teljesíti?

Mi mondható a hibájáról?

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



A célhoz legközelebbinek tűnő csomópont először - a legjobbat-először, avagy a mohó keresés (Ro-MohoK.pdf)

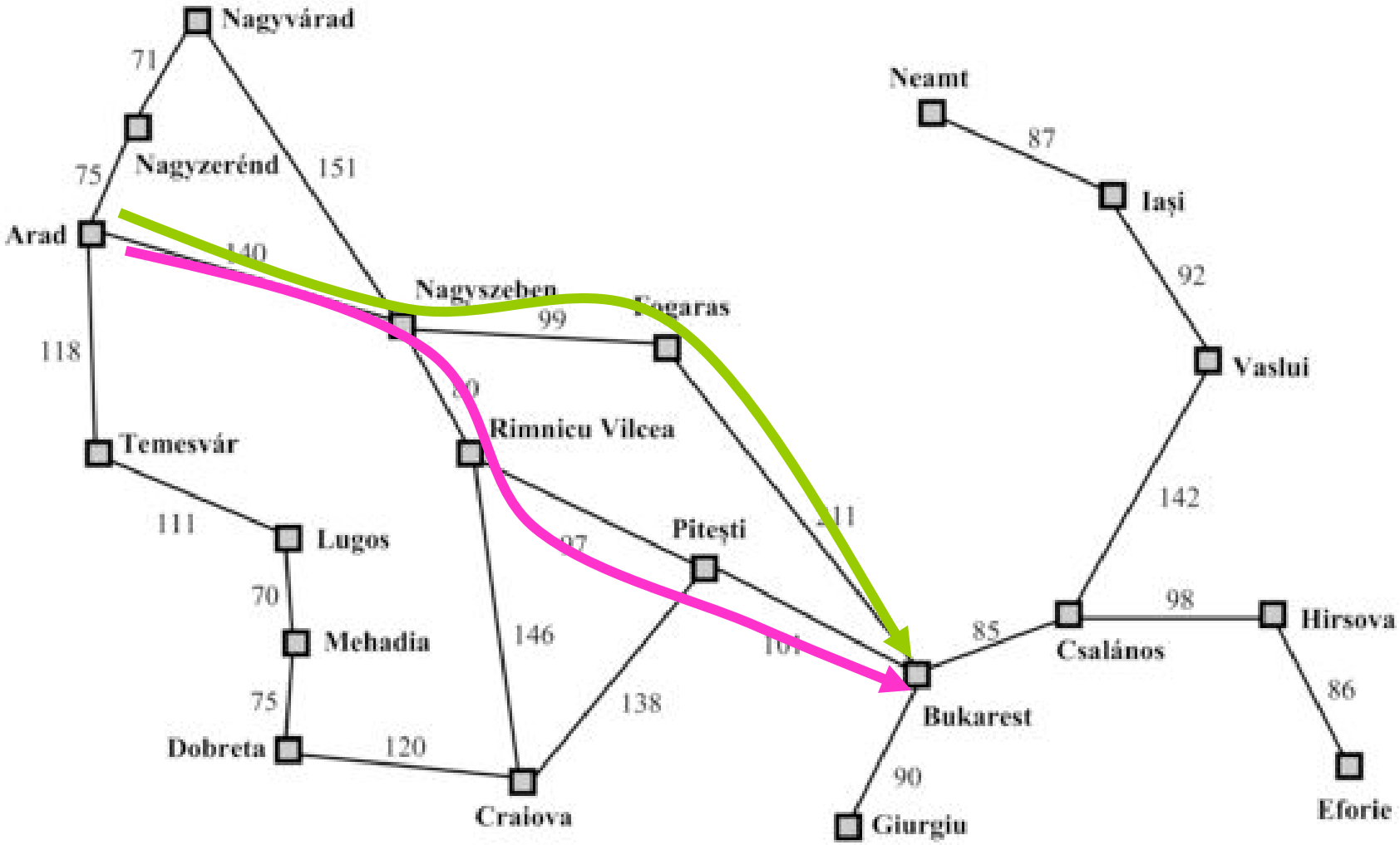
Stratégia: a következő lépésben azt a csp-t fejti ki, amelyhez rendelt probléma-állapotot a legközelebbinek ítéli a célállapothoz.

A ítélethez kell teát egy becslő **heurisztikus függvény h(n)**

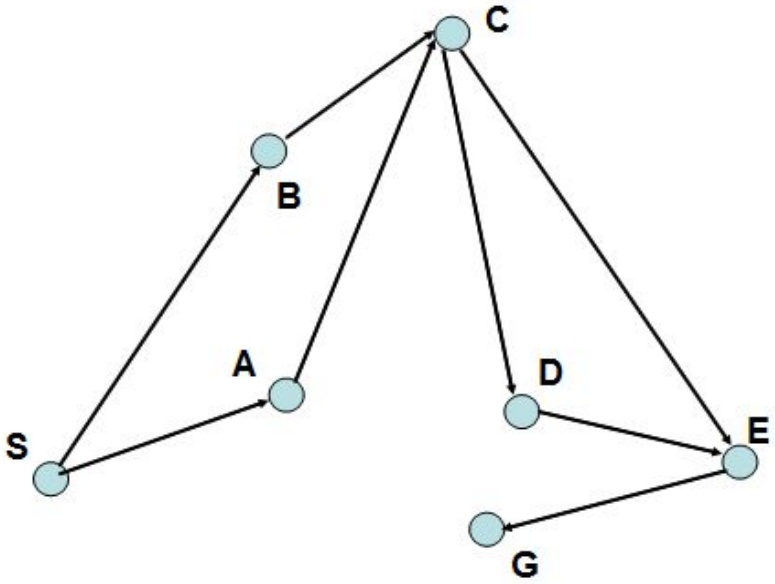
(1) az **n** csp-ból egy cél-állapotba vezető legolcsóbb út becsült költsége

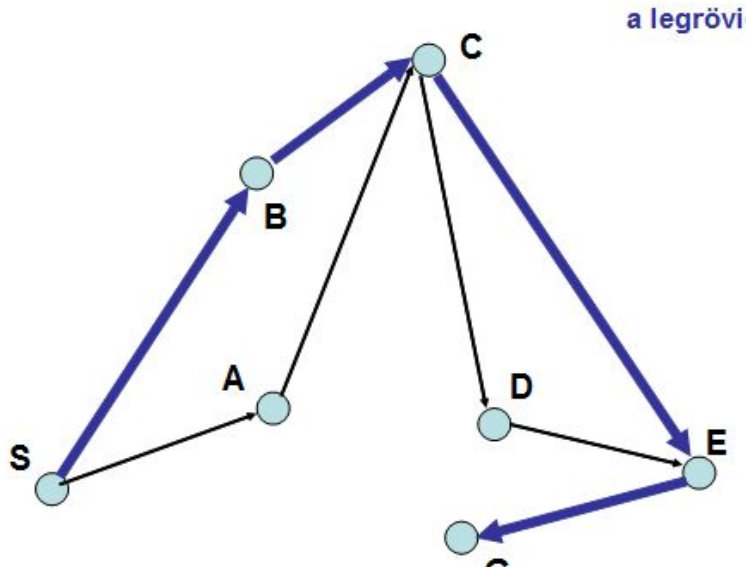
(2) a célállapotban **h**(**n**)= 0, jó célállapottesztnek is

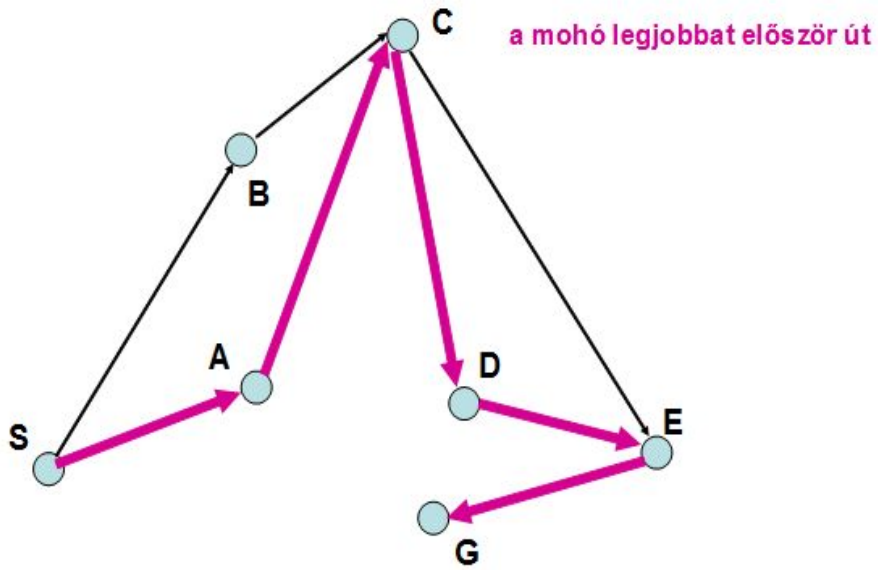
(3) lehet hibája (a tényleges költséghez képest) és elvárjuk, hogy minél kisebb hibának minél ügyesebb keresés feleljen meg.


ezt kaptuk
ezt kellene kapni
Nagyvárad
71
Nagyzeránd
75
151
Arad
140
Nagyszeben
99
Fogaras
118
Rimnicu Vilcea
Temesvár
111
Lugos
Pitești
97
70
Mehadia
146
75
138
Dobreta
120
Craiova
Neamt
87
Iași
92
Vaslui
142
98
Hirsova
85
Csalános
86
Bukarest
90
Giurgiu
Eforie

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







Mohó keresés: mélységi keresésre hasonlít, egyetlen út végigkövetését preferálja a célig: **de zsákutcából visszalép**

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

(worst-case) időigény: $O(b^m)$

Az összes csomópontot a memóriában tartja: tárigény = időigény

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

Tökéletes információ = előretartás elágazások nélkül =
lineáris tár = lineáris idő!

Szóval a mohó keresés ígéretes, de mégsem jó.

A teljes útköltség minimalizálása - A* keresés (Ro-AcsillagK.pdf)

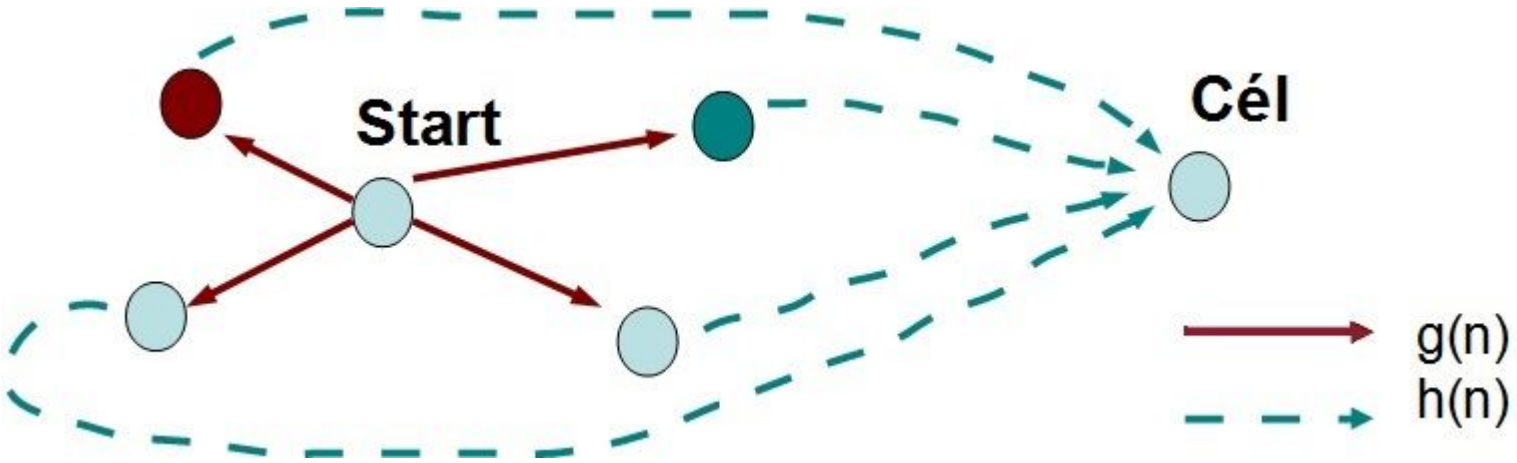
mohó: **min {*h*(*n*)}** a célhoz vezető útköltség becslőjét minimalizálja
nem teljes (nem optimális)

egyenletes költségű: **min {Σ *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*) + Σ *g*(*n*)}**

Σ ***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 (de facto standard az útkeresésben)



Ha a ***h*** függvény soha ne becsüli felül a cél eléréséhez szükséges költséget = **elfogadható heurisztika** (**optimista**, a cél közelebbinek tűnik, mint amilyen)

Ha ***h*** elfogadható, akkor ***f(n)*** soha sem becsüli túl az ***n*** csomóponton át vezető legjobb megoldás valódi költségét

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

(olyan Dijkstra-keresés, ahol a gráfban a csomópontokra cél-felé egy $p(n)$ csökkenő potenciálmezőt értelmeztük, és a figyelembe vett élsúly a redukált $\text{élsúly}_R(n1,n2) = \text{élsúly}(n1,n2) - p(n1) + p(n2)$)

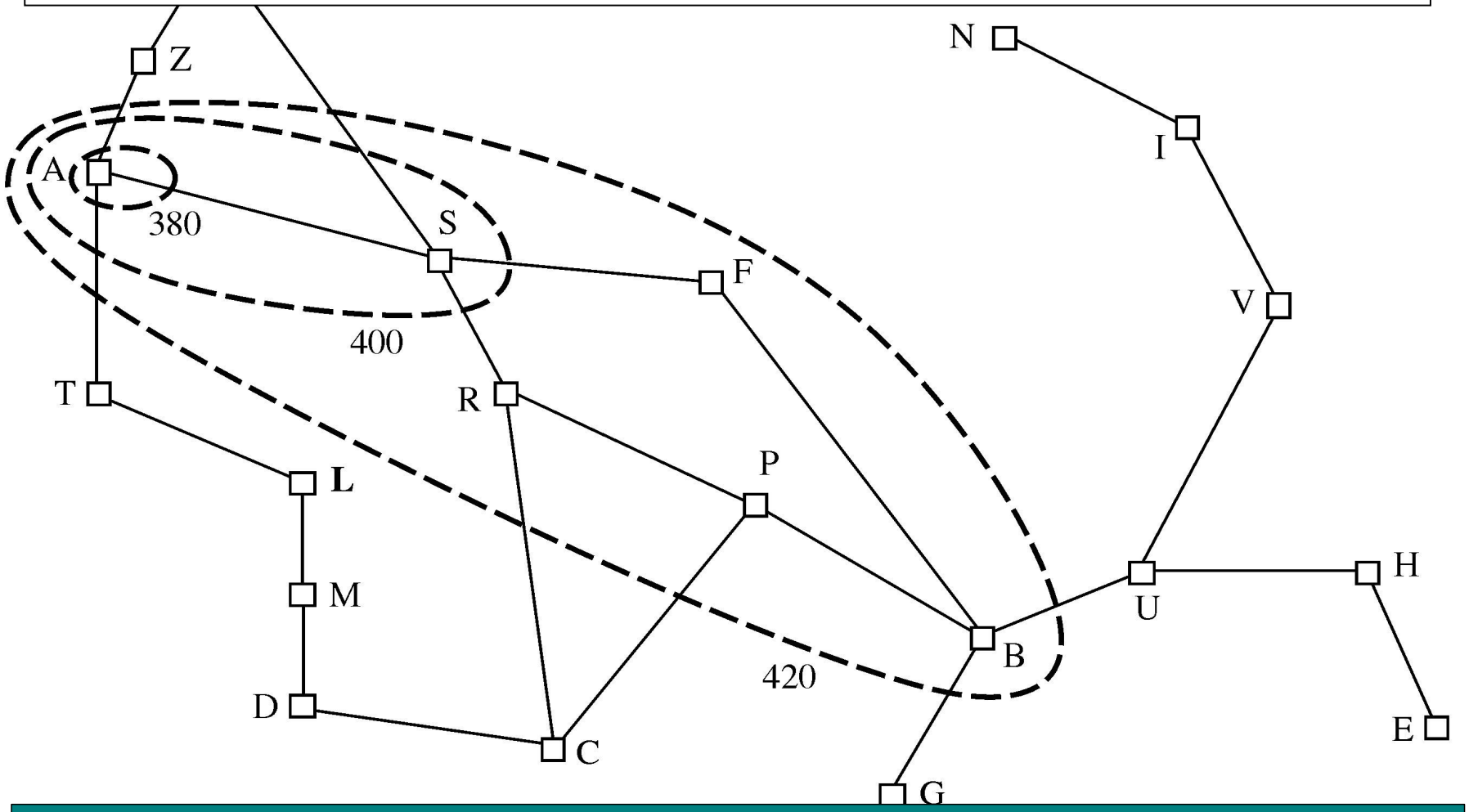
Ha a gyökérből nézve egyetlen út ***f*** értéke nem csökken – a heurisztika **monoton**. Egy heurisztikus függvény aka. monoton, ha teljesíti a háromszög egyenlőtlenséget (**konzisztens heurisztika**, h_{LMT} ilyen).

trükk: **maximális-út** egyenlőség (*n*-ből *n*'-be):

$$f(n') = max(f(n), \Sigma\, g(n') + h(n'))$$

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

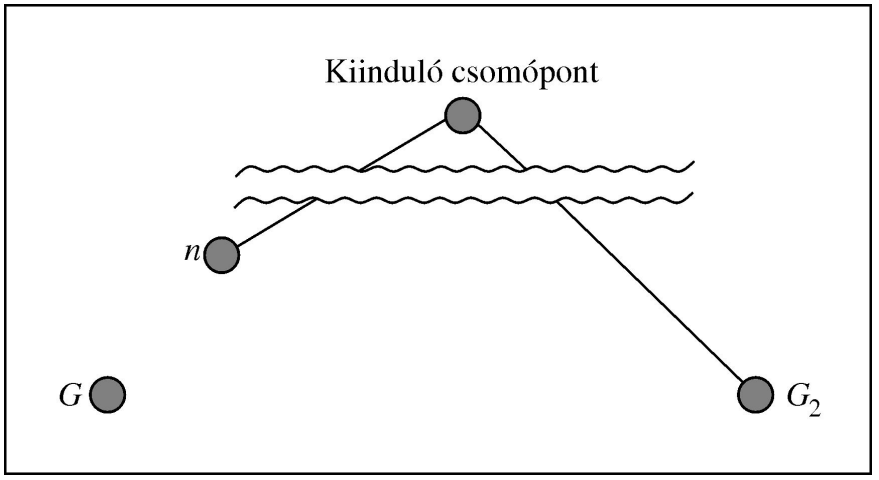
Egyenletes költségű (A* keresés ***h*** = 0 mellett) a csp-sávok a kiinduló csp köré húzott többé-kevésbé koncentrikus „körök”. Koncentrikusak a szélességi keresés esetében.
Pontosabb heurisztikus függvény esetén: a sávok a **cél-állapot felé nyúlnak, fókuszálódnak az optimális út körül.**



http://mialmanach.mit.bme.hu/demonstraciok/a_kereses_usa_terkepen_konturkovetessel

Az A* **teljes** és **optimális**

$$f(G_2) = \Sigma\, g(G_2) > \Sigma\, g(G) > f(n)$$



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.

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 ilyennek formalizálná a problémát)

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.

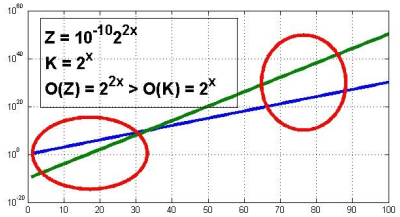
Exponenciális - **kivéve**, ha a heurisztikus függvény hibája legfeljebb az aktuális útköltség logaritmusával nő:

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

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?!

(hogyan?!)



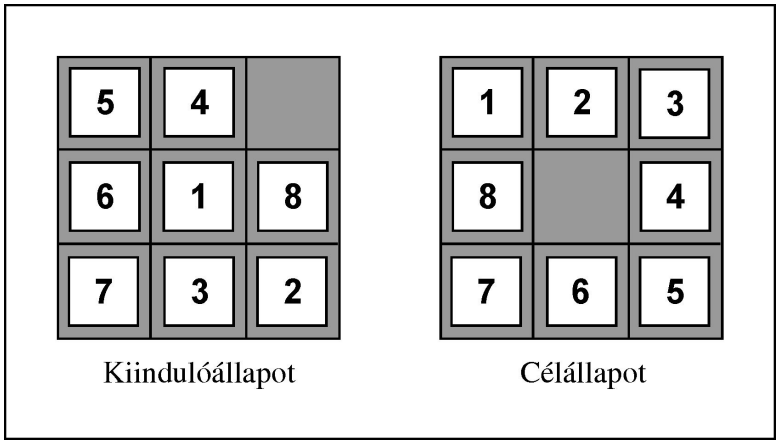
A* algoritmus nagy problémája: az összes legenerált csomópontot eltárolja, lényegesen hamarabb felemészti a memóriát, mintsem kifutna az időből.

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.

.....

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

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

$h3 = ... = ... ?$

Jó lesz pl.:

h3 = (h1 + h2)/2 ?

stb.

Heurisztikus függvény pontossága és hatékonysága

A heurisztikus függvények minősítése: b^* **effektív elágazási tényező**

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

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

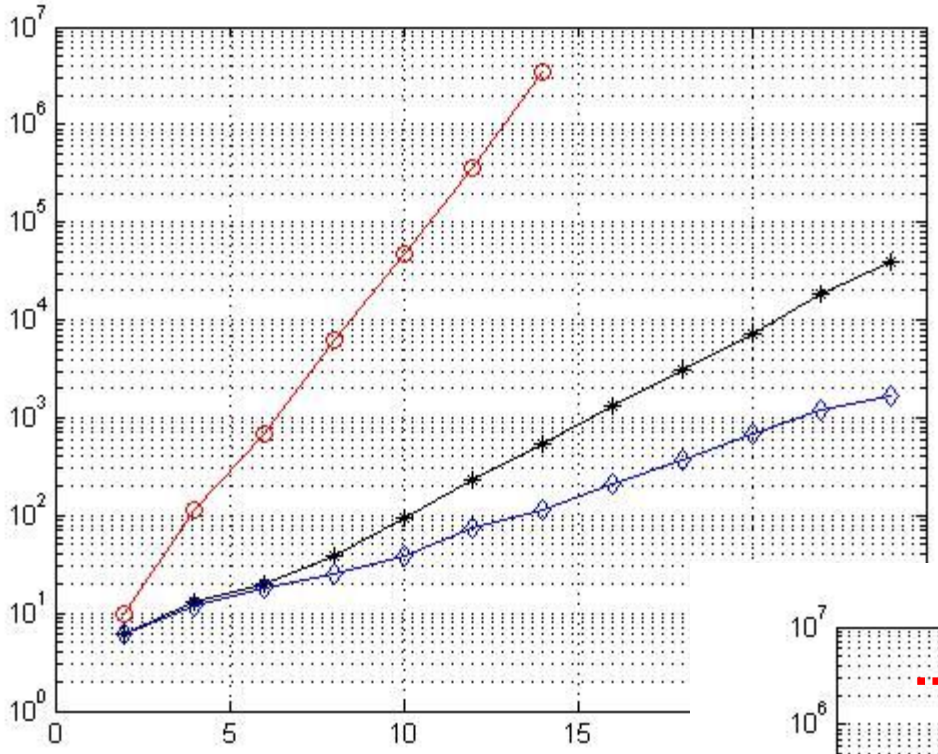
(pl. 5 mélységben fekvő megoldás 52 cs-pont kifejtésével:
az effektív elágazási tényező 1.91)

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

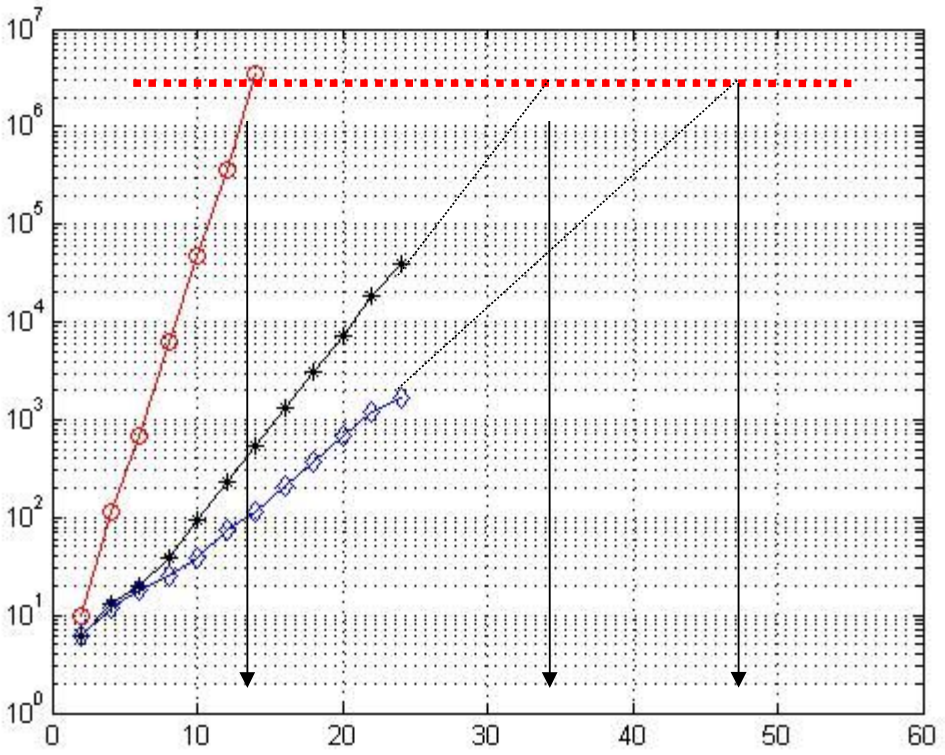
Egy adott heurisztikus függvény által generált fa effektív elágazási tényezője általában nagyjából állandó egy adott problémaosztály számos egyedére.
A b^* kis számú probléma halmazon végzett kísérleti mérése jó becslés.
Egy jól megtervezett heurisztikus függvény effektív elágazási tényezője **1 körüli érték.**

h1 és ***h2*** tesztelése: 100 random probléma példány, 2, 4, …, 24 mélységű megoldással, A*, illetve a neminformált iteratívan mélyülő keresés

	Keres. ktg.			b*		
d	**IMK**	**A*(*h1*)**	**A*(*h2*)**	**IMK**	**A*(*h1*)**	**A*(*h2*)**
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


10^7
10^6
10^5
10^4
10^3
10^2
10^1
10^0
0
5
10
15

idő – tár ?


10^7
10^6
10^5
10^4
10^3
10^2
10^1
10^0
0
10
20
30
40
50
60

Heurisztikus függvény: pontosság és hatékonyság

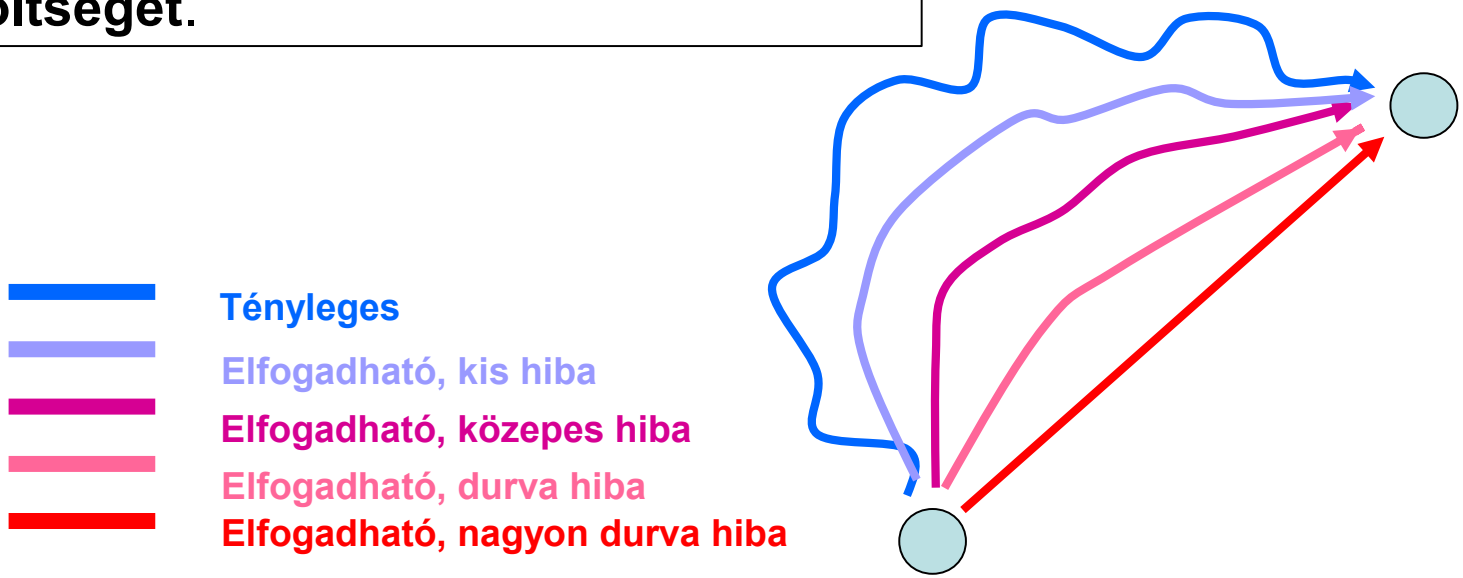
a ***h1, h2*** heurisztikus függvények tesztelése:

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

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

h2 **dominálja** ***h1***–et, **dominació = hatékonyság**
a ***h2***-t használó A* *kevesebb csp-t fog kifejteni, mint a* ***h1***–et használó

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



(Jó) Heurisztikus függvények kitalálása

Egy lapka az A-ról a B-re mozgatható, ha A és B szomszédok és a B mező üres, akkor egy vagy több feltétel törlésével három un. relaxált problémát hozhatunk létre

(a) Egy lapka az A-ról a B-re mozgatható, ha A és B szomszédosak. (h2)

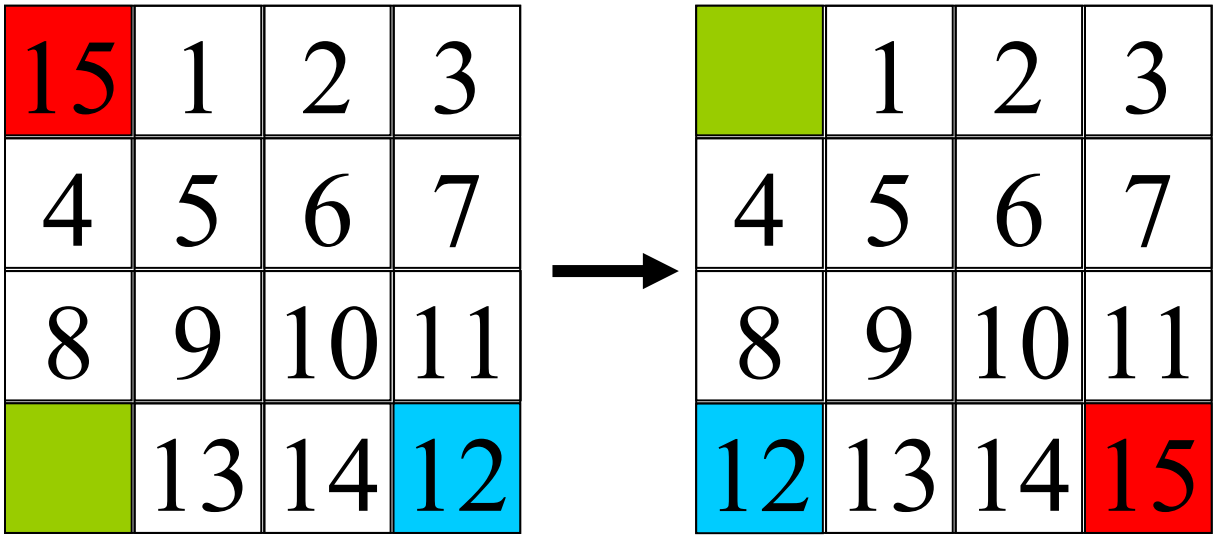
(b) Egy lapka az A-ról a B-re mozgatható, ha a B mező üres.

(c) Egy lapka az A-ról a B-re mozgatható. (h1)

(Absolver: heurisztikus fv-k automatikus generálása probléma definíciójából)

(a) 6 + 3 = 9 lépés

(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 analitikusan is meghatározható.

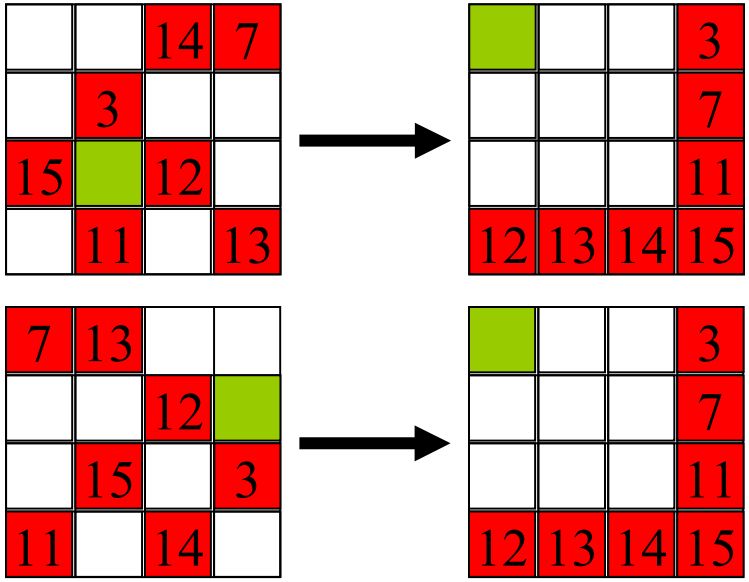
A relaxált probléma pontos megoldása mindig egy elfogadható heurisztika a kevésbé relaxált problémára.

(Jó) Heurisztikus függvények kitalálása

h3: (Disjoint) Pattern Database Heuristics

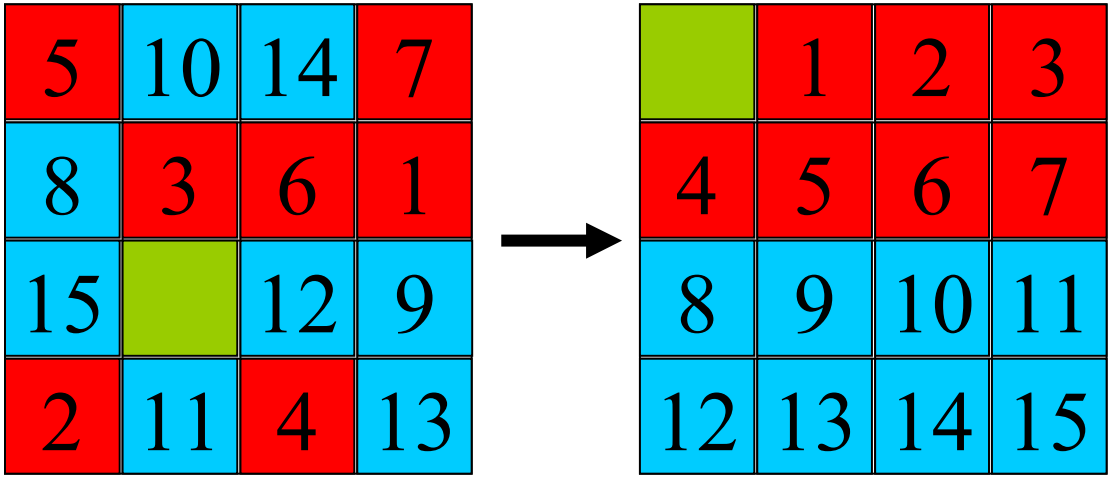
h4: Linear Conflict Heuristics

h5: Gaschnig's Heuristics



Piros lapkák megoldása = 20 lépés
Kék lapkák megoldása = 25 lépés

Telljes heurisztika: h3 = 20+25=45 lépés



h1 = 14
h2 = 39
h3 = 45

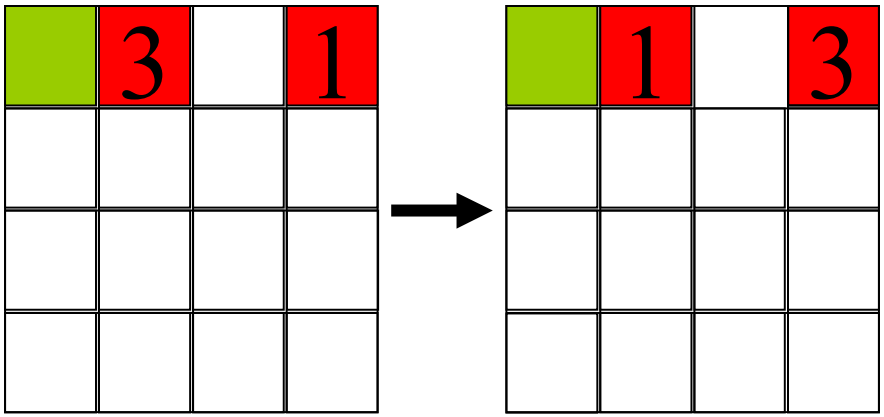
h2 a h3 egy speciális esete, ha minden pattern egy lapka!

(Jó) Heurisztikus függvények kitalálása

h3: (Disjoint) Pattern Database Heuristics

h4: Linear Conflict Heuristics

h5: Gaschnig's Heuristics



h2 = 2 + 2 = 4

de a lapkák egymással szemben nem tolhatók

h4 = h2 + 2 (kikerülés)

h4 = h2 + 2 x (sorbeli lineáris konfliktus, minden sorra
+ oszlopbeli lineáris konfliktus, minden oszlopra)

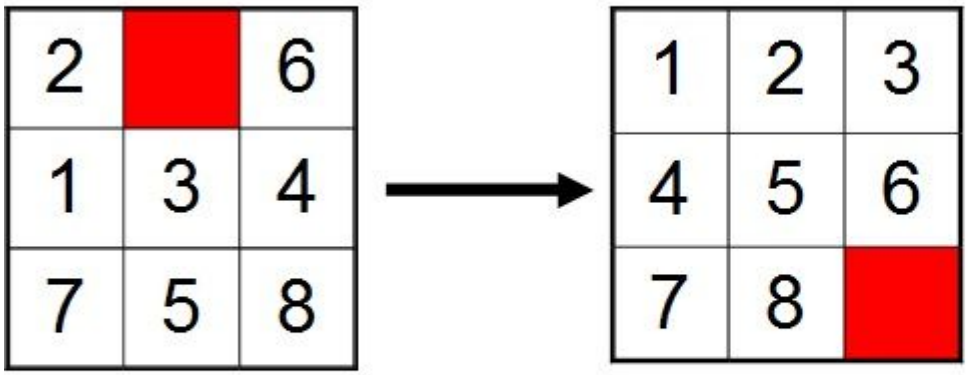
(Jó) Heurisztikus függvények kitalálása

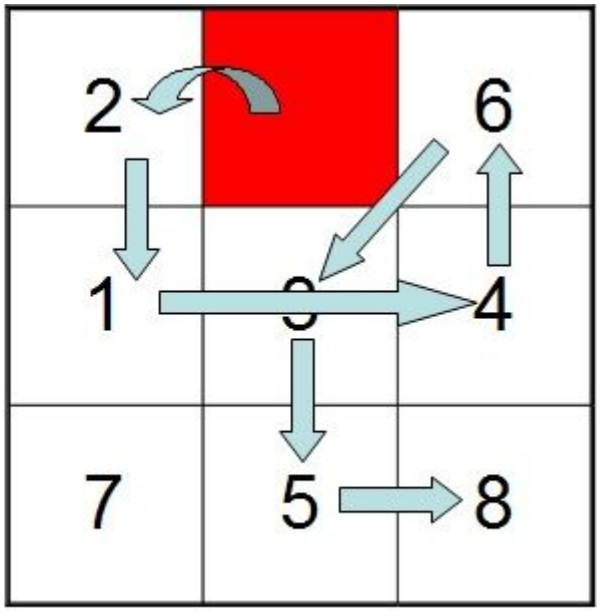
h3: (Disjoint) Pattern Database Heuristics

h4: Linear Conflict Heuristics

h5: Gaschnig's Heuristics

(b) Egy lapka az A-ról a B-re mozgatható,
ha a B mező üres.
elfogadható, jó becslés





(Jó) Heurisztikus függvények kitalálása

Nehéz felismerni a „nyilvánvalóan legjobb” heurisztikus függvényt.

Ha egy problémához adottak a $\boldsymbol{h}_1,\ldots,\boldsymbol{h}_m$ elfogadható heurisztikák
és semelyik sem dominálja a másikat, melyiket kellene választanunk?

Nem kell választanunk. A lehető legjobb:

$$\boldsymbol{h}(\boldsymbol{n}) = \max\{\, \boldsymbol{h}_1(\boldsymbol{n}),\ldots,\boldsymbol{h}_m(\boldsymbol{n}) \,\}$$

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

Probléma: az elemezhetőség, hiszen konkrét h mindig változik?

Heurisztikus függvények megalkotása:

- 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

Memória korlátozott keresés

általában az első korlát, amibe beleütközünk, a rendelkezett memória.

IMA*: az Iteratívan-Mélyülő-Keresés heurisztikus kiterjesztése.

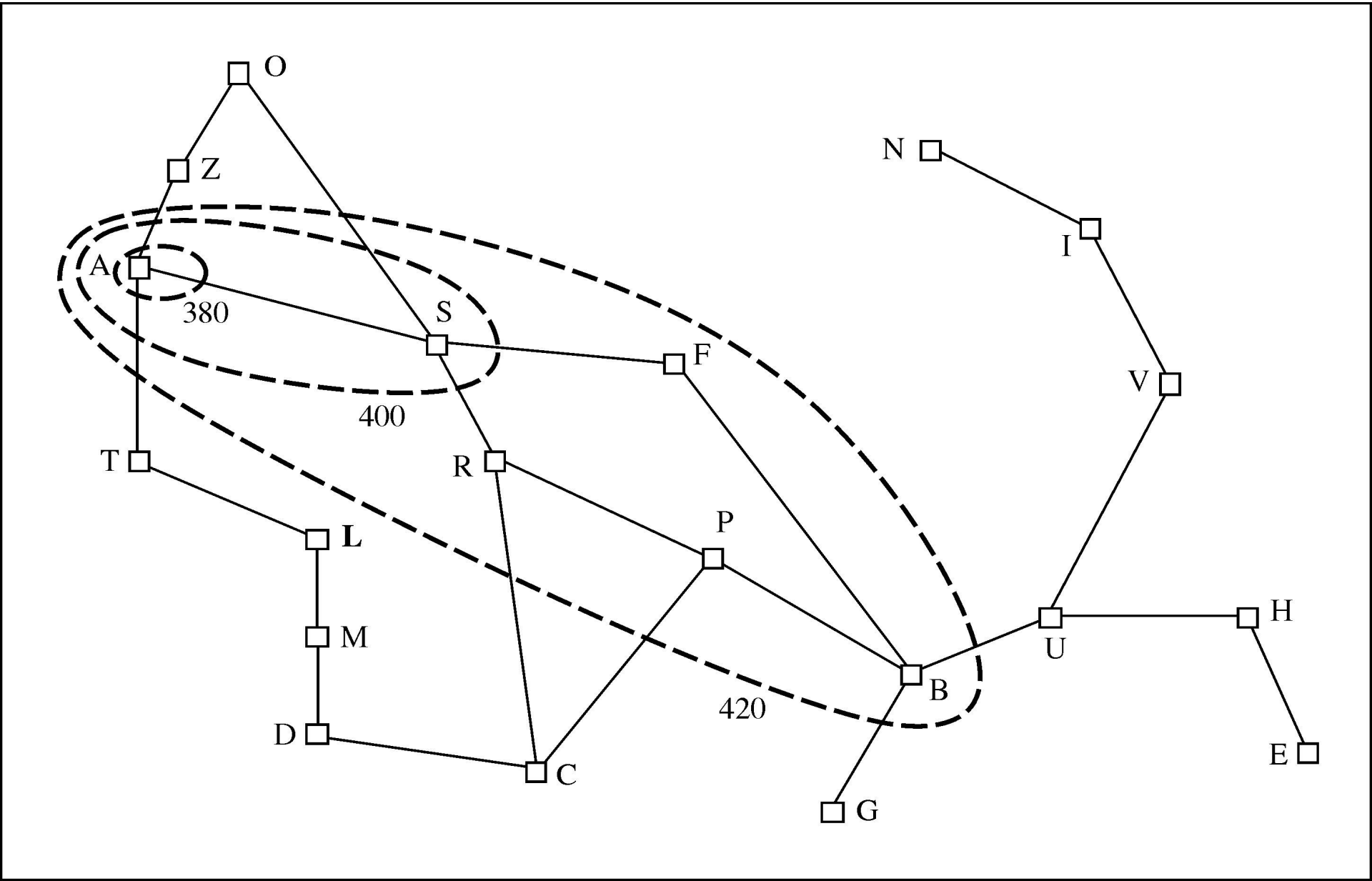
RLEK: Rekurzív Legjobban Először Keresés

EMA*: Egyszerűsített véges memóriájú A*
mint az A*, de korlátozza a sor méretét, hogy az beférjen a memóriába.
stb.

Iteratívan Mélyülő A* keresés (IMA*)

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

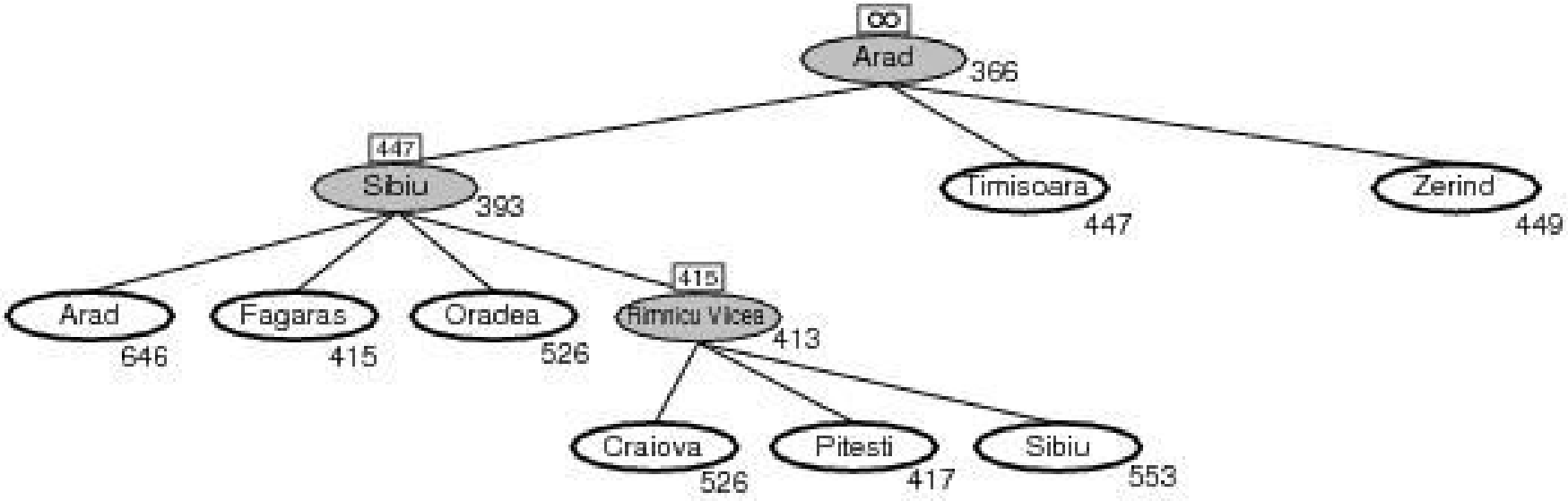
Pl. a 8-as kirakójáték:
Manhattan távolság csak néhány egész értéket vesz fel.
f tipikusan csak 2-3-szor növekszik bármely megoldási út mentén.

Az IMA* algoritmus csak 2-3 iterációt végez, és az idő hatékonysága 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).

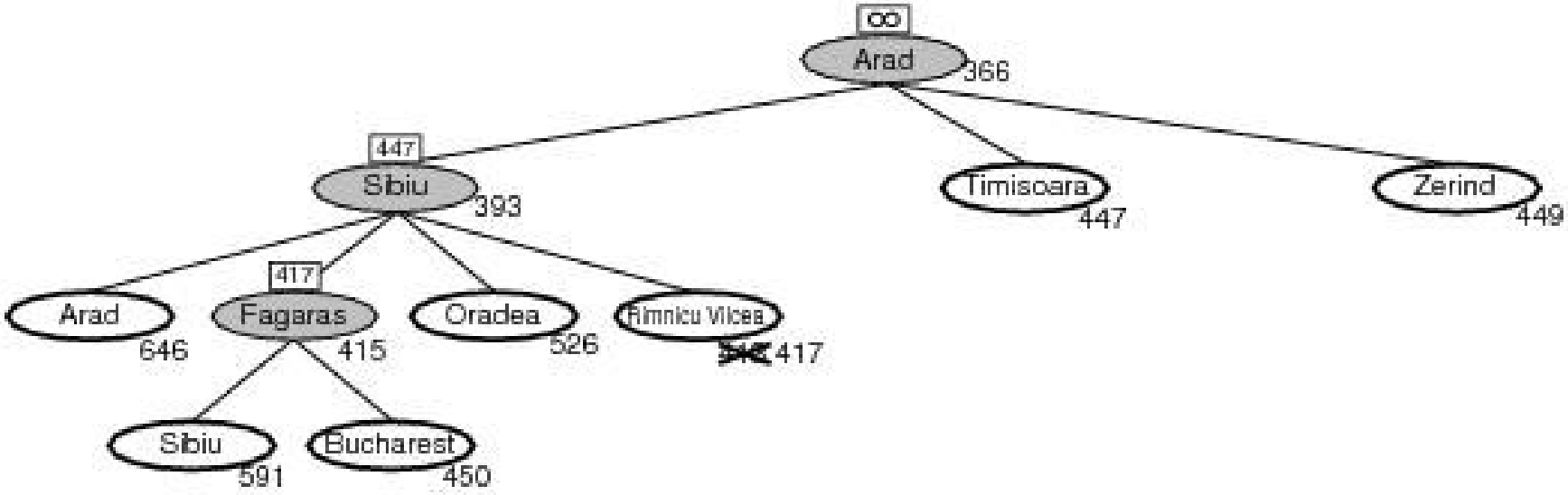
Rekurzív Legjobb Először Keresés (Ro-RLEK.pdf)

Addig halad Legjobbat Először módon *f* érték szerint, amíg jobb alternatívára nem lel hátrahagyott elágazásban.
Átkapcsol, felszabadít memóriát, de a pillanatnyilag követett út költségét memorizálja.



Rekurzív Legjobb Először Keresés (Ro-RLEK.pdf)

Addig halad Legjobbat Először módon *f* érték szerint, amíg jobb alternatívára nem lel hátrahagyott elágazásban.
Átkapcsol, felszabadít memóriát, de a pillanatnyilag követett út költségét memorizálja.



Rekurzív Legjobb Először Keresés (Ro-RLEK.pdf)

Addig halad Legjobbat Először módon *f* érték szerint, amíg jobb alternatívára nem lel hátrahagyott elágazásban.
Átkapcsol, felszabadít memóriát, de a pillanatnyilag követett út költségét memorizálja.