3. Gráfkereső stratégia

- A gráfkereső rendszer olyan KR, amelynek
  - globális munkaterülete a startcsúcsból kiinduló már feltárt utakat (részgráfot) tárolja
    - kiinduló értéke: a startcsúcs,
    - terminálási feltétel: megjelenik egy célcsúcs vagy megakad az algoritmus.
  - keresés egy szabálya: egy csúcs rákövetkezőit állítja elő (kiterjeszti),
  - vezérlés stratégiája: a legkedvezőbb csúcs kiterjesztésére törekszik,

1

3.1. A gráfkereső alapalgoritmus

- ***Jelölés:***
  - *G* - keresőgráf
  - *NYÍLT* - nyílt csúcsok halmaza
  - Γ - kiterjesztés
  - kiterjesztett csúcsok - zárt csúcsok halmaza
- Az absztrakt keresési tér a továbbiakban is egy nem feltétlenül véges δ-gráf.

2

Procedure *GK0*

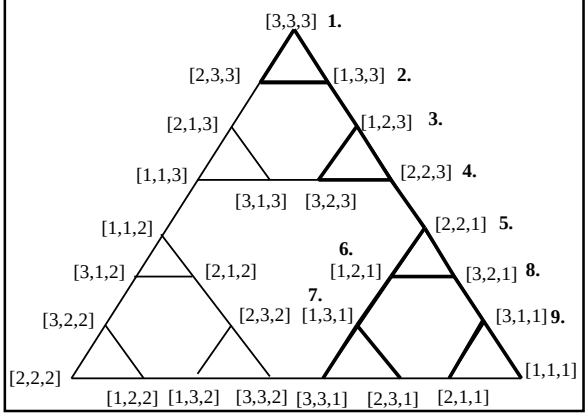
1. $G \leftarrow \{s\}$: $NYÍLT \leftarrow \{s\}$
2. while not *üres*(*NYÍLT*) loop
3. $n \leftarrow$ *elem*(*NYÍLT*)
4. if *cél*(*n*) then return *van megoldás*
5. $G \leftarrow G \cup \Gamma(n)$
6. $NYÍLT \leftarrow NYÍLT - \{n\}$: $NYÍLT \leftarrow NYÍLT \cup \Gamma(n)$
7. endloop
8. return *nincs megoldás*

end

Megjegyzés

- Körökre érzékeny
  - Zárt csúcs ne lehessen újra nyílt?
- Nehezen olvasható ki a megoldás
  - Jelölni kellene az utakat
- Nem feltétlenül talál optimális megoldást

4



3.2. Általános gráfkereső algoritmus

- Módosítások:
  - A következő kiterjesztés eldöntése – kiértékelő függvény
  - A csúcsokhoz (különösen a célcsúcshoz) vezető út nyilvántartása – szülő poiterek
  - A csúcsokhoz vezető minél kisebb költségű út nyilvántartása - út költségek
    - Körök kizárása
    - Optimális út megtalálásának igénye

6

Kiértékelő függvény

- f:*NYÍLT* → **R**
- a 3. lépésben $n \leftarrow min_f(NYÍLT)$
- f egy dinamikus függvény

7

Feszítőfa és költsége

- Pointerek feszítőfája: a G egy s gyökerű feszítőfája
  - π:$G \rightarrow G$ $\quad \pi(n) = n$ csúcs egyik szülője, $\pi(s) = nil$
- Az n csúcshoz vezető, nyilvántartott $\alpha \in \{s \rightarrow n\}$ út költsége
  - g:$G \rightarrow$ **R** $\quad g(n) := c^{\alpha}(s,n)$
- Az n csúcshoz vezető optimális út költsége (a teljes reprezentációs gráfra nézve)
  - g^*:$R \rightarrow$ **R** $\quad g^*(n) := c^*(s,n) \quad \leq g(n)$

8

Konzisztens és optimális költségű feszítőfa

- A G feszítőfája konzisztens, ha a g függvény minden G-beli n csúcshoz a feszítőfában nyilvántartott $s \rightarrow n$ út költségét adja meg.
- A G feszítőfája optimális, ha minden G-beli n csúcshoz G-beli $s \rightarrow n$ optimális utat tárol.
  - Vigyázat! Nem az R-re nézve optimális.
- Mindkét tulajdonság fenntartásához szükséges, hogy egy n csúcs kiterjesztésekor - amennyiben van egy (n,m) él - megvizsgáljuk az m csúcs π illetve g értékét.

9

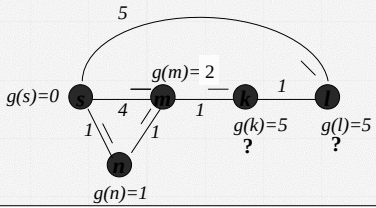
Az m csúcs három esete

- Új csúcs
  - Ha $m \notin G$ akkor
  - $\pi(m) \leftarrow n, \; g(m) \leftarrow g(n)+c(n,m)$
  - $NYÍLT \leftarrow NYÍLT \cup \{m\}$
- Régi csúcs, amelyhez olcsóbb utat találtunk
  - Ha $m \in G$ és $g(n)+c(n,m)<g(m)$ akkor
  - $\pi(m) \leftarrow n, \; g(m) \leftarrow g(n)+c(n,m)$
- Régi csúcs, amelyhez nem találtunk olcsóbb utat
  - Ha $m \in G$ és $g(n)+c(n,m) \geq g(m)$ akkor
  - *SKIP*

10

Optimális költségű konzisztens feszítőfa?

- Ha $m \in G$ és $g(n)+c(n,m)<g(m)$, és m csúcsnak vannak leszármazottai



11

Lehetséges megoldások

1) Az m csúcs összes keresőgráfbeli leszármazottját átvizsgáljuk valamilyen gráfbejárási technikával.
2) Ügyes kiértékelő függvénnyel biztosítjuk, hogy soha nem forduljon elő ilyen eset
3) Nem törődünk a zárt m csúcs leszármazottaival, de magát az m csúcsot visszahelyezzük a *NYÍLT* halmazba.

12

Procedure *GK*

1. $G \leftarrow \{s\}$: *NYÍLT*$\leftarrow\{s\}$: $g(s)\leftarrow 0$: $\pi(s)\leftarrow nil$
2. **while** not *üres*(*NYÍLT*) **loop**
3. $n \leftarrow min_f$(*NYÍLT*)
4. **if** *cél*(n) **then return** megoldás
5. *NYÍLT* $\leftarrow$ *NYÍLT*-$\{n\}$
6. **for** $\forall m \in \Gamma(n)$ **loop**
7. **if** $m \notin G$ or $g(n)+c(n,m)<g(m)$ **then**
8. $\pi(m) \leftarrow n$, $g(m) \leftarrow g(n)+c(n,m)$, *NYÍLT* $\leftarrow$ *NYÍLT*$\cup\{m\}$
9. **endloop**
10. $G \leftarrow G \cup \Gamma(n)$
11. **endloop**
12. **return** nincs megoldás

end

3.1. Lemma

- A *GK* működése során egy csúcsot legfeljebb véges sokszor terjeszt ki.
- Bizonyítás:
- 1. Egy *n* csúcs legfeljebb annyiszor kerülhet be a *NYÍLT*-ba, ahányszor egy minden addiginál olcsóbb utat találunk hozzá.
- 2. Ilyen útból legfeljebb véges sok van:
  - Először egy *C* költségű utat találunk az *n* csúcshoz. Ennél olcsóbb utak a *C*/δ korlátnál biztos rövidebbek. (δ) Megadott korlátnál rövidebb utak száma véges. (σ)

14

3.1. Tétel

- A *GK* véges reprezentációs gráfban mindig terminál.
- Bizonyítás:
- A *GK* véges sok csúcsot (*3.1.lemma*) véges lépésben végleg kiterjeszt, azaz üres *NYÍLT* halmazzal áll meg, hacsak már korábban nem terminál másként.

15

3.2. Invariáns lemma

- Legyen *n* egy tetszőleges *s*-ből elérhető csúcs. A *GK* az *n* csúcs kiterjesztése előtt bármelyik $s^* \rightarrow n$ optimális úton mindig nyilvántart egy olyan *m* csúcs, amelyikre teljesül, hogy
  - (1) $m \in$ *NYÍLT*
  - (2) $g(m)=g^*(m)$
  - (3) minden *m* csúcsot megelőző csúcs végleg zárt, azaz minden ilyen *k* csúcsra $g(k)=g^*(k)$.

16

Bizonyítás

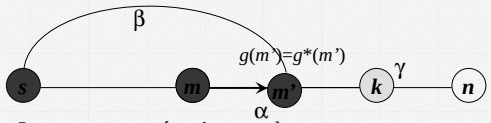
- Teljes indukció a lépések (kiterjesztések) száma szerint
- *1.* lépés előtt: Bármelyik *s*-ből elérhető *n* csúcsra kezdetben fennáll, hogy
  - $s \in$ *NYÍLT* és $s \in s^* \rightarrow n$ és $g^*(s)=0=g(s)$
- *i.* lépés előtt: Tegyük fel, hogy az állítás igaz minden *s*-ből elérhető, az *i*-dik lépésben még ki nem terjesztett *n* csúcsra. Rögzítsünk egy ilyen *n* csúcsot, és egy $\alpha=s^* \rightarrow n$ utat, amelyre tehát
  - $\exists m \in s^* \rightarrow n$ úgy, hogy (1)(2)(3) fennáll.

17

- *i.* lépésben:
  - 1. lehet, hogy nem az *m* csúcsot terjesztjük ki:
    - Ekkor *m*-re továbbra is fennáll (1)(2)(3)
  - 2. ha az *m* csúcsot terjesztjük ki ($m \neq n$), akkor kell keresni egy olyan csúcsot α úton, amelyik átveszi az invariáns állításban szereplő *m* csúcs szerepét. Az *m* csúcsnak az α úton fekvő utóda az *m'*, amelyre az *m* csúcs kiterjesztése után két eset állhat fenn: $m' \in$ *NYÍLT*, vagy $m' \notin$ *NYÍLT*

18

- a) $m' \in$ *NYÍLT*
  - ekkor (1) nyilván teljesül *m'*-re
  - $g(m') = g(m)+c(m,m') = g^*(m)+c(m,m')= g^*(m')$ miatt (2) is
  - (3) pedig azért, mert az *m* előtti csúcsok már eddig is zártak voltak, és most *m* vált végleg zárttá.
- b) $m' \notin$ *NYÍLT*
  - akkor kell lenni egy $ß = s^* \rightarrow m'$ optimális útnak, amely mentén már korábban elértük az *m'* -őt, azaz ß minden *l* csúcsa zárt, és a $g(l)=g^*(l)$.



- Legyen γ az ∝ út $m' \rightarrow n$ szakasza.
- A ßγ egy $s^* \rightarrow n$ út, ezért az indukciós feltevés miatt tartalmaz egy(1)(2)(3) tulajdonságú *k* csúcsot, amely biztosan a γ szakaszon van.
- Ez a *k* csúcs az ∝ útnak is kívánt tulajdonságú eleme.

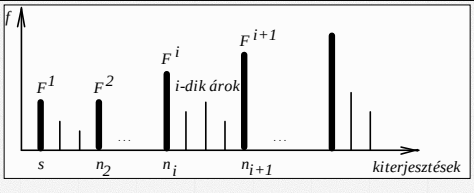
3.2. Tétel

- Ha egy véges reprezentációs gráfban létezik megoldás akkor a *GK* egy célcsúcs megtalálásával terminál.
- Bizonyítás:
- A *3.2. lemma* szerint (legyen az ottani *n* csúcs most a célcsúcs) a *NYÍLT* halmaznak mindig van eleme a célcsúcs elérése előtt, így nem terminálhat az algoritmus üres *NYÍLT* halmazzal.
- *3.1. tétel* miatt viszont az algoritmusnak mindenképpen terminálnia kell.

Csökkenő kiértékelő függvény

- a *NYÍLT* halmazbeli csúcsok kiértékelő függvényértéke az ott tartózkodás alatt nem nő, és
- a *NYÍLT* halmazba visszakerülő csúcs kiértékelő függvényértéke határozottan kisebb, mint a csúcs megelőző kiterjesztésekor felvett függvényértéke.
- A nevezetes gráfkereső algoritmusok ilyen csökkenő kiértékelő függvényt használnak.

Működési grafikon



- Jelöljük meg a startcsúcsot, majd azt a legközelebbi csúcsot (a küszöbcsúcsot), amely kiterjesztésekor mért függvényérték (a küszöbérték) nagyobb vagy egyenlő, mint a megelőző küszöbérték.

3.3. Tétel

- Csökkenő kiértékelő függvény használata mellett a *GK* a küszöbcsúcsok kiterjesztésének pillanatában optimális költségű konzisztens feszítőfát tart nyilván.
- Bizonyítás: HF (A küszöbcsúcsok száma szerinti teljes indukcióval lássuk be, hogy egy küszöbcsúcs kiterjesztésekor nincs zárt csúcs a *NYÍLT* halmazban! Ehhez azt kell megmutatni, hogy amikor egy zárt csúcs újra nyílt lesz, akkor az még a következő küszöbcsúcsot megelőzően biztosan kiterjesztődik.)

3.3. Nevezetes gráfkereső algoritmusok

- Most az *f* kiértékelő függvény megválasztása következik.

Neminformált	Heurisztikus
mélységi,	Előre tekintő,
szélességi,	A, A*, A^c,
egyenletes	B

25

Mélységi gráfkeresés

- *Mélységi gráfkeresésnek* (depth-first) a *GK*-t akkor nevezzük , ha az élköltségeket egységnyinek vesszük (*GK 7. és 8. lépés*) , a kiértékelő függvényt pedig az alábbi módon definiáljuk:
  - $f(n) = -g(n) \quad \forall n \in NYÍLT$ csúcsra.
- Használhatunk mélységi korlátot.
- Mind a mélységi gráfkeresés, mind a visszalépéses keresés mélységi bejárást végez.

26

Szélességi gráfkeresés

- *Szélességi gráfkeresésnek* (breadth-first) a *GK*-t nevezzük, ha az élköltségeket egységnyinek vesszük (*GK 7. és 8. lépés*) , a kiértékelő függvényt pedig az alábbi módon definiáljuk:
  - $f(n) = g(n) \quad \forall n \in NYÍLT$ csúcsra.
- Mindig a legrövidebb megoldást adja.
- Egy csúcsot legfeljebb egyszer terjeszt ki.

27

Egyenletes keresés

- *Egyenletes keresésnek* (uniform-cost) a *GK*-t akkor nevezzük , ha a kiértékelő függvényt az alábbi módon definiáljuk:
  - $f(n) = g(n) \quad \forall n \in NYÍLT$ csúcsra.
- Egy csúcsot legfeljebb egyszer terjeszt ki.
- Mindig az optimális megoldást adja.

28

Heurisztikus függvény

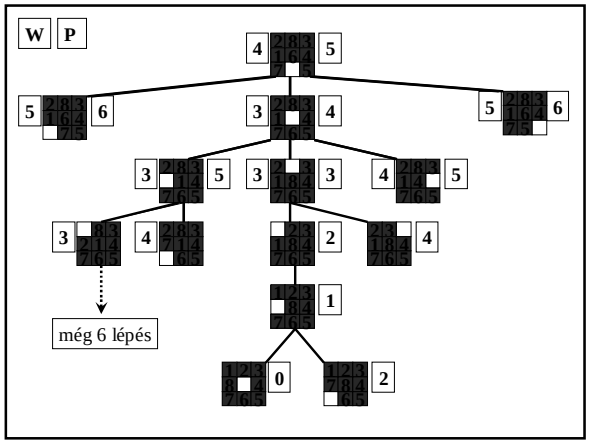
- Azt $h:N \rightarrow \mathbf{R}$ függvényt, amelyik minden *n* csúcsra az abból a célba vezető út költségére ad becslést heurisztikus függvénynek hívjuk.
- $h(n) \approx h^*(n) = \min_{t \in T} c^*(n,t) = c^*(n,T)$
- Egy $s=n_0, n_1, \ldots, n_k = t$ optimális megoldási út esetén $h^*(n_i) = \Sigma_{j=i \ldots k-1} c(n_j, n_{j+1})$

29

Előre tekintő keresés

- Azt a *GK*-t nevezzük előretekintő keresésnek, amelyre
  - $f(n) = h(n) \quad \forall n \in NYÍLT$
- Szeszélyes, eredményessége és hatékonysága erősen függ a heurisztikus függvénytől.

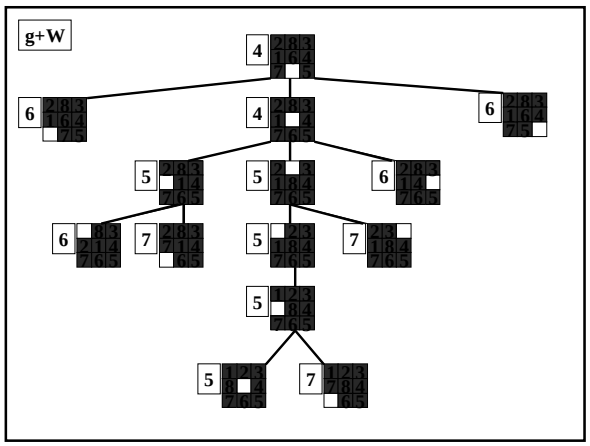
30



A algoritmus

- Azt a *GK*-t nevezzük *A algoritmusnak*, amelyre az
  - $f(n)=g(n)+h(n)$ $\forall n \in NYÍLT$, és
  - $h(n) \geq 0$ $\forall n \in N$
- Példa:
  - $f=g+0$
  - *8*-as játék: $f=g+W$, $f=g+P$

32



Megjegyzés

- Legyen $f^*: N \rightarrow \mathbf{R}$ optimális költségfüggvény olyan, hogy $\forall n \in N$ -re $f^*(n)=g^*(n)+h^*(n)$, azaz $f^*(n)$ a legolcsóbb *n* csúcson átvezető megoldás költsége.
- $f \approx f^*$, ahol $g \geq g^*$ és $h \approx h^*$,
- $f^*(s)$ az optimális megoldás költsége.

34

3.3. Lemma

- Az f^* optimális költségfüggvény egy optimális megoldási út mentén állandó.
- Bizonyítás: Legyen $s=n_0, n_1, ..., n_k = t$ egy optimális megoldás.

$$f^*(n_i) = g^*(n_i)+h^*(n_i) =$$
$$= \Sigma_{j=0...i-1}\, c(n_j,n_{j+1}) + \Sigma_{j=i...k-1}\, c(n_j,n_{j+1}) =$$
$$= \Sigma_{j=0...i-1}\, c(n_j,n_{j+1}) + c(n_i,n_{i+1}) - c(n_i,n_{i+1}) + \Sigma_{j=i...k-1}\, c(n_j,n_{j+1}) =$$
$$= \Sigma_{j=0...i}\, c(n_j,n_{j+1}) + \Sigma_{j=i+1...k-1}\, c(n_j,n_{j+1}) =$$
$$= g^*(n_{i+1}) + h^*(n_{i+1}) = f^*(n_{i+1})$$

35

3.4. Lemma

- Ha az *A algoritmus* nem terminál, akkor minden *NYÍLT* halmazba bekerült *m* csúcs véges sok lépés után kiterjesztődik.
- Bizonyítás:
- Egy csúcs kiértékelő függvényértéke arányos a csúcs mélységével.
  - $f(n) = g(n)+h(n) \geq g^*(n) \geq d(n)*\delta \geq d^*(n)*\delta$
  - ahol $d(n)$ az $s^* \rightarrow n$ optimális út hossza,
  - $d^*(n)$ a legrövidebb $s \rightarrow n$ út hossza.

36

- A $D = \lceil f(m)/\delta \rceil$ korlátnál mélyebben elhelyezkedő csúcsok nem előzik meg az m csúcsot a kiterjesztésben.
  - $f(k) \geq d^*(k)*\delta > D^* \delta > f(m)$
  - Tehát csak a D-nél magasabban fekvő csúcsokra állhat fenn, hogy $f(k) \leq f(m)$
- De egy δ-gráfban a σ-tul. miatt D-nél magasabban fekvő csúcsból csak véges (legfeljebb σ^D) sok van, és ezek véges sok lépésben végleg (lásd *3.1. lemma*) kiterjesztődnek.

3.4. Tétel

- Az *A algoritmus* mindig talál egy megoldást, ha az létezik.
- Bizonyítás:
  - Jelölje α az $s=n_0, n_1, ..., n_k=t$ optimális megoldást.
  - Kezdetben az s nyílt csúcs.
  - Ha n_i egy nyílt csúcs ($i=0\ldots k$), akkor az a *3.4. lemma* miatt véges lépésben kiválasztódik, és $i<k$ esetén az n_{i+1} legkésőbb ekkor (n_i kiterjesztésekor) bekerül a *NYÍLT* halmazba.
  - Tehát az algoritmus véges lépés múlva kiválasztja a t célcsúcsot, hacsak korábban nem terminál.

- De korábban csak egy másik megoldás megtalálásával terminálhat - üres *NYÍLT* halmazzal nem -, hiszen a t kiterjesztése előtt a *NYÍLT* halmaz biztos tartalmazza az α egyik csúcsát (*3.2. lemma*).
- Megjegyzés
  - Az *A algoritmus* nem mindig terminál, csak ha van megoldás,
  - de akkor megtalál egyet.

A^* algoritmus

- Azt az *A algoritmust* nevezzük *A^* algoritmusnak*, amelynek heurisztikája *optimális* (admissible = megengedhető), azaz $h(n) \leq h^*(n) \quad \forall n \in N$
- Megjegyzés:
  - $0 \leq h(n) \leq h^*(n) \quad \forall n \in N$
  - $h(t)=0 \quad \forall t \in T$
- Példa:
  - $f=g+0$
  - *8*-as játék: $f=g+W$, $f=g+P$

3.5. Lemma

- Az *A^* algoritmus* által kiterjesztésre kiválasztott bármely n csúcsra teljesül, hogy $f(n) \leq f^*(s)$.
- Bizonyítás: Tegyük fel, hogy a reprezentációs gráfban létezik megoldás, így $s^* \rightarrow t$ optimális út is. Ellenkező esetben az $f^*(s)=\infty$.
- $\exists m \in s^* \rightarrow t$ úgy, hogy $m \in NYÍLT$ és $g(m)=g^*(m)$ (*3.2. lemma*).
- De az algoritmus az n csúcsot választotta ki az m helyett

$$f(n) \leq f(m) = g(m)+h(m) =$$
$$= g^*(m)+h(m) \leq g^*(m)+h^*(m) = f^*(m) = f^*(s)$$

3.5. Tétel

- Az *A^* algoritmus* mindig optimális megoldás megtalálásával terminál feltéve, hogy létezik megoldás.
- Bizonyítás: Az *A^* algoritmus*, mint speciális *A algoritmus*, biztos talál megoldást. (*3.4. tétel*)
- Tegyük fel indirekt módon, hogy ez a megoldás nem optimális, azaz a termináláskor kiválasztott $t \in T$ célcsúcsra $g(t)>f^*(s)$.
- A *3.5. lemma* szerint (n helyébe t) : $f(t) \leq f^*(s)$
- De t célcsúcs: $f(t) = g(t)+0$

A^c algoritmus

- Azt az *A algoritmust* nevezzük A^c *algoritmusnak*, amelynek heurisztikája *következetes*, azaz
  - *monoton megszorításos*: $h(n)-h(m) \le c(n,m) \quad \forall (n,m) \in A$
  - *célcsúcsokban pontos*: $h(t)=0 \quad \forall t \in T$
- Példa:
  - $f=g+0$
  - *8*-as játék: $f=g+W$, $f=g+P$

43

3.6. Lemma

- Ha a *h* heurisztika monoton megszorításos, akkor bármely $n,m \in N$ csúcsra fennáll a $h(n)-h(m) \le c(n,m)$.
- Bizonyítás.
- Ha nincs $n \to$m út, akkor $c(n,m) = \infty$. Ha van, akkor jelölje azt $\alpha=(n=n_0, n_1, \ldots, n_k=m)$. Ennek éleire teljesül:
  - $h(n)-h(n_1) \le c(n,n_1)$
  - $h(n_1)-h(n_2) \le c(n_1,n_2)$
  - ...
  - $h(n_{k-1})-h(m) \le c(n_{k-1},m)$
- Adjuk össze ezeket az egyenlőtlenségeket:
  - $h(n)-h(m) \le \Sigma\, c(n_{i-1},n_i) = c^{\alpha}(n,m)$.

44

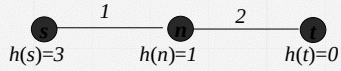
3.6. Tétel

- A következetes heurisztika egyben optimális is.
- Bizonyítás: Kell: $h(n) \le h^*(n) \; \forall n \in N$
- Ha nincs $n \to T$ út, akkor $h^*(n) = \infty$, tehát $h(n) \le h^*(n)$.
- Egyébként $h^*(n) = \min_{t \in T} c^*(n,t) = c^{\alpha}(n,t)$, ahol α a egy $n^* \to T$ út.
- A következetes heurisztika monoton megszorításos, így a *3.6. lemma* miatt $h(n)-h(t) \le c^{\alpha}(n,t)$, továbbá célcsúcsban pontos, azaz $h(t)=0$.
- Összeolvasva $h(n) \le h^*(n)$.

45

Következmény

- Minden A^c *algoritmus* egyben A^* *algoritmus* is, azaz optimális megoldást talál, ha van megoldás.
- Viszont fordítva nem igaz, azaz nem minden A^* *algoritmus* A^c *algoritmus*.



- Általában könnyebb igazolni a következetességet, mint az optimalitást.

46

3.7. Tétel

- Amikor az A^c *algoritmus* egy csúcsot kiterjesztésre kiválaszt, akkor már ismeri hozzá az optimális utat: $g(n)=g^*(n)$.
- Bizonyítás: TF indirekt: *n* kiterjesztésekor $g(n)>g^*(n)$.
- $\exists m \in s^* \to n$ úgy, hogy $m \in NYÍLT$ és $g(m)=g^*(m)$ (*3.2. lemma*) (Az indirekt feltevés miatt az $n \neq m$.)
- A *3.6. lemma* miatt $h(m)-h(n) \le c^*(m,n)$
- De az algoritmus az *n* csúcsot választotta ki *m* ellenében

$$f(n) \le f(m) = g(m)+h(m) = g^*(m)+h(m) \le$$
$$\le g^*(m)+c^*(m,n)+h(n) = g^*(n)+h(n) < g(n)+h(n) = f(n)$$

47

Következmény

- Az A^c *algoritmus* egy csúcsot egynél többször nem terjeszt ki.

48

3.4. A* algoritmus hatékonysága

A továbbiakban olyan problémákat vizsgálunk, amelyeknek van megoldása, így az *A* algoritmus* terminál.

Hatékonyság

- Memória igény: Zárt csúcsok száma a termináláskor
  - •különböző heurisztikák
  - •különböző algoritmusok
- Futási idő: Kiterjesztések száma a zárt csúcsok számához viszonyítva

Különböző heurisztikájú A algoritmusok összehasonlítása*

- Az A_1 (h_1) és A_2 (h_2) *A* algoritmusok* közül az A_2 jobban informált, mint az A_1, ha minden $n \in N \setminus T$ csúcsra teljesül, hogy $h_1(n) < h_2(n)$.
- Megmutatjuk, hogy egy jobban informált *A* algoritmus* nem terjeszt ki több csúcsot, mint a kevésbé informált.

3.8. Tétel

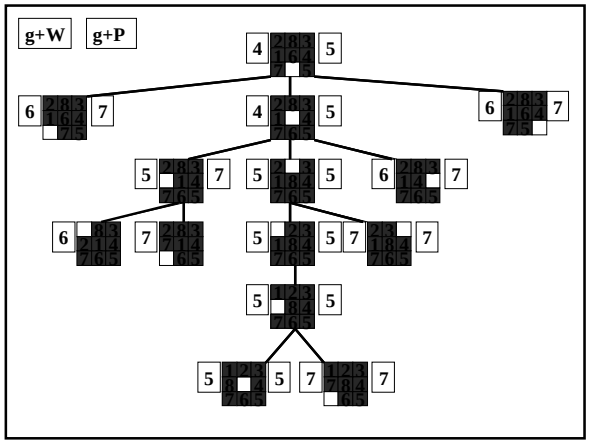
- Legyen A_2 jobban informált *A* algoritmus*, mint az A_1. Ekkor A_2 nem terjeszt ki olyan csúcsot, amelyet A_1 sem terjeszt ki.
- Bizonyítás:
- Teljes indukció az A_2 terminálásakor nyilvántartott feszítőfa mélysége (szintjei) szerint.
- A (*0*)-dik szinten csak a startcsúcs van, amit vagy mindkettő algoritmus kiterjeszt, ha az nem célcsúcs; vagy egyik sem.
- (*d*)-edik szintig minden csúcsról feltesszük, hogy ha azt A_2 kiterjesztette, akkor A_1 is.

- Indirekt tegyük fel, hogy a (*d+1*)-edik szinten van olyan $m \in N$ csúcs, amit csak az A_2 terjeszt ki. (*m* nyilván nem célcsúcs.)
- Egyrészt $f_2(m) \le f^*(s)$ fennáll a *3.5.lemma* miatt. (Az $f_2(m)$ az *m* csúcs A_2 általi kiterjesztéskor mért érték.)
- Másrészt $f_1(m) \ge f^*(s)$. (Az $f_1(m)$ az A_1 terminálásakor mért érték.) (ld. később)
- Harmadrészt $g_2(m) \ge g_1(m)$ (ld. később)
- Összegezve $f_2(m) \le f^*(s) \le f_1(m) = g_1(m) + h_1(m) \le$
  $\le g_2(m) + h_1(m) < g_2(m) + h_2(m) = f_2(m)$

- Az $f_1(m) \ge f^*(s)$, mert az *m d* szinten levő A_2 által kiterjesztett szülőjét (az indukciós feltevés miatt) az A_1 is kiterjeszti, és így az A_1 algoritmus működése alatt az *m* legkésőbb ekkor nyílt csúcs lesz. Ugyanakkor $f_1(m) < f^*(s)$ sohasem lehet, mert ekkor az A_1 kiterjesztené az *m* csúcsot, ami ellentmond az indirekt feltevésnek.
- A $g_2(m) \ge g_1(m)$, mert az A_1 biztosan feltárta az *m* csúcshoz vezető A_2 által talált legolcsóbb (tehát *d* szint alatti) utat, de lehet, hogy talált egy még olcsóbbat.

Megjegyzés

- A gyakorlatban sokszor enyhébb feltételek mellett látványosabb különbségekkel is találkozhatunk:
  - Már $h_1 \le h_2$ esetén is több csúcsot terjeszt ki az A_1, mint A_2
- $W \le P$
- Minél jobban becsli alulról a heurisztika a h^*-ot, várhatóan annál kisebb lesz a memória igénye.



Különböző gráfkereső algoritmusok összehasonlítása

- Optimális heurisztikájú feladatokon hasonlítjuk össze az A algoritmust (amely ilyenkor természetesen egy A* algoritmus) más algoritmusokkal.
- Egy nem-determinisztikus algoritmus determinisztikus változatai („tie-breaking rule") *algoritmusosztályt* alkotnak, ezért valójában algoritmusosztályokat hasonlítunk össze.

56

Jobb algoritmusosztály

- Az *X* és *Y* algoritmusosztályok. Az *X jobb Y* -nál egy adott feladatosztályra nézve**,** ha a feladatosztály minden feladatára van az *X*-nek egy olyan algoritmusa, amely csak olyan csúcsokat terjeszt ki (értékel ki), amelyeket az *Y* minden algoritmusa kiterjeszt (kiértékel).

57

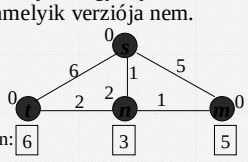
Optimális algoritmusok

- Optimálisnak nevezzük azt az algoritmust/ algoritmusosztályt (nem feltétlenül gráfkeresést), amely optimális heurisztikájú feladatokra optimális megoldást talál feltéve, ha van megoldás.
- Példák:
  - *Egyenletes keresés*
  - *A(*) algoritmus*
  - *A** algoritmus:* $f(n)=\max_{m \in s \to n}(g(m)+h(m))$
  - *IDA* algoritmus*
- Jó lenne, ha az *A algoritmus* lenne a legjobb algoritmus az optimálisak között.

58

*A algoritmus nem jobb az A**-nál*

- Van olyan feladat és optimális heurisztika, ahol az $A^{(*)}$ minden verziója kiterjeszt egy olyan csúcsot, amit az A^{**} valamelyik verziója nem.



Mindkettő az *s* kiterjesztése után:	6	3	5
A alg az *n* kiterjesztése után:	3		2
A^{**} alg az *n* kiterjesztése után:	3		3

59

Megjegyzés

- Melyik optimális algoritmus a legjobb az optimális heurisztikájú feladatosztályon?
  - A fentiek közül egyik sem. (az A algoritmusnál nincs jobb)
- Egy szűkebb feladatosztályon, a következetes heurisztikájú feladatokon viszont az *A algoritmus* a legjobb optimális algoritmus.

60

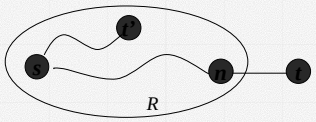
3.9. Tétel

- Következetes heurisztikájú feladatok esetén az *A* (*A*^{c}) *algoritmus* jobb az optimális algoritmusoknál.
- Bizonyítás:
- Tegyük fel indirekt, hogy van olyan *Y* optimális algoritmus, és olyan (*R*,*s*,*T*) feladat a *h* következetes heurisztikával, hogy az *R* egy $n \in N$ csúcsát az *A algoritmus* minden verziója kiterjeszti, de az *Y* algoritmus nem.

61

- Van olyan optimális megoldás, amely nem vezet át *n*-en, hiszen csak ezt tudja az *Y* megtalálni.
- Mivel *A* minden verziója eljut az *n* csúcshoz, ezért van olyan $s^{\alpha} \rightarrow n$ út, amelynek minden *m* csúcsára (többek között az *n*-re is) fennáll, hogy $g^{\alpha}(m)+h(m)<f^*(s)$. (Ekkor ugyanis ennek az útnak minden csúcsa biztos bekerül a *NYÍLT* halmazba, és $f(m) \leq g^{\alpha}(m)+h(m)<f^*(s)$ miatt még a célcsúcs előtt kiterjesztésre kerül.)
- Jelöljük:
  - $C = f^*(s)$
  - $D = g^*(n)+h(n)$
- D < *C* . Ui: $g^*(n)+h(n) \leq g^{\alpha}(n)+h(n) < f^*(s)$

62



- Bővítsük az indirekt feltevésben szereplő feladatot egy új éllel:
  - $(R \cup \{(n,t)\}, s, T \cup \{t\})$
  - $c(n,t)=h(n)+(C - D)/2$
  - $h(t)=0$

63

- A *h* heurisztika az új feladaton is következetes (elég csak az új élt vizsgálni)
  - $h(n)-h(t) = h(n) \leq h(n) +(C - D)/2 = c(n,t)$
- Az új feladat egyetlen optimális megoldása az $s \rightarrow t$ út.
  - Az $s \rightarrow t$ út költsége kisebb az előző feladat optimális megoldásának költségénél (*C*)
  - $g^*(t) = g^*(n) + c(n,t) = g^*(n) + h(n) + (C - D)/2 = D + (C - D)/2 = (C + D)/2 < C$
- Az *Y* algoritmus az *n* csúcsot nem terjeszti ki, ezért az $s \rightarrow t$ megoldást nem találhatja meg: Ez ellentmondás, hiszen egy optimális algoritmusnak ezen az új feladaton is optimális megoldást kell találnia.

64

Az A algoritmusnál nincs jobb

- Ha lenne az *A algoritmusnál* jobb optimális algoritmus, akkor ez a következetes heurisztikájú feladatokra is jobb lenne (hiszen a következetes heurisztika is optimális). Ennek ellent mond a *3.9. tétel*.)
- Az *A algoritmus* nem a legjobb a optimális algoritmusok között, de nincsen nála sem jobb.

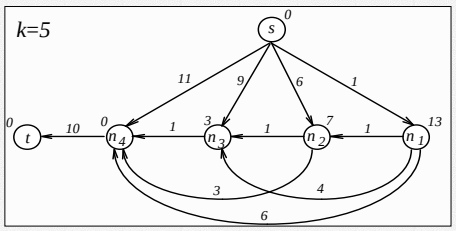
65

Futási idő

- Zárt csúcsok száma: *k*
- Alsókorlát: *k*
  - A következetes heurisztika mellett egy csúcs legfeljebb csak egyszer terjesztődik ki,
  - habár ettől még a kiterjesztett csúcsok száma igen sok is lehet (egyenletes keresés)
- Felsőkorlát: 2^{k-1}

66

Martelli példája

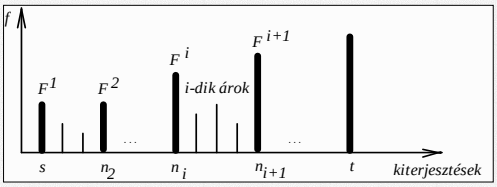


Az $n_1,...,n_{k-1}$ csúcsokhoz rendre $2^0,2^1,...,2^{k-2}$ különböző út vezet.

Megjegyzés

- Másik heurisztikával természetesen javítható a kiterjesztések viszonylagos száma, de nem biztos, hogy ez tényleges javulás lesz.
- A kiterjesztések száma egy viszonylagos szám
  - h_1 heurisztika mellett k_1 darab zárt csúcs, és 2^{k1-1} kiterjesztés
  - h_2 heurisztika mellett k_2 darab zárt csúcs, és k_2 kiterjesztés
  - mégis $2^{k1-1} < k_2$, mivel $k_1 << k_2$.

A probléma oka



- Általában egy árkon belül egy csúcs sokszor kiterjesztésre kerülhet.
- Az árkokban használjunk más kiértékelő függvényt!

AB algoritmus

- Az *AB algoritmust* az *A algoritmusból* kapjuk úgy, hogy bevezetjük az *F* aktuális küszöbértéket és a *q*:*NYÍLT* $\rightarrow R$ belső kiértékelő függvényt , majd
  - az *1. lépést* kiegészítjük az $F \leftarrow f(s)$ értékadással,
  - a *3. lépést* pedig helyettesítjük az
  - if $min_f(NYÍLT)<F$
  - then $n \leftarrow min_q(m \in NYÍLT \mid f(m)<F)$
  - else $n \leftarrow min_f(NYÍLT); F \leftarrow f(n)$
  - endif elágazással.

Megjegyzés

- Nevezetes *AB algoritmusok* :
  - *A algoritmus (egyenletes keresés)* : *q=f*
  - *B algoritmus (Martelli)* : *q=g*
- Az *AB algoritmusok* csökkenő kiértékelő függvényt használnak.

3.10. Tétel

- Az eltérő belső kiértékelő függvényt használó *AB algoritmusok* működésük során ugyanazokat a küszöbcsúcsokat, ugyanabban a sorrendben és ugyanazon küszöbértékkel választják ki, és ekkor ugyanazt a keresőgráfot, ugyanazon feszítőfával és költségértékekkel tartják nyilván.
- Bizonyítás: Teljes indukció a küszöbcsúcsok számára.
- Az *i+1*-dik küszöbcsúcsnál bekövetkező állapot attól függ, hogy előtte mely csúcsokat terjeszti ki a keresés az az *i*-edik árokban. (Ez a kiterjesztések sorrendjétől és számától nem függ.)

Az i-edik árokban kiterjesztett csúcsok

- Egy az *i*-edik árokban kiterjesztett *m* csúcsnak a *NYÍLT* halmazban kell lennie az n_i kiterjesztése után, de még az n_{i+1} kiterjesztése előtt úgy, hogy $f(m) < F_i$ fennálljon
- Az *m* csúcs akkor kerülhet be a *NYÍLT* halmazba, ha *i*-edik árokhoz tartozó csúcsok kiterjesztései során találunk hozzá egy *s*-ből induló n_i -n keresztül vezető utat, amely vagy az első hozzá talált út, vagy egy minden eddigénél olcsóbb út.
- Ha *m* csúcs már benn volt a *NYÍLT* halmazban az n_i kiterjesztésekor, akkor az $f(m) \geq F_i$ állt fenn. Ahhoz, hogy ez megváltozzon ($f=g+h$) kell, hogy találjunk egy minden eddigénél olcsóbb utat *m*-hez *i*-edik árokhoz tartozó csúcsok kiterjesztései során.

Az i-edik árok csúcsai függetlenek q-tól

- $D_i = \{ m \in N \mid$
  - $\exists \alpha \in \{s \rightarrow n_i \rightarrow m\} : \forall n \in n_i \rightarrow m \setminus \{m\} : n \in D_i$
  - $g_i(n_i)+c^{\alpha}(n_i,m)<g_i(m)$ ha $m \in G_i$
    - g_i az algoritmus által nyilvántartott g értékeket mutatja n_i kiterjesztésének pillanatában.
  - $f(m)= g(m)+h(m) \leq g_i(n_i)+c^{\alpha}(n_i,m)+h(m) < F_i$ }

Megjegyzés

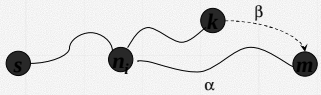
- Az *AB algoritmusok* megoldással terminálnak, ha van megoldás. (Ui: az *A* is egy *AB*)
  - HF: az árkok (az utolsó árok is) véges hosszú.
- Az optimális heurisztikát használó *AB algoritmusok* optimális megoldással terminálnak, ha van megoldás. (Ui: az *A** is egy *AB*)
  - HF: *A** működésekor a célcsúcs az utolsó küszöbcsúcs.
- A következetes heurisztikát használó *AB algoritmus* egy csúcsot legfeljebb egyszer terjesztenek ki. (Ui: az A^c is egy *AB*)
  - HF: A^c árkai üresek.

Megjegyzés

- Az *AB algoritmusok* memória igénye az *A algoritmussal* azonos. (Optimális heurisztikájú feladatokon nincs náluk jobb.)
- Az *AB algoritmusok* futási ideje eltérő, mert az árkon belüli kiterjesztések száma és sorrendje más.
- A futási idő szempontjából az *A algoritmus* nem jó (*A* algoritmus* futási ideje legrosszabb esetben exponenciális).
- A legjobb futási idejű *AB algoritmus* a *B algoritmus* (futási ideje legrosszabb esetben polinomiális)

3.11. Tétel

- A *B algoritmus* egy árkon belül egy csúcsot csak egyszer terjeszt ki.
- Bizonyítás:
- Tegyük fel indirekt, hogy egy *m* csúcs kétszer terjesztődik ki a D_i árkon belül: először az n_i küszöbcsúcsból egy drágább α út mentén érjük el az *m* csúcsot, majd egy olcsóbb β út mentén, azaz
  $c^{\beta}(n_i\, m) < c^{\alpha}(n_i\, m)$



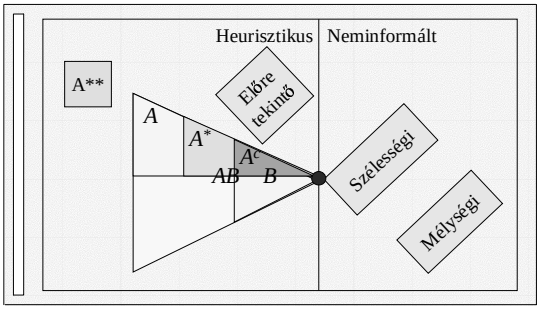
- Amikor az α út mentén elérjük, majd kiterjesztjük az *m* csúcsot ($m \in NYÍLT$ és $g(m) = c(s,n_i)+c^{\alpha}(n_i\, m)$), addigra elértünk a β úton egy *k* csúcshoz ($k \in NYÍLT$ és $g(k) \leq c(s,n_i)+c^{\beta}(n_i\, k)$).
- A *B algoritmus* az *m* csúcsot választotta: $g(m) \leq g(k)$
- $g(k) \leq c(s,n_i) + c^{\beta}(n_i\, k) \leq c(s,n_i)+c^{\beta}(n_i\, m) <$
  $< c(s,n_i)+c^{\alpha}(n_i\, m) = g(m)$

B algoritmus futási ideje

- *k* zárt csúcs esetén legfeljebb *k+1* küszöbcsúcs van (az utolsó a célcsúcs, ha a heurisztika optimális), ilyenkor *k* darab árok van.
- A *B* algoritmus legrosszabb esetben egy árokban az összes *k* csúcsot pontosan egyszer kiterjeszti.
- Ezért a kiterjesztések összes száma legfeljebb k^2.

79

Algoritmus osztályok



80

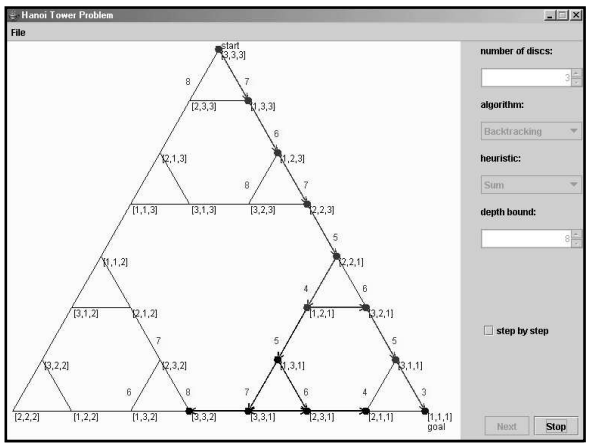
3.5. Heurisztika szerepe

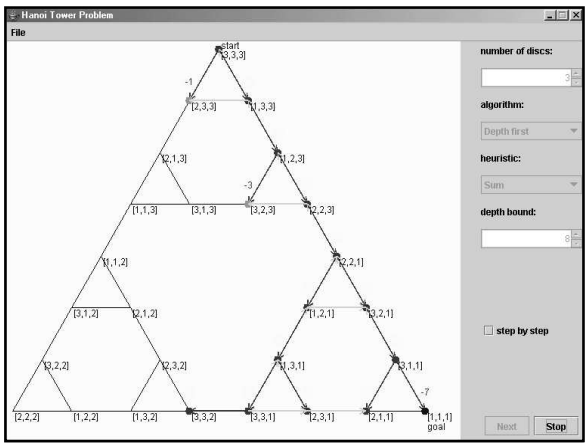
- Milyen a jó heurisztika?
  - optimális : $h(n) \leq h^*(n)$
    - Nincs mindig szükség az optimális megoldásra.
  - jólinformált: $h(n) \sim h^*(n)$
  - monoton megszorításos: $h(n)\text{-}h(m) \leq c(n,m)$
    - Ekkor *A^c algoritmus*, különben *B algoritmus*
- Változó heurisztikák:
  - $f=g+\phi*h$ ahol $\phi \sim d$
  - *B'* algoritmus

81

Gyakorlat

82





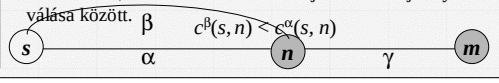
Állítások

- Csökkenő kiértékelő függvény mellett a *GK* küszöbcsúcsai mind különböznek, és egy csúcs csak az első kiterjesztésekor lehet küszöbcsúcs.
- Ha a *GK* terminálásakor egyetlen nyílt csúcs (a célcsúcs) van, akkor a *GK* optimális megoldást talált.
- Az A^* minden olyan n nyílt csúcsot kiterjeszt, amelyre $f(n)<f^*(s)$.
- Ha a h monoton megszorításos, akkor egy tetszőleges n csúcsba vezető optimális út mentén a g^*+h értéke monoton növekvő.

85

Tétel

- Csökkenő kiértékelő függvény használata mellett a *GK* csak konzisztens csúcsot választ kiterjesztésre.
- Bizonyítás: HF (Képzeljük el, hogy miután az n csúcs egy α út mentén kiterjesztődött, egy olcsóbb β út mentén újra nyílt lesz. Eközben bekerült a *NYÍLT* halmazba a γ út menti m is. Az m csúcs az nyílttá válásakor inkonzisztensé válik. Meg kell mutatni, hogy az algoritmus nem terjeszti ki az m csúcsot az n csúcs előtt! Ehhez kövessük nyomon a β és a γ utak felfedezését, az n csúcs korábbi kiterjesztése és újra nyílttá válása között.



86

*A** algoritmus*

- Azt a *GK*-t nevezzük *A** algoritmusnak*, amelyre az
  - $f(n)=\max_{m\in s\to n}(g(m)+h(m))$ $\quad\forall n\in NYÍLT$,
  - h optimális
- Optimális megoldást talál, ha van megoldás
  - Talál megoldást. Ehhez 3.5. lemma:
    - $f(n) \geq g(n)+h(n) \geq g^*(n) \geq d^*(n)*\delta$
  - Optimális a megoldás. Ehhez 3.6. lemma:
    - $f(n) \leq f(m) = g(m_i)+h(m_i) \leq f^*(m_i) = f^*(s)$.

87

IDA algoritmus*

$c \leftarrow f(start)$
loop
  $(megoldás, c) \leftarrow VL2.1(<s>, c)$
  if $megoldás \neq hiba$ then return $megoldás$
  if $c=\infty$ then return $hiba$ // *ha VL2.1 –ben nem volt vágás*
endloop

- *VL2.1*:
  - $f=g+h$
  - n csúcsot kiértékelés nélkül levágja, ha $f(n)>c$
  - c-ben visszaadja a levágott csúcsok f értékeinek minimumát; ha nincs ilyen, akkor a ∞-t.

- Az *IDA* algoritmus* optimális heurisztika mellett optimális megoldást talál, ha van megoldás!
  - Talál megoldást: Egyrészt a c értéke nem válhat ∞ naggyá megoldás megtalálása előtt. Másrészt a keresés nem akadhat meg egy megoldási útnak egy csúcsánál.
  - Optimális a megoldás : Ehhez elég belátni, hogy $c \leq f^*(start)$ mindig fennáll, hiszen ekkor nem találhat a *VL2.1* olyan célcsúcsot, amelyre $g(t)> f^*(start)$.
    - Legyen $n\in s^*\to t$ az a csúcs, amit levágunk ($g(n) = g^*(n)$). Ekkor $c \leq f(n) = g(n)+h(n) = g^*(n)+h(n) \leq f^*(n) = f^*(s)$.

89

B' algoritmus

if $h(n)<min_{m\in\Gamma(n)}\,(c(n,m)+h(m))$
then $h(n)\leftarrow min_{m\in\Gamma(n)}\,(c(n,m)+h(m))$
else for $\forall m\in\Gamma(n)$-re **loop**
  if $h(n)- h(m)>c(n,m)$ **then** $h(m) \leftarrow h(n)- c(n,m)$
endloop

A h optimális marad
A h nem csökken
A mononton megszorításos élek száma nő

Mohó A algoritmus

- Nincs mindig szükség az optimális megoldásra.
  - Ilyenkor a mohó *A* algoritmus is használható.
  - Ha *h* optimális és $\forall t\in T$: $\forall(n,t)\in A$: $h(n)+\alpha\geq c(n,t)$ akkor $g(t)\leq f^*(s)+\alpha$
- A mohó *A algoritmus* optimális heurisztika mellett akkor garantálja az optimális megoldást, ha
  - $\forall t\in T$: $\forall(n,t)\in A$: $h(n)=c(n,t)$ vagy
  - *h* monoton és $\exists\alpha\geq 0$: $\forall t\in T$: $\forall(n,t)\in A$: $h(n)+\alpha=c(n,t)$

91

Lemma

- Az A^c *algoritmus* (!) által kiterjesztésre választott bármely *n* csúcsra teljesül, hogy $f(n)\leq f^*(s)$.
- Bizonyítás: Tegyük fel, hogy a reprezentációs gráfban létezik megoldás, így $s^*\rightarrow t$ optimális út is. Ellenkező esetben az $f^*(s)=\infty$.
- $\exists m\in s^*\rightarrow t$ úgy, hogy $m\in NYÍLT$ és $g(m)=g^*(m)$ (*3.2. lemma*).
- De az algoritmus az *n* csúcsot választotta ki az *m* helyett
  - $f(n) \leq f(m) = g(m)+h(m) =$
  - $= g^*(m)+h(m) \leq g^*(t)+h(t) = g^*(t) = f^*(s)$.

92

Tétel

- Az A^c *algoritmus* (!) optimális megoldás megtalálásával terminál feltéve, hogy létezik megoldás.
- Bizonyítás:
- Megegyezik a *3.5. tétel* bizonyításával, csak most a *3.5. lemma* helyett az előző *lemmára* kell hivatkoznunk.

93

Tétel

- Minden A^c *algoritmus* A^* *algoritmus* is.
- Bizonyítás: Be kell látni: $h(n) \leq h^*(n) \quad \forall n\in N$
- Ha az *n* csúcsból nem vezet út a célcsúcsba, akkor a h^* értékét végtelen nagynak véve magától értetődik az állítás.
- Ha viszont létezik ilyen út, akkor van $n=n_0, n_1, ..., n_k = t$ optimális út is.

94

- Ennek éleire írjuk fel a monoton megszorítást feltételét:
  - $h(n)-h(n_1) \leq c(n,n_1)$
  - $h(n_1)-h(n_2) \leq c(n_1,n_2)$
  - ...
  - $h(n_{k-1})-h(t) \leq c(n_{k-1},t)$
- Adjuk össze ezeket az egyenlőtlenségeket:
  - $h(n)-h(t) \leq \Sigma\, c(n_{i-1},n_i) = h^*(n)$.
- Mivel *t* egy célcsúcs, ezért $h(t)=0$, tehát $h(n) \leq h^*(n)$

95