Gráfalgoritmusok

Gaskó Noémi

2023. március 19.

Tartalomjegyzék

Múlt órán

- hidak, elvágó pontok, kétszeresen összefüggő komponensek
- fák

Steiner fa

- a feladatban meg van adva a csomópontok egy részhalmaza is, a feladat a minimális feszitőfa meghatározása, amely kötelező módon tartalmazza a megadott csomópontokat
- Karp 21 NP-teljes feladatának egyike
- ha két elemet tartalmaz a feladat: minimális út
- ha az összes csomópontot tartalmazza, akkor a feladat megegyezik a minimális feszitőfával

Példa: lásd 4_jegyzet.pdf

Minimális szűk feszítőfa (MBST)

Értelmezés

Egy gráf minimális szűk feszítőfája egy olyan feszítőfa, melynek a legköltségesebb éle minimális.

Tétel

Egy minimális feszítőfa egyben minimális szűk feszítőfa, de egy minimális szűk feszítőfa nem mindig minimális feszítőfa.

Camerini algoritmusa

- divide et impera, két részre osztjuk az élek halmazát, medián alatt és felett
- az A halmaz bejárása a T erdő
- ha T feszítőfa, akkor rekurziv meghívás csak az A-ra
- ha nem, akkor több komponens, összehuzzuk és meghivjuk az összehúzott csomópontokra és a B-re
- O(m) idő

Példa: 4_jegyzet.pdf

Egyéb változatok

- fokszámkorlázozott minimális feszítőfa
- maximális feszítőfa

Algoritmusok

Egy csúcsból minden csúcsba:

- Moore algoritmusa

Algoritmusok

Egy csúcsból minden csúcsba:

- Moore algoritmusa
- Dijkstra algoritmusa

Algoritmusok

Egy csúcsból minden csúcsba:

- Moore algoritmusa
- Dijkstra algoritmusa
- Ford algoritmusa

Algoritmusok

Egy csúcsból minden csúcsba:

- Moore algoritmusa
- Dijkstra algoritmusa
- Ford algoritmusa
- A* algoritmus

Algoritmusok

Egy csúcsból minden csúcsba:

- Moore algoritmusa
- Dijkstra algoritmusa
- Ford algoritmusa
- A* algoritmus

Minden csúcsból egy csúcsba:

Algoritmusok

Egy csúcsból minden csúcsba:

- Moore algoritmusa
- Dijkstra algoritmusa
- Ford algoritmusa
- A* algoritmus

Minden csúcsból egy csúcsba:

- Bellman-Kalaba algoritmus

Algoritmusok

Egy csúcsból minden csúcsba:

- Moore algoritmusa
- Dijkstra algoritmusa
- Ford algoritmusa
- A* algoritmus

Minden csúcsból egy csúcsba:

- Bellman-Kalaba algoritmus

Minden csúcsból minden csúcsba

Algoritmusok

Egy csúcsból minden csúcsba:

- Moore algoritmusa
- Dijkstra algoritmusa
- Ford algoritmusa
- A* algoritmus

Minden csúcsból egy csúcsba:

- Bellman-Kalaba algoritmus

Minden csúcsból minden csúcsba

- Johnson algoritmus
- Floyd-Warshall algoritmus

Legrövidebb utak - algoritmusok - Hol használjuk?

- útvonaltervezés

Legrövidebb utak - algoritmusok - Hol használjuk?

- útvonaltervezés
- üzenetek továbbitása hálozatokban

Legrövidebb utak - algoritmusok - Hol használjuk?

- útvonaltervezés
- üzenetek továbbitása hálozatokban
- pénzváltás - mi előnyősebb?

Legrövidebb utak - algoritmusok - Hol használjuk?

- útvonaltervezés
- üzenetek továbbitása hálozatokban
- pénzváltás - mi előnyősebb?
- ütemezések

Legrövidebb utak - algoritmusok - Hol használjuk?

- útvonaltervezés
- üzenetek továbbitása hálozatokban
- pénzváltás - mi előnyősebb?
- ütemezések
- robotok mozgásának a megtervezése

Egy példa

Currency	£	Euro	¥	Franc	$	Gold
UK Pound	1.0000	0.6853	0.005290	0.4569	0.6368	208.100
Euro	1.4599	1.0000	0.007721	0.6677	0.9303	304.028
Japanese Yen	189.050	129.520	1.0000	85.4694	120.400	39346.7
Swiss Franc	2.1904	1.4978	0.011574	1.0000	1.3941	455.200
US Dollar	1.5714	1.0752	0.008309	0.7182	1.0000	327.250
Gold (oz.)	0.004816	0.003295	0.0000255	0.002201	0.003065	1.0000

Egy példa

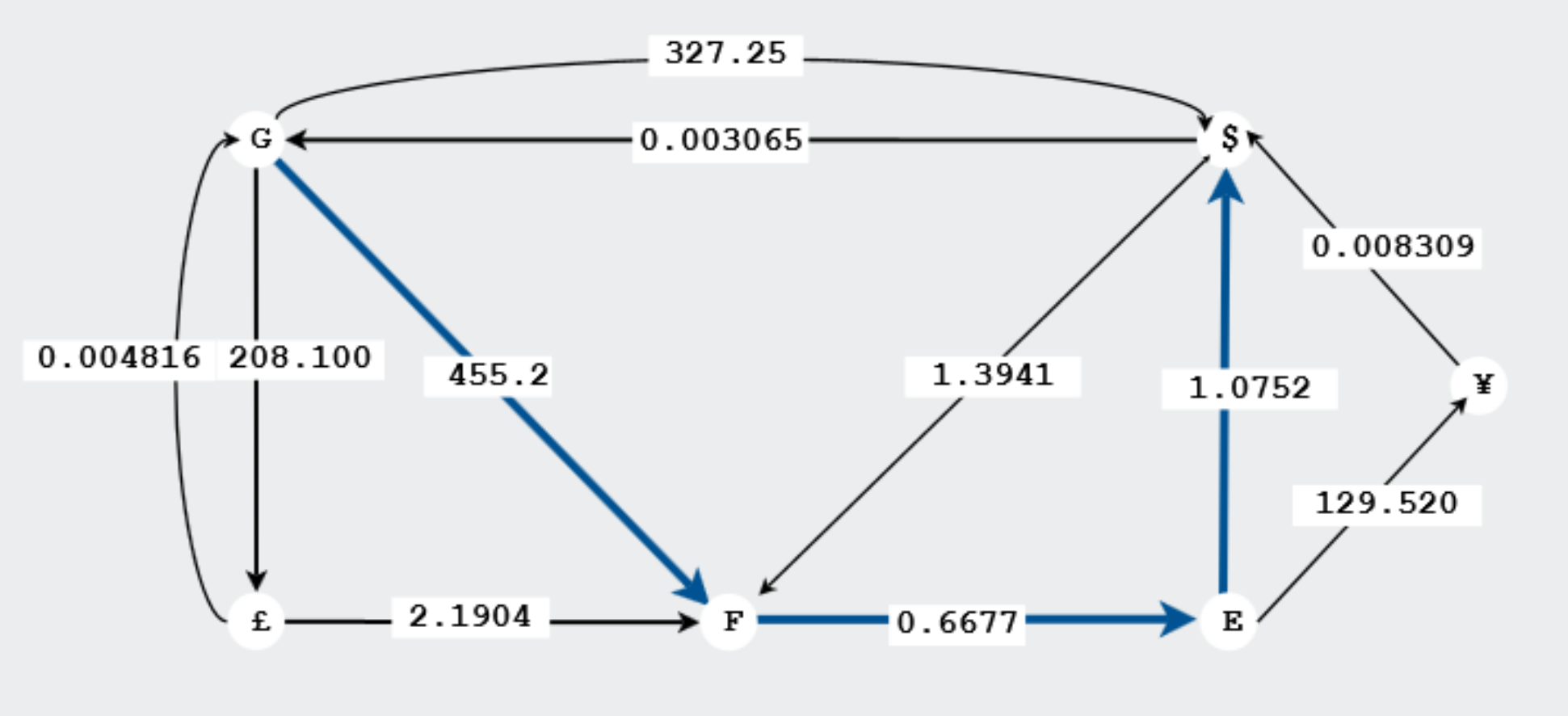
Currency	£	Euro	¥	Franc	$	Gold
UK Pound	1.0000	0.6853	0.005290	0.4569	0.6368	208.100
Euro	1.4599	1.0000	0.007721	0.6677	0.9303	304.028
Japanese Yen	189.050	129.520	1.0000	85.4694	120.400	39346.7
Swiss Franc	2.1904	1.4978	0.011574	1.0000	1.3941	455.200
US Dollar	1.5714	1.0752	0.008309	0.7182	1.0000	327.250
Gold (oz.)	0.004816	0.003295	0.0000255	0.002201	0.003065	1.0000

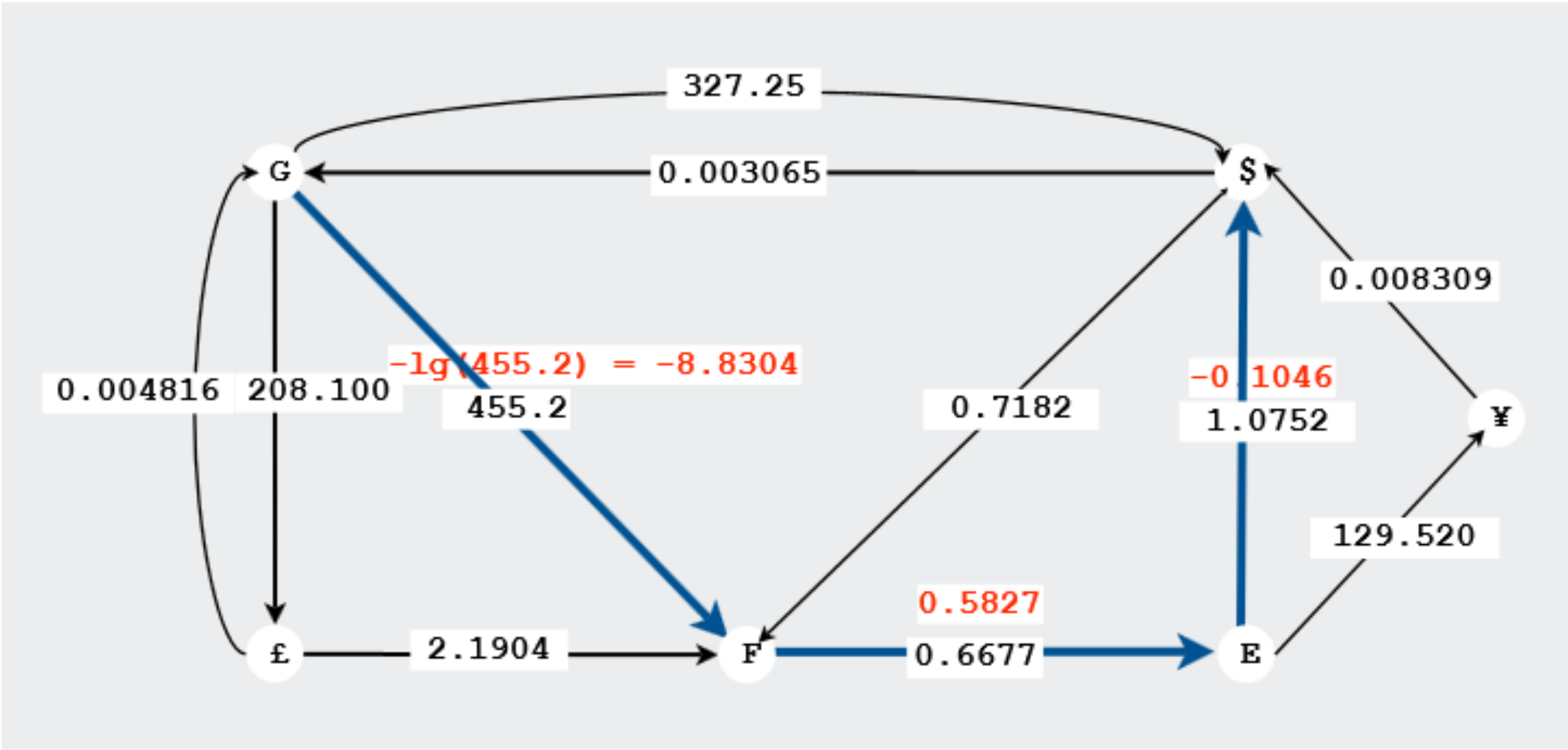
Hogy lesz ebbõl gráf?

Egy példa

Currency	£	Euro	¥	Franc	$	Gold
UK Pound	1.0000	0.6853	0.005290	0.4569	0.6368	208.100
Euro	1.4599	1.0000	0.007721	0.6677	0.9303	304.028
Japanese Yen	189.050	129.520	1.0000	85.4694	120.400	39346.7
Swiss Franc	2.1904	1.4978	0.011574	1.0000	1.3941	455.200
US Dollar	1.5714	1.0752	0.008309	0.7182	1.0000	327.250
Gold (oz.)	0.004816	0.003295	0.0000255	0.002201	0.003065	1.0000

Hogy lesz ebbõl gráf?
csomópont: pénznem
élek: a váltási összegek


327.25
G
0.003065
$
0.008309
0.004816
208.100
455.2
1.3941
1.0752
¥
129.520
£
2.1904
F
0.6677
E


327.25
G
0.003065
$
0.008309
-lg(455.2) = -8.8304
0.004816
208.100
455.2
0.7182
-0.1046
1.0752
¥
129.520
0.5827
£
2.1904
F
0.6677
E

Jelölések az algoritmusokban

- u - kezdőcsúcs
- $l(v)$ - v-nek az u-tól való távolsága
- $p(v)$ - a legrövidebb úton a v-t megelőző csúcs
- Q egy sor

Legrövidebb út nem irányított gráfokban

Moore algoritmusa - szélességi bejáráson alapul

Legrövidebb út nem irányított gráfokban

Moore algoritmusa - szélességi bejáráson alapul

MooreTávolság(G, u)

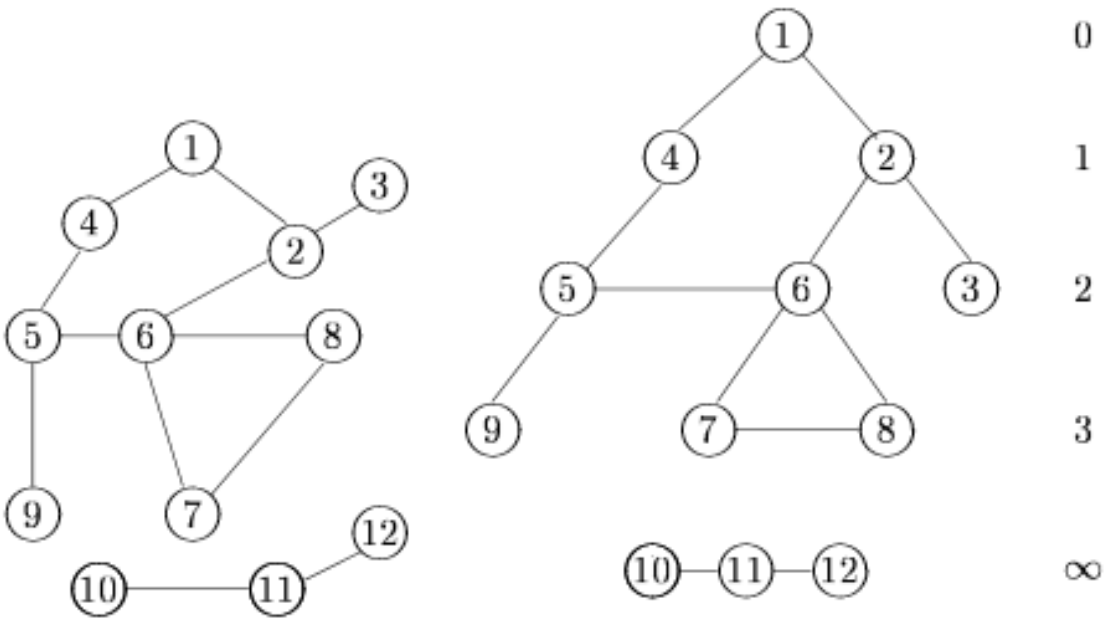
1. $l(u) := 0$
2. **for** minden $v \in V(G),\ v \neq u$ csúcsra **do**
3. $\quad l(v) := \infty$
4. legyen Q egy üres sor
5. $u \to Q$
6. **while** $Q \neq \emptyset$ **do**
7. $\quad Q \to x$
8. $\quad$ **for** minden $y \in N(x)$ **do**
9. $\quad\quad$ **if** $l(y) = \infty$ **then**
10. $\quad\quad\quad p(y) := x$
11. $\quad\quad\quad l(y) := l(x) + 1$
12. $\quad\quad\quad y \to Q$
13. **return** l, p

Algoritmus a legrövidebb u–v utak keresésére

MooreÚt(l, p, v)

1. $k := l(v)$
2. $u_k := v$
3. **while** $k \neq 0$ **do**
4. $\quad u_{k-1} := p(u_k)$
5. $\quad k := k - 1$
6. **return** u

Példa



	1	2	3	4	5	6	7	8	9	10	11	12
l	0	1	2	1	2	2	3	3	3	∞	∞	∞
p		1	2	1	4	2	6	6	5			

Melyik algoritmus származik tőle?

Dijkstra algoritmus

Dijkstra_queue(G)

1: INIT_S(G,s)
2: $P = \varnothing$
3: $Q = V$
4: **while** $Q \neq \varnothing$ **do**
5: $u =$ EXTRACT_MIN(Q)
6: $P = P \cup \{u\}$
7: **for** $v \in G.Adj[u]$ **do**
8: RELAX(u,v,w)

Dijkstra

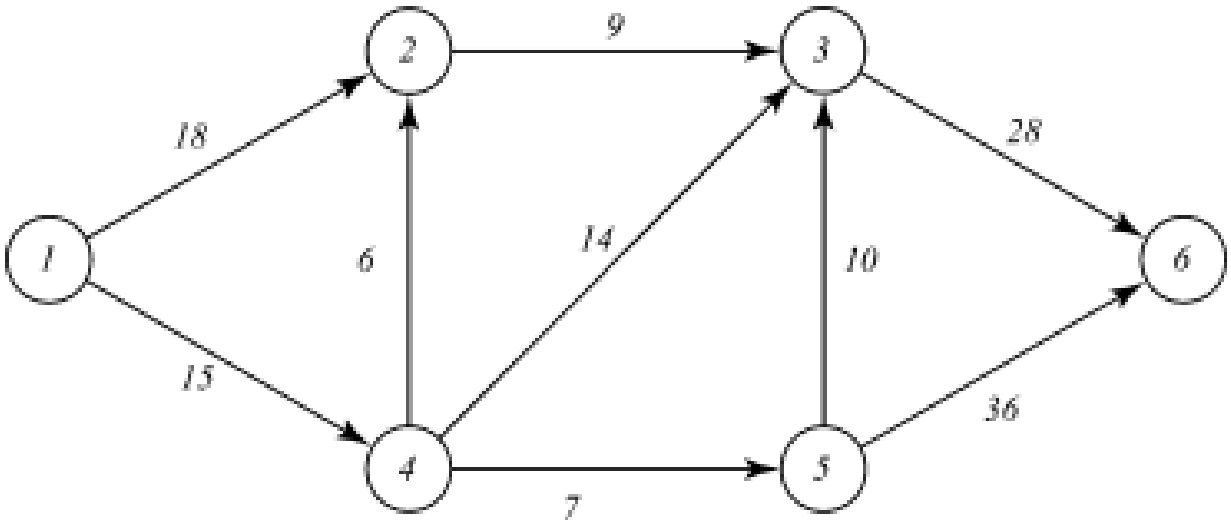
INIT_S(G,s)

1: **for** $v \in V$ **do**
2: $\quad v.d = \infty$
3: $\quad v.\pi = NIL$
4: $s.d = 0$

RELAX(u, v, w)

1: **if** $v.d > u.d + w(u, v)$ **then**
2: $\quad v.d = u.d + w(u, v)$
3: $\quad v.\pi = u$

Dijsktra algoritmus - példa



Megoldás: lásd szemináriumon

Hogyan gyorsíthatjuk a Dijkstra algoritmust?

Kupacok (Heaps) segítségével.

Mi a kupac?
egy adatszerkezet

Kupacok típusai:

- bináris kupac
- Fibonacci kupac
- B-kupac
- ...

Műveletek kupacokkal: beszúrás, törlés, stb.
C++-ban használható: priority_queue vagy saját implementáció

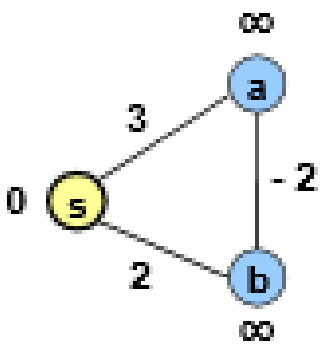
Mikor nem működik Dijkstra algoritmusa?

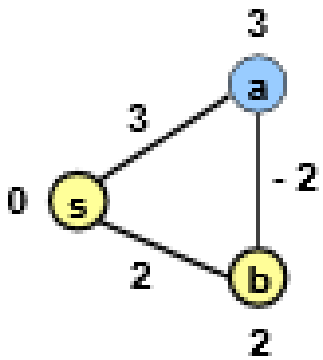
Mikor nem működik Dijkstra algoritmusa?

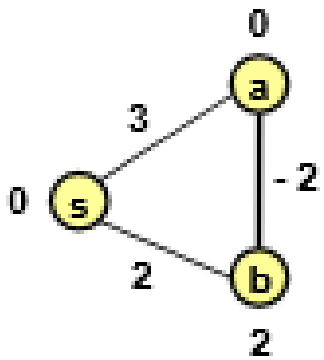
Negatív élek esetén

Mikor nem működik Dijkstra algoritmusa?

Negatív élek esetén
Egy példa:







Dijkstra

Tétel (Dijkstra helyessége)

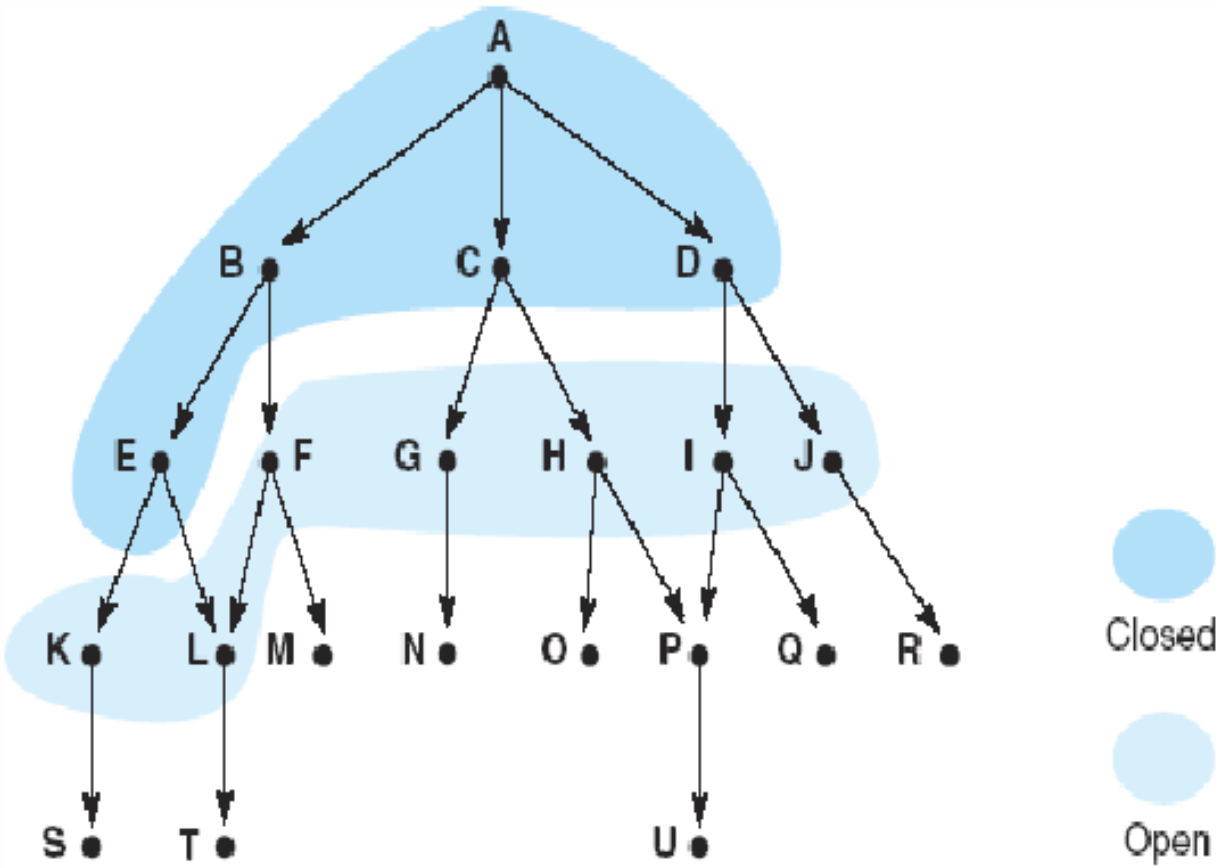
A *Dijkstra_queue(G)* algoritmus esetén, a $G = (V, E)$ gráfra, mely nem tartalmaz negatív súlyokat és s a kezdő csomópont, a végén $u.d = \delta(s, u), \forall u \in V$.

Best-first search algoritmus

-van egy heurisztikus függvény (h(x)), amley megadja a megbecsült költségét az adott célnak

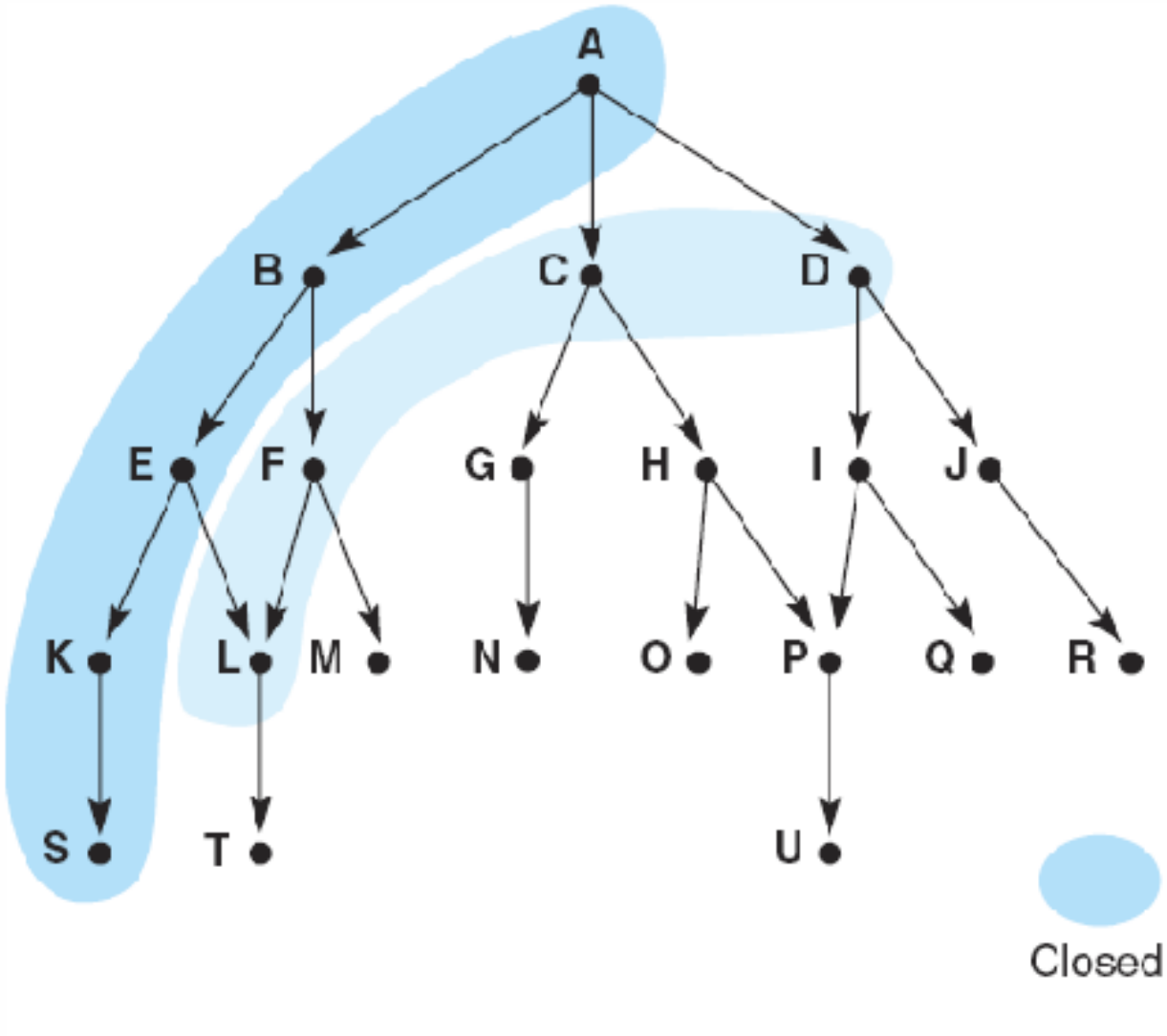
Egy példa

Milyen tipusú bejárás ?



Egy példa

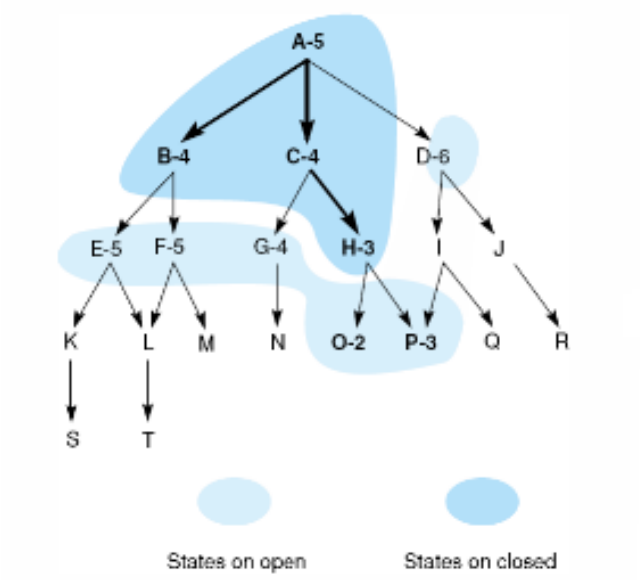
Milyen tipusú bejárás ?

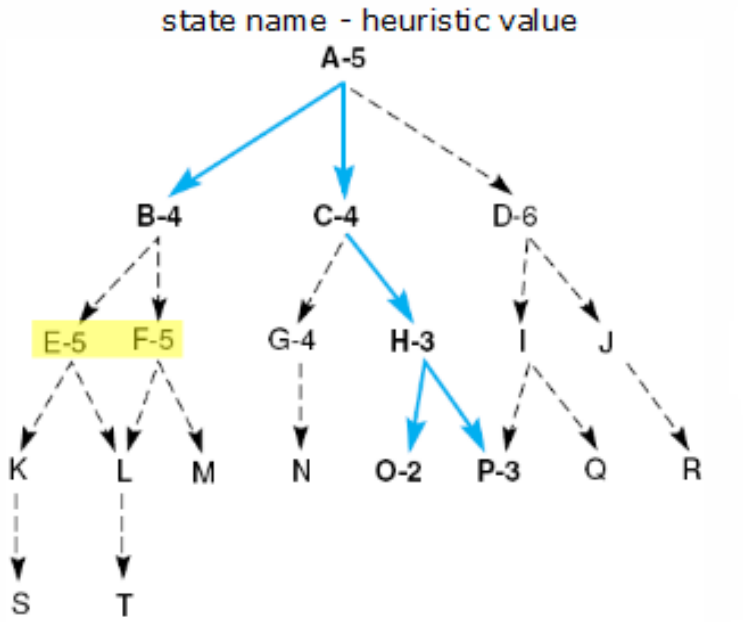


Egy példa

Milyen tipusú bejárás ?

Best first: Trace





1. open=[A5]; closed=[ ]
2. Visit A5; open=[B4,C4,D6]; closed=[A5] Ordered by heuristic values
3. Visit B4; open=[C4,E5,F5,D6]; closed=[B4,A5]
4. visit C4; open=[H3,G4,E5,F5,D6]; closed=[C4,B4,A5]
5. visit H3; open=[O2,P3,G4,E5,F5,D6];
closed=[H3,C4,B4,A5]

.....

A* algoritmus

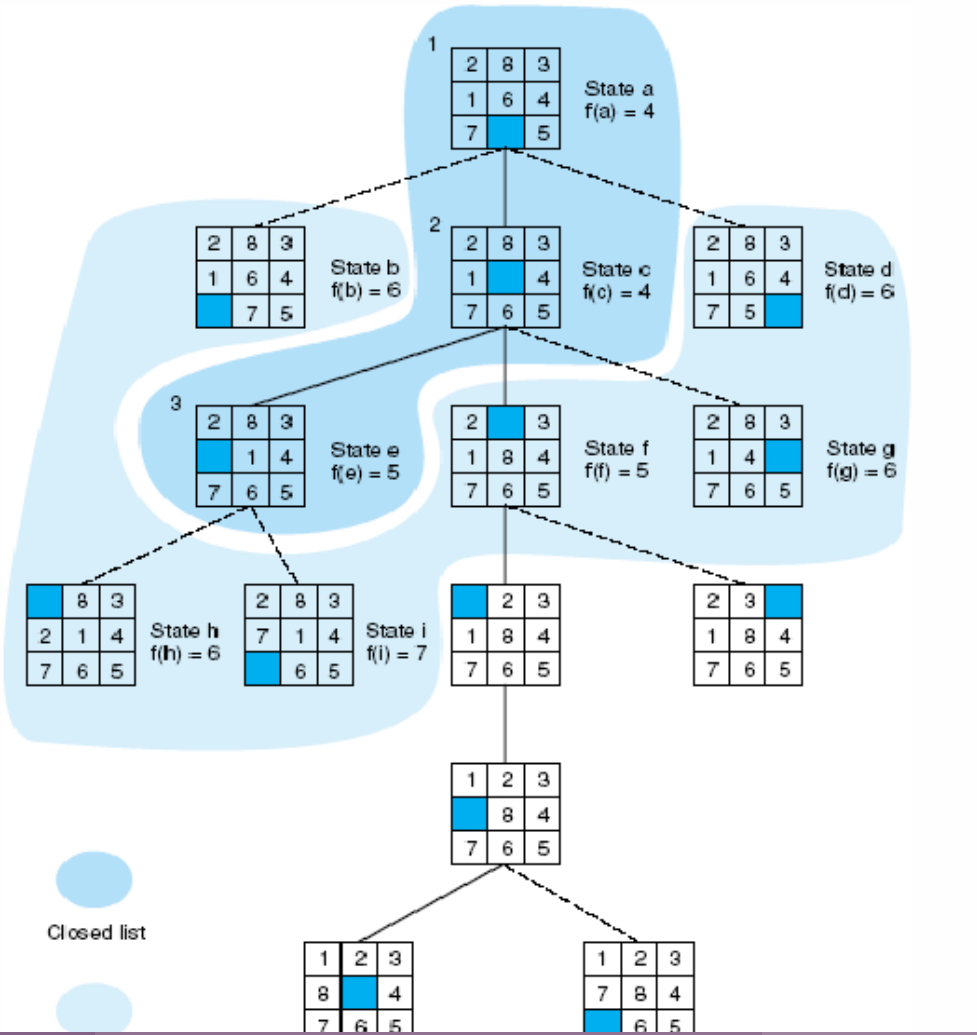
-best-first search algoritmuson alapul

A* algoritmus

-best-first search algoritmuson alapul
-1968-ban írták le elõször

A* algoritmus

-best-first search algoritmuson alapul
-1968-ban írták le elõször -megoldás például erre is:



Egy példa

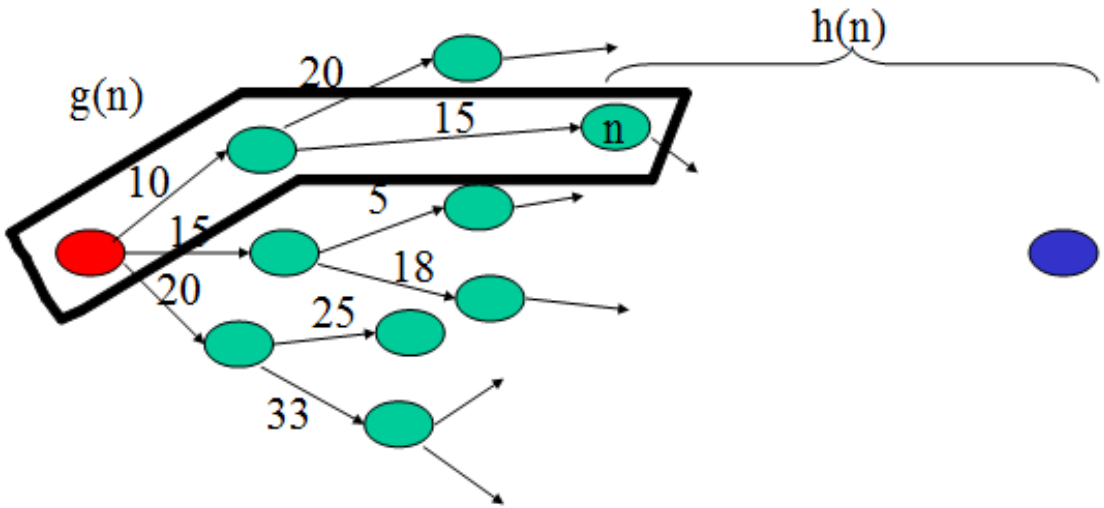
-komplexitása függ a megválasztott heurisztikától

Egy példa

-komplexitása függ a megválasztott heurisztikától

A*

- f(n) = g(n) + h(n)
  - g(n) = "cost from the starting node to reach n"
  - h(n) = "estimate of the cost of the cheapest path from n to the goal node"



Bellman-Ford algoritmus

Bellman_Ford(G)

1: INIT_S(G,s)
2: **for** $i = 1$ **to** $|V| - 1$ **do**
3: **for** minden élre $\{u, v\} \in E$ **do**
4: RELAX(u, v, w)
5: **for** minden élre $\{u, v\} \in E$ **do**
6: **if** $v.d > u.d + w(u, v)$ **then**
7: **return** FALSE
8: **return** TRUE

Bellman-Ford (II)

INIT_S(G,s)

1: **for** $v \in V$ **do**
2: $\quad v.d = \infty$
3: $\quad v.\pi = NIL$
4: $s.d = 0$

RELAX(u, v, w)

1: **if** $v.d > u.d + w(u, v)$ **then**
2: $\quad v.d = u.d + w(u, v)$
3: $\quad v.\pi = u$

Bellman-Ford (III)

- $O(VE)$ futási idő
- INIT_S lépés időtartama $\Theta(V)$
- 2-4 sorok futása $\Theta(E)$
- a for az 5-7 sorokban $O(E)$

Bellman-Ford

Lemma

Legyen $G = (V, E)$ egy súlyozott irányított gráf s a kezdeti csúcs, $w : E \to \mathbb{R}$ a súlyok, feltételezzük, hogy G nem tartalmaz s-ből elérhető negativ kört. A 2-4 sorokban található $|V| - 1$ iterációja után a *Bellman_Ford(G)* algoritmusnak: $v.d = \delta(s, v)$ minden v csomópontra az s-ből.

Következmény

Legyen $G = (V, E)$ egy súlyozott irányított gráf s a kezdeti csúcs, $w : E \to \mathbb{R}$ a súlyok. Minden $v \in V$ csomópont esetén létezik egy út s-ből v-be akkor és csakis akkor, ha *Bellman_Ford(G)* $v.d < \infty$-el fejeződik be.

Bellman-Ford

Tétel (a Bellman-Ford algoritmus helyessége)

Legyen a *Bellman_Ford(G)* eljárás, melyet a súlyozott és irányított $G = (V, E)$ gráfon futattunk s kezdőpontból, $w : E \to \mathbb{R}$ a súlyfőggvény. Ha G nem tartalmaz s-ből elérhető negativ kört, akkor az algoritmus *TRUE*-t térít vissza, $v.d = \delta(s, v), \forall v \in V$ mig a szülők gráfja G_π egy minimáls feszítőfa s gyökérrel. Ha G tartalmaz s-ből elérhető negativ kört, akkor az algoritmus visszatérítési értéke *FALSE*.

Johnson algoritmusa

-1977-ben

Johnson algoritmusa

-1977-ben -negatív élű súlyok esetén is működik

Johnson algoritmusa

-1977-ben -negatív élű súlyok esetén is működik
Az algoritmus lépései:

- hozzáadunk egy új csomópontot (q), amely minden csomóponthoz fog kapcsolódni

Johnson algoritmusa

-1977-ben -negatív élű súlyok esetén is működik
Az algoritmus lépései:

- hozzáadunk egy új csomópontot (q), amely minden csomóponthoz fog kapcsolódni
- összekötjük az összes csomóponttal egy-egy 0 költségű éllel
- lefuttatjuk a Bellman -Ford algoritmust (q a kezdőpont)

Johnson algoritmusa

-1977-ben -negatív élű súlyok esetén is működik
Az algoritmus lépései:

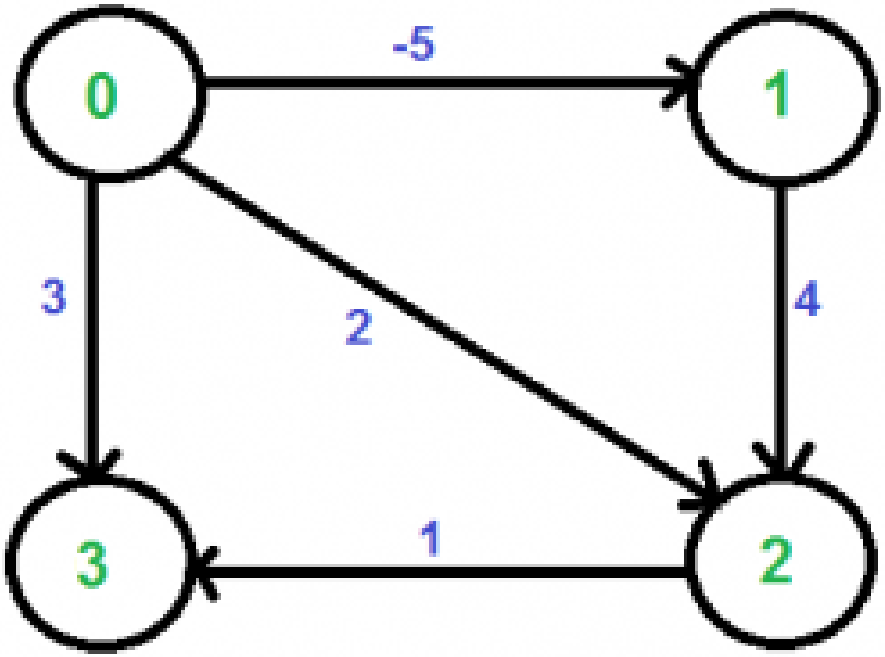
- hozzáadunk egy új csomópontot (q), amely minden csomóponthoz fog kapcsolódni
- összekötjük az összes csomóponttal egy-egy 0 költségű éllel
- lefuttatjuk a Bellman -Ford algoritmust (q a kezdőpont)
- újrasúlyozuk az éleket: $w(u, v) = w(u, v) + h[u] - h[v]$

Johnson algoritmusa

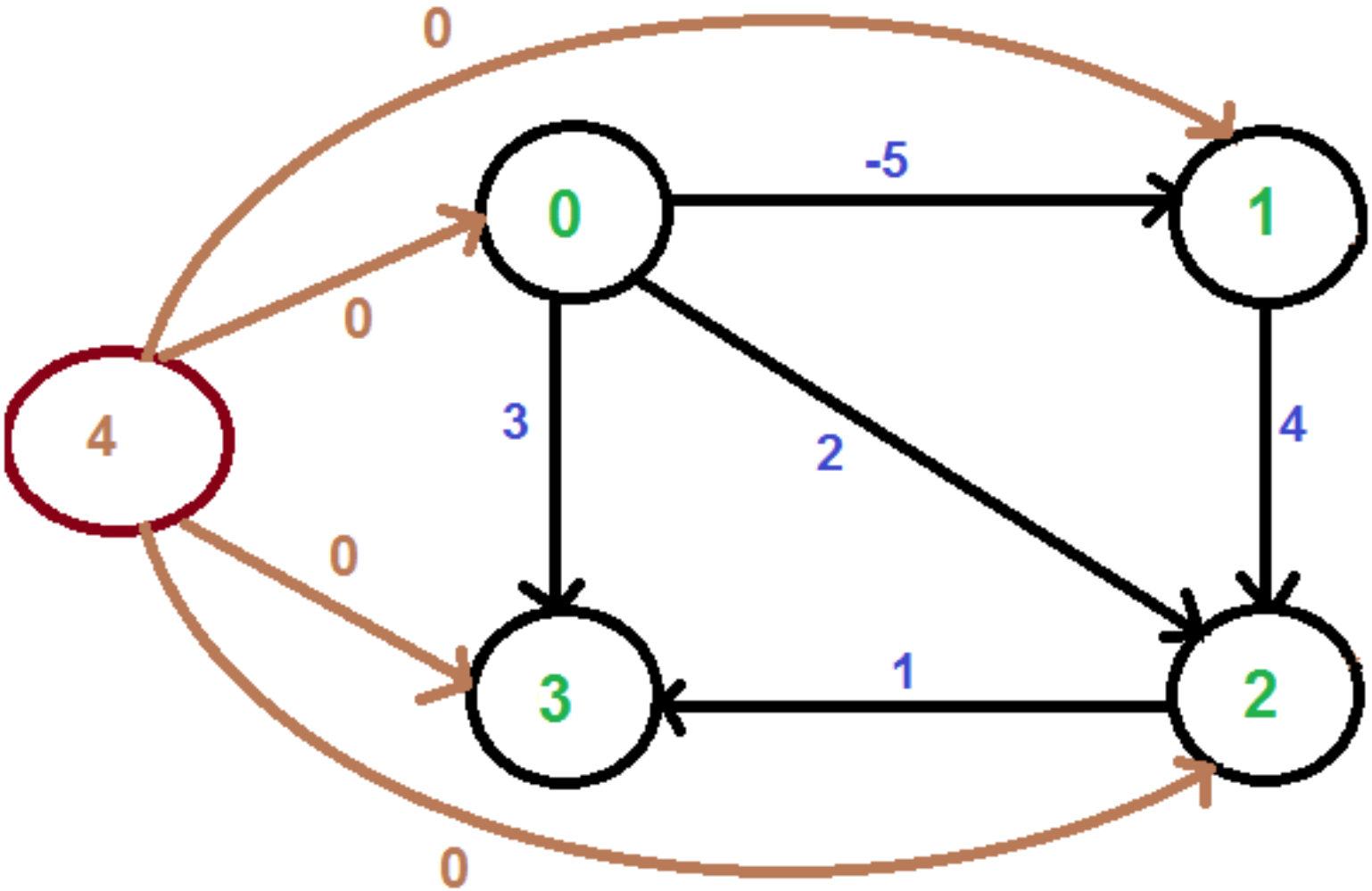
-1977-ben -negatív élű súlyok esetén is működik
Az algoritmus lépései:

- hozzáadunk egy új csomópontot (q), amely minden csomóponthoz fog kapcsolódni
- összekötjük az összes csomóponttal egy-egy 0 költségű éllel
- lefuttatjuk a Bellman -Ford algoritmust (q a kezdőpont)
- újrasúlyozuk az éleket: $w(u, v) = w(u, v) + h[u] - h[v]$
- q-t kiszedjük, és Dijkstra algoritmusát futtatjuk minden csomópontból kiindulva

Johnsons algoritmus

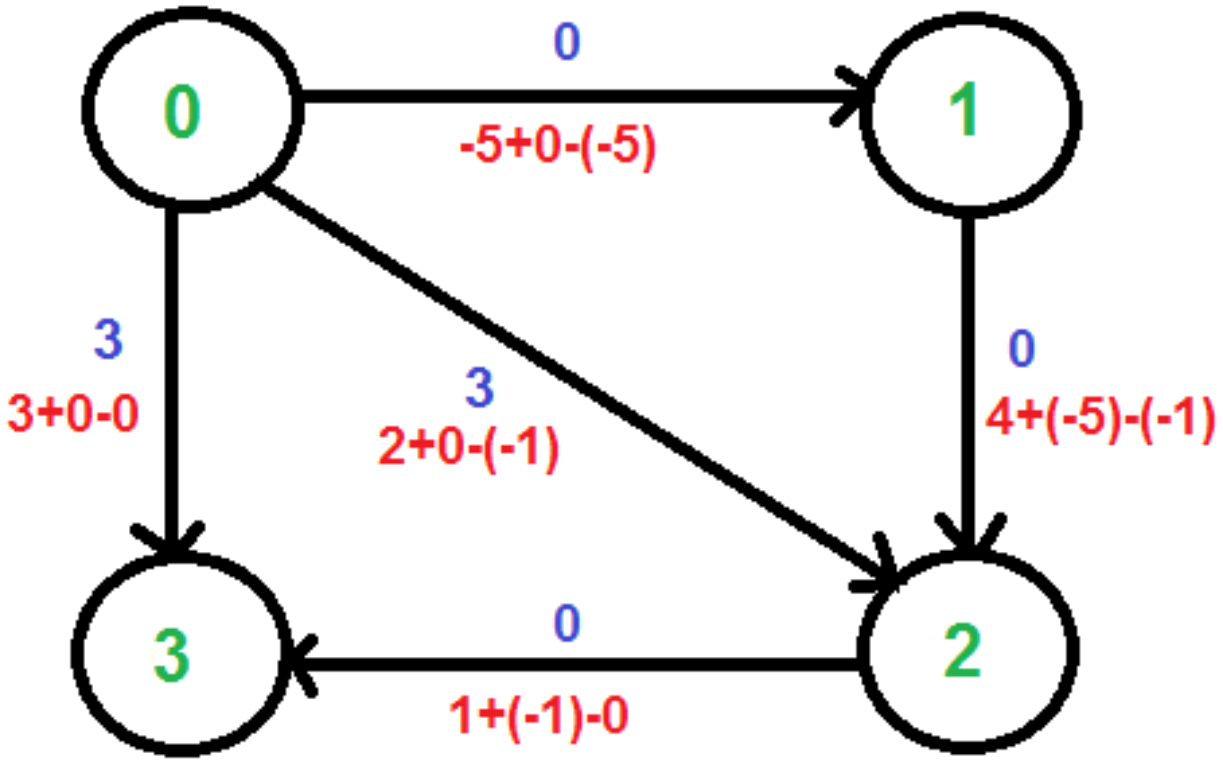


Johnsons algoritmus



Johnsons algoritmus

$$w(u,v) = w(u,v) + h[u] - h[v]$$



Distances from 4 to 0, 1, 2 and 3 are 0, -5, -1 and 0 respectievely.

Bellman-Kalaba algoritmus

- dinamikus programozás módszere
- negatív súlyok is lehetnek a gráfban
- a gráf szomszédsági mátrixát használja
- negativ köröket is megtalál

Bellman-Kalaba algoritmus

kiválasztunk egy csomópontot (egy oszlopot a szomszédsági mátrixból) és minden csomópontból meghatározzuk a távolságot ebbe a csomópontba. Ennek a jelölése $V^{(1)} = (V_i^{(1)})_{i=\overline{1,n}}$, a szomszédsági mátrix $A = (a_{ij})_{i,j=\overline{1,n}}$, ahol $a_{ij} = d_{ij}^{(0)}$. A szomszédsági mátrix elemei a következőek:

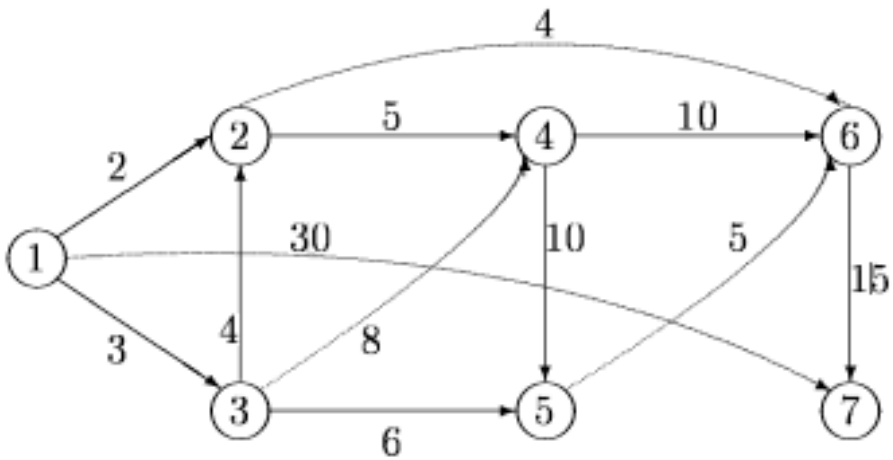
$$a_{ij} = \begin{cases} w(v_i, v_j) & \text{ha } (v_i, v_j) \in E, \\ 0 & \text{ha } i = j, \\ \infty & \text{ha } (v_i, v_j) \notin E. \end{cases}$$

Meghatározzuk $k = 1, 2, ..., n$ esetén:

$$V_i^{(k)} = \min_{j=\overline{1,n}} \{a_{ij} + V_j^{(k-1)}\}, \text{ minden} i = 1, 2, ..., n$$

amig $V^{(t)} = V^{(t-1)}$ (t az időpillanatot, iterációt jelöli).

Egy példa



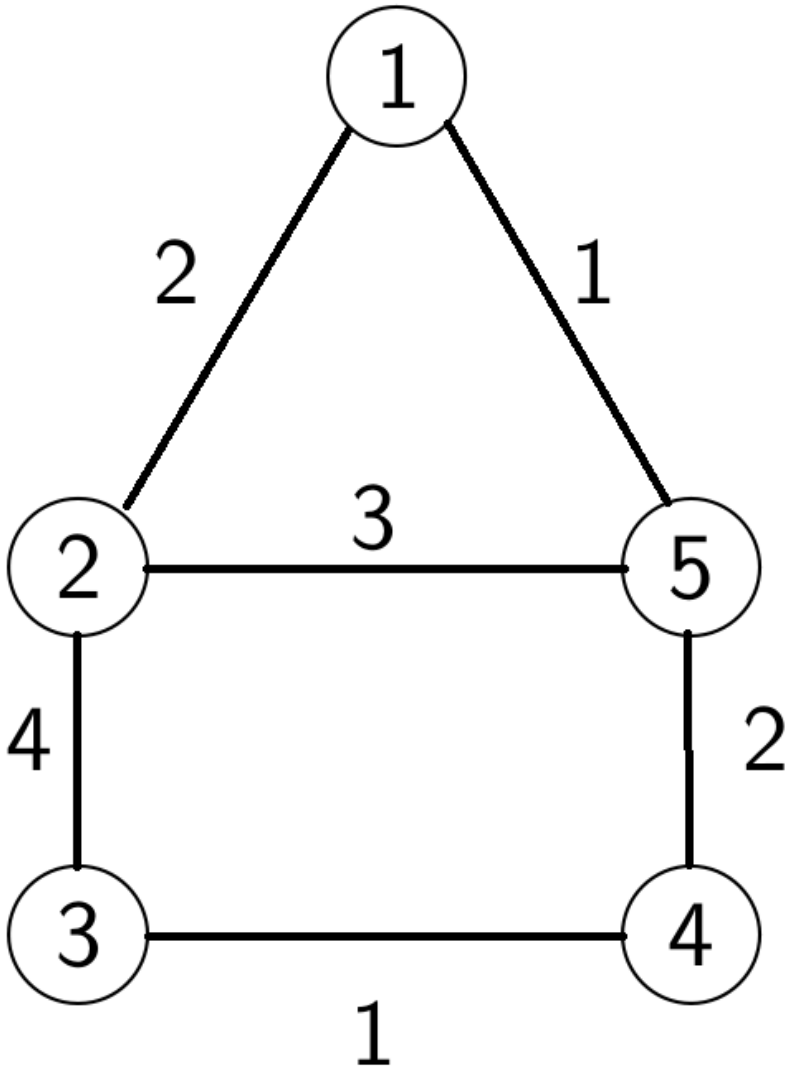
	1	2	3	4	5	6	7	$V^{(1)}$	$V^{(2)}$	$V^{(3)}$	$V^{(4)}$
1	0	2	3	∞	∞	∞	30	30	30	21	21
2	∞	0	∞	5	∞	4	∞	∞	19	19	19
3	∞	4	0	8	6	∞	∞	∞	∞	23	23
4	∞	∞	∞	0	10	10	∞	∞	25	25	25
5	∞	∞	∞	∞	0	5	∞	∞	20	20	20
6	∞	∞	∞	∞	∞	0	15	15	15	15	15
7	∞	∞	∞	∞	∞	∞	0	0	0	0	0

Floyd-Warshall algoritmus

Warshall algoritmus

Warshall(D_0)
1. $D := D_0$
2. **for** $k := 1$ **to** n **do**
3. **for** $i := 1$ **to** n **do**
4. **for** $j := 1$ **to** n **do**
5. $d_{ij} := \min(d_{ij}, d_{ik} + d_{kj})$
6. **return** D

Példa.



1. ábra. Egy súlyozott gráf

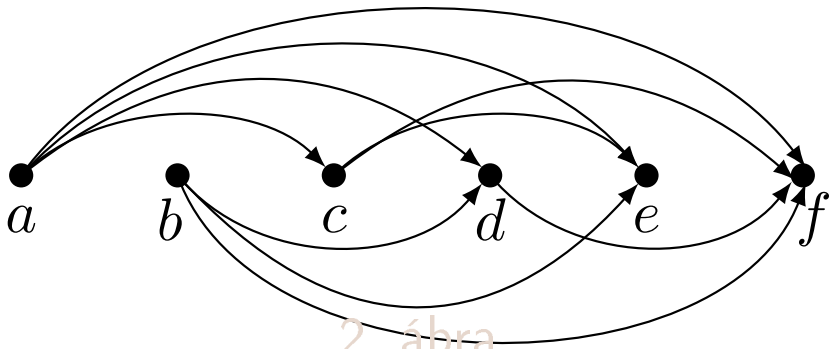
Utak száma gráfokban

FW(A, n)

FW(A, n)

1. $W := A$
2. **for** $k := 1$ **to** n **do**
3. **for** $i := 1$ **to** n **do**
4. **for** $j := 1$ **to** n **do**
5. $w_{ij} := w_{ij} + w_{ik}w_{kj}$
6. **return** W

Egy példa



2. ábra

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Az FW-algoritmus alkalmazása után:

$$W = \begin{pmatrix} 0 & 0 & 1 & 1 & 2 & 3 \\ 0 & 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Utak meghatározása

A fenti algoritmust a latin-négyzetes módszerrel kombinálva, meghatározhatjuk az utakat is.
Olyan szavakat konkatenálunk, amelyek nem tartalmaznak közös betűket.
A szomszédsági mátrix mintájára használjuk az $\mathcal{A}$ mátrixot, amelynek A_{ij} elemei a gráf csúcsaiból képzett szavak. Kezdetben ezek az éleket jelölik, később az utakat. Kezdetben

$$A_{ij} = \begin{cases} \{a_i a_j\}, & \text{ha } (a_i a_j) \text{ irányított él,} \\ \emptyset, & \text{különben,} \end{cases} \qquad i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, n.$$

Ha $\mathcal{A}$ és $\mathcal{B}$ szóhalmazok, legyen $\mathcal{A} \cdot \mathcal{B}$ az a halmaz, amelyet úgy képezünk, hogy $\mathcal{A}$ elemeit megszorozzuk (konkatenáljuk) $\mathcal{B}$ elemeivel, de csak akkor, ha a két szorzandó szó nem tartalmaz közös betűket.

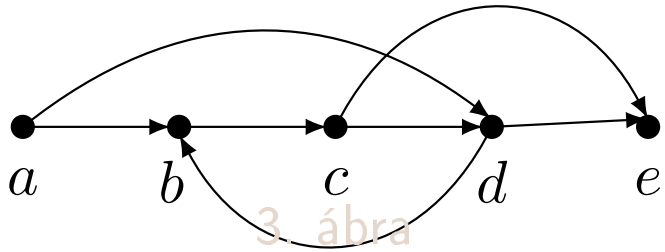
$$\mathcal{A} \cdot \mathcal{B} = \left\{ ab \,\middle|\, a \in \mathcal{A}, b \in \mathcal{B}, a \text{ és } b \text{ különböző betűkből áll} \right\}.$$

- Az $s = s_1 s_2 \cdots s_p$ szóból képezzük az $'s$ szót úgy, hogy elhagyjuk s első betűjét: $'s = s_2 s_3 \cdots s_p$.
- Képezzük az $'A_{ij}$ halmazt az A_{ij} halmazból, annak minden eleméből elhagyván az első betűt.
- Ekkor az $'\mathcal{A}$ mátrix elemei $'A_{ij}$.

Floyd–Warshall–Latin($\mathcal{A}, n$)

1. $W \leftarrow \mathcal{A}$
2. **for** $k \leftarrow 1$ **to** n
3. **for** $i \leftarrow 1$ **to** n
4. **for** $j \leftarrow 1$ **to** n
5. **if** $W_{ik} \neq \emptyset$ and $W_{kj} \neq \emptyset$
6. **then** $W_{ij} \leftarrow W_{ij} \cup W_{ik} \cdot' W_{kj}$
7. **return** W

Egy példa



3. ábra

$$\begin{pmatrix} \emptyset & \{adb, ab\} & \{adbc, abc\} & \{abcd, ad\} & \{ade, adbce, abcde, abce\} \\ \emptyset & \emptyset & \{bc\} & \{bcd\} & \{bcde, bce\} \\ \emptyset & \{cdb\} & \emptyset & \{cd\} & \{cde, ce\} \\ \emptyset & \{db\} & \{dbc\} & \emptyset & \{dbce, de\} \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \end{pmatrix}.$$

Algoritmusok bonyolultsága és összefoglaló

Egy csúcsból a többi csúcsba:

- Moore - $O(n+m)$ - súlyozatlan gráfok esetén
- Dijkstra - $O(mlogn)$ - nemnegatív költségű élek
- Bellman-Ford - $O(n \cdot m)$ - akármilyen gráf

Összes csúcsból az összes csúcsba:

- Roy-Floyd-Warshall - $O(n^3)$ - akármilyen gráf
- Johnson - $O(n \cdot mlogn)$ - akármilyen gráf

Lineáris programozás

Az általános probléma

legyen A $m \times n$-es mátrix, b egy m méretű és c egy n méretű tömb. Határozzuk meg az x n elemű tömböt, amely maximizálja a következő függvényt

$$\sum_{i=1}^{n} c_i x_i$$

és kielégíti az m megszorításokat:

$$Ax \leq b.$$

- néhány esetben csak az a fontos, hogy egy a megszorításokat kielgitő megoldást találjunk, vagy pedig azt kimutatni, hogy nincs ilyen megoldás

Megszorítási rendszerek

- egy megszorítási rendszerben minden sor az A mátrixban -1 illetve 1-ket tartalmaz, a többi érték 0
- igy a $Ax \leq b$ megadott megszorítások m megszorítást tartlamaznak n ismeretlennel, a megszorítások a következő egyenlőtlenségek lesznek

$$x_j - x_i \leq b_k,$$

ahol $1 \leq i, j \leq n, i \neq j$ és $1 \leq k \leq m$.

Példa

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \leq \begin{pmatrix} 0 \\ -1 \\ 1 \\ 5 \\ 4 \\ -1 \\ -3 \\ -3 \end{pmatrix}$$

Példa (II)

- a feladat, hogy olyan x_1, x_2, x_3, x_4, x_5 értékeket találjunk, amelyek kielégítik a 8 megszorítást

$$x_1 - x_2 \leq 0,$$

$$x_1 - x_5 \leq -1,$$

$$\ldots$$

- több megoldás létezik:

$$x = (-5, -3, 0, -1, -4)$$

$$x' = (0, 2, 5, 4, 1)$$

Megszorítási rendszerek (II)

Lemma 4.1

legyen $x = (x_1, x_2, ..., x_n)$ egy megoldás $Ax \leq b$ és d egy konstans. Ebben az esetben az $x + d = (x_1 + d, x_2 + d, ..., x_n + d)$ is megoldás a megszorításokra $Ax \leq b$.

Bizonyítás.

minden x_i és x_j esetén $(x_j + d) - (x_i + d) = x_j - x_i$. Ha x teljesíti $Ax \leq b$ akkor $x + d$ is egy megoldás. □

Megszorítási gráfok

- a megszorításokat kezelhetjük egy gráfként
- az $Ax \leq b$ megszorítási egyenlőtlenségek esetén, az A mátrix melynek mérete $m \times n$ tekinthető egy incidencia mátrixként melynek n csomópontja és m éle van
- minden $v_i \in V, i = 1, 2, ..., n$ csomópont egy x_i változónak felel meg
- minden $(i, j) \in E$ él egy egyenlőtlenségnek

Értelmezés

legyen $Ax \leq b$ egy megszorítási rendszer, a rendszernek megfelelő gráf egy irányított és súlyozott gráf $G = (V, E)$ ahol $V = \{v_0, v_1, ..., v_n\}$ és

$$\begin{aligned} E &= \{(v_1, v_j) | x_j - x_i \leq b_k \text{ megszorítás}\} \\ &\cup \{(v_0, v_1), (v_0, v_2), ..., (v_0, v_n)\} \end{aligned}$$

Megszorítási gráfok (II)

- a gráfhoz hozzáadunk egy v_0 csomópontot, amely minden más csomóponthoz kapcsolódik
- $v_i \in V, i = 1, ..., n$
- ha $(x_j - x_i \leq b_k$ akkor $w(v_i, v_j) = b_k$
- $w(v_0, v_i) = 0, \forall i = 1, ..., n$

Megszorítási gráfok (III)

Tétel

legyenek $Ax \leq b$ megszorítások és $G = (V, E)$ a megszorítások gráfja. Ha G nem tartalmaz negativ kört, akkor

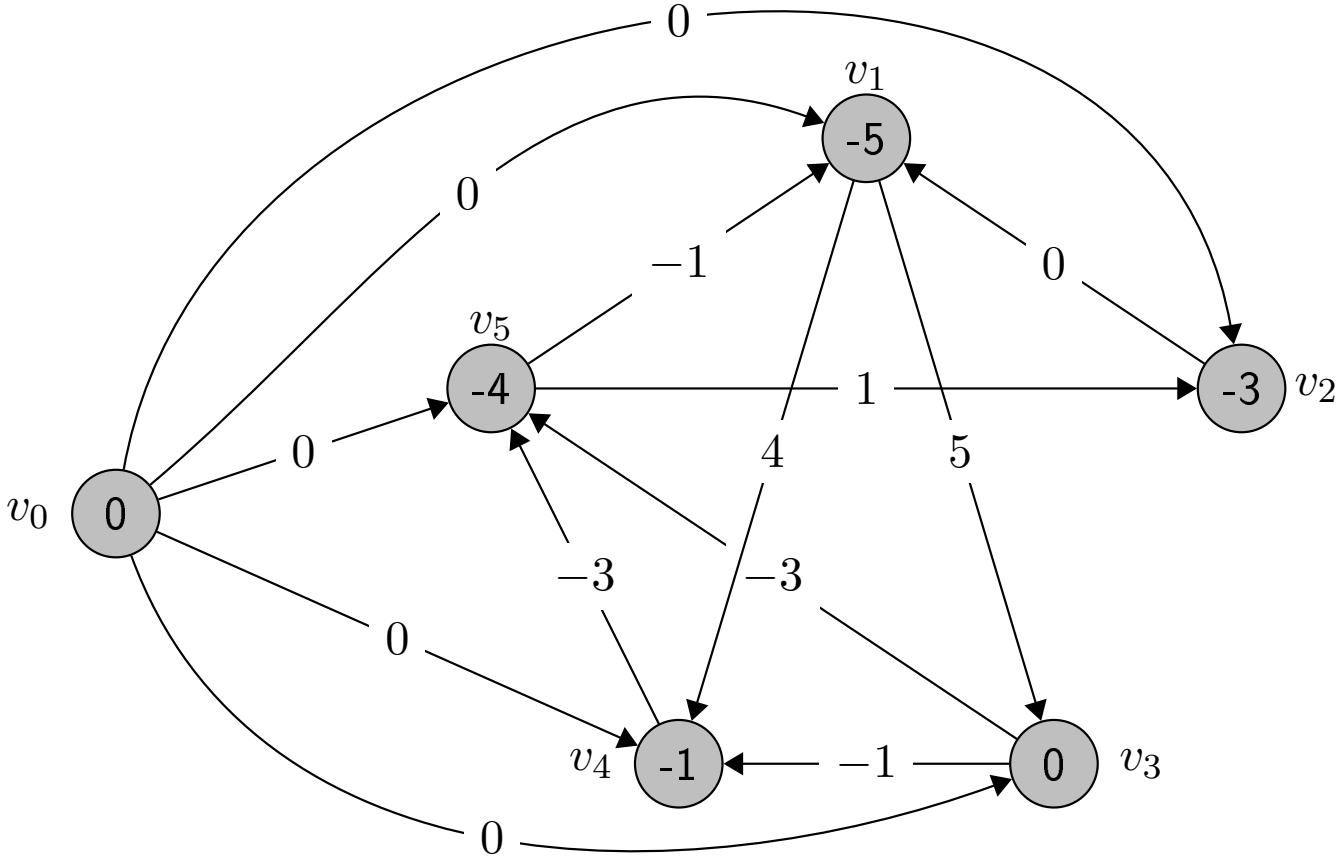
$$x = (\delta(v_0, v_1), \delta(v_0, v_2), ..., \delta(v_0, v_n))$$

egy megoldása az egyenletnek. Ha a gráf tartalmaz egy negativ kört, akkor nem létezik megoldás.

- a megoldást megkaphatjuk a legrövidebb út megadásával

Példa

- a $\delta(v_0, v_i)$ minden csomópontban megjelenik
- egy lehetséges megoldás $x = (-5, -3, 0, -1, -4)$



Forrásanyag

- http://www.cs.princeton.edu/courses/archive/spr10/cos226
- Kása jegyzet
- Santanu Saha Ray, Graph Theory with Algorithms and its Applications, Springer, 2013.