Algoritmusok és adatszerkezetek II. előadásjegyzet

Élsúlyozott gráfok és algoritmusaik

Ásványi Tibor – asvanyi@inf.elte.hu

2020. augusztus 9.

Tartalomjegyzék

Hivatkozások

[1] Ásványi Tibor, Algoritmusok és adatszerkezetek II.
Útmutatások a tanuláshoz, jelölések, tematika,
fák, gráfok,
mintaillesztés, tömörítés
http://aszt.inf.elte.hu/~asvanyi/ad/ad2jegyzet/

[2] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.,
magyarul: Új Algoritmusok, *Scolar Kiadó*, Budapest, 2003.
ISBN 963 9193 90 9
angolul: Introduction to Algorithms (Third Edititon),
The MIT Press, 2009.

[3] Fekete István, Algoritmusok jegyzet
http://ifekete.web.elte.hu/

[4] Koruhely Gábor, Szalay Richárd,
Algoritmusok és adatszerkezetek 2, 2015/16 tavaszi félév
(hallgatói jegyzet, lektorált és javított)
http://aszt.inf.elte.hu/~asvanyi/ad/

[5] Rónyai Lajos – Ivanyos Gábor – Szabó Réka, Algoritmusok,
TypoTEX Kiadó, 1999. ISBN 963 9132 16 0
https://www.tankonyvtar.hu/hu/tartalom/tamop425/2011-0001-526_ronyai_algoritmusok/adatok.html

[6] Tarjan, Robert Endre, Data Structures and Network Algorithms,
CBMS-NSF Regional Conference Series in Applied Mathematics, 1987.

[7] Weiss, Mark Allen, Data Structures and Algorithm Analysis,
Addison-Wesley, 1995, 1997, 2007, 2012, 2013.

[8] Carl Burch, Ásványi Tibor, B+ fák
http://aszt.inf.elte.hu/~asvanyi/ad/B+ fa.pdf

[9] Ásványi Tibor, Algoritmusok és adatszerkezetek I. előadásjegyzet (2019)
http://aszt.inf.elte.hu/~asvanyi/ad/ad1jegyzet.pdf

1. Élsúlyozott gráfok és ábrázolásaik ([2] 22)

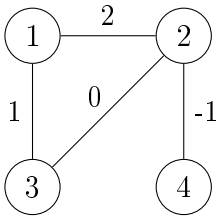
1.1. Definíció. Élsúlyozott gráf *alatt egy* $G = (V, E)$ *gráfot értünk a* $w : E \to \mathbb{R}$ *súlyfüggvénnyel, ahol* V *a csúcsok (vertices) tetszőleges, véges halmaza,* $E \subseteq V \times V \setminus \{(u, u) : u \in V\}$ *az élek (edges) halmaza.*

A $w : E \to \mathbb{R}$ *minden egyes élhez hozzárendeli annak* súlyát, *más néven* hosszát, *illetve* költségét. *(Az* élsúly, élhossz *és* élköltség *elnevezések szinonímák.)*

1.2. Definíció. *Élsúlyozott gráfban tetszőleges út* hossza, *más néven* költsége, *illetve* súlya *az út mentén található élek összsúlya. Hasonlóképpen tetszőleges gráf/fa súlya az élei súlyainak összege.*

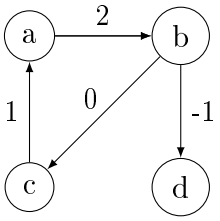
1.1. Grafikus ábrázolás

Az éleket súlyukkal címkézzük.



1 – 2, 2 ; 3, 1.
2 – 3, 0 ; 4, -1.

1. ábra. Ugyanaz az élsúlyozott irányítatlan gráf grafikus (balra) és szöveges (jobbra) ábrázolással.



a → b, 2.
b → c, 0 ; d, -1.
c → a, 1.

2. ábra. Ugyanaz az élsúlyozott irányított gráf grafikus (balra) és szöveges (jobbra) ábrázolással.

1.2. Szöveges ábrázolás

Az irányítatalan gráfoknál „$u - v_{u_1}, w_{u_1}; \cdots ; v_{u_k}, w_{u_k}$." azt jelenti, hogy $(u, v_{u_1}), \cdots , (u, v_{u_k})$ élei a gráfnak, sorban $w(u, v_{u_1}) = w_{u_1}, \cdots , w(u, v_{u_k}) = w_{u_k}$ súlyokkal. (Ld. az 1. ábrát!)

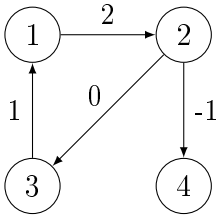
Az irányított gráfoknál pedig „$u \to v_{u_1}, w_{u_1}; \cdots ; v_{u_k}, w_{u_k}$." azt jelenti, hogy a gráfban az u csúcsból az $(u, v_{u_1}), \cdots , (u, v_{u_k})$ irányított élek indulnak ki, most is sorban $w(u, v_{u_1}) = w_{u_1}, \cdots , w(u, v_{u_k}) = w_{u_k}$ súlyokkal. (Ld. a 2. ábrát!)

1.3. Szomszédossági mátrixos (adjacency matrix), más néven csúcsmátrixos reprezentáció

A szomszédossági mátrixos, vagy más néven csúcsmátrixos ábrázolásnál a $G = (V, E)$ gráfot a $w : E \to \mathbb{R}$ súlyfüggvénnyel ($V = \{v_1, \cdots , v_n\}$) egy $A/1 : \mathbb{R}[n, n]$ mátrix reprezentálja, ahol $n = |V|$ a csúcsok száma, $1..n$ a csúcsok sorszámai, és tetszőleges $i, j \in 1..n$ csúcssorszámokra

$$A[i, j] = w(v_i, v_j) \iff (v_i, v_j) \in E$$
$$A[i, i] = 0$$
$$A[i, j] = \infty \iff (v_i, v_j) \notin E \wedge i \neq j$$

A 3. ábrán látható irányított gráfot például a mellette lévő szomszédossági mátrix reprezentálja.



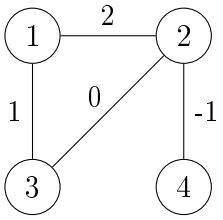
A	1	2	3	4
1	0	2	∞	∞
2	∞	0	0	-1
3	1	∞	0	∞
4	∞	∞	∞	0

3. ábra. Ugyanaz az élsúlyozott, irányított gráf grafikus (balra) és szomszédossági mátrixos (jobbra) ábrázolással.

A főátlóban mindig nullák vannak, mert csak egyszerű gráfokkal foglalkozunk (amelyekben nincsenek hurokélek), és tetszőleges csúcsból önmaga közvetlenül, nulla költségű úton érhető el.

Vegyük észre, hogy irányítatlan esetben a szomszédossági mátrixos reprezentáció mindig szimmetrikus, ui. $(v_i, v_j) \in E$ esetén $(v_j, v_i) = (v_i, v_j) \in E$.

Az 1. ábráról már ismerős irányítatlan gráf csúcsmátrixos ábrázolása a 4. ábrán látható.



A	1	2	3	4
1	0	2	1	∞
2	2	0	0	-1
3	1	0	0	∞
4	∞	-1	∞	0

4. ábra. Ugyanaz az élsúlyozott, irányítatlan gráf grafikus (balra) és szomszédossági mátrixos (jobbra) ábrázolással.

1.4. Szomszédossági listás (adjacency list) reprezentáció

A szomszédossági listás ábrázolás hasonlít a szöveges reprezentációhoz. A $G = (V, E)$ gráfot a $w : E \to \mathbb{R}$ súlyfüggvénnyel ($V = \{v_1, \cdots, v_n\}$) az $A : Edge*[n]$ pointertömb

Edge
$+v : \mathbb{N}$
$+w : \mathbb{R}$
$+next : Edge*$

segítségével ábrázoljuk, ahol a v attributumok szerepe ugyanaz, mint az élsúlyozatlan gráfoknál, w pedig a megfelelő él súlya. Az élsúlyozatlan gráfokhoz hasonlóan az élsúlyozott gráfoknál is: irányítatlan gráfok esetén minden élet kétszer ábrázolunk, irányított gráfok esetén csak egyszer. Élsúlyozott, irányított gráfra láthatunk példát az 5. ábrán.

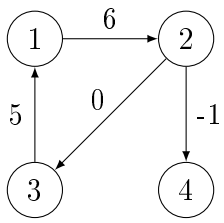
1.5. Élsúlyozott gráfábrázolások tárigénye

1.5.1. Szomszédossági mátrixok

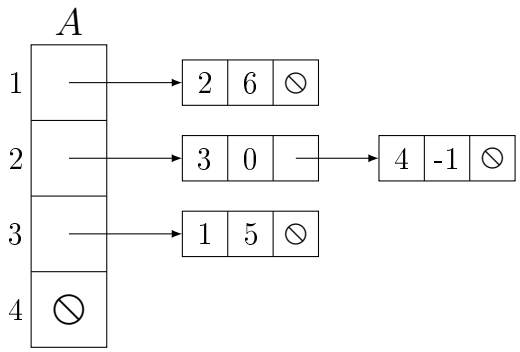
Tárigényük hasonlóan számolható, mint élsúlyozatlan esetben. Feltéve, hogy egy valós számot egy gépi szóban tárolunk, a szomszédossági mátrixos (más néven csúcsmátrixos) ábrázolás tárigénye alapesetben n^2 szó. Irányítatlan gráfoknál, csak az alsóháromszög mátrixot tárolva, $n * (n-1)/2$ szó. Mivel $n * (n-1)/2 \in \Theta(n^2)$, az aszimptotikus tárigény mindkét esetben $\Theta(n^2)$.

1.5.2. Szomszédossági listák

Mindegyik élben eggyel több mező van, mint az élsúlyozatlan esetben. Ez az aszimptotikus tárigényt nyilván nem befolyásolja.



$A[1] \to v = 2$; $A[1] \to w = 6$; $A[1] \to next = \oslash$
$A[2] \to v = 3$; $A[2] \to w = 0$; $A[2] \to next \to v = 4$
; $A[2] \to next \to w = -1$; $A[2] \to next \to next = \oslash$
$A[3] \to v = 1$; $A[3] \to w = 5$; $A[3] \to next = \oslash$
$A[4] = \oslash$

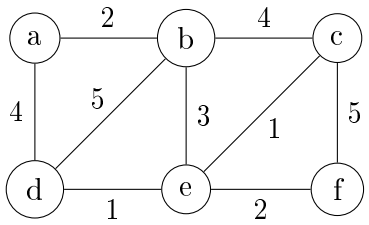


5. ábra. Ugyanaz az élsúlyozott, irányított gráf grafikus (balra) és szomszédossági listás (jobbra) ábrázolással.

1.6. Élsúlyozott gráfok absztrakt osztálya

$\mathcal{G}_w$
$+ V : \mathcal{V}\{\}$ $+ E : \mathcal{E}\{\}$ // $E \subseteq V \times V \setminus \{(u,u) : u \in V\}$ $+ w : E \to \mathbb{R}$ // weights of edges

2. Minimális feszítőfák ([2] 23; [4])



a – b, 2 ; d, 4.
b – c, 4 ; d, 5 ; e, 3.
c – e, 1 ; f, 5.
d – e, 1.
e – f, 2.

6. ábra. Összefüggő, élsúlyozott, irányítatlan gráf. A csúcsok pl. városok, az élek lehetséges összeköttetések, a megépítésük költségeivel. A lehető legkisebb építési költséggel szeretnénk elérni, hogy tetszőleges városból bármelyik másikba el lehessen jutni.

A 6. ábrán megfogalmazott (és még sok más hasonló) feladat megoldása céljából definiáljuk a *minimális feszítőfa* (MST = Minimum Spanning Tree) fogalmát. Ebben a fejezetben tetszőleges összefüggő, irányítatlan, élsúlyozott gráf *minimális feszítőfáját* keressük. (Az élsúlyok negatívak is lehetnek.)

2.1. Definíció. *A $G = (V, E)$ irányítatlan, összefüggő gráf* feszítőfája *a $T = (V, F)$ gráf, ha T irányítatlan, összefüggő fa, és $F \subseteq E$. Amennyiben G élsúlyozott a $w : E \to \mathbb{R}$ súlyfüggvénnyel, akkor a T fa súlya az élei súlyainak összege:*

$$w(T) = \sum_{e \in F} w(e)$$

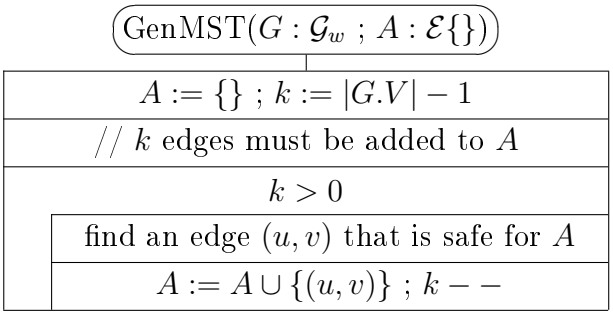
2.2. Definíció. *A G irányítatlan, összefüggő, élsúlyozott gráf* minimális feszítőfája *T, ha T a G feszítőfája, és G bármely T' feszítőfájára $w(T) \leq w(T')$.*

Az ezután következő kidolgozatlan részek fő forrásai:

- az ezen jegyzet bevezetésben hivatkozott hallgatói jegyzet [4],
- az ugyanitt azonosított Új Algoritmusok c. könyv [2],
- a Rónyai könyv [5],
- és mindenekelőtt az Ön jegyzetei, amelyeket az előadásaimon készített.

2.1. Egy általános algoritmus

Az alábbi általános algoritmus az $A = \{\}$ üres élhalmazból indul, és ezt úgy bővíti újabb és újabb élekkel, hogy A végig a G összefüggő, irányítatlan, élsúlyozott gráf valamelyik minimális feszítőfájának a részhalmaza marad: Éppen az így választott éleket nevezzük az *A élhalmazra nézve biztonságosnak* (*safe for A*). Amikor az élek száma eléri a $|G.V|-1$ értéket, az A szükségszerűen feszítőfa, és így minimális feszítőfa is lesz.



2.3. Definíció. *Tegyük fel, hogy $G = (V, E)$ élsúlyozott, irányítatlan, összefüggő gráf, és $A \subseteq$ a G valamelyik minimális feszítőfája élhalmazának! Ekkor az $(u, v) \in E$ él* biztonságosan hozzávehető *az A élhalmazhoz (*safe for A*), ha $(u, v) \notin A$ és $A \cup \{(u, v)\} \subseteq$ a G valamelyik (az előzővel nem okvetlenül egyező) minimális feszítőfája élhalmazának.*

2.4. Következmény. *Tegyük fel, hogy $G = (V, E)$ élsúlyozott, irányítatlan, összefüggő gráf! Ha egy kezdetben üres A élhalmazt újabb és újabb biztonságosan hozzávehető éllel bővítünk, akkor $|G.V|-1$ bővítés után éppen a G egyik minimális feszítőfáját kapjuk meg.*

A kérdés most az, hogyan tudunk mindig biztonságos élet választani az A élhalmazhoz. Ehhez lesz szükségünk az alábbi fogalmakra és tételre.

2.5. Definíció. *Ha $G = (V, E)$ gráf és $\{\} \subsetneq S \subsetneq V$, akkor a G gráfon $(S, V \setminus S)$ egy* vágás.

2.6. Definíció. *$G = (V, E)$ gráfon az $(u, v) \in E$ él* keresztezi *az $(S, V \setminus S)$* vágást, *ha $(u \in S \wedge v \in V \setminus S) \vee (u \in V \setminus S \wedge v \in S)$.*

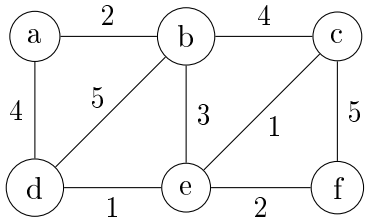
2.7. Definíció. *$G = (V, E)$ élsúlyozott gráfon az $(u, v) \in E$* könnyű él *az $(S, V \setminus S)$* vágásban, *ha (u, v) keresztezi a vágást, és $\forall (p, q)$, a vágást keresztező élre $w(u, v) \leq w(p, q)$.*

2.8. Definíció. *A $G = (V, E)$ gráfban az $A \subseteq E$ élhalmazt* elkerüli *az $(S, V \setminus S)$* vágás*, ha az A egyetlen éle sem keresztezi a vágást.*

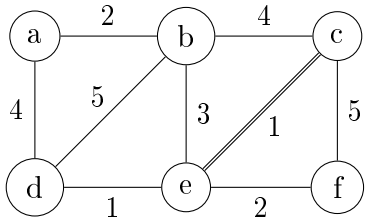
2.9. Tétel. *Ha a $G = (V, E)$ élsúlyozott gráfon (1) az $(S, V \setminus S)$ vágás elkerüli az $A \subseteq E$ élhalmazt, (2) A részhalmaza G valamelyik minimális feszítőfája élhalmazának, és (3) az $(u, v) \in E$* könnyű él *az $(S, V \setminus S)$* vágásban, *$\Longrightarrow$ az (u, v) él biztonságosan hozzávehető az A élhalmazhoz.*

Az előbbi tétel segítségével már módszeresen tudunk minimális feszítőfát építeni. Jejöljük a gráfban dupla vonallal az A élhalmaz elemeit!

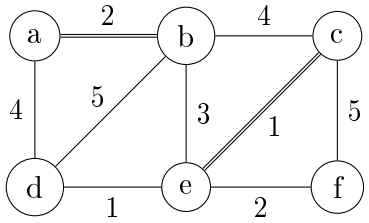
Az alábbi példában minden lépésben választunk egy A-t elkerülő vágást, majd ebben egy könnyű élt, amit hozzáveszünk A-hoz. A hozzávétel eredménye a következő lépés kiinduló pontja.



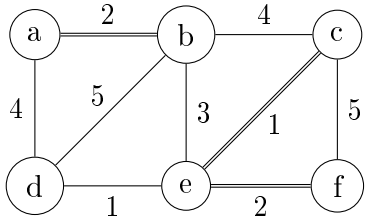
Kezdetben $A = \{\}$, így bármelyik vágás jó. Válasszuk pl. az $(\{a, b, c\}, \{d, e, f\})$ vágást! Ebben (c, e) a könnyű, tehát biztonságos él. Vegyük hozzá A-hoz!



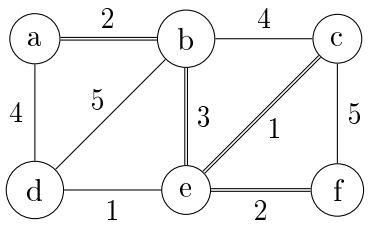
Most $A = \{(c, e)\}$. Ezt elkerüli pl. az $(\{a, f\}, \{b, c, d, e\})$ vágás, amiben (a, b) és (e, f) a könnyű, tehát biztonságos élek. Válasszuk pl. (a, b)-t!



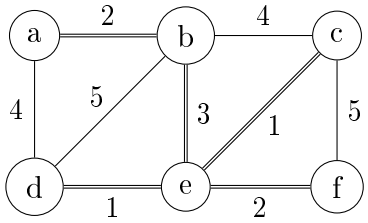
Most $A = \{(a, b), (c, e)\}$. Ezt elkerüli pl. az $(\{a, b, c, d, e\}, \{f\})$ vágás, amiben (e, f) a könnyű, tehát biztonságos él.



Most $A = \{(a, b), (c, e), (e, f)\}$. Ezt elkerüli pl. az $(\{a, b\}, \{c, d, e, f\})$ vágás, amiben (b, e) a könnyű, tehát biztonságos él.



$A = \{(a,b),(b,e),(c,e),(e,f)\}$. Ezt egyedül az $(\{a,b,c,e,f\},\{d\})$ vágás kerüli el, amiben (d,e) a könnyű, tehát biztonságos él.



$A = \{(a,b),(b,e),(c,e),(d,e),(e,f)\}$. $|A| = 5 = |G.V|-1$, tehát A a minimális feszítőfa (MST) élhalmaza.

A fenti módszer még nem tekinthető szigorú értelemben vett algoritmusnak, mert nem adtuk meg, hogyan válasszunk ki az A halmazt elkerülő vágást, és abban könnyű élt. A következő alfejezetekben ismertetendő Kruskal és Prim algoritmusok éppen ezt teszik, mégpedig

$$O(m * \lg n)\,,$$

tehát nagyon jó maximális műveletigénnyel. (Mint mindig, n a gráf csúcsainak, m pedig az éleinek száma.) Érdekes módon egyik sem adja meg az A-t elkerülő vágást expliciten, a vágásban (az egyik) könnyű élt viszont igen.

2.10. Feladat. *Adjon meg olyan 3 csúcsú, összefüggő irányítatlan, élsúlyozott gráfot, amelynek pontosan 2 minimális feszítőfája van. Hány feszítőfája van összesen? Adjon olyan 3 csúcsú gráfot, amelynek 3 minimális feszítőfája van! Létezik olyan 3 csúcsú gráf, aminek háromnál több feszítőfája van? (Gráf alatt, mint mindig, most is egyszerű gráfot értünk.)*

2.2. Kruskal algoritmusa

Kruskal algoritmusa a $G = (V, E)$ gráf éleit a súlyuk (hosszuk) szerint monoton növekvően veszi sorba. Azokat az éleket eldobja, amelyek az A bizonyos éleivel együtt kört képeznének. A többit hozzáveszi A-hoz. Az explicit körkeresés azonban nem lenne hatékony, mert minden egyes e élre bejárást kellene indítani a $(V, A \cup \{e\})$ gráfon. Ehelyett a következő definíción és invariánson alapuló megoldáshoz folyamodunk.

2.11. Definíció. *A $G = (V, E)$ gráf* feszítő erdeje *a (V, A) gráf, ha egymástól diszjunkt fákból, mint komponensekből áll, és $A \subseteq E$. (Két fa egymástól diszjunkt, ha nincs közös csúcsuk [és így közös élük sem].)*

A **Kruskal algoritmus invariánsa,** hogy (V, A) *a $G = (V, E)$ összefüggő, irányítatlan, élsúlyozott gráf feszítő erdeje, és A részhalmaza a G valamelyik minimális feszítőfája élhalmazának.*

Kruskal algoritmusa: $A = \{\}$-val indulunk, ami azt jelenti, hogy a kezdeti feszítő erdő fái a $G = (V, E)$ gráf egycsúcsú fái. A G gráf éleit a súlyuk (hosszuk) szerint monoton növekvően vesszük sorba, és tetszőleges élet pontosan akkor veszünk hozzá A-hoz, ha a (V, A) erdő két fáját köti össze (azaz nem egy fán belül fut, és így zár be egyetlen kört sem). Ezért minden egyes él hozzávételével eggyel csökken az erdő fáinak száma, de továbbra is feszítő erdőt alkotnak. Mivel G összefüggő, bármelyik két fa között van út, így előbb-utóbb összekapcsolódnak és már csak egy T fából áll az erdő, T feszítőfa. A fenti invariáns miatt T az élhalmaza részhalmaza valamelyik M minimális feszítőfa élhalmazának. A G minden feszítőfájának $|V|-1$ éle van, így a T és M élhalmaza megegyezik. Mindkettő csúcshalmaza V, tehát $T = M$, azaz T minimális feszítőfa.

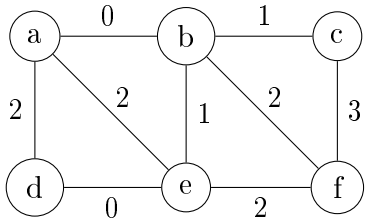
Kérdés még, hogy a fenti invariáns miért igaz. Kezdetben a (V, A) úgy feszítő erdő, hogy $A = \{\}$, tehát A részhalmaza a G bármelyik minimális feszítőfája élhalmazának, azaz az invariáns igaz.

Tekintsünk most az algoritmus futása során egy olyan pillanatot, amikor az invariáns még igaz, és egy újabb e élet készülünk megvizsgálni. Belátjuk, hogy az invariáns az e él feldolgozása után is igaz marad.

Ha e a jelenlegi feszítő erdő valamelyik fáján belül fut, eldobjuk, a feszítő erdő nem változik és az invariáns igaz marad.

Ha e a jelenlegi feszítő erdő két fáját köti össze, legyen S az egyik fa csúcshalmaza, és tekintsük az $(S, V \setminus S)$ vágást, ami nyilván elkerüli A-t. Ekkor e könnyű él a vágásban [mert a G gráf éleit a súlyuk (hosszuk) szerint monoton növekvően vesszük sorba, tehát az aktuális élnél kisebb súlyú éleket már

korábban feldolgoztuk, így ezek vagy benne vannak A-ban, vagy nincsenek benne A-ban, de a (V, A) feszítő erdő egyik fájának két csúcsát kötik össze, ezért eldobtuk őket]. Teljesülnek tehát a 2.9. tétel feltételei. Ennélfogva e biztonságosan hozzávehető A-hoz, és hozzá is vesszük. A fentiek szerint tehát az invariáns ebben az esetben is igaz marad.



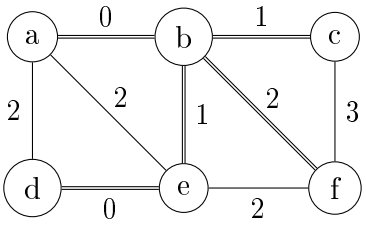
a – b, 0 ; d, 2 ; e, 2.
b – c, 1 ; e, 1 ; f, 2.
c – f, 3.
d – e, 0.
e – f, 2.

7. ábra. Összefüggő, élsúlyozott, irányítatlan gráf

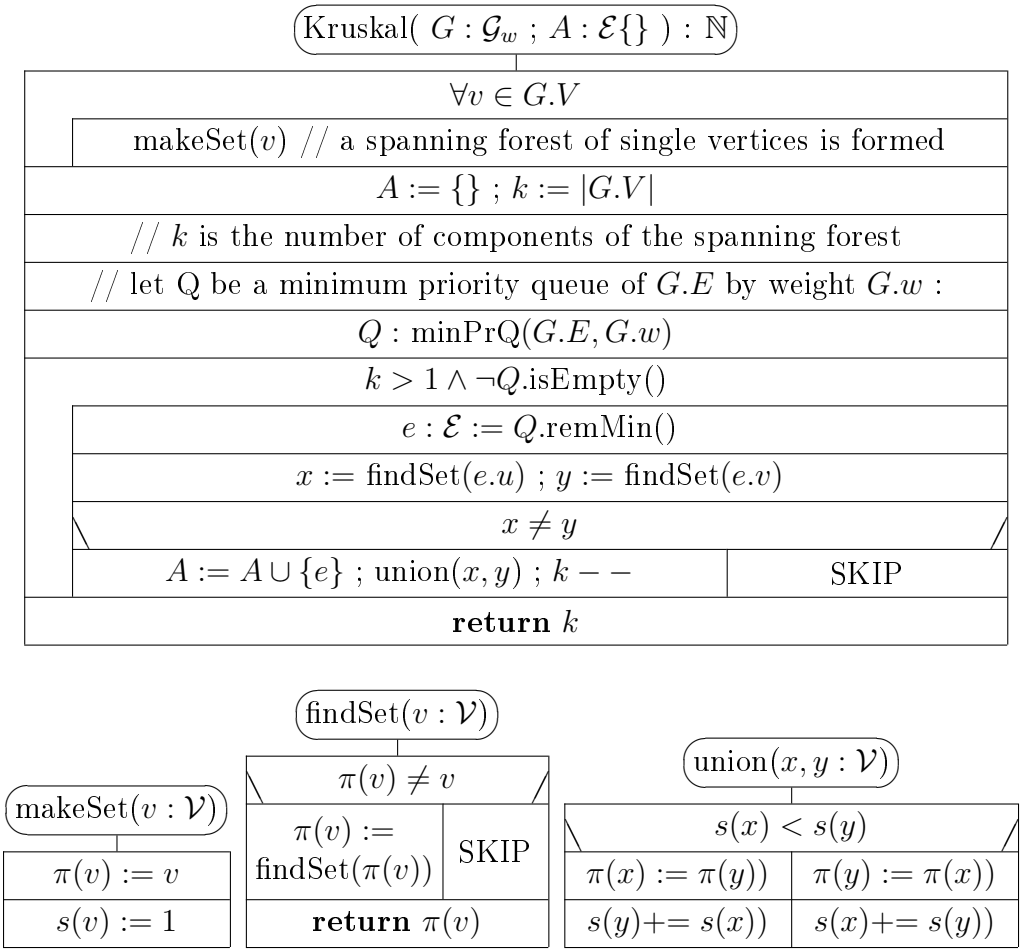
2.12. Példa. *Szemléltessük Kruskal algoritmusát a 7. ábrán látható gráfon!*

Komponensek	él	él súlya	biztonságos?
a, b, c, d, e, f	(a,b)	0	+
ab, c, d, e, f	(d,e)	0	+
ab, c, de, f	(b,c)	1	+
abc, de, f	(b,e)	1	+
abcde, f	(a,d)	2	-
abcde, f	(a,e)	2	-
abcde, f	(b,f)	2	+
abcdef	-	-	-

;



8. ábra. A 7. ábráról ismerős gráfon a minimális feszítőfa éleit dupla vonallal jelöltük.



Ez az algoritmus kivételesen „ellenőrzi”, hogy G összefüggő-e. Ha ugyanis összefüggő, $k = 1$ értékkel tér vissza. Ha nem, $k > 1$ lesz.

2.13. Feladat. *Hogyan tudná kifinomultabban értelmezni azt az esetet, ha a Kruskal algoritmus $k > 1$ értékkel tér vissza? Mit jelent a k értéke általában? Tudna-e értelmet tulajdonítani a Kruskal algoritmus által kiszámolt A élhalmaznak, ha végül $k > 1$ értéket ad vissza?*

2.14. Feladat. *Implementálja a Kruskal algoritmust az elméleti $O(m * \lg n)$ maximális műveletigény megtartásával!*

2.3. Prim algoritmusa

Kijelölünk a $G = (V, E)$ összefüggő, élsúlyozott, irányítatlan gráfban egy tetszőleges $r \in V$ csúcsot. A $T = (\{r\}, \{\})$, egyetlen csúcsból álló fából kiindulva építünk fel egy minimális (V, F) feszítőfát: Minden lépésben a $T = (N, A)$ fához egy újabb biztonságos élet és hozzá tartozó csúcsot adunk.

Ez azt jelenti, hogy a $T = (N, A)$ fa végig ennek a minimális feszítőfának a része marad, azaz végig igaz az $N \subseteq V \wedge A \subseteq F$ invariáns. Ehhez mindegyik lépésben egy könnyű élet választunk ki az $(N, V \setminus N)$ vágásban. (Ld. a 2.9. tételt!)

A megfelelő könnyű él hatékony meghatározása céljából egy Q min prioritásos sorban tartjuk nyilván a $V \setminus N$ csúcshalmazt. Mindegyik $u \in V \setminus N$ csúcshoz tartozik egy $c(u)$ és egy $p(u)$ címke. Ha van él az u csúcs és a T fa N csúcshalmaza között, azaz az $(N, V \setminus N)$ vágásban, akkor a $(p(u), u)$ a gráf egyik éle a vágásban, $c(u) = w(p(u), u)$, és a minden olyan x csúcsra, amelyre (x, u) vágásbeli él, $w(x, u) \geq w(p(u), u)$.

Ez azt jelenti, hogy $(p(u), u)$ minimális súlyú él azok között, amelyek az u csúcsot a T fához kapcsolják. Ha nincs él a u csúcs és a T fa között, akkor $p(u) = \oslash \wedge c(u) = \infty$.

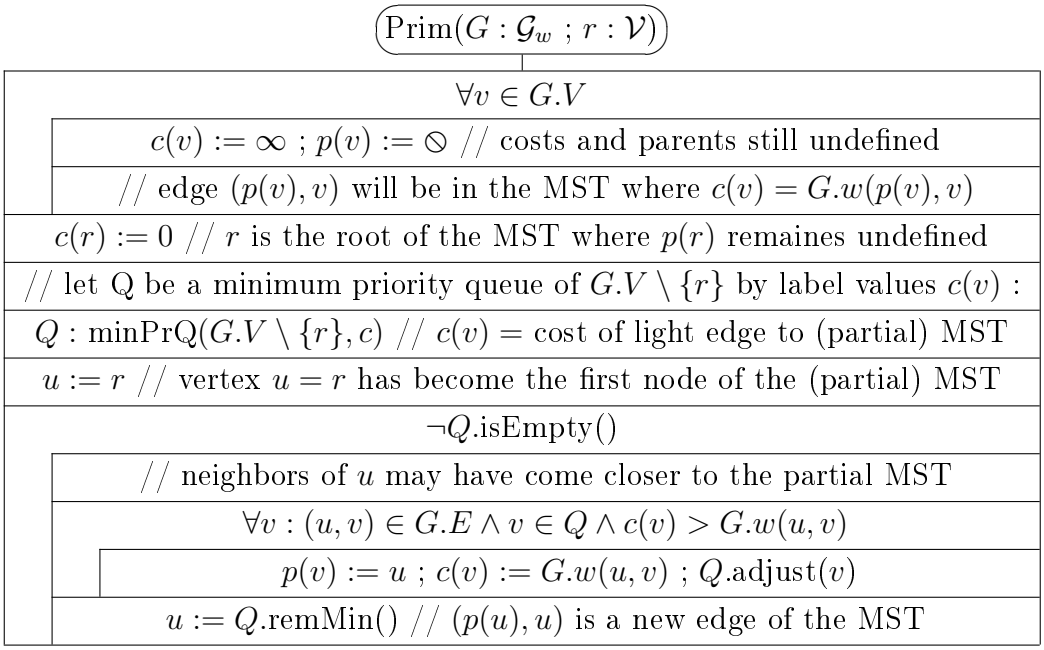
A Q min prioritásos sorban a csúcsokat a $c(u)$ értékeik szerint tartjuk nyilván. Az $u := Q$.remMin() utasítás tehát egy olyan u csúcsot vesz ki Q-ból, amire a $(p(u), u)$ könnyű él az $(N, V \setminus N)$ vágásban. Ezt az élt így a 2.9. tétel szerint biztonságosan adjuk hozzá a T fához, azaz T továbbra is kiegészíthető minimális feszítőfává, ha még nem az.

Így az eredetileg egyetlen csúcsból álló T fa, $n-1$ db ilyen $(p(u), u)$ él hozzáadásával MST (minimális feszítőfa) lesz.

Amikor u bekerül a T fába, a Q-beli v szomszédai közelebb kerülhetnek a fához, így ezeket meg kell vizsgálni, és ha némelyikre $w(u, v) < c(v)$, akkor a $c(v) := w(u, v)$ és a $p(v) := u$ utasításokkal aktualizálni kell a v csúcs címkéit.

Ilyen módon a $c(v)$ érték csökkenése miatt a Q helyreállítása is szükségessé válhat. Ha például Q reprezentációja egy minimum kupac, a v csúcsot lehet, hogy meg kell cserélni a szülőjével, esetleg újra és újra, mígnem a szülőjének c értéke $\leq c(v)$ lesz, vagy v fölér kupac gyökerébe.

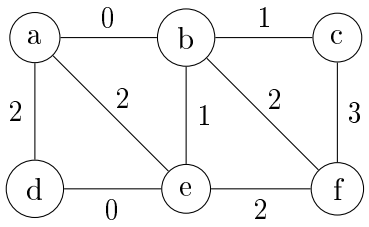
Az alábbi algoritmusban valaki hiányolhatja a T fa explicit reprezentációját. Világos, hogy implicite $N = V \setminus Q$, T éleit pedig az $N \setminus \{r\}$-beli x csúcsok és $p(x)$ címkéik segítségével $(p(x), x)$ alakban definiálhatjuk.



2.15. Feladat. *Implementálja a Prim algoritmust szomszédossági mátrixos gráfábrázolás esetén! A prioritásos sort rendezetlen tömbbel reprezentálja! Mekkora lesz a műveletigény? Lehet-e az aszimptotikus műveletigényen javítani a prioritásos sor kifinomultabb megvalósításával?*

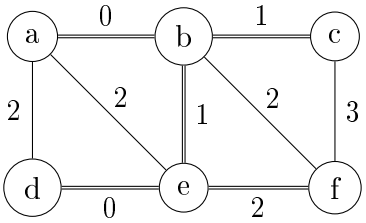
2.16. Feladat. *Implementálja a Prim algoritmust szomszédossági listás gráfábrázolás esetén! Ügyeljen arra, hogy a prioritásos sor megfelelő megvalósításával biztosítani kell az elméleti $O(m * \lg n)$ műveletigényt!*

Az alábbi, már ismerős gráfon szemléltetjük a Prim algoritmus működését, a **d** csúcsból indítva.



a – b, 0 ; d, 2 ; e, 2.
b – c, 1 ; e, 1 ; f, 2.
c – f, 3.
d – e, 0.
e – f, 2.

c értékek Q-ban						minimális feszítő-fába:	p címkék változásai					
a	b	c	d	e	f		a	b	c	d	e	f
∞	∞	∞	0	∞	∞		$\oslash$	$\oslash$	$\oslash$	$\oslash$	$\oslash$	$\oslash$
2	∞	∞		0	∞	d	d				d	
2	1	∞			2	e		e				e
0		1			2	b	b		b			
		1			2	a						
					2	c						
						f						
0	1	1	0	0	2	eredmény	b	e	b	$\oslash$	d	e



A Kruskal algoritmus eredményéhez képest most a gráf másik minimális feszítőfáját kaptuk meg.

3. Legrövidebb utak egy forrásból([2] 24; [4])

Feladat: A $G : \mathcal{G}_w$ gráf tetszőlegesen rögzített s start csúcsából (source vertex) legrövidebb, azaz optimális utat keresünk mindegyik, az s-ből G-ben elérhető csúcsba.

A most következő algoritmusokban az irányítatlan gráfokat olyan irányított gráfoknak tekintjük, ahol a gráf tetszőleges (u, v) élével együtt (v, u) is éle a gráfnak, és $w(u, v) = w(v, u)$.

A feladat pontosan akkor oldható meg, ha G-ben nem létezik s-ből elérhető negatív kör, azaz olyan kör, amely mentén az élsúlyok összege negatív.

Mivel az irányítatlan gráfokat ebben a fejezetben speciális irányított gráfoknak tekintjük, ez a feltétel irányítatlan gráfok esetében azt jelenti, hogy a feladatot akkor tudjuk megoldani, ha nincs a gráfban s-ből elérhető negatív él. (Ha pl. az irányítatlan gráfban (u, v) s-ből elérhető negatív él, akkor a fenti egyszerűsítés miatt $\langle u, v, u\rangle$ egy s-ből elérhető negatív kör.)

Amennyiben a feladat megoldható, mindegyik $v \in G.V\backslash\{s\}$ csúcsra két lehetőség van:
- Ha létezik út s-ből v-be, $d(v)$-ben az optimális út hosszát szeretnénk megkapni, $\pi(v)$-ben pedig egy ilyen optimális úton a v csúcs közvetlen megelőzőjét, azaz szülőjét.
- Ha nem létezik út s-ből v-be, a $d(v) = \infty$ és $\pi(v) = \oslash$ értékeket szeretnénk kapni.

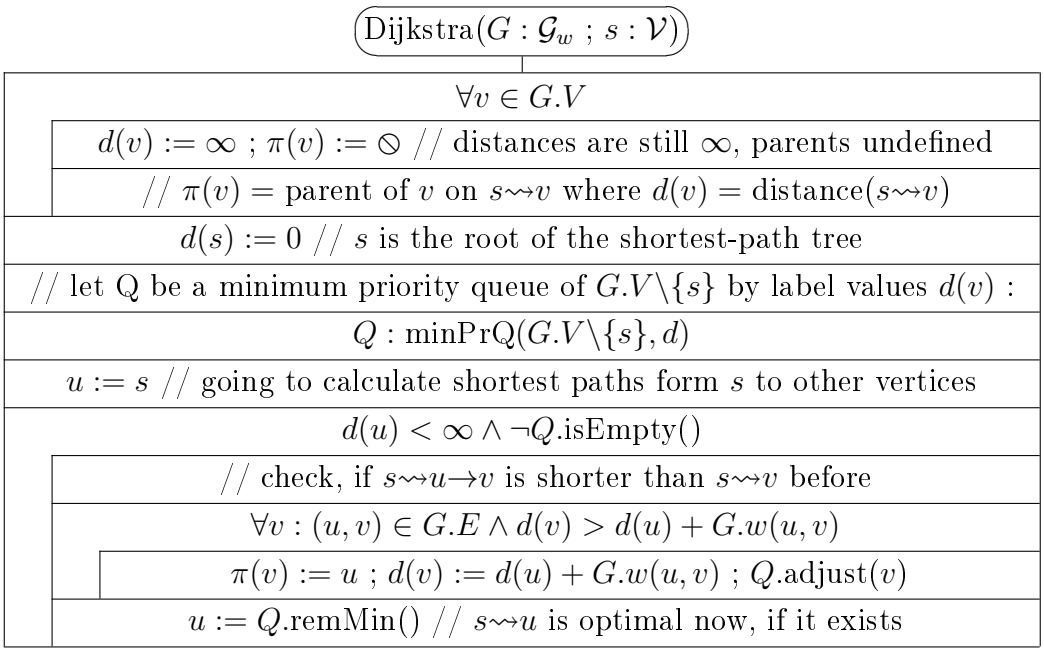
Az s csúcsra a helyes eredmény $d(s) = 0$ és $\pi(s) = \oslash$, mivel az optimális út csak az s csúcsból áll.

A legrövidebb utakat kereső algoritmusok futása során az optimális utakat fokozatosan közelítjük. Jelölje $s \rightsquigarrow v$ az adott pillanatig kiszámolt s-ből v-be vezető legrövidebb utat! Az algoritmusok invariánsa, hogy $d(v)$ ennek hossza, $\pi(v)$ pedig ezen az úton a v csúcs közvetlen megelőzője. Ha még nem számoltunk ki $s \rightsquigarrow v$ utat, akkor végtelen hosszúnak tekintjük, azaz $d(v) = \infty$ és $\pi(v) = \oslash$.

3.1. Dijkstra algoritmus

Előfeltétel: A $G : \mathcal{G}_w$ gráfban $\forall (u, v) \in G.E$-re $G.w(u, v) \geq 0$, azaz a gráf mindegyik élének élsúlya nemnegatív.

Ebben az esetben nem lehet a gráfban negatív kör, tehát a *legrövidebb utak egy forrásból* feladat megoldható.



$$MT(n, m) \in O((n + m) * \lg n)$$

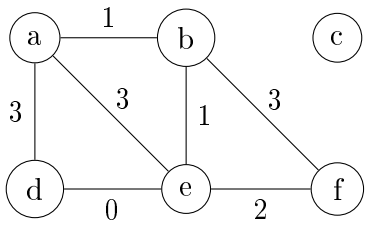
$$mT(n, m) \in \Theta(n)$$

3.1. Feladat. *Mikor lesz a legkisebb az algoritmus futási ideje? Mikor lesz a legnagyobb? Miért?*

3.2. Feladat. *Implementálja a Dijkstra algoritmust szomszédossági mátrixos gráfábrázolás esetén! A prioritásos sort rendezetlen tömbbel reprezentálja! Mekkora lesz a műveletigény? Lehet-e az aszimptotikus műveletigényen javítani a prioritásos sor kifinomultabb megvalósításával?*

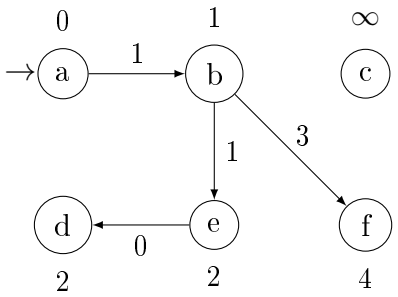
3.3. Feladat. *Implementálja a Dijkstra algoritmust szomszédossági listás gráfábrázolás esetén! Ügyeljen arra, hogy a prioritásos sor megfelelő megvalósításával biztosítani lehet az elméleti $O((m + n) * \lg n)$ műveletigényt!*

Az alábbi gráfon szemléltetjük a Dijkstra algoritmus működését, az **a** csúcsból indítva.



a – b, 1 ; d, 3 ; e, 3.
b – e, 1 ; f, 3.
c.
d – e, 0.
e – f, 2.

d értékek Q-ban						kiter-	π címkék változásai					
a	b	c	d	e	f	jesztett	a	b	c	d	e	f
0	∞	∞	∞	∞	∞	csúcs : d	$\oslash$	$\oslash$	$\oslash$	$\oslash$	$\oslash$	$\oslash$
	1	∞	3	3	∞	a : 0		a		a	a	
		∞	3	2	4	b : 1					b	b
		∞	2		4	e : 2				e		
		∞			4	d : 2						
		∞				f : 4						
0	1	∞	2	2	4	eredmény	$\oslash$	a	$\oslash$	e	b	b



A legrövidebb utak fája $s =$ a esetén. Az s-ből elérhetetlen csúcsot vagy csúcsokat is feltüntetjük, ∞ d értékkel.

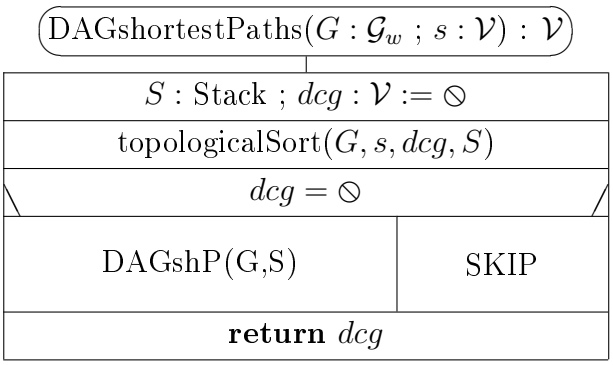
3.2. DAG legrövidebb utak egy forrásból

Előfeltétel: A $G : \mathcal{G}_w$ gráf irányított, és a gráfban nincs s-ből elérhető irányított kör.

Ebben az esetben nincs a gráfban s-ből elérhető negatív kör sem, így a *legrövidebb utak egy forrásból* feladat megoldható.

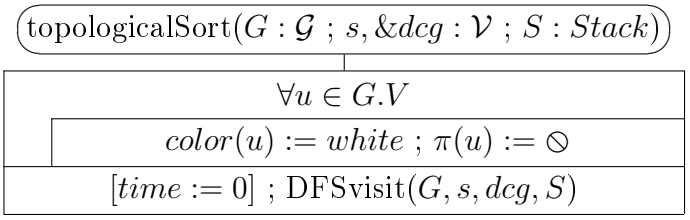
Ez az algoritmus ellenőrzi az előfeltételét. Ha teljesül, a DAGshortestPaths() függvény $\oslash$ értékkel tér vissza. Különben megtalál egy irányított

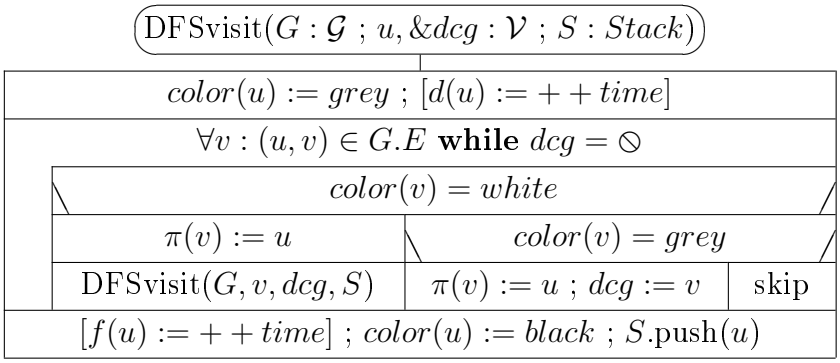
kört, aminek egyik csúcsával tér vissza. Ebből indulva a π címkék mentén a kör fordított irányban bejárható.

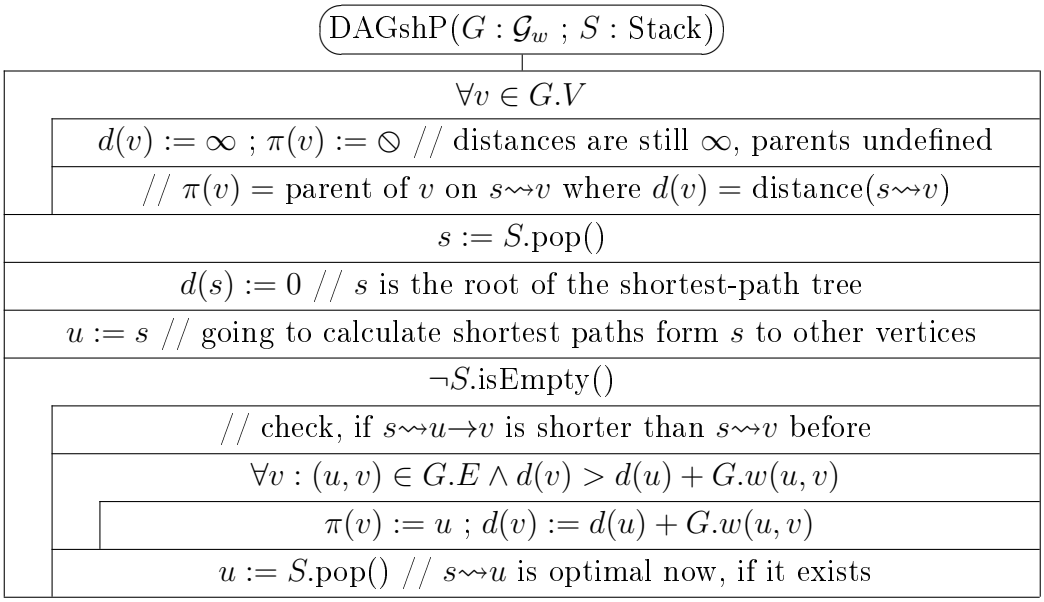


A topologikus rendezés csak az s-ből elérhető részgráfot próbálja topologikusan rendezni.

A *time* nevű változóra – amit az egyszerűség kedvéért globálisnak képzelünk – és a hozzá kapcsolódó utasításokra csak a topologikus rendezés szemléltetésének megkönnyítése végett van szükségünk. Ezek az implmentációkból elhagyhatók, ezért a vonatkozó utasításokat szögletes zárójelbe tettük.







$$MT(n, m) \in \Theta(n + m)$$

$$mT(n, m) \in \Theta(n)$$

3.4. Feladat. *Mikor lesz a legkisebb az algoritmus futási ideje? Mikor lesz a legnagyobb? Miért?*

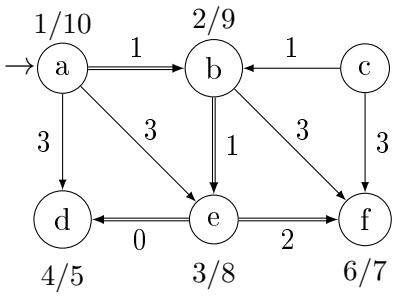
3.5. Feladat. *Implementálja a* DAG legrövidebb utak egy forrásból *algoritmust szomszédossági mátrixos gráfábrázolás esetén!*

Mekkora lesz a műveletigény? Tartható-e az elméleti maximális, illetve minimális műveletigény ezzel az ábrázolással?

3.6. Feladat. *Implementálja a* DAG legrövidebb utak egy forrásból *algoritmust szomszédossági listás gráfábrázolás esetén!*

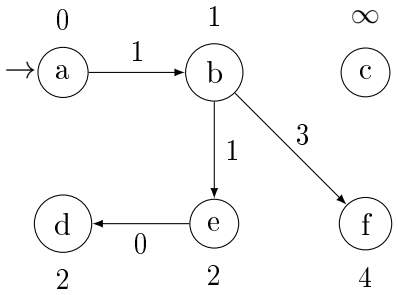
Mekkora lesz a műveletigény? Tartható-e az elméleti maximális, illetve minimális műveletigény ezzel az ábrázolással?

Az alább látható gráfon szemléltetjük a *DAG legrövidebb utak egy forrásból* algoritmust, ahol $s = \text{a}$. Először topologikusan rendezzük az s-ből elérhető részgráfot. (A dupla nyilak a mélységi fa *fa-éleit* jelölik.) Ezután – a szokásos inicializálásokat követően – a csúcsokat a topologikus sorrendjüknek (S) megfelelően terjesztjük ki.



a → b, 1 ; d, 3 ; e, 3.
b → e, 1 ; f, 3.
c → b, 1 ; f, 3.
e → d, 0 ; f, 2.
$S = \langle$ a, b, e, f, d $\rangle$

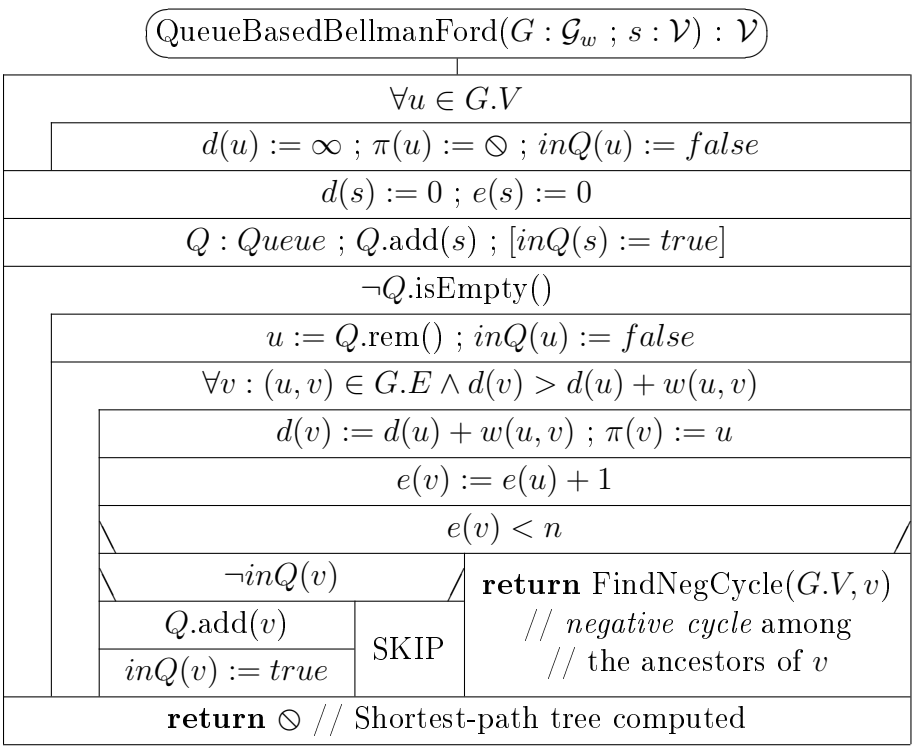
d értékek változásai						kiter-	π címkék változásai					
a	b	c	d	e	f	jesztett	a	b	c	d	e	f
0	∞	∞	∞	∞	∞	csúcs : d	$\oslash$	$\oslash$	$\oslash$	$\oslash$	$\oslash$	$\oslash$
	1		3	3		a : 0		a		a	a	
				2	4	b : 1					b	b
			2			e : 2				e		
						f : 4						
						d : 2						
0	1	∞	2	2	4	eredmény	$\oslash$	a	$\oslash$	e	b	b

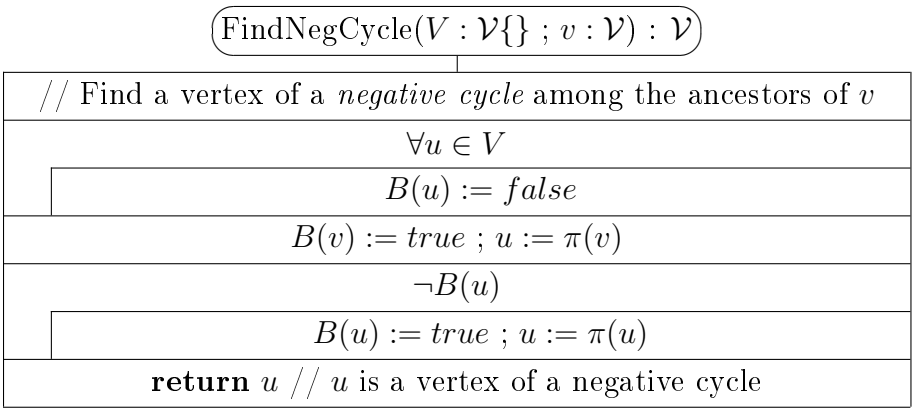


A legrövidebb utak fája $s =$ a esetén. Az s-ből elérhetetlen csúcsot vagy csúcsokat is feltüntetjük, ∞ d értékkel.

3.3. Sor-alapú Bellman-Ford algoritmus

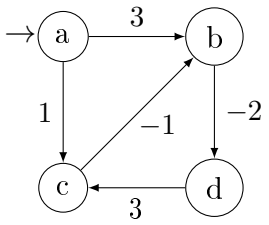
$e(v)$: ennyi élt tartalmaz $s \rightsquigarrow v$. Ha biztosan nincs s-ből elérhető negatív kör, akkor az e címkékkel és a negatív körrel kapcsolatos részek elhagyhatók.





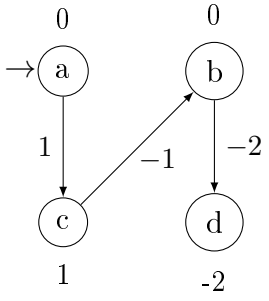
$$MT(n, m) \in O(n * m)$$

Az elméleti műveletigény becslés [6] meglehetősen rossz hatékonyságot sugall. Gyakorlati tesztek pozitítív élsúlyú, véletlenszerűen generált, nagyméretű, s-ből összefüggő ritka gráfokra[1] ($m \in O(n)$) statisztikusan $\Theta(n)$ átlagos műveletigényt adtak, szomszédossági listás gráfábrázolás esetén. Ez viszont aszimptotikusan az elméleti minimummal egyezik. (A hálózatok jelentős része nagyméretű, ritka gráffal modellezhető. Nem véletlen, hogy hálózatokon való útkeresésnél alkalmazzák is ezt a viszonylag egyszerű, de általános algoritmust.)



a → b, 3 ; c, 1.
b → d, −2.
c → b, −1.
d → c, 3.

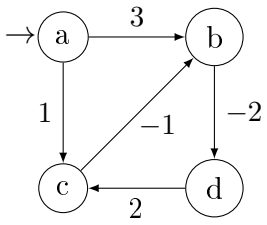
$d; e$ változásai				kiter-	Q :	π változásai			
a	b	c	d	jesztett	Queue	a	b	c	d
$0; 0$	∞	∞	∞	csúcs: $d; e$	$\langle$ a $\rangle$	⊘	⊘	⊘	⊘
	$3; 1$	$1; 1$		a: $0; 0$	$\langle$ b, c $\rangle$		a	a	
			$1; 2$	b: $3; 1$	$\langle$ c, d $\rangle$				b
	$0; 2$			c: $1; 1$	$\langle$ d, b $\rangle$		c		
				d: $1; 2$	$\langle$ b $\rangle$				
			$-2; 3$	b: $0; 2$	$\langle$ d $\rangle$				b
				d: $-2; 3$	$\langle$ $\rangle$				
0	0	1	-2	végső d és π értékek		⊘	c	a	b



A legrövidebb utak fája s = a esetén. A csúcsoknál csak a d értékeket tüntettük föl.

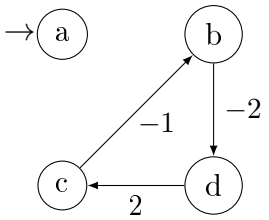
[1]Egy gráf egy *adott csúcsból összefüggő*, ha ebből a csúcsból a gráf mindegyik csúcsa elérhető. A *ritka gráfokra* az $m \leq k * n$, ahol k előre adott konstans, gyakran $k \leq 4$.

Negatív kör esetének szemléltetése:



a → b, 3 ; c, 1.
b → d, −2.
c → b, −1.
d → c, 2.

$d; e$ változásai				kiter-	Q :	π változásai			
a	b	c	d	jesztett	Queue	a	b	c	d
$0; 0$	∞	∞	∞	csúcs: $d; e$	⟨ a ⟩	⊘	⊘	⊘	⊘
	$3; 1$	$1; 1$		a: $0; 0$	⟨ b, c ⟩		a	a	
			$1; 2$	b: $3; 1$	⟨ c, d ⟩				b
	$0; 2$			c: $1; 1$	⟨ d, b ⟩		c		
				d: $1; 2$	⟨ b ⟩				
			$-2; 3$	b: $0; 2$	⟨ d ⟩				b
		$0; 4$		d: $-2; 3$	⟨ c ⟩			d	

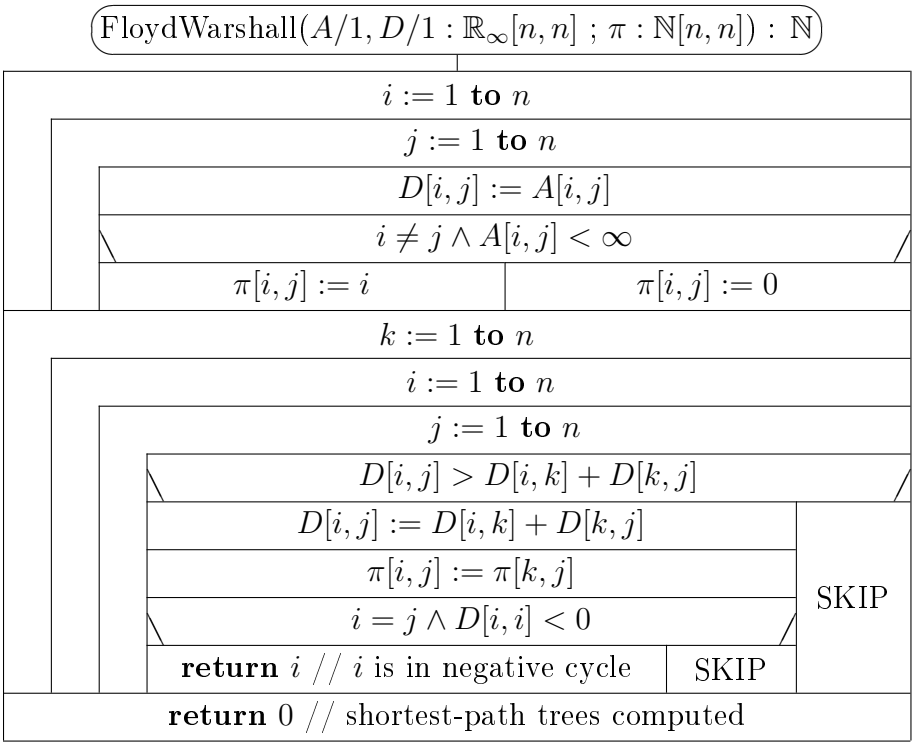


Itt megállunk, mert $e(\mathrm{c}) = 4 = n$, tehát negatív kör található a „c” csúcs ősei közt. A π értékek szerint visszafelé haladva megtalálhatjuk a negatív kört.

4. Legrövidebb utak minden csúcspárra ([2] 25; [4])

4.1. Floyd-Warshall algoritmus

4.1. Jelölés. $\mathbb{R}_\infty = \mathbb{R} \cup \{\infty\}$



4.2. Gráf tranzitív lezártja

4.2. Jelölés. $\mathbb{B} = \{0; 1\}$