Legrövidebb utak egy forrásból*

Vadász Péter

1 Elméleti összefoglaló [1]

Gyakori feladat, hogy egy gráf egy adott csúcsából más csúcsokba vezető legrövidebb utakat keresünk. Gondolhatunk például navigációs rendszerre, ahol a feladat két város közti legrövidebb út meghatározása. Ebben az esetben a városokat csúcsokkal, a köztük vezető utakat élekkel reprezentálhatjuk. Egy ilyen feladat esetén az úthossz többféleképpen is értelmezhető, az előző példa esetén vehetjük a városok között vezető utak hosszát és ezzel súlyozhatjuk az éleket. Korábban a szélességi bejárásnál az élek száma jelentette az út hosszát, most az élek súlyozását fogjuk figyelembe venni.

A megoldandó feladat általánosan: Adott egy $G : G_w$ tetszőleges élsúlyozott gráf és s start csúcs. Keressük a start csúcsból induló legrövidebb utakat minden olyan G-beli csúcsba, amely elérhető s-ből.

Külön meg kell említenünk a **negatív élsúlyok** esetét is. A következő algoritmusok közül kettő (Bellman-Ford, DAG) negatív élsúlyokkal is tud dolgozni, egy (Dijkstra) nem. Amennyiben a gráf tartalmaz a start csúcsból elérhető **negatív összsúlyú kört**, akkor a legrövidebb utak problémának nincs megoldása. Egy negatív összsúlyú körön újra és újra végighaladva folyamatosan legrövidebb utakat találnánk megállás nélkül. Fontos feladat, hogy amennyiben negatív élsúlyú gráfokkal dolgozunk, fel tudjuk ismerni a negatív összsúlyú kört (Bellman-Ford algoritmus), ellenkező esetben az algoritmusunk végtelen működésbe kezdhet.

Alapvetően **irányított gráfokkal** foglalkozunk, de a feladat analóg módon **irányítatlan gráfok** esetén is megoldható, ebben az esetben azonban nem engedhetünk meg negatív éleket

*Dr Ásványi Tibor jegyzete alapján

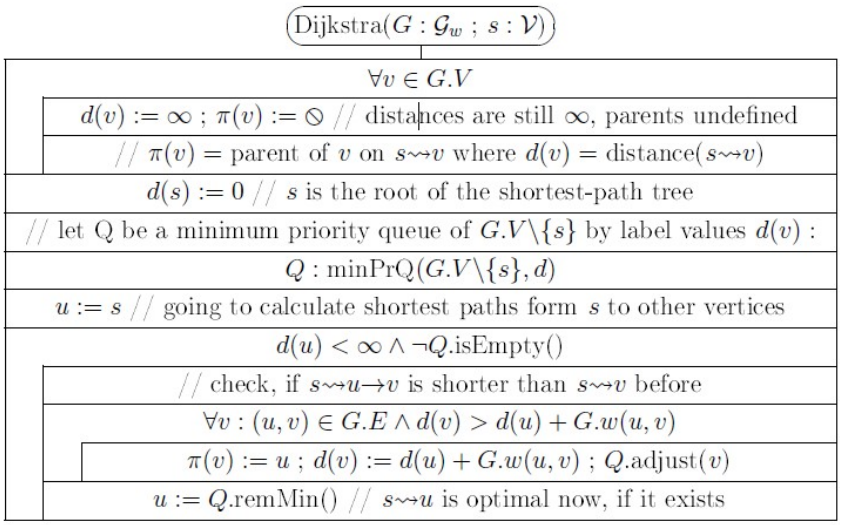
sem. Irányítatlan esetben egy G gráfot egy speciális irányított gráffal reprezentálunk, ahol ha $(u, v) \in G.E$, akkor $(v, u) \in G.E$ és $G.w(u, v) = G.w(v, u)$ (ld. előadás).

A következő algoritmusok esetén a **fokozatos közelítés** technikáját alakalmazzuk a legrövidebb utak kiszámításához. Minden $u \in G.V$ csúcs esetén nyilvántartjuk az oda vezető eddig talált legrövidebb út hosszát: ezt $d(u)$ jelöli, valamint ezen az úton az u csúcs közvetlen megelőzőjét, amit $\pi(u)$ jelöl. Az algoritmus működése során amennyiben egy $v \in G.V$ csúcs esetén az eddigieknél rövidebb utat találunk, akkor módosítjuk a csúcshoz tartozó $d(v)$ és $\pi(v)$ értékeket. A start csúcsra $d(s) = 0$ és $\pi(s) = \emptyset$, ha pedig egy $v \in G.V$ csúcs nem érhető el s-ből, akkor $d(v) = \infty$ és $\pi(v) = \emptyset$.

2 Dijkstra algoritmus

A Dijkstra algoritmus előfeltétele, hogy a gráfban nem lehet negatív súlyú él, azaz egy $G : G_w$ gráfra $\forall (u, v) \in G.E$ élre $G.w(u, v) \geq 0$.

Az algoritmus a 1. ábrán látható.



Ábra 1.: Dijkstra algoritmus [1]

A Dijkstra algoritmus egy mohó algoritmus, minden lépésben azt a csúcsot választja kiterjesztésre (feldolgozásra), melybe a start csúcsból a legrövidebb utat találta (az adott időpontig). Ehhez egy minimum prioritásos sort használ segédadatszerkezetként, a prioritásokat minden v csúcsra $d(v)$ értékek adják. Kezdetben minden csúcs bent van a prioritásos sorban, a start csúcs prioritása 0, a többi csúcsé ∞. Amikor egy csúcs kikerül

a sorból, akkor oda már a legrövidebb út vezet (bizonyítható), ezért a későbbiekben nem kell többet foglalkozni vele. Az algoritmus második ciklusának feltételében megfigyelhető a $d(u) < \infty$ feltétel is. Ha a sorból egy olyan csúcs kerül ki, melynek prioritása ∞, az csak akkor lehet, ha nem érhető el a start csúcsból (ellenkező esetben már tudnánk az oda vezető út hosszát). Ez azonban azt jelenti, hogy a sorban már csak olyan csúcsok vagy csúcs maradt, ami tehát nem érhető el a start csúcsból ezért az algoritmus megállhat.

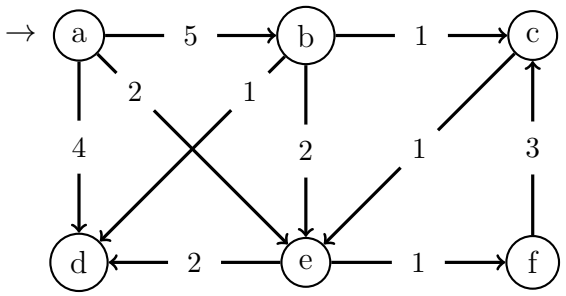
Az algoritmus **futási idejét** befolyásolja a prioritásos sor implementációja is, amit célszerű minimum kupaccal elkészíteni. Ha egy csúcshoz jobb utat találunk, mint az eddig kiszámolt, akkor a prioritásos sort is aktualizálni kell. A gráf minden csúcsát és minden élét legfeljebb egyszer dolgozzuk fel, a futási idő:

$MT(n, m) \in O((n + m) * \log n), mT(n, m) \in \Theta(n)$, ahol n a gráf csúcsainak, m a gráf éleinek száma.

2.1 Példák

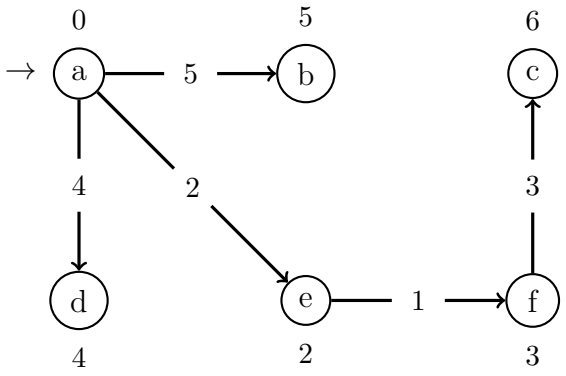
1. példa: Szemléltessük a Dijkstra algoritmus működését az alábbi G gráfon! Minden $v \in G.V$ csúcshoz adjuk meg $d(v)$ és $\pi(v)$ értékeket, az aktuálisan kiterjesztett csúcsot és a prioritásos sor tartalmát! Adjuk meg a legrövidebb utak fáját is!

Start csúcs: a



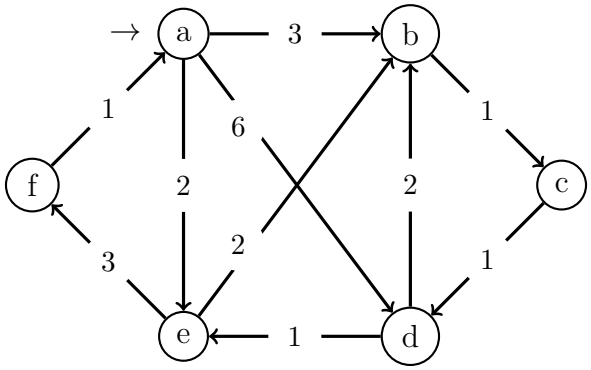
d értékek Q-ban						kiterjesztett csúcs:d	π címkék változásai					
a	b	c	d	e	f		a	b	c	d	e	f
0	∞	∞	∞	∞	∞	–	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
	5	∞	4	2	∞	a : 0		a		a	a	
	5	∞	4		3	e : 2						e
	5	6	4			f : 3			f			
	5	6				d : 4						
		6				b : 5						
						c : 6						
0	5	6	4	2	3	eredmény	$\emptyset$	a	f	a	a	e

A legrövidebb utak fája:



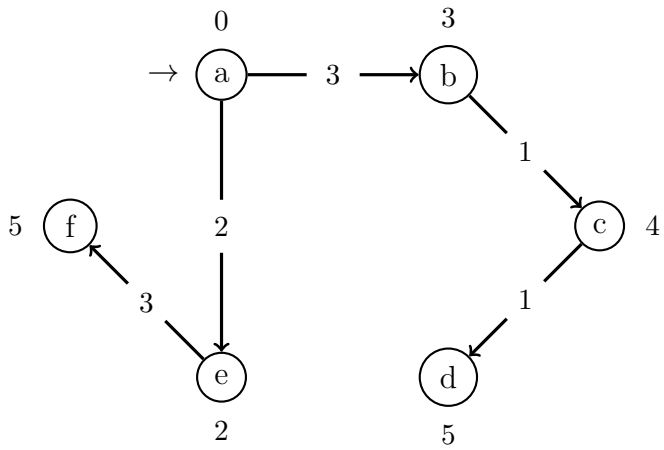
2. példa: Szemléltessük a Dijkstra algoritmus működését az alábbi G gráfon! Minden $v \in G.V$ csúcshoz adjuk meg $d(v)$ és $\pi(v)$ értékeket, az aktuálisan kiterjesztett csúcsot és a prioritásos sor tartalmát! Adjuk meg a legrövidebb utak fáját is!

Start csúcs: a

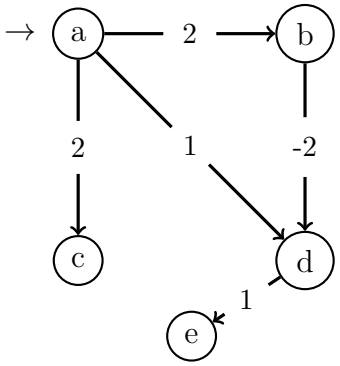


d értékek Q-ban						kiterjesztett csúcs:d	π címkék változásai					
a	b	c	d	e	f		a	b	c	d	e	f
0	∞	∞	∞	∞	∞	–	∅	∅	∅	∅	∅	∅
	3	∞	6	2	∞	a : 0		a		a	a	
	3	∞	6		5	e : 2						e
		4	6		5	b : 3			b			
			5		5	c : 4				c		
					5	d : 5						
0	3	4	5	2	5	eredmény	∅	a	b	c	a	e

A legrövidebb utak fája:



3. példa: Az alábbi gráf segítségével nézzük meg, mi történik, ha negatív élsúlyokat is használunk!



d értékek Q-ban					kiterjesztett csúcs:d	π címkék változásai				
a	b	c	d	e		a	b	c	d	e
0	∞	∞	∞	∞	–	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
	2	2	1	∞	a : 0		a	a	a	
	2	2		2	d : 1					d
		2	0	2	b : 2				b	
				2	c : 2					
0	2	2	0	2	eredmény	$\emptyset$	a	a	b	d

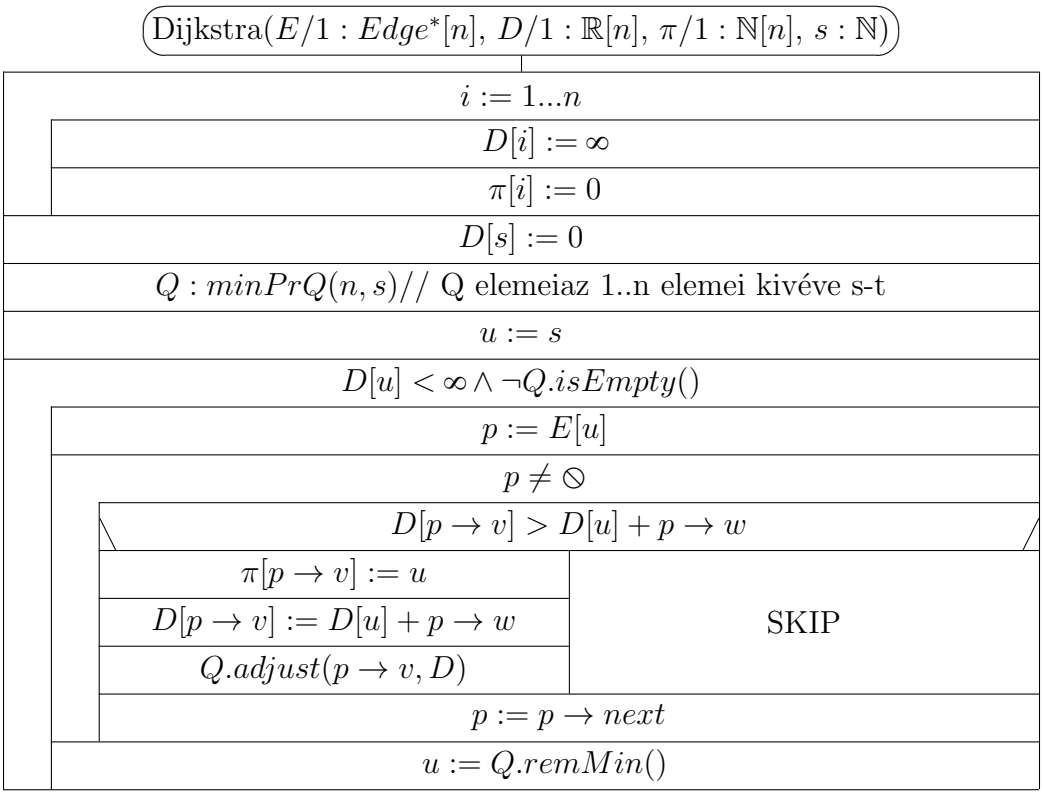
Láthatjuk, hogy ***b*** csúcs kiterjesztésekor optimálisabb utat találunk ***d*** csúcsba, hiszen $2 + -2 = 0$, ami jobb, mint a korábban talált 1 hosszú út. A problémát az okozza, hogy ***d*** csúcs már nincs a prioritásos sorban, és többet nem kerül kiterjesztésre, ami összeségében hibás végeredményhez vezet, mivel így az algoritmus nem "veszi észre", hogy ***e*** csúcshoz is vezet optimálisabb út ***b***-n keresztül.

A Dijkstra algoritmus elméleti alapjainál megfogalmaztunk egy állítást, miszerint, ha egy csúcs kikerül a sorból, akkor oda már optimális út vezet. Észrevehető, hogy ez az állítás nem teljesül negatív élsúlyok esetén.

4. példa:

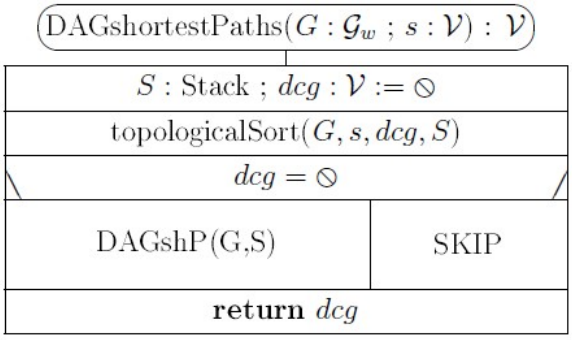
Készítsük el a Dijkstra algoritmust, szomszédossági éllistás gráfábrázolás esetén!

Az alábbi algoritmusban feltesszük, hogy a gráf csúcsait 1-től n-ig sorszámozzuk, ahol ***n*** értéke adott.

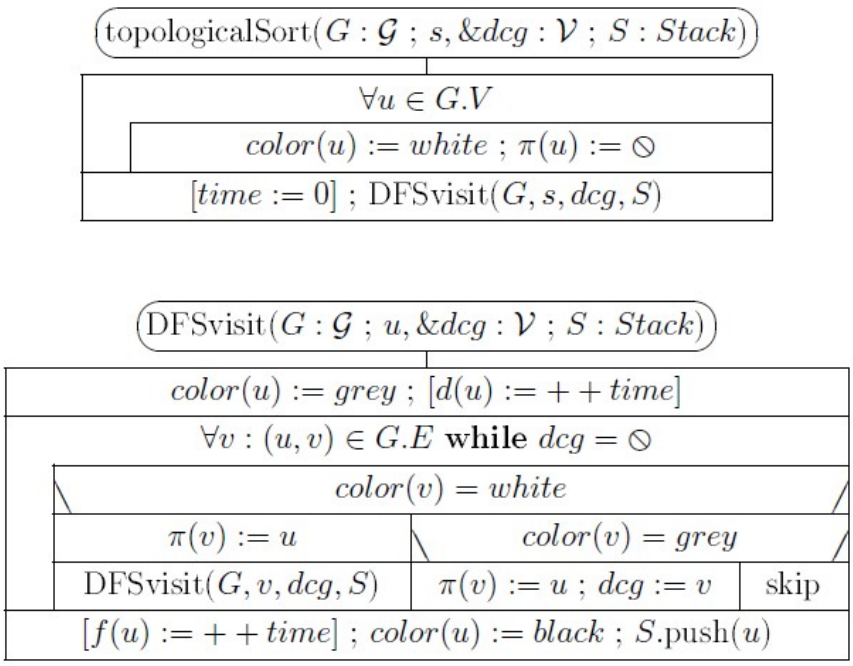


3 Legrövidebb utak egy forrásból DAG esetén

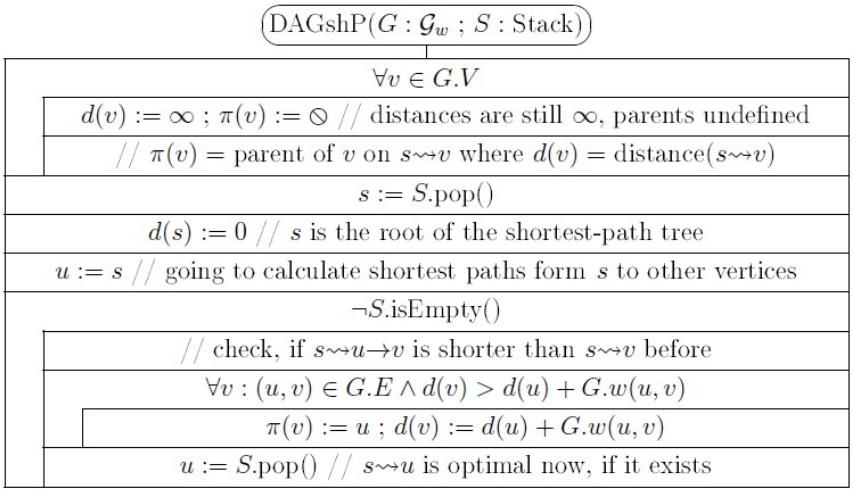
Adott $G : G_w$ irányított gráf, melyben nincs a start csúcsból elérhető kör (így természetesen negatív összsúlyú kör sem lehet), a gráf tartalmazhat negatív éleket is. Ebben az esetben létezik aszimptotikusan optimális algoritmus az adott csúcsból induló legrövidebb utak meghatározásához. Készítsük el a gráf részleges topologikus rendezését (teljes topologikus rendezés is készíthető) mélységi keresés (DFS) segítségével. Ha a gráf mégis tartalmazna kört, a DFS futása közben ez kiderül és ekkor leállíthatjuk az algoritmust. Ezután terjesszük ki a csúcsokat a topologikus rendezésnek megfelelő sorrendben.



Ábra 2.: Legrövidebb utak DAG esetén - fő eljárás [1]



Ábra 3.: Legrövidebb utak DAG esetén - topologikus rendezés [1]



Ábra 4.: Legrövidebb utak DAG esetén - kiterjesztés [1]

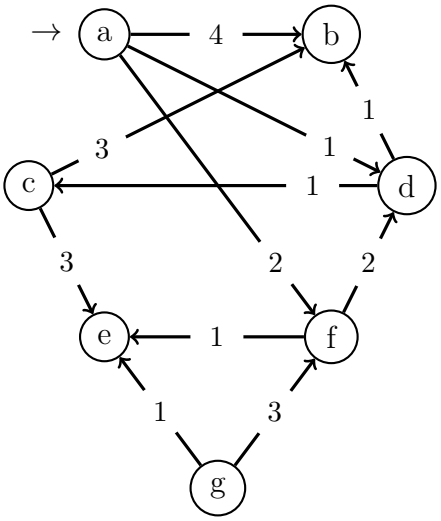
A **futási idő** a topologikus rendezés és a csúcsok kiterjesztése (minden csúcsot és élt legfeljebb egyszer terjesztünk ki) alapján: $MT(n, m) \in \Theta(n + m), mT(n, m) \in \Theta(n)$, ahol n a gráf csúcsainak, m a gráf éleinek száma.

3.1 Példák

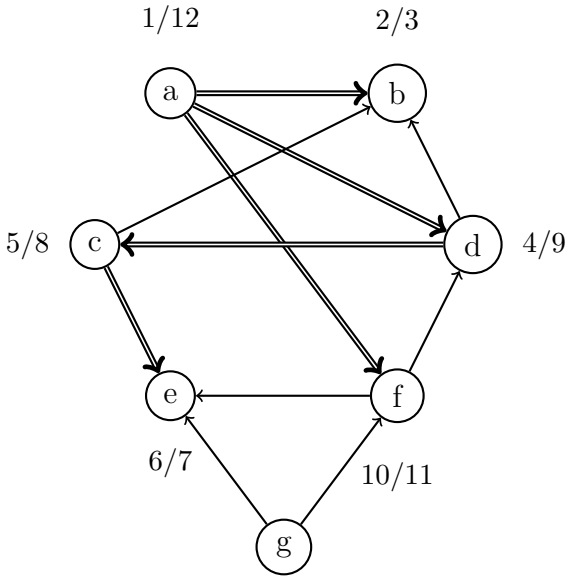
5. példa: Szemléltessük a *DAG legrövidebb utak egy forrásból* algoritmus működését az alábbi G gráfon! Először készítsük el a részleges topologikus rendezést, jelöljük az egyes

csúcsok elérési és befejezési számait is! Minden $v \in G.V$ csúcshoz adjuk meg $d(v)$ és $\pi(v)$ értékeket és az aktuálisan kiterjesztett csúcsot! Adjuk meg a legrövidebb utak fáját is!

Start csúcs: a



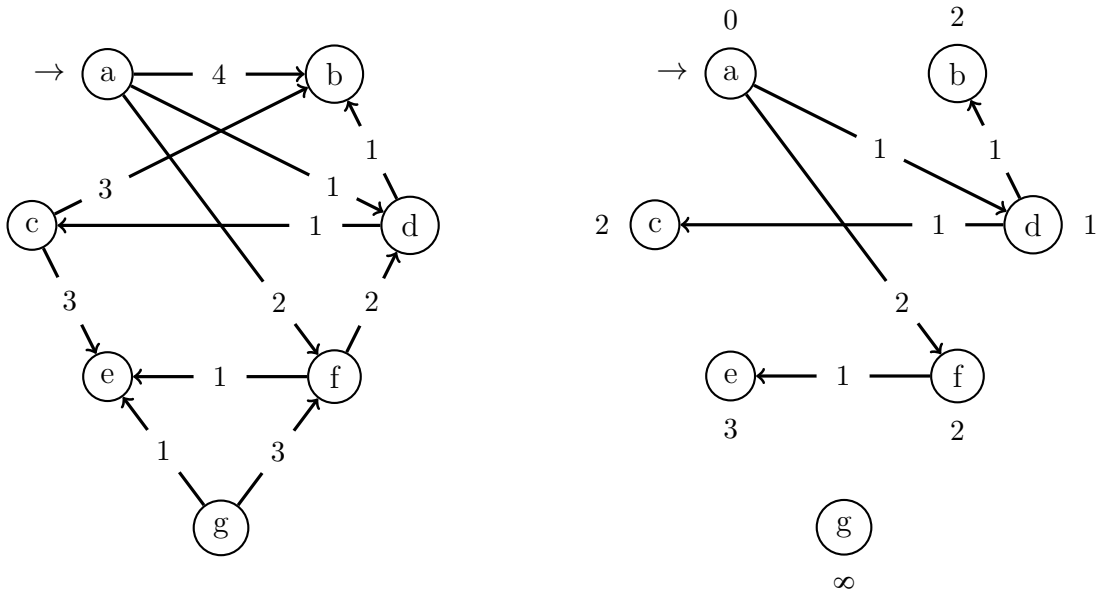
Mélységi keresés segítségével készítsük el a részleges topologikus rendezést, az a csúcsból indulva:



A csúcsok topologikus sorrendjét megkaphatjuk, a befejezési számok szerint csökkenő sorrendben haladva: **a, f, d, c, e, b** (A csúcsot, feldolgozásának befejezésekor, verembe tesszük. A start csúcs kerül a verem tetejére.)

d értékek							kiterjesztett csúcs:d	π címkék változásai						
a	b	c	d	e	f	g		a	b	c	d	e	f	g
0	∞	∞	∞	∞	∞	∞	–	∅	∅	∅	∅	∅	∅	∅
	4		1		2		a : 0		a		a		a	
				3			f : 2					f		
	2	2					d : 1		d	d				
							c : 2							
							e : 3							
0	2	2	1	3	2	∞	eredmény	∅	d	d	a	f	a	∅

Az eredeti gráf (balra) és a legrövidebb utak fája (jobbra):



4 Sor alapú Bellman-Ford algoritmus

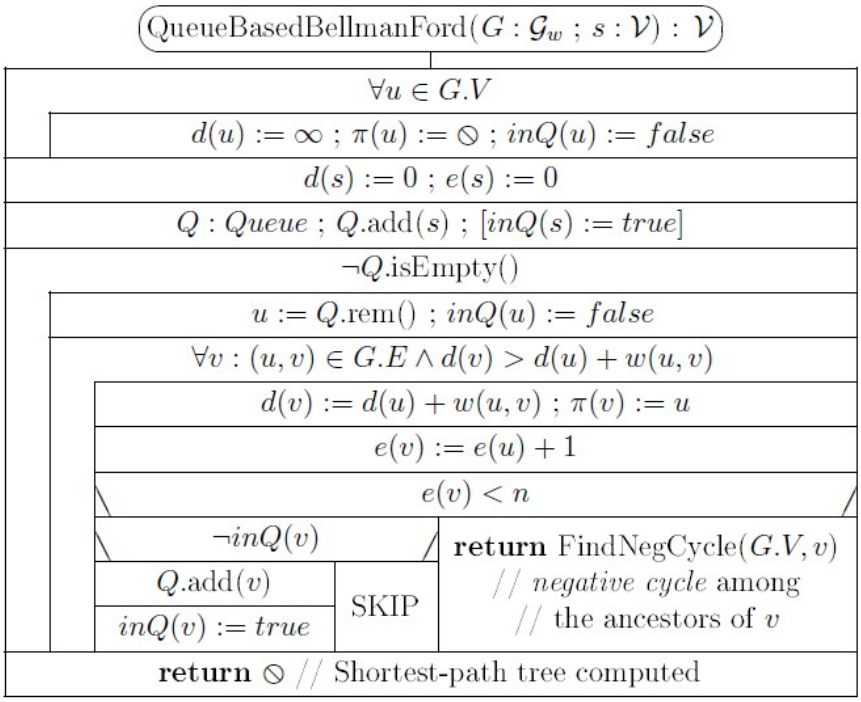
A sor alapú Bellman-Ford algoritmus a szélességi bejárás egy speciális módosulata. Irányítatlan gráf esetén az algoritmus akkor ad megoldást, ha a gráf nem tartalmaz a start csúcsból elérhető negatív súlyú élt, irányított esetben azonban megendett a negatív súlyú él, de a start csúcsból elérhető negatív összsúlyú kör nem. A sor alapú Bellman-Ford algoritmus egyik előnye, hogy képes felismerni, ha van a gráfban a start csúcsból elérhető negatív összsúlyú kör és ekkor az algoritmus leállítható, továbbá a kör is detektálható.

A sor alapú Bellman-Ford algoritmus a szélességi bejáráshoz hasonlóan egy sort használ segédadatszerkezetként. Kezdetben csak a start csúcs van a sorban, de egy csúcs többször

is visszakerülhet a sorba. A szélességi bejáráshoz hasonlóan azt is tároljuk, hogy az egyes csúcsokhoz hány élből álló út vezet (jelölés $e(u)$). Egy n csúcsú gráf esetén két különböző csúcs között legfeljebb $n-1$ élből álló optimális út vezethet, ezért ha az algoritmus működése során valamely $u \in G.V$ csúcs kiterjesztésekor $e(u) = n$, akkor negatív összsúlyú kört találtunk.

A sor alapú Bellman-Ford algoritmus először a start csúcsból elérhető 1 élből álló legrövidebb utakat határozza meg, majd a start csúcsból elérhető 2 élből álló optimális utakat, egészen $n-1$-ig. Használjuk a **menet** fogalmát is, amely legkönnyebben rekurzív módon definiálható:

- 0. menet: start csúcs feldolgozása
- $(i+1)$. menet: az i-edik menet végén a sorban lévő csúcsok feldolgozása

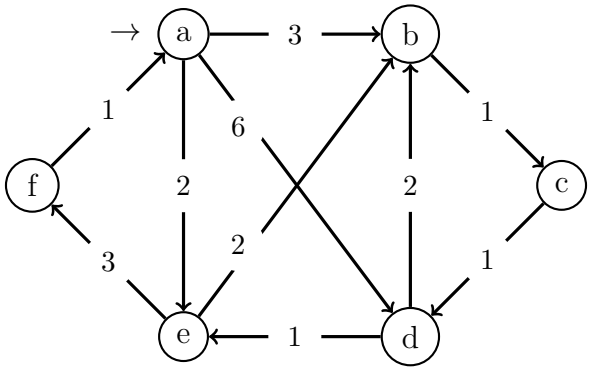


Ábra 5.: A sor alapú Bellman-Ford algoritmus [1]

4.1 Példák

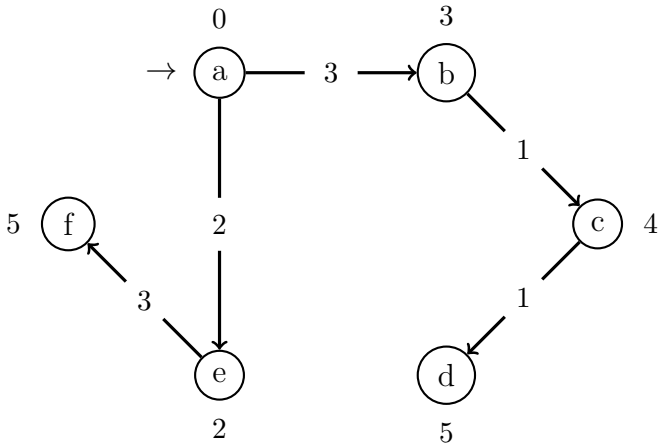
6. példa: Szemléltessük a Bellman-Ford algoritmus működését az alábbi G gráfon! Minden $v \in G.V$ csúcshoz adjuk meg $d(v)$ és $\pi(v)$ értékeket, az aktuálisan kiterjesztett csúcsot és a sor tartalmát! Adjuk meg a legrövidebb utak fáját is!

Start csúcs: a



$d; e$ változásai						kiterjesztés:d;e	Q	π változásai						menet
a	b	c	d	e	f			a	b	c	d	e	f	
0; 0	∞	∞	∞	∞	∞	–	⟨ a ⟩	∅	∅	∅	∅	∅	∅	–
	3;1		6;1	2;1		a : 0;0	⟨ b, d, e ⟩		a		a	a		0
		4;2				b : 3;1	⟨ d, e, c ⟩			b				1
						d : 6;1	⟨ e, c ⟩							1
					5;2	e : 2;1	⟨ c, f ⟩						e	1
			5;3			c : 4;2	⟨ f, d ⟩				c			2
						f : 5;2	⟨ d ⟩							2
						d : 5;3	⟨ ⟩							3
0	3	4	5	2	5	eredmény	–	∅	a	b	c	a	e	–

A legrövidebb utak fája:

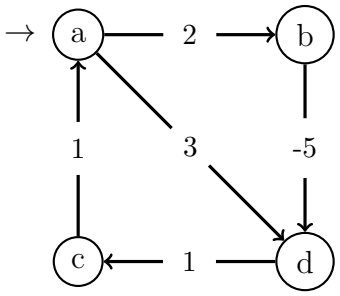


A példa során négy menetet hajtott végre az algoritmus ([0..3]). Láthatjuk, hogy a nulladik menetben a start csúcsot dolgozzuk fel, majd az első menetben azokat a $v \in G.V$

csúcsokat, melyeket a start csúcsból legalább egy élen keresztül érünk el, azaz $e(v) \geq 1$, és így tovább.

7. példa: Szemléltessük a Bellman-Ford algoritmus működését az alábbi negatív összsúlyú kört tartalmazó G gráfon! Minden $v \in G.V$ csúcshoz adjuk meg $d(v)$ és $\pi(v)$ értékeket, az aktuálisan kiterjesztett csúcsot és a sor tartalmát!

Start csúcs: a



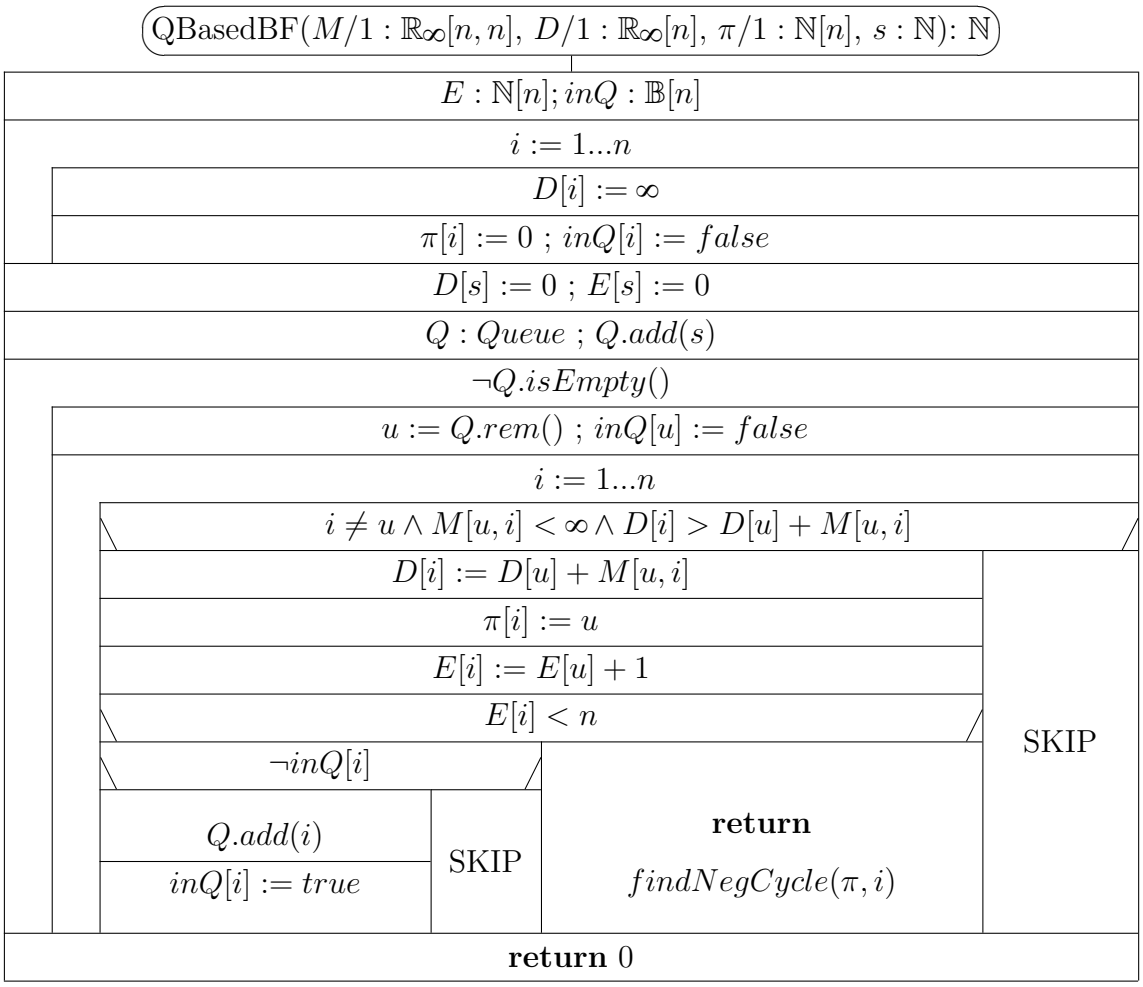
$d; e$ változásai				kiterjesztés:d;e	Q	π változásai				menet
a	b	c	d			a	b	c	d	
0; 0	∞	∞	∞	–	$\langle$ a $\rangle$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	–
	2;1		3;1	a : 0;0	$\langle$ b, d $\rangle$		a		a	0
			-3;2	b : 2;1	$\langle$ d $\rangle$				b	1
		-2;3		d : -3;2	$\langle$ c $\rangle$			d		1
-1; ④				c : -2;3	$\langle$ a $\rangle$	c				2

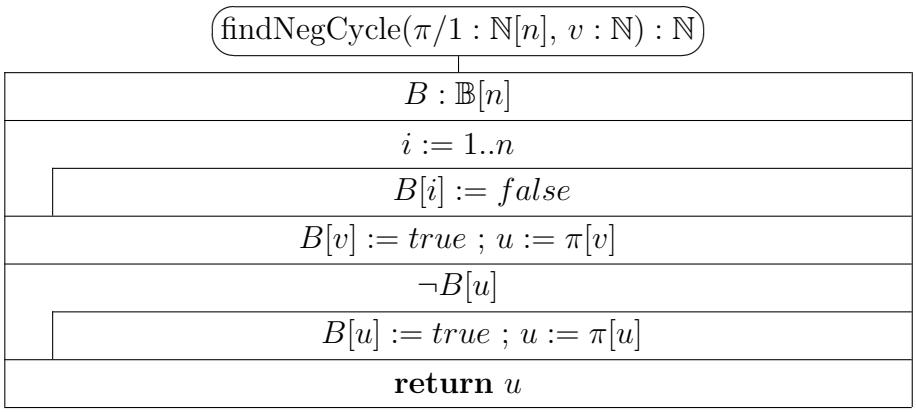
A táblázat utolsó sorában azt láthatjuk, hogy ***a*** csúcshoz 4 élből álló utat találtunk, ami ellentmond annak, hogy a 4 csúcsú gráfban tetszőleges körmentes út legfeljebb 3 élből állhat. Ezért kört tartalmazó utat találtunk. Ha viszont a talált kör nem lenne negatív, a talált út nem lehetne javító út a korábban talált körmentes úthoz képest. Következőleg negatív kört találtunk. Az is ezt mutatja, hogy most a start csúcsnak a ***c*** csúcs a szülője.

8. példa:

Készítsük el a sor-alapú Bellman-Ford algoritmust, csúcsmátrixos gráfreprezentáció esetén!

Az alábbi algoritmusban feltesszük, hogy a gráf csúcsait 1-től n-ig sorszámozzuk, ahol ***n*** értéke adott.





4.2 Gyakorlati alkalmazás[2]

A sor-alapú Bellman-Ford algoritmus egy elosztott (távolságvektoros) változatát számítógépes hálózatoknál routing protokollok is használják az optimális útvonalválasztáshoz. Korábban az ARPANET útválasztó algoritmusa volt.

Irodalomjegyzék

[1] Dr Ásványi Tibor *Algoritmusok és adatszerkezetek II. előadásjegyzet - Élsúlyozott gráfok és algoritmusaik*

[2] Andrew S. Tanenbaum, David J.Wetherall *Számítógéphálózatok*