25. MINIMÁLIS KÖLTSÉGŰ UTAK II.

A minimális költségű utak problémája olyan esetben is felmerül, amelyben a gráf élei *negatív költségértékkel* (súllyal) is rendelkezhetnek. Mindössze azt *zárjuk ki*, hogy a gráf *negatív összköltségű kört* tartalmazzon.

Gondoljunk egy olyan útvonal tervezési feladatra, amelyekben bizonyos útszakaszain bevételhez jutunk. A gráfos modellben egy él pozitív súlya azt jelenti, hogy azon a szakaszon a ráfordítás a nagyobb, míg a negatív költség arra utal, hogy ott a bevétel felülmúlja a kiadásainak. Ha egy negatív körön ismételten végighaladva, egyre növekedne a nyereségünk. Ezért a *negatív körök kizárása* a gyakorlat szempontjából is érthető feltétel.

Ebben a feladatban is egy kezdőcsúcsból a gráf *összes* csúcsához keressük a legkisebb költségű utat. A feladat megoldására a *Bellman-Ford-algoritmust* ismertetjük. Ez ugyanazt a *közelítést* alkalmazza az éleken, mint a Dijkstra-algoritmus, vagyis ha egy csúcshoz az addigi értéknél kisebb költségű utat talál az éppen vizsgált élen keresztül, akkor a csúcshoz ezt a javított értéket jegyzi be, valamint módosítja is az elérési útvonalat.

Az eljárás viszont *nem* követhet *mohó* stratégiát, szemben Dijkstra algoritmusával, ugyanis bármely már elért csúcs esetében előfordulhat az, hogy a még be nem járt csúcsok bevonásával – a negatív élköltségek folytán – javíthatunk a forrásból odavezető út addigi költségén.

25.1. A minimális költségű utak problémája (negatív élekkel)

Feladat. Adott egy $G = (V, E)$ élsúlyozott, irányított vagy irányítás nélküli, negatív összköltségű kört nem tartalmazó véges gráf, továbbá egy $s \in V$ forrás (kezdőcsúcs). Határozzuk meg minden $v \in V$ csúcsra az s-ből v-be vezető legrövidebb utat, annak költségével együtt.

Vegyük szemügyre még egyszer a *negatív kör* jelenségét! A kezdőcsúcsból elérhető negatív összköltségű körön nem léteznének legkisebb költségű utak, mivel az illető körön tetszőlegesen sokszor végig menve az utak költsége mindig tovább csökkenthető lenne.

Irányítatlan gráf esetén, egy (u, v) negatív súlyú irányítatlan élen oda-vissza haladva az út költsége szintén korlátlanul csökkenthető lenne, azaz úgy viselkedne, mint egy negatív összköltségű kör. Tekintsük az irányítás nélküli élte tehát negatív összköltségű, két élből álló irányított körnek. Ez egybevág az ábrázolás szintjén megvalósított irányítatlan gráffal, ahol egy irányítatlan élt, egy oda-vissza irányított élpárral valósítunk meg. Tehát irányítatlan gráf esetén a megszorításunk az, hogy egyáltalán ne tartalmazzon negatív súlyú élt, mert az negatív irányított körnek tekinthető.

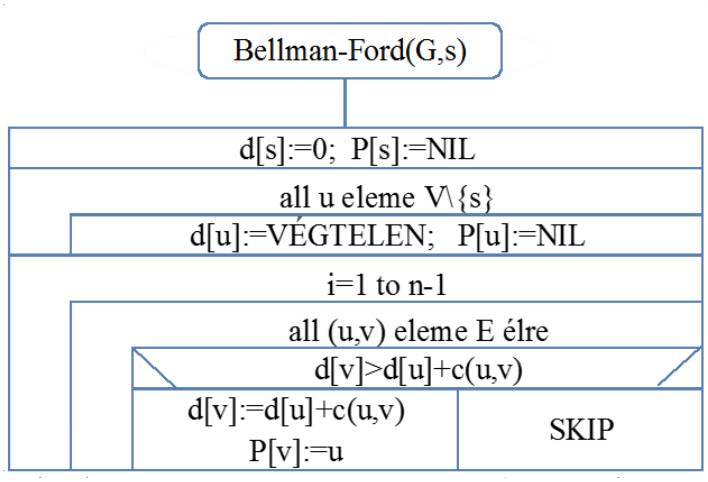
25.2. A Bellman-Ford algoritmus

Minden csúcsra, ha létezik legrövidebb út, akkor létezik *egyszerű legrövidebb út* is, mivel a körök összköltsége nem negatív, így a kört elhagyva az út költsége nem nőhet. Egy n pontú gráfban, a *legnagyobb élszámú* egyszerű út hossza *legfeljebb* $n - 1$ lehet.

A *Bellman-Ford-algoritmus* a Dijkstra algoritmusnál megismert közelítés műveletét végzi, azaz egy csúcson át a szomszédba vezető él mentén vizsgálja, hogy az illető él része-e a legrövidebb útnak, javító él-e. Egy menetben az összes élre megvizsgálja, hogy javító él-e vagy sem. Összesen $n - 1$ menetet végez. (Az alfejezet végén visszatérünk az iterációk számára.)

Vizsgáljunk meg egy $p^* = s \rightsquigarrow u$ legrövidebb utat. Minden menetben a p^* minden élén végzünk közelítést. Legyen $v \to w$ él része p^*-nak. Miután p^* v-ig tartó részútja ${p^*}_v$ ismertté válik, a következő menetben a ${p^*}_w$ is ismert lesz, mivel az (v, w) éllel is végzünk közelítést. Azonban az élek feldolgozásának (közelítésének) sorrendjére nem tettünk semmilyen megkötést, így csak azt tudjuk garantálni, hogy az első lépés után az 1 élszámú legrövidebb utak, a második lépés után a 2 élszámú legrövidebb utak, és így tovább, válnak ismerté. Mivel a leghosszabb egyszerű út $n - 1$ élszámú, ezért szükséges lehet az $n - 1$ menet.

A *Bellman-Ford-algoritmus* ADT szintű leírása a 25.1. ábrán látható.



25.1. ábra. A Bellman-Ford algoritmus

Állítás. Ha adott egy $G = (V, E)$ élsúlyozott, irányított vagy irányítás nélküli, negatív összköltségű irányított kört nem tartalmazó véges gráf, továbbá egy $s \in V$ forrás (kezdőcsúcs). Ekkor a Bellman-Ford algoritmus meghatározza $\forall v \in V$ csúcsra legrövidebb utat és annak hosszát.

Bizonyítás. Legyen $p^* = \langle v_0, \dots, v_{k-1}, v_k \rangle$ egy s-ből u-ba vezető, egyszerű legrövidebb út, ahol $s = v_0$ és $u = v_k$. Teljes indukcióval belátjuk, hogy az i-dik menet után már ismert $\langle v_0, \dots, v_i \rangle$ legrövidebb részút, azaz $d[v_i] = \delta(s, v_i)$ és $P[v_i] = v_{i-1}$, és ez már nem romlik el később sem.

Kezdetben az inicializáló lépés után $d[s] = d[v_0] = \delta(s, v_0) = 0$. Ez fennmarad, különben létezne egy olyan s-ből elérhető u csúcs, amelyre $s \rightsquigarrow u \to s$ és $d(s \rightsquigarrow u) + c(u, s) < 0$, ami azt jelenti, hogy találtunk egy negatív kört.

A bizonyítás $i - 1 \to i$ általános lépésében tegyük fel, hogy ismert p^*-nak $s \rightsquigarrow v_{i-1}$ részútja. Az i-dik menetben a (v_{i-1}, v_i) éllel is végzünk közelítést (feljegyezzük $d[v_i] = \delta(s, v_i)$ és $P[v_i] = v_{i-1}$), és ez csak akkor nem történik meg ((v_{i-1}, v_i) nem javító él), ha $\delta(s, v_i)$ már ismert, azaz a v_i-be vezető egyik legrövidebb utat már korábban megtaláltuk. A későbbiek során ez már nem változhat, mivel ha ez változna, az azt jelentené, hogy létezik a legrövidebb útnál rövidebb út, mivel p^* is legrövidebb út és annak bármely részútja, így p^*-nak $s \rightsquigarrow v_i$ részútja is legrövidebb út.

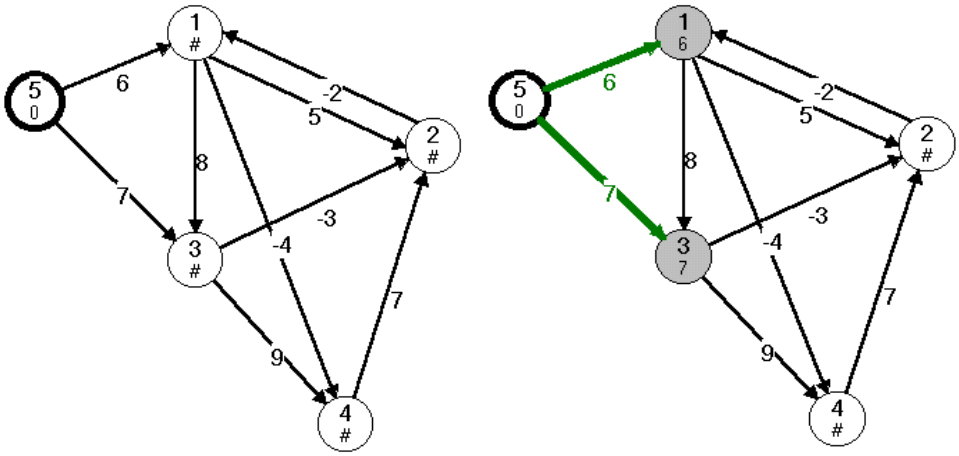
Mivel a legnagyobb élszámú, egyszerű, legrövidebb út élszáma is legfeljebb n-1, ezért a fenti indukciós állításból következik az algoritmus helyessége.

A Bellman-Ford algoritmussal gyakran abban a változatban lehet találkozni, amelyben még egy *n-edik iteráció* is szerepel, annak *ellenőrzésére*, hogy a gráf nem tartalmaz *negatív kört*. Ezt az jelzi, hogy a csúcsok költségértékek már nem változnak ebben az „utolsó utáni” menetben. (Negatív kör esetén a költségértékek tömbjében valahol csökkenést tapasztalnánk.)

25.3. Az algoritmus szemléltetése

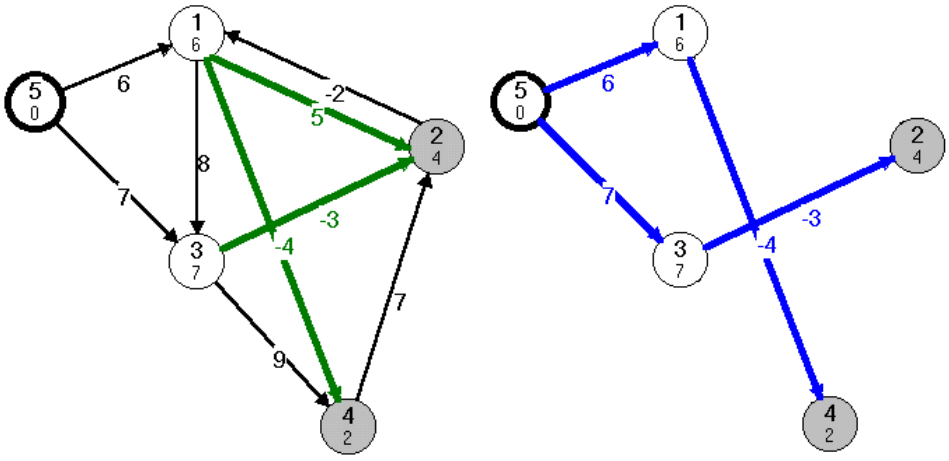
Nézzük meg egy példán, ADS szinten az algoritmus működését. Tegyük fel, hogy az élek feldolgozási sorrendje a csúcsok címkéje szerint rendezett: (1,2), (1,3), …, (5,3).

Az inicializáló lépés során beállítjuk a $d[1..n]$ és $P[1..n]$ tömb értékeit (25.2. ábra). A végtelen értéket most is '#' jellel jelöljük. Az első 7 él $((1,2),(1,3),\ldots,(4,2))$ közelítésénél nem történik változás, mivel végtelen értékek növelésénél szintén végtelent kapunk, ami nem javít. Csak két javító élt találunk. Most állíthatjuk, hogy minden csúcshoz megtaláltuk az s-ből hozzá vezető, minimális költségű, 1 élszámú utat.



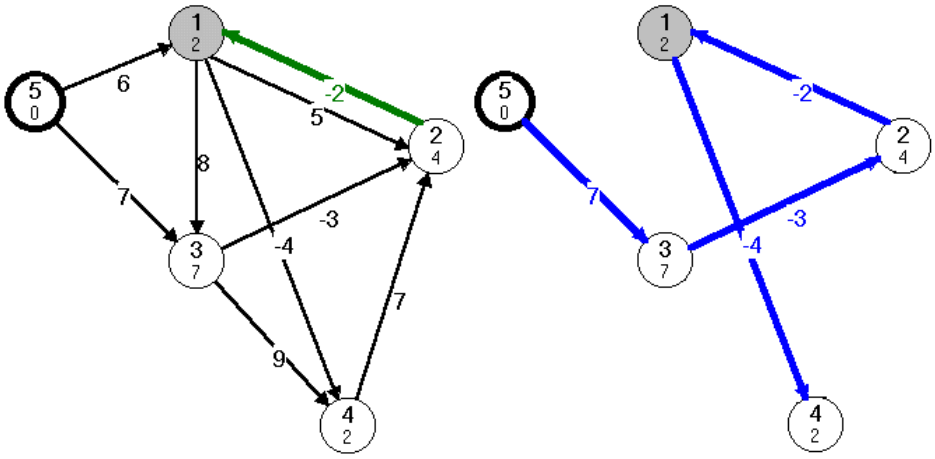
25.2. ábra. Az algoritmus inicializáló és 1. lépése

A 2. lépésben a 2. és 4. csúcsokhoz találunk javító élt (25.3. ábra). Ezzel minden csúcshoz meghatároztuk a legkisebb költségű, 1 vagy 2 élszámú utat. Az utolsó lépésen látható az egyes csúcsokba vezető, 1 vagy 2 élszámú legrövidebb utakból kialakult fa. Ez a fa változhat, mivel lehet, hogy egy csúcsba el lehet jutni nagyobb élszámú olcsóbb úton is.



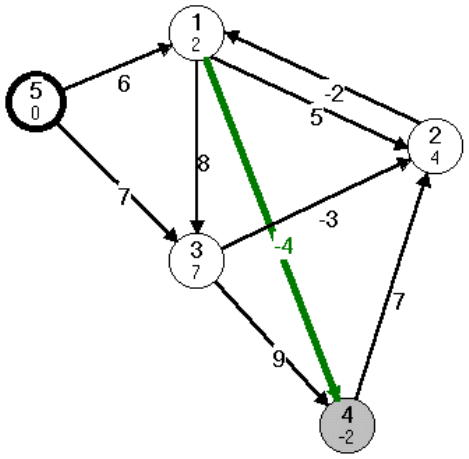
25.3. ábra. Az algoritmus 2. lépése és a kialakult fa

A 3. lépésben az 1-be olcsóbb 3 élszámú utat találtunk (25.4. ábra). A fa változott mivel az 1-be már nem 1 élszámú, hanem 3 élszámú, de kisebb költségű úton juthatunk el a kezdőcsúcsból. A 4 megelőzője a korábban talált 1-es, csak most nem 2 élhosszú úttal, hanem 4 élhosszúval ($\langle 5,3,2,1,4\rangle$). Mivel az (1,4) élt korábban dolgoztuk fel, mint (2,1) élt, így a 4-es csúcsnál bejegyzett költség nem konzisztens a fával. (Az élek sorrendje hathat úgy, hogy az algoritmus az utak keresésében „előresiet” és megtalál egy hosszabb utat, mint ami az iterációk számából következne. Az ehhez számolt költség lehet helyes, és lehet helytelen, de ezt a megfelelő későbbi menetben korrigálja az eljárás.)



25.4. ábra. Az algoritmus 3. lépése és a kialakult fa

A 4. lépésben (25.5. ábra) a fa már nem változik, csak 4-es csúcsnál bejegyzett költség veszi fel a helyes értéket.



25.5. ábra. Az algoritmus 4. lépése és a kialakult fa

25.4. Műveletigény

Mivel $n-1$ iteráció végzünk, és minden lépés során, minden élre végrehajtunk egy közelítést, ezért $T(n) = \Theta((n-1)\cdot e)$. Ez a műveletigény javítható a következő gyorsítással. (A buborék rendezésnél már láttunk hasonló gyorsítási lehetőséget.)

Állítás. Ha egy iteráció során nem következett be változás a közelítések során, akkor megállhatunk, az eljárás megtalálta az össze legkisebb költségű utat.

Bizonyítás. Indirekt módon tegyük fel, hogy létezik az algoritmus által megadott olyan legrövidebb $p^* = s \rightsquigarrow v \longrightarrow w \rightsquigarrow u$ út, hogy az i-dik lépésben az $s \rightsquigarrow v$ részutat már ismerjük, de $v \longrightarrow w$ él még nem része a fának, vagy $d[w]$ értéke nem konzisztens, tehát mindkét esetben $d[w] > \delta(s,w)$, továbbá az i-dik lépésben nem történik változás. Azonban $d[v] = \delta(s,v)$ és $\delta(s,w) = \delta(s,v) + c(v,w)$, továbbá az i-dik lépésben a (v,w) él közelítése során $d[w] \leq \delta(s,v) + c(v,w) = \delta(s,w)$, ami ellentmond az indirekt feltevésnek.

Ha a fenti gyorsítási lehetőséget beépítjük az algoritmusba, akkor az élek *feldolgozási sorrendje* befolyásolja az iterációk számát. Ha példánkban így átcímkéztük a csúcsokat: 5→1, 1→4, 2→3, 3→2, 4→5, és az élek feldolgozási sorrendje továbbra is csúcsok címkéje szerint rendezettséget követi, akkor *egyetlen* iterációs lépésben megkapjuk a megoldást. (Ezzel valójában az élek feldolgozási sorrendjét módosítottuk.)