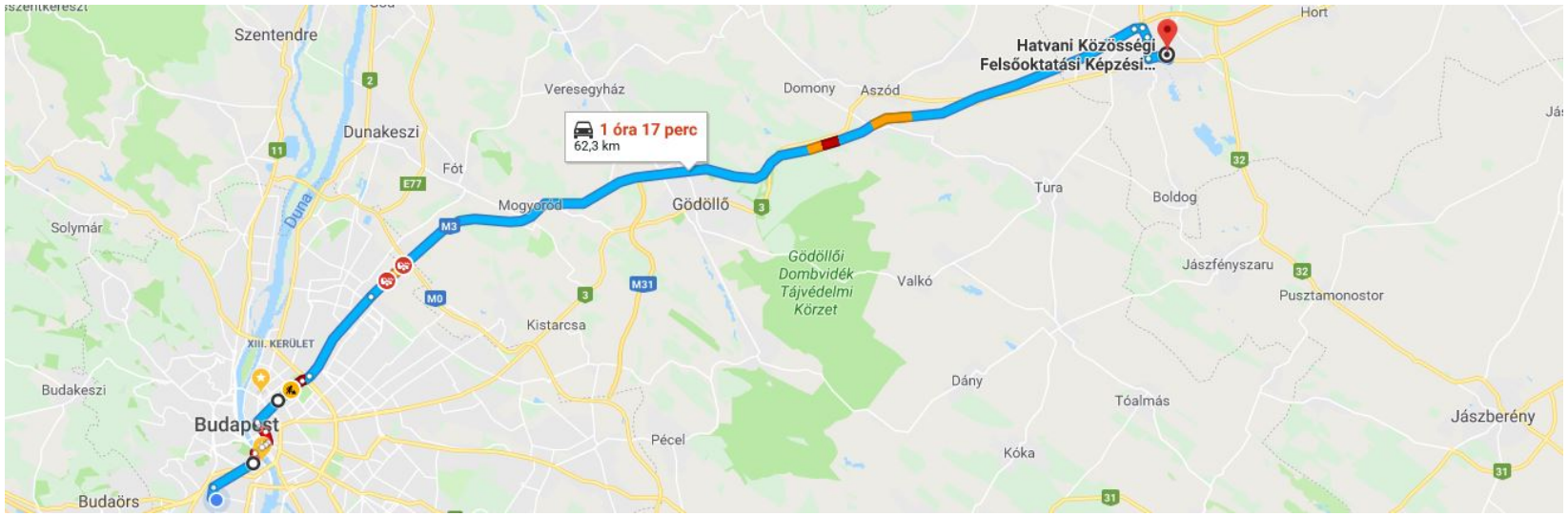
Legrövidebb utat kereső algoritmusok

László Papp

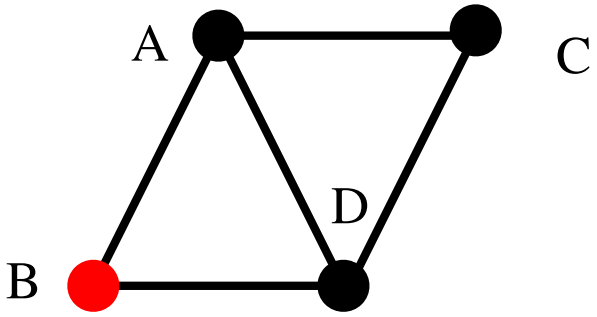
BME

2022. szeptember 24.



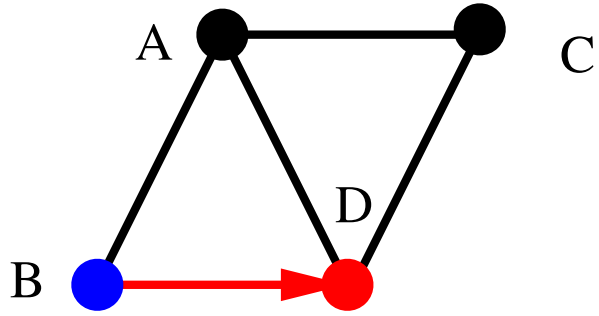
Gráfbejárások

Egy *G* gráf egy **bejárásán** a *G* csúcsainak valamilyen sorrendben történő végiglátogatását értjük. A cél az, hogy egy *v* csúcsot lehetőleg úgy látogassunk meg, hogy egy már bejárt *u* csúcsból lépünk át az *uv* élen keresztül. A cél a gráf összes csúcsának a bejárása.



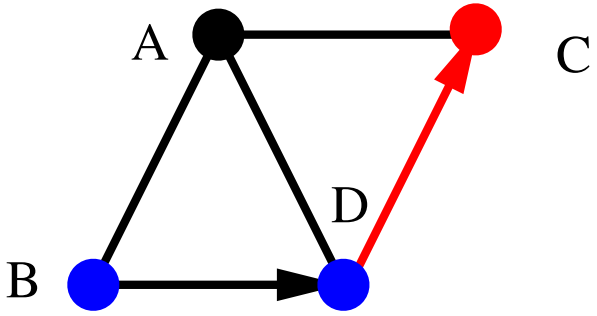
Gráfbejárások

Egy *G* gráf egy **bejárásán** a *G* csúcsainak valamilyen sorrendben történő végiglátogatását értjük. A cél az, hogy egy *v* csúcsot lehetőleg úgy látogassunk meg, hogy egy már bejárt *u* csúcsból lépünk át az *uv* élen keresztül. A cél a gráf összes csúcsának a bejárása.



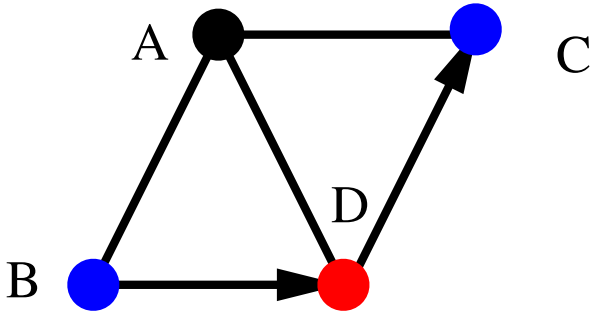
Gráfbejárások

Egy *G* gráf egy **bejárásán** a *G* csúcsainak valamilyen sorrendben történő végiglátogatását értjük. A cél az, hogy egy *v* csúcsot lehetőleg úgy látogassunk meg, hogy egy már bejárt *u* csúcsból lépünk át az *uv* élen keresztül. A cél a gráf összes csúcsának a bejárása.



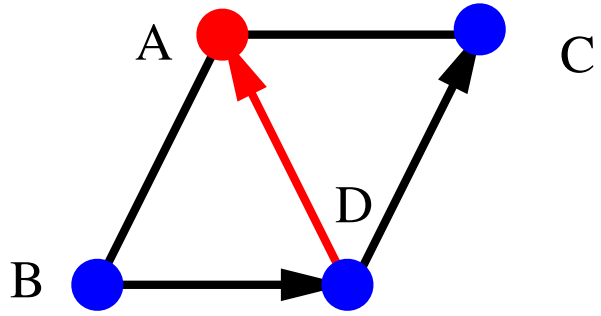
Gráfbejárások

Egy *G* gráf egy **bejárásán** a *G* csúcsainak valamilyen sorrendben történő végiglátogatását értjük. A cél az, hogy egy *v* csúcsot lehetőleg úgy látogassunk meg, hogy egy már bejárt *u* csúcsból lépünk át az *uv* élen keresztül. A cél a gráf összes csúcsának a bejárása.



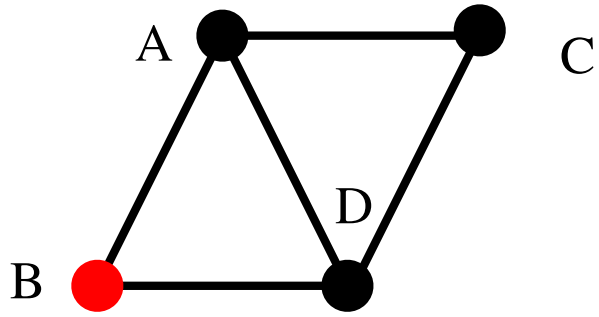
Gráfbejárások

Egy G gráf egy **bejárásán** a G csúcsainak valamilyen sorrendben történő végiglátogatását értjük. A cél az, hogy egy v csúcsot lehetőleg úgy látogassunk meg, hogy egy már bejárt u csúcsból lépünk át az uv élen keresztül. A cél a gráf összes csúcsának a bejárása.



Gráfbejárások

Egy *G* gráf egy **bejárásán** a *G* csúcsainak valamilyen sorrendben történő végiglátogatását értjük. A cél az, hogy egy *v* csúcsot lehetőleg úgy látogassunk meg, hogy egy már bejárt *u* csúcsból lépünk át az *uv* élen keresztül. A cél a gráf összes csúcsának a bejárása.

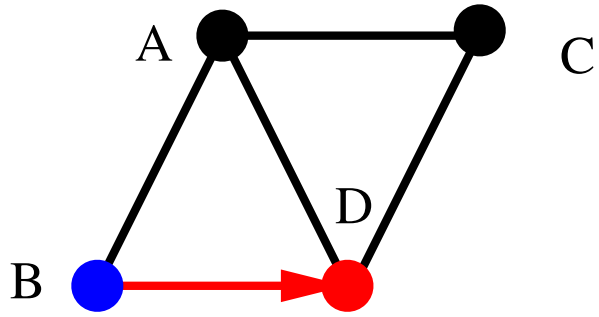


Minden bejáráshoz tartozik egy **elérési sorrend**, ami nem más mint *G* csúcsainak egy sorrendje, hogy mikor láttuk őket először a bejárás során. **Befejezési sorrenden** pedig *G* csúcsainak azon sorrendjét tekintjük amely megmondja, hogy milyen sorrendben foglalkoztunk velük utoljára.

A példa bejárás elérési sorrendje B, D, C, A, a befejezési sorrendje pedig B, C, D, A. (Ez a bejárás se nem BFS se nem DFS, BFS és DFS def később.)

Gráfbejárások

Egy *G* gráf egy **bejárásán** a *G* csúcsainak valamilyen sorrendben történő végiglátogatását értjük. A cél az, hogy egy *v* csúcsot lehetőleg úgy látogassunk meg, hogy egy már bejárt *u* csúcsból lépünk át az *uv* élen keresztül. A cél a gráf összes csúcsának a bejárása.

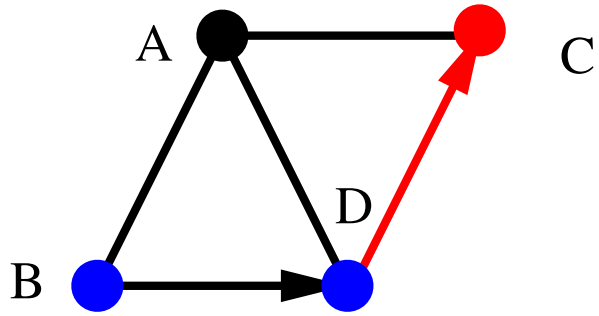


Minden bejáráshoz tartozik egy **elérési sorrend**, ami nem más mint *G* csúcsainak egy sorrendje, hogy mikor láttuk őket először a bejárás során. **Befejezési sorrenden** pedig *G* csúcsainak azon sorrendjét tekintjük amely megmondja, hogy milyen sorrendben foglalkoztunk velük utoljára.

A példa bejárás elérési sorrendje B, D, C, A, a befejezési sorrendje pedig B, C, D, A. (Ez a bejárás se nem BFS se nem DFS, BFS és DFS def később.)

Gráfbejárások

Egy *G* gráf egy **bejárásán** a *G* csúcsainak valamilyen sorrendben történő végiglátogatását értjük. A cél az, hogy egy *v* csúcsot lehetőleg úgy látogassunk meg, hogy egy már bejárt *u* csúcsból lépünk át az *uv* élen keresztül. A cél a gráf összes csúcsának a bejárása.

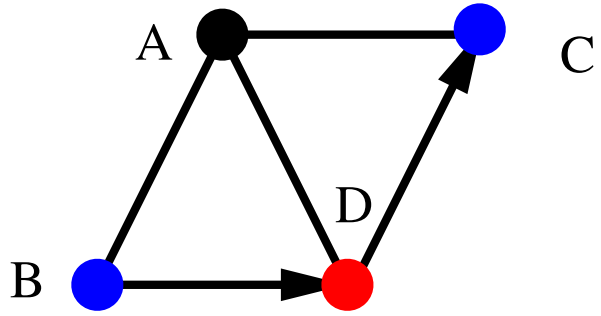


Minden bejáráshoz tartozik egy **elérési sorrend**, ami nem más mint *G* csúcsainak egy sorrendje, hogy mikor láttuk őket először a bejárás során. **Befejezési sorrenden** pedig *G* csúcsainak azon sorrendjét tekintjük amely megmondja, hogy milyen sorrendben foglalkoztunk velük utoljára.

A példa bejárás elérési sorrendje B, D, C, A, a befejezési sorrendje pedig B, C, D, A. (Ez a bejárás se nem BFS se nem DFS, BFS és DFS def később.)

Gráfbejárások

Egy *G* gráf egy **bejárásán** a *G* csúcsainak valamilyen sorrendben történő végiglátogatását értjük. A cél az, hogy egy *v* csúcsot lehetőleg úgy látogassunk meg, hogy egy már bejárt *u* csúcsból lépünk át az *uv* élen keresztül. A cél a gráf összes csúcsának a bejárása.

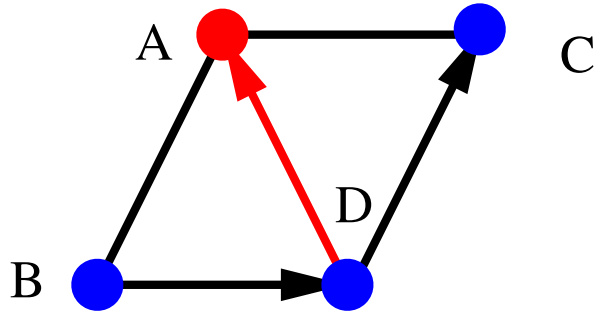


Minden bejáráshoz tartozik egy **elérési sorrend**, ami nem más mint *G* csúcsainak egy sorrendje, hogy mikor láttuk őket először a bejárás során. **Befejezési sorrenden** pedig *G* csúcsainak azon sorrendjét tekintjük amely megmondja, hogy milyen sorrendben foglalkoztunk velük utoljára.

A példa bejárás elérési sorrendje B, D, C, A, a befejezési sorrendje pedig B, C, D, A. (Ez a bejárás se nem BFS se nem DFS, BFS és DFS def később.)

Gráfbejárások

Egy *G* gráf egy **bejárásán** a *G* csúcsainak valamilyen sorrendben történő végiglátogatását értjük. A cél az, hogy egy *v* csúcsot lehetőleg úgy látogassunk meg, hogy egy már bejárt *u* csúcsból lépünk át az *uv* élen keresztül. A cél a gráf összes csúcsának a bejárása.

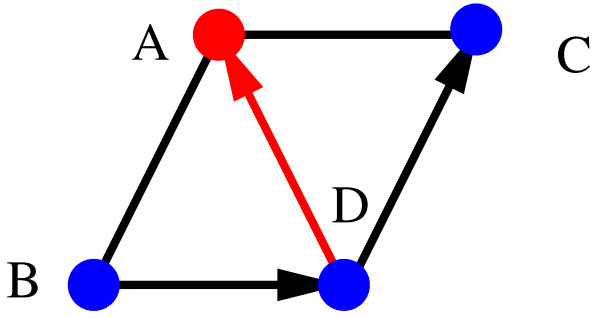


Minden bejáráshoz tartozik egy **elérési sorrend**, ami nem más mint *G* csúcsainak egy sorrendje, hogy mikor láttuk őket először a bejárás során. **Befejezési sorrenden** pedig *G* csúcsainak azon sorrendjét tekintjük amely megmondja, hogy milyen sorrendben foglalkoztunk velük utoljára.

A példa bejárás elérési sorrendje B, D, C, A, a befejezési sorrendje pedig B, C, D, A. (Ez a bejárás se nem BFS se nem DFS, BFS és DFS def később.)

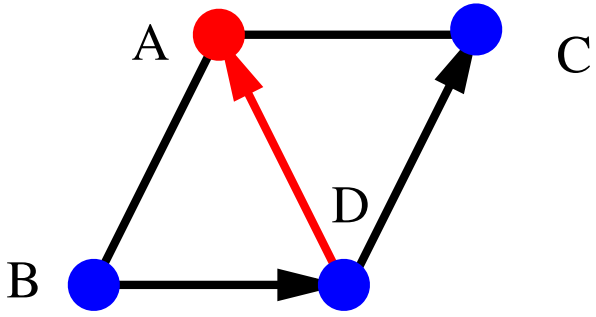
Bejárási fa

A csúcsok nagy részébe egy korábbi csúcsból kiinduló él mentén érkeztünk meg először. **Bejárási fának** nevezzük azt az irányított gráfot, mely pontosan ezen éleket az újonnan elért csúcsok fele irányítva tartalmazza. Ez a gráf körmentes, viszont nem feltétlen fa! Lehet erdő is.



Bejárási fa

A csúcsok nagy részébe egy korábbi csúcsból kiinduló él mentén érkeztünk meg először. **Bejárási fának** nevezzük azt az irányított gráfot, mely pontosan ezen éleket az újonnan elért csúcsok fele irányítva tartalmazza. Ez a gráf körmentes, viszont nem feltétlen fa! Lehet erdő is.



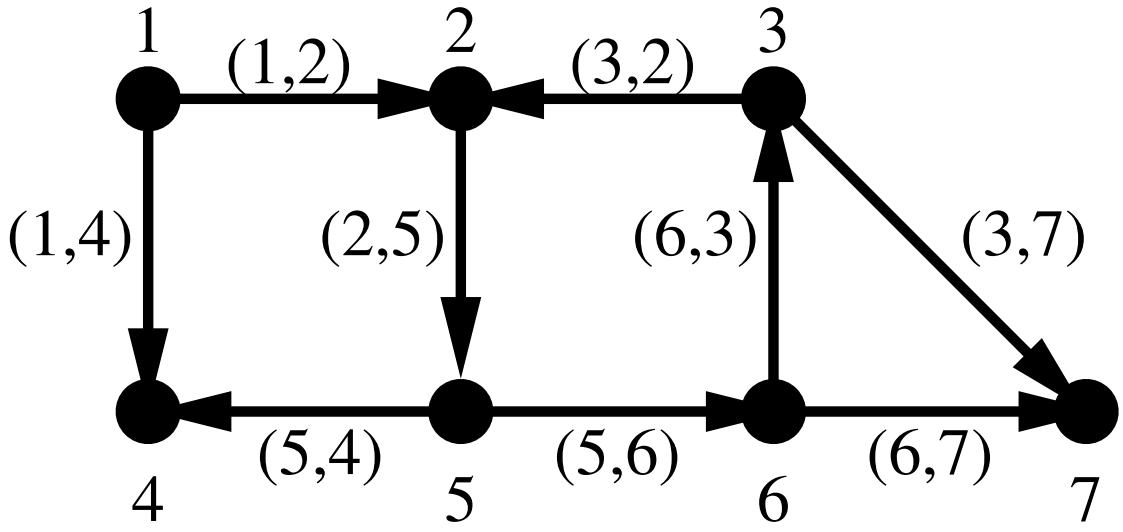
Definíció: Az *uv* él/irányított él **faél** a bejárásra nézve, ha a bejárási fának az éle. Amenyiben *uv* **nem faél**, akkor:

- uv **előreél** ha a bejárási fában *u*-ból vezet irányított út *v*-be,
- uv **visszaél** ha a bejárási fában *v*-ből vezet irányított út *u*-be,
- uv **keresztél** ha az előbbiek közül egyik sem.

Megjegyzés: Irányítatlan esetben az előreélek és a visszaélek ugyanazok.

Néhány definíció:

Definíció: Ha G egy irányított gráf és $(u, \vec{v})$ egy irányított él, akkor v-t az u **kiszomszédjának**, u-t pedig a v **beszomszédjának** nevezzük.



Példa: Az 2-es csúcsnak az 1-es és 3-as csúcsok beszomszédjai, az 5-ös csúcs a kiszomszédja.
Jelölés: Ha E egy lista, akkor az E lista i. elemét $E(i)$-vel jelöljük.

Szélességi bejárás, BFS(Breadth-first search)

Input: G (esetleg irányított) gráf, v_1 gyökérpont.

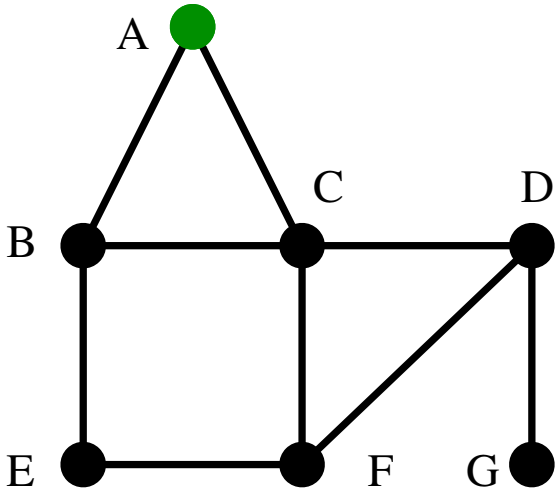
0. Létrehozunk egy kezdetben üres E listát.

1. Bejárjuk a v_1 csúcsot, v_1-et betesszük az E-be. $i := 1$.

2. Az $E(i)$ csúcs összes még be nem járt (ki-)szomszédját bejárjuk valamint ezen csúcsokat az E végére fűzzük. Az $E(i)$ csúcs és a most bejárt szomszédai között futó élekek bevesszük a bejárási fába, irányítatlan esetben úgy, hogy az $E(i)$ csúcs legyen a kezdőpontjuk.

3. Ha az E listában szerepel $i + 1$. csúcs, akkor az i-t növeljük 1-gyel és a 2-es lépésre ugrunk.

4. Output: E ami az elérési sorrend + bejárási fa



Elérési sorrend:

A

Befejezési sorrend:

Szélességi bejárás, BFS(Breadth-first search)

Input: G (esetleg irányított) gráf, v_1 gyökérpont.

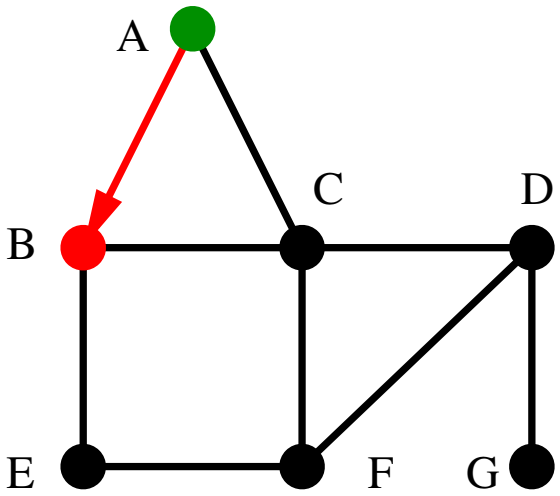
0. Létrehozunk egy kezdetben üres E listát.

1. Bejárjuk a v_1 csúcsot, v_1-et betesszük az E-be. $i := 1$.

2. Az $E(i)$ csúcs összes még be nem járt (ki-)szomszédját bejárjuk valamint ezen csúcsokat az E végére fűzzük. Az $E(i)$ csúcs és a most bejárt szomszédai között futó élekek bevesszük a bejárási fába, irányítatlan esetben úgy, hogy az $E(i)$ csúcs legyen a kezdőpontjuk.

3. Ha az E listában szerepel $i + 1$. csúcs, akkor az i-t növeljük 1-gyel és a 2-es lépésre ugrunk.

4. Output: E ami az elérési sorrend + bejárási fa



Elérési sorrend:
A, B
Befejezési sorrend:

Szélességi bejárás, BFS(Breadth-first search)

Input: G (esetleg irányított) gráf, v_1 gyökérpont.

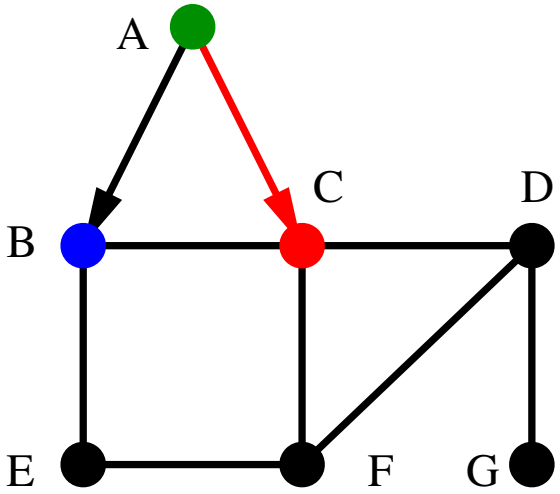
0. Létrehozunk egy kezdetben üres E listát.

1. Bejárjuk a v_1 csúcsot, v_1-et betesszük az E-be. $i := 1$.

2. Az $E(i)$ csúcs összes még be nem járt (ki-)szomszédját bejárjuk valamint ezen csúcsokat az E végére fűzzük. Az $E(i)$ csúcs és a most bejárt szomszédai között futó élekek bevesszük a bejárási fába, irányítatlan esetben úgy, hogy az $E(i)$ csúcs legyen a kezdőpontjuk.

3. Ha az E listában szerepel $i + 1$. csúcs, akkor az i-t növeljük 1-gyel és a 2-es lépésre ugrunk.

4. Output: E ami az elérési sorrend + bejárási fa



Elérési sorrend:
A, B, C
Befejezési sorrend:
A

Szélességi bejárás, BFS(Breadth-first search)

Input: G (esetleg irányított) gráf, v_1 gyökérpont.

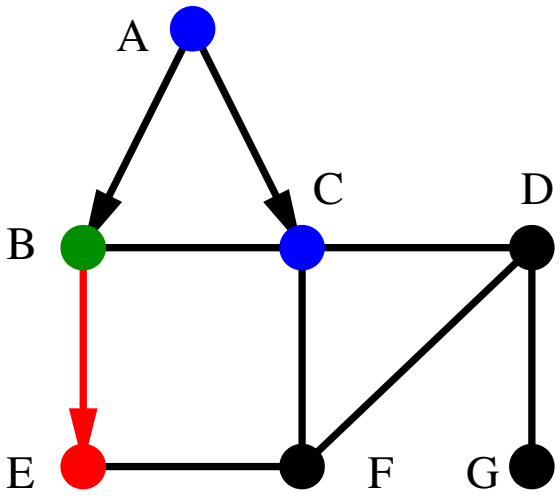
0. Létrehozunk egy kezdetben üres E listát.

1. Bejárjuk a v_1 csúcsot, v_1-et betesszük az E-be. $i := 1$.

2. Az $E(i)$ csúcs összes még be nem járt (ki-)szomszédját bejárjuk valamint ezen csúcsokat az E végére fűzzük. Az $E(i)$ csúcs és a most bejárt szomszédai között futó élekek bevesszük a bejárási fába, irányítatlan esetben úgy, hogy az $E(i)$ csúcs legyen a kezdőpontjuk.

3. Ha az E listában szerepel $i + 1$. csúcs, akkor az i-t növeljük 1-gyel és a 2-es lépésre ugrunk.

4. Output: E ami az elérési sorrend + bejárási fa



Elérési sorrend:
A, B, C, E
Befejezési sorrend:
A, B

Szélességi bejárás, BFS(Breadth-first search)

Input: G (esetleg irányított) gráf, v_1 gyökérpont.

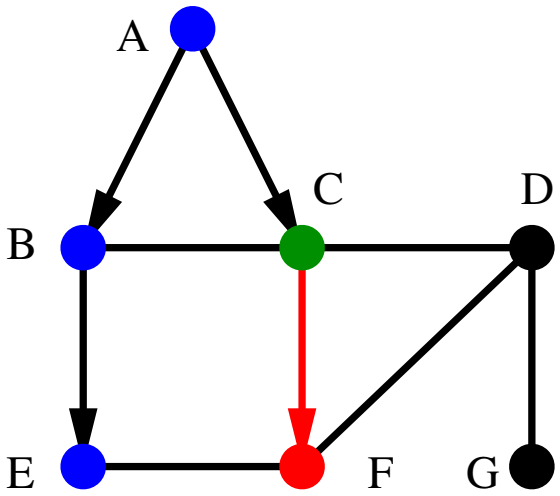
0. Létrehozunk egy kezdetben üres E listát.

1. Bejárjuk a v_1 csúcsot, v_1-et betesszük az E-be. $i := 1$.

2. Az $E(i)$ csúcs összes még be nem járt (ki-)szomszédját bejárjuk valamint ezen csúcsokat az E végére fűzzük. Az $E(i)$ csúcs és a most bejárt szomszédai között futó élekek bevesszük a bejárási fába, irányítatlan esetben úgy, hogy az $E(i)$ csúcs legyen a kezdőpontjuk.

3. Ha az E listában szerepel $i + 1$. csúcs, akkor az i-t növeljük 1-gyel és a 2-es lépésre ugrunk.

4. Output: E ami az elérési sorrend + bejárási fa



Elérési sorrend:
A, B, C, E, F
Befejezési sorrend:
A, B

Szélességi bejárás, BFS(Breadth-first search)

Input: G (esetleg irányított) gráf, v_1 gyökérpont.

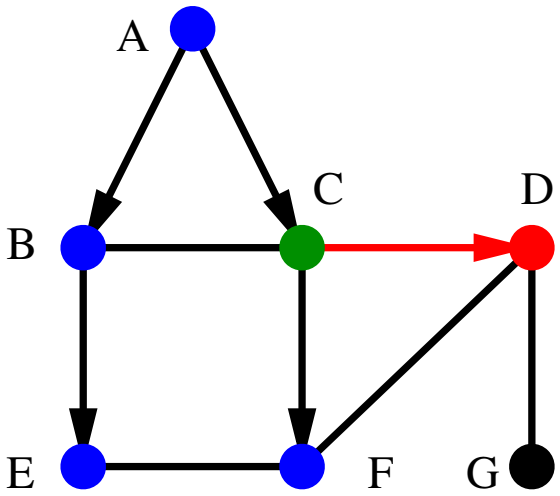
0. Létrehozunk egy kezdetben üres E listát.

1. Bejárjuk a v_1 csúcsot, v_1-et betesszük az E-be. $i := 1$.

2. Az $E(i)$ csúcs összes még be nem járt (ki-)szomszédját bejárjuk valamint ezen csúcsokat az E végére fűzzük. Az $E(i)$ csúcs és a most bejárt szomszédai között futó élekek bevesszük a bejárási fába, irányítatlan esetben úgy, hogy az $E(i)$ csúcs legyen a kezdőpontjuk.

3. Ha az E listában szerepel $i + 1$. csúcs, akkor az i-t növeljük 1-gyel és a 2-es lépésre ugrunk.

4. Output: E ami az elérési sorrend + bejárási fa



Elérési sorrend:
A, B, C, E, F, D
Befejezési sorrend:
A, B, C

Szélességi bejárás, BFS(Breadth-first search)

Input: G (esetleg irányított) gráf, v_1 gyökérpont.

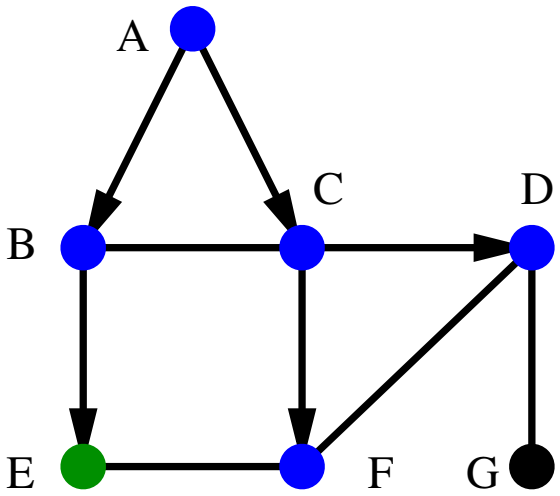
0. Létrehozunk egy kezdetben üres E listát.

1. Bejárjuk a v_1 csúcsot, v_1-et betesszük az E-be. $i := 1$.

2. Az $E(i)$ csúcs összes még be nem járt (ki-)szomszédját bejárjuk valamint ezen csúcsokat az E végére fűzzük. Az $E(i)$ csúcs és a most bejárt szomszédai között futó élekek bevesszük a bejárási fába, irányítatlan esetben úgy, hogy az $E(i)$ csúcs legyen a kezdőpontjuk.

3. Ha az E listában szerepel $i + 1$. csúcs, akkor az i-t növeljük 1-gyel és a 2-es lépésre ugrunk.

4. Output: E ami az elérési sorrend + bejárási fa



Elérési sorrend:
A, B, C, E, F, D
Befejezési sorrend:
A, B, C, E

Szélességi bejárás, BFS(Breadth-first search)

Input: G (esetleg irányított) gráf, v_1 gyökérpont.

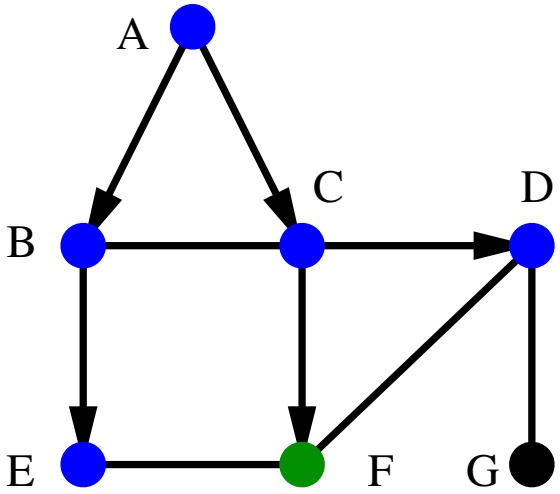
0. Létrehozunk egy kezdetben üres E listát.

1. Bejárjuk a v_1 csúcsot, v_1-et betesszük az E-be. $i := 1$.

2. Az $E(i)$ csúcs összes még be nem járt (ki-)szomszédját bejárjuk valamint ezen csúcsokat az E végére fűzzük. Az $E(i)$ csúcs és a most bejárt szomszédai között futó élekek bevesszük a bejárási fába, irányítatlan esetben úgy, hogy az $E(i)$ csúcs legyen a kezdőpontjuk.

3. Ha az E listában szerepel $i + 1$. csúcs, akkor az i-t növeljük 1-gyel és a 2-es lépésre ugrunk.

4. Output: E ami az elérési sorrend + bejárási fa



Elérési sorrend:
A, B, C, E, F, D
Befejezési sorrend:
A, B, C, E, F

Szélességi bejárás, BFS(Breadth-first search)

Input: G (esetleg irányított) gráf, v_1 gyökérpont.

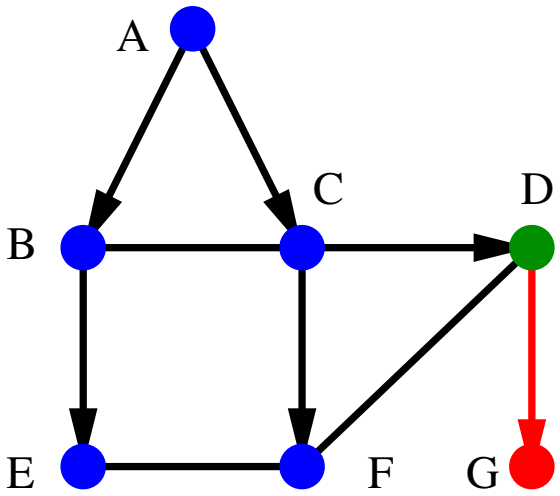
0. Létrehozunk egy kezdetben üres E listát.

1. Bejárjuk a v_1 csúcsot, v_1-et betesszük az E-be. $i := 1$.

2. Az $E(i)$ csúcs összes még be nem járt (ki-)szomszédját bejárjuk valamint ezen csúcsokat az E végére fűzzük. Az $E(i)$ csúcs és a most bejárt szomszédai között futó élekek bevesszük a bejárási fába, irányítatlan esetben úgy, hogy az $E(i)$ csúcs legyen a kezdőpontjuk.

3. Ha az E listában szerepel $i + 1$. csúcs, akkor az i-t növeljük 1-gyel és a 2-es lépésre ugrunk.

4. Output: E ami az elérési sorrend + bejárási fa



Elérési sorrend:
A, B, C, E, F, D, G
Befejezési sorrend:
A, B, C, E, F, D

Szélességi bejárás, BFS(Breadth-first search)

Input: G (esetleg irányított) gráf, v_1 gyökérpont.

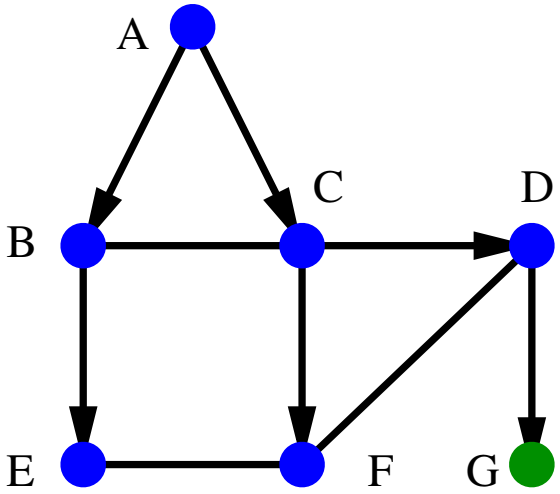
0. Létrehozunk egy kezdetben üres E listát.

1. Bejárjuk a v_1 csúcsot, v_1-et betesszük az E-be. $i := 1$.

2. Az $E(i)$ csúcs összes még be nem járt (ki-)szomszédját bejárjuk valamint ezen csúcsokat az E végére fűzzük. Az $E(i)$ csúcs és a most bejárt szomszédai között futó élekek bevesszük a bejárási fába, irányítatlan esetben úgy, hogy az $E(i)$ csúcs legyen a kezdőpontjuk.

3. Ha az E listában szerepel $i + 1$. csúcs, akkor az i-t növeljük 1-gyel és a 2-es lépésre ugrunk.

4. Output: E ami az elérési sorrend + bejárási fa



Elérési sorrend:
A, B, C, E, F, D, G
Befejezési sorrend:
A, B, C, E, F, D, G

Szélességi bejárás, BFS(Breadth-first search)

Input: G (esetleg irányított) gráf, v_1 gyökérpont.

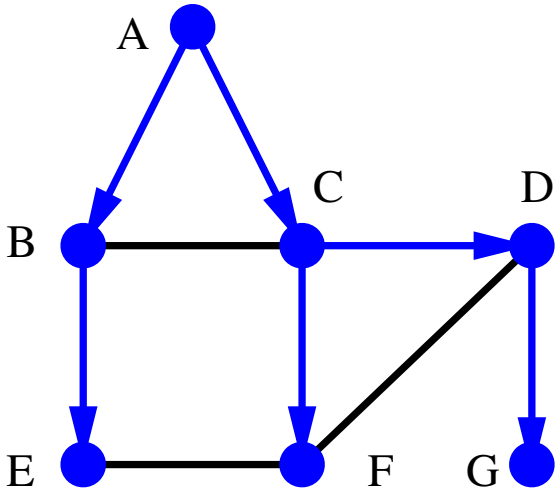
0. Létrehozunk egy kezdetben üres E listát.

1. Bejárjuk a v_1 csúcsot, v_1-et betesszük az E-be. $i := 1$.

2. Az $E(i)$ csúcs összes még be nem járt (ki-)szomszédját bejárjuk valamint ezen csúcsokat az E végére fűzzük. Az $E(i)$ csúcs és a most bejárt szomszédai között futó élekek bevesszük a bejárási fába, irányítatlan esetben úgy, hogy az $E(i)$ csúcs legyen a kezdőpontjuk.

3. Ha az E listában szerepel $i + 1$. csúcs, akkor az i-t növeljük 1-gyel és a 2-es lépésre ugrunk.

4. Output: E ami az elérési sorrend + bejárási fa

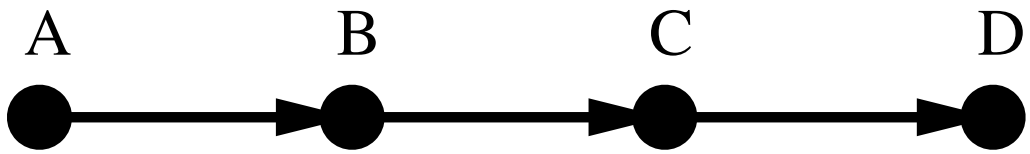


Elérési sorrend:
A, B, C, E, F, D, G
Befejezési sorrend:
A, B, C, E, F, D, G

A BFS nem feltétlen jár be minden csúcsot

Az itt megadott BFS algoritmus nem felétlenül járja be az egész gráfot ha a gráf irányított, illetve biztosan nem járja be ha nem összefüggő.

Példa: Legyen C a kezdő csúcs. Ekkor az algoritmus végén $E = (C, D)$ és az A és B csúcsok nem lettek bejárva.

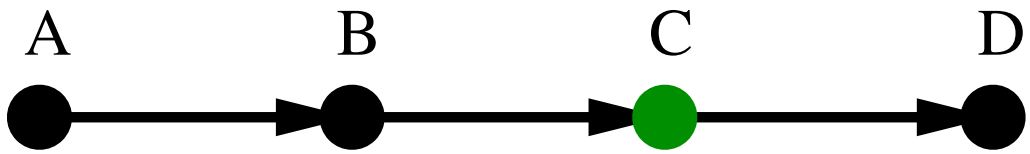


Ha a BFS algoritmust egy v csúcsból indítom egy irányítatlan gráf esetén, akkor pont a v csúcsot tartalmazó összefüggő komponensben lévő csúcsokat fogja bejárni. Emiatt a BFS segítségével meg lehet találni az összefüggő komponenseket.

A BFS nem feltétlen jár be minden csúcsot

Az itt megadott BFS algoritmus nem felétlenül járja be az egész gráfot ha a gráf irányított, illetve biztosan nem járja be ha nem összefüggő.

Példa: Legyen C a kezdő csúcs. Ekkor az algoritmus végén $E = (C, D)$ és az A és B csúcsok nem lettek bejárva.

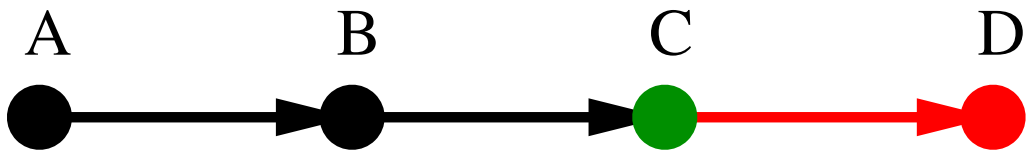


Ha a BFS algoritmust egy v csúcsból indítom egy irányítatlan gráf esetén, akkor pont a v csúcsot tartalmazó összefüggő komponensben lévő csúcsokat fogja bejárni. Emiatt a BFS segítségével meg lehet találni az összefüggő komponenseket.

A BFS nem feltétlen jár be minden csúcsot

Az itt megadott BFS algoritmus nem felétlenül járja be az egész gráfot ha a gráf irányított, illetve biztosan nem járja be ha nem összefüggő.

Példa: Legyen C a kezdő csúcs. Ekkor az algoritmus végén $E = (C, D)$ és az A és B csúcsok nem lettek bejárva.

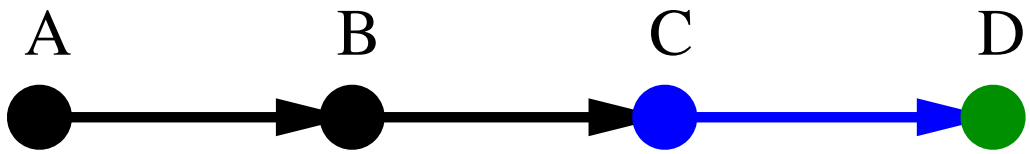


Ha a BFS algoritmust egy v csúcsból indítom egy irányítatlan gráf esetén, akkor pont a v csúcsot tartalmazó összefüggő komponensben lévő csúcsokat fogja bejárni. Emiatt a BFS segítségével meg lehet találni az összefüggő komponenseket.

A BFS nem feltétlen jár be minden csúcsot

Az itt megadott BFS algoritmus nem felétlenül járja be az egész gráfot ha a gráf irányított, illetve biztosan nem járja be ha nem összefüggő.

Példa: Legyen C a kezdő csúcs. Ekkor az algoritmus végén $E = (C, D)$ és az A és B csúcsok nem lettek bejárva.

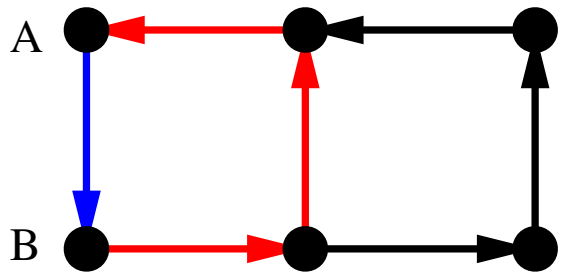


Ha a BFS algoritmust egy v csúcsból indítom egy irányítatlan gráf esetén, akkor pont a v csúcsot tartalmazó összefüggő komponensben lévő csúcsokat fogja bejárni. Emiatt a BFS segítségével meg lehet találni az összefüggő komponenseket.

Legrövidebb utak fája

Definíció: Egy (irányított) út **hossza** az őt alkotó élek száma. A G (irányított) gráfban található u és v csúcsok **távolságán** a legrövidebb u és v közötti (irányított) út hosszát értjük. Ezt a mennyiséget általában $d(u, v)$-vel jelöljük.

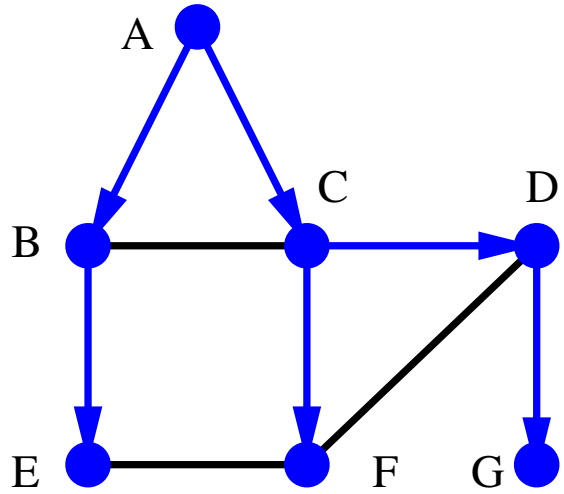
Megjegyzés: Irányított gráf esetén $d(u, v)$ és $d(v, u)$ tipikusan nem egyenlő!
$d(A, B) = 1 \neq 3 = d(B, A)$



Állítás

Ha a G gráf egy BFS fájában irányított út vezet u-ból v-be akkor ez az irányított út egyben egy legrövidebb út u és v között.

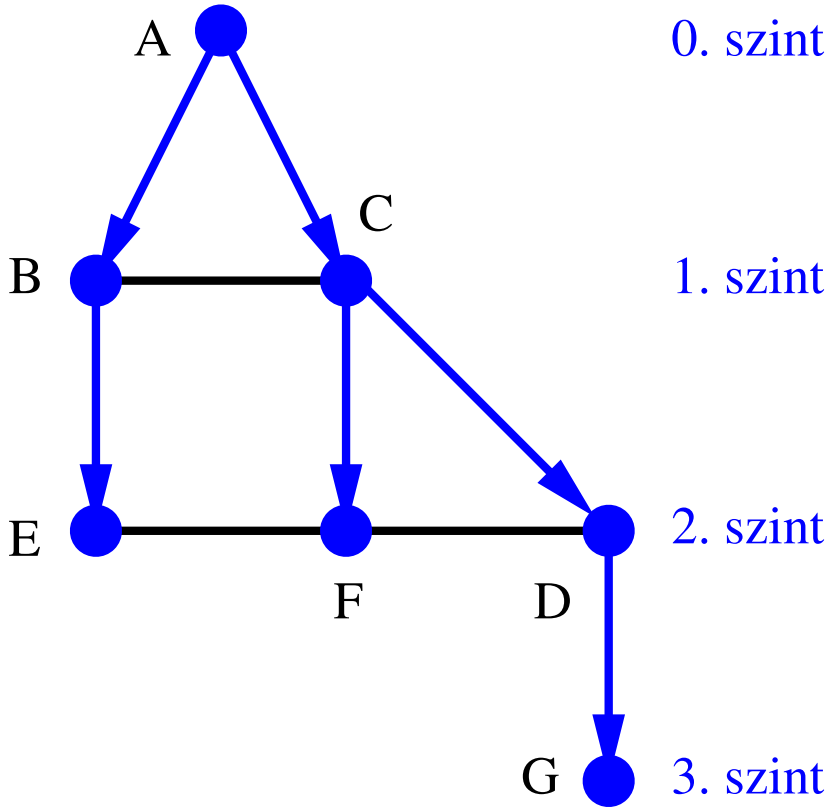
Következmény: Egy szélességi kereséssel meghatározhatóak a gyökérpont és bármely másik csúcs közötti távolságok.
Például az A és F távolsága 2.



Emeletekre bontás a BFS-sel

A BFS algoritmussal emeletekre bonthatjuk a gráfot:

- 0. szinten a gyökér van
- 1. szinten a gyökértől 1 faélnyire lévő csúcsok
- 2. szinten a gyökértől 2 faélnyire lévő csúcsok
- i. szinten a gyökértől i db faélnyire lévő csúcsok
- Van egy ∞ szint ahol a gyökérből faéleken el nem érhető csúcsok találhatóak.



Miért ad a BFS legrövidebb utakat?

Állítás: Ha egy u csúcs a gyökérből elérhető úton/irányított úton, akkor faéleken keresztül is elérhető.

Bizonyítás: Ezen út/irányított út mentén található csúcsokat a BFS bejárja.

Állítás: Egy irányítatlan gráf minden éle csak azonos és szomszédos szint között fut.

Bizonyítás: Legyen u és v olyan csúcsai a gráfnak, hogy az u legalább 2-vel kisebb sorszámú szinten található. Ha a gráfban lenne $\{u, v\}$ él, akkor a bejárás ezt az élet faélnek választotta volna és a v csúcs az u-nál egyel nagyobb sorszámú szintre került volna.

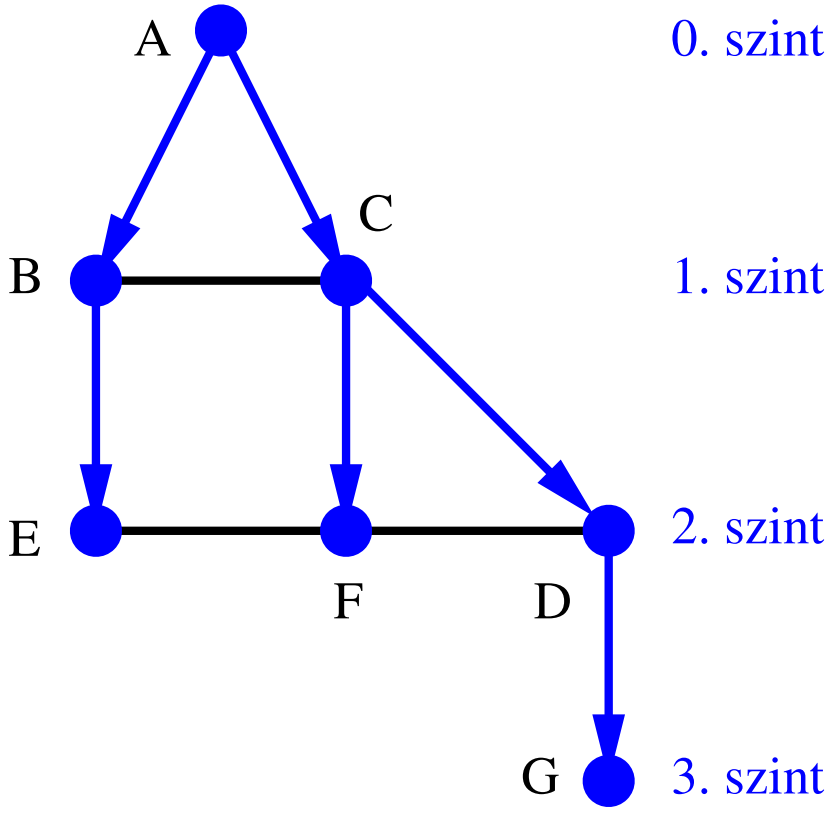
Állítás: Egy irányított gráf minden ir. élének a végpontja legfeljebb egyel lehet nagyobb sorszámú szinten mint a kezdőpontja.

Következmény: Egy csúcsnak a gyökértől vett távolsága megegyezik a szintszámával, hiszen ennyi faélen keresztül elérhető a gyökérből, egy rövidebb út pedig nem szomszédos szintek közötti élet tartalmazna ami nincs.

Emeletekre bontás valójában

A BFS algoritmussal emeletekre bonthatjuk a gráfot:

- 0. szinten a gyökér van
- 1. szinten a gyökértől 1 távol lévő csúcsok
- 2. szinten a gyökértől 2 távol lévő csúcsok
- i. szinten a gyökértől i távol lévő csúcsok
- Van egy ∞ szint ahol a gyökérből úton/irányított úton el nem érhető csúcsok találhatóak.



A BFS néhány tulajdonsága

Állítás

A BFS algoritmus lépésszáma $n + e$-vel arányos, ahol n a gráf csúcsszáma e pedig az élszáma.

Bizonyítás: Minden csúcsot legfeljebb egyszer írunk fel a listába, ezért a lista karbantartása a csúcsszámmal arányos lépést kíván. Minden bejárt csúcsnak megvizsgáljuk a szomszédait, hogy bejártak-e, de ezekből a vizsgálatokból legfeljebb kétszer annyi van mint él.□

A BFS néhány tulajdonsága

Állítás
A BFS algoritmus lépésszáma $n + e$-vel arányos, ahol n a gráf csúcsszáma e pedig az élszáma.

Bizonyítás: Minden csúcsot legfeljebb egyszer írunk fel a listába, ezért a lista karbantartása a csúcsszámmal arányos lépést kíván. Minden bejárt csúcsnak megvizsgáljuk a szomszédait, hogy bejártak-e, de ezekből a vizsgálatokból legfeljebb kétszer annyi van mint él.□

Állítás:
A BFS bejárásra nézve nincsenek előre élek.

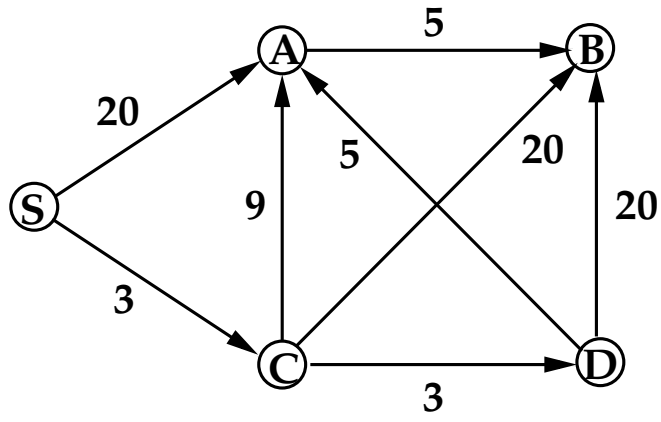
Feladat

Adott egy G irányított vagy irányítatlan gráf, illetve a G élhalmazán értelmezett l hosszfüggvény mely megmondja, hogy egy él milyen hosszú.

Definíció: Egy **út hossza** az utat alkotó élek hosszainak az összege. Az u és v csúcsok **távolságán** az u és v között található legrövidebb (irányított gráf esetén irányított) út hosszát értjük és ezt a mennyiséget $dist(u, v)$-vel jelöljük.

Példa:

$dist(S, D) = 6$
$dist(D, S) = \infty$
$dist(S, A) = 11$



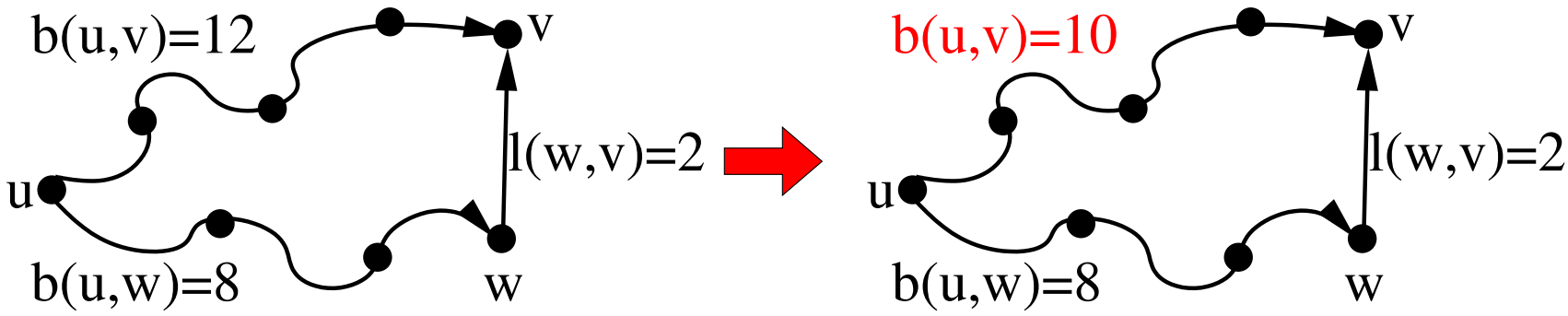
Adjunk eljárást amely tetszőleges két csúcs távolságát meghatározza és meg is ad egy legrövidebb utat közöttük!

Legrövidebb utakat kereső algoritmusok ötlete

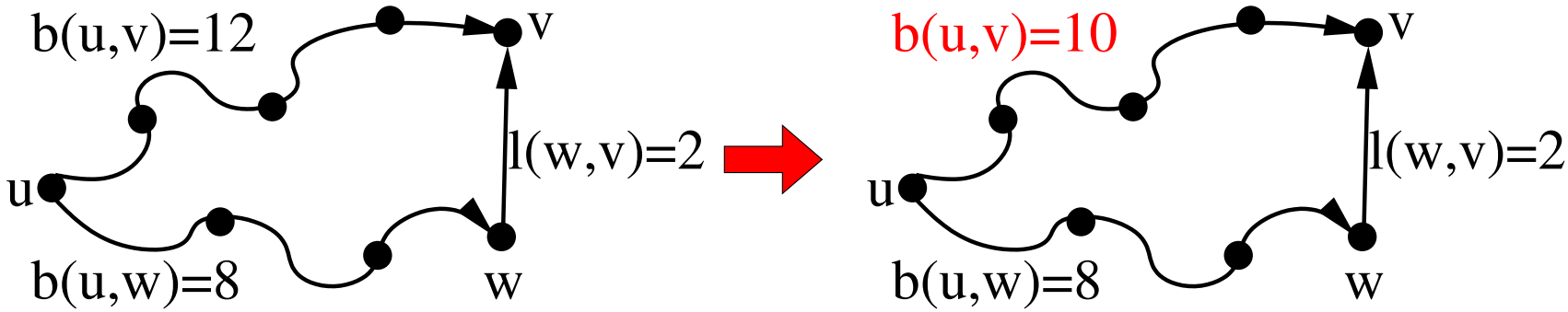
Definíció: A G irányított vagy irányítatlan gráfban lévő $dist(u, v)$ távolságfüggvényre nézve egy **felső becslés** a $b(u, v)$ ha minden u, v csúcspárra $dist(u, v) \leq b(u, v)$.
Ha $dist(u, v) = b(u, v)$, akkor a $b(u, v)$ becslést **pontosnak** nevezzük.

Él menti javítás

Ha $b(u, v) > b(u, w) + l(w, v)$ akkor cseréljük le $b(u, v)$ értékét $b(u, w) + l(w, v)$-re! Ebben az esetben azt mondjuk, hogy a wv él mentén javítottuk a felső becslésünket!



Él menti javítás haszna



Állítások:

1. Ha egy felső becslésre alkalmazzuk az él menti javítást akkor felső becslést kapunk.
2. Ha egy u és v között futó legrövidebb $u = v_1, v_2, \ldots v_n = v$ út mentén élei mentén sorban javítjuk a $b(v_1, v_2), b(v_1, v_3), \ldots, b(v_1, v_n)$ felső becsléseket, akkor a végén $d(u, v) = b(u, v)$.
3. Ha semelyik él mentén sem tudunk egy felső becslést javítani, akkor az a felső becslés megegyezik a távolságfüggvénnyel.

Állítások haszna: Az él menti javítást megfelelően sokszor ismételve fogjuk kiszámolni a távolságokat.

Dijkstra algoritmusa

Input: $G = (V, E)$ (irányított/irányítatlan) gráf, egy $l : E \to \mathbb{R}^+ \cup 0$ hosszfüggvény és egy $s \in V$ csúcs.

0. KÉSZ:= $\{s\}$, $d(s) := 0$, $d(v) := \infty$ minden s-től különböző v csúcsra, és legyen $u := s$.

1. Az u csúcsból a KÉSZ halmazon kívüli csúcsokba futó élek mentén próbáljuk javítani a $d()$ felső becslést, azaz ha $v \notin$KÉSZ, $uv \in E$ és $d(v) > d(u) + l(u, v)$, akkor $d(v) := d(u) + l(u, v)$ valamint $T(v) := u$.

2. Legyen u a KÉSZ halmazon kívüli csúcsok közül az amelyikre a $d(u)$ érték a legkisebb. Tegyük be az u-t a KÉSZ halmazba.

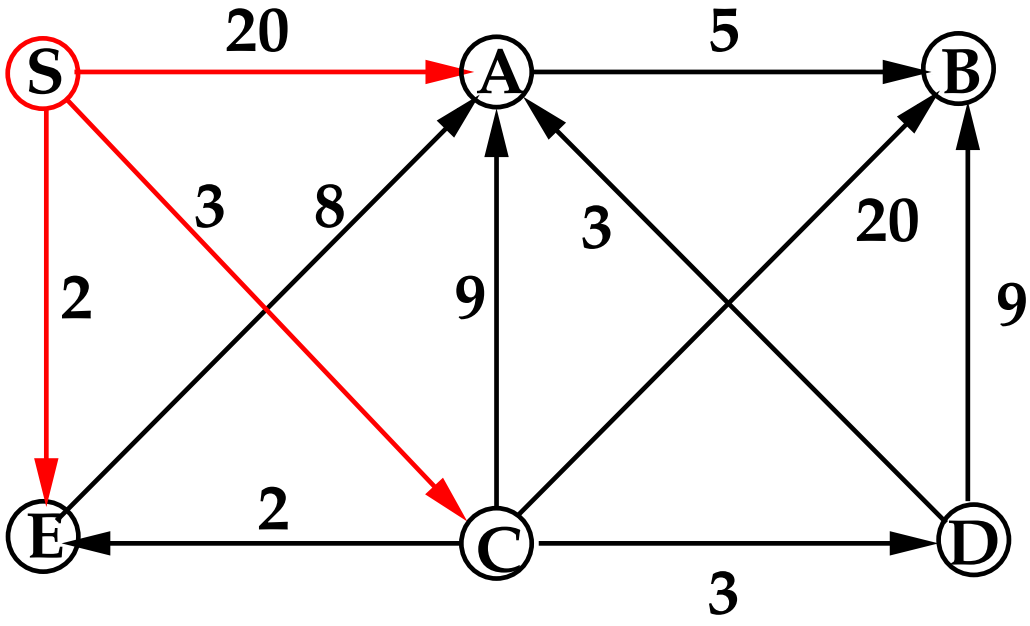
3. Ha minden csúcs a KÉSZ halmazban van akkor STOP, különben folytassuk az 1-es lépéssel.

Output: $d(v) = dist(s, v)$ értékek és T tömb melyből a legröbidebb utak kiolvashatóak.

A dijkstra algoritmus magyarázata (nem bizonyítás)

Az algoritmus minden v csúcsra meghatározza a $dist(s, v)$ távolságot, ezt a $d(v)$ számolásával éri el, ami sohasem nő, és minden egyes lépésben felső becslése a $dist(s, v)$-nek.
Egy KÉSZ nevezetű halmazba helyezi egyesével a csúcsokat.
A KÉSZ halmazban lévő csúcsokra a felső becslés már pontos lesz, azaz ha $v \in$KÉSZ akkor $d(v) = dist(s, v)$. Kezdésnek csak az s van a kész halmazban.
Az algoritmus futása során $T(v)$ azt mondja meg, hogy egy s és v közötti $d(v)$ hosszú út esetén melyik csúcs előzi meg v-t.
Emiatt a végső $T(v)$ megmondja, hogy egy s és v közötti legrövidebb úton melyik csúcs előzi meg v-t.
Ezért ha egy s és v közötti legrövidebb utat szeretnénk megkapni, akkor ennek a fordítottját adja a $v, T(v), T(T(v)), \ldots, T(T(\ldots T(v)\ldots)) = s$ sorozat, amit csak meg kell fordítani.

Dijkstra algoritmus példa

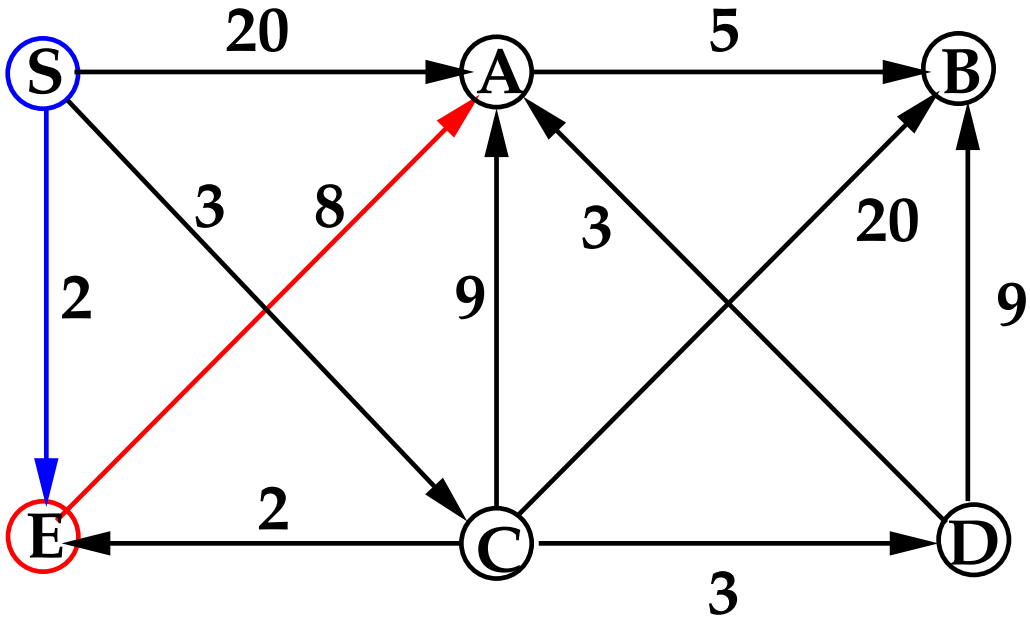


KÉSZ={S}

d()						
S	0	0				
A	∞	20				
B	∞	∞				
C	∞	3				
D	∞	∞				
E	∞	2				

T()					
S					
A	S				
B					
C	S				
D					
E	S				

Dijkstra algoritmus példa

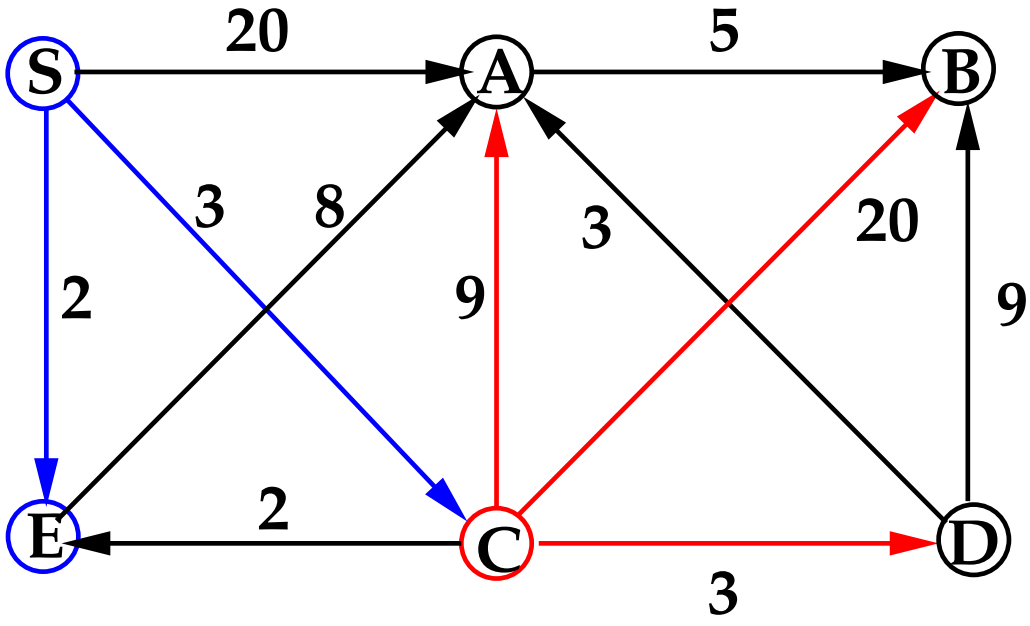


KÉSZ={S,E}

d()						
S	0	0	0			
A	∞	20	10			
B	∞	∞	∞			
C	∞	3	3			
D	∞	∞	∞			
E	∞	2	2			

T()					
S					
A	S	E			
B					
C	S	S			
D					
E	S	S			

Dijkstra algoritmus példa

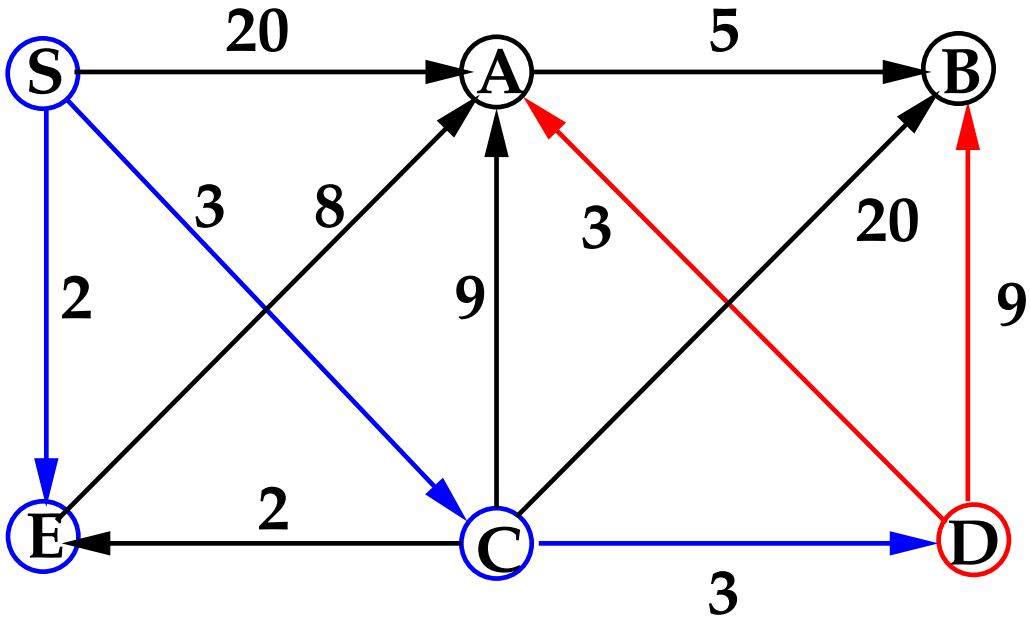


KÉSZ={S,E,C}

d()						
S	0	0	0	0		
A	∞	20	10	10		
B	∞	∞	∞	23		
C	∞	3	3	3		
D	∞	∞	∞	6		
E	∞	2	2	2		

T()					
S					
A	S	E	E		
B			C		
C	S	S	S		
D			C		
E	S	S	S		

Dijkstra algoritmus példa

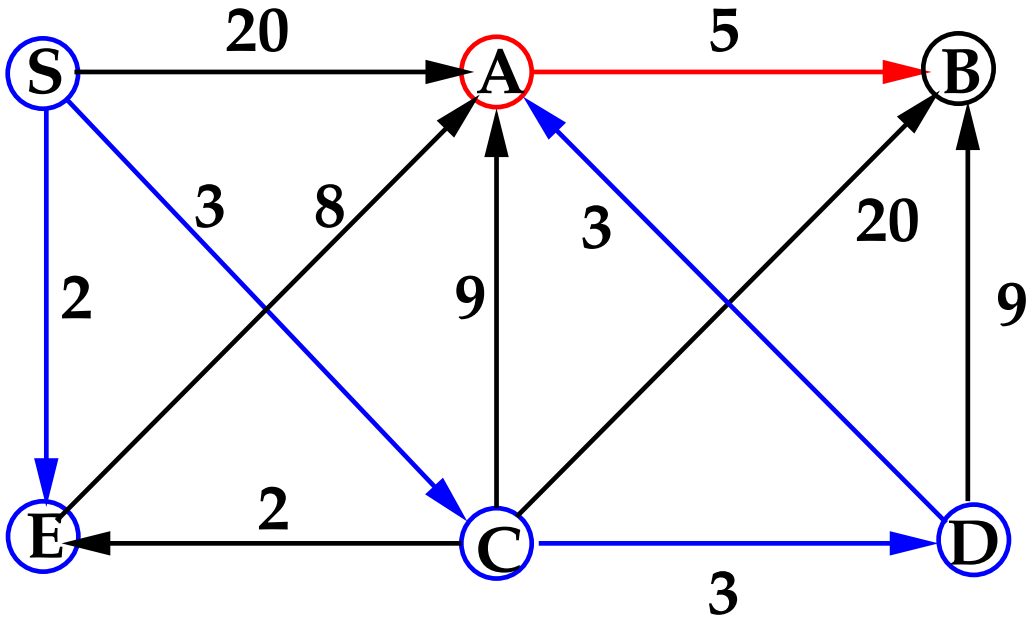


KÉSZ={S,E,C,D}

d()						
S	0	0	0	0	0	
A	∞	20	10	10	9	
B	∞	∞	∞	23	15	
C	∞	3	3	3	3	
D	∞	∞	∞	6	6	
E	∞	2	2	2	2	

T()					
S					
A	S	E	E	D	
B			C	D	
C	S	S	S	S	
D			C	C	
E	S	S	S	S	

Dijkstra algoritmus példa

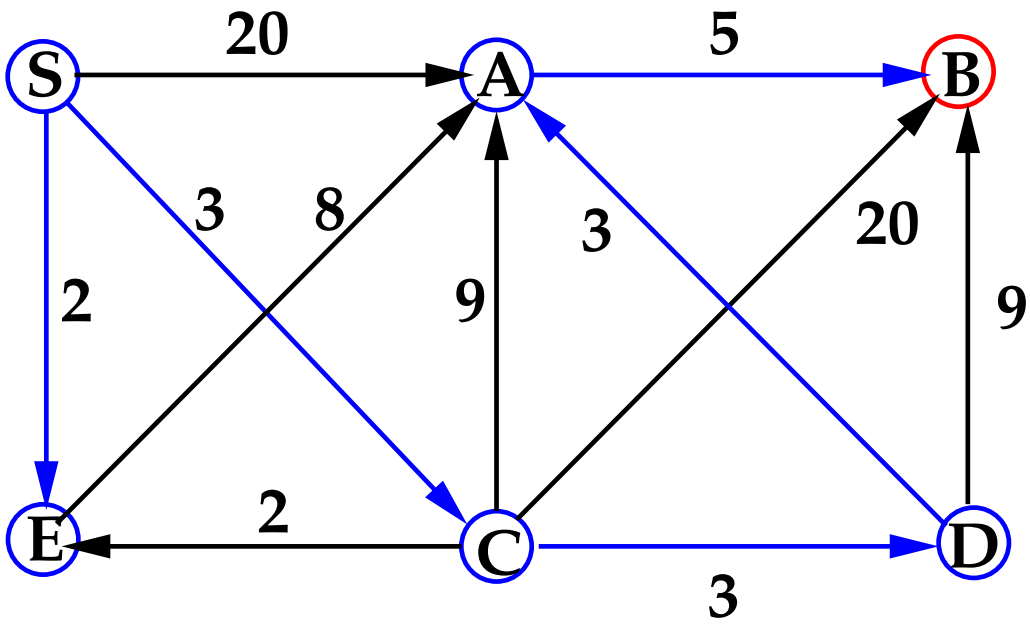


KÉSZ={S,E,C,D,A}

d()						
S	0	0	0	0	0	0
A	∞	20	10	10	9	9
B	∞	∞	∞	23	15	14
C	∞	3	3	3	3	3
D	∞	∞	∞	6	6	6
E	∞	2	2	2	2	2

T()					
S					
A	S	E	E	D	D
B			C	D	A
C	S	S	S	S	S
D			C	C	C
E	S	S	S	S	S

Dijkstra algoritmus példa



KÉSZ={S,E,C,D,A,B}
dist(S,B)=14 és a legrövidebb út *S* és *B* között SCDAB.

d()						
S	0	0	0	0	0	0
A	∞	20	10	10	9	9
B	∞	∞	∞	23	15	14
C	∞	3	3	3	3	3
D	∞	∞	∞	6	6	6
E	∞	2	2	2	2	2

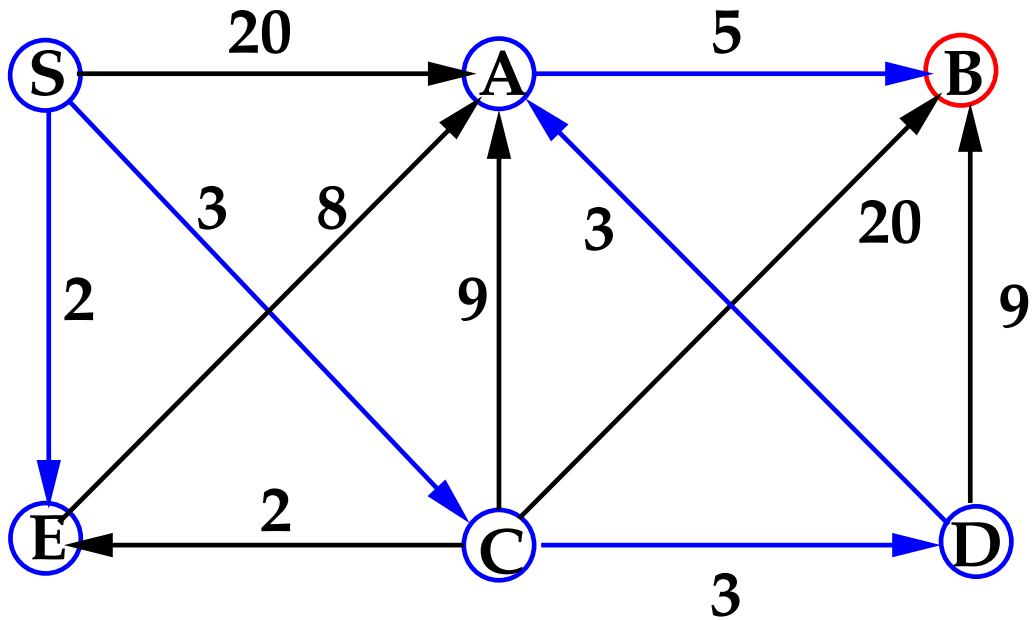
T()					
S					
A	S	E	E	D	D
B			C	D	A
C	S	S	S	S	S
D			C	C	C
E	S	S	S	S	S

Dijkstra jóságának bizonyítása

Azt bizonyítjuk be, hogy az algoritmus futása során a KÉSZ halmazban szereplő csúcsok felső becslése már pontos. Ehhez indukciót használunk a KÉSZ halmaz méretére.
Legyen u a legkisebb felső becsléssel rendelkező nem KÉSZbeli csúcs. Indukció miatt a KÉSZben lévő csúcsok becslése pontos. Minden v KÉSZbeli csúcsból kiinduló él mentén javítottunk már úgy, hogy a v becslése pontos volt, ezért ezen élek menti javítás már nem csökkenti a felső becslést. Ezért nem KÉSZ beli csúcsok között vezető élek mentén tudunk javítani. Ezek hossza nemnegatív, emiatt $d(u)$ alá egyik KÉSZen kívüli csúcs becslése sem csökkenhet. Emiatt u becslése nem csökkenthető tovább javításokkal. Így $d(u)$ szükségszerűen pontos. Szabály szerint u kerül a KÉSZ halmazba következőnek, így készen vagyunk.

Legrövidebb utak fája Dijkstra esetén

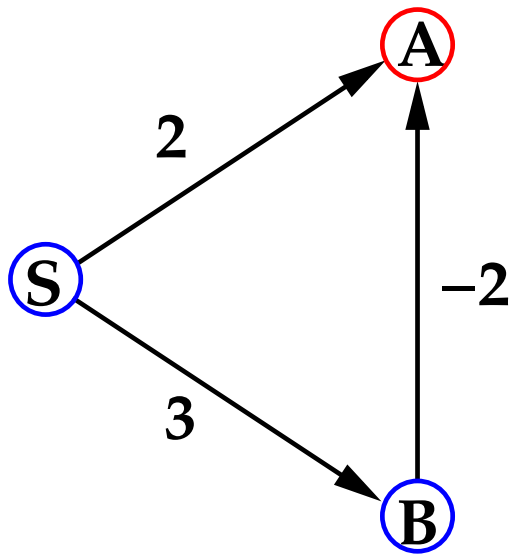
Minden egyes v csúcs esetén jelöljük meg azt az élet, amely menti javítás utoljára csökkentette a v s-től vett távolságát. Ez a v csúcs esetén pontosan a $(T(v), v)$ élet jelenti. Így ha G összefüggő, akkor G-nek egy olyan feszítőfáját kapjuk, amely tartalmaz s-ből minden más csúcsba egy legrövidebb utat.



Negatív élhosszok

Mi történik, hogy ha vannak negatív hosszú élek?
Dijkstra algoritmusa nem működik, rossz eredményt ad.

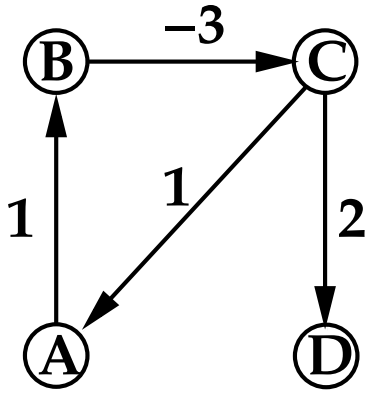
Példa:



A Dijkstra az SA él menti javítás után az A csúcsot a KÉSZ halmazba teszi $d(A) = 2$-vel, míg az SBA út 1 hosszú.

Probléma negatív élhosszokkal

Az ismert, gyakorlatban használható legrövidebb utat kereső algoritmusok valójában nem utat keresnek hanem legrövidebb élsorozatot. Irányítatlan esetben egy negatív élen oda-vissza mászkálnak. Irányított gráf esetén ha van negatív összhosszúságú irányított kör akkor ezen körbemenve tetszőlegesen sokszor lehet csökkenteni az élsorozat hosszát.



Definíció: Egy irányított gráf élhalmazán értelmezett hosszfüggvény **konzervatív** ha a gráf nem tartalmaz negatív összhosszú irányított kört.

Innentől csak konzervatív élhosszfüggvényekkel foglalkozunk.

Ford algoritmusa

Input: Egy $G = (V, E)$ irányított vagy irányítatlan gráf, egy az élhalmazán értelmezett konzervatív hosszfüggvény: l és egy $s \in V$ csúcs.

0. Rögzítjük az éleknek egy tetszőleges sorrendjét, legyen ez $e_1, e_2, \ldots, e_{|E|}$. Kezdetben $d(s) := 0$ és $d(v) =: \infty$ minden s-től különböző v csúcs esetén. Legyen $i := 1$.

1. A rögzített sorrendben haladva próbálunk meg javítani minden él mentén a $d()$ felső becslésen.

2. Ha $i = |V|$, akkor STOP, különben i-t megnöveljük eggyel ($i := i + 1$) és az 1-es lépésre ugrunk.

Output: Az s-től mért távolságok: $d(v)$ minden v csúcsra.

Ford algoritmusa

Input: Egy $G = (V, E)$ irányított vagy irányítatlan gráf, egy az élhalmazán értelmezett konzervatív hosszfüggvény: l és egy $s \in V$ csúcs.

0. Rögzítjük az éleknek egy tetszőleges sorrendjét, legyen ez $e_1, e_2, \ldots, e_{|E|}$. Kezdetben $d(s) := 0$ és $d(v) =: \infty$ minden s-től különböző v csúcs esetén. Legyen $i := 1$.

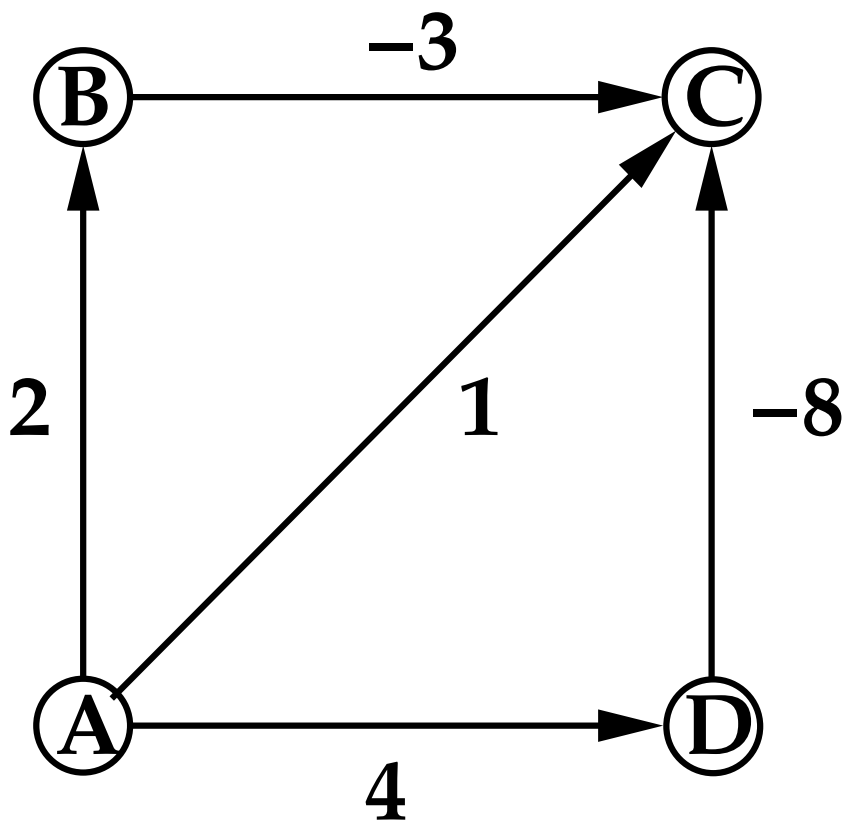
1. A rögzített sorrendben haladva próbálunk meg javítani minden él mentén a $d()$ felső becslésen.

2. Ha $i = |V|$, akkor STOP, különben i-t megnöveljük eggyel ($i := i + 1$) és az 1-es lépésre ugrunk.

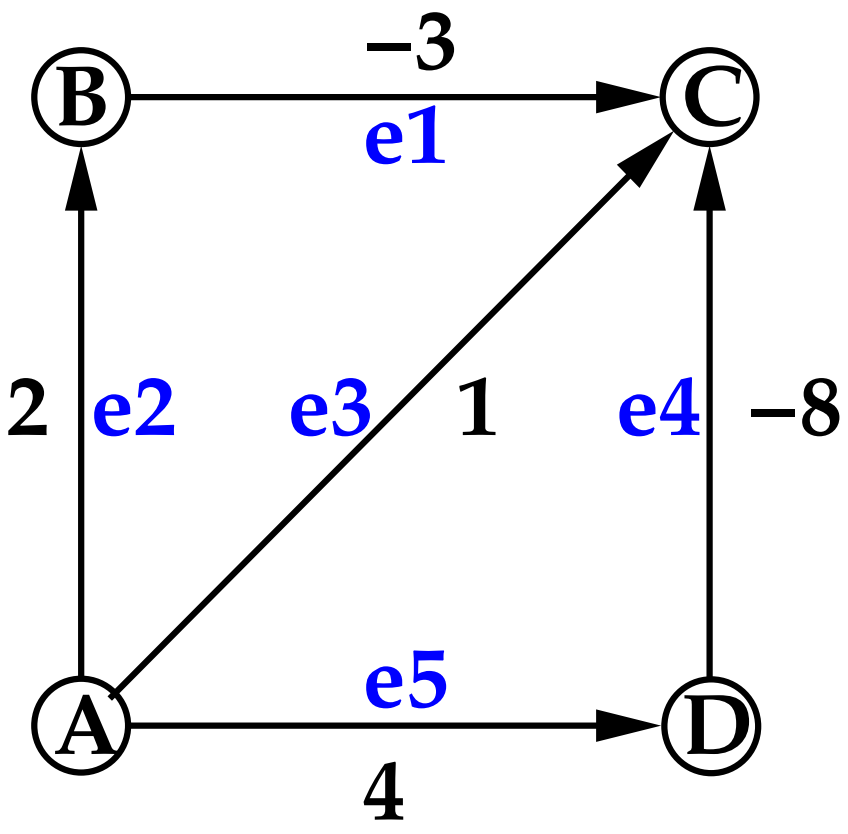
Output: Az s-től mért távolságok: $d(v)$ minden v csúcsra.

Megjegyzés: A legrövidebb utak meghatározásához ugyan úgy járhatunk el mint a Dijkstra esetében, azaz ha a $d(v)$ felső becslés javul az $\vec{uv}$ él menti javítás során, akkor $T(v)$-ben eltároljuk u-t. Végül pedig a legrövidebb út fordítottját kiolvassuk a T tömbből.

Ford algoritmusára példa

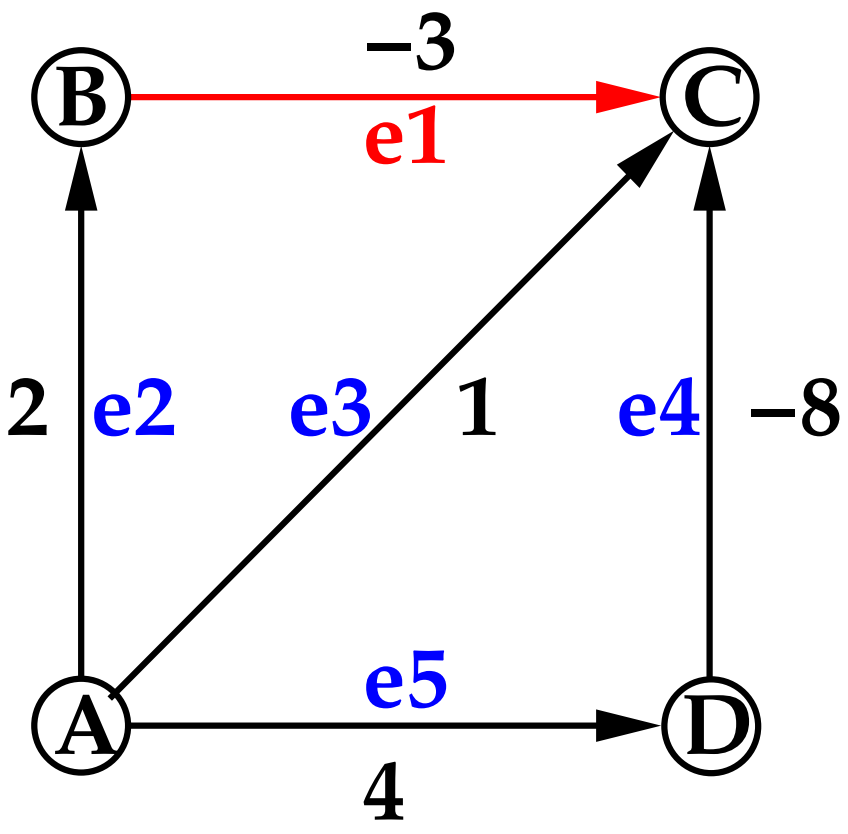


Ford algoritmusára példa



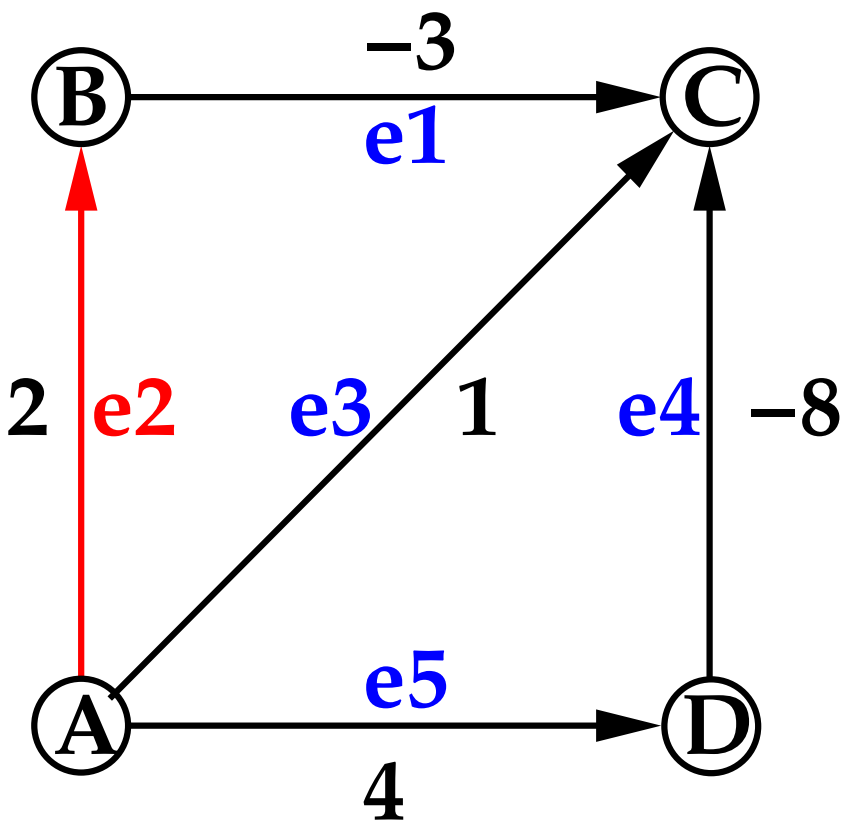
d()					
A	0				
B	∞				
C	∞				
D	∞				

Ford algoritmusára példa



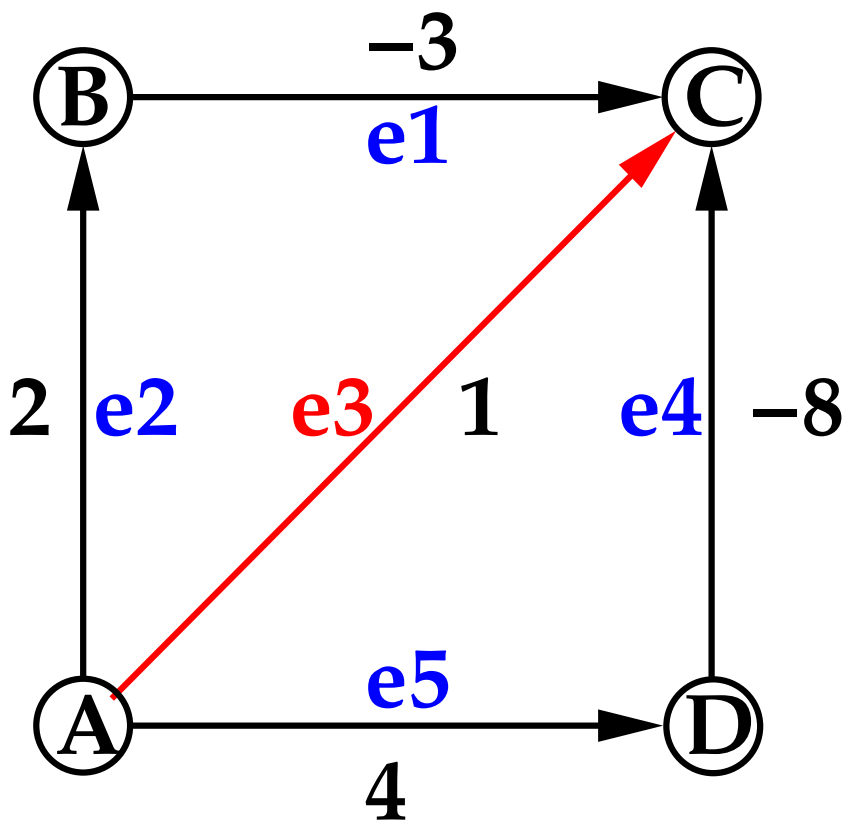
d()					
A	0				
B	∞				
C	∞				
D	∞				

Ford algoritmusára példa



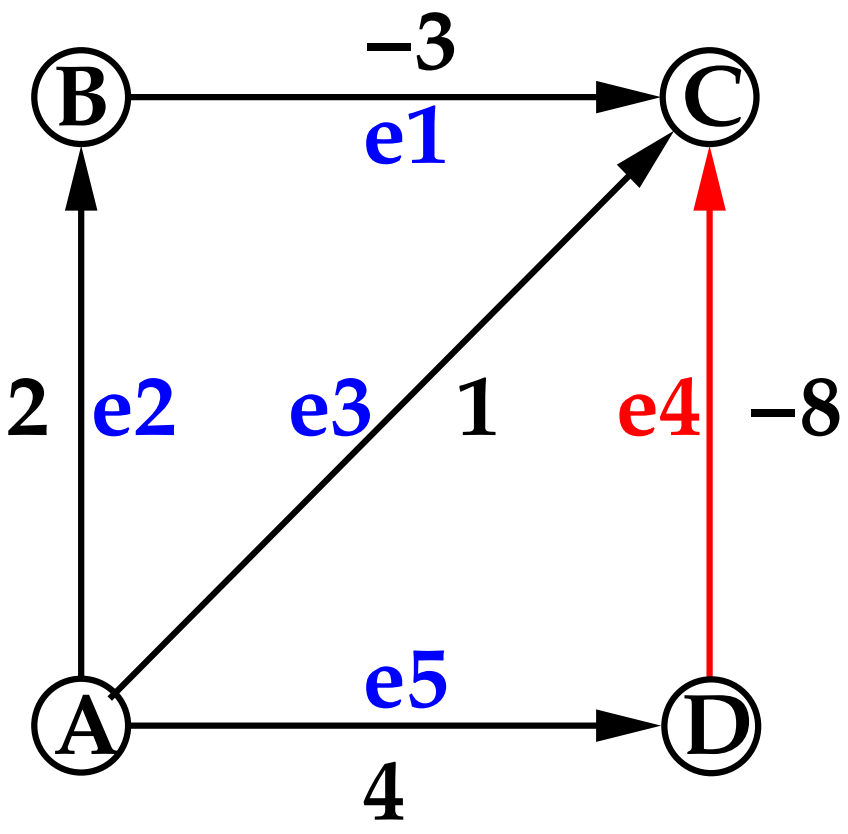
d()					
A	0				
B	∞	2			
C	∞				
D	∞				

Ford algoritmusára példa



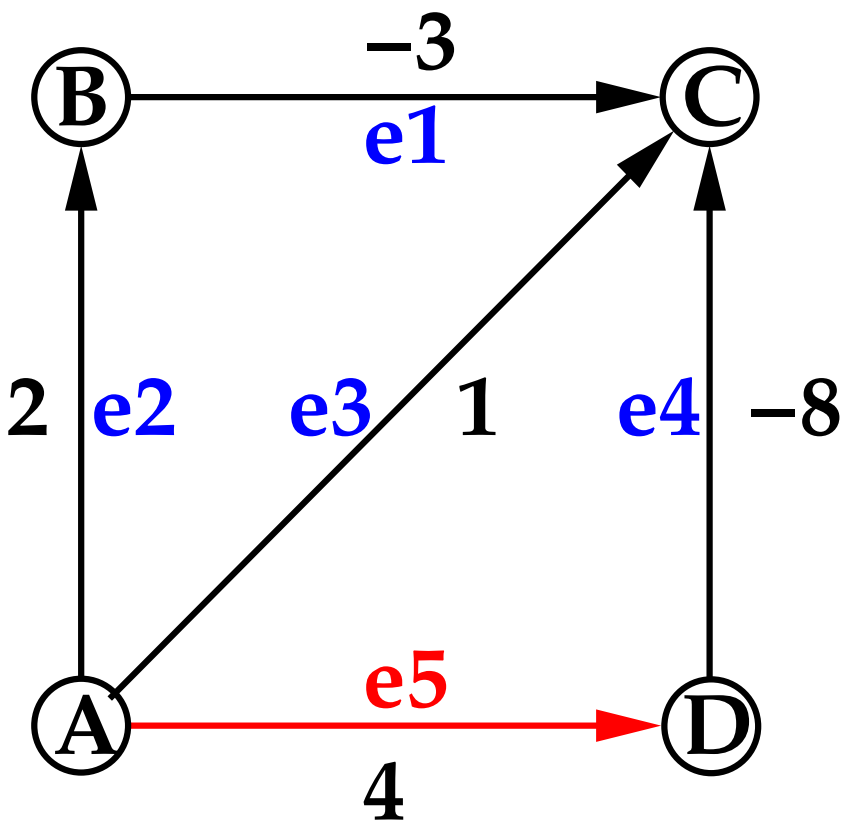
d()					
A	0				
B	∞	2			
C	∞	1			
D	∞				

Ford algoritmusára példa



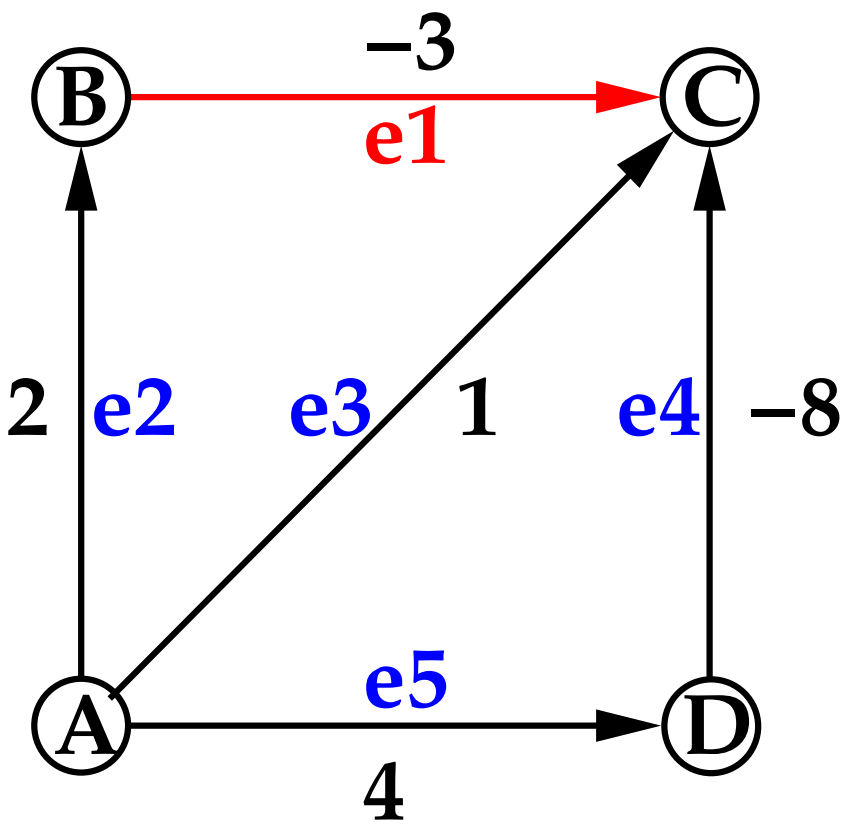
d()					
A	0				
B	∞	2			
C	∞	1			
D	∞				

Ford algoritmusára példa



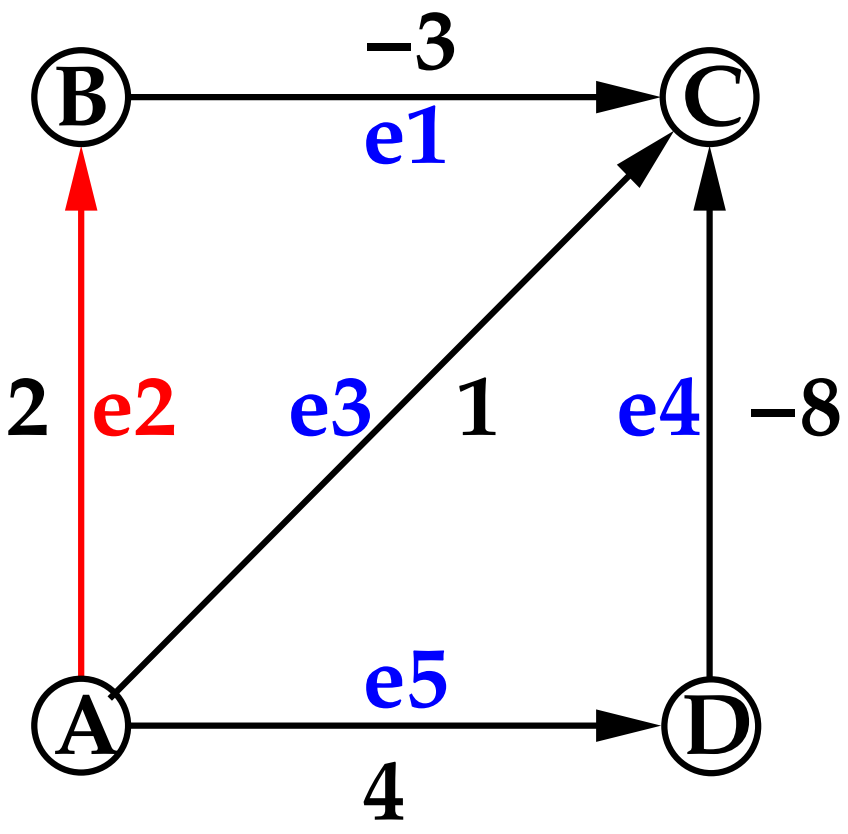
d()					
A	0				
B	∞	2			
C	∞	1			
D	∞	4			

Ford algoritmusára példa



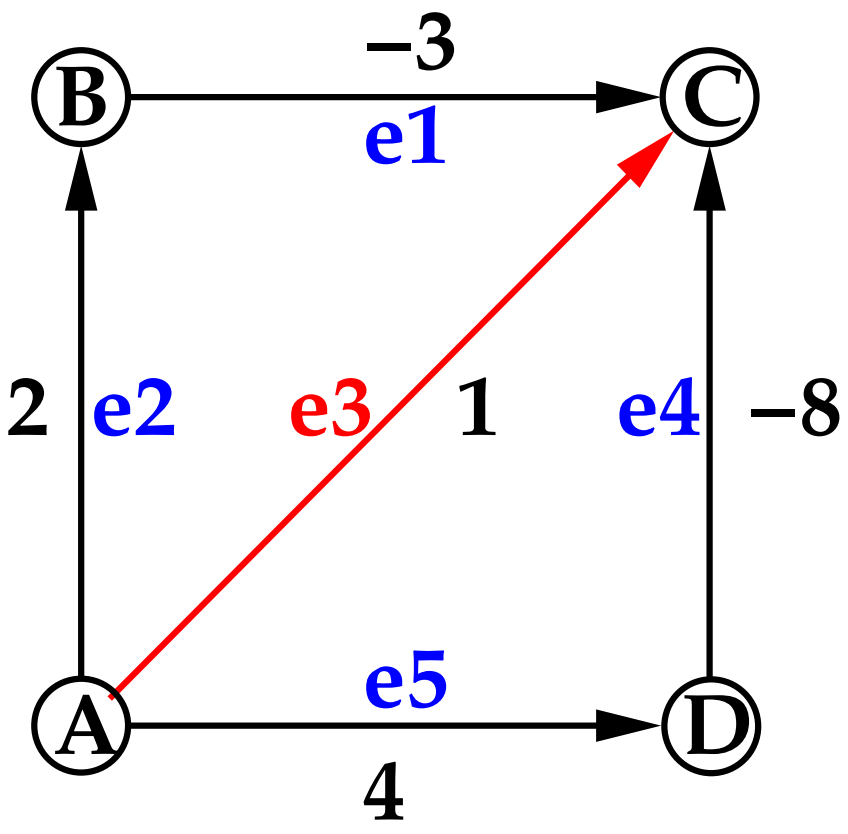
d()					
A	0	0			
B	∞	2			
C	∞	1	-1		
D	∞	4			

Ford algoritmusára példa



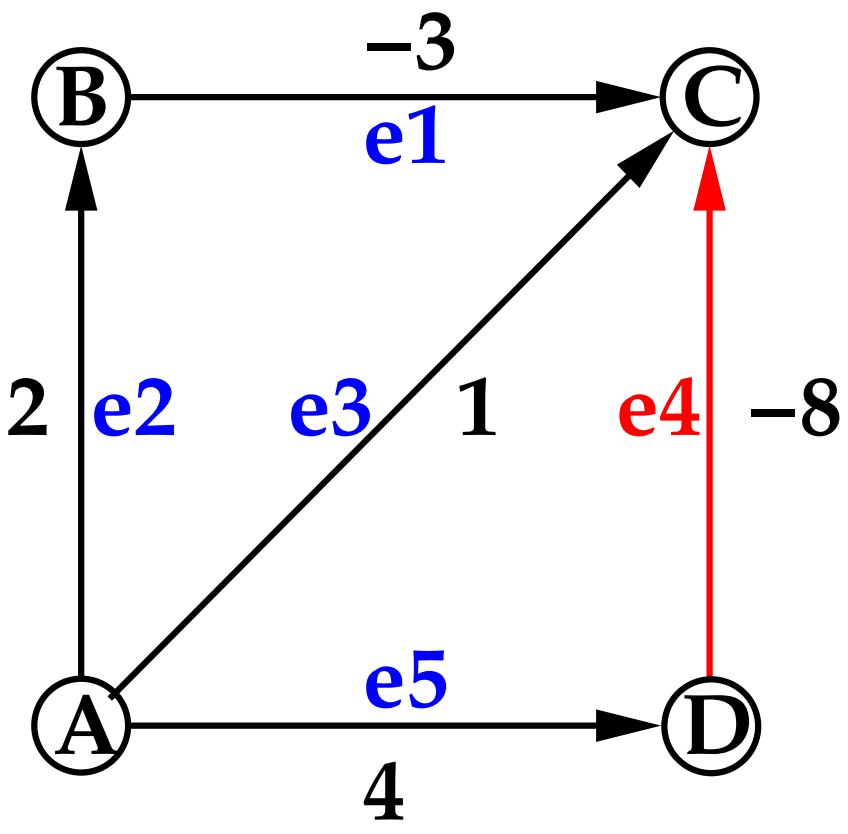
d()					
A	0	0			
B	∞	2			
C	∞	1	-1		
D	∞	4			

Ford algoritmusára példa



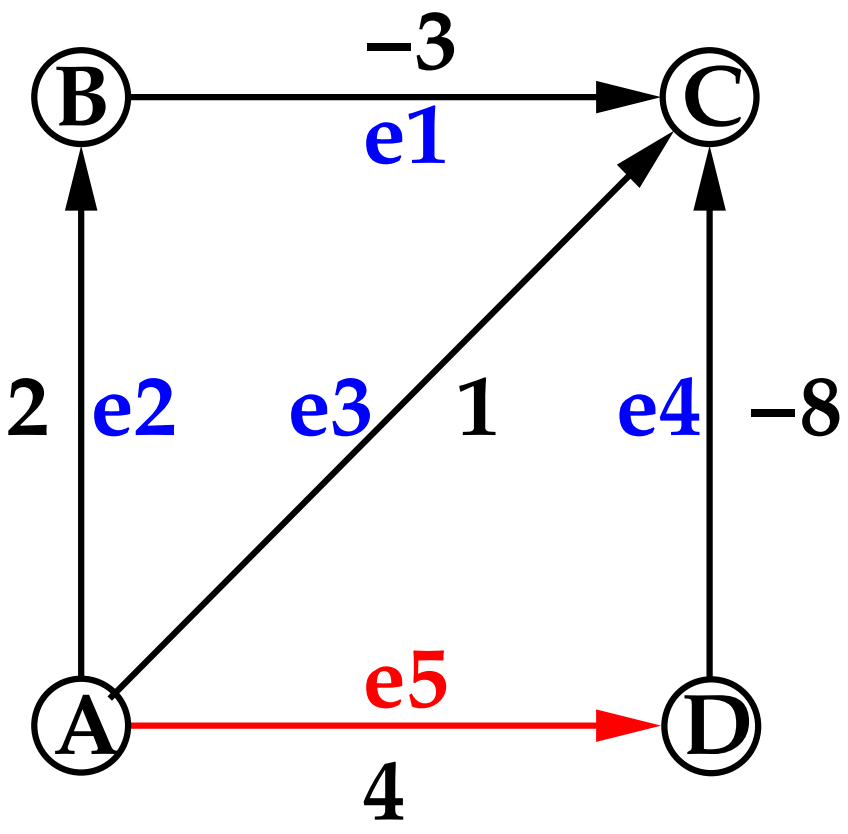
d()					
A	0	0			
B	∞	2			
C	∞	1	-1		
D	∞	4			

Ford algoritmusára példa



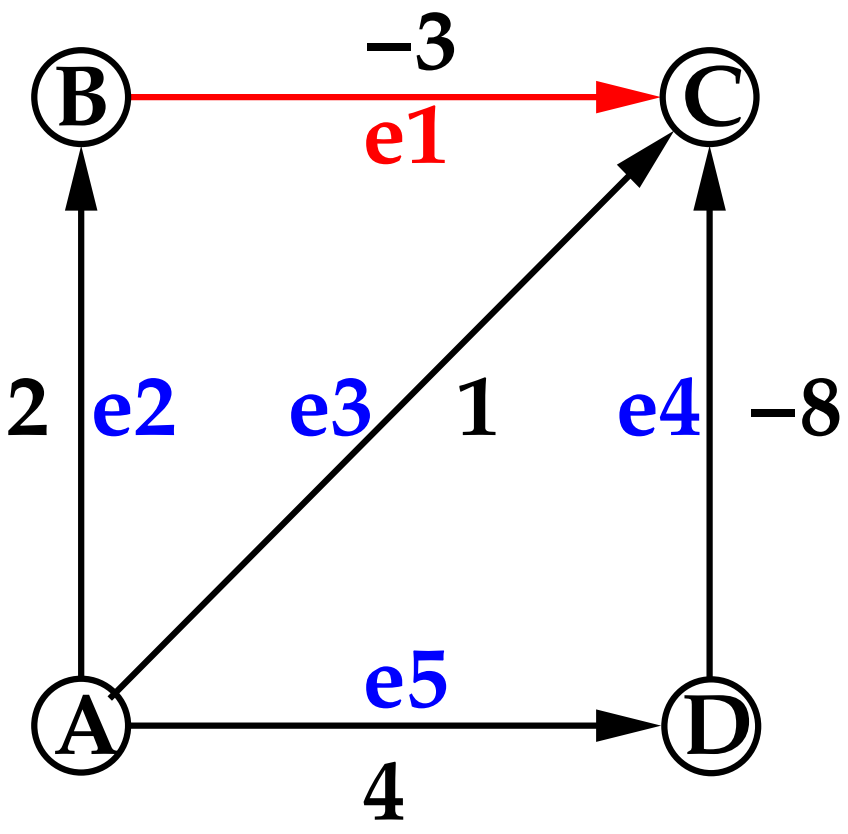
d()					
A	0	0			
B	∞	2			
C	∞	1	-1	-4	
D	∞	4			

Ford algoritmusára példa



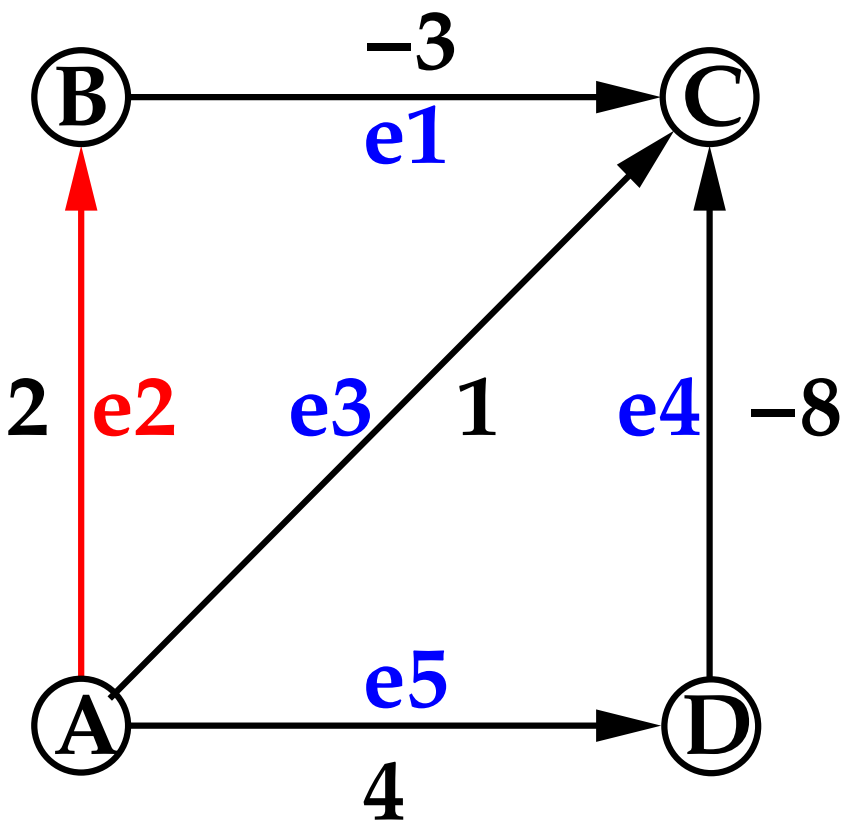
d()					
A	0	0			
B	∞	2			
C	∞	1	-1	-4	
D	∞	4			

Ford algoritmusára példa



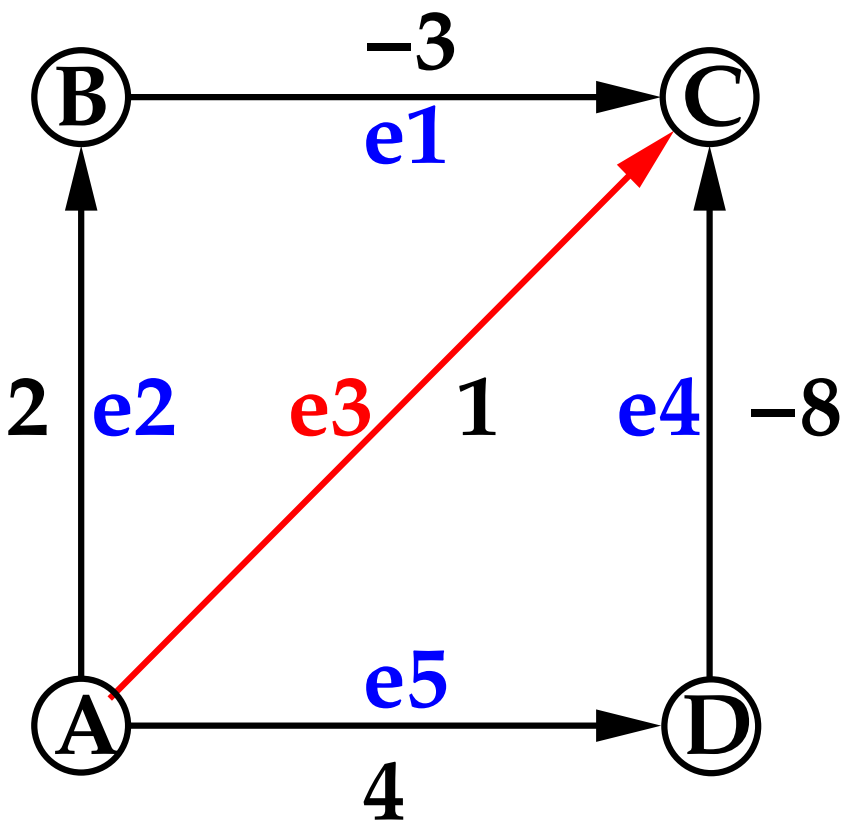
d()					
A	0	0		0	
B	∞	2		2	
C	∞	1	-1	-4	
D	∞	4		4	

Ford algoritmusára példa



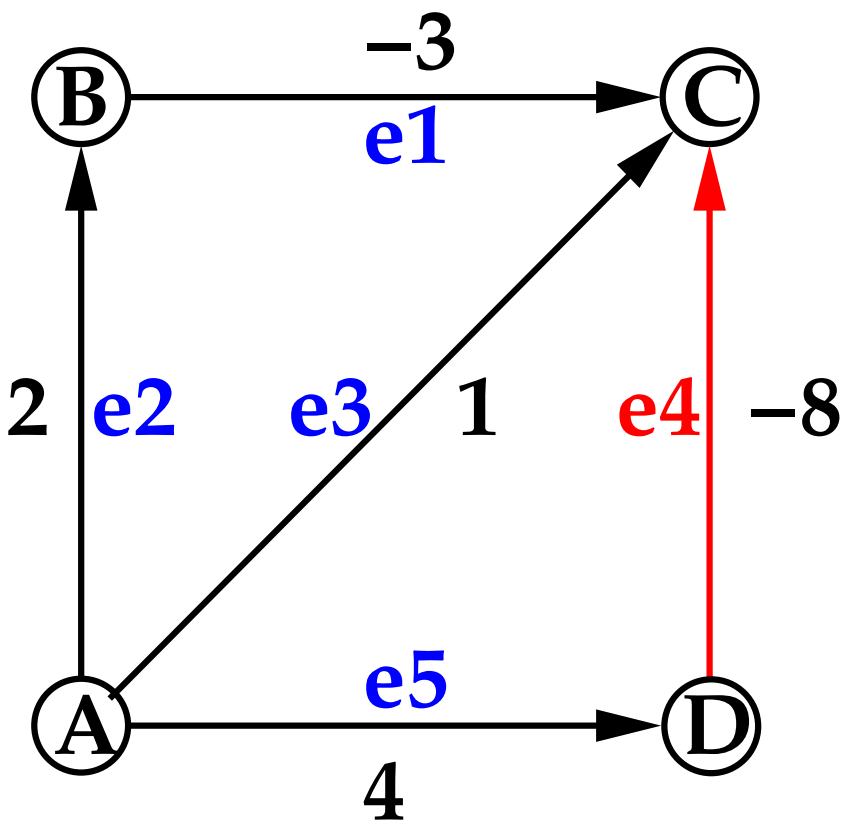
d()					
A	0	0		0	
B	∞	2		2	
C	∞	1	-1	-4	
D	∞	4		4	

Ford algoritmusára példa



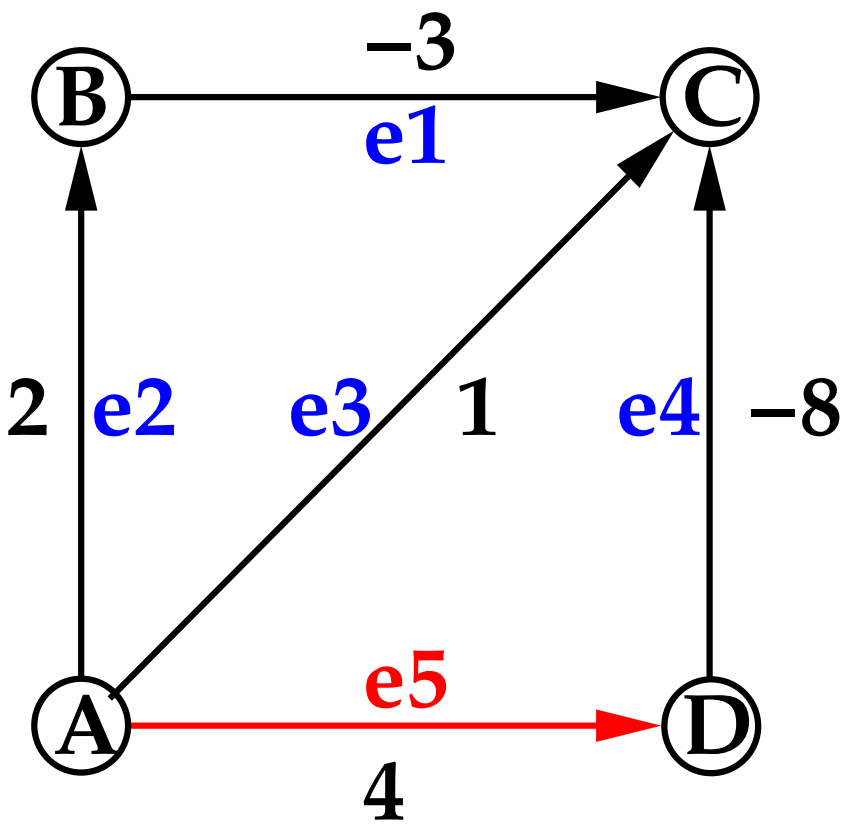
d()					
A	0	0		0	
B	∞	2		2	
C	∞	1	-1	-4	
D	∞	4		4	

Ford algoritmusára példa



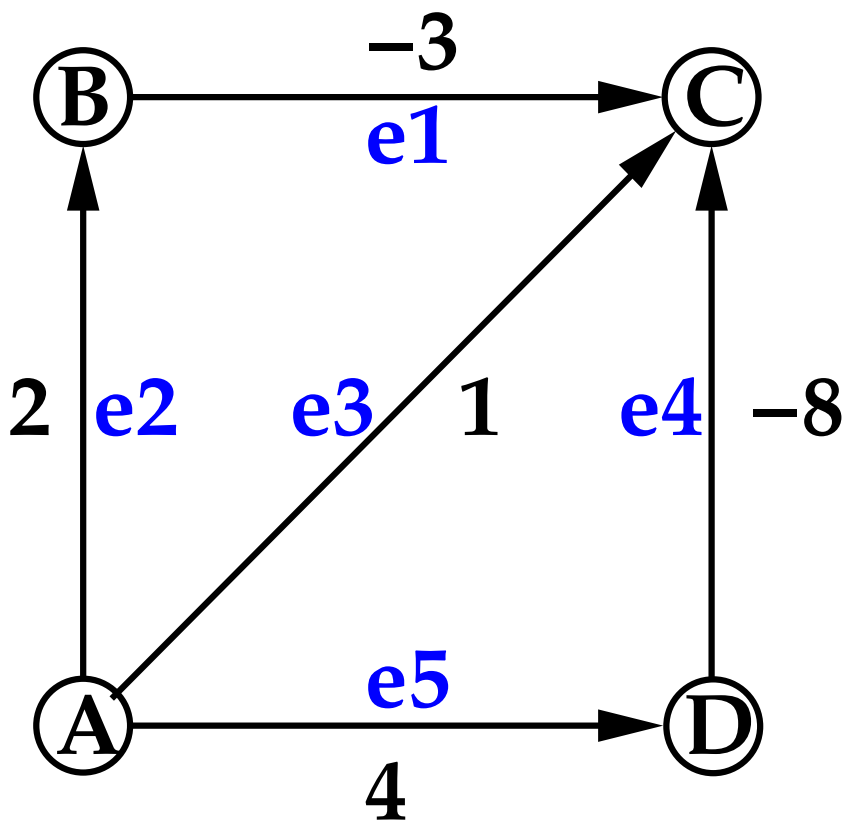
d()					
A	0	0		0	
B	∞	2		2	
C	∞	1	-1	-4	
D	∞	4		4	

Ford algoritmusára példa



d()					
A	0	0		0	
B	∞	2		2	
C	∞	1	-1	-4	
D	∞	4		4	

Ford algoritmusára példa

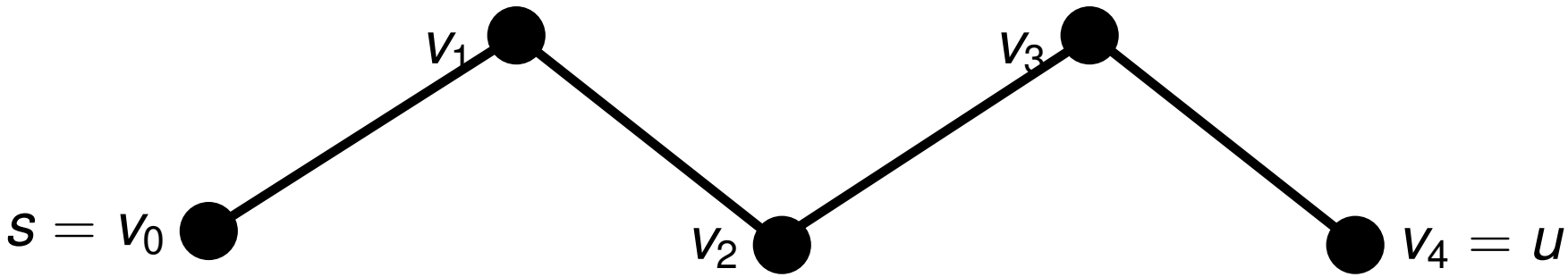


d()					
A	0	0		0	0
B	∞	2		2	2
C	∞	1	-1	-4	-4
D	∞	4		4	4

Észrevétel: Ha az algoritmus 1-es pontjának végrehajtása során semelyik él mentén sem tudunk javítani, akkor ezt követően sem fogunk tudni. Ezért ebben az esetben megállhatunk, akkor is ha az *i* még nem érte el a csúcsszámot.

Miért ad jó eredményt a Ford algoritmus?

Legyen s az a csúcs amiből a Ford algot indítottuk és legyen u egy tetszőleges másik csúcsa a gráfnak. Ekkor egy s és u közötti legrövidebb út, mely az $s = v_1, v_2, \ldots v_k = u$ csúcsokat tartalmazza ebben a sorrendben, maxmimum $n - 1$ élből áll.

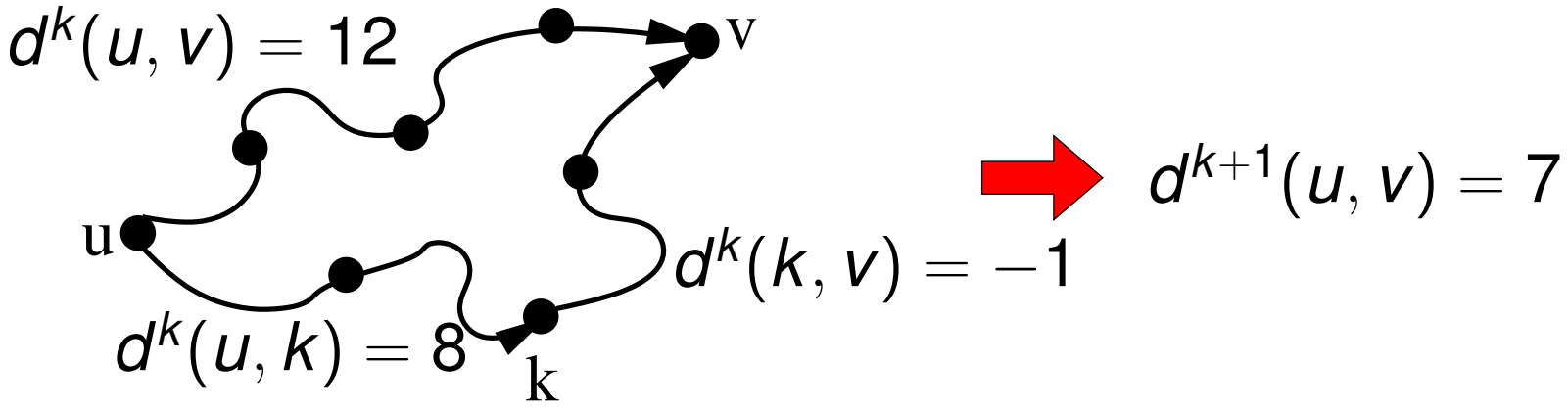


Ekkor s, v_1 egy s és v_1 közötti legrövidebb út, s, v_1, v_2 egy s és v_2 közötti legrövidebb út, stb. Az első javítási fázis során javítok az (s, v_1) él mentén, emiatt innentől v_1 felső becslése pontos. A második javítási fázis során javítok az (v_1, v_2) él mentén, és mivel v_1 becslése már pontos volt, így v_2 becslése is pontos. A k. javítási fázis során javítok a (v_{k-1}, v_k) él mentén és mivel v_{k-1} becslése már pontos volt, emiatt v_k becslése is pontos. $n - 1$ fázis után készen vagyok.

Összes csúcspár közötti távolságok meghatározása

Ez a feladat megoldható csúcsszám darab Ford lefuttatásával. Ehelyett most egy másik megoldásról, a Floyd algoritmusról lesz szó:

Ötlet: Ha u-ból v-be akarok menni és van egy u-ból k-ba vezető, illetve egy k-ból v-be vezető rövid utam, akkor ezek egymás után fűzése is lehet rövid.



Ezt felhasználva, ha $d^k()$ egy felső becslés, akkor készíthetünk egy jobb $d^{k+1}()$ felső becslést:
$d^{k+1}(u,v) := \min\left(d^k(u,v), d^k(u,k) + d^k(k,v)\right)$

Floyd működése

Egy $d^k(u, v)$ felső becslést számol, melynek a jelentése: A legrövidebb u és v között futó út hossza mely u-n és v-n kívül csak k-nál kisebb sorszámú csúcsokat használ. Nyilván $d^{|V|+1}(u, v) = dist(u, v)$.
A Floyd tehát úgy működik, hogy az első körben csak közvetlen, egy élet tartalmazó utakat néz. A második körben olyan utakat néz melyek a kezdő és végpontjukon kívül tartalmazhatják az 1-es csúcsot is. A k. körben pedig olyan utakat néz melyek tartalamzhatják az $1, 2 \ldots k-1$. csúcsot. Minden körben meghatározza a legrövidebb ilyen tipusú utak hosszát a $d^{k+1}(u, v) := \min\left(d^k(u, v), d^k(u, k) + d^k(k, v)\right)$ formula segítségével.

Floyd algoritmusa

Input: Egy $G = (V, E)$ irányított vagy irányítatlan gráf, egy az élhalmazán értelmezett konzervatív hosszfüggvény: l.

0. Címkézzük a csúcsokat 1-től $|V|$-ig egész számokkal. Minden szomszédos u, v (különböző) csúcspár esetén legyen $d^1(u, v) := l(u, v)$, a nem szomszédosokra pedig $d^1(u, v) := \infty$. Legyen továbbá $d^1(u, u) := 0$ minden u csúcsra. Legyen $k := 1$.

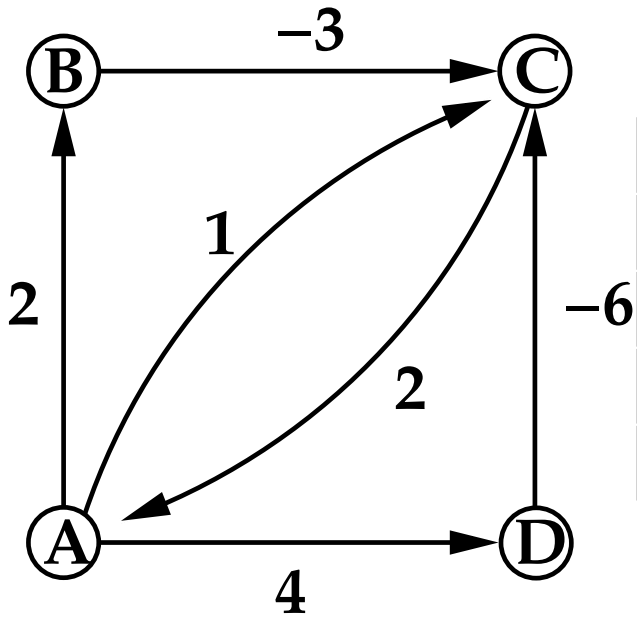
1. Minden u, v csúcspárra $d^{k+1}(u, v) := \min\left(d^k(u, v), d^k(u, k) + d^k(k, v)\right)$.

2. Ha $k = V(G) + 1$ akkor STOP, különben $k := k + 1$ és folytassuk az 1-es lépéssel.

Output: $d^{|V|+1}()$ ami megmondja bármely két csúcs távolságát.

Floyd algoritmus példa

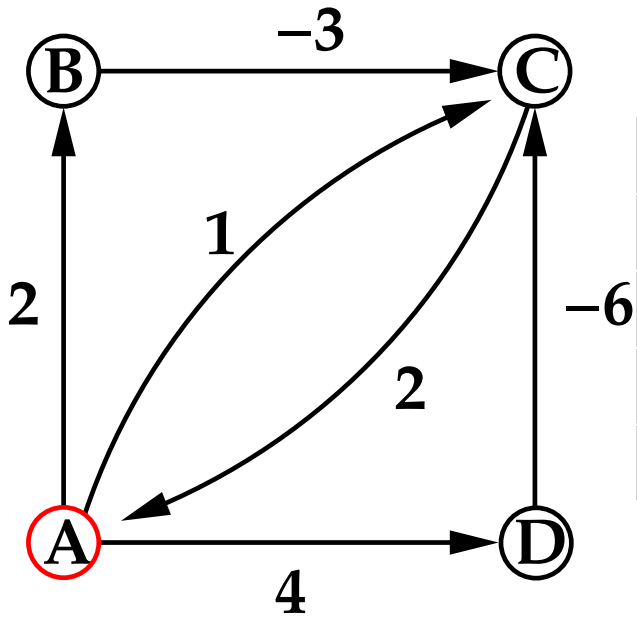
A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.



d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.

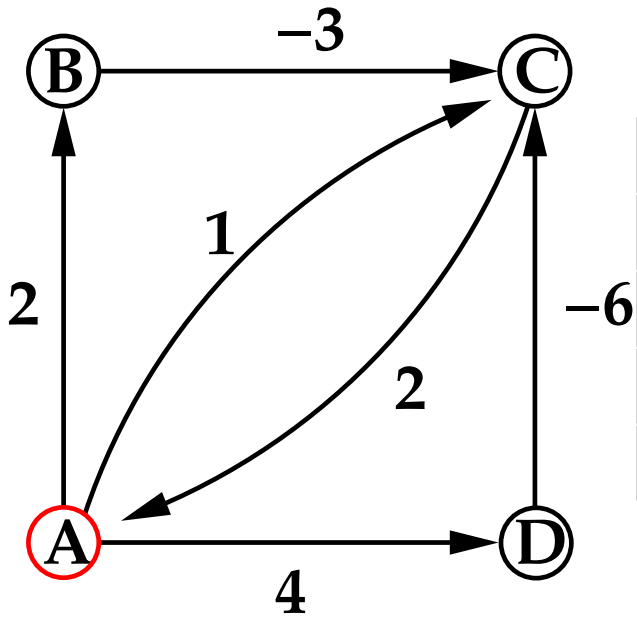


d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

d^2	A	B	C	D
A	0	2	1	4
B	∞	0		
C	2		0	
D	∞			0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.

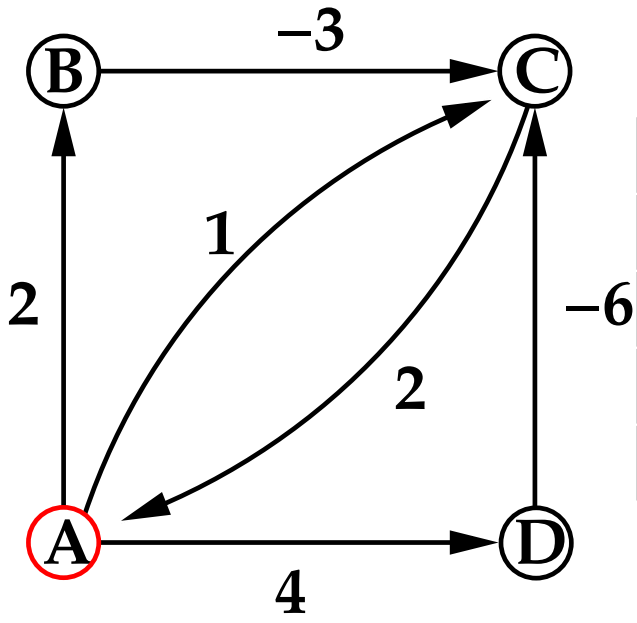


d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2		0	
D	∞			0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.

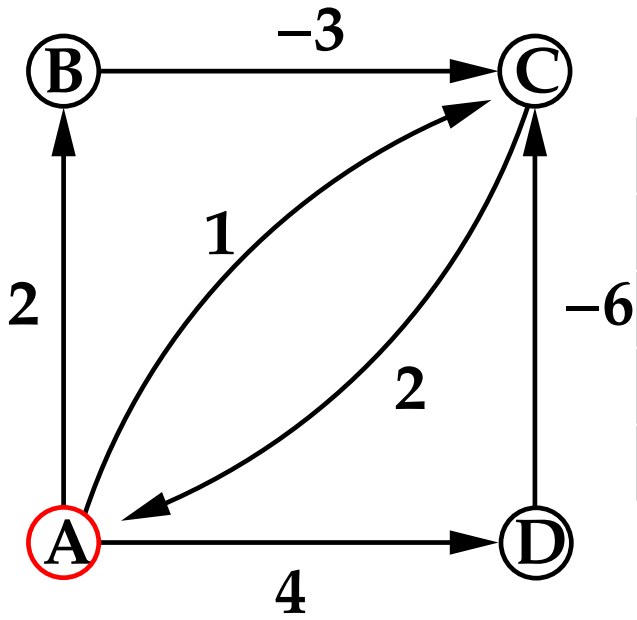


d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	
D	∞			0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.

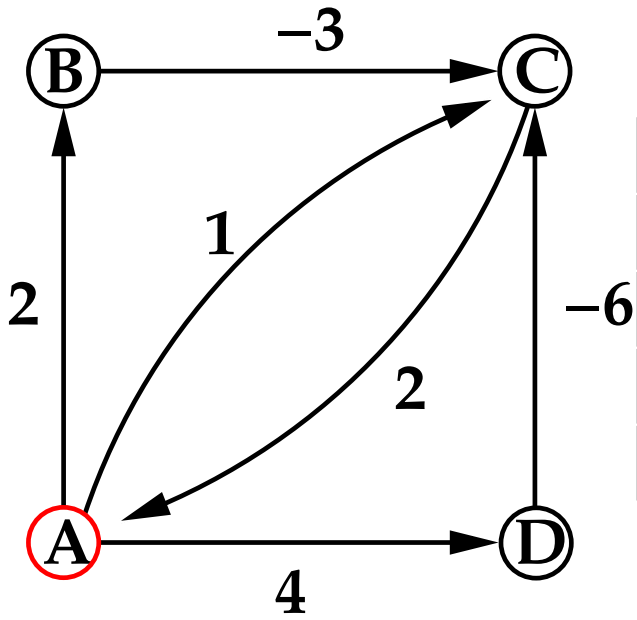


d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞			0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.

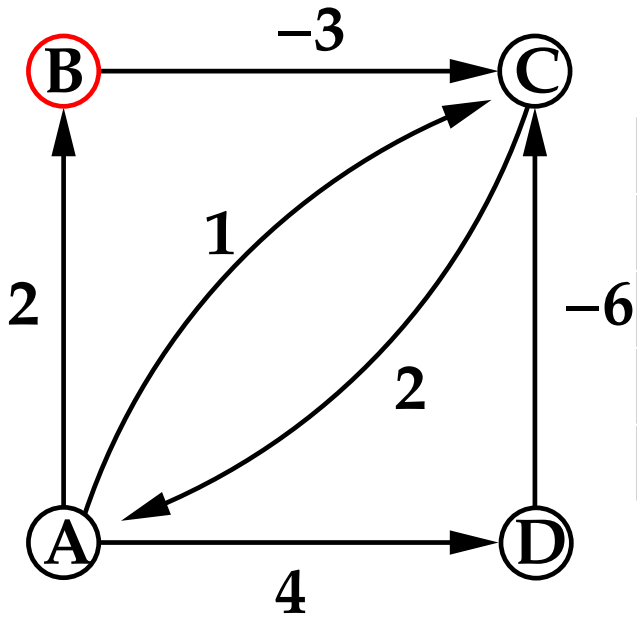


d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.



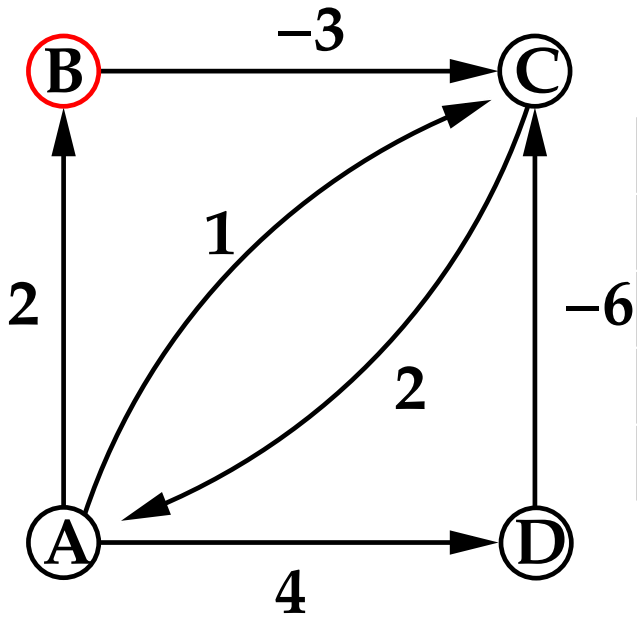
d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^3	A	B	C	D
A	0	2		
B	∞	0	-3	∞
C		4	0	
D		∞		0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.



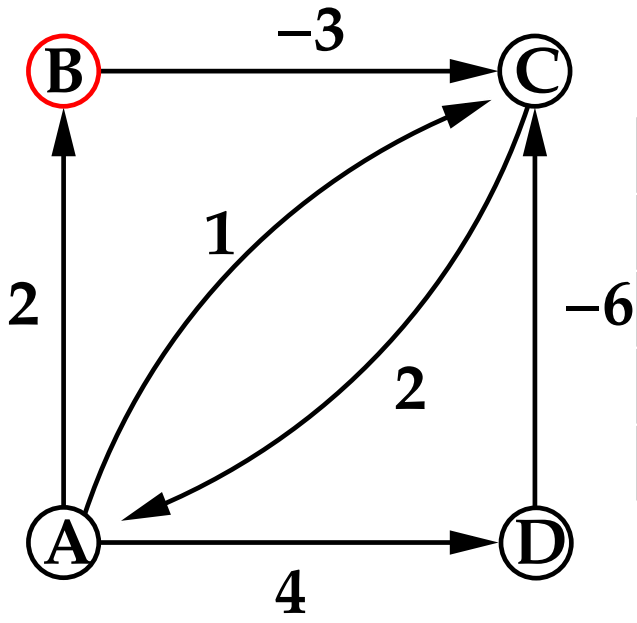
d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^3	A	B	C	D
A	0	2	-1	
B	∞	0	-3	∞
C		4	0	
D		∞		0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.



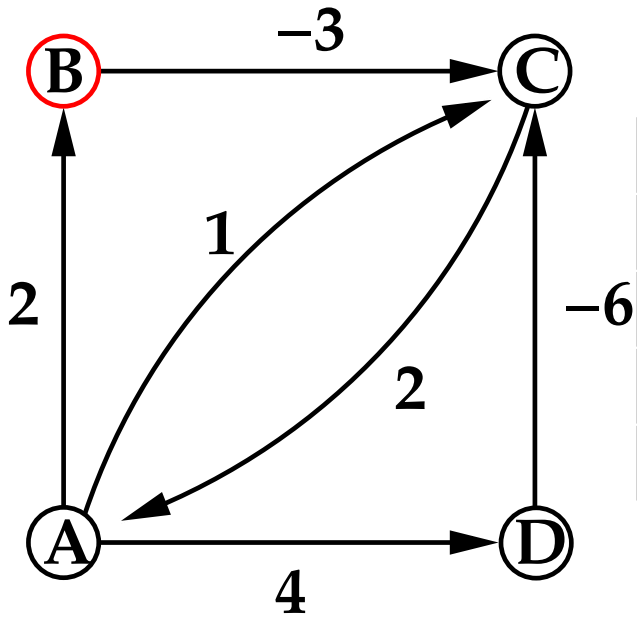
d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^3	A	B	C	D
A	0	2	-1	4
B	∞	0	-3	∞
C		4	0	6
D		∞		0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.



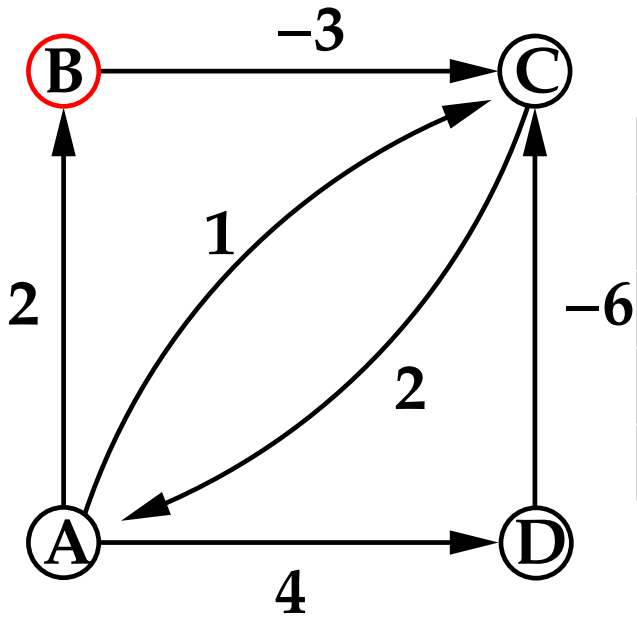
d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^3	A	B	C	D
A	0	2	-1	4
B	∞	0	-3	∞
C	2	4	0	6
D		∞		0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.



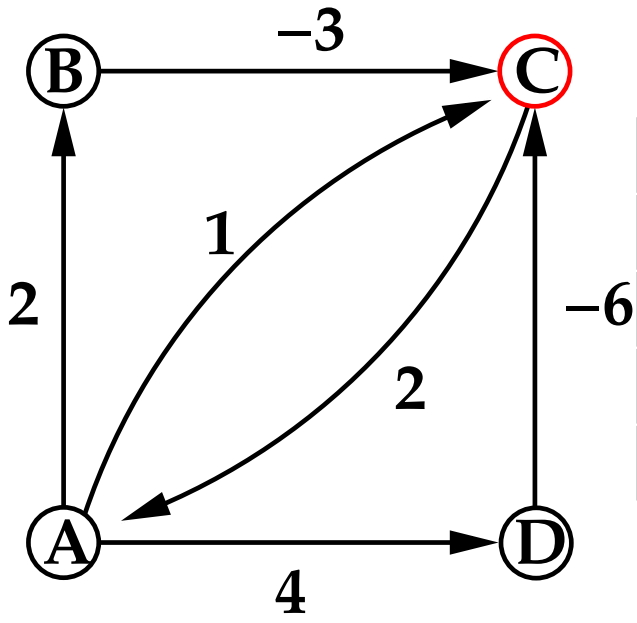
d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^3	A	B	C	D
A	0	2	-1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.



d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

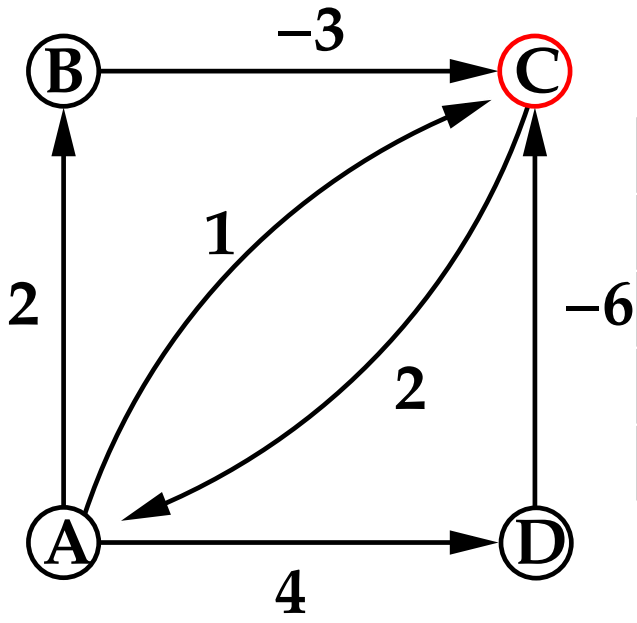
d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^3	A	B	C	D
A	0	2	-1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^4	A	B	C	D
A	0		-1	
B		0	-3	
C	2	4	0	6
D			-6	0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.



d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

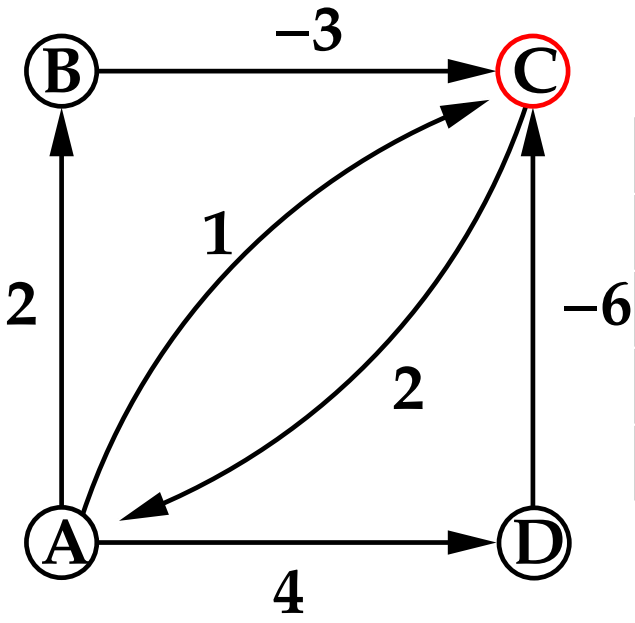
d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^3	A	B	C	D
A	0	2	-1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^4	A	B	C	D
A	0	2	-1	
B		0	-3	
C	2	4	0	6
D			-6	0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.



d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

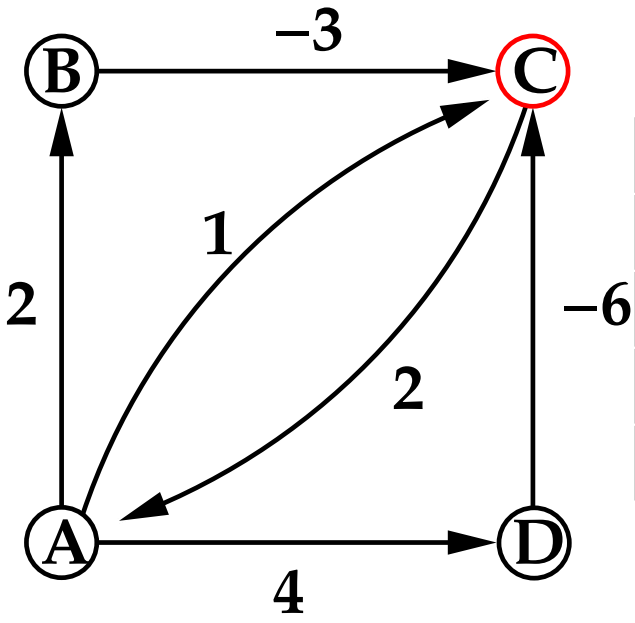
d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^3	A	B	C	D
A	0	2	-1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^4	A	B	C	D
A	0	2	-1	4
B		0	-3	
C	2	4	0	6
D			-6	0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.



d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

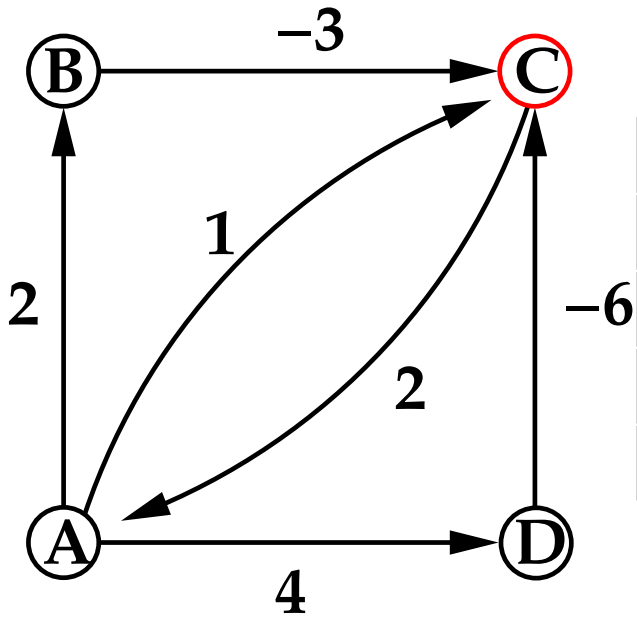
d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^3	A	B	C	D
A	0	2	-1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^4	A	B	C	D
A	0	2	-1	4
B	-1	0	-3	3
C	2	4	0	6
D			-6	0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.



d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

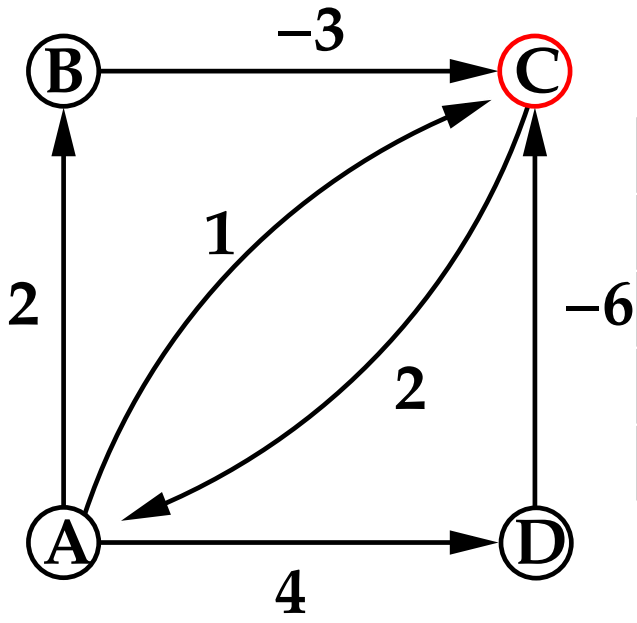
d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^3	A	B	C	D
A	0	2	-1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^4	A	B	C	D
A	0	2	-1	4
B	-1	0	-3	3
C	2	4	0	6
D	-4		-6	0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.



d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

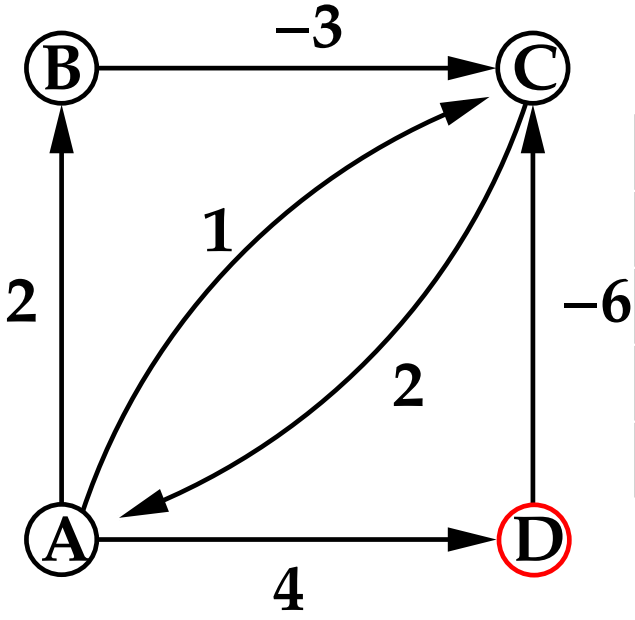
d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^3	A	B	C	D
A	0	2	-1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^4	A	B	C	D
A	0	2	-1	4
B	-1	0	-3	3
C	2	4	0	6
D	-4	-2	-6	0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.



d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

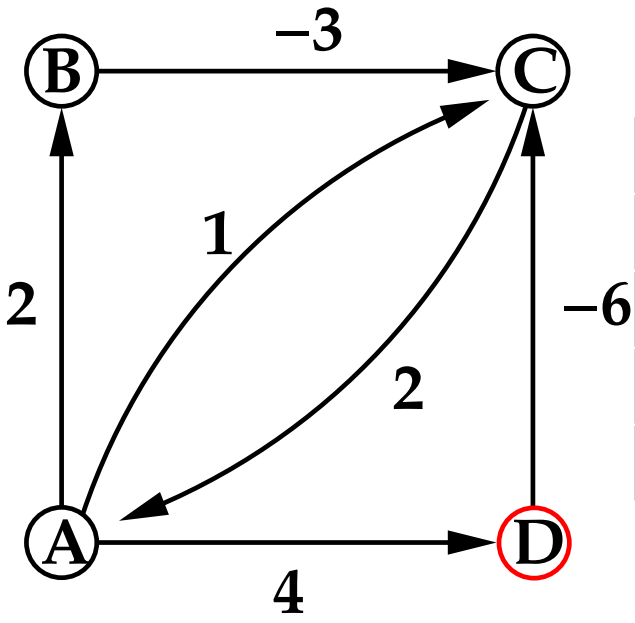
d^3	A	B	C	D
A	0	2	-1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^4	A	B	C	D
A	0	2	-1	4
B	-1	0	-3	3
C	2	4	0	6
D	-4	-2	-6	0

d^5	A	B	C	D
A	0			4
B		0		3
C			0	6
D	−4	−2	-6	0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.



d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

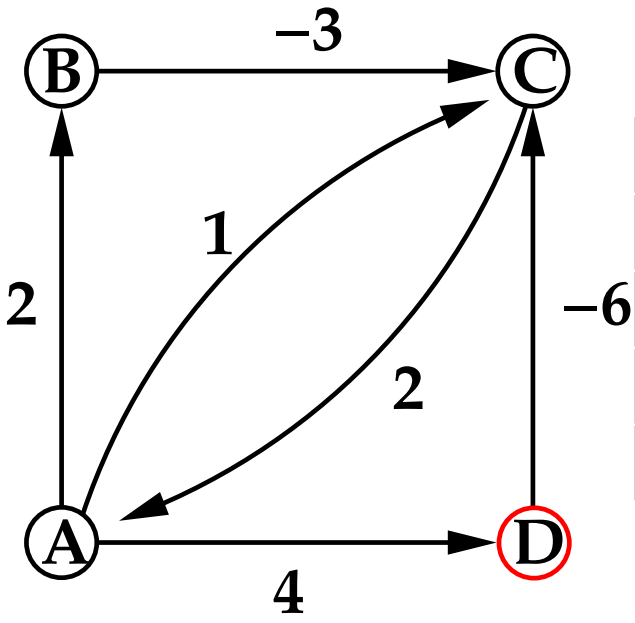
d^3	A	B	C	D
A	0	2	-1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^4	A	B	C	D
A	0	2	-1	4
B	-1	0	-3	3
C	2	4	0	6
D	-4	-2	-6	0

d^5	A	B	C	D
A	0	2		4
B		0		3
C			0	6
D	−4	−2	-6	0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.



d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

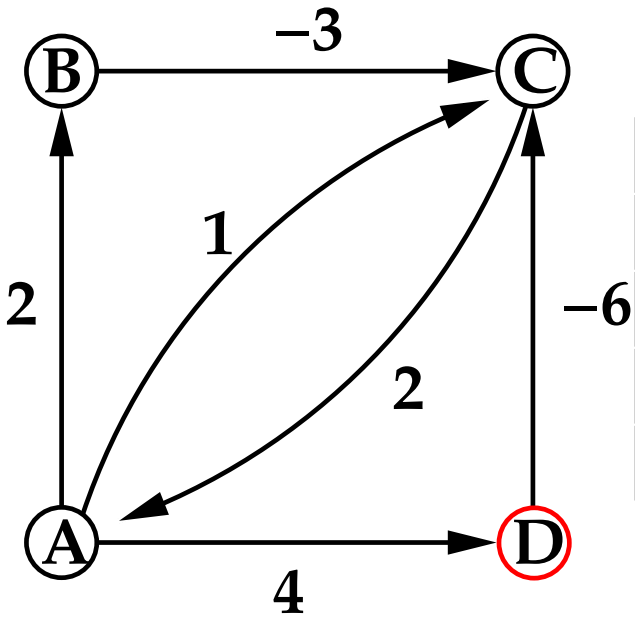
d^3	A	B	C	D
A	0	2	-1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^4	A	B	C	D
A	0	2	-1	4
B	-1	0	-3	3
C	2	4	0	6
D	-4	-2	-6	0

d^5	A	B	C	D
A	0	2	-2	4
B		0		3
C			0	6
D	−4	−2	-6	0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.



d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

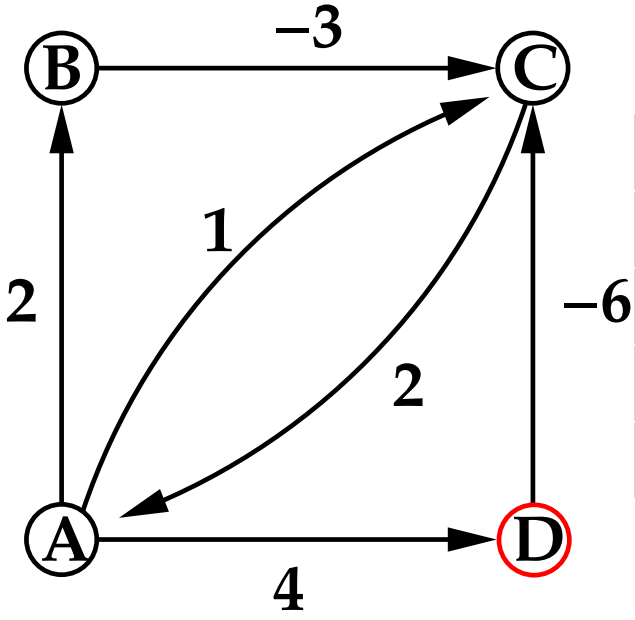
d^3	A	B	C	D
A	0	2	-1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^4	A	B	C	D
A	0	2	-1	4
B	-1	0	-3	3
C	2	4	0	6
D	-4	-2	-6	0

d^5	A	B	C	D
A	0	2	-2	4
B	-1	0		3
C			0	6
D	−4	−2	-6	0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.



d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

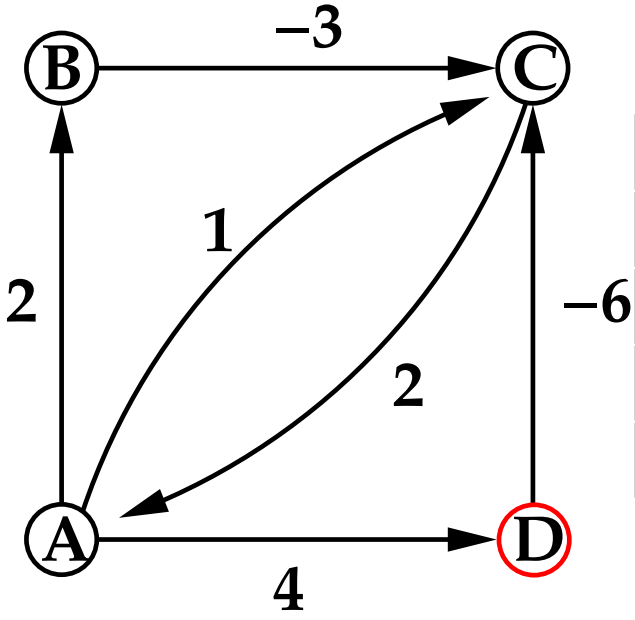
d^3	A	B	C	D
A	0	2	-1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^4	A	B	C	D
A	0	2	-1	4
B	-1	0	-3	3
C	2	4	0	6
D	-4	-2	-6	0

d^5	A	B	C	D
A	0	2	-2	4
B	-1	0	-3	3
C			0	6
D	−4	−2	-6	0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.



d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

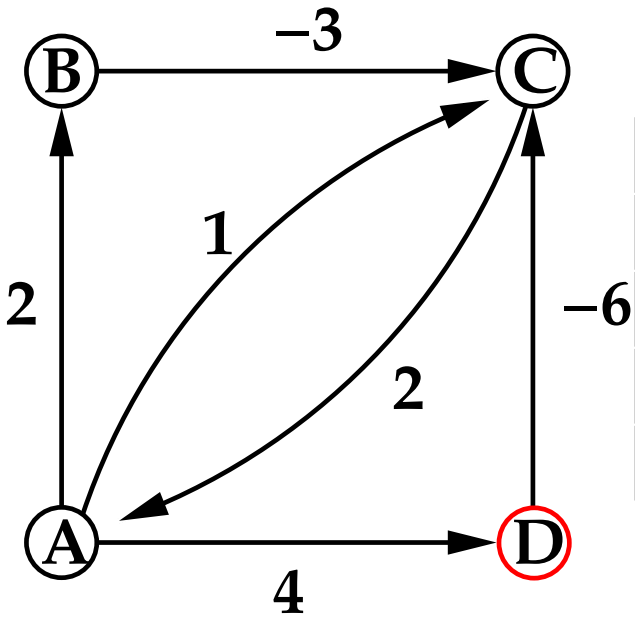
d^3	A	B	C	D
A	0	2	-1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^4	A	B	C	D
A	0	2	-1	4
B	-1	0	-3	3
C	2	4	0	6
D	-4	-2	-6	0

d^5	A	B	C	D
A	0	2	-2	4
B	-1	0	-3	3
C	2		0	6
D	−4	−2	-6	0

Floyd algoritmus példa

A csúcsok számozása megegyezik az ABC-ben elfoglalt helyükkel. Az utolsó d^5 táblázat tartalmazza a távolságokat.



d^1	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	∞	0	∞
D	∞	∞	-6	0

d^2	A	B	C	D
A	0	2	1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^3	A	B	C	D
A	0	2	-1	4
B	∞	0	-3	∞
C	2	4	0	6
D	∞	∞	-6	0

d^4	A	B	C	D
A	0	2	-1	4
B	-1	0	-3	3
C	2	4	0	6
D	-4	-2	-6	0

d^5	A	B	C	D
A	0	2	-2	4
B	-1	0	-3	3
C	2	4	0	6
D	−4	−2	-6	0

Tanácsok a Floydhoz:

- Az átló mindig 0!
- Ha az i. csúcson keresztül javítunk, azaz éppen d^{i+1}-et számoljuk, akkor az i. csúcs sora és oszlopa másolható az előző táblázatból!

Legrövidebb utakat kereső algoritmusok összehasolítása

Algoritmus	Negatív él?	Mit ad meg	Lépésszám
Dijkstra	Nem lehet	$dist(s, v)$ ahol s fix	cn^2
Ford	Lehet	$dist(s, v)$ ahol s fix	cne
Floyd	Lehet	$dist(u, v)$ $\forall u, v \in V$	cn^3

Lépésszámoknál e a gráf éleinek száma, n a gráf csúcsainak száma, c pedig az input gráftól nem függő konstans ami a konkrét implementációtól függ.

Legrövidebb utakat kereső algoritmusok összehasolítása

Algoritmus	Negatív él?	Mit ad meg	Lépésszám
Dijkstra	Nem lehet	$dist(s, v)$ ahol s fix	cn^2
Ford	Lehet	$dist(s, v)$ ahol s fix	cne
Floyd	Lehet	$dist(u, v)$ $\forall u, v \in V$	cn^3

Lépésszámoknál e a gráf éleinek száma, n a gráf csúcsainak száma, c pedig az input gráftól nem függő konstans ami a konkrét implementációtól függ.

Megjegyzés: Az élszám egyszerű gráfok esetén is lehet jóval nagyobb a csúcsszámnál, (Gondoljunk a teljes gráfra ahol $\approx n^2/2$.). Emiatt ha az összes csúcspár esetén kíváncsiak vagyunk a távolságukra, akkor ezek egy Floyd futtatásával gyorsabban kiszámíthatóak mint n darab Forddal.

Konklúzió: Ha nincsenek negatív élek, akkor használjuk a Dijkstrát mert az a leggyorsabb! A Floyd a leglasabb, de az a legáltalánosabb.