Minimális feszítőfát kereső algoritmusok

Csima Judit
BME SZIT
csima@cs.bme.hu

2019. december 11.

A BFS és DFS eljárásoknál láttuk, hogy egy irányítatlan gráfban pontosan akkor van feszítőfa, ha a gráf összefüggő és azt is láttuk, hogy összefüggő gráfban mind a BFS (szélességi bejárás), mind a DFS (mélységi bejárás) talál is egy feszítőfát. Ha a gráf élei nincsenek súlyozva, akkor minden feszítőfa ugyanolyan jó, mindegyikben $n-1$ él van egy n csúcsú gráf esetén, de ha a gráf élsúlyozott, ahol a súlyok valamiféle költséget jelentenek, akkor vannak olcsóbb és drágább feszítőfák. Ebben a részben olyan algoritmusokat fogunk tanulni, amik megtalálják egy összefüggő, irányítatlan gráf egy olyan feszítőfáját, melynek a súlya a lehető legkisebb (a fa súlya alatt a fát alkotó élek súlyainak összegét értjük).

Ez a feladat számos esetben előkerül a gyakorlatban, ilyen például a következő helyzet:

Adott n város és a köztük vezető közvetlen utak. Fel szeretnénk újítani néhány utat a lehető legolcsóbban úgy, hogy bármely városból el lehessen jutni bármely városba felújított útvonalon, ehhez ismert hogy melyik útszakasz felújítás mennyibe kerül (ha van él két csúcs között, azaz közvetlen út két város között).
A felújított utakból, azaz élekből álló résznek összefüggőnek kell tehát lennie (ezt jelenti az, hogy bárhonnan bárhova van felújított útvonal), minden csúcsot (azaz minden várost) le kell fednie és ha a legolcsóbb megoldást keressük, akkor körmentesnek is kell lennie, hiszen ha lenne kör, akkor annak bármelyik élét elhagyva egy olcsóbb, de még mindig összefüggő, minden csúcsot lefedő megoldást kapnánk.

Azt kaptuk tehát, hogy a feladat egy olyan feszítőfa keresését jelenti, melynek súlya a lehető legkisebb.

Prim algoritmusa

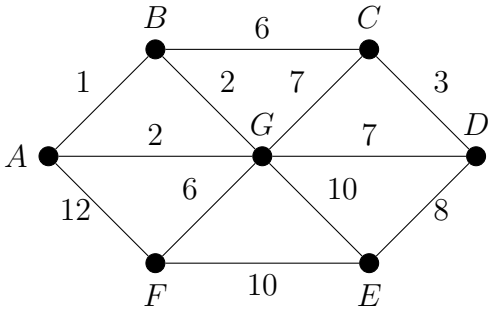
A megoldandó feladat tehát a következő: adott egy irányítatlan, összefüggő, élsúlyozott G gráf, ebben kell egy lehető legkisebb súlyú feszítőfát találnunk. A gráf élsúlyozását a $c(e) = c(u, v)$ függvény adja meg.

Prim algoritmusának elve

- Választunk egy kezdőcsúcsot, ez a gráf bármelyik csúcsa lehet, legyen ennek neve s.

- Lesz egy LEFEDVE halmaz, ebben azok a csúcsok lesznek, amiket már lefedtem a készülő feszítőfával, az elején LEDEDVE $=\{s\}$.
- Lesz egy F lista, ebbe kerülnek majd a feszítőfa élei, az elején F még üres.
- Minden lépésben egy új élet fogok hozzáadni F-hez, ami egy új, eddig nem lefedett v csúcsba vezet: azt az élet fogom választani, ami a LEFEDVE halmazból kivezető legkisebb súlyú él a gráfban (ha több ilyen is van, akkor ezek egyikét). Ez az él bekerül az F-be, az az eddig le nem fedett csúcs, amibe vezet, az pedig bekerül a LEFEDVE halmazba.
- Ezt az eljárást addig folytatom, amíg van kivezető él a LEFEDVE halmazból.

Nézzük meg az algoritmus futását az alábbi gráfon, az A csúcsból indítva.



1. Az elején F üres, LEFEDVE-ben pedig csak A van.
2. A legkisebb kivezető él LEFEDVE-ből az AB él, azaz ekkor $F = \{AB\}$ és $LEFEDVE = \{A, B\}$.
3. Most a legkisebb kivezető él LEFEDVE-ből vagy az AG vagy a BG él, válasszuk most AG-t, azaz ekkor $F = \{AB, AG\}$ és $LEFEDVE = \{A, B, G\}$.
4. Most a legkisebb kivezető él LEFEDVE-ből a BC él, azaz ekkor $F = \{AB, AG, BC\}$ és $LEFEDVE = \{A, B, G, C\}$.
5. Most a legkisebb kivezető él LEFEDVE-ből a CD él, azaz ekkor $F = \{AB, AG, BC, CD\}$ és $LEFEDVE = \{A, B, G, C, D\}$.
6. Most a legkisebb kivezető él LEFEDVE-ből a GF él, azaz ekkor $F = \{AB, AG, BC, CD, GF\}$ és $LEFEDVE = \{A, B, G, C, D, F\}$.
7. Most a legkisebb kivezető él LEFEDVE-ből a DE él, azaz ekkor $F = \{AB, AG, BC, CD, GF, DE\}$ és $LEFEDVE = \{A, B, G, C, D, F, E\}$.
8. Most már nincsen kivezető él, minden csúcs le van fedve, az algoritmus leáll.

Prim algoritmusának helyessége

Az eljárás helyességéről csak azt látjuk be, hogy az eljárás végén a kapott F halmaz egy feszítőfát alkot, azt nem bizonyítjuk, hogy ez egy minimális súlyú feszítőfa. Természetesen ez is igaz, csak a bizonyítás nem fér bele az időnkbe.

1. F körmentes: az újonnan bevett él mindig egy új csúcsba vezet, sose megyünk vissza már lefedett csúcsba, nem keletkezik kör.

2. F összefüggő: az újonnan bevett él az egyik végpontjánál (ami LEFEDVE-ben van) kapcsolódik a korábban bevett élek gráfjához.

3. F minden csúcsot lefed a végén: az eljárás csak akkor áll le, ha már nincsen kivezető él a LEFEDVE halmazból, de mivel G összefüggő, ezért mindaddig amíg vannak nem lefedett csúcsok, addig vannak ezekbe vezető élek is (legalább egy), vagyis az algoritmus nem áll le. Ez azt jelenti, hogy ha az eljárás leáll, akkor az csak úgy lehetséges, hogy már minden csúcs bekerült a LEFEDVE halmazba.

4. A kapott F gráf minimális összesúlyú az összes feszítőfa között: ezt nem bizonyítjuk.

Prim algoritmusának pszeudokódja

A pszeudokódban az alábbi jelöléseket használjuk, illetve a kód az alábbi logikát követi:

1. A választott kezdőcsúcs neve s.

2. A LEFEDVE halmazban azok a csúcsok vannak, amiket már lefedtünk a készülő feszítőfával, az elején LEDEDVE $=\{s\}$.

3. F a feszítőfába beválasztott élek listája, az a elején F még üres.

4. Azokra a v csúcsokra, amik nincsenek benne a LEFEDVE halmazban nyilvántartom egy *közeli* nevű tömbben, hogy melyik a hozzájuk legközelebbi csúcs a LEFEDVE halmazban (azaz *közeli*$[v]$ azt adja meg, hogy melyik lefedett csúcsból vezet v-be a legkisebb súlyú él), egy *legolcsóbb* nevű tömbben pedig ennek az élnek a súlyát tárolom (azaz *legolcsóbb*$[v]$ a LEFEDVE halmazból v-be vezető legkisebb súlyú él súlya). Azokra a csúcsokra, amik már benne vannak a LEFEDVE halmazban, azokra *legolcsóbb*$[v] = *$.

5. Az eljárás elején még csak s lesz benne a LEFEDVE halmazban, ezért *legolcsóbb*$[s] = *$, *legolcsóbb*$[v] = c(s, v)$, ha van él s-ből v-be (mert ekkor ez az egyetlen lehetőség v elérésére) és *legolcsóbb*$[v] = \infty$, ha nincs él s-ből v-be (mert ekkor nem lehet elérni v-t a LEFEDVE halmazból).

6. Az eljárás elején a fentiekkel összehangban *közeli*$[s] = s$, *közeli*$[v] = s$, ha van él s-ből v-be (mert ekkor ez az s-ből jövő egyetlen lehetőség a legjobb v elérésére) és *közeli*$[v] = \infty$, ha nincs él s-ből v-be (mert ekkor nem lehet elérni v-t a LEFEDVE halmazból).

7. Minden fázisban a LEFEDVE halmazon kívüli csúcsok közül (vagyis azok közül, akikre *legolcsóbb*$[v]$ még nem $*$) kiválasztjuk azt a v^* csúcsot, melynek *legolcsóbb* értéke minimális és ezt a csúcsot beletesszük a LEFEDVE halmazba, a (*közeli*$[v^*], v^*$) élet pedig F-be tesszük, majd *legolcsóbb*$[v^*]$-ot $*$-ra állítjuk. Ez az a lépés, amikor a legkisebb, LEFEDVE-ből kivezető élet kiválasztjuk.

8. Miután v^* bekerült a LEFEDVE halmazba, lehetséges, hogy v^* egy w szomszédjára rövidebb élet kapunk v^*-ból, mint az eddig ide vezető legrövidebb él hossza, ezért v^* minden w szomszédjára megnézzük, hogy $c(v^*, w)$ kisebb-e, mint *legolcsóbb*$[w]$ (azaz rövidebb-e a v^*-ból jövő él w-be) és ha igen, akkor *legolcsóbb*$[w]$-t $c(v^*, w)$-re, *közeli*$[w]$-t pedig v^*-ra állítjuk (hiszen a legrövidebb él v^*-ból jön).

9. Az éppen bekerülő v^* csúcs szomszédainak *legolcsóbb* és *közeli* értékét egy ciklussal fogjuk frissíteni, ahol végigmegyünk a gráf összes csúcsán és minden olyan w csúcsra, amibe megy él v^*-ból ($A[v^*, w]$ nem végtelen) és amely w csúcs még nincs LEFEDVE-ben (azaz *legolcsóbb*$[w]$ nem *) megnézzük, hogy $c(v^*, w)$ kisebb-e, mint *legolcsóbb*$[w]$ és ha igen, akkor *legolcsóbb*$[w]$-t $c(v^*, w)$-re, *közeli*$[w]$-t pedig v^*-ra állítjuk.

10. Az eljárást addig csináljuk, amíg vannak csúcsok a LEFEDVE halmazon kívül, azaz amíg van olyan v, amire *legolcsóbb*$[v]$ nem $*$.

A teljes kód a következő lesz:

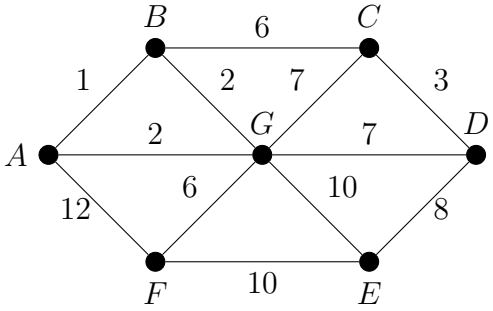
```
LEFEDVE = {s}
F = üres lista

legolcsóbb[v] = *, ha v = s
legolcsóbb[v] = c(s,v), ha van él s-ból v-be, egyébként legolcsóbb[v] = végtelen

közeli[v] = s, ha v = s vagy ha van él s-ból v-be, egyébként közeli[v] = végtelen

ciklus amíg van olyan csúcs, ahol legolcsóbb[v] nem *:
    v* := az a csúcs, ahol legolcsóbb[v] minimális
    v* LEFEDVE-be megy
    legolcsóbb[v*] := *
    (közeli[v*], v*) F-be kerül
    ciklus w = 1-tól n-ig:
        ha A[v*,w] nem végtelen és legolcsóbb[w] nem *:
            ha c(v*, w) < legolcsóbb[w]:
                legolcsóbb[w] := c(v*, w)
                közeli[w] := v*
    ciklus vége
ciklus vége
```

Nézzük meg most ennek a kódnak a futását az előbb látott gráfon az A csúcsból:



Az elején $LEFEDVE = \{A\}$, F üres, a két tömb pedig:
legolcsóbb:

A	B	C	D	E	F	G
*	1	∞	∞	∞	12	2

közeli:

A	B	C	D	E	F	G
A	A	∞	∞	∞	A	A

Ekkor a legkisebb érték a *legolcsóbb* tömbben B-nél található, ezért B kerül be a LEFEDVE halmazba és mivel *közeli*$[B] = A$ ezért az AB él kerül be F-be, vagyis $LEFEDVE = \{A, B\}$, $F = \{AB\}$. Mivel B került be a LEFEDVE halmazba, ezért B még nem lefedett w szomszédaira nézem meg, hogy $c(B, w)$ kisebb-e, mint *legolcsóbb*$[w]$ és ha igen, akkor módosítom *legolcsóbb*$[w]$-t $c(B, w)$-re és *közeli*$[w]$-t B-re. B-nek A olyan szomszédja, aki már le van fedve, ezért itt nem kell módosítani. G-nél *legolcsóbb*$[G] = c(B, G)$, ezért itt sem kell módosítani, de C-nél kell, mert *legolcsóbb*$[C] = \infty > 6 = c(B, C)$.

A két tömb tehát így alakul:

legolcsóbb:

A	B	C	D	E	F	G
*	*	6	∞	∞	12	2

közeli:

A	B	C	D	E	F	G
A	A	B	∞	∞	A	A

Ekkor a legkisebb érték a *legolcsóbb* tömbben G-nél található, ezért G kerül be a LEFEDVE halmazba és mivel *közeli*$[G] = A$ ezért az AG él kerül be F-be, vagyis $LEFEDVE = \{A, B, G\}$, $F = \{AB, AG\}$. Mivel G került be a LEFEDVE halmazba, ezért G még nem lefedett w szomszédaira nézem meg, hogy $c(G, w)$ kisebb-e, mint *legolcsóbb*$[w]$ és ha igen, akkor módosítom *legolcsóbb*$[w]$-t $c(G, w)$-re és *közeli*$[w]$-t G-re. G szomszédai közül A és B már le van fedve, C felé nem lenne javulás a G-ből jövő él, de a másik három csúcs, D, E, F esetén kisebb a G-ből

jövő él súlya, mint a *legolcsóbb* tömb értéke, így ezeknél a csúcsoknál módosítunk és a két tömb végül így alakul:

legolcsóbb:

A	B	C	D	E	F	G
*	*	6	7	10	6	*

közeli:

A	B	C	D	E	F	G
A	A	B	G	G	G	A

Ekkor a legkisebb érték a *legolcsóbb* tömbben C-nél található, ezért C kerül be a LEFEDVE halmazba és mivel $közeli[C] = B$ ezért az BC él kerül be F-be, vagyis $LEFEDVE = \{A, B, G, C\}$, $F = \{AB, AG, BC\}$. Mivel C került be a LEFEDVE halmazba, ezért C még nem lefedett szomszédainál lehet változás, de csak D-nél lesz, a két tömb így alakul:

legolcsóbb:

A	B	C	D	E	F	G
*	*	*	3	10	6	*

közeli:

A	B	C	D	E	F	G
A	A	B	C	G	G	A

Ekkor a legkisebb érték a *legolcsóbb* tömbben D-nél található, ezért D kerül be a LEFEDVE halmazba és mivel $közeli[D] = C$ ezért az CD él kerül be F-be, vagyis $LEFEDVE = \{A, B, G, C, D\}$, $F = \{AB, AG, BC, CD\}$. Mivel D került be a LEFEDVE halmazba, ezért D még nem lefedett szomszédainál lehet változás, de csak E-nél lesz, a két tömb így alakul:

legolcsóbb:

A	B	C	D	E	F	G
*	*	*	*	8	6	*

közeli:

A	B	C	D	E	F	G
A	A	B	C	D	G	A

Ekkor a legkisebb érték a *legolcsóbb* tömbben F-nél található, ezért F kerül be a LEFEDVE halmazba és mivel *közeli*$[F] = G$ ezért az GF él kerül be F-be, vagyis $LEFEDVE = \{A, B, G, C, D, F\}$, $F = \{AB, AG, BC, CD, GF\}$. Mivel F került be a LEFEDVE halmazba, ezért F egyetlen még nem lefedett szomszédjánál lehet változás, de nem lesz, a két tömb így alakul:

legolcsóbb:

A	B	C	D	E	F	G
*	*	*	*	8	*	*

közeli:

A	B	C	D	E	F	G
A	A	B	C	D	G	A

Ekkor a legkisebb érték a *legolcsóbb* tömbben E-nél található, ezért végül E kerül be a LEFEDVE halmazba és mivel *közeli*$[E] = D$ ezért az DE él kerül be F-be, vagyis $LEFEDVE = \{A, B, G, C, D, F, E\}$, $F = \{AB, AG, BC, CD, GF, DE\}$.
Mivel E volt az utolsó csúcs aki bekerült a LEFEDVE halmazba, ezért itt már nem kell a szomszédokat megvizsgálni, a két tömb így alakul:

legolcsóbb:

A	B	C	D	E	F	G
*	*	*	*	*	*	*

közeli:

A	B	C	D	E	F	G
A	A	B	C	D	G	A

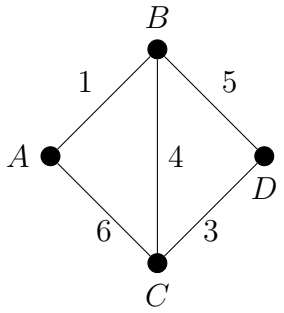
Mivel ekkor már nincsen lefedetlen csúcs (minden érték $*$ a *legolcsóbb* tömbben), ezért az algoritmus leáll.

Prim algoritmusának lépésszáma

A kezdeti lépések $O(n)$-ben mennek, mert két darab n méretű tömböt kell feltöltenünk és a LEFEDVE-t és az F-et kell beállítanunk (ami konstans lépés).
Az amíg-ciklus magja legfeljebb n-szer fut le, mert minden lépésben egy csúcs LEFEDVE-be kerül és leállunk, ha már nincs csúcs LEFEDVE-n kívül. A ciklusmagban van egy minimumkeresés a *legolcsóbb* tömbben, ez $O(n)$, aztán konstans sok lépés, majd egy belső ciklus, aminek a magja n-szer fut le és a mag konstans, tehát a belső ciklus $O(n)$-es. Vagyis az amíg-ciklus ciklusmagja $O(n) + O(1) + O(n) = O(n)$ és mivel n-szer fut el, ezért ez $O(n^2)$, ez az $O(n)$-es előkészítő résszel együtt is $O(n^2)$.

1. megjegyzés
Prim algoritmusának fenti pszeudokódja strukturájában és logikájában teljesen ugyanaz, mint a Dijkstra algoritmus kódja: mindkét esetben egy már készen levő halmazból kivezető élek közül választunk egy legjobbat és ezzel bővítjük a keresett megoldást. A különbség abban áll a két eljárás között, hogy mi számít legjobbnak, mi a kiválasztás kritériuma.

Noha a két kód eléggé hasonló és mindkét kód végén egy-egy feszítőfát kapunk (Dijktsra algoritmusának végén a meghtalált legrövidebb utak összessége is egy feszítőfát asd) a két feladat nem ugyanaz. Ezt mutatja az alábbi példa, amiben a minimális súlyú feszítőfában található utak nem egyeznek meg az A csúcsból a többi csúcsba vezető utakkal: a minimális feszítőfába az AB, BC, CD élek kerülnek, de az A-ból D-be vezető legrövidebb út ABD, mely nincsen benne a minimális feszítőfában.



2. megjegyzés
A minimális feszítőfa keresése során nincs megkötés az élsúlyokra, lehetnek negatív élek is a gráfban. Noha nem láttuk Prim algoritmusának helyességét, ez abból is látszik, hogy ha vannak negatív élek, akkor a legnagyobb abszolút értékű negatív él abszolút értékét hozzáadva minden élhez egy olyan gráfot kapunk, ahol minden él súlya nemnegatív és ebben az új gráfban ugyanaz a feszítőfa lesz minimális, ami az eredetiben az volt, mert minden potenciális feszítőfa súlyát pontosan ugyanannyival, nevezetesen $(n-1)$-szer a hozzáadott értékkel növeltük meg.

Kruskal algoritmusa

Egy másik ismert algoritmus minimális feszítőfa keresésére Kruskal algoritmusa, amit nem tárgyalunk részletesen, csak az algoritmus elvét nézzük meg és szemléltetjük az algoritmus futását egy konkrét példán.

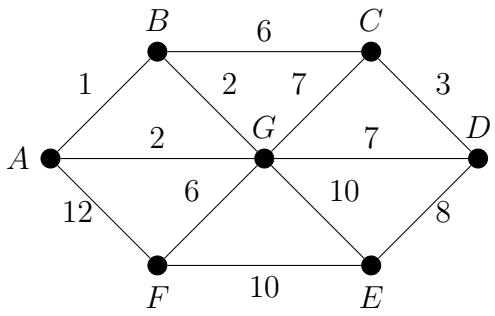
Az algoritmus elve a következő:

1. Rendezzük a gráf éleit élsúly szerint növően (ha vannak egyforma súlyok akkor nemcsökkenően).

2. Legyen F üres halmaz, ebbe gyűjtöm majd a feszítőfa éleit a futás során. Végigmegyünk a gráf élein az 1. pontban kapott sorrendben és megnézzük, hogy az éppen vizsgált e él alkot-e kört az aktuális F halmaz éleivel. Ha nem, akkor e-t F-be teszem, egyébként e-t átlépem, nem kerül F-be.

Be lehet látni, hogy ez az eljárás egy összefüggő gráf esetén feszítőfát ad, mégpedig a legkisebb súlyú feszítőfát. Ezt a bizonyítást kihagyjuk idő hiányában, megnézzük viszont az

eljárás futását a már korábban látott gráfra:



1. Az élek egy lehetséges sorrendje: $AB, AG, BG, CD, BC, FG, CG, DG, DE, EG, EF, AF$.

2. Az AB élet bevesszük, mert nem alkot kört F-fel (hiszen F üres), ekkor tehát $F = \{AB\}$.
   Az AG élet bevesszük, mert nem alkot kört $F = \{AB\}$-vel, így most $F = \{AB, AG\}$.
   A BG élet nem vesszük be, mert kört alkot $F = \{AB, AG\}$ éleivel.
   A CD élet bevesszük, mert nem alkot kört $F = \{AB, AG\}$-vel, így most $F = \{AB, AG, CD\}$.
   A BC élet bevesszük, mert nem alkot kört $F = \{AB, AG, CD\}$-vel, így most $F = \{AB, AG, CD, BC\}$.
   Az FG élet bevesszük, mert nem alkot kört $F = \{AB, AG, CD, BC\}$-vel, így most $F = \{AB, AG, CD, BC, FG\}$.
   A CG és DG éleket nem vesszük be, mert kört alkotnak $F = \{AB, AG, CD, BC, FG\}$ éleivel.
   A DE élet bevesszük, mert nem alkot kört $F = \{AB, AG, CD, BC, FG\}$-vel, így most $F = \{AB, AG, CD, BC, FG, DE\}$.
   A többi élet, EG, EF és AF, nem vesszük be, mert kört alkotnak
   $F = \{AB, AG, CD, BC, FG, DE\}$ éleivel.
   A végül kapott $F = \{AB, AG, CD, BC, FG, DE\}$ lesz a keresett minimális feszítőfa.

Kruskal algorimusának lépésszáma

Kruskal algoritmusának fenti leírása még nem elég pontos, hiányzik belőle annak a résznek a pontos leírása, hogy hogyan döntjük el, hogy az aktuális e él alkot-e kört az aktuális F halmazzal. Ezt természetesen meg lehet csinálni rendesen és ezen pontos leírás ismeretében beszélhetnénk a lépésszámról is.
Az első pont lépésszáma $O(e \log e) = O(e \log n)$ (mert e darab élet kell rendeznünk, ezt megtehetjük összefésüléses rendezéssel és mert $e \leq n^2$, így $\log e \leq \log n^2 = 2 \log n$).
A 2. pont lépésszáma szintén $O(e \log n)$, ha ügyesen nyilvántartjuk az F éleit és így el tudjuk dönteni, hogy egy e él alkot-e kört velük vagy sem. Ennek a részleteteit nem tárgyaljuk, csak azt kell tudni, hogy az egész eljárás megoldható $O(e \log n)$ lépésben.

Megjegyzés
Mind Kruskal, mind Prim algoritmusa minimális feszítőfát talál, de a két eljárás elve teljesen különböző: például Prim algoritmusában az F halmaz végig összefüggő, Kruskal eljárásában pedig csak az algorimus végén állíthatjuk ezt biztosan.