23. Minimális feszítőfák

Elektromos áramkörök tervezésénél gyakori feladat az, amikor elektromosan ekvivalenssé kell tennünk az áramkör bizonyos pontjait. Ahhoz, hogy n darab ilyen pont között kapcsolatot teremtsünk, elég, ha $n-1$ darab alkalmasan megválasztott pontpárt egy-egy vezetékszállal összekötünk. Mivel a pontoknak többféle ilyen összekapcsolása is lehet, kívánatos ezek közül azt megtalálnunk, amelyikhez összességében a legkevesebb (legrövidebb) vezetékre van szükség.

Ezt a huzalozási problémát egy összefüggő, irányítatlan $G = (V, E)$ gráf segítségével modellezhetjük, ahol V-beli csúcsok az áramkör pontjait, E élei az elvileg összeköthető pontpárokat jelölik, továbbá minden $(u, v) \in E$ él rendelkezik egy $w(u, v)$ súllyal, amely az u és v összekötésének költségét (az összekötéshez szükséges vezetékszál hosszát) adja meg. Célunk az, hogy megtaláljuk azt a körmentes $T \subseteq E$ részhalmazt, amelynek segítségével az összes pont összekapcsolható, és amelynek a

$$w(T) = \sum_{(u,v)\in T} w(u, v)$$

teljes súlya a lehető legkisebb. A körmentes és minden csúcspontot összekapcsoló T nyilvánvalóan egy fa, amelyik „átfogja", „kifeszíti" a G gráfot, ezért ***feszítőfának*** nevezzük. A T fa meghatározásának problémáját ***minimális feszítőfa problémának***[1] hívjuk. A 23.1. ábra egy összefüggő gráfra és annak minimális feszítőfájára mutat példát.

Ebben a fejezetben két, a minimális feszítőfa problémát megoldó algoritmust mutatunk be: Kruskal és Prim algoritmusát. Mindkettő futási ideje közönséges bináris kupacok használata mellett $O(E \lg V)$. Fibonacci-kupacok alkalmazásával a Prim-algoritmus futási ideje $O(E + V \lg V)$-re csökkenthető, amely azonban csak abban az esetben jelent javulást, ha $|V|$ sokkal kisebb, mint $|E|$.

Ez a két algoritmus a 16. fejezetben vázolt mohó algoritmusok közé tartozik. Általában egy algoritmus minden lépésben több lehetőség közül választ. A mohó stratégia ezek közül azt részesíti előnyben, amelyik az adott pillanatban a legjobbnak látszik. Az optimális megoldás megtalálását ez a stratégia általában nem garantálja. A minimális feszítőfa problémánál azonban bebizonyítható, hogy bizonyos mohó stratégiák minimális súlyú feszítőfához vezetnek. A most bemutatásra kerülő mohó módszerek a 16. fejezetben ismertetett

[1] A minimális feszítőfa kifejezés egy rövidített formája a minimális súlyú feszítőfa megnevezésnek. A T-beli élek számát például nem is minimalizálhatnánk, hiszen a B.2. tétel miatt minden feszítőfa pontosan $|V| - 1$ élből áll.

23.1. ábra. Egy összefüggő gráf minimális feszítőfája. Az élek súlyait az ábrán feltüntettük. A minimális feszítőfa éleit vastag vonalak jelölik. A fa teljes súlya 37. Nem ez az egyetlen minimális feszítőfa: cseréljük ki a (b, c) élt az (a, h) élre, és egy másik 37 súlyú feszítőfát kapunk.

elmélet klasszikus alkalmazásai, amelyeket azonban az említett fejezet elolvasása nélkül is megérthetünk.

A 23.1. alfejezetben egy olyan „általános" minimális feszítőfa algoritmust mutatunk be, amelyik fokozatosan, újabb és újabb élek hozzáadásával állítja elő a feszítőfát. A 23.2. alfejezetben ennek az általános algoritmusnak kétféle megvalósítását adjuk meg. Az első, Kruskaltól származó algoritmus, a 21.1. alfejezet összefüggő-komponensek algoritmusához hasonlít. A Prim nevével fémjelzett második algoritmus Dijkstra legrövidebb-utak algoritmusával (24.3. alfejezet) mutat közös vonásokat.

23.1. Minimális feszítőfa növelése

Egy minimális feszítőfát szeretnénk találni egy összefüggő, irányítatlan, $w : E \to \mathbf{R}$ súlyfüggvénnyel élsúlyozott $G = (V, E)$ gráfban. Az a két, mohó stratégiára épülő algoritmus, amelyet e probléma megoldására alkalmazunk ebben a fejezetben, abban különbözik egymástól, hogy eltérő módon alkalmazza ezt a stratégiát.

Ez a mohó stratégia azon az „általános" algoritmuson keresztül érthető meg, amely fokozatosan, újabb és újabb élek hozzáadásával növeli a minimális feszítőfát. Az algoritmus az éleknek azt az A-val jelölt halmazát kezeli, amelyre az alábbi invariáns állítás teljesül:

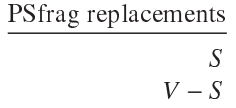
Az iterációk előtt az A valamelyik minimális feszítőfának a részhalmaza.

Az algoritmus minden lépésben azt az (u, v) élt határozza meg, amelyiket az A-hoz téve, továbbra is fennáll az előbbi invariáns állítás, miszerint $A \cup \{(u, v)\}$ is egy részhalmaza az egyik minimális feszítőfának. Egy ilyen élt A-ra nézve ***biztonságos élnek*** hívunk, mivel A-hoz történő hozzávétele biztosan nem rontja el az A-ra vonatkozó invariáns állítást.

```
MFF(G, w)
1  A ← ∅
2  while az A nem feszítőfa
3        do keresünk az A-ra nézve egy (u, v) biztonságos élt
4           A ← A ∪ {(u, v)}
5  return A
```

A ciklusinvariánst az alábbiak szerint használjuk:

Teljesül: Az 1. sor után az A magától értetődően elégíti ki a ciklusinvariánst.



23.2. ábra. Kétféleképpen mutatjuk be a 23.1. ábra gráfjának egy $(S, V - S)$ vágását. **(a)** Fekete színnel jelöltük az S halmazbeli, és fehérrel a $V - S$ halmazbeli csúcsokat. A vágást keresztező élek fehér csúcsokat kötnek össze feketékkel. A (d, c) él a vágást keresztező egyetlen könnyű él. Az élek egy A részhalmazát megvastagított vonalakkal jelöltük; a $(S, V - S)$ vágás elkerüli az A-t, mivel A egyik éle sem keresztezi a vágást. **(b)** Az előbbi gráfnak S-beli csúcsait a bal oldalon, $V - S$-beli csúcsait a jobb oldalon helyeztük el. Mindegyik vágást keresztező él egy bal oldali csúcsot egy jobb oldali csúccsal köt össze.

Megmarad: A 2–4. sorok ciklusa kizárólag biztonságos élekkel bővíti az A-t, így az invariáns továbbra is fenn áll.

Befejeződik: Minden A-hoz hozzávett él egy minimális feszítőfa része, ezért az 5. sorban visszaadott A halmaz egy minimális feszítőfa.

A trükkös rész természetesen a biztonságos él 3. sorban történő kiválasztása. Ilyen él biztosan létezik, hiszen a 3. sor végrehajtásakor az invariáns szerint van olyan T feszítőfa, hogy $A \subseteq T$. A **while** magjában az A egy valódi részhalmaza a T-nek, és ezért van olyan $(u, v) \in T$ él, amelyre $(u, v) \notin A$ és az (u, v) él biztonságos A-ra nézve.

A továbbiakban egy szabályt (23.1. tétel) adunk a biztonságos élek felismerésére. A következő alfejezetben azt a két algoritmust mutatjuk be, amelyek ezt a szabályt hatékonyan alkalmazzák.

Először néhány fogalmat vezetünk be. A $G = (V, E)$ irányítatlan gráf egy $(S, V - S)$ ***vágásának*** a V-nek egy kettéosztását nevezzük. A 23.2. ábra erre mutat egy példát. Azt mondjuk, hogy egy $(u, v) \in E$ él ***keresztezi*** az $(S, V - S)$ vágást, ha annak egyik végpontja S-ben, a másik $V - S$-ben található. Egy vágás ***elkerül*** egy A halmazt, ha az A egyetlen éle sem keresztezi a vágást. Egy vágást keresztező él ***könnyű él,*** ha a vágást keresztező élek közül neki van a legkisebb súlya. Megjegyezzük, hogy egynél több vágást keresztező könnyű él is lehet egyszerre. Általánosabban úgy is fogalmazhatunk, hogy egy él egy adott tulajdonságot kielégítő ***könnyű él,*** ha az adott tulajdonságot kielégítő élek között neki van a legkisebb súlya.

A biztonságos élt felismerő szabályt a következő tételből nyerjük.

23.3. ábra. A 23.1. tétel bizonyítása. Az S-beli csúcsok feketék, a $(V - S)$-beliek fehérek. Az ábrán csak a T minimális feszítőfa élei láthatók, a G gráf élei nem. Az A-beli éleket vastagított vonalak jelölik, és (u, v) egy könnyű él az $(S, V - S)$ vágásban. Az (x, y) él az egyetlen u-t v-vel összekötő T-beli p úton található. Az (u, v) élt tartalmazó T' minimális feszítőfát a T-ből kapjuk úgy, hogy eltávolítjuk belőle az (x, y) élt, és hozzávesszük az (u, v) élt.

23.1. tétel. *Legyen $G = (V, E)$ egy összefüggő, irányítatlan, a $w : E \to \mathbf{R}$ függvénnyel élsúlyozott gráf. Legyen A egy olyan részhalmaza E-nek, amelyik G valamelyik minimális feszítőfájának is része. Legyen $(S, V - S)$ tetszőleges A-t elkerülő vágása a G-nek. Legyen (u, v) az $(S, V - S)$ vágást keresztező könnyű él. Ekkor az (u, v) él biztonságos az A-ra nézve.*

Bizonyítás. Legyen T egy A-t tartalmazó minimális feszítőfa. Feltehetjük, hogy az (u, v) könnyű él nincs benne a T-ben, mert ha benne van, akkor máris készen vagyunk a bizonyítással. Kivág-beilleszt technikával meg fogjuk szerkeszteni azt a T' minimális feszítőfát, amelyik az $A \cup \{(u, v)\}$-t tartalmazza, ami azt mutatja, hogy az (u, v) él az A-ra nézve biztonságos.

Az (u, v) él az u és v csúcsokat T-ben összekötő p úttal együtt egy kört alkot, mint azt a 23.3. ábrán láthatjuk. Mivel u és v az $(S, V - S)$ vágás ellentétes oldalán helyezkedik el, kell lenni a p úton legalább egy olyan élnek, ami szintén keresztezi a vágást. Legyen egy ilyen él az (x, y). Az (x, y) nincs A-ban, mert A-t elkerüli a vágás. Mivel (x, y) az u-t és v-t összekötő egyetlen T-beli úton van, az (x, y)-t eltávolítva a T két komponensre esik szét. Az (u, v) hozzáadása azonban újra összekapcsolja ezt a két komponenst, és a $T' = T - \{(x, y)\} \cup \{(u, v)\}$ egy új feszítőfa lesz.

Most megmutatjuk, hogy T' egy minimális feszítőfa. Mivel (u, v) egy könnyű él az $(S, V - S)$ vágást keresztező élek között, és (x, y) szintén keresztezi ezt a vágást, a $w(u, v) \leq w(x, y)$ fennáll. Ennek megfelelően

$$w(T') = w(T) - w(x, y) + w(u, v) \leq w(T).$$

De T egy minimális feszítőfa, azaz $w(T) \leq w(T')$; így T'-nek is egy minimális feszítőfának kell lennie.

Már csak az maradt hátra, hogy megmutassuk, az (u, v) él biztonságos A-ra nézve. Tudjuk, hogy $A \subseteq T'$, hiszen $A \subseteq T$ és $(x, y) \notin A$; ezért $A \cup \{(u, v)\} \subseteq T'$. Következésképpen, mivel T' egy minimális feszítőfa, az (u, v) él biztonságos A-ra nézve. ■

A 23.1. tétel az általános MFF algoritmus jobb megértését segíti. Ennek végrehajtása során a $G = (V, E)$ összefüggő gráf éleiből felépített A mindig körmentes; máskülönben az A-t tartalmazó minimális feszítőfa is kört tartalmazna, ami nem lehetséges. A végrehajtás bármelyik pillanatában a $G_A = (V, A)$ gráf egy erdő, és a G_A minden összefüggő komponense fa. (Lehet, hogy némelyik fa csak egyetlen csúcsot tartalmaz, mint például abban az esetben, amikor az algoritmus elindul. Ekkor A üres, és az erdő $|V|$ darab egyetlen csúcsot tartalmazó fából áll.) Ezenkívül bármelyik A-ra nézve biztonságos (u, v) él G_A-nak két különböző komponensét köti össze, mivel $A \cup \{(u, v)\}$-nek körmentesnek kell lennie.

Az általános MFF algoritmus 2–4. sorainak ciklusa $|V| - 1$-szer hajtódik végre; minden lépésben egy újabb élét határozza meg a $|V| - 1$ élt tartalmazó minimális feszítőfának. Kezdetben, amikor $A = \emptyset$, $|V|$ darab fa van G_A-ban, és minden iteráció eggyel csökkenti ezek számát. Amikor az erdő már csak egyetlen fából áll, az algoritmus leáll.

A 23.2. alfejezetben szereplő két algoritmus a 23.1. tétel alábbi következményére támaszkodik.

23.2. következmény. *Legyen $G = (V, E)$ egy összefüggő, irányítatlan, a $w : E \to \mathbf{R}$ függvénnyel élsúlyozott gráf. Legyen A olyan részhalmaza E-nek, amelyik G valamelyik minimális feszítőfájának is része. Legyen C egy összefüggő komponens a $G_A = (V, A)$ erdőben. Ha (u, v) a C-t és a G_A valamely másik komponensét összekötő könnyű él, akkor (u, v) biztonságos az A-ra nézve.*

Bizonyítás. A $(C, V - C)$ elkerüli az A-t, és (u, v) egy könnyű él ebben a vágásban. Ezért az (u, v) biztonságos az A-ra nézve. ■

Gyakorlatok

23.1-1. Legyen (u, v) minimális súlyú él a G gráfban. Mutassuk meg, hogy (u, v) hozzátartozik a G valamelyik minimális feszítőfájához.

23.1-2. Sabatier professzor feltételezi a 23.1. tétel alábbi megfordítását. Legyen $G = (V, E)$ egy összefüggő, irányítatlan, a $w : E \to \mathbf{R}$ függvénnyel súlyozott gráf. Legyen A egy olyan részhalmaza E-nek, amelyik G valamelyik minimális feszítőfájának is része. Legyen $(S, V - S)$ egy tetszőleges A-t elkerülő vágása a G-nek. Legyen (u, v) egy biztonságos él az A-ra nézve az $(S, V - S)$ vágásban. Ekkor az (u, v) él könnyű az $(S, V - S)$ vágásban. Mutassuk meg egy számpéldán keresztül, hogy a professzor feltételezése helytelen.

23.1-3. Mutassuk meg, hogy ha az (u, v) él valamelyik minimális feszítőfához tartozik, akkor van olyan vágás, amelyben ez könnyű él.

23.1-4. Adjunk egyszerű példát olyan gráfra, amelyben azon élek halmaza, amelyek könnyűek valamely vágásban, nem alkot minimális feszítőfát.

23.1-5. Legyen e maximális súlyú él egy összefüggő $G = (V, E)$ gráf valamelyik körén. Bizonyítsuk be, hogy a $G' = (V, E - \{e\})$ gráfnak van olyan minimális feszítőfája, ami G-nek is minimális feszítőfája.

23.1-6. Mutassuk meg, hogy ha egy gráfnak bármelyik vágásában csak egyetlen könnyű él található, akkor a gráfnak egyetlen minimális feszítőfája van. Adjunk ellenpéldát arra, hogy az állítás visszafelé nem igaz.

23.1-7. Igaz-e, hogy ha egy gráf összes élsúlya pozitív, akkor az éleknek bármelyik olyan részhalmaza, amelyik az összes csúcsot összekapcsolja, és minimális összsúlyú, biztosan fát alkot? Adjunk példát, ahol az állítás nem teljesül nempozitív súlyok előfordulása esetén.

23.1-8. Legyen T minimális feszítőfa G-ben, és legyen L a T-beli élsúlyok rendezett listája. Mutassuk meg, hogy a G-nek bármely másik T' minimális feszítőfájára a T' élsúlyainak rendezett listája megegyezik az L listával.

23.1-9. Legyen T minimális feszítőfa G-ben, és legyen V' a V részhalmaza. Legyen T' a T-nek V' által meghatározott részgráfja, és legyen G' a G-nek V' által meghatározott részgráfja. Mutassuk meg, hogy ha T' összefüggő, akkor T' minimális feszítőfa G'-ben.

23.1-10. Legyen T minimális feszítőfa G-ben, és tegyük fel, hogy egy T-beli élnek csökkentjük a súlyát. Mutassuk meg, hogy T még ezután is minimális feszítőfa lesz G-ben. Formálisan: legyen T egy minimális feszítőfa abban a G-ben, amelynek súlyfüggvénye w. Válasszunk egy $(x, y) \in T$ élt és egy pozitív k számot, majd definiáljuk az alábbi w' súlyfüggvényt:

$$w'(u, v) = \begin{cases} w(u, v), & \text{ha } (u, v) \neq (x, y), \\ w(x, y) - k, & \text{ha } (u, v) = (x, y). \end{cases}$$

Mutassuk meg, hogy T egy minimális feszítőfa abban a G gráfban, amelynek súlyfüggvénye a w'.

23.1-11.$\star$ Legyen T minimális feszítőfa G-ben, és tegyük fel, hogy egy T-n kívül eső élnek csökkentjük a súlyát. Adjunk algoritmust, amely az így módosított gráfban megtalálja a minimális feszítőfát.

23.2. Kruskal és Prim algoritmusai

Ebben az alfejezetben a minimális feszítőfát előállító általános algoritmus két speciális változatát mutatjuk be. Mindkettő sajátos szabályt alkalmaz a biztonságos él meghatározására az általános MFF 3. sorában. Kruskal algoritmusában az A halmaz egy erdő. Az A-hoz hozzávett biztonságos él mindig az erdő két különböző komponensét összekötő legkisebb súlyú él. Prim algoritmusában az A egyetlen fa. Az A-hoz hozzáadott biztonságos él mindig a legkisebb súlyú olyan él, amelyik egy A-beli és egy A-n kívüli csúcsot köt össze.

Kruskal-algoritmus

Kruskal algoritmusa közvetlenül a 23.1. alfejezetben megadott általános MFF-algoritmuson alapul. Minden lépésben megkeresi a $G_A = (V, A)$ erdő két tetszőleges komponensét összekötő élek közül a legkisebb súlyú (u, v) élt, és ezt veszi hozzá az egyre bővülő erdőhöz. Legyen C_1 és C_2 az erdőnek az a két fája, amit az (u, v) él összeköt. Mivel (u, v) egyben a legkisebb súlyú olyan él is, amelyik C_1-et valamelyik másik komponenssel összeköti, a 23.2. következmény miatt (u, v) biztonságos C_1-re nézve. A Kruskal-algoritmus egy mohó algoritmus, mert minden lépésben a lehetséges legkisebb súlyú élt adja hozzá az erdőhöz.

Az általunk megvalósított Kruskal-algoritmus a 21.1. alfejezet összefüggő komponenseket előállító algoritmusához hasonlít. Az elemek diszjunkt halmazainak kezelésére egy diszjunkt-halmaz adatszerkezetet használ. Mindegyik halmaz az aktuális erdő egy fájának csúcsait tartalmazza. A HALMAZT-KERES(u) az u csúcsot tartalmazó halmazt jelölő elemet ad meg. Az, hogy az u és a v csúcsok ugyanahhoz a fához tartoznak-e, úgy határozható meg, hogy ellenőrizzük a HALMAZT-KERES(u) és HALMAZT-KERES(v) egyenlőségét. Két fa összekapcsolását az EGYESÍT eljárás végzi.

```
MFF-KRUSKAL(G, w)
1  A ← ∅
2  for minden v ∈ V[G]-re
3      do HALMAZT-KÉSZÍT(v)
4  rendezzük az E éleit a w súly szerint növekvő sorrendben
5  for minden (u, v) ∈ E élre, az élek súly szerint növekvő sorrendjében
6      do if HALMAZT-KERES(u) ≠ HALMAZT-KERES(v)
7           then A ← A ∪ {(u, v)}
8                UNIÓ(u, v)
9  return A
```

Kruskal algoritmusának működését a 23.4. ábra mutatja be. Az 1–3. sorban üres kezdőértéket kap az A halmazt, létrejön a gráf csúcsait külön-külön tartalmazó $|V|$ darab fa. A 4. sor súlyuk szerint növekvő sorba rendezi az E éleit. Az 5–8. sorok **for** ciklusa minden (u, v) élre megvizsgálja, hogy annak végpontjai ugyanahhoz a fához tartoznak-e. Ha igen, akkor ezt az élt nem lehet az erdőhöz hozzávenni anélkül, hogy kör alakulna ki, ezért nem kell vele foglalkozni. Ellenkező esetben a két csúcs különböző fákhoz tartozik. Ekkor az (u, v) élt a 7. sorban hozzávesszük az A-hoz, majd a két fa csúcsait a 8. sorban összevonjuk.

A $G = (V, E)$ gráfon működő Kruskal-algoritmus futási ideje a diszjunkt-halmaz megvalósításától függ. Vegyük a 21.3. alfejezetben megvalósított diszjunkt-halmaz erdőt a „rang szerinti egyesítés" és „úttömörítés" heurisztikákkal, mivel ez az aszimptotikusan leggyorsabbnak ismert megvalósítás. Ekkor az 1. sor kezdeti értékadása $O(1)$ ideig tart, a 4. sor rendezése pedig $O(E \lg E)$ idejű. (Mindjárt kitérünk a 2–3. sorbeli **for** ciklus $|E|$ darab HALMAZT-KÉSZÍT műveletének futási idejére is.) Az 5–8. sorok **for** ciklusa $O(E)$ HALMAZT-KERES és EGYESÍT műveletet tartalmaz a diszjunkthalmaz erdőre. Ezek teljes futási ideje a $|E|$ HALMAZT-KÉSZÍT művelettel együtt $O((V + E)\alpha(V))$, ahol α a 21.4. alfejezetben definiált nagyon lassan növekedő függvény. Mivel a G-ről feltettük, hogy összefüggő, fennáll, $|E| \geq |V|1$, és így a diszjunkthalmaz műveletek $O((E)\alpha(V))$ idejűek. Továbbá, mivel $\alpha(|V|) = O(\lg E)$, a Kruskal-algoritmus teljes futási ideje $O(E \lg E)$. Az $|E| < |V|^2$ mellett azt kapjuk, hogy $\lg |E| = O(\lg V)$, és ekkor a Kruskal-algoritmus teljes futási ideje $O(E \lg V)$ lesz.

Prim-algoritmus

Prim algoritmusa, akárcsak Kruskalé, a 23.1. alfejezet általános MFF algoritmusának speciális változata. A Prim-algoritmus működése nagyon hasonlít Dijkstra legrövidebb utakat kereső algoritmusáéhoz (lásd 24.3. alfejezet). Prim algoritmusában az A halmaz élei mindig egyetlen fát formáznak. A 23.5. ábrán is látható, hogy a fa építése egy tetszőlegesen kiválasztott r gyökérpontból indul, és addig növekszik, amíg a V összes csúcsa nem kerül be a fába. Minden lépésben azt a könnyű élt vesszük hozzá az A fához, amelyik egy A-beli csúcsot köt össze a $G_A = (V, A)$ egy izolált csúcsával. A 23.2. következmény szerint ez a szabály biztonságos élt ad A-ra nézve; így amikor az algoritmus befejeződik az A-beli élek minimális feszítőfát alkotnak. Ez a stratégia mohó, mivel minden lépésben egy olyan éllel bővíti a fát, amely a lehető legkisebb mértékben növeli a fa teljes súlyát.

A Prim-algoritmus hatékony megvalósításának kulcsa az A-hoz hozzáadandó új él kiválasztásának ügyes megvalósításán múlik. Az alább megadott pszeudokód bemenő adatai

PSfrag replacements
(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)

23.4. ábra. Kruskal algoritmusának működése a 23.01. ábra gráfján. A vastagított vonalakkal jelölt élek az egyre bővülő A erdőhöz tartoznak. Az algoritmus az éleket súlyuk szerint rendezett sorrendben vizsgálja meg. Egy nyíl mutat arra az élre, amellyel az adott lépésben az algoritmus dolgozik. Ha ez két különböző fát köt össze az erdőben, akkor ezt hozzávesszük az erdőhöz, miközben a két fát összevonjuk.

az összefüggő G gráf és az előállítandó minimális feszítőfa r gyökere. A fában még nem szereplő csúcsok a végrehajtás közben egy *kulcs* értéken alapuló Q elsőbbségi sorban helyezkednek el. A $kulcs[v]$ a vt valamelyik fabeli csúccsal összekötő minimális súlyú él súlya; amennyiben nincs ilyen él, akkor megállapodás szerint $kulcs[v] = \infty$. A $\pi[v]$ érték a v csúcs fabeli szülőjét adja meg. Az általános MFF A halmazát a Prim-algoritmus implicit módon az

$$A = \{(v, \pi[v]) : v \in V - \{r\} - Q\}$$

képlettel írja le. Amikor az algoritmus befejeződik, a Q elsőbbségi sor üres; a G-beli A minimális feszítőfa az

$$A = \{(v, \pi[v]) : v \in V - \{r\}\}$$

halmaz lesz.

MFF-PRIM(G, w, r)

```
 1  for minden v ∈ V[G]re
 2      do kulcs[v] ← ∞
 3         π[u] ← NIL
 4  kulcs[r] ← 0
 5  Q ← V[G]
 6  while Q ≠ ∅
 7        do u ← KIVESZ-MIN(Q)
 8           for minden v ∈ Szomszéd[u]-ra
 9               do if v ∈ Q and w(u, v) <kulcs[v]
10                     then π[v] ← u
11                          kulcs[v] ← w(u, v)
```

A Prim-algoritmus működését a 23.5. ábrán követhetjük nyomon. Az 1–5. sorok beállítják a csúcsoknál *kulcs* értéket ∞-re (kivéve az r gyökérpontot, amelyik *kulcs* értéke 0 lesz, és ezáltal ő lesz az elsőként feldolgozott csúcs), a szülő értéket NIL-re, és elhelyezik az összes csúcsot a Q minimalizált elsőbbségi sorban. Az algoritmus által karbantartott invariáns állítás a következő három részből tevődik össze:

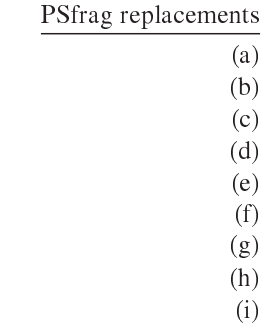
A 6–11. sorok **while** ciklus minden iterációja előtt

1. $A = \{(v, \pi[v]) : v \in V - \{r\} - Q\}$.
2. A minimális feszítőfában elhelyezett csúcsok a $V - Q$ halmazban vannak.
3. Minden $v \in Q$ csúcsra, ha $\pi[v] \neq$ NIL, akkor $key[v] < \infty$ és a $key[v]$ egy olyan $(u, \pi[v])$ könnyű él súlya, amelyik a v-t a minimális feszítőfában már elhelyezett valamelyik u csúccsal köti össze.

A 7. sor meghatározza azt az $u \in Q$ csúcsot, amely a $(V - Q, Q)$ vágás egyik könnyű élének végpontja (kivéve az első iterációt, ahol a 4. sornak megfelelően $u = r$). Eltávolítva az u-t a Q halmazból, az átkerül a fa csúcsait tartalmazó $V - Q$ halmazba, ezáltal az $(u, \pi[u])$ éllel módosul az A. A ciklusinvariáns harmadik részét a 8–11. sorok **for** ciklusa biztosítja úgy, hogy minden fán kívüli u-val szomszédos csúcsra időszerűsíti a *kulcs* és π értékeket.

A Prim-algoritmus hatékonysága a Q elsőbbségi sor megvalósításán múlik. Ha Q-t bináris minimum-kupacként (lásd 6. fejezet) valósítjuk meg, akkor a KUPACOT-ÉPÍT eljárással az 1–5. sorok kezdeti értékadásait $O(V)$ időben elvégezhetjük. A **while** ciklus magja $|V|$-szer kerül végrehajtásra, és mivel a KIVESZ-MIN művelet $O(\lg V)$ idejű, a KIVESZ-MIN művelet összes meghívása $O(V \lg V)$ ideig tart. A 8–11. sorok **for** ciklusának végrehajtási ideje $O(E)$, mivel a szomszédsági listák hosszainak összege $2|E|$. A **for** cikluson belül a 9. sorban egy elem Q-hoz tartozásának vizsgálata konstans időben történik, ha minden csúcsnál fenntartunk egy bitet annak jelzésére, hogy a csúcs Q-beli-e, vagy sem. Ezt akkor kell megváltoztatnunk, amikor a csúcs kikerül a Q-ból. A 11. sor értékadása magában hordozza a KULCSOT-CSÖKKENT művelet végrehajtását a kupacra, amelyet bináris kupac esetén $O(\lg V)$ időben lehet megvalósítani. Így a Prim-algoritmus teljes futási ideje $O(V \lg V + E \lg V) = O(E \lg V)$, amelyik aszimptotikusan egyenlő a Kruskal-algoritmusra adott megvalósítás futási idejével.

A Prim-algoritmus aszimptotikus futási ideje azonban Fibonacci-kupacok segítségével javítható. A 20. fejezetben láthatjuk, hogy ha $|V|$ darab elemet Fibonacci-kupacba szervezünk, akkor egy KIVESZ-MIN művelet futási ideje $O(\lg V)$-re csökkenthető, és a (11. sor meg-



23.5. ábra. Prim algoritmusának működése a 23.1. ábra gráfján. A gyökérpont a. A vastagított vonalakkal jelölt élek és a feketére színezett csúcsok az egyre bővülő fát mutatják. A fa csúcsai az algoritmus minden lépésében egy vágást határoznak meg, és ezt a vágást keresztező könnyű élt vesszük hozzá a fához. Például a második lépésben az algoritmus a (b, c) vagy az (a, h) élek valamelyikét adja hozzá a fához, mivel mindkettő könnyű éle a vágásnak.

valósításában szereplő) KULCSOT-CSÖKKENT művelet ideje $O(1)$ lesz. Mindent egybevetve a Fibonacci-kupaccal megvalósított elsőbbségi sort használó Prim-algoritmus futási ideje $O(E + V \lg V)$.

Gyakorlatok

23.2-1. Kruskal algoritmusa ugyanazon G gráf esetén eltérő feszítőfákat eredményezhet attól függően, hogy az azonos súlyú éleket hogyan rendeztük sorba. Mutassuk meg, hogy

minden T minimális feszítőfához megadható olyan rendezés, hogy az algoritmus éppen a T-t állítja elő.

23.2-2. Tegyük fel, hogy a $G = (V, E)$ gráfot egy szomszédsági mátrixban ábrázoljuk. Adjunk ekkor $O(V^2)$ futási idejű megvalósítást Prim algoritmusára.

23.2-3. Vajon aszimptotikusan gyorsabb-e a Prim-algoritmus Fibonacci-kupacos megvalósítása a bináris kupacos változatnál ritka $G = (V, E)$ gráf esetén, ahol $|E| = \Theta(V)$? Mit mondhatunk a sűrű gráfokról, ahol $|E| = \Theta(V^2)$? Milyen kapcsolatnak kell lenni a $|E|$ és $|V|$ között ahhoz, hogy a Fibonacci-kupacos megvalósítás aszimptotikusan gyorsabb legyen a bináris kupacos változatnál?

23.2-4. Tegyük fel, hogy egy gráf súlyai 1 és $|V|$ közé eső egész számok. Milyen gyorssá tehető a Kruskal-algoritmus? Mit mondhatunk arról az esetről, amikor a súlyok 1 és W közé esnek, ahol W állandó?

23.2-5. Tegyük fel, hogy egy gráf súlyai 1 és $|V|$ közé eső egész számok. Milyen gyorssá tehető a Prim-algoritmus? Mit mondhatunk arról az esetről, amikor a súlyok 1 és W közé esnek, ahol W állandó?

23.2-6.⋆ Tegyük fel, hogy egy gráf élsúlyai egyenletesen helyezkednek el a $[0, 1)$ félig nyílt intervallumon. Kruskal vagy Prim algoritmusa a gyorsabb ilyenkor?

23.2-7.⋆ Tegyük fel, hogy a G gráf minimális feszítőfáját már előállítottuk. Milyen gyorsan lehet módosítani ezt a fát, ha G-hez egy új csúcsot és ahhoz kapcsolódó éleket veszünk hozzá?

23.2-8. Toole professzor az alábbi új oszd-meg-és-uralkodj stratégiájú algoritmust tervezi minimális feszítőfák előállítására. Adva van egy $G = (V, E)$ gráf, amelyben a csúcsok V halmazát két részre, V_1 és V_2 halmazokra vágjuk úgy, hogy a $|V_1|$ és $|V_2|$ különbsége legfeljebb egy. Legyen E_1 azon élek halmaza, amelyek V_1-beli csúcsokat kötnek össze, E_2 pedig azon élek halmaza, amelyek végpontjai V_2-beli csúcsok. Oldjuk meg rekurzív módon a minimális feszítőfa problémát mindegyik $G_1 = (V_1, E_1)$ és $G_2 = (V_2, E_2)$ részgráfra. Ezután keressük meg azt a legkisebb súlyú E-beli élet, amelyik keresztezi a V_1, V_2 vágást, és ezzel az éllel kössük össze az előbb talált minimális feszítőfákat.

Vagy mutassuk meg, hogy az így kapott feszítőfa minimális a G-ben, vagy adjunk ellenpéldát arra, amikor nem az.

Feladatok

23-1. Második legjobb feszítőfa

Legyen $G = (V, E)$ irányítatlan, összefüggő gráf a $w : E \to \mathbf{R}$ súlyfüggvénnyel, és tegyük fel, hogy $|E| \geq |V|$, és az élsúlyok mind különböznek.

A második legjobb feszítőfát az alábbiak szerint definiáljuk. Legyen $\mathcal{T}$ a G összes feszítőfájának halmaza, és T' egy minimális feszítőfa G-ben. A második legjobb feszítőfa egy olyan feszítőfa, amelyre $w(T) = min_{T'' \in \mathcal{T} - \{T'\}}\{w(T'')\}$.

a. Mutassuk meg, hogy a minimális feszítőfa egyértelmű, de a második legjobb feszítőfa nem szükségképpen az.

b. Legyen T egy minimális feszítőfa G-ben. Bizonyítsuk be, hogy léteznek olyan $(u, v) \in T$ és $(x, y) \notin T$ élek, hogy a $T - \{(u, v)\} \cup \{(x, y)\}$ egy második legjobb feszítőfája a G-nek.

c. Legyen T feszítőfája a G-nek, és tetszőleges $u, v \in V$ csúcsok esetén a $max[u, v]$ legyen a T-ben az u-t v-vel összekötő egyetlen út legnagyobb súlyú élének a súlya. Adjunk olyan $O(V^2)$ idejű algoritmust, amelyik adott T-re minden $u, v \in V$ csúcs pár esetén kiszámolja a $max[u, v]$-t.

d. Adjunk egy hatékony algoritmust a második legjobb feszítőfa előállítására.

23-2. Minimális feszítőfa ritka gráfokban

Egy nagyon ritka összefüggő $G = (V, E)$ gráf esetén tovább javíthatjuk a Fibonacci-kupacokat használó Prim-algoritmus $O(E + V \lg V)$ futási idejét, ha a Prim-algoritmus előtt a G egy alkalmas előfeldolgozásával csökkentjük a csúcsok számát. Nevezetesen minden u csúcshoz választunk egy hozzákapcsolódó minimális súlyú (u, v) élet, amelyet beteszünk a fokozatosan felépülő minimális feszítőfába. Ezután összevonunk minden kiválasztott élt (lásd B.4. alfejezet). Ahelyett, hogy ezeket az éleket egyesével vonnánk össze, először meghatározzuk azoknak a csúcsoknak a halmazait, amelyeket egyazon új csúccsá egyesítünk. Utána létrehozzuk azt a gráfot, amelyet az élek egyesével történő összevonásával kapnánk, de az éleket a végpontjaikat tartalmazó halmazok alapján átnevezzük. Így az eredeti gráf számos élét ugyanúgy fogjuk hívni. Az azonos nevű élek közül annak a neve marad meg, amelyiknek a súlya a megfelelő eredeti élek súlyai közül a legkisebb.

MFF-VÁZ($G, kapocs, c, T$)

```
 1  for minden v ∈ V[G]-re
 2      do volt[v] ← HAMIS
 3         HALMAZT-KÉSZÍT(v)
 4  for minden u ∈ V[G]-re
 5      do if volt[u] = HAMIS
 6            then legyen v ∈ Szomszéd[u] a legkisebb c[u, v] értékű csúcs
 7                 EGYESÍT(u, v)
 8                 T ← T ∪ {kapocs[u, v]}
 9                 volt[u] ← volt[v] ← IGAZ
10  V[G'] ←{HALMAZT-KERES(v): v ∈ V[G]}
11  E[G'] ← ∅
12  for minden (x, y) ∈ E[G]-re
13      do u ← HALMAZT-KERES(x)
14         v ← HALMAZT-KERES(y)
15         if (u, v) ∉ E[G']
16            then E[G'] ← E[G'] ∪ {(u, v)}
17                 kapocs'[u, v] ← kapocs[x, y]
18                 c'[u, v] ← c[x, y]
19            else if c[x, y] < c'[u, v]
20                    then kapocs'[u, v] ← kapocs[x, y]
21                         c'[u, v] ← c[x, y]
22  Felépítjük a G' Szomszéd szomszédsági listáját
23  return G', kapocs', c' és T
```

Kezdetben a T minimális feszítőfa üres, és minden $(u, v) \in E$ élre a $kapocs[u, v] = (u, v)$, és $c[u, v] = w(u, v)$. A *kapocs* tulajdonság a kezdeti gráf élei és az összevont gráf élei közötti

kapcsolatot adja meg. A c tulajdonság egy él súlyát jelzi, és amikor összevonunk éleket, c-t az élsúlyok választására vonatkozó fenti séma szerint kell módosítani. Az MFF-váz eljárás bemenő adatként a G, *kapocs*, c és T értékeket kapja meg, és visszaadja az összevont G' gráfot, valamint a G' gráfban a módosított *kapocs'* és c' tulajdonságokat. Az eljárás ezenkívül összegyűjti azokat a G-beli éleket, amelyek a minimális feszítőfához tartoznak.

a. Legyen T az MFF-váz által visszaadott élek halmaza, és legyen A a G' gráf minimális feszítőfája, amelyet a MFF-PRIM(G', c', t) állít elő, ahol r egy tetszőleges $V[G']$-beli csúcs. Bizonyítsuk be, hogy a $T \cup \{kapocs[x, y] : (x, y) \in A\}$ egy minimális feszítőfája a G-nek.

b. Igaz-e, hogy $|V[G']| \leq |V|/2$?

c. Mutassuk meg, hogyan valósítható meg az MFF-váz úgy, hogy a futási ideje $O(E)$ legyen. (*Útmutatás.* Használjon egyszerű adatszerkezetet.)

d. Tegyük fel, hogy k-szor egymás után lefuttatjuk az MFF-váz eljárást úgy, hogy az egyik menet eredményeként előállt G', *kapocs'* és c' a következő menetnek a G, *kapocs* és c bemenete legyen, miközben a kezdetben üres T-ben felhalmozzuk az éleket. Igaz-e, hogy a k menet teljes futási ideje $O(kE)$?

e. Tegyük fel, hogy a (d) részben látott k menetes MFF-váz után a Prim-algoritmust a MFF-PRIM$(G', kapocs', c')$ meghívásával hajtjuk végre, ahol a G' és c' az utolsó menetben kapott eredmények, az r pedig egy tetszőleges $V[G']$-beli csúcs. Mutassuk meg, hogyan lehet úgy megválasztani k-t, hogy a teljes futási idő $O(E \lg \lg V)$ legyen. Igaz-e, hogy a k értékének ez a megválasztása a minimális teljes aszimptotikus futási időt eredményezi?

f. Az $|E|$ milyen ($|V|$ segítségével kifejezett) értékére lesz aszimptotikusan jobb az előfeldolgozásos Prim-algoritmus az előfeldolgozás nélkülinél?

23-3. Üvegnyakú feszítőfa

Egy G irányítatlan gráf T ***üvegnyakú feszítőfája*** egy olyan feszítőfája a G-nek, amelynek maximális élsúlya a legkisebb a G összes feszítőfája között. Azt mondjuk, hogy az üvegnyakú feszítőfa értéke a T-beli legnagyobb élsúly.

a. Igaz-e, hogy egy minimális feszítőfa egyben üvegnyakú is.

Az (a) rész azt mutatja, hogy egy üvegnyakú feszítőfát nem nehezebb megtalálni, mint egy minimális feszítőfát. A további részekben azt is megmutatjuk, hogy ehhez lineáris idő elég.

b. Adjunk egy olyan lineáris idejű algoritmust, amely adott G gráf és b egész szám esetén meghatározza, hogy van-e legfeljebb b értékű üvegnyakú feszítőfa.

c. Használjuk fel a (b) rész algoritmusát alprogramként az üvegnyakú feszítőfa probléma megoldó algoritmusában. (*Útmutatás.* Használhat egy olyan alprogramot, amelyik a 23-2. feladatbeli MFF-váz eljáráshoz hasonlóan összevonja az élek halmazait.)

23-4. Alternatív minimális feszítőfa algoritmus

Ebben a feladatban három különböző algoritmus pszeudo kódját adjuk meg. Mindegyik egy gráfot kap bemenetként és egy T élhalmazt ad vissza. Mindegyik algoritmusnál el kell dönteni, hogy a T minimális feszítőfa vagy nem. Mindegyik algoritmusnál – akár talál minimális feszítőfát, akár nem – vázolni kell a leghatékonyabb megvalósítást.

a. LEHET-MFF-A(G, w)

```
1  az élek nemnövekvő sorrendű rendezése a w súlyaik alapján
2  T ← E
3  for minden súlyaik szerinti nemnövekvő sorrendben vett e élre
4      do if T − {e} egy összefüggő gráf
5          then T ← T − {e}
6  return T
```

b. LEHET-MFF-B(G, w)

```
1  T ← ∅
2  for minden tetszőleges sorrendben vett e élre
3      do if T ∪ {e} nem tartalmaz kört
4          then T ← T − {e}
5  return T
```

c. LEHET-MFF-C(G, w)

```
1  T ← ∅
2  for minden tetszőleges sorrendben vett e élre
3      do if T ∪ {e} tartalmaz egy c kört
4          then legyen e′ egy maximális súlyú él a c-n
5               T ← T − {e′}
6  return T
```

Megjegyzések a fejezethez

Tarjan [**?**] kitűnő értekezésben tekinti át a minimális feszítőfa problémát. A minimális feszítőfa probléma történetét Graham és Hell [**?**] írta meg.

Tarjan az első minimális feszítőfa algoritmust O. Borůvka 1926-ban megjelent cikkének tulajdonítja. Borůvka algoritmusa a 23-2. feladatban leírt MFF-váz eljárás $O(\lg V)$ számú iterációból áll. A Kruskal-algoritmust 1956-ban Kruskal [**?**] adta meg. A Prim-algoritmusként ismert algoritmust valóban Prim [**?**] találta ki, de korábban, 1930-ban már V. Jarník is felfedezte.

Annak az oka, hogy a mohó algoritmus hatékonyan keres minimális feszítőfát az, hogy egy gráf erdőinek halmaza egy matroidot alkot. (Lásd 16.4. alfejezet.)

Amikor $|E| = \Omega(V \lg V)$, a Fibonacci-kupacok segítségével megvalósított Prim-algoritmus $O(E)$ futási idejű. Ritka gráfokra, a Prim-algoritmus, a Kruskal-algoritmus és Borůvka algoritmusának ötleteit, valamint fejlett adatszerkezeteket felhasználva, Fredman és Tarjan [**?**] adott egy $O(E \lg^* V)$ futási idejű algoritmust. Gabow, Galil, Spencer és Tarjan [**?**] ezt az algoritmust javította $O(E \lg \lg^* V)$ idejűre. Chazelle [**?**] adott egy $O(E\, \widehat{\alpha}(E, V))$ futási idejű algoritmust, ahol $\widehat{\alpha}(E, V)$ az Ackermann-függvény inverz függvénye. (Lásd a 21. fejezet megjegyzéseit az Ackermann függvény és annak inverzének elemzésénél.) Eltérően az előző minimális feszítőfa algoritmusoktól, Chazelle algoritmusa nem a mohó stratégiát követi.

Egy kapcsolódó probléma a ***feszítőfa ellenőrzés,*** amelyben adott egy $G = (V, E)$ gráf és egy $T \subset E$ fa, és azt kell eldönteni, hogy T egy minimális feszítőfa-e a G-ben. King [**?**]

ad egy olyan lineáris idejű algoritmust a feszítőfa ellenőrzésre, amely Komlós [**?**] és Dixon, Rauch és Tarjan [**?**] korábbi munkáira épül.

A fenti algoritmusok mind determinisztikusak és a 8. fejezetben leírt összehasonlítás alapú modellek közé tartoznak. Karger, Klein és Tarjan [**?**] ad egy véletlenített minimális feszítőfa algoritmust, amelyik várható futási ideje $O(V + E)$. Ez az algoritmus, hasonlóan a 9.3 alfejezet lineáris idejű kiválasztás algoritmushoz, rekurziót alkalmaz: egy rekurzív hívás azt a segédproblémát oldja meg, amely tetszőleges minimális feszítőfához nem tartozó élek E' halmazát határozza meg. Ezután az $E - E'$-re tett másik rekurzív hívással meghatározza a minimális feszítőfát. Ez az algoritmus Borůvka algoritmusának és King feszítőfa ellenőrző algoritmusának ötleteit is felhasználja.

Fredman és Willard [**?**] megmutatta, hogyan lehet $O(E+V)$ idő alatt minimális feszítőfát találni egy nem összehasonlítás alapú determinisztikus algoritmussal. Algoritmusuk feltételezi, hogy az adatok b bites egész számok, és a számítógép memóriája b bites szavanként címezhető.

Tárgymutató

Névmutató