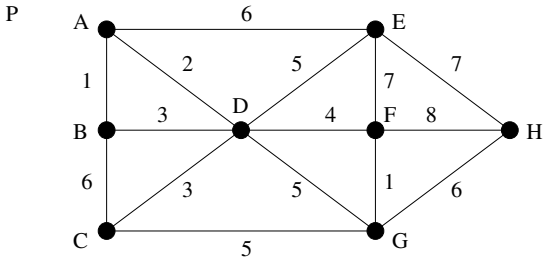
Algoritmusok és gráfok
TIZENKETTEDIK GYAKORLAT, 2019. december 13.
Megoldások

2. (a) Melyik éleket választja be és milyen sorrendben Prim algoritmusa az A csúcsból indulva? Hogyan változik a futás során lépésről lépésre a LEFEDVE halmaz és az F fa?
(b) Melyik éleket választja be és milyen sorrendben Kruskal algoritmusa?



Megoldás
(a) A LEFEDVE halmazba ebben a sorendben kerülnek be a csúcsok: A, B, D, C, F, G, E, H
Az F feszítőfába ebben a sorendben kerülnek be az élek: AB, AD, DC, DF, FG, DE, GH

3. **(Vizsga 2018)** Az alábbi irányítatlan G gráfban (ahol az x élsúly nem ismert) futtatjuk az a csúcsból a Prim algoritmust és azt tapasztaljuk, hogy az ab, bd, de, bc élek kerülnek be a minimális feszítőfába, ebben a sorrendben. Bizonyítsa be, hogy x értéke csak 2 lehet.
G élei: $ab(1), ac(x), bc(2), bd(x), cd(3), ce(4), de(1)$.

Megoldás Amikor az algoritmus beválasztja az x súlyú bd élet, akkor a LEFEDVE halmazban a és b van, azaz ekkor választhatta volna a 2 súlyú bc élet is, de nem tette, vagyis $x \leq 2$.
Amikor az algoritmus beválasztja a 2 súlyú bc élet, akkor a LEFEDVE halmazban a, b, d, e van, azaz ekkor választhatta volna az x súlyú ac élet is, de nem tette, vagyis $x \geq 2$.

4. A szoftverpiacon n féle grafikus formátum közötti oda-vissza konverzióra használatos programok kaphatók: az i-edik és a j-edik között oda-vissza fordító program ára a_{ij}, futási ideje pedig t_{ij} (ha létezik).

   (a) Javasoljunk módszert annak megtervezésére, hogy minden egyes formátumról a saját grafikus terminálunk által megértett formátumra a lehető leggyorsabban konvertáljunk! (Az ár nem számít.)

   (b) Javasoljunk módszert annak eldöntésére, hogy mely programokat vásároljuk meg, ha azt szeretnénk a lehető legolcsóbban megoldani, hogy a megvett programok segítségével bármelyik formátumról bármelyik más formátumra képesek legyünk konvertálni. (Itt a futási idő nem számít).

Megoldás (a) Ebben az esetben az idő számít, tehát egy olyan gráfot készítek, ahol a csúcsok a formátumok és az élek súlyai (ha van két formátum között program) a t_{ij} futási idő. Most minket csak az érdekel, hogy a mi formátumunkra mennyi a leggyorsabb konverzió minden más formátumra, ahol egy több lépéses konvertálás ideje az élek súlyainak összege, vagyis a legrövidebb utat keressük a mi csúcsunkból az összes többibe ebben a gráfban pl. Dijkstra algoritmussal és azokat a programokat veszem meg, amik ebben fában szerepelnek.
(b) Ebben az esetben az ár számít, tehát egy olyan gráfot készítek, ahol a csúcsok a formátumok és az élek súlyai (ha van két formátum között program) az a_{ij} ár. Most minket csak az érdekel, hogy bárhonnan bárhova lehessen konvertálni, vagyis a kiválasztott élek (kiválasztott programok élei) összefüggő, minden csúcsot (minden formátumot lefedő) gráfot alkossanak. Mivel a lehető legolcsóbb megoldást keressük, ezért kört nem akarunk (ott egy él biztos felesleges), tehát egy feszítőfát akarunk és abból is a legolcsóbbat, vagyis erre Prim algoritmusa lesz jó és azokat a programokat veszem meg, amik ebben a fában szerepelnek.

5. **(Vizsga 2018)** Mátrixával adott egy város úthálózatának élsúlyozott, irányított gráfja: a csúcsok a csomópontok, az élek a csomópontok közötti közvetlen utak, az élek súlya pedig azt mutatja, hogy mennyi az átlagos idő, ami az út megtételéhez autóval szükséges.

   Útfelújítások miatt a következő héten le fogják zárni a város két csomópontját, a-t és b-t (ezeken nem lehet autóval áthaladni). Adott a gráfban két kijelölt csúcs, S és T és azt szeretnénk eldönteni, hogy az a és b csomópontok lezárása miatt növekedni fog-e az S-ből T-be eljutás ideje és ha igen, akkor mennyivel. (Tételezzük fel, hogy a közvetlen utakhoz rendelt átlagos idők nem változnak a lezárások következtében.)

Melyik tanult algoritmust lehet alkalmazni, hogyan és miért, ha $O(n^2)$ lépésben meg akarjuk oldani ezt a feladatot (a szokásos módon n a csomópontok számát jelöli)?

Megoldás

Az algoritmus elve: Először lefuttatjuk Dijkstra algoritmusát S-ből az eredeti gráf mátrixával, ebből megtudjuk, hogy mekkora a távolság (az átlagos eljutási idő a legrövidebb úton) S-ből T-be a lezárás előtt.
Ezután módosítjuk a mátrixot úgy, hogy az a és b csúcsok minden élét kitöröljük, majd ebben az új gráfban futtatjuk a Dijkstra algoritmust újra S-ből. Így megtudjuk, hogy mennyi a lezárás után az átlagos eljutási idő T-be.
Megnézzük a két, T-re kapott értéket és ha nem ugyanaz, akkor nő az eljutási idő annyival, amennyivel a második érték nagyobb az elsőnél.
Pontosabban a mátrix módosítása: A szomszédossági mátrix a-hoz és b-hez tartozó sorainak minden elemét ∞-re állítjuk, ugyanígy az oszlopokat is:

```
ciklus j =1-től n-ig:
    A[a,j] := végtelen és A[b,j] := végtelen
ciklus vége

ciklus i =1-től n-ig:
    A[i,a] := végtelen és A[i,b] := végtelen
ciklus vége
```

Jóság: Az új mátrixnak megfelelő gráfban pontosan azok az utak vannak maradnak benne, amik nem mennek át se a-n, se b-n, az ezek között talált legrövidebb út lesz a legrövidebb, a lezárás mellett is járható út.

Lépésszám: A mátrix módosítása $O(n)$, mert mindkét ciklus n-szer fut le, a mag pedig konstans lépés, vagyis a módosítás $O(n) + O(n) = O(n)$. A két Dijkstra $O(n^2)$, mert mindkettő egy n csúcsú gráfon fut, azaz az egész algoritmus $O(n^2) + O(n) + O(n^2) + O(1)$ (itt az $O(1)$ a végén a két, T-re kapott érték különbségének kiszámolása).

6. **(Vizsga 2018)** Mátrixával adott egy város úthálózatának összefüggő, élsúlyozott, irányítatlan, egyszerű gráfja: a csúcsok a csomópontok, az élek a csomópontok közötti közvetlen utak, az élek súlya pedig azt mutatja, hogy hány hómunkás tudja az adott útszakaszt letakarítani 1 óra alatt. Szeretnénk tudni, hogy legalább mennyi hómunkásra van szükség összesen ahhoz, hogy egy éjszakai hóesés után (ami reggel 6-kor elállt), a 7 órás munkakezdés után 1 órán belül a főtérről (ami egy csúcs a gráfban) a város összes csomópontja elérhető legyen letakarított úton.
Melyik tanult algoritmust lehet alkalmazni, hogyan és miért, ha $O(n^2)$ lépésben választ akarunk kapni, ahol n a csomópontok száma?

Megoldás

Arra van szükségünk, hogy néhány utat letakarítsunk úgy, hogy a letakarított részek éleiből álló gráf egy összefüggő gráf legyen, ami minden csúcsot lefed, mert ezt jelenti gráfos nyelven az, hogy bárhonnan bárhova el lehet jutni letakarított utakon át. Mivel egy ilyen választás költsége a benne szereplő élekhez tartozó hómunkások száma, vagyis a szereplő élek össszege, ezért a lehető legkisebb élsúlyú megoldást keressük, ha a lehető legkevesebb munkást akarjuk.
A legolcsóbb élkiválasztásban biztos nem lesz kör, mert ha lenne kör, akkor annak az egyik élét nem kellett volna letakarítani, enélkül is összefüggő maradt volna a letakarított rész. Ezt azt jelenti, hogy amit keresünk az egy minimális feszítőfa a gráfban.
Erre Prim algoritmusát tudjuk használni, ami pont $O(n^2)$ lépéssben fut.