Kombinatorikus optimalizálás 2023. tavasz

7. gyakorlat — 2023. április 27.

Gyakorlatok

1. Tegyük fel, hogy a G összefüggő gráfnak egyetlenegy elvágó pontja van. Tegyük fel továbbá, hogy a G komplementerének minden e élhez ismert az e kiépítésének költsége. Tervezzünk olyan hatékony algoritmust, amely az iménti inputhoz megtalál egy olyan minimális költségű élhalmazt, melynek kiépítésétől a kapott gráf 2-összefüggővé válik, azaz nem lesz elvágó pontja.

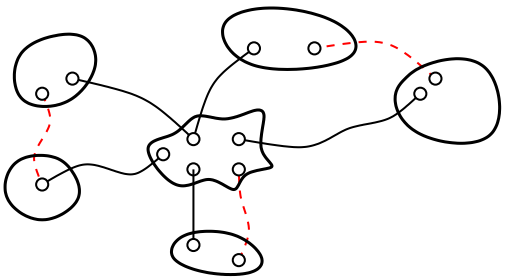
   Mivel G összefüggő, ezért azt kell elérni, hogy ne legyen elvágó pontja. Ezt pedig úgy tudjuk megtenni, a v elvágó pont elhagyásával keletkező komponensek közé addig húzunk be éleket, amíg összefüggővé nem válik az a gráf, aminek csúcsai e komponensek, élei pedig a behúzott élek. Ebből a szempontból kizárólag annak van jelentősége, hogy $G-v$ melyik komponenspárjai között futnak az élek, és az érdektelen, hogy az él a két adott komponensnek konkrétan melyik csúcsait köti össze. Két komponenst szintén nincs értelme két éllel összekötni.

   Ezért minden komponenspárra megkeressük az őket összekötő legolcsóbb élt, és csak ezekkel foglalkozunk. Így a feladat nem más, mint a $G-v$ komponensei közé a lehető legolcsóbb módon kell éleket behúzni úgy, hogy összefüggő gráfot kapjunk. Ebben a formában azonban ismerős a feladat, épp a minimális költségű feszítőfa kereséséről van szó. Az eljárásunk tehát az, hogy $G-v$ komponensein, mint csúcsokon egy Kruskal algoritmust futtatunk, ahol két komponens között futó él költsége a két komponens pontjait összekötő élek legolcsóbbikának költsége lesz. □

2. Tegyük fel, hogy a G összefüggő gráfnak van olyan 2-komponense, amely G minden elvágó élének tartalmazza valamelyik végpontját. Hogyan függ a G 2-komponenseinek számától azon élek minimális száma, amelyeket behúzva G-be a kapott gráf 2-élösszefüggővé válik?

   A feladatbeli feltétel azt jelenti, hogy a szóban forgó 2-komponensen kívül G minden más 2-komponense levélkomponens, és G minden elvágó éle mentén egy-egy ilyen levélkomponens kapcsolódik a szóban forgó 2-komponenshez. Az órán tanult tétel szerint a 2-élöf tulajdonság eléréséhez szükséges élek minimális száma $\left\lceil \frac{\ell(G)+2\ell'(G)}{2} \right\rceil$, és a mi esetünkben $\ell'(G) = 0$, $\ell(G)$ pedig G elvágó éleinek száma, legalábbis akkor, ha a „központi” 2-komponens nem levélkomponens. Ennek megfelelően a formula nem más, mint $\lceil \frac{\ell}{2} \rceil$, ahol ℓ a G elvágó éleinek száma. Ha pedig a központi 2-komponens is levélkomponens, akkor ebből is pontosan egy elvágó él indul ki, amikoris G-nek összesen egy elvágó éle, és 2 db levél 2-komponense van, ám a formula szerencsére ekkor is helyes, hiszen egy élre van szükség a 2-élöf tulajdonság eléréséhez, és a formula is ennyit ad.

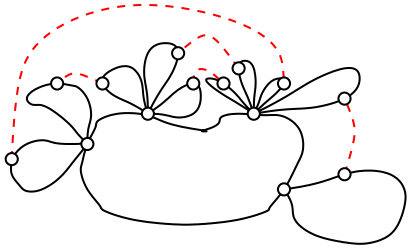
   Egyébként az órai tétel alkalmazása nélkül is látszik, hogy $\frac{\ell}{2}$ él behúzásával elérhető a 2-élöf tulajdonság. Ha ugyanis a levélkomponenseket összepárosítom és a párokat a egy-egy él mentén összekötjük, majd az esetlegesen kimaradó párosítatlan levélkomponenst (mondjuk) a központi komponenshez kötjük be, akkor épp $\lceil \frac{\ell}{2} \rceil$ élt használunk, amiknek a behúzása után nem marad elvágó él, tehát G-t csakugyan 2-élöf-vé tettük. □



3. Tegyük fel, hogy a G összefüggő gráfnak van olyan maxblokkja, amely tartalmazza G minden elvágó pontját. Tervezzünk olyan hatékony algoritmust, amely az iménti inputhoz megtalál egy olyan minimális méretű élhalmazt, melynek kiépítésétől a kapott gráf 2-szeresen (pont)összefüggővé válik. (A G gráf maxblokkjai a G olyan összefüggő maximális részgráfjai, amelyek nem tartalmaznak elvágó pontot.)

   A szóban forgó maxblokkon kívül minden más maxblokk levél, és a behúzandó élek száma legalább a levélblokkok számának fele, azaz legalább $\left\lceil \frac{m(G)}{2} \right\rceil$. Azonban ugyanazon az elvágó ponton több levélblokk is kapcsolódhat, így figyelni kell arra is, hogy legalább $b(G)-1$ élre van szükség. Az eljárás ezért a következő. Ha a levélblokkok összepárosíthatók úgy, hogy minden pár két tagja különböző elvágó pontokhoz tartozzék (esetleg egy kimaradó maxblokk megengedésével, akkor a párok közé éleket behúzva, a kimaradó maxblokkot pedig tetszőleges másik maxblokkhoz hozzákötve (de nem az elvágó pontjából induló éllel) $\left\lceil \frac{m(G)}{2} \right\rceil$ élt használunk. Ez optimális megoldás.

   Ha viszont nem lehet a levélblokkokat így párosítani, annak az oka, hogy $b(G)$ több mint fele a levélkomponensek számának. Ekkor a $b(G)-1$ db azonos elvágó pontra illeszkedő levélblokkból annyit, amennyit lehet más elvágó ponton üló maxblokkal kötünk össze, a $b(G)$ maxblokkból ezek után megmaradó k maxblokkot pedig pedig úgy kötözgetem össze $k-1$ éllel, hogy az elvágó pont elhagyásával keletkező részeik összefüggő gráfot alkossanak. (Tkp egy egy fát építek az elvágó pont elhagyásával keletkező komponenseken.) □



4. Tegyük fel, hogy a G összefüggő gráfban nem található három egymástól pontdiszjunkt részgráf, melyek mindegyike pontosan egy éllel kapcsolódik a maradék gráfhoz. Mutassuk meg, hogy G legfeljebb egy további éllel hozzávételével 2-élösszefüggővé tehető.

   A feladatbeli feltétel lényegében azt fogalmazza meg, hogy G 2-komponensei között nincs 2-nél több levélkomponens. Mivel G összefüggő, ezért izolált kompnense sincs. Ezek szerint a 2-élöf tulajdonság eléréséhez szükséges új élek száma a levélkomponensek számának a fele, vagyis 1. (Egyébként az is könnyen látszik, hogy G 2-komponensei egy út mentén helyezkednek el, és az út két végén található levélkomponensek közé behúzott él megfelelő a 2-élof tulajdonság eléréséhez. □

5. Tegyük fel, hogy G olyan összefüggő gráf, melynek 11 maxblokkja és 6 elvágó pontja van. Igazoljuk, hogy 5 él behúzásával elérhető, hogy G 2-szeresen (pont)összefüggő legyen.

   A tanultak szerint szükséges élek min. száma $\max(b(G)-1, \left\lceil \frac{m(G)+2m'(G)}{2} \right\rceil)$. Mivel G öf, nincs izolált blokk ($m'(G)=0$) mivel 1-nél több elvágó pont van, ezért nem lehet minden maxblokk levélblokk, tehát $m(G) \leq 10$. Akkor kéne legalább 6 él, ha $b(G) \geq 7$ lenne. De ekkor a 7-es elvágó ponttól különböző minden elvágó pontot tartalmazza egy olyan maxblokk, ami az adott pont elhagyásakor a 7-es elvágó pontot nem tartalmazó komponensbe esik, és ezek a maxblokkok pként különbözők. De ekkor legalább $7+5=12$ maxblokkja lenne G-nek. □

6. Adjunk meg olyan eljárást, ami tetszőleges irányítatlan G input gráf esetén meghatározza a legkisebb olyan k pozitív egész számot, amire hozzáadható G-hez k él úgy, hogy a kapott gráf 2-szeresen élösszefüggő legyen, és a hozzáadott élek utat alkossanak.

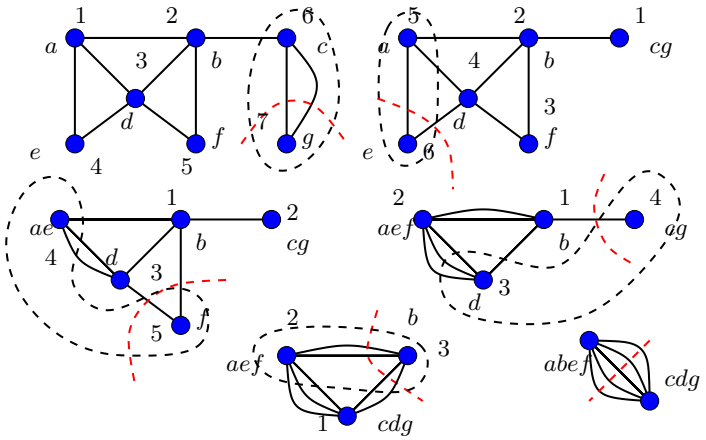
   Ha G minden 2-komponense izolált, akkor ha G összefüggő, akkor 0 a válasz. Ha G nem összefüggő, és minden 2-komponense 1-pontú, akkor G-nek nincs éle, ezért bárhogyan is adunk hozzá egy utat, nem lesz 2-élösszefüggő.

   Minden egyéb esetben vagy van G-nek olyan izolált 2-komponense, ami legalább 2 csúcsot tartalmaz, vagy van G-nek két olyan levél 2-komponense, amik ugyanahhoz a komponenshez tartoznak.

   Úgy adunk hozzá egy utat G-hez, hogy az minden izolált és levél 2-komponenst érintsen, továbbá, ha van egy legalább kétcsúcsú izolált 2-komponens, akkor annak egyik csúcsából induljon és egy másikba érkezzen, ill. ha ilyen nincs, akkor egy levélkomponensből induljon, és ugyanezen komponens egy másik levélkomponensébe érkezzen az út. Egy így konstruált út hozzáadása nyomán egyetlen elvágó él sem marad a kapott gráfban, továbbá az ilyen út $\ell(G)+\ell'(G)-1$ élt tartalmaz. Mivel minden izolált és levél 2-komponenst érintenie kell a hozzáadott éleknek, ezért ennél kevesebb élű úttal nem lehet megoldani a feladatot, ez tehát a keresett minimum.

7. Keressünk minimális vágást a Nagamochi-Ibaraki algoritmus segítségével egy legalább 6-pontú (nem feltétlenül egyszerű) gráfban.

   Hát tessék, itt van a példa az előadásról. Esetleg érdemes vmi 6-7 pontú random gráfon is megpróbálni.



8. Bizonyítsuk be, hogy tetszőleges véges $G=(V,E)$ gráfban találhatók olyan (u_1,w_1) és (u_2,w_2) pontpárok, amikre $w_1 \neq w_2$, valamint $\lambda(u_1,w_1)=d(w_1)$ és $\lambda(u_2,w_2)=d(w_2)$ teljesül.

   Az órán azt tanították, hogy a maxvissza sorrend utolsó két pontja rendelkezik ezzel a tulajdonsággal, azaz ha $v_1,v_2,\ldots,v_n$ egy maxvissza sorrend, akkor $u_1=v_{n-1}, w_1=v_n$ jó választás. Csupán egy másik hasonló tulajdonságú (u_2,w_2) párt kell találni úgy, hogy w_2 különbözzék w_1-től. Ehhez elegendő egy másik maxvissza sorrendet találni, aminél garantálni tudjuk, hogy az utolsó csúcs nem w_1 lesz.

Azért annyira ez se bonyolult: készítsünk úgy egy másik maxvissza sorrendet, hogy w_1 az első csúcs ebben a másik sorrendben. Az utolsó csúcs így garantáltan nem w_1 lesz, tehát az így kapott maxvissza sorrend utolsó két csúcsa megfelel u_2-nek és w_2-nek. □

Megj: a Gomory-Hu fa bármely levele és annak szomszédja ilyen párt alkotnak.

9. Tegyük fel, hogy amikor a Nagamochi-Ibaraki algoritmus segítségével határozzuk meg a G (nem feltétlenül egyszerű) gráf élösszefüggőségét, akkor a max-vissza sorrendben utolsó csúcsok fokszámai rendre az alábbiak: $7, 9, 4, 6, 7, 7, 6, 8, 6, 7, 9$. Határozzuk meg G élösszefüggőségi számát, $\lambda(G)$-t. Állapítsuk meg, legkevesebb hány élt kell behúzni G-be ahhoz, hogy a kapott gráf 6-szorosan élösszefüggő legyen.

   Az órán tanultak szerint $\lambda(G)$ megegyezik a Nagamochi-Ibaraki algoritmus során a maxvissza sorrendbeli utolsó csúcsok fokszámai közül a legkisebbikkel. Ez a konkrét esetben $\lambda(G) = 4$-nek adódik.

   Mivel a G gráf 4 él elhagyásával szétvágható, ezért a 6-szoros élösszefüggőség eléréséhez legalább 2 él behúzása szükséges.

   Azt állítjuk, hogy 2 él elegendő is. Vegyük észre, hogy a harmadik összeolvasztás után kapott G_3 gráf élösszefüggősége 6, hiszen G_3-ra a Nagamochi-Ibaraki algoritmus abban különbözik az eredeti G-re végrehajtottól, hogy az utóbbi esetben még három további összeolvasztás miatt három további minvágásjelölt adódik (konkrétan $7, 9$ és 4.

   Az is igaz, hogy az elsőnek összeolvasztott két csúcs között volt 7 pként élidegen út, ezért ezt a két csúcsot a G semmilyen 6-nál kevesebb élt tartalmazó vágása nem szeparálja. Ugyanez elmondható a másodiknak összeolvasztot két csúcsra: ennek a műveletnek a során sem veszett el G-nek 6-nál kevesebb élt tartalmazó vágása. Ezért ha a harmadiknak összeolvasztott két csúcs közé két további élt húzunk, akkor ezáltal a harmadik maxvissza sorrend továbbra is maxvissza sorrend marad, ám a minvágás jelölt immár 6 élt fog tartalmazni. Mivel innentől minden vágásjelölt legalább 6 élt tartalmaz, ezért a két új éllel kiegészített gráf 6-szorosan élösszefüggő lesz. A feladat második kérdésére tehát pontosan 2 a valasz. □

   Megjegyzés: a fenti gondolatmenetben a „harmadiknak összeolvasztott két csúcs" közé kellett két új élt behúzni, miközben e két csúcs (a korábbi összeolvasztások miatt) nem biztos, hogy az *eredeti* G gráfnak is csúcsa volt. Sebaj, ha az eredeti G-ben úgy húzzuk be a két említett élt, hogy az a harmadiknak összeolvasztott két csúcs között vezessen, akkor az megfelelő lesz, hiszen az első három alkalommal olyan csúcspárokat olvasztottunk össze, amelyeket nem szeparált 6-nál kisebb vágás.

10. Tegyük fel, hogy amikor a Nagamochi-Ibaraki algoritmus segítségével határozzuk meg a G multigráf élösszefüggőségét, akkor a max-vissza sorrendben utolsó csúcsok fokszámai rendre az alábbiak: $7, 9, 6, 4, 7, 5, 4, 8, 4, 7, 9$. Határozzuk meg G élösszefüggőségi számát, $\lambda(G)$-t. Igaz-e, hogy G-nek bizonyosan van olyan legfeljebb 4 elemű X ponthalmaza, hogy X és a komplementere között futó élek száma megegyezik $\lambda(G)$-vel?

    Az órán tanultak szerint $\lambda(G)$ megegyezik a Nagamochi-Ibaraki algoritmus során a maxvissza sorrendbeli utolsó csúcsok fokszámai közül a legkisebbikkel. (2 pont)
    Ez a konkrét esetben $\lambda(G) = 4$-nek adódik. (1 pont)
    Olyat is tanítottak, hogy $\lambda(v_{n-1}, v_n) = d(v_n)$, azaz a maxvissza sorrend utolsó két pontját elválasztó vágások között minimális számú élt tartalmaz az a vágás, amelynek egyik oldalán a maxvissza sorrendbeli utolsó csúcs áll. (2 pont)
    Mivel minden egyes maxvisszasorrend-számítás után a sorrend két utolsó pontját összeragasztjuk, ezért amikorra a negyedik maxvissza sorrendben az utolsó csúcs foka 4 lett, addig összesen 3 összeragasztást hajtottunk végre. Ezért a negyedik maxvissza sorrend utolsó csúcsa által reprezentált X ponthalmaz G-nek legfeljebb 4 pontját tartalmazhatja. (3 pont)
    Mivel ez az X a G egy minimális vágását határozza meg, ezért a második kérdésre igenlő a válasz: van olyan legfeljebb 4 elemű ponthalmaza G-nek, ami G minimális vágását indukálja. (2 pont)

Kombinatorikus optimalizálás 2023. tavasz

8. gyakorlat 2023. május 4.

Gyakorlatok

1. Igazoljuk, hogy ha G-nek van fülfelbontása, akkor G bármely két fülfelbontása ugyanannyi fület tartalmaz.

   Bármilyen fület is ragasztunk egy gráfra, ennek nyomán az újonnan bevett csúcsai számánál eggyel több új él keletkezik. Mivel a kiindulási gráfnak 1 csúcsa és 0 éle volt, és minden fül 1-gyel növelte az élszám és csúcsszám különbségét, ezért ha G-nek van fülfelbontása, abban pontosan $|E(G)| - |V(G)| + 1$ fülnek kell szerepelnie, bármelyik fülfelbontást is tekintjük.

2. Tegyük fel, hogy a G gráf 2-élöf és nincs 2-fokú csúcsa. Bizonyítsuk be, hogy van G-nek olyan e éle, amire a $G - e$ gráf is 2-élöf. Igaz-e hasonló állítás a 2-élöf helyett 2-öf tulajdonsággal?

   Mindkét állítás igaz. Tekintsük G egy fülfelbontását, és nézzük az utolsó fület. Ez vagy a keresett e él, vagy tartalmaz 2-fokú pontot.

3. Tegyük fel, hogy a G gráfnak van olyan fülfelbontása, amelyikben egy fül páros sok, az összes többi pedig páratlan sok élt tartalmaz. Igazoljuk, hogy G-nek van teljes párosítása.

   Tfh a páros fül az u és v csúcsokat köti össze. A páros fül felragasztása előtt a gráf a 4 feladat szerint kritikus volt, ezért létezik olyan párosítása, ami u-t fedetlenül hagyja, minden más csúcsot lefed. Ekkor a páros fül éleiből található olyan párosítás, ami a fül csúcsait és u-t fedi, v-t nem. A két párosítás együtt a páros fül felragasztása után keletkező gráf teljes párosítása lesz. (Ld. a 3. ábrát.)

   Minden ezt követő páratlan fül páros sok új csúcsot ad a gráfhoz. Ha a meglévő párosításunkhoz mindig hozzáadjuk a fül belső csúcsait kipárosító fül-éleket, akkor mindig az épp felépített gráf egy teljes párosítását kapjuk. (Ld. a 4. ábrát.)

   Ezért a feladatban leírt G-nek bizonyosan van teljes párosítása. □

4. Tegyük fel, hogy a G gráfnak olyan fülfelbontása van, amelyben minden fül páratlan sok élt tartalmaz. Mutassuk meg, hogy G minden v csúcsához található G egy olyan M párosítása (azaz közös végpont nélküli éleinek halmaza), hogy a v csúcs kivételével G minden csúcsára illeszkedik M-beli él.

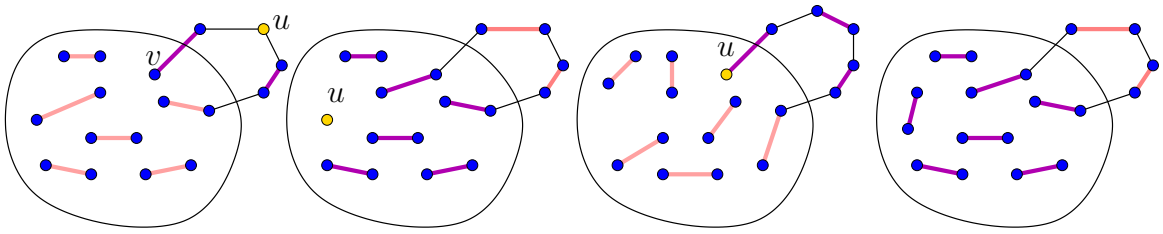
   A feladatban leírt tulajdonsággal rendelkező gráfot a továbbiakban kritikus gráfnak nevezzük.

   Tekintsük G páratlan fülfelbontását. Az első fül egy ptn kör, ez bizonyosan kritikus: bármely csúcsát is hagyjuk el egy ptn körnek, egy páros sok csúcsot tartalmazó utat kapunk, aminek van teljes párosítása. Azt ellenőrizzük, hogy bárhol is állunk meg a fülek felragasztásával, a kapott gráf szintén kritikus lesz. Ehhez azt kell ellenőrizni, hogy ha egy G kritikus gráfra páratlan fület ragasztunk, akkor a kapott G' gráf szintén kritikus lesz.

   Azt kell tehát bizonyítani, hogy G' bármely u csúcsára $(G' - u)$-nak van teljes párosítása. Ha u a fül egy csúcsa,akkor a fül éleiből találunk olyan párosítást, ami a fül minden más csúcsát fedi. Ráadásul a fül páratlan volta miatt a fül egyik végpontját (amit hívjunk v-nek) fedni fogják az így választott élek, a másikat nem. Ezt a párosítást ki lehet egészíteni a $G - v$ gráf egy teljes párosításával, és így megkapjuk $G' - u$ egy teljes párosítását. (Ld. az 1. ábrát. Az első körben választott párosítás éleit sötét, az ezt kiegészítő párosítás éleit világos színnel jelöltük.)

   Ha azonban u nem a fülön, hanem a G gráfon van, akkor $G - u$-nak van teljes párosítása, hisz G kritikus, és a G'-t kialakító fülön páros sok további csúcs van, amiket a fülön lehet kipárosítani. (Ld. a 2. ábrát.)

   Minden u-ra van tehát $G' - u$-nak teljes párosítása, ezért a páratlan fül felragasztása megőrzi a gráf kritikus voltát, ezzel pedig az állítást igazoltuk. □



5. Tegyük fel, hogy egy G gráfban $\nu(G) < \frac{|V(G)|-1}{2}$ teljesül a maximális párosítás $\nu(G)$ méretére. Bizonyítsuk be, hogy ha G-nek van fülfelbontása, akkor G bármely fülfelbontásában a páros sok élt tartalmazó fülek száma legalább $|V(G)| - 2 \cdot \nu(G) + 1$.

   Ha G egy fülfelbontása nem tartalmaz páros fület, akkor a 4. gyakorlat miatt G maximális párosítása egy csúcs híján G minden csúcsát fedi, így $\nu(G) = \frac{|V(G)|-1}{2}$. Ezt kizárja a feltétel, tehát G bármely fülfelbontása tartalmaz legalább egy páros fület.

Jelölje $p(G)$ a G gráf fülfelbontásához szükséges páros sok élt tartalmazó fülek minimimális számát. Azt kell igazolni, hogy $p(G) \geq |V(G)| - 2\nu(G) + 1$. Ezzel ekvivalens a $\nu(G) \geq \frac{|V(G)|-p(G)+1}{2}$ összefüggés. Ez utóbbi pedig azonnal látszik, ha sikerül igazolni, hogy G-nek van olyan párosítása, ami legfeljebb $p(G) - 1$ csúcsot hagy fedetlenül, hiszen egy ilyen párosítás élszáma $\frac{|V(G)|-p(G)+1}{2}$, így $\nu(G)$ ennél kisebb nem lehet.

Rögzítsük a G gráf $p(G)$ páros fület tartalmazó fülfelbontását. Ennek segítségével megadjuk G egy olyan párosítását, ami összesen $p(G) - 1$ csúcsot hagy fedetlenül. Tekintsük a rögzített fülfelbontásban az első páros fület és álljunk meg ezen páros fül hozzáadása után. Az így kapott félkész G' gráfnak a 3. gyakorlat miatt van teljes párosítása, azaz egy olyan párosítása, ami G' minden csúcsát fedi. Ezt a párosítást bővítjük G egy $p(G) - 1$ csúcsot fedetlenül hagyó párosításává. Ehhez csupán azt kell tenni, hogy minden páratlan fül hozzáadásakor (amikoris páros sok csúcsot adunk G-hez), az addigi párosítást olyan élekkel bővítjük, amik a fülön hozzáadott új csúcsokat teljesen kipárosítják. Egy páros fül hozzáadásakor páratlan sok új csúccsal növekszik a félkész gráf; ilyenkor úgy bővítjük a meglevő párosítást, hogy a fülön lévő új csúcsokat egy híján párosítjuk ki. Ezáltal minden páros fül hozzáadásával 1-gyel nő a fedetlen csúcsok száma, tehát az összes fül hozzáadása után G olyan párosítását kapjuk, amiben a fedetlen csúcsok száma pontosan $p(G) - 1$.

6. Igazoljuk, hogy minden erősen összefüggő G gráf előállítható egy irányított körből kiindulva irányított fülek egymás utáni hozzávételével.

   Az órai bizonyítás irányított esetben is elmondható.

   Tetsz. uv él esetén uv egy vu-úttal irányított kört alkot, kiindulásnak épp jó. Ha a félkész gráfról lelóg egy él, akkor a megkonstruálatlan csúcsból/csúcsba van irányított út a már megkonstruált csúcsokba/csúcsokból. Ez az út a lelógó éllel együtt egy irányított fület alkot. Ha már G minden csúcsát megkonstruáltuk, akkor a hiányzó élek fülekként hozzávehetők.

7. Tegyük fel, hogy a G gráf erősen összefüggő. Bizonyítsuk be, hogy ha G nem egy kör, akkor elhagyható G egy éle úgy, hogy a kapott gráf erősen összefüggő maradjon vagy található G-ben olyan irányított út, aminek a csúcsait G-ből elhagyva erősen összefüggő gráfot kapunk.

   Az irányított fülfelbontás utolsó füle megfelel.

8. Tegyük fel, hogy a G gráfnak van olyan fülfelbontása, amelyikben az első fül egy C_3, minden tovább fül pedig 2 élt tartalmaz. Bizonyítsuk be, hogy G minden éle kiszínezhető a piros, vagy zöld színekkel úgy, hogy egyetlen él kapja meg mindkét színt, továbbá a piros és a zöld élek is G egy-egy feszítőfáját alkossák.

   A C_3 egy élét kétszínűre, egy élét pirosra és egy élét zöldre színezzük. Minden további fül egyik élét pirosra, a másikat zöldre festjük. Így minden félkész gráfnak lesz egy fehér éle és egy piros ill. zöld feszítőfája.

Kombinatorikus optimalizálás 2023. tavasz

9. gyakorlat 2023. május 11.

Gyakorlatok

1. Igazoljuk, hogy a maximális párosítás $\nu(G)$ méretének meghatározására $\frac{1}{2}$-közelítést kapunk, ha a G gráfban mohó módon választunk diszjunkt éleket egészen addig, amíg már nem lehet a korábban választottaktól diszjunkt, újabb élt találni.

   Mutassuk meg azt is, hogy ha ezt az algoritmust úgy fejlesztjük tovább, hogy megpróbálunk minden mohón talált uv élt helyettesíteni egy xu és egy yv éllel a párosítás által fedetlen, alkalmas x és y csúcsokra, akkor amennyiben már egyik megadott módon sem növelhető a párosítás, akkor az algoritmus egy $\frac{2}{3}$ közelítést ad.

   A nem bővíthető párosítás végpontjai egy lefogó ponthalmazt adnak, ezért, ha M nem bővíthető, akkor $2|M| \geq \tau(G) \geq \nu(G)$.

   Ha nincs M-hez 3 élű javító út, akkor minden javító út legfeljebb 3/2-szerannyi új élt tartalmaz, mint régit.
2. Futtassuk le a halmazfedési problémára tanult mohó algoritmust konkrét példán.
3. Keressünk hatékony közelítő algoritmust a halmazfedési probléma alábbi általánosítására. Adott egy n elemű U alaphalmazon az $S_1, S_2, \ldots, S_k$ részhalmazok rendszere és adottak a $c(S_i) \geq 0$ költségek. A cél, hogy úgy válasszunk ki néhány részhalmazt a fentiek közül, hogy U minden elemét legalább két kiválasztott részhalmaz tartalmazza és a kiválasztott részhalmazok összköltsége a lehető legkevesebb legyen. Milyen multiplikatív hibával tudjuk közelíteni az optimális megoldást?

   Mohón választunk halmazokat, mindig azt, amelyik a legkisebb fajlagos költséggel fedi az eddig (elégszer) le nem fedett pontokat. Mindig csak azon részhalmazok közül választunk, amelyek még nem szerepelnek a fedésben. Ha $a_1, a_2, \ldots, a_{2n}$ sorrendben fedjük a csúcsokat (minden csúcs kétszer szerepel a sorrendben), akkor a_i fedésének a fajlagos költsége legfeljebb $\frac{1}{2n-i+1} \cdot OPT$ lesz, ahol OPT az optimális duplánfedés összköltsége. (Ha ugyanis az optimális fedésből elhagyjuk a már kiválasztott halmazokat, akkor az optimális fedés maradék halmazai megfelelő multiplicitással fedik a még lefedendő pontokat.) Így aztán az összköltség felső becslésére $OPT \cdot (\frac{1}{2n} + \frac{1}{2n-1} + \ldots + \frac{1}{1})$ adódik.
4. Keressünk hatékony közelítő algoritmust a halmazfedési probléma alábbi általánosítására. Adott egy n elemű U alaphalmazon az $S_1, S_2, \ldots, S_m$ részhalmazok rendszere és adottak a $c(S_i) \geq 0$ költségek. A cél, hogy úgy válasszunk ki néhány részhalmazt a fentiek közül, hogy a a kiválasztott részhalmazok U-nak legalább k elemét tartalmazzák és a kiválasztott részhalmazok összköltsége a lehető legkevesebb legyen. Mennyire tudjuk leszorítani a közelítés multiplikatív hibáját?

   Legyen OPT az U halmaz k eleme lefedésének minimális költsége. A feladat most abban különbözik a halmazfedéstől, hogy elég tetszőleges k elemet lefedni, nem kell az összeset. Természetesen ismét mohó algoritmussal dolgozunk, mint a legolcsóbb fedés esetén, csak annak érdekében, hogy használható becslést kapjunk, kicsit másképp érdemes számolni a fajlagos költséget. Nevezetesen, tfh már néhány elemet lefedtünk a mohón választott halmazainkkal és még ℓ elemet kell fedni a k lefedéséhez. Ekkor egy S^i halmaz által meghatározott fajlagos költség $c(S^i)/m$ lesz, ahol m most nem az S_i-beli mindeddig lefedetlen pontok száma, hanem ennek amennyiségnek kell a minimumát képezni ℓ-lel. Az S_i szerinti fajlagos költség tehát $f(S^i) = c(S^i)/\min\left(\ell, \left|U \setminus (S^1 \cup S^2 \cup \ldots \cup S^{i-1})\right|\right)$. A mohó algoritmus által i-diknek választott halmaz tehát az az S^i lesz, ami az előző mennyiséget minimalizálja.

   Amikor az alaphalmazból a j-dik elemet fedjük le a mohó algoritmus szerint, akkor legfeljebb $j-1$ elem volt már lefedve, tehát az OPT költségű optimális lefedésben vannak olyan halmazok, amik szóba jönnek a j-dik elem lefedésére, sőt, ezek a halmazok legalább $k-(j-1)$ elemet fednek le az eddig fedetlenek közül, összköltségük pedig legfeljebb OPT. Ezért már ezen halmazok között is van olyan, ami legfeljebb $OPT/(k-j+1)$ fajlagos költséggel fed valamely elemet, így a mohó választásból kifolyólag az j-diknek lefedett elem fajlagos költsége sem haladja meg ezt az értéket. A mohó algoritmus által megtalált megoldás összköltségét úgy is megkaphatjuk, mint az egyes lefedett elemekhez tartozó fajlagos költségek összegét, amit ezek szerint így tudunk becsülni: $\sum_i c(S^i) \leq OPT \cdot (1/k + 1/(k-1) + \ldots + 1/1) \leq OPT \cdot (1 + lnk)$.
5. Az inputként megadott 2-élösszefüggő G gráfnak egy lehető legkevesebb élű 2-élösszefüggő feszítő részgráfját keressük. Igazoljuk, hogy az alábbi algoritmus ennek a feladatnak egy 2-közelítését adja. Válasszuk ki G egy F feszítőfáját, majd a $G-F$ gráfnak egy F' feszítő erdejét (azaz $G-F$ minden komponensének egy feszítőfáját). Az output az $F \cup F'$ élhalmaz.

Két dolgot kell igazolni: egyrészt, hogy $F \cup F'$ 2-élöf feszítő részgráfot határoz meg, másrészt pedig, hogy legfeljebb 2-szer annyi éle van, mint az optimumnak Ha $F \cup F'$-ben lenne egy e elvágó él, akkor $e \in F$, hiszen ha F minden élét megőriznénk, akkor összefüggő maradna a gráf. Mivel $F - e$ két komponense között nem fut éle se F-nek, se F'-nek, ezért G egyetlen éle sem kötheti össze e két részt. Tehát e a G-nek is elvágó éle, volt, ami ellentmondás, így $F \cup F'$ valóban 2-élösszefüggő gráfot alkot.

Ha H egy minimális élszámú 2-élöf feszítő részgráf, akkor minden fokszáma legalább 2 (különben ugyanis lenne elvágó éle), így H élszáma legalább a fokszámösszeg fele, ami legalább n. Az F élszáma $n-1$, az F'-é is legfeljebb ennyi, szóval az output legfeljebb $2n-2$ élt tartalmaz, ami kevesebb, mint az optimum 2-szerese.

6. Gyakoroljuk az SPT, LS ill LPT ütemezés keresését ill. az FF és FFD ládapakolási eljárásokat konkrét példákon.

7. Bizonyítsuk be, hogy minden $Pm||C_{\max}$ típusú ütemezési feladat esetén van a munkáknak olyan sorrendje, amire listás ütemezés (azaz az LS algoritmus) az adott inputhoz tartozó minimális átfutási idővel ütemez.

   Tekintsünk egy olyan S ütemezést, ami a megadott inputra garantálja a minimális átfutási időt. Módosítsjuk az S ütemezést úgy, hogy egyetlen gépen se legyen üresjárat: minden M géphez az M-re ütemezett minden egyes J_i munkát kezdjük el azonnal, amint a J_i munkát megelőző munka M-en befejeződött. Világos, hogy egy optimális ütemezés ettől optimális marad, hiszen így egyetlen gép sem dolgozik tovább annál, mint ameddig a S ütemezés szerint dolgozna. Tfh ebben a módosított S' ütemezésben a munkák kezdési időpontja a $J_1, J_2, \ldots$ sorrendet határozza meg. (Az egyszerre kezdőző munkák sorrendje tetszőlegesen választható.)

   Ha az LS algoritmust a munkáknak ebben a $J_1, J_2, \ldots$ sorrendjében futtatjuk, akkor az így kapott ütemezés ugyan eltérhet S'-től, de csak annyiban, hogy egy munka más gépre kerül (de ugyanakkor kezdődik), mint S' szerint. Ezáltal a munkák kezdési (így befejezési) időpontja sem változik. Ezért az LS szerinti ütemezés átfutási ideje megegyezik S'-évl, tehát minimális.

8. Tegyük fel, hogy a $J_1, J_2, \ldots$ munkákat az S ütemezés úgy ütemezni 42 gépre, hogy az átfutási idő 42, az átlagos átfutási idő pedig 24 legyen. Mutassuk meg, hogy ugyanezek a munkák ütemezhetők 21 gépre úgy, hogy az átfutási idő legfeljebb 84, az átlagos átfutási idő pedig legfeljebb 45 legyen. (Egyébként olyan ütemezés is van, amire az átfutási idő legfeljebb 84, az átlagos átfutási idő pedig legfeljebb 39.)

   Könnyű olyan ütemezést készíteni 21 gépre, amivel a munkák átfutási ideje legfeljebb 84 lesz: képezzünk az S ütemezésben szereplő 42 gépből 21 párt és feleljen meg minden ilyen $(M(a), M(b))$ géppárnak egy $M(ab)$ gép az S' ütemezésben. Ütemezzük erre az $M(ab)$ gépre mindazon munkákat, amelyeket S az $M(a)$ vagy $M(b)$ gépre ütemezett. Mivel az $M(a)$-ra ütemezett munkák összmegmunkálási ideje legfeljebb 42, és ugyanez igaz az $M(b)$ gépre ütemezett munkákra is, az S' ütemezés mind a 21 géphez úgy fog munkákat hozzárendelni, hogy az összmegmunkálási idő egyik gépen se legyen több $42 + 42 = 84$-nél.

   Úgy akarjuk az S'-t megválasztani, hogy az átlagos átfutási idő se legyen több 45-nél. Ezért a 21 gép mindegyikén az oda ütemezett munkákat SPT sorrendben, azaz a megmunkálási idő növekvő sorrendjében végezzük el. Az órán tanult tétel szerint ez a sorrend minden egyes gépre minimalizálja az ottani átlagos átfutási időt. Ahhoz, hogy megmutassuk, hogy az átlagos átfutási idő ezen ütemezés mellett legfeljebb 45, mutatunk egy másik, ennél rosszabb átlagos átfutási időt adó S'' ütemezést, és erről mutatjuk meg, hogy ez legfeljebb 45 átlagos átfutási időt eredményez.

   Ezt az S'' ütemezést minden $M(ab)$ gépre az alábbiak szerint határozzuk meg. Tegyük fel, hogy S az $M(a)$ gépre legalább annyi munkát ütemezett, mint $M(b)$-re. Ekkor S'' az $M(ab)$ gépen először az eredetileg $M(a)$-ra ütemezett munkákat ütemezi (az S szerinti sorrendben), majd ezeket követik az $M(b)$-re ütemezett munkák, szintén az S szerinti sorrendben. Ezáltal minden, eredetileg $M(a)$-ra ütemezett munka ugyanakkor fog kezdődni (és befejeződni is) mint S szerint, és minden, eredetileg $M(b)$-re ütemezett munka legfeljebb 42-vel később fog kezdődni (ill. végződni), mint S szerint. A $\sum_i C_i^S$ összegben szereplő tagoknak így legfeljebb a fele változik, és minden megváltozott tag legfeljebb 42-vel növekszik meg. Ha tehát e helyett az összeg minden tagjához 21-et adunk, akkor attól az összeg nem csökken, az átlag azonban pontosan 21-gyel növekszik. Ezért az átlagos átfutási idő az S'' ütemezés szerint legfeljebb $24+21 = 45$, és nekünk pontosan egy ilyen tulajdonságú ütemezés létezését kellett igazolnunk. Hab a tortán, hogy a gépeket SPT szerint ütemező S' még ennél is jobb lehet.

9. Tegyük fel, hogy 7 munka összvégrehajtási ideje 42. Igazoljuk, hogy lehetséges ezeket a munkákat egy gépre ütemezni úgy, hogy az átlagos átfutási idő legfelejebb 24 legyen.

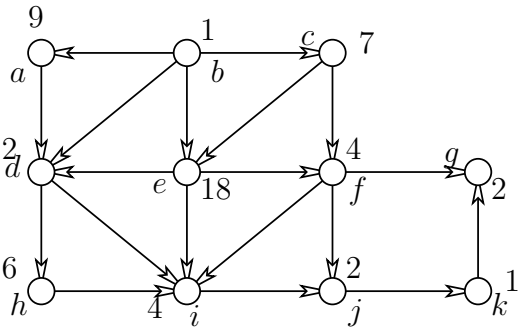
   Az órán tanultak szerint az SPT sorrendben történő ütemezés adja a lekisebb átlagos átfutási időt, ezért azt kell megmutatni, hogy ebben a sorrendben ez a mennyiség legfeljebb 24. A legrövidebb munka hossza legfeljebb $42/7 = 6$. A két legrövidebb munka összhossza legfeljebb $2 \cdot 42/7 = 12$, sít, az i legrövidebb munka összhossza legfeljebb $i \cdot 42/7$. Ezek szerint az első munka legkésőbb $t = 6$-ban, a második legkésőbb $t = 12$-ben, az i-dik legkésőbb $t = 6i$-ben fejeződik be. Az átlagos átfutási időt tehát felülről becsüljük akkor, ha ezekkel az értékekkel számolunk, azaz, ha feltesszük, hogy mind a 7 megmunkálási idő pontosan 6. Mivel a $6, 12, \ldots, 42$ számok átlaga $(6 + 42)/2 = 24$, ezért az átlagos átfutási idő is legfeljebb 24 lesz az SPT sorrendben történő ütemezés esetén.

10. Egy gépen 10 munkát 100 időegység alatt végeztünk el úgy, hogy az átlagos átfutási idő épp 68 volt. Mennyi idő alatt végeztük volna el a munkákat fordított sorrendben, és mennyi lett volna így az átlagos átfutási idő?

    Természetesen a munkák fordított sorrendben történő elvégzéséhez is 100 időegységre van szükség. (Ha ez nem világos, gondolkodjunk el azon, hogy ha a farkasok egymás vállára állva szeretnék elérni az ágvégen kuksoló kismalacot, akkor hogyan érdemes ezt megtenniük: alulra álljanak a magasak és felülre az alacsonyak, vagy fordítva.)

    Az átlagos átfutási idő meghatározásához reprezentáljuk az eredeti sorrendben végrehajtott munkákat a számegyenesen, mégpedig a végrahajtásuknak megfelelő időintervallumokkal. Ekkor az utolsónak befejezett munka kivételével minden egyes munka befejezéséhez tartozik fordított sorrendben egy másik munka befejezése, úgy, hogy a két megfelelő atfutási idő összege pontosan 100. Ezért e két sorrendhez tartozó átfutási idők összege $9 \cdot 100 + 100 + 100$ lesz, ahol a második tag az utolsó munka átfutási ideje (ennek nincs párja), a harmadik tag pedig a fordított sorrendben utolsó munka átfutási ideje (ami szintén nem párja semelyik másik munkának. Az átlagos átfutási idő a két sorrenddel számolva $1100/20 = 55$. Ezek szerint az eredeti sorrendbeli átlagos átfutási idő és a fordított sorrendbeli átlagos átfutási idő átlaga éppen 55. Itt nem részletezett számítások azt mutatják, hogy ekkor a fordított sorrendhez tartozó átlagos átfutási idő 42.

11. Az ábrán látható gráf csúcsai munkákat jelentenek, és minden csúcsra az adott munka megmunkálási ideje van ráírva. A gráf egy uv élének jelentése az, hogy a v munkát csak az u munka befejezése után lehet elkezdeni. Keressünk minimális átfutási idejű ütemezést arra az esetre, ha ugyanazon a gépen kell minden munkát elvégezni. Mennyi a minimális átfutási idő, ha tetszőlegesen sok gépet használhatunk?

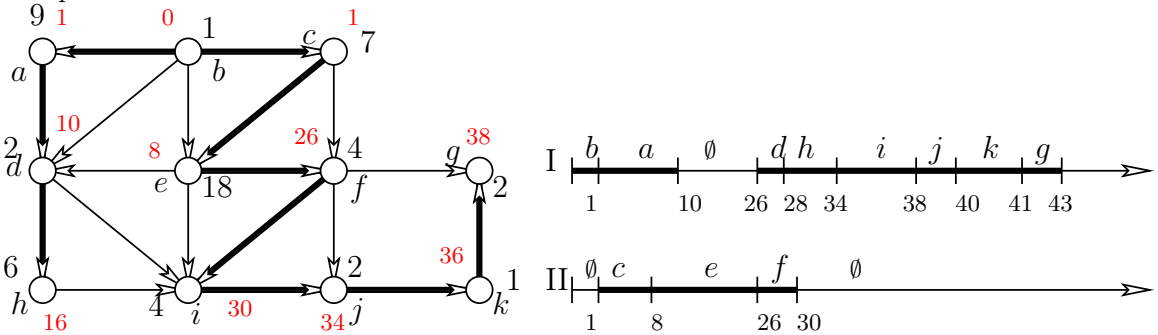


    Határozzuk meg az átfutási időt 2 gépre történő listás ütemezés esetén, ha mindig azt a munkát ütemezzük következőnek az elvégezhetők közül, amelyik neve az ABC-ben a legelöl áll.

    A munkákat olyan sorrendben kell elvégezni, amelyben a gráf minden éle balról jobbra mutat. Ez a topologikus sorrend definíciója, ilyet kell keresni pl. forrástörléssel. Ez a sorrend adja meg az ütemezést, és az átfutási idő természetesen az összmegmunkálási idő, konkrétan 56 lesz. (Topologikus sorrend pl a $b, a, c, e, d, h, f, i, j, k, g$.)

    Ha tetszőlegesen sok gépünk van, akkor minden munkát tudunk külön gépre ütemezni, így az átfutási idő a precedenciafeltételekből adódik. Az utolsónak befejezett J_1 munkát akkor tudtuk elkezdeni, amikor az utolsó olyan J_2 munkát fejeztük be, ami szükséges volt J_1 elkezdéséhez. A J_2 kezdete persze az utolsónak befejezett olyan J_3 munka befejezési időpontja, ami J_2 kezdéséhez szükséges. És így tovább. Azt kapjuk, hogy a minimális átfutási idő megkapható a $J_1, J_2, \ldots$ munkák összmegmunkálási idejeként, ahol ezek a munkákat a precedenciafeltételek miatt csak egyféle sorrendben, egymás után tudjuk elvégezni. Tehát a minimális átfutási idő megegyezik az ilyen munkaláncok összmegmunkálási idejének maximumával. Ezt úgy is megfogalmazhatjuk, hogy a gráfba bevezetünk egy új t csúcsot, G minden csúcsából vezetünk t-be egy irányított élt, majd az így kapott gráf minden uv irányított élére élhosszként ráírjuk az u munka megmunkálási idejét. Nekünk ebben a gráfban kell maximális hosszúságú irányított utat keresnünk.

    Hát ez meg épp a PERT feladat, amit már pazarul megtanultunk, a topologikus sorrend megkeresését pedig a feladat első része miatt már el is végeztük. (A PERT feladat megoldása minden csúcshoz hozzárendel egy kezdési időpontot, és pontosan ez lesz az adott munkához tartozó megmunkálási idő kezdete a minimális átfutási időt biztosító ütemezés szerinti.) A bal oldali ábra a PERT szerinti kezdési időponto az egyes csúcsok mellett piros színnel szerepelnek. A legkorábbi kezdési időpontokat meghatározó élek szintén megvastagítva szerepelnek.



    A kétgépes ütemezés során az alábbi események követik egymást (ld. a jobb oldali ábrát):

    - $t = 0$-ban b-t elkezdjük az I. gépen.
    - $t = 1$-ben a-t elkezdjük az I. gépen, c-t a II-on.

- $t = 8$-ban e-t elkezdjük a II. gépen.
- $t = 26$-ban d-t elkezdjük az I. gépen, f-et a II-on.
- $t = 28$-ban h-t elkezdjük az I. gépen.
- $t = 30$-ban f befejeződik a II. gépen
- $t = 34$-ban i-t elkezdjük az I. gépen.
- $t = 38$-ban j-t elkezdjük az I. gépen.
- $t = 40$-ban k-t elkezdjük az I. gépen.
- $t = 1$-ban g-t elkezdjük az I. gépen.
- $t = 43$-ban g befejeződik az I. gépen és ezzel minden munkát végrehajtottunk.

12. Tegyük fel, hogy a ládapakolási feladat egy konkrét inputjához az FF algoritmus 42 ládát használ fel. Bizonyítsuk be, hogy ha ugyanehhez az inputhoz két és félszer akkora ládák állnak rendelkezésre, akkor a konkrét feladatot az FF algoritmus meg tudja oldani legfeljebb 21 ládával. Bizonyítsuk be, hogy 21 ládánál kevesebbet az FF algoritmus nem tud garantálni.

    21 nagy ládánál kevesebbet sem az FF, sem más algoritmus nem tud garantálni. Ha ugyanis 42 db $0,9$ méretű tárgyat kell elpakolni, akkor ahhoz mindenképp 42 kis ládára ill. 21 nagy ládára van szükség.

    Az FF kisládás megoldása segítségével definiáljuk a következő referenciamegoldást 21 nagy ládára. Az első két kisláda tartalmát pakoljuk az első nagyba, a harmadik és negyedik kisláda tartalmát a második nagy ládába, és így tovább. Látjuk, hogy 21 láda elég, így most azt igazoljuk, hogy alkalmas sorrend esetén az FF algoritmus is garantálja-e legfeljebb 21 láda használatát.

    Az FF algoritmust most a tárgyak olyan sorrendjében alkalmazzuk, hogy előre vesszük a referenciamegoldás szerint az első nagy ládába pakolt dolgokat, ezek után jön a második nagy láda tartalma és így tovább. Ekkor minden tárgy vagy a referenciamegoldás szerinti ládájába kerül, vagy egy azt megelőzőbe. Ezért az FF algoritmus ilyen sorrend esetén legfeljebb a referenciamegoldásban szereplő 21 nagy ládát fogja használni.

    Megjegyzések: (1) A fenti bizonyítás garantálja, hogy nem lesz szükség 21-nél több ládára. Elképzelhető persze, hogy kevesebb láda is elég. Ha pl 84 db $0,35$ méretű tárgyat kell elcsomagolni, akkor az FF-nek kis ládából 42 kell, nagyból viszont csak 12.

    (2) A fenti bizonyítás akkor is működik, ha a nagy ládák nem két és félszer, hanem csak kétszer akkorák mint a kicsik. Azonban ebben az esetben az FF algoritmus nem biztos, hogy boldogul 21 ládával akkor, ha nem módosítunk a tárgyak elcsomagolási sorrendjén. Ha pl. a doboz mérete (a törtekkel történő számolás elkerülése végett) 10, és a tárgyak $6, 6, 6, 6, 5, 5, 5, 5, 4, 4, 4, 4$ sorrendben érkeznek (tehát az FFD-ről van szó valójában), akkor pontosan 6 ládára van szükség. Ha azonban a láda mérete 20, akkor már 4 láda kell ugyanehhez a sorrendhez, 3 láda nem elég. Tanulságos ezt ellenőrizni.

    (3) Ha a nagy ládák két és félszeresei a kicsiknek, akkor nem világos, hogy a (2) megjegyzésben bemutatott jelenség előfordulhat-e. Nevezetesen, hogy a referenciamegoldásban szereplő nagy ládaszám nem elég az FF algoritmusnak, ha ugyanabban a sorrendben pakol. Ezt se bebizonyítani, se megcáfolni nem tudom.