Kombinatorikus optimalizálás 2020. tavasz

6. gyakorlat 2020. május 8.

Tudnivalók

Közelítő algoritmusok

Def: Tfh egy problémának minden I inputjához tartozik egy X_I megoldáshalmaz, a cél pedig olyan $x \in X_I$ megoldás keresése, amelyik az $f_I(x_I)$ célfüggvényt minimalizálja. Az A algoritmus *additív hibája* legfeljebb C, ha tetszőleges I inputhoz olyan y_I megoldást ad, amire $f_I(y_I) \leq \min\{f_I(x_I) + C : x_I \text{ az } I \text{ megoldása}\}$.

Példa: (1) $\chi'(G)$ meghatározása. Tetszőleges egyszerű gráf éleinek $\Delta(G)+1$ színnel történő színezésére van gyors eljárás, így a $\chi'(G) \geq \Delta(G)$ triviális becslés miatt ennek az additív hibája legfeljebb 1.

(2) A gráf leghosszabb körének meghatározása tartalmazza a Hamilton-kör létezésének eldöntési problémáját, így NP-teljes (reménytelen rá polinomidejű algoritmust találni). Azonban konstans additív hibával sem lehet hatékonyan megoldani a feladatot, és ez abból látszik, ha az inputgráf minden élét egy C hosszúságú úttal helyettesítjük.

Def: Az A algoritmus *multiplikatív hibája* legfeljebb α, más szóval A egy α- *(avagy α-approximációs) algoritmus*, ha minden I inputra $f_I(y_I) \leq k \cdot \min\{f_I(x_I) + C : x_I \text{ az } I \text{ megoldása}\}$.

Példa: A minimális lefogó ponthalmaz $\tau(G)$ méretének meghatározása általában NP-teljes feladat, ám 2-közelítést kapunk, ha kiválasztunk egy nem bővíthető M párosítást G-ben, és a $V(M)$ lefogó ponthalmaz méretét adjuk válaszként.

Halmazfedési probléma Adott az U n-elemű alaphalmaz, az $S_1, S_2, \ldots, S_k \subseteq U$ részhalmazok, és ezek $c(S_i)$ nemnegatív költsége. Cél az S_i-k egy minimális összköltségű részrendszere, ami a teljes U alaphalmazt lefedi.

Mohó algoritmus a halmazfedési feladatra Egymás után választjuk az fedésbe bekerülő részhalmazokat. Mindig azt a halmazt választjuk, amelyik fajlagosan a legolcsóbban fedi az eddig le nem fedett pontokat, azaz azt az S_i-t, amire $c(S_i)/m$ minimális, ahol m az S_i által lefedett, de az eddig választott halmazok által le nem fedett U-beli elemek száma.

Tétel: n elemű alaphalmazon futtatva a mohó algoritmus $(1 + \ln n)$-közelítés a halmazfedési feladatra.

Ütemezési problémák **Input:** $J_1, J_2, \ldots$ munkák p_i megmunkálási idők, gépek m száma.

Feladat: A munkák gépekre ütemezése. Egy S ütemezésre $S_t(i)$ és $S_m(i)$ adja meg, hogy J_i-t mikor és melyik gépen kell elkezdeni. $C_i^S = S_t(i) + p(i)$ a J_i befejezési időpontja, $C_{max}^S = \max_i C_i^S$ pedig az S ütemezés átfutási ideje.

Megkötések: (1) (Itt) a gépek egyformák. (2) Egy gépen egyszerre egy munka végezhető. (3) Minden munkát megszakítás nélkül kell elvégezni.

Lehetséges feltételek J_i-kre: r_i (rendelkezésre állási idő), w_i (súly), d_i (határidő), $\prec$ (precedencia)

Optimalizálandó mennyiségek: C_{max}^S, $(\sum_i C_i^S)/n$

Tömör jelölés: $\alpha|\beta|\gamma$, ahol $\alpha \in \{1, Pm, P\}$ (1 gép, m gép, sok párhuzamos gép) $\beta \in \{p = 1, r_j, \prec\}$ (feltételek megadása) $\gamma \in \{C_{max}, \sum_i C_i\}$ (célfv megadása)

Példa: : (1) $1||C_{max}$: 1 gépen kell mihamarabb végezni. $OPT = \sum_i p_i$.
(2) $1|\prec|C_{max}$. Nem mindegy a sorrend, de itt is $OPT = \sum_i p_i$.

Tétel: Az $1||\sum_i C_i$ feladatra a munkák p_i szerint növekvő (SPT) sorrendben történő elvégzése optimális ütemezés. Az $1||\sum_i w_i C_i$ feladatra a munkák p_i/w_i szerinti növekvő sorrendben történtő elvégzése optimális ütemezés.

Tétel: (1) A $P2||C_{\max}$ probléma optimális megoldása reménytelen. (2) a $Pm||C_{\max}$ feladatra a listás ütemezést használó LS algoritmus $(2 - \frac{1}{m})$-approximációt ad. (Itt a soron következő munkát mindig az elsőnek felszabaduló gépen végezzük.) (3) Ha az LS algoritmust p_i szerint csökkenő (LPT) sorrendben végezzük, akkor $\frac{4}{3}$-közelítést kapunk.

Ládapakolási (bin packing) feladat
Minél kevesebb egységnyi térfogatú ládába kell bepakolni az $a_1, a_2, \ldots, a_n$ méretű tárgyakat.

FF (first fit) algoritmus minden tárgyat az első olyan ládába teszünk, amibe elfér.

FFD (first fit decreasing) algoritmus Az FF algoritmust a méretek csökkenő sorrendjében futtatjuk.

Tétel: Az FF algoritmus legfeljebb $\frac{17}{10}OPT + 2$, az FFD legfeljebb $\frac{11}{9}OPT + 7$ ládát használ.

Gyakorlatok

1. 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 opt fedés maradék halmazai megfelelő multiplicitással fedik a még lefedendő pontokat.) Így aztán az összköltségre a felső becslésre $OPT \cdot (\frac{1}{2n} + \frac{1}{2n-1} + \ldots + \frac{1}{1})$ adódik.

2. 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 a mennyiségnek kell a minimumát képezni ℓ-lel. Az S_i szerinti fajlagos költség tehát $f(S^i) = c(S^i)/\min\left(\ell, \left|U \setminus (S \cup S^2 \cup \ldots \cup S^{i-1})\right|\right)$. 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: $\sum_i c(S^i) \leq OPT \cdot (1/k + 1/(k-1) + \ldots + 1/1) \leq OPT \cdot (1 + lnk)$.
3. 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.
4. 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'-évek, tehát minimális.
5. 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.

   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.

6. 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.

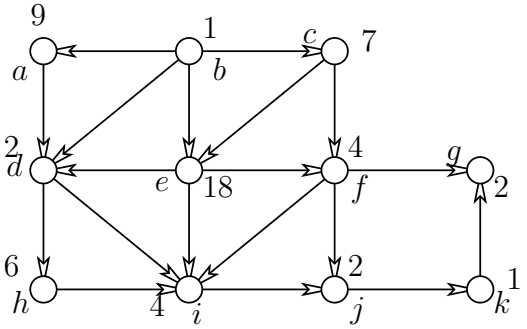
   Az órán tanultak szerint az SPT sorrendben történő ütemezés adja a lekisebb átlagos átfutási időt, ezért azt kell megmutatni, hogy ebben a sorrendben ez a mennyiség legfeljebb 24. A legrövidebb munka hossza legfeljebb $42/7 = 6$. A két legrövidebb munka összhossza legfeljebb $2 \cdot 42/7 = 12$, sít, 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.

7. 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.

8. 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.

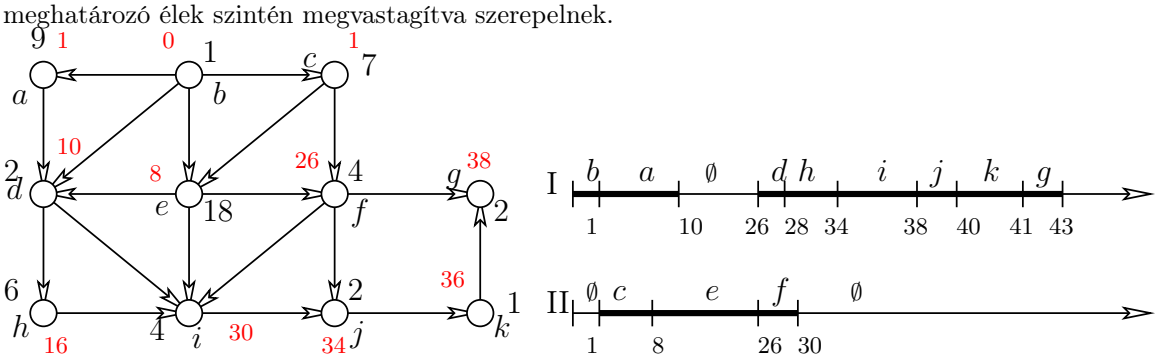


   A munkákat olyan sorrendben kell elvégezni, amelyben a gráf minden éle balról jobbra mutat. Ez a topologikus sorrend definíciója, ilyet kell keresni pl. forrástörléssel. Ez a sorrend adja meg az ütemezést, és az átfutási idő természetesen az összmegmunkálási idő, konkrétan 56 lesz. (Topologikus sorrend 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.

9. 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 második és harmadik kisláda tartalmát a második nagy ládába, és így tovább.

   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 ésígy 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.

(4) A feladatra korábban feltöltött megoldás hibás volt. (Ez ugyanazzal az FF sorrenddel próbált bizonyítani, de sajnos előfordulhat ilyenkor, hogy egy tárgy a referenciamegoldáshoz képest későbbi nagy ládába kerül.)