Kombinatorikus optimalizálás 2024. II. félév

10. gyakorlat. Fleiner Tamás feladatsora alapján összeállította: Héger Tamás (heger@cs.bme.hu)

Tudnivalók.

Közelítő algoritmusok **Definíció.** 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 (abszolút) hibája* legfeljebb C, ha tetszőleges I inputhoz olyan $A(I)$ megoldást ad, amire $f_I(A(I)) \leq \min\{f_I(x_I) \colon x_I \in X_I\} + C$, azaz az A kimenetére a célfüggvény értéke legfeljebb C-vel nagyobb az optimumnál.

Példa. (1) $\chi'(G)$ meghatározása. Tetszőleges egyszerű gráf éleinek $\Delta(G)+1$ színnel történő színezésére van gyors eljárás, így a $\chi'(G) \geq \Delta(G)$ 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.

Definíció. Az A algoritmus *multiplikatív (relatív) hibája* legfeljebb α, más szóval A egy *α-közelítő (avagy α-approximációs) algoritmus*, ha minden I inputra $f_I(A(I)) \leq \alpha \cdot \min\{f_I(x_I) \colon x_I \in X_I\}$, azaz az A kimenetére a célfüggvény értéke legfeljebb az optimum α-szorosa.

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 (ekkor $|M| \leq \nu(G)$, és az M-beli élek $2|M|$ végpontját adjuk kimenetként (ez mindig lefogó ponthalmaz lesz, és $\nu(G) \leq \tau(G) \leq 2|M| \leq 2\nu(G) \leq 2\tau(G)$ miatt legfeljebb kétszer nagyobb az optimumnál).

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 $c(S_i)/m_i$ minimális, ahol m_i az S_i által lefedett, de az eddig választott halmazok által le nem fedett U-beli elemek száma.

Tétel. (1) A fenti mohó algoritmust n elemű alaphalmazon futtatva a halmazfedési feladat egy $(1+\ln n)$-közelítését kapjuk, azaz a kimenet összköltsége az optimális összköltségnek legfeljebb $(1+\ln n)$-szerese.

(2) A halmazfedési feladatra nem létezik $konst \cdot \ln n$-nél lényegesen jobb approximáció.

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

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 az S ütemezésnél, $C_{max}^S = \max_i C_i^S$ az S átfutási ideje.

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

Néhány lehetséges további feltétel J_i-kre: r_i (legkorábbi rendelkezésre állás időpontja), w_i (súly), d_i (határidő), $\prec$ (precedencia, azaz sorrendiségi feltételek)

Optimalizálandó (minimalizálandó) mennyiségek: C_{max}^S (átfutási idő), $(\sum_i C_i^S)/n$ (átlagos átfutási idő; ez ekvivalens $\sum_i C_i^S$ minializálásával), $(\sum_i w_i C_i^S)/n$ (súlyozott átlagos átfutási idő).

Tömör jelölés: $\alpha|\beta|\gamma$, ahol $\alpha \in \{1, Pm, P\}$ (1 gép, m gép, sok párhuzamos gép)
$\beta \in \{p=1, r_i, d_i, \prec\}$ (feltételek megadása) $\gamma \in \{C_{max}, \sum_i C_i, \sum_i w_i C_i\}$ (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 optimális ütemezés a munkák p_i szerint növekvő (SPT) sorrendben történő elvégzése. Az $1||\sum_i w_i C_i$ feladatra optimális ütemezést ad p_i/w_i szerinti növekvő sorrend.

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. A tárgyakat valamilyen sorrendben véve, minden tárgyat az első olyan ládába teszünk, amibe elfér.

FFD (first fit decreasing) algoritmus. A tárgyakat méret szerint csökkenő sorrendbe rendezve futtatjuk az FF algoritmust.

Tétel. Az FF algoritmus legfeljebb $\frac{17}{10}OPT + 2$, az FFD legfeljebb $\frac{11}{9}OPT + 7$ ládát használ (azaz durván $1,7$-, illetve $1,22$-közelítő algoritmusok).

Gyakorlatok

1. a) 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.

b) 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.

Megoldás: **a)** A nem bővíthető párosítás végpontjai egy lefogó ponthalmazt adnak, ezért, ha M nem bővíthető, akkor $2|M| \geq \tau(G) \geq \nu(G)$, tehát a talált M párosítás mérete valóban legalább az optimum (itt ν) $\frac{1}{2}$-szerese.

b) Vegyük az M és egy maximális méretű M' párosítás unióját. A keletkező gráf komponensei élek, körök, vagy utak (hiszen minden csúcs foka legföljebb 2). Csak az utóbbiakban lehet M'-nek több éle, mint M-nek (az ilyen utak javító utak M-re nézve), ezekben k db M-beli és $k+1$ db M'-beli él van valami k-ra. Ha nincs M-hez 3 élű javító út (azaz az előbbi k mindig legalább 2), akkor minden javító út legfeljebb $3/2$-szer annyi új élt tartalmaz, mint régit, így $\nu = |M'| \leq 32|M|$, másképp szólva $\nu \cdot \frac{2}{3} \leq |M|$, tehát az algoritmusunk valóban eléri az optimumnak legalább a $\frac{2}{3}$-át.

2. A $H = \{a,b,c,d,e\}$ alaphalmazt szeretnénk minél olcsóbban lefedni. A fedéshez az alábbi részhalmazokat használhatjuk, minden részhalmaz után a költsége áll: $\{a\}$, 2; $\{a,b\}$, 4; $\{a,d\}$, 4; $\{a,c,e\}$, 3; $\{b,c,e\}$, 4; $\{c,d\}$, 6; $\{b,d\}$, 7.

a) Állapítsuk meg, hogy a mohó algoritmus milyen költséggel tudja lefedni a H halmazt.

b) Lehetséges-e valamelyik részhalmaz költségét úgy növelni, hogy a mohó algoritmus olcsóbb fedést szolgáltasson?

Megoldás: **a)** Futtassuk a mohó algoritmus. Ehhez készítsünk egy táblázatot, melyben az oszlopok a választható részhalmazoknak felelnek meg, és minden sorba felírjuk az adott részhalmaz által újonnan fedett elemeinek fajlagos költségét (azaz a halmaz költségét elosztjuk a részhalmazban levő, jelenleg még fedetlen elemek számával). A minimális fajlagos költségű részhalmazt bevesszük a fedésbe (ezt keretezés fogja jelezni); ennek az elemei tehát már le lesznek fedve, így a fajlagos költségeket újra kell számolni, ezt fogja tartalmazni a következő sor. (Ha egy részhalmaznak már minden eleme le van fedve, azt nincs értelme bevenni; ennek tényét kihúzással jelöljük.) Ezt addig ismételjük, míg a kiválasztott részhalmazok együttesen az alaphalmaz összes elemét le nem fedik.

részhalmaz és költsége	$\{a\}$, 2	$\{a,b\}$, 4	$\{a,d\}$, 4	$\{a,c,e\}$, 3	$\{b,c,e\}$, 4	$\{c,d\}$, 6	$\{b,d\}$, 7
1. lépés, fajlagos ktsg-ek	2	2	2	$\boxed{1}$	$\frac{4}{3}$	3	$\frac{7}{2}$
2. lépés, fajlagos ktsg-ek	–	4	4	–	4	6	$\boxed{\frac{7}{2}}$

A mohó algoritmus tehát az $\{a,c,e\}$ és a $\{b,d\}$ részhalmazokat veszi be a fedésbe, összesen $3+7=10$ költséggel. Ezek persze valóban lefedik az alaphalmazt.

b) Lehetséges, növeljük az $\{a,c,e\}$ halmaz költségét mondjuk 30-ra. Ekkor a mohó algoritmus futása:

részhalmaz és költsége	$\{a\}$, 2	$\{a,b\}$, 4	$\{a,d\}$, 4	$\{a,c,e\}$, 30	$\{b,c,e\}$, 4	$\{c,d\}$, 6	$\{b,d\}$, 7
1. lépés, fajlagos ktsg-ek	2	2	2	10	$\boxed{\frac{4}{3}}$	3	$\frac{7}{2}$
2. lépés, fajlagos ktsg-ek	2	4	$\boxed{2}$	30	–	6	7

A mohó algoritmus a második lépésben az $\{a\}$ részhalmazt is választhatná, de a jelzett $\{a,d\}$-t is. Így a kapott fedés a $\{b,c,e\}$ és $\{a,d\}$ részhlamazokat tartalmazza $4+4=8$ összköltséggel.

3. Gyakoroljuk az SPT, LS, illetve LPT ütemezést, továbbá az FF és FFD ládapakolási eljárásokat konkrét példákon.

4. a) Mutassunk példát, melyben az FF algoritmus másfélszer annyi ládát használ föl, mint az optimum.

b) Igaz-e, hogy a ládapakolási feladatban mindig található a tárgyaknak olyan sorrendje, melyre az FF algoritmus optimális szmú ládát használ?

Megoldás:

5. 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.

Megoldás: 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'-ével, tehát minimális.

6. Tegyük fel, hogy a $J_1, J_2, \ldots$ munkákat az S ütemezés úgy ütemezi 42 gépre, hogy az átfutási idő 42, az átlagos átfutási idő pedig 24. Mutassuk meg, hogy ugyanezek a munkák ütemezhetők 21 gépre úgy, hogy az átfutási idő legfeljebb 84, az átlagos átfutási idő pedig legfeljebb 45 legyen.

Megoldás: Könnyű olyan ütemezést készíteni 21 gépre, amivel a munkák átfutási ideje legfeljebb 84 lesz: képezzünk az S ütemezésben szereplő 42 gépből 21 párt, és az új, 21 gépes ütemezésben az eredetiben az i-edik géppárra ütemezett munkákat ütemezük az újban az i-edik gépre. Mivel eredetileg bármely gépen az elvégzett munkák összdeje legfeljebb 42, az egy pár munkáit végző gépen levő munkák összes megmunkálási ideje legfeljebb 84.
Vegyük azt az új S' ütemezést, ahol egy géppár munkáit úgy ütemezzük az új gépre, hogy először a több munkát végző gép munkáit $(J_1^1, \ldots, J_k^1)$ végezzük el az eredeti sorrendjükben, majd a kevesebb (vagy ugyanannyi) munkát végző gép munkáit $(J_1^2, \ldots, J_l^2,\ l \le k)$ vesszük szintén az eredeti sorrendjükben. Ekkor az átlagos átfutási idő (számlálójában) szereplő $\sum C_i^S$ tagban a J_i^1 munka befejezési idejére $C^S(J_i^1) = C^{S'}(J_i^1)$, míg a pár második gépének munkáit legfeljebb 42-vel később kezdjük, tehát $C^{S'}(J_i^2) \le C^S(J_i^2) + 42$. Ha összesen n munkát kell ütemezni, legfeljebb $n/2$ munka befejezési idejéhez adunk hozzá 42-t, tehát $\sum C_i^{S'} \le \sum C_i^S + (n/2)\cdot 42$, így az átlagos átfutási időre fennáll, hogy

$$\frac{\sum C_i^{S'}}{n} \le \frac{\sum C_i^S + (n/2)\cdot 42}{n} = \frac{\sum C_i^S}{n} + 21 = 24 + 21 = 45.$$

Hab a tortán, hogy az új gépeken SPT szerint ütemezve még ennél is jobbat kaphatunk.

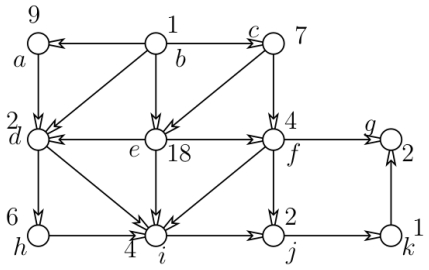
7. Tegyük fel, hogy 7 munka összvégrehajtási ideje 42. Igazoljuk, hogy lehetséges ezeket a munkákat egy gépre ütemezni úgy, hogy az átlagos átfutási idő legfelejebb 24 legyen.

Megoldás: 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 (ez azzal egyenértékű, 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.

8. 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?

Megoldás: 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.

9. 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.



Megoldás: 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 (ami az összes munka elkészült állapotának felel meg), 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 (még SZA-ból). Nem nehéz megoldani: vesszük a csúcsoknak egy topologikus sorrendjét (amit már meg is tettünk az előbb), és abban balról jobbra haladba minden csúcshoz feljegyezzük az oda vezető leghosszabb út hosszát. Mivel egy v csúcsba csak tőle balról jönnek élek (így utak is), egyszerűen megnézünk minden v-be belépő uv élt, uv hosszát hozzáadjuk az u-ba vezető leghosszabb út hosszához (ezt már megnéztük u-ra, hiszen az v-nél balrább van), és ezek közül az összegek közül a legnagyobb lesz a v-be vezető leghosszabb út hossza. Ha még az it feljegyezzük, hogy honnan érkezett a leghosszabb út v-be (tehát hogy melyik él mentén realizálódott a maximális összeg), a végén könnyen visszafejthetünk egy leghosszabb utat. Táblázatban összefoglalva:

csúcs	b	a	c	e	d	h	f	i	j	k	g	t
max út hossza	0	1	1	8	10	12	26	30	34	36	37	39
honnan ekkora	–	b	b	c	a	d	e	f	i	j	k	g

Tehát az átfutási idő 39 lesz, a leghosszabb munkalánc a táblázatból kiolvasva g, k, j, i, f, e, c, b (fordított sorrendben).

A kétgépes ütemezés során az alábbi események követik egymást :

- $t = 0$-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
- $t = 34$-ben i-t elkezdjük az I. gépen.
- $t = 38$-ban j-t elkezdjük az I. gépen.
- $t = 40$-ben k-t elkezdjük az I. gépen.
- $t = 41$-ben g-t elkezdjük az I. gépen.
- $t = 43$-ban g befejeződik az I. gépen és ezzel minden munkát végrehajtottunk.

10. 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.

Megoldás: 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.

11. 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.

12. 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 (különböző) 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?

Megoldás: 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.

13. 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?