Gyakorlatokhoz emlékeztető

1. Bevezető példák, síbérlés

1.1. Feladat Adott egy parkoló, ahol egy professzor a kocsiját tartja. A parkolóhelyeket egy $-n$ és n közötti szám azonosítja, az azonosító szerint helyezkednek el balról jobbra. A professzor kijön az egyetemről a 0 parkolóhelynél. Nem tudja hol parkol, és meg akarja keresni a kocsiját. Adjunk egy konstans versenyképes algoritmust, ami megtalálja a kocsit.

Megold: A következő algoritmusokat vizsgáljuk meg:

1.) A természetes algoritmus (előbb valamelyik irányba elmegy a parkoló végéig, és ha nem találja meg a kocsit visszajön) nem konstans versenyképes, mert a legrosszabb esetben a kocsi a másik oldal első parkolójában van, így az ALG/OPT arány $2n+1$.

2.) A következő gondolat, hogy a 2i+1-dik fázisban elmegyünk jobbra az i-edik helyig, majd onnan a 2i+2-dik fázisban balra az -i-edik helyig. És ezeket a fázisokat hajtjuk végre, amíg meg nem találtuk a kocsit. Ez az algoritmus sem konstans versenyképes, hiszen ha a kocsi a $-n$ helyen áll, akkor az optimális költség n az algoritmus költsége $4\sum_{i=1}^{n-1} i + 3n = n(2n+1)$.

3.) Ezt követően adódik a gondolat, hogy kevesebb irányváltoztatással keressük a kocsit. A 2i+1-dik fázisban elmegyünk jobbra az 2^i-edik helyig, majd onnan a 2i+2-dik fázisban balra az -2^i-edik helyig. És ezeket a fázisokat hajtjuk végre, amíg meg nem találtuk a kocsit (értelemszerűen, ha $2^i > n$, akkor csan n-ig megyünk). Vizsgáljuk ezt az algoritmust, tegyük fel, hogy a kocsi a k helyen van. Legyen i olyan kitevő, amelyre $2^i < |k| \leq 2^{i+1}$. Ekkor az algoritmus által megtett lépések száma kisebb, mint $4\sum_{j=1}^{i+1} 2^j < 4 \cdot 2^{i+2} \leq 16|k|$, azaz az algoritmus valóban konstans versenyképes.

1.2. Feladat Vegyük a következő véletlenített algoritmust a síbérlési feladatra (a sí vásárlási ára T egység). Az R algoritmus 1/2 valószínűséggel a 3/4 T időpontig vár és utána vásárol, 1/2 valószínűséggel pedig a T időpontban. Igazoljuk, hogy az algoritmus 15/8-versenyképes.

Megold: Vegyünk egy tetszőleges inputot, jelölje I. Azaz I napig síelünk. Különböztessük meg a következő eseteket.

Ha $I < 3/4T$, akkor az optimális költség I, továbbá R költsége is I mindkét döntés esetén, így az $E(R(I))/OPT(I)$ arány 1.

Ha $3/4T \leq I < T$, akkor az optimális költség I, továbbá R költsége vagy $T+3/4T-1$ vagy I. Tehát $E(R(I)) \leq 1/2 \cdot 7/4 \cdot T + 1/2 \cdot I$. Tehát felhasználva, hogy $I \geq 3/4T$ kapjuk, hogy $E(R(I))/OPT(I) \leq (1/2 \cdot 7/4 \cdot T + 1/2 \cdot I)/I \leq 1/2 \cdot 7/4 \cdot 4/3 + 1/2 = 10/6$.

Ha $T \leq I$, akkor $OPT(I) = T$. Az R algoritmus költsége pedig vagy $T + 3/4T - 1$ vagy $2T - 1$, így $E(R(I)) \leq 1/2 \cdot 7/4 \cdot T + 1/2 \cdot 2 \cdot T = 15/8T$. Következésképpen $E(R(I))/OPT(I) \leq 15/8$.

1.3. Feladat Adjunk a Yao elv alapján egy alsó korlátot a síbérlési problémát megoldó véletlenített algoritmusok versenyképességi hányadosára. Használjuk a következő valószínűségi eloszlást. Az input legyen $1/2$ valószínűséggel $T/2$ és $1/2$ valószínűséggel $3/2 \cdot T$.

Megold Elsőként vegyük észre, hogy a két lehetséges input közül, az első esetén az optimális költség $T/2$ a második esetében T. Yao elv használatához meg kell határoznunk azt az optimális determinisztikus algoritmusra az $A(I)/OPT(I)$ hányados várható értékét. Vegyünk egy tetszőleges determinisztikus online algoritmust. Ezt egyértelműen meghatározza egyetlen érték, ami azt adja meg hányadik napot követően vásárol sílécet, ha addig nem fejeződik be a síelés. Jelölje az algoritmust meghatározó értéket x. Különböztessünk meg három esetet x értékétől függően.

Ha $x \leq T/2$, akkor az algoritmus költsége mindkét lehetséges inputon $x+T$. Tehát $E(A(I)/OPT(I)) = 1/2(x+T)/(T/2) + 1/2(X+T)/T \geq 3/2$.

Ha $T/2 < x \leq 3/2T$, akkor az algoritmus költsége az első lehetséges inputra $T/2$ a másodikra $X+T$, így $E(A(I)/OPT(I)) = 1/2(T/2)/(T/2) + 1/2(X+T)/T \geq 1/2 + 1/2 \cdot 3/2 = 5/4$.

Ha $3/2T \leq X$, akkor az algoritmus költsége az első lehetséges inputra $T/2$ a másodikra $3/2T$, így $E(A(I)/OPT(I)) = 1/2(T/2)/(T/2) + 1/2(3/2 \cdot T)/T = 5/4$.

Következésképpen minden determinisztikus online algoritmusra teljesül, hogy az adott input eloszlás mellett $E(A(I)/OPT(I)) \geq 5/4$, így a Yao elv alapján adódik, hogy nincs véletlenített algoritmus, amelynek a versenyképességi hányadosa kisebb lenne, mint $5/4$.

2. Lapozás és k-szerver

2.1. Feladat Igazoljuk, hogy a LFU (legkevésbé gyakorian használt) lapozási stratégia nem versenyképes.

Megold: tegyük fel, hogy kezdetben az $s_1, \ldots, s_k$ lapok vannak a memóriában. Vegyük a következő sorozatot: $(s_1)^n, \ldots, (s_k)^n, (a,b)^n - 1$. Ekkor LFU az utolsó 2n-2 kérés esetén hibát okoz, hiszen mindig az a,b lapok valamelyikét rakja ki a memóriából. Ezzel szemben az optimáis offline algoritmus b-t is valamely s_i lap helyére teszi be, és a teljes költsége kettő hiba.

2.2. Feladat Legyen $k = 3$, tegyük fel, hogy kezdetben űres a memória. Hajtsuk végre az $A, B, C, A, B, D, A, C, B$ sorozaton az LRU, FIFO, LFD algoritmusokat.

Megold: A memória tartalmának változásait írjuk fel, zárójelben jelölve, hogy hányadik kérést követően változott.

LRU: $A, -, -(1)$, $A, B, -(2)$, $A, B, C(3)$, $A, B, D(6)$, $A, C, D(8)$, $A, C, B(9)$.

FIFO: $A, -, -(1)$, $A, B, -(2)$, $A, B, C(3)$, $D, B, C(6)$, $D, A, C(7)$, $D, A, B(9)$

LFD: $A, -, -(1)$, $A, B, -(2)$, $A, B, C(3)$, $A, D, C(6)$, $(B, D, C(9)$, mivel B az utolsó elem feltesszük, hogy LFD a első memóriahelyre teszi, hiszen nincs további kérés egyetlen lapra sem.

2.3. Feladat Igazoljuk, hogy a lapozási probléma a k-szerver feladat speciális esete.

Megold: Legyen a metrikus tér egy uniform (bármely két különböző pont távolsága 1) tér, amelyben a pontok a lehetséges lapok. A gyorsmemória celláit feleltessük meg a szervereknek, a szerver mindig azon a ponton van, amely lap az adott cellában szerepel. Könnyen látható, hogy az így kapott speciális k-szerver feladat ekvivalens a lapozási problémával.

2.4. Feladat Az egyensúly algoritmus viselkedése: Tekintsük a kétdimenziós Euklideszi teret, mint metrikus teret. A pontok (x, y) valós számpárokból állnak, két (a, b) és (c, d) pontnak a távolsága $\sqrt{(a-c)^2 + (b-d)^2}$. Legyen két szerverünk, kezdetben a $(0, 0)$ és $(1, 1)$ pontokban. A kérések sorozata legyen az $(1, 4)$, $(2, 4)$, $(1, 4)$ pontsorozat. Hajtsuk végre az EGYENSULY algoritmust!

Megold: A kiindulási állapotban $D_1 = D_2 = 0$, $s_1 = (0, 0)$, $s_2 = (1, 1)$. Az első kérés az $(1, 4)$ pontban van. Ekkor $D_1 + d((0, 0), (1, 4)) = \sqrt{17} > D_2 + d((1, 1), (1, 4)) = 3$, így a második szervert használjuk és a kérés kiszolgálása után $D_1 = 0, D_2 = 3$, $s_1 = (0, 0)$, $s_2 = (1, 4)$ teljesül. A második kérés után $D_1 + d((0, 0), (2, 4)) = \sqrt{20} > D_2 + d((1, 4), (2, 4)) = 3 + 1 = 4$, így ismét a második szervert használjuk, és a kérés kiszolgálása után $D_1 = 0, D_2 = 4$, $s_1 = (0, 0)$, $s_2 = (2, 4)$ teljesül. A harmadik kérésnél $D_1 + d((0, 0), (1, 4)) = \sqrt{17} < D_2 + d((2, 4), (1, 4)) = 4 + 1 = 5$, így az első szervert használjuk és a kérés kiszolgálása után $D_1 = \sqrt{17}, D_2 = 4$, $s_1 = (1, 4)$, $s_2 = (2, 4)$ teljesül.

2.5. Feladat Legyen a metrikus tér egy egyenes. Tegyük fel, hogy három szerverünk van, s_1, s_2, s_3 amelyek az egyenes $0, 1, 2$ pontjaiban helyezkednek el. Hajtsuk végre a Dupla Lefedő (DL) algoritmust a 4, 2 pontsorozatra.

Megold: Az első kérésnél DL a legközelebbi s_3 szervert küldi a kérés kiszolgálására. A többi szerver helye nem változik, a költség 2 és a kérés kiszolgálása után a szerverek a $0, 1, 4$ pontokban lesznek. A második kérésnél DL a legközelebbi s_2 szervert küldi a kérés kiszolgálására, de mivel a kérés másik oldalán is van szerver, ezért s_3 is megtesz egy egységnyi utat a kérés felé, így a költség 2 és a kérés kiszolgálása után a szerverek a $0, 2, 3$ pontokban lesznek.

2.6. Feladat Igazoljuk, hogy a mohó algoritmus, ami minden kérést a legközelebbi szerverrel szolgál ki, nem versenyképes két szerver esetén az egyenesen.

Megold: Legyenek a szerverek kezdetben az A és B pontokban. A pontokat valós számokkal azonosítjuk. Az általánosság megszorítása nélkül feltehetjük, hogy $A < B$. Legyen $C = B + (B - A)/2$. A kérések sorozata legyen $(C, B)^n$. Ekkor a mohó algoritmus minden kérést a második szerverrel szolgál ki, így a költsége $2n(B-A)/2$. Másrészt bármely n esetén az optimális megoldás költsége legfeljebb $3(B - A)/2$, az első kérést az első szerverrel kiszolgálva többet nem kell mozogni. Következésképp a $MOHO(I)/OPT(I)$ hányados végtelenhez tart, ahogy n tart a végtelenbe.

3. Ütemezés

3.1. Feladat Tekintsük a hasonló párhuzamos gépek esetét, ahol 3 darab gép van és a sebességek $s_1 = s_2 = 1$, $s_2 = 3$. Vegyük a következő $I = (2, 1, 1, 3, 2)$ munkasorozatot, ahol a munkák a megmunkálási idők által vannak megadva, adjuk meg a MOHO algoritmus által adott ütemezést.

Megold: Ekkor az első munka 2/3-kor fejezhető be az M_3 gépen és az 1 időpontban a többi gépen, így M_3-hoz rendeljük. A következő munka az 1 időpontra fejezhető be az összes gépen, így M_3 kapja, mivel ott a legkisebb a megmunkálási idő. A következő munka az 1 időpontra fejezhető be M_1-en és M_2-n, és a 4/3 időpontban az M_3 gépen, így M_1 kapja. A negyedik munka M_1-en a 4, M_2-n a 3 időpontban fejeződne be, az M_3-on a 2 időpontban, így M_3-ra kerül. Végül az utolsó munka M_1-en 3, M_2-n a 2 időpontban fejeződne be, az M_3-on 8/3, így M_2-re kerül.

3.2. Feladat Tekintsük az általános párhuzamos gépek esetét, ahol 2 darab gép van. Vegyük a következő $I = ((1, 2), (1, 2), (1, 3), (1, 3))$ munkasorozatot, ahol a munkák a megmunkálási idővektorok által vannak megadva. Adjuk meg a MOHO algoritmus által adott ütemezést.

Megold: Ekkor az első munka az 1 időpontban fejezhető be az M_1 gépen és a 2 időpontban a második gépen, így M_1-hez rendeljük. A következő munka az 2 időpontra fejezhető be mindkét gépen, így M_1 kapja. A következő munka a 3 időpontra fejezhető be mindkét gépen, így ismét M_1 kapja. Végül az utolsó munka a 4 időpontra fejezhető be az M_1 gépen és a 3 időpontra fejezhető be az M_2 gépen, így az M_2 géphez rendeljük.

3.3. Feladat Tekintsük két párhuzamos gép esetét. Legyen a munkák sorozata a következő $I = (1, 0), (1/2, 0), (1/2, 0), (1, 3/2), (1, 3/2), (2, 2)$, ahol a munkákat a (megmunkálási idő, érkezési idő) párral adtuk meg. Adjuk meg az Intervallum algoritmus által adott ütemezést.

Megold: Az első iterációs részben az első három munkát ütemezzük, egy optimális ütemezés az első munkát az első géphez, a második és harmadik munkákat a második géphez rendeli, és az 1 időpontban befejezi a munkák végrehajtását. Utána, mivel nem érkezett új munka és nincs vége a sorozatnak várunk, egészen a 3/2 időpontig. Ekkor egy új iterációs rész kezdődik, az algoritmus ütemezi a negyedik és ötödik munkákat az első és második gépen, a munkákat az 5/2 időpontra fejezi be. Ekkor elkezdődik a harmadik iterációs rész, és az algoritmus ütemezi a hatodik munkát valamely gépen, az $[5/2, 9/2]$ intervallumra.

3.4. Feladat Igazoljuk, hogy a $P2|online|C_{max}$ (két egyforma gép, cél a max befejezési idő minimalizálása) feladatra nincs c-versenyképes algoritmus $c < 3/2$ értékkel.

Megold: Vegyünk egy tetszőleges algoritmust. Legyen az input első két munkájára a végrehajtási idő 1. Ha az algoritmus ugyanazon a gépen ütemezi őket a sorozat véget ért $A(I) = 2$ és $OPT(I) = 1$ tehát az algoritmus nem jobb, mint 2-versenyképes. Ha az algoritmus különböző gépekre rakja a

munkákat, akkor egy további munka jön 2 végrehajtási idővel. Ezt követően $A(I) = 3$, $OPT(I) = 2$, így az algoritmus versenyképességi hányadosa legalább 3/2. (LISTA két gépen 3/2-versenyképes, tehát jobb korlát nem adható).

3.5. Feladat Igazoljuk, hogy a $R2|online|C_{max}$ (két független gép cél a max befejezési idő minimalizálása) feladatra nincs c-versenyképes algoritmus $c < 2$ értékkel.

Megold: Vegyünk egy tetszőleges algoritmust. Legyen az input első munkájára a végrehajtási vektor $(1, 1)$. Ha az algoritmus az első gépen ütemezi, akkor a második munka vektora $(1, 2)$, így $A(I) = 2$ és $OPT(I) = 1$ tehát az algoritmus nem jobb, mint 2-versenyképes. Ha az algoritmus a második gépre rakja a munkát, akkor egy további munka jön $(2, 1)$ végrehajtási vektorral. Ezt követően ismét $A(I) = 2$, $OPT(I) = 1$, így az algoritmus versenyképességi hányadosa legalább 2. (MOHO két gépen 2-versenyképes, tehát jobb korlát nem adható).

3.6. Feladat Igazoljuk, hogy ha a $Pm||C_{max}$ feladatban minden munkára teljesül $p_i \leq \sum_{j=1}^{n} p_j/2m$, akkor a LISTA algoritmus $1 + \frac{m-1}{2m}$-versenyképes.

Megold: Legyen $\sigma = \{j_1, \ldots, j_n\}$ tetszőleges munkasorozat rendre $p_1, \ldots, p_n$ végrehajtási időkkel. Tekintsük a LISTA algoritmus által kapott ütemezést. Legyen j_l az a munka, amely legkésőbb fejeződik be. Vizsgáljuk ezen munka S_l kezdési idejét. Mivel egyetlen gép sem kezdte el a munkát ütemezni S_l előtt, ezért minden gép szünet nélkül dolgozott az S_l időpontig. Ebből azt kapjuk, hogy

$$S_l \leq \frac{1}{m} \sum_{\substack{j=1 \\ j \neq l}}^{n} p_j = \frac{1}{m}(\sum_{j=1}^{n} p_j - p_l) = \frac{1}{m}(\sum_{j=1}^{n} p_j) - \frac{1}{m} p_l.$$

Következésképp

$$LISTA(\sigma) = S_l + p_l \leq \frac{1}{m}(\sum_{j=1}^{n} p_j) + \frac{m-1}{m} p_l \leq (1 + \frac{m-1}{2m})(\frac{1}{m} \sum_{j=1}^{n} p_j).$$

Másrészt az optimális ütemezésben is végre kell hajtani az összes munkát, így $OPT(\sigma) \geq \frac{1}{m}(\sum_{j=1}^{n} p_j)$. Ezen becslések alapján egyből adódik, hogy

$$LISTA(\sigma) \leq (1 + \frac{m-1}{2m}) OPT(\sigma),$$

amivel bizonyítottuk, hogy LISTA $1 + \frac{m-1}{2m}$-versenyképes ebben az esetben.

3.7. Feladat Igazoljuk, hogy a visszautasításos ütemezési modellben az $RP(\alpha)$ nem konstans versenyképes minden m-re.

Megold: Elsőként vegyünk egy listát, amely valamely nagy N-re Nm darab tárgyat tartalmaz, amelyek mindegyike az $(1, \alpha)$ paraméterekkel rendelkezik. Ekkor minden munkát elutasítunk, a költség $Nm\alpha$. Ha elfogadjuk a munkákat

párhuzamosan ütemezhetünk és az ütemezés költsége N. A két költség hányadosa $m\alpha$.

Most vegyünk egy listát, amely egyetlen tárgyat tartalmaz, amelynek a paraméterei $(\alpha+\varepsilon, 1)$. Ekkor a munkát elfogadjuk és ütemezzük a költség 1. Ha a munkát elutasítjuk a költség $(\alpha+\varepsilon)$. Így a két költség hányadosa $1/(\alpha+\varepsilon)$.

Tehát az algoritmus versenyképességi hányadosa nem lehet kisebb, mint $\max\{m\alpha, 1/(\alpha+\varepsilon)$, amely érték tart a $\sqrt{m}$ értékhez, ha ε tart 0–hoz, amivel igazoltuk, hogy az algoritmus nem lehet konstans versenyképes minden m-re.

3.8. Feladat Igazoljuk, hogy ha az INTV algoritmusban az optimális megoldás helyett mindig egy c-approximációs megoldást használunk, akkor az algoritmus 2c-versenyképes.

Megold: Vizsgáljuk meg az INTV algoritmus 2-versenyképességének a bizonyítását. Ott a $T_1+T_3-T_2 \leq OPT(I)$ és $T_2-T_1 \leq OPT(I)$ egyenlőtlenségeket használtuk, ahol T_3 a befejezési idő T_2 az utolsó T_1 pedig az utolsó előtti fázis kezdete. Amennyiben a fázisokban c-approximációs algoritmusokat használunk, akkor mindkét egyenlőtlenségben $c\cdot OPT(I)$ teljesül felső korlátnak, és a feladat állítása következik.

3.9. Feladat Hajtsuk végre a TRP(1/2) algoritmust a következő munkasorozatra az 5 gépes visszautasításos ütemezési problémára: (6,1), (6,1), (9,2), (9,2) (9,2), (18,4). A munkák a végrehajtási idő büntetés párokkal vannak megadva.

Megold: Az első két munkára $6/5 \geq 1$, így az algoritmus az első lépésében visszautasítja őket. A harmadik munkánál $9/5 < 2$ de $9/2 > 2$, így az algoritmus a második lépésben utasítja vissza. A negyedik munkánál $9/5 < 2$ de $9/2 > 2+2$, így az algoritmus a második lépésben utasítja vissza. Az ötödik munkánál $9/5 < 2$ és $9/2 \leq 4+2$, így az algoritmus elfogadja. A hatodik munkánál $18/5 \leq 4$ de $18/2 > 4+4$, így az algoritmus a második lépésben utasítja vissza.

3.10. Feladat Igazoljuk, hogy a visszautasításos ütemezési modellben két gépre nincs kisebb versenyképességű hányadossal rendelkező algoritmus, mint $(1+\sqrt{5})/2$. Vizsgáljuk meg, mit kaphatunk a módszer kiterjesztésével több gép esetén.

Megold: Tegyük fel, hogy egy algoritmus versenyképességi hányadosa kisebb, mint $(1+\sqrt{5})/2$. Legyen az első munka $((1+\sqrt{5})/2, 1)$. Ekkor a munkát az algoritmusnak vissza kell utasítani. A második munka legyen $((1+\sqrt{5})/2, (1+\sqrt{5})/2)$. Ekkor az optimális költség $(1+\sqrt{5})/2$, az algoritmus költsége $(1+\sqrt{5})/2+1$, így a két költség hányadosa $(1+\sqrt{5})/2$, azaz azt kaptuk, hogy az algoritmus versenyképességi hányadosa mégsem lehet kisebb, mint $(1+\sqrt{5})/2$.

Az általános esetben legyen c az $x^{m-1}+x^{m-2}+\cdots+1 = x$ egyenlet megoldása.Tegyük fel, hogy van olyan algoritmus, amelynek kisebb a versenyképességi hányadosa, mint c. Vegyük a következő munkasorozatot. Legyen az i-edik munka $(1, 1/c^i)$ $i = 1, \ldots, m-1$ és az m-edik munka $(1,1)$. Ha az algoritmus az első $m-1$ munka közül valamelyiket elfogadja, akkor a sorozat véget ér. Tegyük fel, hogy a j-edik munkát fogadta elsőként. Ekkor az algoritmus költsége $1+\sum_{i=1}^{j-1} 1/c^i$ az optimális költség $\sum_{i=1}^{j} 1/c^i$, így a hányados c, ami

ellentmondás. Tehát az algoritmus el kell fogadja az első $m-1$ munkát így a költsége függetlenül attól, hogy az utolsó munkát elfogadja -e $1+\sum_{i=1}^{m-1} 1/c^i$, míg az optimális költség 1. Tehát a hányados ismét c, amivel ismét ellentmondáshoz jutottunk.

4. Ládapakolás

4.1. Feladat Egy ládapakolási feladatról tudjuk, hogy minden tárgy mérete legfeljebb 1/5, igazoljuk, hogy ekkor a NF algoritmus aszimptotikus versenyképességi hányadosa 5/4.

Megold: Mivel minden tárgy mérete legfeljebb 1/5, ezért az NF algoritmus által kapott pakolásban, az utolsó ládán kívül minden ládában legalább 4/5 a ládába pakolt tárgyak méreteinek összege (ellenkező esetben meg belefért volna a következő láda első eleme). Következésképpen $\sum_{i\in I} a_i \geq 4/5(NF(I)-1)$. Másrészt mivel az optimális algoritmusnak is el kell helyeznie az összes tárgyat a ládákban, ezért $OPT(I) \geq \sum_{i\in I} a_i$. Következésképpen $NF(I) \leq 5/4OPT(I) + 5/4$, amiből következik, hogy ebben az esetben NF aszimptotikusan 5/4-versenyképes.

Másrészt ennél nem kisebb az aszimptotikus hányadosa, amint ezt a következő példa mutatja. Legyen $\varepsilon > 0$ egy nagyon kicsi szám és legyen $I_n = (1/5, 1/5, 1/5, 1/5, \varepsilon)^{5n}$ (az inputban a fenti ötös lista ismétlődik $5n$-szer). Erre az inputra NF $5n$ ládát használ el, minden ismétlődést külön ládába rak. Az optimális megoldás megoldja az inputot $4n+1$ ládával, a $20n$ darab 1/5 kerül $4n$ ládába, és a kicsi tárgyak egy további ládába. Az $NF(I_n)/OPT(I_n)$ hányados tart 5/4-hez, ahogy n és ezzel együtt $OPT(I_n)$ tart a végtelenbe, így az állítás második felét is igazoltuk.

4.2. Feladat Legyen $L_1 = 1/3, 1/3, 1/2, 1/3+\varepsilon, 1/6-2\varepsilon, 1/3$ és $L_2 = 1/3, 1/3, 1/2, 1/3+\varepsilon, 1/6-2\varepsilon, 1/6+2\varepsilon, 1/6-\varepsilon$. Hajtsuk végre az NF, FF, BF ládapakolási algoritmusokat ezeken a listákon.

Megold: Az NF L_1-en és L_2-n is az első ládába rakja az első két elemet a másodikba a következő három elemet és a harmadikba a többit. FF az L_1-en, az első ládába rakja az első kettő és az ötödik elemet, a másodikba a harmadik és negyedik elemeket és egy harmadikba a hatodikat. FF az L_2-n, az első ládába rakja az első kettő, az ötödik és a hatodik elemet, a másodikba a harmadik és negyedik és hetedik elemeket így két ládát használ. BF az L_1-en, az első ládába rakja az első kettő és a hatodik elemet, a másodikba a harmadik a negyedik és ötödik elemeket így csak két ládát használ. Ezzel szemben az L_2-n első ládába rakja az első kettő és a hatodik elemet, a másodikba a harmadik a negyedik és ötödik elemeket így a hetedik elemhez egy harmadik ládát kell kinyitnia.

4.3. Feladat Legyen két sorozatunk L_1 tartalmazzon n db $1/3+\varepsilon$ méretű tárgyat L_2 tartalmazzon n darab $1/5+\varepsilon$ méretű tárgyat, ahol ε egy nagyon kicsi pozitív szám. Adjuk meg a lehetséges pakolási mintákat.‘

Megold: A lehetséges minták:

$(0,1),(0,2),(0,3),(0,4),(1,0),(1,1),(1,2),(1,3),(2,0),(2,1)$.

4.4. Feladat Legyen két sorozatunk L_1 tartalmazzon n db $1/3+\varepsilon$ méretű tárgyat L_2 tartalmazzon n darab $1/2+\varepsilon$ méretű tárgyat, ahol ε egy nagyon kicsi pozitív szám. Tegyük fel, hogy n páros.

- Adjuk meg a lehetséges pakolási mintákat.

- A mintákhoz rendelt változók alapján írjuk fel a ládapakolási feladatot egy egészértékű programozási feladatként.

- A változók alapján írjuk fel az IP módszernek megfelelően az alsó korlátot meghatározó egészértékű programozási feladatot.

Megold: A lehetséges minták: (1,0), (2,0), (0,1), (1,1) a változók legyenek x_1, x_2, x_3, x_4. Ekkor a feltételrendszer:

$x_1 + 2x_2 + x_4 = n$ (L_1 tárgyai)

$x_3 + x_4 = n$ (L_2 tárgyai)

a célfüggvény pedig az $x_1 + x_2 + x_3 + x_4$ érték (ládák száma) minimalizálása.

Amennyiben alsó korlátot számolunk, akkor meg kell határozni mind az online algoritmus mind pedig az optimális offline algoritmus költségét az L_1 és L_1L_2 listákon. Az L_1 listán az online algoritmus költsége $x_1 + x_2 + x_4$, ezen mintáknak megfelelő ládákat nyitja ki az algoritmus L_1 pakolása során. Az optimális költség pedig $n/2$, minden láda 2 elemet tartalmaz. Az L_1L_2 listán az algoritmus költsége az $x_1 + x_2 + x_3 + x_4$ érték, az optimális költség n, minden láda egy kisebb és egy nagyobb elemet tartalmaz. Tehát a fenti feltételrendszer mellé a célfüggvény

$$\min_x \max\{(x_1 + x_2 + x_4)/(n/2), (x_1 + x_2 + x_3 + x_4)/n\}.$$

4.5. Feladat Vegyük a sávpakolási feladatra a következő téglalapok sorozatát: $(1/2, 3/4)$, $(3/4, 1/4), (3/4, 5/8), (1/8, 3/16)$. Hajtsuk végre az inputon az $NFS_{1/2}$ algoritmust.

Megold: Az első tárgy 1 magasságú polcra kerül. Ekkor létrehozunk egy 1 magasságú polcot a sáv legalján, ez lesz az 1-magasságú aktív polc, és ennek a polcnak a bal sarkára helyezzük el a tárgyat. A második tárgy 1/4 magasságú polcra kerül. Mivel nincs ilyen aktív polc, ezért létrehozunk egy 1/4 magasságú polcot az előző 1 magasságú polc tetején, ez lesz az 1/4-magasságú aktív polc, és ennek a polcnak a bal sarkára helyezzük el a tárgyat. A harmadik tárgy ismét 1 magasságú polcra kerül. Mivel az 1 magasságú aktív polcon nem fér el, ezért azt lezárjuk, és létrehozunk egy új 1 magasságú polcot az előző 1/4 magasságú polc tetején, ez lesz az 1-magasságú aktív polc, és ennek a polcnak a bal sarkára helyezzük el a tárgyat. A negyedik tárgy 1/4 magasságú polcra kerül. Az 1/4 magasságú aktív polcon még elfér a tárgy, ezért arra polcra rakjuk annyira balra amennyire lehetséges a második tárgy mellé.

4.6. Feladat Vegyük a nyújtható sávpakolási feladatra a következő téglalapok sorozatát: $(1/2, 3/4), (3/4, 1/4), (3/4, 5/8), (1/8, 3/16)$ (ugyanaz, mint 4.5.-ben). Hajtsuk végre az inputon az $NFS_{1/2}$ algoritmust.

Megold: Az első tárgy 1 magasságú polcra kerül. Tehát az új mérete $(3/8, 1)$. Ekkor létrehozunk egy 1 magasságú polcot a sáv legalján, ez lesz az 1-magasságú aktív polc, és ennek a polcnak a bal sarkára helyezzük el a tárgyat. A második tárgy 1/4 magasságú polcra kerül, és nem változik a mérete. Mivel nincs ilyen aktív polc, ezért létrehozunk egy 1/4 magasságú polcot az előző 1 magasságú polc tetején, ez lesz az 1/4-magasságú aktív polc, és ennek a polcnak a bal sarkára helyezzük el a tárgyat. A harmadik tárgy ismét 1 magasságú polcra kerül, tehát az új mérete $(15/32, 1)$ lesz. Az 1 magasságú aktív polcon elfér, ezért oda rakjuk balra amennyire lehetséges. A negyedik tárgy 1/4 magasságú polcra kerül, ezért a méretét $(3/32, 1/4)$-re változtatjuk. Az 1/4 magasságú aktív polcon még elfér a tárgy, ezért arra polcra rakjuk annyira balra amennyire lehetséges a második tárgy mellé.

4.7. Feladat Határozzuk meg az offline sávpakolási probléma optimális megoldását, ha a tárgyak nyújthatóak.

Megold: Az optimális megoldás értéke nem lehet kisebb, mint a maximális magassága a tárgyaknak. Továbbá az optimális megoldás értéke nem lehet kisebb, mint az összterülete a tárgyaknak. Másrészt ha T ezen két érték maximuma ($T = \max\{\max_{i\in I}\{h_i\}, \sum_{i\in I} h_i w_i\}$) akkor az az algoritmus, ami minden tárgyat pontosan T magasságúra nyújt, egy T magasságú sávban el tudja helyezni az összes tárgyat.

4.8. Feladat Igazoljuk, hogy a DUPLÁZÓ algoritmus a nyújtható téglalapok mellett nem jobb, mint 4 versenyképes.

Megold: Vegyük a következő L_i listát. Az első i téglalap legyen $(1/4, 2^j)$, $j = 1, \ldots, i$, az utolsó pedig legyen $(1/4, 2^i+1)$. Végrehajtva az algoritmust, és megkonstruálva az optimális offline megoldást azt kapjuk, hogy $DS(L_i)/OPT(L_i) = 2^{i+2}/(2^i + 1)$, ami tart 4-hez, amint i tart a végtelenbe.

4.9. Feladat A ládapakolási feladat esetén egy tárgy méretéhez a következő súlyfüggvényt rendeljük:

$$w(x) = \begin{cases} 6x/5, & \text{ha } 0 \le x \le 1/6 \\ 9x/5 - 1/10, & \text{ha } 1/6 \le x \le 1/3 \\ 6x/5 + 1/10, & \text{ha } 1/3 \le x \le 1/2 \\ 6x/5 + 2/5, & \text{ha } 1/2 < x. \end{cases}$$

Igazoljuk, hogy ha van olyan tárgy egy ládában, amelynek a mérete nagyobb, mint 1/2 akkor a láda tartalmára teljesül a $\sum w(a_i) \le 1.7$ egyenlőtlenség.

Megold: A feladatot esetszétválasztással oldjuk meg. Ha nincs a tárgyak között 1/6 és 1/3 közötti elem, akkor minden szorzó a súlyfüggvényben 6/5. Másrészt egy 1/2-nél nagyobb elem van és így legfeljebb egy elemnek a mérete nagyobb 1/3-nál. Következésképp a tárgyaknál a súlyfüggvényben szereplő additív tag legfeljebb $2/5+1/10$, így az összsúly legfeljebb $6/5+2/5+1/10 = 1.7$. Ha a tárgyak között egyetlen 1/6 és 1/3 közötti elem, akkor legfeljebb egy 1/3 méretű tárgy szorzója 9/5 és minden más tárgynál a szorzó a súlyfüggvényben 6/5. Másrészt egy 1/2-nél nagyobb elem van és már nincs olyan elem,

amelynek a mérete nagyobb 1/3-nál. Következésképp a tárgyaknál a súlyfüggvényben szereplő additív tag legfeljebb $2/5 - 1/10$, így az összsúly legfeljebb $9/5 \cdot 1/3 + 6/5 \cdot 2/3 + 2/5 - 1/10 = 1.7$. Ha a tárgyak között kettő 1/6 és 1/3 közötti elem, akkor ezek összmérete legfeljebb 1/2, így legfeljebb 1/2 összméretű tárgy szorzója 9/5 és minden más tárgynál a szorzó a súlyfüggvényben 6/5. Másrészt egy 1/2-nél nagyobb elem van és már nincs olyan elem, amelynek a mérete nagyobb 1/3-nál. Következésképp a tárgyaknál a súlyfüggvényben szereplő additív tag legfeljebb $2/5 - 2/10$, így az összsúly legfeljebb $9/5 \cdot 1/2 + 6/5 \cdot 1/2 + 2/5 - 2/10 = 1.7$. Mivel van egy 1/2-nél nagyobb tárgy ezért három vagy több tárgy mérete nem lehet 1/6 és 1/3 között, így az állítást igazoltuk.

4.10. Feladat Igazoljuk, hogy a sávpakolási feladatban nincs olyan algoritmus, amelynek a versenyképességi hányadosa kisebb lenne, mint 3/2.

Megold: Vegyünk két tárgyat, mindkettő mérete legyen $(1/2, 1/2)$. Amennyiben az algoritmus úgy helyezi el őket, hogy legyen olyan vízszintes egyenes, amely mindkettőt metszi, akkor a sáv magasság legalább 1, az optimális sávmagasság 1/2, így az algoritmus nem lehet jobb 2-versenyképesnél. Ha az algoritmus úgy helyezi el őket, hogy ne legyen olyan vízszintes egyenes, amely mindkettőt metszi, akkor legyen a következő tárgy $1/2, 1$. Az első két elem elhelyezkedése alapján ebben az esetben az algoritmus költsége legalább 3/2, míg az optimális költség 1, így az állítást igazoltuk.

4.11. Feladat Egy nyújtható sávpakolási feladatról tudjuk, hogy minden tárgy magassága legfeljebb 1, szélessége legfeljebb 1/5, igazoljuk, hogy ekkor a NFS_r algoritmus aszimptotikus versenyképességi hányadosa legfeljebb 5/4.

Megold Az aktív polcok összmagasságára vonatkozó konstans korlát pontosan úgy adódik, mint abban az esetben ha nem korlátozzuk a tárgyak szélsségét. A lezárt polcokra vonatkozó korlát pedig a NF algoritmusra vonatkozó korlát közvetlen következménye. Ez így van ebben az esetben is, az eredeti bizonyításban a NF 2-versenyképességére vonatkozó indoklást kell kicserélni a speciális esetre vonatkozó gondolatmenettel.

4.12. Feladat Vegyük azt az ütemezési feladatot, ahol a tárgyak nem csak egy gépet követelnek meg a végrehajtásukhoz, hanem a j-edik tárgy végrehajtásához k_j darab egymás melletti gépet kell felhasználni.

- Adjunk 2-versenyképes algoritmust arra az esetre, ha minden munka végrehajtási ideje 1.

- Adjunk egy konstans versenyképes algoritmust az általános esetre.

Megold: Vegyük észre, hogy ha minden munka végrehajtási ideje 1, akkor tekinthetjük az időegységeket ládáknak a munkákat pedig olyan tárgyaknak, amiknek a mérete k_j. Tehát a feladat megfogalmazható ládapakolási feladatként, amiből adódik 2-versenyképes algoritmus (NF). Az általános esetben kezeljük a feladatokat téglalapként, aminek a szélessége k_j hossza p_j, és használjunk egy sávpakolási polc (NFS vagy FFS) algoritmust. Mivel minden k_j egész, ezért a kapott megoldás valóban egy jó ütemezés lesz.