További feladatok

1. Feladat A nyugtázási feladatban a csomagok a $0, 1/2, 3/5, 1, 4/3$ időpontokban érkeznek. Adjuk meg az Ébreszt algoritmus viselkedését.

Megoldás Az első csomag a 0 időpontban érkezik, azaz $a_1 = 0$, ekkor ÉBRESZT beállítja az ébresztőt az $e_1 = 1$ értékkel. A következő csomag érkezési ideje $a_2 = 1/2$, ekkor még nem járt le az ébresztő, így nem küldtünk nyugtát, tehát újra állítjuk az ébresztőt az $e_2 = (1 - 1/2)/2 = 1/4$ értékkel az $1/2 + 1/4$ időpontra. A következő csomag érkezési ideje $a_3 = 5/8$. Ekkor még nem járt le az ébresztő és nem küldtünk nyugtát, így újra állítjuk az ébresztőt az $e_3 = (1 - 5/8 - 1/8)/3 = 1/12$ értékkel az $5/8 + 1/12$ időpontra. A következő csomag érkezési ideje $a_4 = 1$, mivel az $5/8 + 1/12$ időpontig nem érkezett újabb csomag, ezért abban időpontban nyugtát küldtünk, és most az ébresztőt az $e_4 = 1$ értékkel a 2 időpontra állítjuk be. A következő csomag érkezési ideje $a_5 = 4/3$, mivel nem járt le az ébresztő, beállítjuk az új értékre $e_5 = 1/3$ az ébresztő időpontja $5/3$.

2. Feladat Vegyük a nyugtázási feladatnak azt a változatát, ahol a célfüggvény

$$k + \sum_{j=1}^{k} \nu_j,$$

ahol k a nyugták száma és $\nu_j = \max_{t_{j-1} < a_i \leq t_j}(t_j - a_i)$, a j-edik nyugta által összegyűjtött maximális késedelem. Igazoljuk, hogy van olyan optimális megoldás, amelyben ha $t_i - t_{i-1} \geq 1$, akkor t_{i-1}-et már t_i előtt lenyugtázzuk.

Megoldás Vegyünk egy tetszőleges optimális megoldást, ha ez a megoldás t_{i-1}-t nyugtázza t_i előtt, akkor az állítás nyilvánvalóan igaz. Egyébként módosítsuk úgy, hogy egy további nyugtát hozzáveszünk a t_{i-1} időpontban. Vizsgáljuk meg a költség változásait. Egyrészt a nyugtázási költség növekszik 1-el, másrészt a késedelmi költség csökken $t_i - t_{i-1}$-el. Következésképpen az így kapott megoldás is optimális.

3. feladat Vegyük a nyugtázási feladatnak azt a változatát, ahol a célfüggvény

$$k + \sum_{j=1}^{k} \nu_j,$$

ahol k a nyugták száma és $\nu_j = \max_{t_{j-1} < a_i \leq t_j}(t_j - a_i)$, a j-edik nyugta által összegyűjtött maximális késedelem. Igazoljuk, hogy nincs olyan algoritmus, amelynek a versenyképességi hányadosa kisebb, mint 2.

Megoldás Vegyük észre, hogy a teljes késedelmet használó célfüggvény esetén adott alsó korlát bizonyításban, az online algoritmus mindig egy csomagot nyugtáz, az offline pedig 2-t, de ebből az egyiknek nincs késedelme. Következésképpen az alsó korlát igazolásakor használt inputon az ott vizsgált algoritmusokra

a két célfüggvény ugyanazt az értéket adja. Tehát a teljes késedelmre vonatkozó bizonyítás igazolja ezt az állítást is.

4. feladat Tegyük fel, hogy $k = 10$ és az adott pillanatban a Haziur algoritmus HA memóriája három lapot tartalmaz: g_1-re $s(g_1) = 2, cr(g_1) = 1$, g_2-re $s(g_2) = 4, cr(g_2) = 3$, g_3-re $s(g_3) = 3, cr(g_3) = 3$. Legyen a következő letöltendő lap g_4, amelyre $s(g_4) = 4, c(g_4) = 4$. adjuk meg mit tesz a Háziúr algoritmus a lap letöltése során.

Megoldás Ez a lap nem fér el a HA memóriában, ki kell tennünk lapokat. A HÁZIÚR algoritmus által számolt érték $\Delta = 1/2$ és a megváltoztatott cr értékek: $cr(g_1) = 0, cr(g_2) = 1, cr(g_3) = 3/2$, így g_1-et eltávolítjuk a HA memóriából. Ekkor a g_4 lap még mindig nem fér el a HA memóriában. Az új Δ érték $\Delta = 1/4$ és a megváltoztatott cr értékek: $cr(g_2) = 0, cr(g_3) = 3/4$, így a g_2 lapot is eltávolítjuk a HA memóriából. Ekkor már van hely g_4 számára, elhelyezzük a memóriában a $cr(g_4) = 4$ értékkel.

5. feladat Legyen a kezdeti lista x_1, x_2, x_3, x_4, x_5 hajtsuk végre az MTF algoritmust a következő inputra x_3, x_5, x_3, x_2.

Megoldás A kapott listák rendre:
x_3, x_1, x_2, x_4, x_5,
x_5, x_3, x_1, x_2, x_4,
x_3, x_5, x_1, x_2, x_4,
x_2, x_3, x_5, x_1, x_4

6. feladat Igazoljuk, hogy a listakarbantartási feladatra Transponse algoritmus nem konstans versenyképes.

Megoldás Legyen a lista $x_1, \ldots, x_n$ és vegyük a következő kéréssorozatot $(x_n, x_{n-1})^M$. Ekkor Transponse mindig a kért utolsó elemet felcseréli az előtte álló elemmel, így a teljes költsége $2M \cdot n$. Másrészt az MTF algoritmusnak, amely mindig legelőre hozza az aktuális elemet a költsége az első két kéréstől eltekintve mindig 2, így összesen $2n+2(M-1)2$. Következésképpen $Transpose(I)/OPT(I) \geq 2M \cdot n/(2n + 4(M - 1) \to n/2$ ha M tart végtelenbe.

7. feladat Igazoljuk, hogy a listakarbantartási feladatra az FC algoritmus nem konstans versenyképes.

Megoldás Legyen a kezdeti lista $x_1, \ldots, x_n$, a kéréssorozat pedig legyen n-szer x_1, $n-1$-szer x_2, és így tovább $n+1-i$-szer x_i. Ekkor FC sosem változtat a listán és a kiszolgálás teljes költsége

$$\sum_{i=1}^{n}(n+1-i)i = (n+1)\sum_{i=1}^{n} i - \sum_{i=1}^{n} i^2 = (n+1)^2 n/2 - n(n+1)(2n+1)/6 = \Theta(n^3).$$

Ezzel szemben MTF költsége $\sum_{i=1}^{n} i + \sum_{i=1}^{n-1} i = \Theta(n^2)$. Következésképpen $FC(I)/OPT(I) \to \infty$, ha n tart végtelenbe.

8. feladat Igazoljuk, hogy a lista karbantartási feladatnál van olyan input, ahol az optimális offline algoritmusnak használnia kell fizetett cserét.

Megoldás Legyen a lista x_1, x_2, x_3 a kéréssorozat x_3, x_2, x_3, x_2. Esetszétválasztással igazolható, hogy amennyiben nem használ egy algoritmus fizetett cseréket a költsége legalább 9. (Az első kérés költsége 3, ezt követően megvizsgálva, hogy hova rakja az algoritmus az x_3 elemet igazolható az állítás.) Ezzel szemben ha két fizetett cserével az utolsó helyre visszük x_1-et, akkor a cserék költsége 2 és utána nem mozdítva több elemet a kérések költsége 6. Következésképpen fizetett cserével valóban jobb megoldást kaphatunk.