**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.

**2. 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, így 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.

**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 van olyan optimális megoldás, amelyben ha $a_i - a_{i-1} \geq 1$, akkor $a_{i-1}$-et már $a_i$ előtt lenyugtázzuk.

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

**4. 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.

**5 Feladat.** Hajtsuk végre az optimális megoldást megadó dinamikus programozási algoritmust a következő inputon: $0, 1/10, 2/10, 3/10, 4/10, 6/10, 8/10, 9/10$.

*Megoldás* A dinamikus programozás által kitöltött a késedelmeket számító táblázat (az előadásban megadott képletek alapján) a következő. Csak az első 3 sort töltjük ki, mert van olyan megoldás, amyel költsége kisebb, mint 4.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1/10 | 3/10 | 6/10 | 1 | 2 | 32/10 | 39/10 |
| 0 | 0 | 1/10 | 2/10 | 4/10 | 8/10 | 12/10 | 14/10 |
| 0 | 0 | 0 | 1/10 | 2/10 | 4/10 | 6/10 | 8/10 |

A segédtáblázat azt tartalmazza hova kell rakni az utolsó előtti nyugtát (a képletben szereplő minimum helye).

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 2 | 2 | 4 | 4 | 5 |
| 0 | 1 | 2 | 3 | 3 | 4 | 5 | 5 |

Az első táblázat utolsó oszlopban levő elemeihez kell hozzáadni azt, hogy hányadik sorban vagyunk (nyugták száma), így kapjuk meg a teljes költséget. Így az optimális megoldás az, ha két nyugtát használunk és a költség $2 + 14/10$. A megoldás a segédtábla második sorából kiolvasható: az első nyugtát az 5-dik, a másodikat a 8-dik csomagnál küldjük.

**6. Feladat** Vegyük a nyugtázási feladatnak azt a változatát, ahol a célfüggvény

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

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. Adjuk meg az optimális megoldást meghatározó dinamikus programozási eljárást.

*Megoldás* Az $I = a_1, \ldots, a_n$ inputra jelölje $F(r, k)$ az első $r$ kérésnek $k$ nyugtával való nyugtázása során elérhető minimális késedelem értékét. Az $F(n, k)$ értékek alapján könnyen megkapható az optimális költség: venni kell a $k + F(n, k)$ értékek minimumát, ahol $1 \geq k \geq n$ egész. Az optimális megoldás pedig mint dinamikus programozásnál általában visszafejtéssel vagy a feljegyzéses módszerrel megkapható. Tehát célunk az $F(r, k)$ értékek kiszámítására egy dinamikus programozási algoritmust megadni. Ehhez kezdeti értékeket kell megadnunk és egy rekurzív összefüggést.

Ha $k = 1$, akkor az egyetlen nyugtát az utolsó csomag érkezésekor kell küldeni. Tehát

$$F(r, 1) = a_r - a_1.$$

Amennyiben $k > 1$, akkor az utolsó nyugtát az $a_r$ időpontban küldjük. Az utolsó előtti, (k-1)-dik nyugtát küldhetjük az $a_{k-1}, a_k, \ldots, a_{r-1}$ időpontokban. Ha valamely $a_q$ időpontban küldjük, akkor az első $k-1$ nyugta minimális késedelme a definíció alapján $F(q, k-1)$ a k-adik nyugta késedelme pedig $a_r - a_{q+1}$. Következésképpen, ha $k > 1$, akkor

$$F(r, k) = \min_{k-1 \leq q \leq r-1} \{\max\{F(q, k-1), a_r - a_{q+1}\}\}.$$

A kezdeti feltételek és a rekurzív összefüggés alapján soronkénti táblázatkitöltéssel megkaphatóak a $F(r, k)$ értékek.

**7. 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. Adjunk meg egy 1-versenyképes algoritmust, abban az esetben, ha az algoritmus mindig látja a következő csomag érkezési idejét.

*Megoldás* A 3-as feladatban igazoltuk, hogy az optimális megoldás, akkor küld nyugtát egy $a_i$ időpontban, ha $a_{i+1} - a_i \geq 1$. Tehát az az algoritmus, amely az $a_i$ időpontban akkor és csak akkor küld nyugtát, ha $a_{i+1} - a_i \geq 1$ optimális megoldást ad, azaz 1-versenyképes. Másrészt ennek megállapításához elegendő ismerni a következő csomag érkezési idejét.

**8. 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. Adjunk meg egy 2-c-versenyképes algoritmust, abban az esetben, ha az algoritmus mindig látja a következő $c < 1$ hosszú intervallumban a csomagok érkezési idejét.

*Megoldás* Legyen TLA az az algoritmus, amely akkor küld nyugtát egy $a_i$ kérést követően, ha az előrenéző intervallum alapján látja, hogy $a_{i+1} - a_i \geq 1$. Vegyünk egy tetszőleges $I$ inputot és osszuk fázisokra. Legyen $S_1 = \{a_1, \ldots, a_{k(1)}\}$, ahol $k(1)$ az első olyan index, amelyre $a_{k(1)+1} - a_{k(1)} \geq 1$. A többi fázis hasonlóan definiálható $S_{j+1} = \{a_{k(j)+1}, \ldots, a_{k(j+1)}\}$, ahol $k(j+1)$ az első olyan index $k(j)$ után, amelyre $a_{k(j+1)+1} - a_{k(j+1)} \geq 1$. Az utolsó fázist az utolsó munka zárja. Ekkor az optimális offline algoritmus, minden fázis utolsó csomagjánál nyugtát küld. Ezzel szemben TLA a fázis utolsó csomagja után $1 - c$ egységgel észleli, hogy a fázis véget ért, így, akkor küld nyugtát. Tehát egy $L$ hosszú fázisra vonatkozóan az optimális költség $L+1$, TLA költsége $L + 1 - c + 1$. Következésképpen a két megoldás hányadosa $(L + 2 - c)/(L + 1) \leq (2 - c)/1$.

**9 Feladat.** Hajtsuk végre a LIP algoritmust a következő inputon: $0, 1/10, 2/10, 3/10, 7/10, 8/10, 11/10, 12/10, 13/10, 23/10, 24/10$, ahol az algoritmus előrelátása $c = 12/10$

*Megoldás* Az első blokk a $0, 12/10$ időintervallum. Könnyen látható, hogy ebben az optimális megoldás három nyugtát küld, egyet a $3/10$ időpontban, a másikat a $8/10$ időpontban, a harmadikat a $12/10$ időpontban. A második blokk a $13/10$ időpontban kezdődik és ott véget is ér (utána egy 1 hosszú csomagmentes szakasz jön), így egy nyugtát küld a $13/10$ időpontban. Az utolsó blokk a $23/10$ időponttól tart, az algoritmus egy nyugtát küld a $24/10$ időpontban.

**10 Feladat.** Igazoljuk, hogy nincs 1-versenyképes algoritmus a nyugtázási feladatra abban a modellben, ahol az algoritmus a következő 1 hosszú intervallumban látja előre a csomagok érkezési idejét.

*Megoldás* Vegyük a következő inputkezdeményt $0, 3/4$. Ha az algoritmus a 0 időpontban nyugtát küld, akkor a sorozat nem tartalmaz több elemet, így a költsége legalább 2, az optimális költség pedig $7/4$ (egy nyugta a $3/4$ időpontban). Így ilyen algoritmus nem lehet 1-versenyképes. Ha az algoritmus nem küld nyugtát a 0 időpontban, akkor jön még egy csomag az $5/4$ időpontban. Az optimális megoldás nyugtát küld a 0 és $5/4$ időpontokban, a költsége $5/2$. Tetszőleges online algoritmusnak nyugtát kell küldenie $5/4$

után. Ha más nyugtát nem küld, akkor a késedelme legalább $5/4 + 1/2$, így a költsége legalább $11/4$. Ha még egy nyugtát küld, akkor azt nem küldheti az $1/4$ időpont előtt, így a késedelme vagy legalább az $1/4 + 1/2$ (ha külön nyugtázzza az első csomagot) vagy legalább $3/4$, ha az első két csomagot nyugtázza együtt. A költsége mindeképpen legalább $11/4$, így nem lehet 1-versenyképes.

**11. Feladat** Tegyük fel, hogy $k = 10$ és az adott pillanatban a $HA$ memória 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 a HÁZIÚR algoritmus viselkedését.

*Megoldás* Ekkor ez a lap nem fér el a $HA$ memóriában, ki kell tennünk lapokat. A HÁZIÚR algoritmus által számolt érték $\Delta = 1/2$ (a cr/s értékek minimuma) és a megváltoztatott $cr$ értékek ($cr - s \cdot \Delta$): $cr(g_1) = 0, cr(g_2) = 1, cr(g_3) = 3/2$, így $g_1$-et eltávolítjuk a $HA$ memóriából. Ekkor a $g_4$ lap még mindig nem fér el a $HA$ memóriában. Az új $\Delta$ 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.

**12. Feladat** Tegyük fel, hogy $k = 13$ és az adott pillanatban a $HA$ memória három lapot tartalmaz: $g_1$-re $s(g_1) = 2, cr(g_1) = 1$, $g_2$-re $s(g_2) = 4, cr(g_2) = 2$, $g_3$-re $s(g_3) = 6, cr(g_3) = 3$. Legyen a következő letöltendő lap $g_4$, amelyre $s(g_4) = 4, c(g_4) = 4$. Adjuk meg a HÁZIÚR algoritmus viselkedését.

*Megoldás* Ekkor 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: $cr(g_1) = 0, cr(g_2) = 0, cr(g_3) = 0$, így mindhárom lapot eltávolítjuk a $HA$ memóriából, majd berakjuk $g_4$-et beállítva a $cr(g_4) = 4$ értéket.

**13. Feladat** Tegyük fel, hogy minden lapra $s(p) = c(p) = 1$. Adjuk meg a HÁZIÚR algoritmus viselkedését.

*Megoldás* Ekkor minden $p$ lapra a bekerüléskor $cr(p) = 1$. Másrészt, ha valamely lapok kreditjét csökkenteni kell, akkor minden lapra a $cr/s$ arány 1, így a minden lapra a creditérték 0 lesz a csökkentés után. Következésképpen az algoritmus egyenként rakja a lapokat a memóriába, amíg meg nem tölti (k

darab lap lesz benne), majd ezt követően, ha tele van és olyan lapot kérnek, ami nincs benne, akkor kidobja az összes lapot a memóriából.

**14. Feladat** Vegyük a torlódásszabályozás problémájára a bináris keresés algoritmusát. Igazoljuk, hogy a keresési költségek összegének nagyságrendje $\Theta(N^2 \log N)$.

*Megoldás* Tegyük fel, hogy a sávszélességre adott N felső korlát 2 hatvány. Ekkor a bináris keresés algoritmusa $\log_2 N$ lépésben kitalálja a számot. Ha a keresését egy bináris keresőfával ábrázoljuk, akkor a keresőfa $i$-edik szintjén a lehetséges próbák $(2k+1)N/2^i$, ahol $k = 0, 1, \ldots, 2^i - 1$.

Vizsgáljuk meg egy adott k-ra a $(2k+1)N/2^i$ próbához tartozó költségeket. Az alatta levő értékek, ahol a túllépés miatt a teljes sávszélesség elveszik $(2k)N/2^i, (2k)N/2^i + 1, (2k)N/2^i + 2, \ldots, (2k+1)N/2^i - 1$. Ezen lehetőségekre a teljes költség ezen számok összege, ami több, mint $2k(N/2^i)(N/2^i)$. Következésképpen a teljes költség a keresőfa $i$-edik szintjén legalább

$$\sum_{k=0}^{2^i-1} 2k(N/2^i)(N/2^i) \approx N^2/4.$$

Következésképpen az algoritmus teljes költsége legalább $(\log_2 N)N^2/4$. Másrészt a maximális keresési költség $\Theta(N \log N)$, így adódik, hogy a teljes keresési költség $O(N^2 \log N)$.

**15. feladat** Vegyük a torlódásszabályozás problémájára a TCP algoritmusát. Igazoljuk, hogy a keresési költségek összege $\Theta(N^3)$.

*Megoldás* Tegyük fel, hogy a sávszélességre adott N felső korlát páros. Vizsgáljuk az $N/2 + i$ $(0 \leq i \leq N/2)$ alakú számok költségét. Az első próba (N/2) költsége $i$ ( a nem kihasznált sávszélesség), majd ezt követően egyenként növeljük a próba értékét, így a költségek $i-1, i-2, \ldots, 1$. Tehát az elem megtalálásának költsége $\sum_{j=1}^{i} j = i(i+1)/2$. Következésképpen ezen elemek kiszámolásának teljes költsége:

$$\sum_{i=1}^{N/2-1} i(i+1)/2 \geq 1/2 \sum_{i=1}^{N/2-1} i^2 = (N/2-1)(N/2)(N-1)/6 = \Omega(N^3).$$

Másrészt az előadáson láttuk, hogy a maximális költség, amely egy elem kiszámításához szükséges a TCP algoritmus esetén $\Theta(N^2)$, így a teljes költség nyilvánvalóan $O(N^3)$.

**16. feladat** Tekintsük azt a hálózatot, amely 4 pontból, $(A, B, C, D)$ és négy élből $(A, B), (B, D), (A, C), (C, D)$, áll, ahol az élek maximális sávszélességére $u(A, B) = 1$, $u(B, D) = 3/2$, $u(A, C) = 2$,$u(C, D) = 3/2$. Tegyük fel, hogy $\mu = 10$ és a $j$-edik kérés előtt az eddig lefoglalt sávszélességek $3/4$ az $A, B, D$ úton, $5/4$ az $A, C, D$ úton, $1/2$ a $(B, D)$ élen, $1/2$ az $(A, C)$ élen. Legyen a következő kérés $1/8$ sávszélesség lefoglalása $A$ és $D$ között, adjuk meg az exponenciális algoritmus milyen profitok esetén fogadja el a kérést.

*Megoldás* Ekkor az $l_e(j)$ értékek: $l_{(A,B)}(j) = (3/4) : 1 = 3/4$, $l_{(B,D)}(j) = (3/4 + 1/2) : (3/2) = 5/6$, $l_{(A,C)}(j) = (5/4 + 1/2) : 2 = 7/8$, $l_{(C,D)}(j) = (5/4) : (3/2) = 5/6$.

A két lehetséges útra a $C(P)$ értékek

$$C(A, B, D) = 1/8 \cdot 10^{3/4} + 1/12 \cdot 10^{5/6} = 1.265.$$

$$C(A, C, D) = 1/16 \cdot 10^{7/8} + 1/12 \cdot 10^{5/6} = 1.032.$$

A minimális érték az 1.032. Tehát azt kell megvizsgálni, hogy $2mb_j = 8b_j$ érték nagyobb -e, mint 1.032. Következésképpen ha a profit legalább $1.032/8 = 1.29$, akkor a kérést az algoritmus elfogadja.

**17. Feladat** Igazoljuk, hogy amennyiben a nyereségmaximalizáló forgalomirányítási modellben az $r(j)/u(e)$ hányadosokra semmiféle kikötést nem teszünk, akkor nem adható meg konstans-versenyképes algoritmus.

*Megoldás* Vegyünk egy egyetlen élből álló hálózatot, ahol a sávszélesség 1. Legyen az első kérés $1 - \varepsilon$ sávszélesség a profitja 1. Ha az algoritmus nem fogadja el a sorozat véget ér és az algoritmus nem versenyképes. Ha elfogadja, akkor sok csomag érkezik $2\varepsilon$ sávszélességgel, és 1 profittal, amelyek közül már egyet sme tud elfogadni. Az optimum az első csomagot visszautasítja, és a többieket fogadja el, így az algoritmus ez esetben sem lehet konstans versenyképes.