O, Ω, Θ, mintaillesztés

1. Ugyanarra a feladatra van két algoritmusunk A és B. A maximális lépésszámot leíró függvényeket jelölje f_A és f_B. Tudjuk, hogy $f_A(n) \in O(f_B(n))$. Következik-e ebből, hogy

   a) A minden bemeneten gyorsabb, mint B?

   b) A véges sok bemenet kivételével gyorsabb, mint B?

   c) A megfelelően nagy bemenetekre gyorsabb, mint B?

   Megoldás: Mindegyikre *nem* a válasz, pl. $f_A(n) = 2n$ és $f_B(n) = n$ esetén igaz, hogy $f_A(n) \in O(f_B(n))$, de $f_A(n) > f_B(n)$ minden n-re.

2. Az alábbi függvények közül melyikre igaz, hogy $O(n^2)$ és melyikre, hogy $\Omega(n^2)$?
   $f_1(n) = 11n^2 + 100000$ $\quad f_2(n) = 8n^2 \log_2 n$ $\quad f_3(n) = 1{,}5n + 3\sqrt{n}$

   Megoldás: $f_1(n) \le 12n^2$, ha $n \ge 1000$, ezért $f_1 \in O(n^2)$ ($c = 12$, $n_0 = 1000$)
   (vagy pl. $f_1(n) \le 100011n^2$ és ezért $c = 100011$, $n_0 = 1$)
   Másrészt nyilván $f_1(n) \ge 11n^2$, tehát $f_1(n) \in \Omega(n^2)$ (és így $f_1(n) \in \Theta(n^2)$ is teljesül.)

   $f_2(n) \notin O(n^2)$, mert ha $f_2(n) \le cn^2$ valamely c konstansra, akkor $8 \log n \le c$, ami csak $n \le 2^{c/8}$ esetén teljesül, nem minden nagy n-re.
   Nyilván $f_2(n) \ge 8n^2$, azaz $f_2 \in \Omega(n^2)$.

   $f_3(n) \le 1{,}5n + 3n = 4{,}5n$ mindig teljesül ha $n \ge 1$, ezért ez $O(n) \subset O(n^2)$. Másrészt, ha $f_3(n) \ge cn^2$ valamely c konstansra, akkor $cn^2 \le f_3(n) \le 4{,}5n$ is igaz és ezért $n \le 4{,}5/c$, tehát az egyenlőtlenség nem teljesülhet ha n nagy, $f_3 \notin \Omega(n^2)$.

3. Mely $a, b > 1$ egész számokra teljesülnek az alábbiak?
   $n^a \in O(n^b)$ $\quad 2^{an} \in O(2^{bn})$ $\quad \log_a n \in O(\log_b n)$

   Megoldás: Az első esetben az kell, hogy $n^a \le cn^b$, azaz $n^{a-b} \le c$ teljesüljön valamilyen $c > 0$ konstanssal, ha $n \ge n_0$. Az $a = b$ esetben ez nyilván teljesül, $n^a \in O(n^a)$ $\quad (c = 1,\ n_0 = 1)$.
   Az $a < b$ esetben a kitevő negatív, ezért $n^{a-b} \le 1$, tehát $n_0 = 1$ választással már $c = 1$ esetén is teljesül.
   Ha viszont $a > b$, akkor a kitevő pozitív, a függvény monoton nő, végtelenhez tart, tehát nem létezhet ilyen c konstans.

   A másodiknál, az előzőhöz hasonlóan $2^{an-bn} \le c$ kell, ami $a \le b$ esetén teljesül például $c = 1$, $n_0 = 1$ választással, de ha $a > b$, akkor nincs ilyen c konstans.

   A harmadik esetben használjuk fel, hogy $\log_a n = \dfrac{\log_b n}{\log_b a}$, tehát $c = \dfrac{1}{\log_b a} > 0$ jó. ($n_0 = 1$)

4. Az alábbi függvényeket rendezze nagyságrend szerint nem csökkenő sorozatba: ha f_i után közvetlenül f_j következik a sorban, akkor $f_i(n) \in O(f_j(n))$ teljesüljön!
   $f_1(n) = 8n^3$ $\quad f_2(n) = 5\sqrt{n} + 1000n$ $\quad f_3(n) = 2^{(\log_2 n)^2}$ $\quad f_4(n) = 1514n^2 \log_2 n$

   Megoldás: Megadunk egy sorrendet és megindokoljuk, hogy jó: f_2, f_4, f_1, f_3
   $f_2(n) \le 1005n \le 1005n^2 \log n \le f_4(n)$, $c = 1, n_0 = 1$ jó.
   $f_4(n) \le 1514n^3 = 1514/8 f_1(n)$, $c = 1514/8, n_0 = 1$ jó.
   Vegyük észre, hogy $f_3(n) = (2^{\log n})^{\log n} = n^{\log n} \ge n^3$, ha $\log n \ge 3$, így $c = 8$, $n_0 = 2^3 = 8$ jó.

5. Adjon O becslést a következő függvényekre:
   $(n^2 + 8)(n + 1)$ $\quad (n \log n + n^2)(n^3 + 2)$ $\quad (n! + 2^n)(n^3 + \log(n^2 + 1))$ $\quad (2^n + n^2)(n^3 + 3^n)$

   Megoldás: Sokféle jó megoldás van, általában az O-ba valami egyszerű, de minél kisebb függvényt akarunk tenni, most úgy csináljuk, hogy számolni se nagyon kelljen. Ehhez az előforduló összegek nagyságrendjére adunk becslést, és ezeket szorozzuk össze.
   $(n^2 + 8)(n + 1) \le 2n^2 \cdot 2n \in O(n^3)$ (a becslés minden $n \ge 1$ esetén igaz).
   $(n \log n + n^2)(n^3 + 2) \le 2n^2 \cdot 2n^3 \in O(n^5)$ (a becslés minden $n > 1$ esetén igaz).

$(n!+2^n)(n^3+\log(n^2+1)) \le 2n!\cdot 2n^3 \in O(n^3 n!)$ (a becslés minden $n \ge 4$ esetén igaz, amikortól már $n! > 2^n$).
$(2^n+n^2)(n^3+3^n) \le 2\cdot 2^n \cdot 2\cdot 3^n$ (a becslés minden $n \ge 4$ esetén igaz, mert innen $2^n \ge n^2$, és már $3^n > n^3$ is teljesül).

Minden eset O helyett Θ-val is helyes, ehhez az alsó becsléseket kell hasonló módon meggondolni.

6. Tekintsük az $f_1(n) = 1{,}5\,n!$ és $f_2(n) = 200\,(n-1)!$ függvényeket. Melyik igaz és melyik nem az alábbiak közül?
$f_1 \in O(f_2)$ $\quad f_2 \in O(f_1)$ $\quad f_1 \in \Omega(f_2)$ $\quad f_2 \in \Omega(f_1)$ $\quad f_1 \in \Theta(f_2)$ $\quad f_2 \in \Theta(f_1)$

Megoldás: Vegyük észre, hogy $f_1(n) = \dfrac{1{,}5n}{200} f_2(n) = cn \cdot f_2(n)$. Ebből látszik, hogy $f_1 \notin O(f_2)$, $f_2 \in O(f_1)$, $f_1 \in \Omega(f_2)$, $f_2 \notin \Omega(f_1)$, tehát $f_1 \notin \Theta(f_2)$ és $f_2 \notin \Theta(f_1)$.

7. Jelölje egy algoritmus maximális lépésszámát az n méretű bemeneteken $L(n)$. Adjunk felső becslést az $L(n)$ nagyságrendjére, ha tudjuk, hogy $L(1) = 2$ és $n > 1$ esetén

(a) $L(n) = L(n-1) + 3$
(b) $L(n) = L(n-1) + 5$
(c) $L(n) = L(n-1) + 3n$
(d) $L(n) = 2L(n-1) + 3$
(e) $L(n) = L(\lceil n/2 \rceil) + 3$
(f) $L(n) = L(\lceil n/2 \rceil) + n^k$
(g) $L(n) = 2L(\lceil n/2 \rceil) + 3$
(h) $L(n) = 4L(\lceil n/2 \rceil) + 3$

Az (e)-(h) esetben elegendő 2 hatványra meggondolni.

Mi változik, ha egyenlőség helyett $\le$ vagy $\ge$ áll?
És ha O helyett Θ a feladat?

Megoldás:
(a) $L(n) = L(n-1)+3 = (L(n-2)+3)+3$. Ezt tovább folytatva a $0 \le i \le n-1$ esetben azt kapjuk, hogy $L(n) = L(n-i)+3i$. Alkalmazzuk ezt az $i = n-1$ válazstással: $L(n) = L(1)+3(n-1) = 3n-1 \in O(n)$, sőt $\Theta(n)$ is igaz.
(b) $L(n) = L(n-1)+5 = (L(n-2)+5)+5 = L(n-i)+5i = L(1)+5(n-1) = 5n-3 \in O(n)$, sőt $\Theta(n)$ is igaz.
(c) $L(n) = L(n-1)+3n \le (L(n-2)+3(n-1))+3n \le L(n-2)+6n \le L(n-i)+3in \le L(1)+3(n-1)n \in O(n^2)$ (Ebből a felső becslésből nem következik, de ha pontosabban számolunk, $\Theta(n^2)$ is kijön.)
(d) $L(n) = 2L(n-1)+3 = 2(2L(n-2)+3)+3 = 2^2L(n-2)+2\cdot 3+3 = 2^iL(n-i)+3(2^{i-1}+\cdots+2+1) = 2^{n-1}L(1)+3(2^{n-1}-1) = \dfrac{5}{2}2^n - 3 \in \Theta(2^n)$

(e)–(h) részeket nézzük a 2 hatvány esetekre (azaz az egész részt mindig elhagyhatjuk).
(e) $L(n) = L(n/2)+3 = L(n/4)+3+3 = L(n/2^i)+3i$. Az $i = \log n$ értékre kapjuk, hogy $L(n) = L(1)+3\log n$, tehát $L(n) \in O(\log n)$ (Az is igaz, hogy $L(n) \in \Theta(\log n)$.)
(f) $L(n) = L(n/2)+n^k < L(n/4)+2n^k < L(n/2^i)+in^k$. Most is $i = \log n$ értékig kell mennünk, ekkor $L(n) \le L(1)+n^k\log n \in O(n^k \log n)$. (Ez most valóban csak egy felső becslés, pontosabban számolva kiderül, hogy $\Theta(n^k)$. Miért?)
(g) $L(n) = 2L(n/2)+3 = 2^2L(n/4)+3(2+1) = 2^iL(n/2^i)+3(2^{i-1}+\cdots+1)$ Most $i = \log n$ értékig kell mennünk, erre $L(n) = 2^{\log n}L(1)+3(2^{\log n}-1) = nL(1)+3n-3 \in \Theta(n)$.
(h) $L(n) = 4L(n/2)+3 = 4^2L(n/4)+3(4+1) = 4^iL(n/2^i)+3(4^{i-1}+\cdots+1)$, ami $i = \log n$ értéknél $L(n) = 4^{\log n}L(1)+3(4^{\log n}-1)/3 = n^2(L(1)+1)-1 \in \Theta(n^2)$.

Amennyiben $=$ helyett $\le$ áll, akkor a felső becsléseink továbbra is igazak, tehát az O eredmények helyesek (de a Θ nem, mert pl az $L(n) = 2$ konstans függvényre teljesülnek a feltételek).

A $\ge$ esetben viszont nem marad érvényben az O, hiszen az egy felső becslés, és tetszőleges „elég nagy” függvény teljesíti a feltételeket, pl. az $L(n) = a2^n$ a feltételtől függő a-val. Ebben az esetben csak legfeljebb alsó becslést (Ω) lehet bizonyítani.

8. Az egyszerű algoritmussal, illetve a gyorskereséssel állapítsa meg, hogy az $S = ABBABACABCBAC$ szövegben az $M = ABABC$ minta hányszor fordul elő! Hány összehasonlítást használtak az algoritmusok?

Megoldás: Az egyszerű algoritmus minden lehetséges illesztésnél az első hibáig megy (vagy amíg a minta végére nem ér), ez $3+1+1+4+1+2+1+3+1 = 17$ összehasonlítás (és 0-szor fordul elő a minta).

A gyorskereséshez kell az ugrófüggvény, aminek értékei: $U[A] = 3$, $U[B] = 2$, $U[C] = 1$. 0 eltolásnál most is 3 összehasonlítás történik. Utána $U[S[6]] = U[A] = 3$-mal toljuk el, itt 4 összehasonlítás lesz, majd $U[S[9]] = U[B] = 2$ jön, ahol 2 összehasonlítás lesz, azután $U[S[11]] = U[B] = 2$ ahol 3 az összehasonlítások száma, és $U[S[12]] = 1$ következik 1 összehasonlítással, azaz összesen $3 + 4 + 2 + 3 + 1 = 13$.

9. Álljon a minta és a szöveg is csupa 0-ból, a minta hossza m, a szövegé pedig $n \geq m$. Hány összehasonlítást végez

   a) az egyszerű algoritmus, ha csak a minta első előfordulását keressük?

   b) az egyszerű algoritmus, ha a minta összes előfordulását keressük?

   c) a gyorskeresés, ha csak a minta első előfordulását keressük?

   d) a gyorskeresés, ha a minta összes előfordulását keressük?

   Megoldás: (a) m, mert az első m karakter összevetése után leállunk.
   (b) $n - m + 1$ lehetséges eltolás van, ezek mindegyikénél m összehasonlításra kerül sor, azaz összesen $m(n - m + 1)$ összehasonlítás lesz.
   (c) m, mert az első m karakter összevetése után leállunk.
   (d) minden eltolásnál m összehasonlítás van. Mivel a minta utolsó karaktere (is) 0, az ugrás mindig 1 lesz, azaz mind az $n - m + 1$ eltolásra sor kerül, összesen tehát $m(n - m + 1)$ összehasonlítás lesz.

10. Az $n > 2$ hosszú csupa 0-ból álló szöveghez adjon meg olyan m hosszú mintát, melyen az egyszerű algoritmus m-től függetlenül $O(n)$ összehasonlítást használ!

    Megoldás: Mivel $n-m+1$ eltolás lehetséges, és ez $O(n)$, ezért elég arról gondoskodni, hogy minden esetben a minta hosszától függetlenül konstans sok összehasonlításra kerüljön sor. Például, ha a minta 1-gyel kezdődik, akkor az jó, hiszen minden eltolásnál csak 1 összehasonlítás, azaz összesen $n - m + 1 \in O(n)$ összehasonlítás lesz.

    Vagy például ha 001-gyel kezdődik a minta, akkor annak hosszától függetlenül mindig 3 összehasonlítás, összesen $3(n - m + 1) \in O(n)$ történik.

    (A gyorskeresés jó esetben még ennél is gyorsabb tud lenni, például a csupa 1 minta esetén mindig $(m+1)$-et ugrik, azaz $n - m + 1$ helyett az eltolások száma kevesebb, mint n/m, ami persze továbbra is $O(n)$.)

11. Igazolja, hogy az egyszerű algoritmus várható futási ideje $O(n)$, ha a szöveg és a minta is véletlen 0/1 sorozat (a bitek egymástól függetlenek, mindegyik 1/2 - 1/2 valószínűséggel 0 vagy 1).
    Mi a helyzet, ha csak a minta véletlen?

    Megoldás: Jelölje t_i azt a valószínűségi változót, amelynek értéke az i-vel való eltoláskor történő összehasonlítások száma. Ekkor az összehasonlítások száma összesen $\sum_{i=0}^{n-m} t_i$. Ennek a várható értéke $E(\sum t_i) = \sum(E(t_i))$. Még azt kell meghatározni, mennyi az $E(t_i)$. Tetszőleges szöveg esetén 1/2 valószínűséggel 1 összehasonlítás kell (a minta első karaktere eltér a szövegétől), 1/4 valószínűséggel 2 (az első karakter megegyezett, de a második eltért, stb. Ezek alapján $E(t_i) = \sum_{i=1}^{\infty} i2^{-i} = 2$ Tehát az összehasonlítások várható értéke $E(\sum t_i) = \sum(E(t_i)) = 2(n - m + 1) \in O(n)$.

    A fenti gondolatmenet akkor is működik, ha csak a minta véletlen.

12. Az A algoritmus 0/1 sorozatok mintaillesztési feladatát oldja meg, m bites minta és $n > m$ bites szöveg esetén $T(n, m) \geq n$ lépésben megadja a minta összes előfordulását (növekvő sorrendben).

    Hogyan lehet ennek segítségével egy tetszőleges (legalább 2 elemű, de nem feltétlenül konstans méretű) Σ ábécé feletti szöveg–minta párra $O((n + T(n, m)) \log |\Sigma|)$ időben megtalálni egy m hosszú minta összes előfordulását egy n hosszú szövegben?

    Megoldás: Az ábécé betűit elkódolhatjuk binárisan, de ekkor több gond is van: vigyázni kell, hogy a bináris sorozatoknál csak az olyan illesztés szabályos, ami egy betű első bitjével kezdődik, és ezzel a szöveg és a minta hossza is megnőtt, ha $\alpha = \lceil \log |\Sigma| \rceil$, akkor mindkettő α-szorosára, azaz a lépésszám $T(\alpha n, \alpha m)$-mel lesz csak becsülhető.

Hogy a szöveg és a minta hossza ne legyen nagyobb, csináljuk a következőt: a betűket binárisan kódoljuk, és az eredeti feladatot szétbontjuk α darab mintaillesztési feladatra, az ℓ-edik feladatban a szövegbeli betűk ℓ-edik bitjeit tekintjük, és ugyanígy a minta betűinek ℓ-edik bitjeit.

Vegyük észre, hogy az eredeti feladatnál pontosan akkor szerepel a minta k eltolással, ha mind az α feladatban van illeszkedés a k eltolásnál. Tehát ennek az α darab feladatnak a közös megoldásait keressük. Amire egy megoldás lehet, hogy ezekre párhuzamosan futtatjuk az algoritmust, és ha a k eltolásnál mindegyik találatot jelez, akkor van egy megoldásunk.

Párhuzamos futtatás helyett ügyes könyveléssel is megoldhatjuk, hogy a közös megoldás észrevétele is beleférjen az adott időbe: felveszünk egy n méretű B tömböt, aminek minden eleme kezdetben 0. Futtatjuk az A algoritmust az $\ell = 1, 2, \ldots, \alpha$ bináris feladaton. Ha az ℓ-edikben k eltolásra illeszkedést találunk, akkor eggyel megnöveljük $B[k]$ értékét. A végén megkeressük, hogy a B tömbben hol szerepel α, azaz melyik eltolás szerepelt mindenhol.

A lépésszám $O(\alpha T(n, m) + n) = O((n + T(n, m))\alpha)$. Felhasználva, hogy $\log |\Sigma| \leq \alpha \leq 2 \log |\Sigma|$, kapjuk a kívánt lépésszámot.