1. Bizonyítsa be, hogy ha f_1,f_2,g_1,g_2 pozitív függvények és $f_1(n) \in O(g_1(n))$ és $f_2(n) \in O(g_2(n))$, akkor

   (a) $f_1(n) + f_2(n) \in O(\max(g_1(n),g_2(n)))$,

   (b) $f_1(n)f_2(n) \in O(g_1(n)g_2(n))$.

   Megoldás: $\exists c_1,n_1$, hogy $f_1(n) \leq c_1 g_1(n)$, ha $n \geq n_1$,
   $\exists c_2,n_2$, hogy $f_2(n) \leq c_2 g_2(n)$, ha $n \geq n_2$,
   ekkor
   $f_1(n) + f_2(n) \leq \max(c_1,c_2)(g_1(n) + g_2(n)) \leq 2\max(c_1,c_2)\max(g_1(n),g_2(n))$, ha $n \geq \max(n_1,n_2)$,
   és
   $f_1(n)f_2(n) \leq c_1c_2g_1(n)g_2(n)$, ha $n \geq \max(n_1,n_2)$.

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

   Megoldás: $f_1(n) \leq 12n^2$, ha $n \geq 1000$, ezért $f_1 \in O(n^2)$ ($c = 12$, $n_0 = 1000$)
   (vagy pl. $f_1(n) \leq 100011n^2$ és ezért $c = 100011$, $n_0 = 1$)

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

   $f_3(n) \leq 1{,}5n + 3n = 4{,}5n$ mindig teljesül ha $n \geq 1$, ezért ez $O(n) \subset O(n^2)$.

3. Jelölje egy algoritmus maximális lépésszámát az n méretű bemeneteken $L(n)$. Tegyük fel, hogy az algoritmus szerkezetéből következik, hogy $L(n) \leq 2\,L(\frac{n}{2}) + n$ ha $n \geq 2$ és $L(1) = 1$. (Tegyük fel, hogy $n = 2^k$, azaz n 2-hatvány.)

   Lássa be, hogy ekkor $L(n) \in O(n \log n)$. Igaz-e, hogy $L(n) \in O(n^2)$?

   Megoldás: Belátjuk, hogy $c = 2, n_0 = 2$ jó választás az ordó definíciójához.

   Kifejtve a rekurziót:

$$L(n) \leq 2 \cdot L(\frac{n}{2}) + n \leq 2\Big(2L(\frac{n}{4}) + \frac{n}{2}\Big) + n = 4L(\frac{n}{4}) + n + n \leq$$

$$\leq 4\Big(2L(\frac{n}{8}) + \frac{n}{4}\Big) + n + n = 8L(\frac{n}{8}) + n + n + n \leq \ldots \leq 2^k L(1) + n + \ldots + n = 2^k + n \cdot k$$

   Az utolsó lépésben azt használjuk, hogy k tag van, mindegyik n és $L(1) = 1$.

   Mivel $n = 2^k$ és ezért $k = \log n$:

$$2^k + n \cdot k = n + n\log n \leq n\log n + n\log n = 2n\log n$$

   Tehát $c = 2$ jó konstans és a fenti lépések minden $n \geq 2$ esetben fennáltak, ezért $n_0 = 2$.

   Mivel $O(n \log n)$-ből következik $O(n^2)$ is, ezért $L(n) \in O(n^2)$ is igaz.

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
   $f_2(n) \leq 1005n \leq 1005n^2 \log n \leq f_4(n)$, $c = 1, n_0 = 1$ jó.
   $f_4(n) \leq 1514n^3 = \frac{1514}{8} f_1(n)$, $c = \frac{1514}{8}, n_0 = 1$ jó.
   Vegyük észre, hogy $f_3(n) = (2^{\log n})^{\log n} = n^{\log n} \geq n^3$, ha $\log n \geq 3$, így $c = 8$, $n_0 = 2^3 = 8$ jó.

5.

Az alábbi pszeudokódban egy $*$ kiírása számít lépésnek. Mutassa meg, hogy a kód lépésszáma $O(n^3)$.

```
for i = 0 to n-1:
    for j = i+1 to n:
        print  j  darab *
```

Megoldás: A belső ciklus ciklusmagja legfeljebb n lépés és ez a ciklusmag legfeljebb n-szer fut le, ezért a belső ciklus lépésszáma $O(n^2)$. A külső ciklus, ami maga a kód, n-szer fut le, ennek magja $O(n^2)$-es (az előbb vizsgált belső ciklus), így a kód lépésszáma $n \cdot O(n^2) = O(n^3)$.

6. 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).

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?

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 $i = n-1$ választással: $L(n) = L(1) + 3(n-1) = 3n - 1 \in O(n)$
(b) $L(n) = L(n-1) + 5 = (L(n-2)+5) + 5 = L(n-i) + 5i = L(1) + 5(n-1) = 5n - 3 \in O(n)$
(c) $L(n) = L(n-1) + 3n = (L(n-2) + 3(n-1)) + 3n \le L(n-2) + 6n \le L(n-i) + 3in \le L(1) + 3(n-1)n \in O(n^2)$
(d) $L(n) = 2L(n-1)+3 = 2(2L(n-2)+3)+3 = 2^2L(n-2)+2\cdot 3+3 = 2^iL(n-i)+3(2^{i-1}+\cdots+2+1) = 2^{n-1}L(1) + 3(2^{n-1}-1) = \dfrac{5}{2}2^n - 3 \in O(2^n)$
(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 $L(n) \in O(\log n)$
(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 $L(n) \le L(1) + n^k \log n \in O(n^k \log n)$.
(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 $L(n) = 2^{\log n}L(1) + 3(2^{\log n} - 1) = nL(1) + 3n - 3 \in O(n)$
(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 $L(n) = 4^{\log n}L(1) + 3(4^{\log n} - 1)/3 = n^2(L(1)+1) - 1 \in O(n^2)$.

Amennyiben $=$ helyett $\le$ áll, akkor a felső becsléseink továbbra is igazak, tehát az O eredmények helyesek.

A $\geq$ 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 lehet bizonyítani.

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

   Megoldás: Az első esetben az kell, hogy $n^a \leq cn^b$, azaz $n^{a-b} \leq c$ teljesüljön valamilyen $c > 0$ konstanssal, ha $n \geq n_0$. 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 $n^{a-b} \leq 1$, 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 $2^{an-bn} \leq c$ kell, ami $a \leq b$ 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, akármi is a és b. $(n_0 = 1)$

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

   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 $f_2 \notin \Theta(f_1)$.

10. 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 $3+4+2+3+1 = 13$.

11. 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 $O(n)$.)

12. 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 $\frac{1}{2}$ – $\frac{1}{2}$ 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 $\frac{1}{2}$ valószínűséggel 1 összehasonlítás kell (a minta első karaktere eltér a szövegétől), $\frac{1}{4}$ valószínűséggel 2 (az első karakter megegyezett, de a második eltért), stb. Másrészről t_i nem lehet nagyobb a minta hosszánál. Ezek alapján $E(t_i) \leq \sum_{i=1}^{\infty} i2^{-i} = 2$ Tehát az összehasonlítások várható értéke $E(\sum t_i) = \sum(E(t_i)) \leq 2(n-m+1) \in O(n)$.

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