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

Megoldás: $f_1(n) \leq 12n^2$, ha $n \geq 1000$, ezért $f_1 \in O(n^2)$ ($c = 12$, $n_0 = 1000$)
Megj: Igazából az $n_0 = \sqrt{100000} \approx 316{,}24$ is jó választás, de nem feltétlenül célunk a legkisebb n_0 kiválasztása.
(másik indoklás pl. $f_1(n) \leq 100011n^2$ ha $n \geq 1$ 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 $O(n) \subset O(n^2)$.

2. 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 $n \cdot O(n^2) = O(n^3)$.

3. (a) Lássa be, hogy a buborékrendezés lépésszáma $O(n^2)$ (ezt a rendező algoritmust Prog1-ből már tanulták, de itt láthatják a pszeudokódját ismét). Releváns lépésnek az összehasonlítás és a csere számít. Igaz-e, hogy az algoritmus lépésszáma $O(n^3)$?
(b) Lássa be, hogy a buborékrendezés lépésszáma $\Omega(n^2)$.
(c) Igaz-e, hogy a buborékrendezés lépésszáma $\Theta(n^2)$?

```
ciklus i = n-tól 2-ig:
    ciklus j = 1-tól (i-1)-ig:
        ha   A[j] > A[j+1]:
            csere A[j] és A[j+1]
    ciklus vége
ciklus vége
```

Megoldás: (a) A belső ciklus magja konstans lépés és a belső ciklus legfeljebb n-szer fut le, mert $i - 1 \leq n - 1$, ezért a belső ciklus lépésszáma $O(n)$. A külső ciklus legfeljebb n-szer fut le, ennek a magja az $O(n)$ lépésszámú belső ciklus, így a külső ciklus (és így a teljes kód) $O(n^2)$.
Ha egy függvény $O(n^2)$ nagyságrendű, akkor $O(n^3)$ is igaz a definíció alapján.
(b) Ha azt az inputot tekintjük, amiben a számok csökkenően vannak, akkor a futás során $n - 1 + n - 2 + \ldots + 2 + 1 = \dfrac{n(n-1)}{2}$ összehasonlítás van és ugyanennyi csere, azaz a teljes lépésszám $n(n-1) \geq n \cdot \dfrac{n}{2} = 0.5n^2$, amennyiben $n \geq 2$.)
(c) Mivel $O(n^2)$ és $\Omega(n^2)$ is igaz, ezért $\Theta(n^2)$ 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!
$f_1(n) = 8n^3$ $f_2(n) = 5\sqrt{n} + 1000n$ $f_3(n) = 2^{(\log_2 n)^2}$ $f_4(n) = 1514n^2 \log_2 n$

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

5. Az $A[1 : n]$ tömb (bináris keresést használó) beszúrásos rendezése $n - 1$ körből áll: az i. körben ($1 \leq i \leq n-1$) a már rendezett $A[1 : i]$ tömbben megkeressük az $A[i+1]$ elem helyét bináris kereséssel, majd az $A[i+1]$ elemet szomszédos elemek cseréjével balra mozgatjuk addig, amíg a megtalált pozícióba nem érkezik. (a) Lássa be, hogy az algoritmus lépésszáma $O(n^2)$, releváns lépésnek az összehasonlítás és a csere számít. (b) Lássa be, hogy az algoritmus lépésszáma $\Omega(n^2)$ és $\Theta(n^2)$ is.

Megoldás: (a) Legfeljebb n kör van, mindegyik körben egy legfeljebb n méretű tömbben csinálunk bináris keresést és a keresés után legfeljebb n cserével az aktuális elem a helyére kerül. Ez azt jelenti,

hogy egy kör lépésszáma $O(\log n+n) = O(n)$ és mivel legfeljebb n kör van, ezért a teljes futás lépésszáma $n \cdot O(n) = O(n^2)$.

(b) Ha a tömb csökkenően rendezett, akkor a cserék száma $1 + 2 + \ldots + n - 2 + n - 1 = \frac{n(n-1)}{2} \geq n \cdot \frac{n}{4} = 0.25n^2$, ha $n \geq 2 = n_0$. Mivel $O(n^2)$ és $\Omega(n^2)$ is igaz, azért $\theta(n^2)$ is igaz, definíció szerint.

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?
$f_1 \in O(f_2)$ $f_2 \in O(f_1)$ $f_1 \in \Omega(f_2)$ $f_2 \in \Omega(f_1)$ $f_1 \in \Theta(f_2)$ $f_2 \in \Theta(f_1)$

Megoldás: Vegyük észre, hogy $f_1(n) = \frac{1{,}5n}{200} f_2(n) = cn \cdot f_2(n)$. 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. Adjon O becslést a következő függvényekre:
$(n^2+8)(n+1)$ $(n\log n + n^2)(n^3+2)$ $(n! + 2^n)(n^3 + \log(n^2+1))$ $(2^n + n^2)(n^3 + 3^n)$

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.
$(n^2+8)(n+1) \leq 2n^2 \cdot 2n \in O(n^3)$ (a becslés minden $n \geq 3$ esetén igaz).
$(n\log n + n^2)(n^3+2) \leq 2n^2 \cdot 2n^3 \in O(n^5)$ (a becslés minden $n > 1$ esetén igaz).
$(n! + 2^n)(n^3 + \log(n^2+1)) \leq 2n! \cdot 2n^3 \in O(n^3 n!)$ (a becslés minden $n \geq 4$ esetén igaz, amikortól már $n! > 2^n$).
$(2^n + n^2)(n^3 + 3^n) \leq 2 \cdot 2^n \cdot 2 \cdot 3^n$ (a becslés minden $n \geq 4$ esetén igaz, mert innen $2^n \geq n^2$, és már $3^n > n^3$ is teljesül).

---

8. 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.

9. Bizonyítsa be, hogy ha f_1, f_2, g_1, g_2 pozitív értékkészletű 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) $f_1(n) \cdot f_2(n) \in O(g_1(n) \cdot g_2(n))$.

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
$f_1(n) f_2(n) \leq c_1 c_2 g_1(n) g_2(n)$, ha $n \geq \max(n_1,n_2)$.

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

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, $n^a \in O(n^a)$ $(c = 1,\ n_0 = 1)$.
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. ($n_0 = 1$)

11. Tegyük fel, hogy n 2-hatvány. Bizonyítsuk be, hogy ahhoz, hogy n különböző számból a két legnagyobb elemet kiválasszuk, $n + \log_2 n - 2$ összehasonlítás elégséges.

    Megoldás: Kieséses versennyel ($n-1$ összehasonlítás) határozzuk meg a legnagyobb elemet. Utána a második azon $\log_2 n$ közül kerül ki, akik csak a győztestől kaptak ki. Ez további $\log_2 n - 1$ összehasonlítással megkereshető.

12. Az $A[1:n]$ tömb piros és zöld elemeket tartalmaz. Szeretnénk átrendezni úgy, hogy az egyszínű elemek folytonosan helyezkedjenek el (elöl az összes piros, utána a zöldek vagy fordítva). Egy megengedett *lépés* két szomszédos tömbelem cseréje. Javasoljunk konstans szorzó erejéig optimális lépésszámú algoritmust.

    Megoldás: Világos, hogy csak különböző színű elemeket érdemes cserélni. Ha a kezdetben az első és az utolsó $n/4$ elem piros, akkor akármelyik végére is akarjuk rendezni a pirosakat, valamelyik $n/4$ elem helyet kell cseréljen az összes zölddel, tehát szükséges lehet $n^2/8$ lépés. $O(n^2)$ lépéssel meg meg is lehet csinálni, hisz tetszőleges sorrendet el lehet érni n^2-nél kevesebb szomszéd cseréjével például buborékrendezéssel (piros $<$ zöld, ha a pirosakat akarjuk a tömb elejére tenni).

13. Adott n chip, melyek képesek egymás tesztelésére a következő módon: ha összekapcsolunk két chipet, mindkét chip nyilatkozik a másikról, hogy hibásnak találta-e. Egy hibátlan chip korrektül felismeri, hogy a másik hibás-e, míg egy hibás chip akármilyen választ adhat. Tegyük fel, hogy a chipek több, mint a fele korrekt. Adjunk algoritmust, mely n-nél kevesebb fenti tesztet használva kikeres egy jó chipet.

    Megoldás:

1. chip	2. chip	1. kimenet	2. kimenet
jó	jó	jó	jó
jó	rossz	rossz	jó/rossz
rossz	jó	jó/rossz	rossz
rossz	rossz	jó/rossz	jó/rossz

    Készítsünk elő 3 dobozt: T=tesztelt, N=nem tesztelt, S=szemét. Kezdetben minden chip az N dobozban van. Vegyünk ki kettőt belőle. Ha egy pár jó-jót mond, akkor egyformák, vagy jó-jó, vagy rossz-rossz. Ha a pár jó-jót mond, akkor tegyük mindkettőt a T dobozba. Ha nem jó-jót mondanak, akkor legalább az egyik rossz, ilyenkor dobjuk ki mindkettőt az S dobozba. Ha az N és T doboz sem üres még, akkor mindig a T dobozból egyet párosítunk egy N dobozban levővel. Ha jó-jó, akkor a T dobozba tesszük mindkettőt, különben az S dobozba. Ha kiürül a T doboz, akkor két N dobozbelit veszünk megint, ha van benne még kettő.

    A tesztek előtt is és minden teszt után is fennáll a következő két tulajdonság (ún. invariáns tulajdonságok ezek):

    (1) $T \cup N$-ben többen vannak a jó chipek, mint a rosszak.

    (2) A T-ben mindig ugyanolyan chipek vannak (vagy csak jók, vagy csak rosszak, de lehet üres is).

    Az (1) miatt tesztek után $T \cup N$ nem lehet üres.

    Addig tesztelünk, ameddig van elvégezhető teszt. Mi lehet a végén, amikor már nem tudunk további tesztet elvégezni? Ekkor N-ben 0 vagy 1 elem van. Ha 0, akkor (1) és (2) miatt T nem üres és csupa jó chipből áll.

    Ha N-ben csak egy elem van, akkor T üres és (1) miatt N-ben csak jó chip lehet. Mindkét esetben igaz, hogy a végén $T \cup N$-ben csupa jó chip van.