# Cvičení 9

**Příklad 1:** Podrobně zdůvodněte, proč pro funkce

$$f(n) = 5n^3 + 2n^2 - 9n + 13 \qquad g(n) = n^3$$

platí $f \in O(g)$ a $f \in \Omega(g)$, a proč pro ně neplatí $f \in o(g)$ a $f \in \omega(g)$.
Z předchozího pak vyvoďte, že tedy také platí $g \in O(f)$, $g \in \Omega(f)$, $f \in \Theta(g)$, $g \in \Theta(f)$, $g \notin o(f)$ a $g \notin \omega(f)$.

**Příklad 2:** Následující funkce seřaďte podle rychlosti jejich růstu, tj. seřaďte je do posloupnosti $g_1, g_2, \ldots, g_{15}$, kde $g_1 \in O(g_2)$, $g_2 \in O(g_3)$, $\ldots$, $g_{14} \in O(g_{15})$. Uveďte také, pro které dvojice funkcí $g_i$ a $g_{i+1}$ v této posloupnosti platí $g_i \in \Theta(g_{i+1})$ a pro které $g_i \in o(g_{i+1})$.

| | | | | |
|---|---|---|---|---|
| $n^2$ | $2^n$ | $n$ | $\log_2 n$ | $n^n$ |
| $n!$ | $\log_2(n^2)$ | $(\log_2 n)^2$ | $n^3$ | $\sqrt{n}$ |
| $2^{2^n}$ | $10^n$ | $n^{1000}$ | $\sqrt[3]{n}$ | $n \log_2 n$ |

**Příklad 3:** Pro následující trojice funkcí $f_1, f_2, f_3$ určete, které vztahy tvaru $f_i \in O(f_j)$, $f_i \in \Omega(f_j)$, $f_i \in \Theta(f_j)$, $f_i \in o(f_j)$ a $f_i \in \omega(f_j)$ platí a které ne.

a) $f_1(n) = 3n^2 + 5n - 1$, $f_2(n) = 2n^3 - 15n - 183$, $f_3(n) = (n+1)(n-1)$
b) $f_1(n) = 4n^2 + n^2 \log_2 n$, $f_2(n) = \log_2^5 n$, $f_3(n) = 17n + 3$
c) $f_1(n) = n\sqrt[5]{n}$, $f_2(n) = n$, $f_3(n) = \sqrt{n}$
d) $f_1(n) = 2^n$, $f_2(n) = n^{1024}$, $f_3(n) = n!$
e) $f_1(n) = 2^n$, $f_2(n) = n^n$, $f_3(n) = n!$
f) $f_1(n) = 2^n$, $f_2(n) = n^n$, $f_3(n) = n^{\log_2 n}$
g) $f_1(n) = 10^n$, $f_2(n) = 2^n$, $f_3(n) = 2^{2^n}$
h) $f_1(n) = \log_{10}(n^2)$, $f_2(n) = \log_2 n$, $f_3(n) = \log_2(n^2)$
i) $f_1(n) = n + \sqrt{n} \cdot \log_2 n$, $f_2(n) = n \cdot \log_2 n$, $f_3(n) = \sqrt{n} \cdot \log_2^2 n$
j) $f_1(n) = 2^n$, $f_2(n) = 2^{\sqrt{n}}$, $f_3(n) = n!$
k) $f_1(n) = n/2048$, $f_2(n) = \sqrt{n} \cdot 3n$, $f_3(n) = n + n \cdot \log_2 n$
l) $f_1(n) = (\log_2 n)^n$, $f_2(n) = n^n$, $f_3(n) = 10^{\sqrt{n}}$

**Příklad 4:** Určete co nejpřesněji časovou a paměťovou složitost Algoritmu 1.
Předpokládejte, že hodnota $n$ udává počet prvků v poli $A$ a že toto pole je indexováno od nuly.

**Příklad 5:** Určete co nejpřesněji časovou a paměťovou složitost Algoritmu 2 (připomeňte si tento algoritmus z minulého cvičení).
(Předpokládejte, že hodnota $n$ udává počet prvků v poli $A$, že toto pole je indexováno od nuly, a že $x$ je hodnota hledaného prvku.)

**Algoritmus 1:** Třídění přímým výběrem

```
SELECTION-SORT (A, n):
    i := n − 1
    while i > 0 do
        k := 0
        for j := 1 to i do
            if A[k] < A[j] then
                k := j
        x := A[k];  A[k] := A[i];  A[i] := x
        i := i − 1
```

**Algoritmus 2:** Binární vyhledávání

```
BSEARCH (x, A, n):
    ℓ := 0
    r := n
    while ℓ < r do
        k := ⌊(ℓ + r) / 2⌋
        if A[k] < x then
            ℓ := k + 1
        else
            r := k
    if ℓ < n and A[ℓ] = x then
        return ℓ
    return NOTFOUND
```

**Příklad 6:** Popište pseudokódem libovolný algoritmus pro řešení následujícího problému a určete co nejpřesněji jeho časovou a paměťovou složitost. (Co je vhodné v případě tohoto problému považovat za velikost vstupu?)

VSTUP: Matice $A, B$, jejichž prvky jsou celá čísla.

VÝSTUP: Matice $A \cdot B$.

*Poznámka:* V algoritmu se nemusíte zabývat načítáním vstupu a výpisem výstupu.

Nepředpokládejte, že matice $A$ a $B$ musí být čtvercové, můžete však předpokládat, že jejich rozměry jsou takové, aby se obě matice daly vynásobit, tj. že velikost matice $A$ je $m \times n$ a velikost matice $B$ je $n \times p$, kde $m$, $n$ a $p$ jsou nějaká přirozená čísla.

**Příklad 7:** Navrhněte (nějaký) algoritmus řešící následující problém:

VSTUP: Číslo $n$ a sekvence čísel $a_1, a_2, \ldots, a_n$, kde pro všechna $i = 1, 2, \ldots, n$ platí $a_i \in \{1, 2, \ldots, n\}$.

OTÁZKA: Je v sekvenci $a_1, a_2, \ldots, a_n$ každé $x \in \{1, 2, \ldots, n\}$ obsaženo právě jednou?

Analyzujte časovou složitost vašeho algoritmu. Pokud je větší než $O(n)$, zkuste navrhnout algoritmus s časovou složitostí $O(n)$.