

## 8. cvičení z Programování

**Definice 1.** Říkáme, že  $f$  je  $O(g)$  když  $\exists n_0, c$  takové, že  $\forall n > n_0 : f(n) \leq c \cdot g(n)$ .

Tuto skutečnost vyjadřujeme  $f = O(g)$  či  $f \in O(g)$ . Naopak nepoužíváme  $O(g) = f$  (notace není symetrická).

**Příklad 2.** Rozhodněte, jestli jsou následující tvrzení pravdivá:

$$n \log n = O(n^2) \quad n \log n = O(n^{1.01}) \quad f(n) = O(f^2(n))$$

**Řešení 2:**

- $n \log n = O(n^2)$  – **platí**, neboť evidentně  $\log n = O(n)$ .
- $n \log n = O(n^{1.01})$  – také **platí**. Funkce  $\log n$  je  $O(n^{0.01})$ , můžeme se například podívat, jak se chová podíl obou funkcí, když  $n$  jde k nekonečnu:

$$\lim_{n \rightarrow +\infty} \frac{\log n}{n^{0.01}}.$$

Pokud dojdeme k tomu, že tato limita konverguje, bude to implikovat, že nějaký násobek funkce ve jmenovateli má od nějakého  $n_0$  všude větší hodnoty než funkce v čitateli. Hodnotu této limity můžeme vypočítat např. l'Hospitalovým pravidlem:

$$\lim_{n \rightarrow +\infty} \frac{\log n}{n^{0.01}} = \lim_{n \rightarrow +\infty} \frac{(\log n)'}{(n^{0.01})'} = \lim_{n \rightarrow +\infty} \frac{n^{-1}}{0.01 \cdot n^{-0.99}} = \frac{1}{0.01} \lim_{n \rightarrow +\infty} n^{-0.01} = 0$$

Jiný způsob jak vztah ze zadání nahlédnout, je podívat se na inverzní funkce k těm zadaným (exponenciála roste od nějakého místa rychleji než libovolný polynom – i přesto, že uvažovaný polynom má stupeň 100).

- $f(n) = O(f^2(n))$  – **neplatí** obecně, nicméně platí pro funkce, které je možné (od nějakého  $n_0$ ) odhadnout ze spodu nenulovou konstantou. Při analýze algoritmů nás zajímají téměř výhradně takové funkce.

**Příklad 3.** Předpokládejte, že následující funkce vyjadřují počet kroků potřebných k provedení nějakého algoritmu. Jaká je jeho časová složitost (odhadněte danou funkci co nejtěsněji pomocí  $O(g)$ )?

$$\begin{aligned} f_1(n) &= 5 + 0.001n^3 + 0.025n && \in O(n^3) \\ f_2(n) &= \log_{20}n + \log_3n && \in O(\log n) \\ f_3(n) &= n^2 \log_2n + n(\log_2n)^2 && \in O(n^2 \log n) \end{aligned}$$

Příklady z tohoto cvika:

**Příklad 1.** Navrhněte algoritmus, který zjistí (ideálně v lineárním čase a konstantním prostoru), jestli je spojový seznam zacyklený.

Toto jsme si víceméně nechali jako jeden z úkolů za body.

**Příklad 2.** Dostali jste posloupnost závorek různých druhů. Například:

$$\{ () () \} \langle \langle \rangle \rangle \{ \langle \rangle \}$$

V paměti může být taková posloupnost reprezentována posloupností celých čísel, přičemž číslo  $i$  značí otevírací závorku  $i$ -tého typu a  $-i$  značí uzavírací závorku  $i$ -tého typu. Navrhněte algoritmus, který ověří, že je tato posloupnost dobře uzávorkována.

**Řešení 2:** Požili jsme zásobník. Ukázali jsme si, jak ho naprogramovat pomocí pole a pomocí spojáku.

**Příklad 3.** Navrhněte algoritmus na nalezení nejkratší cesty ven z bludiště.

**Řešení 3:** Použili jsme frontu. Opět jde implementovat pomocí pole či spojáku. V druhém zmiňovaném případě je vhodné udržovat si ukazatel na konec fronty, abychom do ní mohli přidávat v konstantním čase.

**Příklad 4.** Navrhněte algoritmus na nalezení největší nezávislé množiny v binárním stromu.