Kapitola 1

Úvod

1.1 Základní pojmy

1.1.1 Algoritmus. *Algoritmem* rozumíme dobře definovaný proces, tj. posloupnost výpočetních kroků, který přijímá hodnoty (zadání, vstup) a vytváří hodnoty (řešení, výstup).

1.1.2 Problém, úloha. *Úloha*, též *problém*, je obecná specifikace vztahu zadání/řešení. *Instancí* problému, úlohy $\mathcal{U}$ rozumíme konkrétní zadání všech parametrů, které daná úloha (problém) obsahuje. Jinými slovy, instance úlohy je správný příklad zadání.

1.1.3 Řekneme, že algoritmus $\mathcal{A}$ *řeší* úlohu $\mathcal{U}$, jestliže pro každý vstup (každou instanci problému $\mathcal{U}$) vydá správné řešení.

Poznamenejme, že předchozí věta znamená, že každý algoritmus, který řeší nějakou úlohu, se vždy zastaví. To znamená, že algoritmus, který se na nějakém vstupu nezastaví, nemůže řešit žádnou úlohu.

1.1.4 Analýza časové složitosti algoritmu. Existují dva základní způsoby měření časové náročnosti algoritmů.

1. Analýza nejhoršího případu. Jedná se o asymptotický odhad $T(n)$ času potřebného pro vyřešení každé instance velikosti n.
2. Průměrná složitost. Jedná se o asymptotický odhad $T_{aver}(n)$ průměrného času, který je potřeba pro vyřešení instance velikosti n.

1.2 Asymptotický růst funkcí

1.2.1 Symbol $\mathcal{O}$. Je dána nezáporná funkce $g(n)$. Řekneme, že nezáporná funkce $f(n)$ *je* $\mathcal{O}(g(n))$, jestliže existuje kladná konstanta c a přirozené číslo n_0 tak, že

$$f(n) \leq c\, g(n) \quad \text{pro všechny } n \geq n_0.$$

$\mathcal{O}(g(n))$ můžeme též chápat jako třídu všech nezáporných funkcí $f(n)$:

$$\mathcal{O}(g(n)) = \{f(n) \mid \exists c > 0, n_0 \text{ tak, že } f(n) \leq c\, g(n) \;\; \forall n \geq n_0\}.$$

1.2.2 Symbol Ω. Je dána nezáporná funkce $g(n)$. Řekneme, že nezáporná funkce $f(n)$ *je* $\Omega(g(n))$, jestliže existuje kladná konstanta c a přirozené číslo n_0 tak, že

$$f(n) \geq c\, g(n) \quad \text{pro všechny } n \geq n_0.$$

$\Omega(g(n))$ můžeme též chápat jako třídu všech nezáporných funkcí $f(n)$:

$$\Omega(g(n)) = \{f(n) \mid \exists c > 0, n_0 \text{ tak, že } f(n) \geq c\, g(n) \ \ \forall\, n \geq n_0\}.$$

1.2.3 Poznámka. Fakt, že funkce $f(n)$ je $\Omega(g(n))$ je ekvivalentní faktu, že funkce $g(n)$ je $\mathcal{O}(f(n))$.

1.2.4 Symbol Θ. Je dána nezáporná funkce $g(n)$. Řekneme, že nezáporná funkce $f(n)$ *je* $\Theta(g(n))$, jestliže existují kladné konstanty c_1, c_2 a přirozené číslo n_0 tak, že

$$c_1\, g(n) \leq f(n) \leq c_2\, g(n) \quad \text{pro všechny } n \geq n_0.$$

$\Theta(g(n))$ můžeme též chápat jako třídu všech nezáporných funkcí $f(n)$:

$$\Theta(g(n)) = \{f(n) \mid \exists c_1, c_2 > 0, n_0 \text{ tak, že } c_1\, g(n) \leq f(n) \leq c_2\, g(n) \ \ \forall\, n \geq n_0\}.$$

1.2.5 Poznámka. Platí $f(n)$ je $\Theta(g(n))$ právě tehdy, když $f(n)$ je zároveň $\mathcal{O}(g(n))$ a $\Omega(g(n))$.

1.2.6 Symbol malé o. Je dána nezáporná funkce $g(n)$. Řekneme, že nezáporná funkce $f(n)$ *je* $o(g(n))$, jestliže pro každou kladnou konstantu c existuje přirozené číslo n_0 tak, že

$$0 \leq f(n) < c\, g(n) \quad \text{pro všechny } n \geq n_0.$$

$o(g(n))$ můžeme též chápat jako třídu všech nezáporných funkcí $f(n)$:

$$o(g(n)) = \{f(n) \mid \forall\, c > 0\ \exists\, n_0 \text{ tak, že } 0 \leq f(n) < c\, g(n) \ \ \forall\, n > n_0\}.$$

1.2.7 Poznámka. Fakt, že nezáporná funkce $f(n)$ je $\mathcal{O}(g(n))$, zhruba řečeno znamená, že funkce $f(n)$ neroste asymptoticky více než funkce $g(n)$. Naproti tomu fakt, že nezáporná funkce $f(n)$ je $o(g(n))$, znamená, že funkce $f(n)$ roste asymptoticky méně než funkce $g(n)$.

1.2.8 Symbol malé ω. Je dána nezáporná funkce $g(n)$. Řekneme, že nezáporná funkce $f(n)$ *je* $\omega(g(n))$, jestliže pro každou kladnou konstantu c existuje přirozené číslo n_0 tak, že

$$0 \leq c\, g(n) < f(n) \quad \text{pro všechny } n \geq n_0.$$

$\omega(g(n))$ můžeme též chápat jako třídu všech nezáporných funkcí $f(n)$:

$$\omega(g(n)) = \{f(n) \mid \forall\, c > 0\ \exists\, n_0 \text{ tak, že } 0 \leq c\, g(n) < f(n) \ \ \forall\, n > n_0\}.$$

1.2.9 Poznámka. Fakt, že nezáporná funkce $f(n)$ je $\Omega(g(n))$, zhruba řečeno znamená, že funkce $f(n)$ roste asymptoticky alespoň tak, jako funkce $g(n)$. Naproti tomu fakt, že nezáporná funkce $f(n)$ je $\omega(g(n))$, znamená, že funkce $f(n)$ roste asymptoticky více než funkce $g(n)$.

1.2.10 Značení. Protože symboly $\mathcal{O}, \Omega, \Theta$ představují množiny funkcí, budeme v dalším textu psát $f(n) \in \mathcal{O}(g(n))$. Je ovšem pravda, že v literatuře najdete i zápis $f(n) = \mathcal{O}(g(n))$. Při tomto zápisu je třeba mít na paměti, že znak rovnosti v zápise $f(n) = \mathcal{O}(g(n))$ nemá stejné vlastnosti jako klasická rovnost. Obdobně pro ostatní symboly.

1.2.11 Tvrzení. Jsou dány dvě nezáporné funkce $f(n)$ a $g(n)$. Existuje-li $\lim_{n\to\infty} \frac{f(n)}{g(n)}$, pak

- $f(n) \in o(g(n))$, jestliže $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$;
- $f(n) \in \omega(g(n))$, jestliže $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$.

1.2.12 Tranzitivita. Máme dány tři nezáporné funkce $f(n)$, $g(n)$ a $h(n)$.

1. Jestliže $f(n) \in \mathcal{O}(g(n))$ a $g(n) \in \mathcal{O}(h(n))$, pak $f(n) \in \mathcal{O}(h(n))$.
2. Jestliže $f(n) \in \Omega(g(n))$ a $g(n) \in \Omega(h(n))$, pak $f(n) \in \Omega(h(n))$.
3. Jestliže $f(n) \in \Theta(g(n))$ a $g(n) \in \Theta(h(n))$, pak $f(n) \in \Theta(h(n))$.

1.2.13 Reflexivita. Pro všechny nezáporné funkce $f(n)$ platí: $f(n) \in \mathcal{O}(f(n))$, $f(n) \in \Omega(f(n))$ a $f(n) \in \Theta(f(n))$.

1.2.14 Tvrzení. $f(n) \in \Theta(g(n))$ právě tehdy, když $g(n) \in \Theta(f(n))$.

1.2.15 Příklady.

1. Pro každé $a > 1$ a $b > 1$ platí
$$\log_a(n) \in \Theta(\log_b(n)).$$
2. V celém textu značíme logaritmus o základu 2 symbolem lg, tj. $\lg(n) = \log_2(n)$. Platí
$$\lg n! \in \Theta(n \lg n).$$

Druhá část tvrzení vyplývá např. z následující věty.

1.2.16 Věta (Gauss). Pro každé $n \geq 1$ platí
$$n^{\frac{n}{2}} \leq n! \leq \left(\frac{n+1}{2}\right)^n.$$

1.2.17 Věta. Máme dánu nezápornou funkci $f(n)$, která je pro dostatečně velká n neklesající. Jestliže platí $f(\frac{n}{2}) \in \Theta(f(n))$, pak
$$\sum_{i=1}^{n} f(i) \in \Theta(n\, f(n)).$$

1.2.18 Poznámka. Vlastnost z předchozí věty má např. funkce $f(n) = n^d$ pro přirozené číslo $d \geq 1$, nemá ji však funkce $f(n) = 2^n$. Pro asymptotický odhad $\sum_{i=1}^{n} 2^i$ se dá využít následující metoda:

Matematickou indukcí dokážeme, že existuje $c > 0$ takové, že

$$\sum_{i=1}^{n} 2^i \leq c\, 2^n.$$

1. Základní krok. Víme, že $\sum_{i=1}^{1} 2^i = 2$ a $2 \leq c\, 2$ pro každou konstantu $c \geq 1$.
2. Indukční krok. Předpokládejme, že platí $\sum_{i=1}^{n} \leq c\, 2^n$. Pak

$$\sum_{i=1}^{n+1} 2^i = \sum_{i=1}^{n} 2^i + 2^{n+1} \leq c\, 2^n + 2^{n+1} = \left(\frac{1}{2} + \frac{1}{c}\right) c\, 2^{n+1}.$$

Nyní k dokončení důkazu stačí zajistit, aby $\frac{1}{2} + \frac{1}{n} \leq 1$. A to je ekvivalentní s podmínkou $c \geq 2$.

1.3 Řešení rekursivních vztahů

1.3.1 Věta — „Master Theorem“. Jsou dána přirozená čísla $a \geq 1$, $b > 1$ a funkce $f(n)$. Předpokládejme, že funkce $T(n)$ je dána na přirozených číslech rekurentním vztahem

$$T(n) = a\, T\left(\frac{n}{b}\right) + f(n),$$

kde $\frac{n}{b}$ znamená buď $\lfloor \frac{n}{b} \rfloor$ nebo $\lceil \frac{n}{b} \rceil$.

1. Jestliže $f(n) \in \mathcal{O}(n^{\log_b a - \varepsilon})$ pro nějakou konstantu $\varepsilon > 0$, pak $T(n) \in \Theta(n^{\log_b a})$.
2. Jestliže $f(n) \in \Theta(n^{\log_b a})$, pak $T(n) \in \Theta(n^{\log_b a} \lg n)$.
3. Jestliže $f(n) \in \Omega(n^{\log_b a + \epsilon})$ pro nějakou konstantu $\varepsilon > 0$ a jestliže $a\, f(\frac{n}{b}) \leq c\, f(n)$ pro nějakou konstantu $c < 1$ pro všechna dostatečně velká n, pak $T(n) \in \Theta(f(n))$.

1.3.2 Poznámka. Věta 1.3.1 nepokrývá všechny možné případy. Případy, které nejsou pokryty:

1. Funkce $f(n) \in \mathcal{O}(n^{\log_b a})$, ale $f(n) \notin \mathcal{O}(n^{\log_b a - \varepsilon})$ pro žádné $\varepsilon > 0$. Jinými slovy, $f(n)$ není polynomiálně menší než $\mathcal{O}(n^{\log_b a})$.
2. Funkce $f(n) \in \Omega(n^{\log_b a})$, ale $f(n) \notin \mathcal{O}(n^{\log_b a + \varepsilon})$ pro žádné $\varepsilon > 0$ (jinými slovy, $f(n)$ není polynomiálně větší než $\mathcal{O}(n^{\log_b a})$) nebo neplatí $a\, f(\frac{n}{b}) \leq c\, f(n)$.

1.3.3 Tvrzení. Jestliže $f(n) \in \Theta(n^{\log_b a} \lg^k n)$ pro $k \geq 0$, pak pro funkci $T(n)$ danou rovnicí

$$T(n) = a\, T\left(\frac{n}{b}\right) + f(n),$$

platí: $T(n) \in \Theta(n^{\log_b a} \lg^{k+1} n)$.

1.3.4 Řešení rekursivních vztahů pomocí rekursivních stromů. Kromě Master Theorem můžeme k řešení rekursivních vztahů použít i metodu rekursivních stromů. Tuto metodu si ukážeme na dvou příkladech.

1.3.5 Příklad 1. Řešme rekurentní vztah

$$T(n) = 3T\left(\frac{n}{4}\right) + n^2.$$

Řešení: Vytvoříme si jednotlivé hladiny stromu, který popisuje rekursivní výpočet funkce $T(n)$. V nulté hladině máme pouze $T(n)$ a hodnotu n^2, kterou potřebujeme k výpočtu $T(n)$ (známe-li $T\left(\frac{n}{4}\right)$.

V první hladině se nám výpočet $T(n)$ rozpadl na tři výpočty $T(\frac{n}{4})$. K tomu potřebujeme hodnotu $3 \cdot (\frac{n}{4})^2 = \frac{3}{16}\, n^2$.

Při přechodu z hladiny i do hladiny $i+1$ se každý vrchol rozdělí na tři a kazdý přispěje do celkové hodnoty jednou šestnáctinou předchozího. Je proto součet v hladině i roven $(\frac{3}{16})^i\, n^2$.

Poslední hladina má vrcholy označené hodnotami $T(1)$ a tím rekurse končí. Počet hladin odpovídá $\log_4 n$. V poslední hladině je $3^{\log_4 n} = n^{\log_4 3}$ hodnot $T(1)$. Proto platí

$$T(n) = \sum_{i=0}^{\log_4 n} \left(\frac{3}{16}\right)^i\, n^2 \;+\; \Theta(n^{\log_4 3}).$$

Odtud

$$T(n) < n^2 \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i + \Theta(n^{\log_4 3}) = n^2 \frac{1}{1-\frac{3}{16}} + \Theta(n^{\log_4 3}) = \frac{16}{13} n^2 + \Theta(n^{\log_4 3}).$$

Proto $T(n) \in \Theta(n^2)$.

Poznamenejme, že tento příklad jsme také mohli řešit pomocí Master Theorem.

1.3.6 Příklad 2. Řešme rekurentní vztah

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + n.$$

Řešení: Vytvoříme si jednotlivé hladiny stromu, který popisuje rekursivní výpočet funkce $T(n)$. V nulté hladině máme pouze $T(n)$ a hodnotu n, kterou potřebujeme k výpočtu $T(n)$ (známe-li $T(\frac{n}{3})$ a $T(\frac{2n}{3})$.

V první hladině se nám výpočet $T(n)$ rozpadl na výpočet $T(\frac{n}{3})$ a $T(\frac{2n}{3})$. K tomu potřebujeme hodnotu $\frac{n}{3} + \frac{2n}{3}$.

Ve druhé hladině se vrchol $T(\frac{n}{3})$ rozpadne na $T(\frac{n}{9})$ a $T(\frac{2n}{9})$; vrchol $T(\frac{2n}{3})$ se rozpadne na $T(\frac{2n}{9})$ a $T(\frac{4n}{9})$. Součet v druhé hladině je

$$\frac{n}{9} + \frac{2n}{9} + \frac{2n}{9} + \frac{4n}{9} = n.$$

V hladině i je součet n právě tehdy, když $\frac{n}{3^i}$ je větší než 1. Ve vyšších hladinách už je součet menší. Poslední nenulová hladina odpovídá takovému i, že

$$n \to \frac{2n}{3} \to \frac{2^2 n}{3^2} \to \ldots \to \frac{2^i n}{3^i} = 1,$$

t.j. $(\frac{2}{3})^i n = 1$, nebo-li $n = (\frac{3}{2})^i$ a $i = \log_{\frac{3}{2}} n$.

Proto

$$T(n) \leq n \log_{\frac{3}{2}} n, \text{ tedy } T(n) \in \mathcal{O}(n \lg n).$$

1.3.7 Strassenův algoritmus. Jedná se o algoritmus pro rychlé násobení čtvercových matic. Algoritmus vychází z následujících rovností, které platí pro násobení matic typu 2×2:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} s_1 + s_2 - s_4 + s_6 & s_4 + s_5 \\ s_6 + s_7 & s_2 - s_3 + s_5 - s_7 \end{pmatrix} \tag{1.1}$$

kde

$$\begin{aligned} s_1 &= (b-d)\cdot(g+h) \\ s_2 &= (a+d)\cdot(e+h) \\ s_3 &= (a-c)\cdot(e+f) \\ s_4 &= (a+b)\cdot h \\ s_5 &= a\cdot(f-h) \\ s_6 &= d\cdot(g-e) \\ s_7 &= (c+d)\cdot e. \end{aligned} \tag{1.2}$$

Vztahů 1.1 a 1.4 využijeme pro násobení čtvercových matic řádu 2^n takto: Matici řádu 2^n rozdělíme na čtyři matice řádu 2^{n-1} a násobíme podle vztahů

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} E & F \\ G & H \end{pmatrix} = \begin{pmatrix} S_1 + S_2 - S_4 + S_6 & S_4 + S_5 \\ S_6 + S_7 & S_2 - S_3 + S_5 - S_7 \end{pmatrix} \tag{1.3}$$

kde

$$\begin{aligned} S_1 &= (B-D)\cdot(G+H) \\ S_2 &= (A+D)\cdot(E+H) \\ S_3 &= (A-C)\cdot(E+F) \\ S_4 &= (A+B)\cdot H \\ S_5 &= A\cdot(F-H) \\ S_6 &= D\cdot(G-E) \\ S_7 &= (C+D)\cdot E. \end{aligned} \tag{1.4}$$

Algoritmus pro matici řádu n vyžaduje 7 násobení matic řádu $\frac{n}{2}$ a 18 sčítání matic řádu $\frac{n}{2}$. Vyřešením rekurentní (diferenční) rovnice

$$T(n) = 7\,T\left(\frac{n}{2}\right) + 18\,\left(\frac{n}{2}\right)^2$$

dostáváme

$$T(n) \in \mathcal{O}(n^{\log_2 7}) = \mathcal{O}(n^{2,81\ldots}),$$

což je $o(n^3)$.