Teoretické základy informatiky II, 2. semestr, dotace 2/1 podrobný sylabus

Martin Kuřil

Úvod

Tento text má sloužit studentům jako pomůcka při studiu předmětu Teoretické základy informatiky II. Je určen především studentům kombinované formy studia. Samozřejmě však může, doufám, pomoci i komukoli jinému.

Na konci, v části 8, najdete pro informaci sylabus předmětu Teoretické základy informatiky II rozdělený do jednotlivých týdnů. Toto rozdělení se pochopitelně týká především studentů prezenční formy studia.

V částech 1 – 7 najdete látku obsaženou v sylabu a ve stejném pořadí, v jakém je řazena v sylabu, avšak seskupenou do tematických celků.

Jednotlivé části textu jsou dvojího typu: Někdy je látka prezentována formou stručného výkladu, včetně několika příkladů. Jindy jsou uvedeny odkazy na literaturu – kam se podívat a co si přečíst a promyslet, abyste se seznámili s probíranou látkou (někdy jsou uvedeny i alternativní odkazy na literaturu).

Literatura, na kterou odkazuji, je uvedena na konci textu, v sekci Reference.

Výkladové části jsou založeny především na následující knize:

Jozef Gruska, *Foundations of Computing*. International Thomson Computer Press, 1997.

Z této knihy jsou převzaty mnohé formulace tvrzení a vět a také mnohé příklady.

Pokud je čtenář vyzván k tomu, aby si samostatně někde něco přečetl (nastudoval), pak je povětšinou odkázán na výukový text

Petr Hliněný, *Základy teorie grafů pro (nejen) informatiky*. Fakulta Informatiky, Masarykova Univerzita, 2010.
`https://is.muni.cz/do/1499/el/estud/fi/js10/grafy/Grafy-text10.pdf`

Výhodou tohoto textu je, že je psán česky a že je volně dostupný na internetu. Je primárně zaměřený pro potřeby studentů informatických oborů na vysokých školách – což jste pravděpodobně také vy (je ale dobře přístupný i ne-informatikům).

Některé odkazy, zvláště alternativní, směřují ke knize

Jiří Matoušek, Jaroslav Nešetřil, *Kapitoly z diskrétní matematiky.* Karolinum, Praha, 2002.

Jde o výbornou knihu, která již vyšla v několika vydáních (naposled, pokud vím, roku 2010). O kvalitě knihy svědčí také fakt, že byla přeložena do angličtiny a pod názvem Invitation to Discrete Mathematics také již vyšla nejméně ve dvou vydáních. Kniha je zaměřena více matematicky než informaticky; doporučit ji mohu především těm, kdo mají zájem o prohloubení znalostí.

Na závěr bych ještě chtěl upozornit na knihu

Ronald L. Graham, Donald E. Knuth, Oren Patashnik, *Concrete Mathematics.* Addison – Wesley Publishing Company, 1989.

Je to velmi dobrá kniha. Nikde na ni v textu neodkazuji. Upozorňuji teď na ni proto, že pro případné zájemce obsahuje dalekosáhlé rozšíření látky prvních čtyř kapitol našeho textu; obsahuje také velké množství cvičení, a to různé obtížnosti, od úloh pro zahřátí až po úlohy velmi obtížné.

Nezapomeňte na to, že pokud budete na univerzitě studovat, pak budete mít svého konkrétního vyučujícího tohoto předmětu. V kontaktní části výuky vám některou část látky vyloží sám, může vám dát jiné odkazy na literaturu, než jsou ty zde uvedené, můžete s ním konzultovat.

Toto je první verze textu. Budou asi následovat verze další, snad lepší. Jistě v textu najdete nějaké chyby, nepřesnosti, ... Omlouvám se za ně a budu rád, když mi o nich dáte vědět.

Obsah

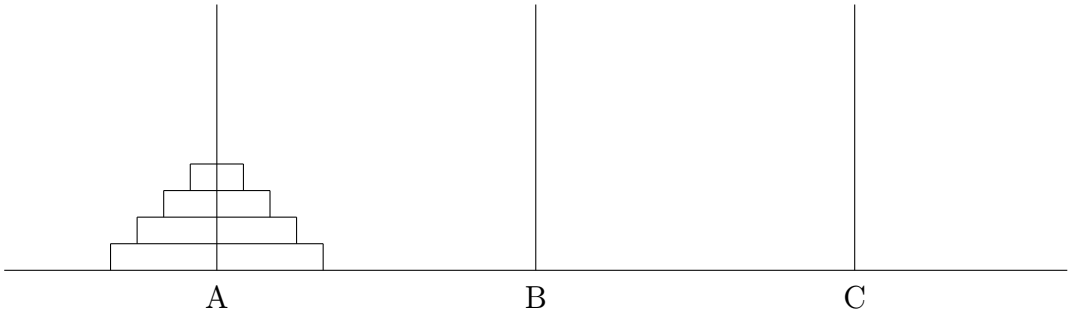
1 Základní metody řešení rekurencí

1.1 Příklady

Volně řečeno, **rekurence** je systém rovností nebo nerovností, které popisují funkci pomocí jejích hodnot pro menší vstupy.

Příklad 1.1.1 (**Problém Hanojských věží**) Jsou dány tři tyčky A,B,C, a n kroužků, které jsou navlečeny v pořadí klesajícího průměru na tyči A, zatímco ostatní dvě tyče jsou prázdné (viz obrázek 1 pro $n = 4$). Úkolem je přemístit kroužky z A na B, možná s použitím C, takovým způsobem, že v jednom kroku je přemístěn jen jeden kroužek a nikdy není kroužek položen na kroužek menšího průměru.

Obrázek 1



Existuje jednoduchý rekurzivní algoritmus pro řešení problému Hanojských věží.

Algoritmus 1.1.2. (**Problém Hanojských věží – rekurzivní algoritmus**)

1. Přesuň $n-1$ horních kroužků z A na C.
2. Přesuň největší kroužek z A na B.
3. Přesuň $n-1$ kroužků z C na B.

Korektnost algoritmu je zřejmá. Počet přesunů jednotlivých kroužků označme $T(n)$. Pro hodnotu $T(n)$ platí rovnosti

$$T(n) = \begin{cases} 1 & \text{pokud} \quad n = 1 \\ 2T(n-1)+1 & \text{pokud} \quad n > 1 \end{cases}$$

Dostali jsme *rekurenci* pro funkci $T(n)$.

Uvažujme následující modifikaci problému Hanojských věží. Cíl je stejný, není však dovoleno přesouvat kroužky z A na B a z B na A. V tomto případě také existuje jednoduchý rekurzivní algoritmus (promyslete si jej!); počet přesunů jednotlivých kroužků v tomto případě označme $T'(n)$. Dostaneme rekurenci

$$T'(n) = \begin{cases} 2 & \text{pokud} \quad n = 1 \\ 3T'(n-1)+2 & \text{pokud} \quad n > 1 \end{cases}$$

Analýza složitosti vede na rekurence vždy, když jsou algoritmy navrženy s využitím metody **rozděl a panuj**.

Příklad 1.1.3. Často lze efektivně vyřešit algoritmický problém P rozměru $n = c^i$ (c, i jsou celá čísla, $c \geq 2$, $i \geq 0$) pomocí následující rekurzivní metody, kde a, b_1, b_2 a d jsou kladné konstanty (a je navíc celé):

1. Rozlož problém P, v čase $b_1 n$, na a podproblémů téhož typu a rozměru $\frac{n}{c}$.

2. Vyřeš všechny podproblémy rekurzivně, s využitím téže metody.

3. Slož, v čase $b_2 n$, řešení problému P z řešení všech jeho podproblémů.

Pro časové nároky $T(n)$ výsledného algoritmu máme rekurenci

$$T(n) = \begin{cases} d & \text{pokud} \quad n = 1 \\ aT(\frac{n}{c}) + b_1 n + b_2 n & \text{pokud} \quad n > 1 \end{cases}$$

1.2 Substituční metoda

Vždy je vhodné začít výpočtem několika prvních hodnot neznámé funkce. To často pomůže

1. uhodnout řešení

2. ověřit získané řešení

Příklad 1.2.1. Pro neznámé funkce $T(n)$ (viz problém Hanojských věží) a $T'(n)$ (viz modifikovaný problém Hanojských věží) dostaneme

n	1	2	3	4	5	6	7	8	9	10
$T(n)$	1	3	7	15	31	63	127	255	511	1023
$T'(n)$	2	8	26	80	242	728	2186	6560	19682	59048

Díky tabulce snadno uhodneme, že $T(n) = 2^n - 1$ a $T'(n) = 3^n - 1$. Samozřejmě, naše odhady musíme dokázat. A k tomu právě můžeme využít substituční metodu – dosadíme do rekurence.

Příklad 1.2.2. (**Problém Hanojských věží**) Ukážeme, že náš odhad $T(n) = 2^n - 1$ je v pořádku. Předně, $T(1) = 2^1 - 1 = 2 - 1 = 1$. Nechť nyní n je celé číslo, $n > 1$. Pak $2T(n-1) + 1 = 2 \cdot (2^{n-1} - 1) + 1 = 2 \cdot 2^{n-1} - 2 + 1 = 2^n - 1 = T(n)$.

Obdobně můžeme ukázat, že $T'(n) = 3^n - 1$ je správné řešení pro modifikovaný problém Hanojských věží.

1.3 Iterační metoda

Iterací rekurence můžeme často rekurenci převést na sumaci, kterou možná snadněji vypočteme či odhadneme.

Příklad 1.3.1. Iterací rekurence pro modifikovaný problém Hanojských věží dostaneme:

$$\begin{aligned}
T'(n) &= 3T'(n-1)+2 \\
&= 3(3T'(n-2)+2)+2 \\
&= 3^2T'(n-2)+3\cdot 2+2 \\
&= 3^2(3T'(n-3)+2)+3\cdot 2+2 \\
&= 3^3T'(n-3)+3^2\cdot 2+3\cdot 2+2 \\
&\vdots \\
&= 3^{n-1}T'(n-(n-1))+3^{n-2}\cdot 2+\cdots+3^2\cdot 2+3\cdot 2+2 \\
&= 3^{n-1}T'(1)+3^{n-2}\cdot 2+\cdots+3^2\cdot 2+3\cdot 2+2 \\
&= 3^{n-1}\cdot 2+3^{n-2}\cdot 2+\cdots+3^2\cdot 2+3\cdot 2+2 \\
&= (3^{n-1}+3^{n-2}+\cdots+3^2+3+1)\cdot 2 \\
&= \frac{3^n-1}{2}\cdot 2 \\
&= 3^n-1
\end{aligned}$$

Příklad 1.3.2. Iterací rekurence

$$T(n)=\begin{cases} b & \text{pokud} \quad n=1 \\ aT(\frac{n}{c})+bn & \text{pokud} \quad n=c^i>1 \end{cases}$$

(klademe $b=b_1+b_2$ a předpokládáme, že $b=d$) získané analýzou algoritmu rozděl a panuj

dostaneme

$$\begin{aligned}
T(n) &= aT\left(\frac{n}{c}\right) + bn \\
&= a\left(aT\left(\frac{n}{c^2}\right) + b\frac{n}{c}\right) + bn \\
&= a^2T\left(\frac{n}{c^2}\right) + bn\frac{a}{c} + bn \\
&= a^2\left(aT\left(\frac{n}{c^3}\right) + b\frac{n}{c^2}\right) + bn\frac{a}{c} + bn \\
&= a^3T\left(\frac{n}{c^3}\right) + bn\left(\frac{a}{c}\right)^2 + bn\frac{a}{c} + bn \\
&\vdots \\
&= a^iT\left(\frac{n}{c^i}\right) + bn\left(\frac{a}{c}\right)^{i-1} + \cdots + bn\left(\frac{a}{c}\right)^2 + bn\frac{a}{c} + bn \\
&= a^iT\left(1\right) + bn\left(\frac{a}{c}\right)^{i-1} + \cdots + bn\left(\frac{a}{c}\right)^2 + bn\frac{a}{c} + bn \\
&= a^ib + bn\left(\frac{a}{c}\right)^{i-1} + \cdots + bn\left(\frac{a}{c}\right)^2 + bn\frac{a}{c} + bn \\
&= bn\left(\frac{a}{c}\right)^{i} + bn\left(\frac{a}{c}\right)^{i-1} + \cdots + bn\left(\frac{a}{c}\right)^2 + bn\frac{a}{c} + bn \\
&= bn\left(\left(\frac{a}{c}\right)^{i} + \left(\frac{a}{c}\right)^{i-1} + \cdots + \left(\frac{a}{c}\right)^2 + \frac{a}{c} + 1\right) \\
&= bn\sum_{0\le j\le i}\left(\frac{a}{c}\right)^j \\
&= bn\sum_{0\le j\le \log_c n}\left(\frac{a}{c}\right)^j
\end{aligned}$$

Takže

- Případ 1, $a < c$: $T(n) = \Theta(n)$
  důvod: suma $\sum_{0\le j}\left(\frac{a}{c}\right)^j$ konverguje, $\sum_{0\le j}\left(\frac{a}{c}\right)^j = s$, $1 \le \sum_{0\le j\le \log_c n}\left(\frac{a}{c}\right)^j \le s$, $bn \le T(n) \le (bs)n$.

- Případ 2, $a = c$: $T(n) = \Theta(n\log n)$
  důvod: $T(n) = bn(1+\log_c n)$ a pro $n \ge c$: $\log_c n \le 1 + \log_c n \le \log_c n + \log_c n \le 2\log_c n$, $bn\log_c n \le bn(1+\log_c n) \le bn\cdot 2\log_c n$, $bn\log_c n \le T(n) \le 2bn\log_c n$, $bn\log_c 10\cdot\log n \le T(n) \le 2bn\log_c 10\cdot\log n$, $(b\log_c 10)(n\log n) \le T(n) \le (2b\log_c 10)(n\log n)$

- Případ 3, $a > c$: $T(n) = \Theta(n^{\log_c a})$
  důvod: $T(n) = bn\sum_{0\le j\le\log_c n}\left(\frac{a}{c}\right)^j \ge bn\left(\frac{a}{c}\right)^{\log_c n} = bn\frac{a^{\log_c n}}{c^{\log_c n}} = bn\frac{a^{\log_c n}}{n} = ba^{\log_c n} =$

$ba^{\log_c a \log_a n} = bn^{\log_c a}$, $T(n) = bn \sum_{0 \le j \le \log_c n} \left(\frac{a}{c}\right)^j = bn \frac{\left(\frac{a}{c}\right)^{\log_c n + 1} - 1}{\frac{a}{c} - 1} \le bn \frac{\left(\frac{a}{c}\right)^{\log_c n + 1}}{\left(\frac{a}{c}\right) - 1} = \frac{b\frac{a}{c}}{\frac{a}{c}-1} n \left(\frac{a}{c}\right)^{\log_c n} = \frac{ba}{a-c} n^{\log_c a}$; celkem $bn^{\log_c a} \le T(n) \le \frac{ba}{a-c} n^{\log_c a}$

Použili jsme zde Θ-notaci; o ní si můžete přečíst v kapitole 3 Asymptotika. Všimněte si, že časové nároky algoritmu rozděl a panuj závisí pouze na poměru $\frac{a}{c}$, a ne na problému, který řešíme ani na výpočetním zařízení, které používáme, ovšem za předpokladu, že dekompozice a kompozice vyžadují pouze lineární čas.

1.4 Redukce na algebraické rovnice

Velká třída rekurencí, tzv. **homogenní lineární rekurence**, může být vyřešena redukcí na algebraické rovnice. Než budeme prezentovat obecnou metodu, ukážeme základní myšlenku na příkladě.

Příklad 1.4.1. (**Fibonacciho čísla**) Leonardo Fibonacci v roce 1202 zavedl posloupnost čísel definovanou rekurencí

$$F_0 = 0,\ F_1 = 1 \text{ (počáteční podmínky)}$$

$$F_n = F_{n-1} + F_{n-2} \text{ pro } n \ge 2 \text{ (induktivní rovnice)}$$

Fibonacciho čísla tvoří jednu z nejzajímavějších posloupností přirozených čísel:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, \ldots$$

Je přirozené se ptát, zda pro dané kladné celé číslo n můžeme určit F_n bez počítání F_i pro všechna celá čísla i, $0 \le i < n$. Tedy: můžeme najít explicitní formuli pro F_n?

Pokusme se nejprve najít řešení induktivní rovnice ve tvaru $F_n = r^n$, kde r je zatím neznámá konstanta. Předpokládejme, že r^n vyhovuje induktivní rovnici. Pak

$$r^n = r^{n-1} + r^{n-2}$$

pro všechna celá čísla n, $n \ge 2$. Jistě je $r \neq 0$ (pro $r = 0$ dostaneme posloupnost samých nul); vydělením číslem r^{n-2} dostaneme

$$r^2 = r + 1$$

Tato rovnice má dva kořeny:

$$r_1 = \frac{1 + \sqrt{5}}{2},\ r_2 = \frac{1 - \sqrt{5}}{2}$$

Bohužel žádná z funkcí r_1^n, r_2^n nesplňuje počáteční podmínky. Naštěstí každá lineární kombinace $\lambda r_1^n + \mu r_2^n$ vyhovuje induktivní rovnici:

$$\begin{aligned}(\lambda r_1^{n-1} + \mu r_2^{n-1}) + (\lambda r_1^{n-2} + \mu r_2^{n-2}) &= \lambda(r_1^{n-1} + r_1^{n-2}) + \mu(r_2^{n-1} + r_2^{n-2}) \\ &= \lambda r_1^{n-2}(r_1 + 1) + \mu r_2^{n-2}(r_2 + 1) \\ &= \lambda r_1^{n-2} r_1^2 + \mu r_2^{n-2} r_2^2 \\ &= \lambda r_1^n + \mu r_2^n\end{aligned}$$

Jestliže se podaří najít λ a μ tak, aby byly splněny také počáteční podmínky, tj.

$$\lambda r_1^0 + \mu r_2^0 = 0,\ \lambda r_1^1 + \mu r_2^1 = 1$$

pak bude $F_n = \lambda r_1^n + \mu r_2^n$.

Čísla λ a μ snadno najdeme vyřešením systému dvou lineárních rovnic o dvou neznámých. Dostaneme

$$\lambda = -\mu = \frac{1}{\sqrt{5}}$$

a tedy

$$F_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

Protože $\left|\frac{1-\sqrt{5}}{2}\right| < 1$, je $\lim_{n\to\infty}\left(\frac{1-\sqrt{5}}{2}\right)^n = 0$, takže dostáváme přibližné vyjádření:

$$F_n \approx \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n \text{ pro } n \to \infty$$

Metodu použitou v předchozím příkladě nyní zobecníme.Uvažme **homogenní lineární rekurenci**, to znamená rekurenci, kde hodnota funkce je vyjádřena lineární kombinací pevně daného počtu jejích hodnot pro menší argumenty:

$$u_n = a_1 u_{n-1} + a_2 u_{n-2} + \cdots + a_k u_{n-k} \text{ pro } n \geq k \text{ (induktivní rovnice)}$$

$$u_i = b_i \text{ pro } 0 \leq i < k \text{ (počáteční podmínky)}$$

kde k je kladné celé číslo, $a_1, \ldots, a_k$ a $b_0, \ldots, b_{k-1}$ jsou konstanty (komplexní čísla).

Polynom

$$P(r) = r^k - (a_1 r^{k-1} + a_2 r^{k-2} + \cdots + a_k) = r^k - \sum_{1\leq j\leq k} a_j r^{k-1}$$

se nazývá **charakteristický polynom** induktivní rovnice a $P(r) = 0$ je její **charakteristická rovnice**. Kořeny charakteristického polynomu se nazývají **charakteristické kořeny**

induktivní rovnice. Následující věta ukazuje, že vždy můžeme najít řešení homogenní lineární rekurence, známe-li charakteristické kořeny.

Věta 1.4.2.

1. *Jestliže charakteristická rovnice $P(r) = 0$ má k vzájemně různých kořenů $r_1, \ldots, r_k$, pak homogenní lineární rekurence má řešení*

$$u_n = \sum_{1 \le j \le k} \lambda_j r_j^n$$

kde λ_j jsou řešení systému lineárních rovnic

$$b_i = \sum_{1 \le j \le k} \lambda_j r_j^i, \ 0 \le i < k$$

2. *Jestliže charakteristická rovnice $P(r) = 0$ má p vzájemně různých kořenů $r_1, \ldots, r_p$, $p < k$, a kořen r_j, $1 \le j \le p$, má násobnost m_j, pak r_j^n, nr_j^n, $n^2 r_j^n$, $\ldots$, $n^{m_j - 1} r_j^n$ jsou řešení induktivní rovnice a existuje řešení splňující počáteční podmínky, které má tvar*

$$u_n = \sum_{1 \le j \le p} P_j(n) r_j^n$$

kde každé $P_j(n)$ je polynom stupně $m_j - 1$, jehož koeficienty lze získat řešením systému lineárních rovnic

$$b_i = \sum_{1 \le j \le p} P_j(i) r_j^i, \ 0 \le i < k$$

Důkaz.

1. Nejprve ukážeme, že $u_n = \sum_{1 \le j \le k} \lambda_j r_j^n$ je řešením induktivní rovnice pro libovolná čísla

$\lambda_1, \ldots, \lambda_k$. Nechť n je celé číslo, $n \geq k$. Počítejme:

$$\begin{aligned}
a_1 u_{n-1} + \cdots + a_k u_{n-k} &= a_1 \sum_{1 \leq j \leq k} \lambda_j r_j^{n-1} + \cdots + a_k \sum_{1 \leq j \leq k} \lambda_j r_j^{n-k} \\
&= \sum_{1 \leq j \leq k} a_1 \lambda_j r_j^{n-1} + \cdots + \sum_{1 \leq j \leq k} a_k \lambda_j r_j^{n-k} \\
&= \sum_{1 \leq j \leq k} a_1 \lambda_j r_j^{n-1} + \cdots + a_k \lambda_j r_j^{n-k} \\
&= \sum_{1 \leq j \leq k} \lambda_j r_j^{n-k} (a_1 r_j^{k-1} + \cdots + a_k) \\
&= \sum_{1 \leq j \leq k} \lambda_j r_j^{n-k} r_j^k \\
&= \sum_{1 \leq j \leq k} \lambda_j r_j^n \\
&= u_n
\end{aligned}$$

Stačí již jen ukázat, že systém lineárních rovnic (s neznámými λ_j)

$$b_i = \sum_{1 \leq j \leq k} \lambda_j r_j^i, \ 0 \leq i < k$$

má řešení. Matice systému má tvar

$$\begin{pmatrix}
1 & 1 & \ldots & 1 \\
r_1 & r_2 & \ldots & r_k \\
r_1^2 & r_2^2 & \ldots & r_k^2 \\
\vdots & \vdots & & \vdots \\
r_1^{k-1} & r_2^{k-1} & \ldots & r_k^{k-1}
\end{pmatrix}$$

To je známá matice z lineární algebry, totiž Vandermondova matice, jejíž determinant je roven

$$\prod_{1 \leq i < j \leq k} (r_j - r_i) \neq 0$$

(je třeba si připomenout, že čísla $r_1, \ldots, r_k$ jsou vzájemně různá). Matice soustavy má nenulový determinant, takže soustava má přesně jedno řešení – hledaná čísla $\lambda_1, \ldots, \lambda_k$ existují a jsou určena jednoznačně.

2. Důkaz druhé části věty je značně technický a nebudeme jej provádět. Základní ideu důkazu lze najít v knize [3], Theorem 1.2.13 (strana 12). Viz také Tvrzení 10.3.1 (strana 308) v knize [9].

Použití Věty 1.4.2. ukážeme na dvou příkladech.

Příklad 1.4.3. Vyřešíme rekurenci

$$u_n = 3u_{n-1} - 2u_{n-2},\ n \geq 2;\ u_0 = 0,\ u_1 = 1$$

Charakteristická rovnice $r^2 = 3r - 2$ má kořeny $r_1 = 1$ a $r_2 = 2$, takže $u_n = \lambda_1 \cdot 1^n + \lambda_2 \cdot 2^n = \lambda_1 + \lambda_2 \cdot 2^n$. Hodnoty λ_1, λ_2 zjistíme vyřešením systému rovnic $0 = \lambda_1 + \lambda_2 \cdot 2^0$, $1 = \lambda_1 + \lambda_2 \cdot 2^1$, tedy $0 = \lambda_1 + \lambda_2$, $1 = \lambda_1 + 2\lambda_2$. Dostaneme $\lambda_1 = -1$, $\lambda_2 = 1$, což dává $u_n = 2^n - 1$.

Příklad 1.4.4. Vyřešíme rekurenci

$$u_n = 5u_{n-1} - 8u_{n-2} + 4u_{n-3},\ n \geq 3; u_0 = 0,\ u_1 = -1,\ u_2 = 2$$

Charakteristická rovnice $r^3 = 5r^2 - 8r + 4$ má kořeny $r_1 = 1$ násobnosti $m_1 = 1$ a $r_2 = 2$ násobnosti $m_2 = 2$. Je tedy $u_n = P_1(n) \cdot 1^n + P_2(n) \cdot 2^n = P_1(n) + P_2(n) \cdot 2^n$, kde $P_1(n)$ je polynom stupně $m_1 - 1 = 0$, $P_2(n)$ je polynom stupně $m_2 - 1 = 1$. Nechť $P_1(n) = a$, $P_2(n) = b + cn$. Hodnoty a, b, c zjistíme vyřešením systému rovnic $0 = P_1(0) + P_2(0) \cdot 2^0$, $-1 = P_1(1) + P_2(1) \cdot 2^1$, $2 = P_1(2) + P_2(2) \cdot 2^2$, tedy $0 = a + (b + c \cdot 0) \cdot 1$, $-1 = a + (b + c \cdot 1) \cdot 2$, $2 = a + (b + c \cdot 2) \cdot 4$, tedy $0 = a + b$, $-1 = a + 2b + 2c$, $2 = a + 4b + 8c$. Dostaneme $a = 6$, $b = -6$, $c = \frac{5}{2}$ a tedy $u_n = 6 + (-6 + \frac{5}{2}n) \cdot 2^n$.

2 Speciální funkce

2.1 Funkce dolní a horní celé části

Existují dvě základní funkce, které převádějí reálná čísla na celá. Nechť x je reálné číslo. Definujeme

$$\text{dolní celá část: } \lfloor x \rfloor \text{ – největší celé číslo} \leq x$$

$$\text{horní celá část: } \lceil x \rceil \text{ – nejmenší celé číslo} \geq x$$

Například

$$\lfloor 3,14 \rfloor = 3 = \lfloor 3,75 \rfloor,\ \lfloor -3,14 \rfloor = -4 = \lfloor -3,75 \rfloor$$
$$\lceil 3,14 \rceil = 4 = \lceil 3,75 \rceil,\ \lceil -3,14 \rceil = -3 = \lceil -3,75 \rceil$$

Snadno se ověří následující základní vlastnosti funkcí dolní a horní celá část (x označuje libovolné reálné číslo):

$$\lfloor x + n \rfloor = \lfloor x \rfloor + n \text{ a } \lceil x + n \rceil = \lceil x \rceil + n, \text{ pokud } n \text{ je celé číslo}$$

$$\lfloor x \rfloor = x \Leftrightarrow x \text{ je celé číslo} \Leftrightarrow \lceil x \rceil = x$$

$$x - 1 < \lfloor x \rfloor \leq x \leq \lceil x \rceil < x + 1$$

$$\lfloor -x \rfloor = -\lceil x \rceil \text{ a } \lceil -x \rceil = -\lfloor x \rfloor$$

$$\lfloor x \rfloor \leq \left\lceil \frac{x}{2} \right\rceil + \left\lfloor \frac{x}{2} \right\rfloor \leq \lceil x \rceil$$

$$\left\lceil \tfrac{n}{2} \right\rceil = \left\lfloor \tfrac{n+1}{2} \right\rfloor \text{ pro každé celé číslo } n$$

$$\lceil x \rceil - \lfloor x \rfloor = \begin{cases} 0 & \text{pokud } x \in \mathbb{Z} \\ 1 & \text{jinak} \end{cases}$$

Výrazy, které obsahují více výskytů funkcí dolní a horní celá část, obvykle není snadné zpracovat. Je proto přirozené se ptát, zda můžeme některé z výskytů těchto funkcí z výrazu odstranit, ovšem bez vlivu na hodnotu výrazu. Případ $\lfloor \lceil x \rceil \rfloor$ je jasný. Je to již méně jasné, pokud chceme zjednodušit výrazy jako třeba $\lfloor \sqrt{\lfloor x \rfloor} \rfloor$. K těmto úpravám se může hodit následující tvrzení.

Tvrzení 2.1.1. *Nechť $f(x)$ je spojitá rostoucí funkce splňující: jestliže $f(x)$ je celé číslo, pak x je také celé číslo. Pak*

$$\lfloor f(\lfloor x \rfloor) \rfloor = \lfloor f(x) \rfloor \text{ a } \lceil f(\lceil x \rceil) \rceil = \lceil f(x) \rceil$$

Tvrzení platí i pro funkci s definičním oborem $\langle c, +\infty)$, kde c je celé číslo.

Důkaz. Rovnost dokážeme jen pro případ dolní celé části. Důkaz pro horní celou část je obdobný. Je $\lfloor x \rfloor \leq x$, takže $f(\lfloor x \rfloor) \leq f(x)$, protože funkce $f(x)$ je rostoucí. Pak $\lfloor f(\lfloor x \rfloor) \rfloor \leq \lfloor f(x) \rfloor$. Předpokládejme, že $\lfloor f(\lfloor x \rfloor) \rfloor \neq \lfloor f(x) \rfloor$. Pak $\lfloor f(\lfloor x \rfloor) \rfloor < \lfloor f(x) \rfloor$ (pak ovšem $\lfloor x \rfloor < x$). To znamená, že existuje celé číslo a takové, že $f(\lfloor x \rfloor) < a \leq f(x)$. Protože funkce $f(x)$ je spojitá, existuje y takové, že $\lfloor x \rfloor < y \leq x$, $f(y) = a$. Protože a je celé číslo, je také y celé číslo. Připomeňme, že $\lfloor x \rfloor < y \leq x$. Dostali jsme spor. Nutně tedy $\lfloor f(\lfloor x \rfloor) \rfloor = \lfloor f(x) \rfloor$.

Vezměme například funkci $\sqrt{x}$. Tato funkce je na intervalu $\langle 0, +\infty)$ rostoucí, spojitá a pokud $\sqrt{x}$ je celé číslo, je také x celé číslo. Z Tvrzení 2.1.1. pak plyne, že pro každé nezáporné reálné číslo x je $\lfloor \sqrt{\lfloor x \rfloor} \rfloor = \lfloor \sqrt{x} \rfloor$. Obdobně, pro každé reálné číslo x, $x \geq 1$, je $\lfloor \log \lfloor x \rfloor \rfloor = \lfloor \log x \rfloor$.

2.2 Logaritmy

Logaritmická funkce $\log_a x$, kde a je reálné číslo, $a > 1$, je spojitá a rostoucí pro $x \geq 1$. Jestliže a je celé číslo, pak $\log_a x$ nabývá celočíselných hodnot pouze pro celá x. Dle Tvrzení 2.1.1. pak pro všechna reálná čísla x, $x \geq 1$, je $\lceil \log_a \lceil x \rceil \rceil = \lceil \log_a x \rceil$ a $\lfloor \log_a \lfloor x \rfloor \rfloor = \lfloor \log_a x \rfloor$.

Logaritmické funkce se základem 2, e a 10 se vyskytují tak často, že se pro ně používá specielní značení:

$$\begin{aligned} \lg n &= \log_2 n \quad \text{(binární logaritmus)} \\ \ln n &= \log_e n \quad \text{(přirozený logaritmus)} \\ \log n &= \log_{10} n \quad \text{(dekadický logaritmus)} \end{aligned}$$

Pro všechna kladná reálná čísla x, a, b, $a \neq 1$, $b \neq 1$, platí:

$$\log_a x = \frac{\log_b x}{\log_b a}$$

.

Za zmínku stojí některé důležité interpretace binárních logaritmů: $\lceil \lg(n+1) \rceil$ je počet bitů v binární reprezentaci kladného celého čísla n; $\lceil \lg n \rceil$ je minimální hloubka binárního stromu s n listy; $\lceil \lg n \rceil$ je počet paralelních kroků při výpočtu $x_1 \circ x_2 \circ \cdots \circ x_n$, kde $\circ$ je asociativní binární operace.

Často se setkáme také s mocninami a kompozicemi logaritmických funkcí. Pro ně se používá následující notace (n je libovolné kladné reálné číslo, k je libovolné kladné celé číslo):

$$\begin{aligned} \lg^k n &= (\lg n)^k, \\ \lg \lg n &= \lg(\lg n), \text{ pokud } \lg n > 0 \\ \lg \lg \lg n &= \lg(\lg(\lg n)), \text{ pokud existuje } \lg(\lg n) > 0 \\ \lg^{(0)} n &= n \\ \lg^{(k)} n &= \underbrace{\lg \lg \ldots \lg}_{k} n, \text{ pokud existuje } \lg^{(k-1)} n > 0 \end{aligned}$$

Příbuznou funkcí je **iterovaný logaritmus**:

$$\lg^* n = \min\{i \in \mathbb{Z} |\ i \geq 0,\ \lg^{(i)} n \leq 1\}$$

Tato funkce roste velmi pomalu:

$$\lg^* 2 = 1,\ \lg^* 4 = 2,\ \lg^* 16 = 3,\ \lg^* 65536 = 4,\ \lg^* 2^{65536} = 5$$

Přitom číslo 2^{65536} je obrovské, k jeho zápisu v desítkové soustavě potřebujene více než 19600 cifer. Avšak také funkce $\lg \lg n$ a $\lg \lg \lg n$ rostou velmi pomalu, například

$$\lg \lg 2^{65536} = 16,\ \lg \lg \lg 2^{65536} = 4$$

Inverzní funkcí k funkci přirozený logaritmus je funkce e^x (někdy se značí $\exp(x)$).

2.3 Binomické koeficienty

Nechť n je reálné číslo, k je celé číslo. Binomický koeficient $\binom{n}{k}$ (čteme "n nad k") definujeme následovně:

$$\binom{n}{k} = \begin{cases} 0 & k < 0 \\ 1 & k = 0 \\ \frac{n(n-1)\ldots(n-k+1)}{k(k-1)\ldots 1} & k > 0 \end{cases}$$

Jestliže k, n jsou celá čísla, $0 \leq k \leq n$, pak $\binom{n}{k}$ je počet k-prvkových podmnožin n-prvkové množiny.

Důležitá je **binomická věta**: pro všechna nezáporná celá čísla n platí

$$(x+y)^n = \sum_{0 \leq k \leq n} \binom{n}{k} x^k y^{n-k}$$

Uvedeme nyní několik důležitých identit pro binomické koeficienty. Lze je snadno dokázat s využitím definice. Jejich znalost umožňuje zjednodušovat poměrně komplikované výrazy a sumy.

1. Nechť n je nezáporné celé číslo a k je celé číslo. Pak

$$\binom{n}{k} = \binom{n}{n-k}$$

2. Nechť n je reálné číslo a k je celé číslo. Pak

$$\binom{n-1}{k-1} + \binom{n-1}{k} = \binom{n}{k}$$

3. Nechť k a n jsou celá čísla, $0 \leq k \leq n$. Pak

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

4. Nechť n je reálné číslo a k je celé číslo, $k \neq 0$. Pak

$$\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$$

Příklad 2.3.1. Dosazením $x = y = 1$ do binomické věty dostaneme, že pro každé nezáporné celé číslo n je

$$\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{n-1} + \binom{n}{n} = 2^n$$

Příklad 2.3.2. Další zajímavé identity získáme opakovaným použitím základní identity $\binom{n-1}{k-1} + \binom{n-1}{k} = \binom{n}{k}$.

Budeme opakovaně rozvíjet první sčítanec. Nechť k je nezáporné celé číslo. Dostaneme:

$$\begin{aligned}
\binom{n+1}{k} &= \binom{n}{k-1} + \binom{n}{k} \\
&= \binom{n-1}{k-2} + \binom{n-1}{k-1} + \binom{n}{k} \\
&= \binom{n-2}{k-3} + \binom{n-2}{k-2} + \binom{n-1}{k-1} + \binom{n}{k} \\
&= \binom{n-3}{k-4} + \binom{n-3}{k-3} + \binom{n-2}{k-2} + \binom{n-1}{k-1} + \binom{n}{k} \\
&\vdots \\
&= \binom{n-k}{k-(k+1)} + \binom{n-k}{k-k} + \binom{n-(k-1)}{k-(k-1)} + \cdots + \binom{n-2}{k-2} + \binom{n-1}{k-1} + \binom{n}{k} \\
&= \binom{n-k}{-1} + \binom{n-k}{0} + \binom{n-(k-1)}{1} + \cdots + \binom{n-2}{k-2} + \binom{n-1}{k-1} + \binom{n}{k} \\
&= 0 + \binom{n-k}{0} + \binom{n-(k-1)}{1} + \cdots + \binom{n-2}{k-2} + \binom{n-1}{k-1} + \binom{n}{k} \\
&= \binom{n-k}{0} + \binom{n-(k-1)}{1} + \cdots + \binom{n-2}{k-2} + \binom{n-1}{k-1} + \binom{n}{k}
\end{aligned}$$

Položíme $m = n - k$ a dostaneme

$$\binom{m}{0} + \binom{m+1}{1} + \binom{m+2}{2} + \cdots + \binom{m+k-1}{k-1} + \binom{m+k}{k} = \binom{m+k+1}{k}$$

Pro každé reálné číslo m a každé nezáporné celé číslo k platí

$$\sum_{0 \le i \le k} \binom{m+i}{i} = \binom{m}{0} + \binom{m+1}{1} + \cdots + \binom{m+k-1}{k-1} + \binom{m+k}{k} = \binom{m+k+1}{k}$$

3 Asymptotika

V této kapitole se budeme zabývat funkcemi, které zobrazují množinu kladných celých čísel do množiny reálných čísel. Symbol n bude vždy (neřekneme-li něco jiného) značit kladné celé číslo.

3.1 Asymptotická hierarchie

Důležitá formalizace intuitivní ideje, že jedna funkce roste podstatně rychleji než druhá funkce, je zachycena relací $\prec$ definovanou takto:

$$f(n) \prec g(n) \Leftrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

(zde předpokládáme, že $g(n) \neq 0$ pro dostatečně velká n, tedy pro všechna $n \geq n_0$, kde n_0 je nějaká konstanta)

Základní vlastnosti relace $\prec$ jsou shrnuty zde:

$$\begin{array}{rcll} f(n) \prec g(n),\, g(n) \prec h(n) & \Rightarrow & f(n) \prec h(n) & \text{(tranzitivita)} \\ f(n) \prec g(n) & \Leftrightarrow & \frac{1}{g(n)} \prec \frac{1}{f(n)} & (f(n) \neq 0,\, g(n) \neq 0 \text{ pro všechna } n) \\ 1 \prec f(n) \prec g(n) & \Rightarrow & c^{f(n)} \prec c^{g(n)} & (c > 1 \text{ je reálné číslo},\, g(n) > 0 \text{ pro všechna } n) \\ n^\alpha \prec n^\beta & \Leftrightarrow & \alpha < \beta & (\alpha,\, \beta \text{ jsou reálná čísla}) \end{array}$$

Uvedené vztahy lze odvodit z definice s použitím elementárních metod analýzy. Ukažme to na příklad pro předposlední vztah. Nechť $1 \prec f(n) \prec g(n)$. Je $f(n) - g(n) = g(n)\left(\frac{f(n)}{g(n)} - 1\right)$. Z $f(n) \prec g(n)$ máme $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ a tedy $\lim_{n\to\infty}\left(\frac{f(n)}{g(n)} - 1\right) = -1$. Díky tranzitivitě relace $\prec$ dostáváme $1 \prec g(n)$ a tedy $\lim_{n\to\infty} \frac{1}{g(n)} = 0$, z čehož plyne, že $\lim_{n\to\infty} g(n) = +\infty$ (využili jsme fakt, že $g(n) > 0$ pro všechna n). Z $\lim_{n\to\infty} g(n) = +\infty$ a $\lim_{n\to\infty}\left(\frac{f(n)}{g(n)} - 1\right) = -1$ máme $\lim_{n\to\infty}(f(n) - g(n)) = -\infty$. Je $\frac{c^{f(n)}}{c^{g(n)}} = c^{f(n)-g(n)}$. Protože $\lim_{n\to\infty}(f(n) - g(n)) = -\infty$ a $c > 1$, je $\lim_{n\to\infty} c^{f(n)-g(n)} = 0$ a tedy $c^{f(n)} \prec c^{g(n)}$.

Následující hierarchie (kde ϵ, c jsou reálná čísla, $0 < \epsilon < 1 < c$) také může být odvozena s použitím elementárních metod analýzy.

$$1 \prec \lg^* n \prec \lg\lg n \prec \lg n \prec n^\epsilon \prec n^c \prec n^{\lg n} \prec c^n \prec n^n \prec c^{c^n}$$

Vztahy zůstanou v platnosti, nahradíme-li lg funkcí ln nebo log.

Příklad 3.1.1. Kde bude v uvedené hierarchii zařazena funkce $2^{\sqrt{\lg n}}$? Je $1 \prec \lg\lg n \prec \sqrt{\lg n} \prec \epsilon \lg n$ pro libovolné kladné reálné číslo ϵ. O tom se snadno přesvědčíme dosazením $n = 2^{2^{2k}}$: $\lg\lg n = 2k$, $\sqrt{\lg n} = 2^k$, $\epsilon \lg n = \epsilon 2^{2k}$. S využitím základních vlastností relace $\prec$ máme $2^{\lg\lg n} \prec 2^{\sqrt{\lg n}} \prec 2^{\epsilon \lg n}$. Jelikož $2^{\lg\lg n} = \lg n$, $2^{\epsilon \lg n} = n^\epsilon$, je

$$\lg n \prec 2^{\sqrt{\lg n}} \prec n^\epsilon$$

Obdobně můžeme formalizovat intuitivní ideu, že funkce $f(n)$ a $g(n)$ mají stejný řád růstu, a to následovně:

$$f(n) \sim g(n) \Leftrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$$

Bylo by pěkné, pokud bychom mohli říci, že pro každé dvě funkce $f(n)$, $g(n)$ platí jeden ze vztahů $f(n) \prec g(n)$, $f(n) \sim g(n)$, $g(n) \prec f(n)$. To však není pravda. Například pro dvojice $f(n) = n$, $g(n) = 2n$ a $f(n) = 1$, $g(n) = \sin n$ neplatí žádný z uvedených vztahů.

Existuje další formalizace intuitivní ideje, že funkce $f(n)$ a $g(n)$ mají stejný řád růstu:

$$f(n) \asymp g(n) \Leftrightarrow |f(n)| \leq c|g(n)| \wedge |g(n)| \leq c|f(n)|$$

pro nějakou kladnou reálnou konstantu c a všechna dostatečně velká n.

Tato formalizace má příjemnou a důležitou vlastnost: Pro libovolné dvě **logaritmicko-exponenciální funkce** $f(n)$ a $g(n)$ platí právě jeden ze vztahů $f(n) \prec g(n)$, $f(n) \asymp g(n)$, $g(n) \prec f(n)$. Třída logaritmicko-exponenciálních funkcí obsahuje téměř všechny funkce, s nimiž se setkáme při asymptotické analýze. Zavedl ji G.H.Hardy roku 1910 a může být definována jako nejmenší třída funkcí $\mathcal{L}$ splňující následující podmínky:

1. Funkce $f(n) = n$ a $f(n) = c$ (c je reálná konstanta) patří do $\mathcal{L}$.
2. Jestliže $f(n)$ a $g(n)$ jsou v $\mathcal{L}$, pak v $\mathcal{L}$ jsou také funkce $f(n)+g(n)$, $f(n)-g(n)$, $f(n)\cdot g(n)$, $e^{f(n)}$, $\lfloor f(n) \rfloor$, $\frac{g(n)}{f(n)}$ a $\ln f(n)$ – pokud $f(n) > 0$ pro dostatečně velká n.

Asymptotická relace $\prec$ a další, které přijdou později, nás vedou k tomu, abychom uvažovali ve velkých číslech.

Například, hierarchie nám říká, že $\log n \prec n^{0,0001}$; to se může zdát jako chyba, pokud uvažujeme v malinkých číslech, jako třeba $n = 10^{100}$. Je totiž $\log 10^{100} = 100$, $(10^{100})^{0,0001} = 10^{0,01} = 1,023\ldots$. Ale pro $n = 10^{10^{100}}$ máme $\log n = 10^{100}$, což je mnohem méně než $n^{0,0001} = 10^{10^{96}}$.

Dokonce i když ϵ bude extrémně malé kladné reálné číslo, bude hodnota $\log n$ mnohem menší než n^ϵ pokud n bude dostatečně velké. Abychom to nahlédli, položme $n = 10^{10^{2k}}$ kde k je kladné celé číslo splňující $\epsilon \geq 10^{-k}$. Máme $\log n = 10^{2k}$, avšak $n^\epsilon \geq 10^{10^k}$. Tudíž $\frac{\log n}{n^\epsilon}$ konverguje k nule pro $n \to \infty$.

Obecně, výsledky asymptotické analýzy mají tím větší výpovědní hodnotu, čím větší argumenty máme. Z toho plyne, že praktický význam výsledků asymptotické analýzy stoupá s růstem velikosti (rozměru) řešených problémů.

3.2 $\mathcal{O}$-, Θ- a Ω- notace

Hlavní pojmy asymptotické analýzy zaznamenává $\mathcal{O}$- (velké ó), Θ- (velké theta) a Ω- (velké omega) notace s funkcí $g(n)$ jako argumentem.

Pro funkci $g(n)$ definujeme množiny $\mathcal{O}(g(n))$, $\Theta(g(n))$, $\Omega(g(n))$ takto:

$$f(n) \in \mathcal{O}(g(n))$$

právě tehdy, když existují reálná čísla c, n_0, $c > 0$, tak, že pro všechna $n \geq n_0$ je

$$|f(n)| \leq c|g(n)|;$$

$$f(n) \in \Theta(g(n))$$

právě tehdy, když existují reálná čísla c_1, c_2, n_0, $c_1 > 0$, $c_2 > 0$, tak, že pro všechna $n \geq n_0$ je

$$c_1|g(n)| \leq |f(n)| \leq c_2|g(n)|;$$

$$f(n) \in \Omega(g(n))$$

právě tehdy, když existují reálná čísla c, n_0, $c > 0$, tak, že pro všechna $n \geq n_0$ je

$$c|g(n)| \leq |f(n)|.$$

$\mathcal{O}$-notaci zavedl roku 1892 německý matematik P.H.Bachmann (1837 – 1920). Její používání dále rozšířil další německý matematik Edmund Landau (1877 – 1938) a byla pak známa jako Landauova notace. D.E.Knuth pak zavedl Θ- a Ω- notaci a všechny tyto notace zpopularizoval.

$\mathcal{O}(g(n))$, $\Theta(g(n))$ a $\Omega(g(n))$ jsou množiny funkcí, avšak

$$\begin{array}{rlll} \text{místo} & f(n) \in \mathcal{O}(g(n)), & f(n) \in \Theta(g(n)), & f(n) \in \Omega(g(n)), \\ \text{píšeme obvykle} & f(n) = \mathcal{O}(g(n)), & f(n) = \Theta(g(n)), & f(n) = \Omega(g(n)), \end{array}$$

To má několik důvodů. Jedním je tradice. $\mathcal{O}$- notace se znamením rovnosti je již zavedena v matematice, především v teorii čísel. Dále, často čteme "$f(n)$ je velké theta $g(n)$", nikoli "$f(n)$ patří do velkého theta $g(n)$". Jsou i další důvody.

Jednou z důležitých vlastností $\mathcal{O}$-, Θ- a Ω- notací je tranzitivita. Například

$$f(n) = \mathcal{O}(g(n)) \wedge g(n) = \mathcal{O}(h(n)) \Rightarrow f(n) = \mathcal{O}(h(n))$$

3.2.1. Příklad. Ukážeme, že $(n+1)^2 = \Theta(n^2)$. Hledáme tedy reálná čísla c_1, c_2, n_0, $c_1 > 0$, $c_2 > 0$, taková, že pro $n \geq n_0$ bude

$$c_1|n^2| \leq |(n+1)^2| \leq c_2|n^2|, \text{ tj. } c_1 n^2 \leq n^2 + 2n + 1 \leq c_2 n^2$$

Po vydělení číslem n^2 máme

$$c_1 \leq 1 + \frac{2}{n} + \frac{1}{n^2} \leq c_2$$

Můžeme tedy vzít $n_0 = 1$, $c_1 = 1$, $c_2 = 4$.

Příklad 3.2.2. Ukážeme, že $\frac{n^2-1}{n+1} = \Theta(n)$. Hledáme tedy reálná čísla c_1, c_2, n_0, $c_1 > 0$, $c_2 > 0$, taková, že pro $n \geq n_0$ bude

$$c_1|n| \leq \left|\frac{n^2-1}{n+1}\right| \leq c_2|n|, \text{ tj. } c_1 n \leq n-1 \leq c_2 n$$

Položme $n_0 = 2$, $c_1 = \frac{1}{2}$, $c_2 = 1$. Jistě $n - 1 \leq 1 \cdot n$. Dále

$$\frac{1}{2}n \leq n-1 \Leftrightarrow n \leq 2n-2 \Leftrightarrow 2 \leq n$$

Příklad 3.2.3. Nechť k je kladné celé číslo. Je

$$\sum_{1 \leq i \leq n} i^k \leq \sum_{1 \leq i \leq n} n^k = n \cdot n^k = n^{k+1}$$

Je tedy

$$\sum_{1 \leq i \leq n} i^k = \mathcal{O}(n^{k+1})$$

Příklad 3.2.4. Ukážeme, že $4n^3 \neq \mathcal{O}(n^2)$. Postupujme sporem, Předpokládejme, že $4n^3 = \mathcal{O}(n^2)$. Existují tedy reálná čísla n_0, c, $c > 0$, tak, že pro všechna $n \geq n_0$ je $|4n^3| \leq c|n^2|$, tj. $4n^3 \leq cn^2$, tj. $n \leq \frac{c}{4}$. Spor.

Při manipulaci s asymptotickou notací musíme být obezřetní. Například $n = \mathcal{O}(n^3)$ a $n^2 = \mathcal{O}(n^3)$, avšak nelze z toho učinit závěr, že $n = n^2$. Musíme si stále uvědomovat, že $\mathcal{O}(g(n))$ je množina funkcí. Pak jistě z informace $n \in \mathcal{O}(n^3)$ a $n^2 \in \mathcal{O}(n^3)$ neučiníme závěr $n = n^2$.

Při analýze složitosti se často stane, že odhady závisí na více než jednom parametru. Například složitost grafového algoritmu může záviset na počtu vrcholů a také na počtu hran. Abychom mohli pracovat s těmito případy, zobecňujeme $\mathcal{O}$-, Θ- a Ω- notace přirozeným způsobem. Například množinu $\mathcal{O}(g(m,n))$ definujeme takto:

$$f(m,n) \in \mathcal{O}(g(m,n))$$

právě tehdy, když existují reálná čísla n_0, c, $c > 0$, tak, že pro všechna $m \geq n_0$, $n \geq n_0$ je

$$|f(m,n)| \leq c|g(m,n)|$$

Poznámka 3.2.5. Jedno z hlavních využití $\mathcal{O}$-, Θ- a Ω- notací je při analýze složitosti algoritmů. Nechť například nějaký algoritmus vstup velikosti n zpracuje v čase $T(n)$. Pak zápis $T(n) = \Theta(f(n))$ znamená, že $f(n)$ je asymptoticky přesnou hranicí, zápis $T(n) = \Omega(f(n))$ znamená, že $f(n)$ je asymptotickou dolní hranicí a zápis $T(n) = \mathcal{O}(f(n))$ znamená, že $f(n)$ je asymptotickou horní hranicí.

3.3 Relace mezi asymptotickýni notacemi

Přímo z definic vyplývají následující vztahy mezi asymptotickými notacemi:

$$\begin{aligned} f(n) = \mathcal{O}(g(n)) &\Leftrightarrow g(n) = \Omega(f(n)) \\ f(n) = \Theta(g(n)) &\Leftrightarrow f(n) = \mathcal{O}(g(n)) \wedge f(n) = \Omega(g(n)) \\ f(n) = \Theta(g(n)) &\Leftrightarrow f(n) \asymp g(n) \end{aligned}$$

Jsou-li dány dvě funkce, nemusí být vždy jasné, jaká asymptotická relace mezi nimi platí. Následující věta obsahuje užitečné postačující podmínky.

Věta 3.3.1. *Jestliže $f(n) > 0$, $g(n) > 0$ pro všechna n, pak*

1. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = a \neq 0 \Rightarrow f(n) = \Theta(g(n))$
2. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f(n) = \mathcal{O}(g(n)) \wedge f(n) \neq \Theta(g(n))$
3. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \Rightarrow g(n) = \mathcal{O}(f(n)) \wedge g(n) \neq \Theta(f(n))$

Důkaz. Dokážeme jen část 1, další dvě části se dokáží obdobně. Nechť tedy $\lim_{n\to\infty} \frac{f(n)}{g(n)} = a \neq 0$. Protože $f(n) > 0$, $g(n) > 0$ pro všechna n, je $a \geq 0$. Navíc však $a \neq 0$, takže $a > 0$. Pro $\frac{a}{2}$ existuje n_0 tak, že pro všechna $n \geq n_0$ je

$$\left| \frac{f(n)}{g(n)} - a \right| < \frac{a}{2}, \text{ tj. } \frac{a}{2} < \frac{f(n)}{g(n)} < \frac{3a}{2}, \text{ tj. } \frac{a}{2} g(n) < f(n) < \frac{3a}{2} g(n)$$

Položíme $c_1 = \frac{a}{2}$, $c_2 = \frac{3a}{2}$. Jsou c_1, c_2, n_0 reálná čísla, $c_1 > 0$, $c_2 > 0$, pro všechna $n \geq n_0$ je $c_1 g(n) < f(n) < c_2 g(n)$. Závěr: $f(n) = \Theta(g(n))$.

Příklad 3.3.2. Pro libovolná kladná celá čísla k, l je $\lim_{n\to\infty} \frac{(\ln n)^k}{n^l} = 0$, takže $(\ln n)^k = \mathcal{O}(n^l)$, $(\ln n)^k \neq \Theta(n^l)$.

3.4 Manipulace s $\mathcal{O}$- notací

Existuje několik pravidel pro práci s $\mathcal{O}$- výrazy. která poměrně snadno plynou z definice $\mathcal{O}$-notace.

$$\begin{aligned}
n^m &= \mathcal{O}(n^{m'}) \text{ pokud } m \leq m' \\
f(n) &= \mathcal{O}(f(n)) \\
c\mathcal{O}(f(n)) &= \mathcal{O}(cf(n)) \\
\mathcal{O}(cf(n)) &= \mathcal{O}(f(n)) \\
\mathcal{O}(\mathcal{O}(f(n))) &= \mathcal{O}(f(n)) \\
\mathcal{O}(f(n)) + \mathcal{O}(g(n)) &= \mathcal{O}(\max\{|f(n)|, |g(n)|\}) \\
\mathcal{O}(f(n)) \cdot \mathcal{O}(g(n)) &= \mathcal{O}(f(n)g(n)) \\
\mathcal{O}(f(n)g(n)) &= f(n)\mathcal{O}(g(n)) \text{ pokud } f(n) \neq 0
\end{aligned}$$

Nezapomeňme, že $\mathcal{O}(h(n))$ je množina funkcí. Uvedená pravidla je třeba chápat následovně:

$$\begin{aligned}
n^m &\in \mathcal{O}(n^{m'}) \text{ pokud } m \leq m' \\
f(n) &\in \mathcal{O}(f(n)) \\
c\mathcal{O}(f(n)) &\subseteq \mathcal{O}(cf(n)) \\
\mathcal{O}(cf(n)) &\subseteq \mathcal{O}(f(n)) \\
\mathcal{O}(\mathcal{O}(f(n))) &\subseteq \mathcal{O}(f(n)) \\
\mathcal{O}(f(n)) + \mathcal{O}(g(n)) &\subseteq \mathcal{O}(\max\{|f(n)|, |g(n)|\}) \\
\mathcal{O}(f(n)) \cdot \mathcal{O}(g(n)) &\subseteq \mathcal{O}(f(n)g(n)) \\
\mathcal{O}(f(n)g(n)) &\subseteq f(n)\mathcal{O}(g(n)) \text{ pokud } f(n) \neq 0
\end{aligned}$$

K tomu ještě musíme dodat, že operace s množinami funkcí mají přirozenou interpretaci, například $\mathcal{O}(f(n)) + \mathcal{O}(g(n)) = \{h_1(n) + h_2(n) |\ h_1(n) \in \mathcal{O}(f(n)), h_2(n) \in \mathcal{O}(g(n))\}$.

Na ukázku zdůvodníme platnost vztahu

$$\mathcal{O}(f(n)) + \mathcal{O}(g(n)) = \mathcal{O}(\max\{|f(n)|, |g(n)|\})$$

Nechť $h_1(n) \in \mathcal{O}(f(n))$, $h_2(n) \in \mathcal{O}(g(n))$. Pak existují reálná čísla n_1, c_1, n_2, c_2, $c_1 > 0$, $c_2 > 0$, tak, že pro všechna n platí:

$$n \geq n_1 \Rightarrow |h_1(n)| \leq c_1|f(n)|,\ n \geq n_2 \Rightarrow |h_2(n)| \leq c_2|g(n)|$$

Chceme ukázat, že $h_1(n) + h_2(n) = \mathcal{O}(\max\{|f(n)|, |g(n)|\})$. Položme $n_0 = \max\{n_1, n_2\}$, $c = \max\{c_1, c_2\}$; n_0, c jsou reálná čísla, $c > 0$. Buď $n \geq n_0$. Pak

$$\begin{aligned} |h_1(n) + h_2(n)| &\leq |h_1(n)| + |h_2(n)| \\ &\leq c_1|f(n)| + c_2|g(n)| \\ &\leq c|f(n)| + c|g(n)| \\ &\leq c\max\{|f(n)|, |g(n)|\} + c\max\{|f(n)|, |g(n)|\} \\ &= 2c\max\{|f(n)|, |g(n)|\} \end{aligned}$$

Tedy: pro $n \geq n_0$ je $|h_1(n) + h_2(n)| \leq 2c|\max\{|f(n)|, |g(n)|\}|$; pak $h_1(n) + h_2(n) \in \mathcal{O}(\max\{|f(n)|, |g(n)|\})$.

Nyní příklad na využití výše uvedených pravidel. Nechť m je nezáporné celé číslo, a_i ($i = 0, \ldots, m$) jsou reálné konstanty. Pak

$$\sum_{0 \leq i \leq m} a_i n^{m-i} = \sum_{0 \leq i \leq m} a_i \mathcal{O}(n^m) = \sum_{0 \leq i \leq m} \mathcal{O}(n^m) = \mathcal{O}(n^m)$$

Pravidlo $\mathcal{O}(f(n)) + \mathcal{O}(g(n)) = \mathcal{O}(\max\{|f(n)|, |g(n)|\})$ se často používá při analýze algoritmů k tomu, abychom získali celkový $\mathcal{O}$-odhad časové složitosti algoritmu z odhadů složitosti jeho částí.

3.5 Asymptotika a rekurence

Předvedeme nyní obecnou metodu pro řešení rekurencí, které dostaneme při analýze algoritmů a systémů navržených na základě metody rozděl a panuj. Jsou to rekurence

$$T(n) = aT\left(\frac{n}{c}\right) + f(n)$$

kde a, c jsou celá čísla, $a \geq 1$, $c > 1$, a pro funkci $f(n)$ známe pouze její asymptotický odhad. Následující věta ukazuje, jak určit $T(n)$ v řadě případů.

Věta 3.5.1 *Nechť a, c jsou celá čísla, $a \geq 1$, $c > 1$. Nechť*

$$T(n) = aT\left(\frac{n}{c}\right) + f(n)$$

pro všechna dostatečně velká n. Nechť $\frac{n}{c}$ zde značí buď $\lfloor \frac{n}{c} \rfloor$ nebo $\lceil \frac{n}{c} \rceil$. Pak

1. *Jestliže $f(n) = \mathcal{O}(n^{(\log_c a)-\epsilon})$, $\epsilon > 0$, pak $T(n) = \Theta(n^{\log_c a})$.*
2. *Jestliže $f(n) = \Theta(n^{\log_c a})$, pak $T(n) = \Theta(n^{\log_c a} \cdot \log n)$.*

3. *Jestliže* $f(n) = \Omega(n^{(\log_c a)+\epsilon})$, $\epsilon > 0$, $af(\frac{n}{c}) \leq bf(n)$ *pro téměř všechna* n *a nějaké* $b < 1$, *pak* $T(n) = \Theta(f(n))$.

Připomeňme ještě:

Pro všechna dostatečně velká n znamená: pro všechna $n \geq n_0$, kde n_0 je nějaká reálná konstanta.

Pro téměř všechna n znamená: pro všechna n s vyjímkou konečného počtu.

Tato věta se někdy nazývá "master theorem". Důkaz této věty je velmi komplikovaný, nebudeme jej zde provádět. Zájemci jej mohou najít v knize [1].

Všimněte si, že není nutné znát funkci $f(n)$ přesně, abychom byli schopni určit $T(n)$ asymptoticky přesně. Pro mnoho složitých systémů je přesné určení fukce $f(n)$ často prakticky nemožné.

Příklad 3.5.2. Uvažme rekurenci $T(n) = 3T(\frac{n}{4}) + n \log n$. Je $a = 3$, $c = 4$, $f(n) = n \log n$. Položme $\epsilon = 1 - \log_4 3$; je $\epsilon > 0$. Dále, pro dostatečně velká n máme:

$$n \log n \geq n = n^{(\log_4 3)+\epsilon}$$

takže $f(n) = \Omega(n^{(\log_4 3)+\epsilon})$. Dále,

$$3f\left(\frac{n}{4}\right) = 3 \cdot \frac{n}{4} \log\left(\frac{n}{4}\right) = \frac{3}{4}n(\log n - \log 4) \leq \frac{3}{4}n \log n = \frac{3}{4}f(n)$$

pro všechna n. Na základě části 3 Věty 3.5.1. je $T(n) = \Theta(n \log n)$

Příklad 3.5.3. (**Násobení celých čísel.**) Uvažme následující algoritmus pro násobení kladných celých čísel navržený na základě metody rozděl a panuj. Nechť x a y jsou dvě kladná n-bitová celá čísla, kde n je sudé kladné celé číslo. Je

$$x = b_0 + b_1 2 + b_2 2^2 + \cdots + b_{n-1} 2^{n-1}$$

kde $b_0, b_1, \ldots, b_{n-1} \in \{0, 1\}$; obdobně pro y. Je tedy $x = x_1 2^{\frac{n}{2}} + x_2$, $y = y_1 2^{\frac{n}{2}} + y_2$, kde x_1, x_2, y_1, y_2 jsou $\frac{n}{2}$-bitová čísla. Pak

$$xy = x_1 y_1 2^n + (x_1 y_2 + x_2 y_1) 2^{\frac{n}{2}} + x_2 y_2$$

Zdá se, že pro výpočet xy budeme muset provést 4 násobení $\frac{n}{2}$-bitových čísel, 3 sčítání a 2 posuny (posunem rozumíme násobení mocninou základu poziční soustavy, v našem případě tedy násobení mocninou čísla 2). Existuje však jiná metoda pro výpočet čísel $x_1 y_1$, $x_1 y_2 + x_2 y_1$, $x_2 y_2$. Nejdříve vypočítáme $x_1 y_1$ a $x_2 y_2$, což vyžaduje 2 násobení. Pak vypočítáme

$$z_1 = (x_1 + x_2)(y_1 + y_2) = x_1 y_1 + x_1 y_2 + x_2 y_1 + x_2 y_2$$

což vyžaduje 1 násobení. Nakonec vypočítáme $z_1 - x_1 y_1 - x_2 y_2 = x_1 y_2 + x_2 y_1$, což vyžaduje pouze 2 odčítání.

Tudíž problém násobení dvou n-bitových kladných celých čísel lze redukovat na problém 3 násobení $\frac{n}{2}$-bitových celých čísel a několika sčítání, odčítání a posunů, přičemž sčítání, odčítání a posuny dohromady vyžadují čas úměrný n (tedy lineární). Metodu lze snadno upravit pro případ lichého n. V tomto algoritmu je $a = 3$, $c = 2$, $f(n) = dn$, d je kladné reálné číslo. Pak $f(n) = \mathcal{O}(n^{(\log_2 3)-\epsilon})$, kde $\epsilon = (\log_2 3) - 1 > 0$. Dle bodu 1 z Věty 3.5.1. je $T(n) = \Theta(n^{\log_2 3})$.

4 Prvočísla a kongruence

4.1 Euklidův algoritmus

Nechť n, m jsou celá čísla, $m > 0$. Pak lze provést **dělení čísla n číslem m se zbytkem**, tedy najít (určit) celá čísla q, r splňující

$$n = mq + r,\ 0 \leq r < m$$

Čísla q a r jsou učena jednoznačně. Číslo q se nazývá **podíl** a číslo r se nazývá **zbytek** po dělení čísla n číslem m.

Nechť nyní n, m jsou libovolná celá čísla. Pak klademe

$$n \bmod m = \begin{cases} n - m\left\lfloor \frac{n}{m} \right\rfloor & \text{pokud } m \neq 0 \\ n & \text{jinak} \end{cases}$$

Předpokládejme, že $m > 0$, $n = mq + r,\ 0 \leq r < m$ (q, r jsou celá čísla). Pak $n - m\left\lfloor \frac{n}{m} \right\rfloor = n - m\left\lfloor q + \frac{r}{m} \right\rfloor = n - mq = r$ (využili jsme fakt, že $0 \leq \frac{r}{m} < 1$), $n \bmod m = r$. Pro $m > 0$ je tedy $n \bmod m$ rovno zbytku po dělení čísla n číslem m. Tudíž také $0 \leq n \bmod m < m$.

Například

$$7 \bmod 5 = 2,\ -7 \bmod 5 = 3,\ 7 \bmod -5 = -3,\ -7 \bmod -5 = -2$$

S operací mod je úzce svázána relace /. Říkáme, že celé číslo m **dělí** celé číslo n (označení: m/n), pokud $n \bmod m = 0$, tedy

$$m/n \Leftrightarrow \exists q \in \mathbb{Z} : n = mq$$

Místo "m dělí n"také někdy říkáme "n je **násobkem** čísla m".

Snadno lze ukázat, že pro všechna celá čísla m, n a všechna $\alpha, \beta \in \{1, -1\}$ platí:

$$m/n \Leftrightarrow \alpha m/\beta n$$

Specielně

$$m/n \Leftrightarrow |m|/|n|$$

Nechť m, n jsou celá čísla. Platí:

$$m/n \wedge n \neq 0 \Rightarrow m \leq |n|$$

Zdůvodnění: Buďte m, n celá čísla, m/n, $n \neq 0$. Chceme: $m \leq |n|$. Z m/n máme $|m|/|n|$. Pak $|n| = |m|q$ pro nějaké celé číslo q. Jelikož $n \neq 0$, je $|n| > 0$ a tedy $|m| > 0$, $q > 0$. Pak $1 \leq q$, $|m| \cdot 1 \leq |m|q$, $|m| \leq |n|$. Je $m \leq |m|$, takže $m \leq |n|$.

Nechť m, n jsou celá čísla, $m \neq 0$ nebo $n \neq 0$. Pak **největší společná dělitel** čísel m, n (označení: $\gcd(m, n)$) definujeme takto:

$$\gcd(m, n) = \max\{d \in \mathbb{Z} |\ d/m \wedge d/n\}$$

Tato definice je korektní, protože množina $M = \{d \in \mathbb{Z} |\ d/m \wedge d/n\}$ splňuje $M \subseteq \mathbb{Z}$, $M \neq \emptyset$ (je $1 \in M$), M je shora omezená: Nechť například $m \neq 0$. Nechť $d \in M$. Pak d/m a tedy $d \leq |m|$. Všimněme si, že $\gcd(m, n) > 0$, jelikož $1 \in M$.

Nechť m, n jsou nenulová celá čísla. Pak **nejmenší společný násobek** čísel m, n (označení: $\mathrm{lcm}(m, n)$) definujeme takto:

$$\mathrm{lcm}(m, n) = \min\{k \in \mathbb{Z} |\ k > 0 \wedge m/k \wedge n/k\}$$

Tato definice je korektní, protože množina $M = \{k \in \mathbb{Z} |\ k > 0 \wedge m/k \wedge n/k\}$ splňuje $M \subseteq \mathbb{Z}^+$, $M \neq \emptyset$: Ukážeme, že $|mn| \in M$. Je $|mn| > 0$, jelikož $mn \neq 0$. Dále, $|mn| = mn$ nebo $|mn| = -mn$; z toho již vidíme, že $m/|mn|$, $n/|mn|$. Všimněme si, že $\mathrm{lcm}(m, n) \neq 0$, protože $M \subseteq \mathbb{Z}^+$.

K výpočtu $\gcd(m, n)$ pro celá čísla m, n, $0 \leq m < n$, můžeme použít následující, více než 2300 let starý rekurzivní algoritmus.

Algoritmus 4.1.1. (**Euklidův algoritmus**) *Pro celá čísla m, n, $0 \leq m < n$, máme*

$$\begin{aligned} \gcd(0, n) &= n \\ \gcd(m, n) &= \gcd(n \bmod m, m) \text{ pro } m > 0 \end{aligned}$$

Ukažme nejprve korektnost Euklidova algoritmu:

1. $\gcd(0, n) = n$: Jistě $n/0$, n/n. Nechť d je celé číslo, $d/0$, d/n. Chceme: $d \leq n$. Víme již, že $d \leq |n|$. Ovšem $|n| = n$.

2. $\gcd(m, n) = \gcd(n \bmod m, m)$ pro $m > 0$: Stačí ukázat, že

   $$\{d \in \mathbb{Z} |\ d/m \wedge d/n\} = \{d \in \mathbb{Z} |\ d/n \bmod m \wedge d/m\}$$

   Je $n \bmod m = n - m \left\lfloor \frac{n}{m} \right\rfloor$, takže chceme ukázat, že

   $$\{d \in \mathbb{Z} |\ d/m \wedge d/n\} = \{d \in \mathbb{Z} |\ d/n - m \left\lfloor \frac{n}{m} \right\rfloor \wedge d/m\}$$

   Je tedy třeba ukázat, že pro všechna celá čísla d platí:

   $$d/m \wedge d/n \Leftrightarrow d/n - m \left\lfloor \frac{n}{m} \right\rfloor \wedge d/m$$

   To je již snadné.

3. $0 \leq n \bmod m < m$ pro $m > 0$: To již víme. Během výpočtu tedy nedostaneme chybové hlášení, protože $(n \bmod m, m)$ je přípustná vstupní dvojice. Dále z uvedeného vztahu plyne, že výpočet skončí po konečně mnoha krocích, jelikož nová dvojice má první složku menší než dvojice (m, n).

Příklad 4.1.2. $\gcd(27, 36) = \gcd(9, 27) = \gcd(0, 9) = 9$, $\gcd(214, 352) = \gcd(138, 214) = \gcd(76, 138) = \gcd(62, 76) = \gcd(14, 62) = \gcd(6, 14) = \gcd(2, 6) = \gcd(0, 2) = 2$

Jsou-li m, n celá čísla, $0 \leq m < n$, pak pomocí Euklidova algoritmu můžeme vypočítat také celá čísla m' a n' splňující

$$m'm + n'n = \gcd(m, n)$$

To je jedna z nejdůležitějších aplikací Euklidova algoritmu.

Vskutku, pro $m = 0$ je $\gcd(m, n) = n$ a můžeme vzít $m' = 0$, $n' = 1$. Nechť $m > 0$ a m'', n'' jsou rekurzí určená celá čísla taková, že

$$m''(n \bmod m) + n''m = \gcd(n \bmod m, m)$$

Pak

$$\begin{aligned}
\gcd(m, n) &= \gcd(n \bmod m, m) \\
&= m''(n \bmod m) + n''m \\
&= m'' \left(n - m \left\lfloor \frac{n}{m} \right\rfloor\right) + n''m \\
&= m''n - m''m \left\lfloor \frac{n}{m} \right\rfloor + n''m \\
&= \left(n'' - m'' \left\lfloor \frac{n}{m} \right\rfloor\right) m + m''n
\end{aligned}$$

Položíme $m' = n'' - m'' \left\lfloor \frac{n}{m} \right\rfloor$, $n' = m''$ a máme $m'm + n'n = \gcd(m, n)$.

Jestliže Euklidův algoritmus použijeme k určení $\gcd(m, n)$ a také celých čísel m' a n' splňujících $m'm + n'n = \gcd(m, n)$, pak říkáme, že jsme použili **rozšířený Euklidův algoritmus**.

Příklad 4.1.3. $\gcd(57, 237) = \gcd(9, 57) = \gcd(3, 57) = \gcd(0, 3) = 3$. Průběh výpočtu můžeme zapsat následovně:

$$\begin{aligned}
237 &= 4 \cdot 57 + 9 \\
57 &= 6 \cdot 9 + 3 \\
9 &= 3 \cdot 3 + 0
\end{aligned}$$

Z toho dostaneme

$$3 = (-6)\cdot 9+1\cdot 57 = (-6)\cdot((-4)\cdot 57+1\cdot 237)+1\cdot 57 = 24\cdot 57+(-6)\cdot 237+1\cdot 57 = 25\cdot 57+(-6)\cdot 237$$

$$3 = 25 \cdot 57 + (-6) \cdot 237$$

Uvědomíme si, že Euklidovým algoritmem lze určit $\gcd(m, n)$ pro libovolná celá čísla m a n, $m \neq 0$ nebo $n \neq 0$. Pro $|m| = |n|$ ani žádný algoritmus nepotřebujeme, protože $\gcd(m, n) = |n|$ (to je snadné si rozmyslet). Nechť tedy $|m| \neq |n|$. Předpokládejme například, že $|m| < |n|$. Pak $0 \leq |m| < |n|$ a Euklidovým algoritmem lze určit $\gcd(|m|, |n|)$. Pro každé celé číslo d a všechna $\alpha, \beta \in \{1, -1\}$ platí

$$d/m \wedge d/n \Leftrightarrow d/\alpha m \wedge d/\beta n$$

z čehož ihned plyne, že $\gcd(m, n) = \gcd(\alpha m, \beta n)$ a specielně $\gcd(m, n) = \gcd(|m|, |n|)$.

A ještě si uvědomíme, že Euklidovým algoritmem lze pro libovolná celá čísla m a n, $m \neq$ nebo $n \neq 0$, určit celá čísla m', n' taková, že $m'm + n'n = \gcd(m, n)$. Pro $|m| = |n|$ ani žádný algoritmus nepotřebujeme, protože $\gcd(m, n) = |n| = 0 \cdot m + \alpha n$, kde $\alpha \in \{1, -1\}$. Nechť tedy $|m| \neq |n|$. Euklidovým algoritmem vypočteme celá čísla m'', n'' taková, že $\gcd(|m|, |n|) = m''|m| + n''|n|$. Ovšem $\gcd(|m|, |n|) = \gcd(m, n)$, $|m| = \alpha m$, $|n| = \beta n$, kde $\alpha, \beta \in \{1, -1\}$. Pak $\gcd(m, n) = m''\alpha m + n''\beta n$ a stačí vzít $m' = \alpha m''$, $n' = \beta n''$.

Jestliže m, n jsou celá čísla, $m \neq 0$ nebo $n \neq 0$, pak říkáme, že čísla m a n jsou **nesoudělná** (označení: $m \perp n$), pokud $\gcd(m, n) = 1$.

Tvrzení 4.1.4. *Nechť a, b, c jsou celá čísla. Nechť p je prvočíslo. Platí:*

1.
$$a/bc \wedge a \perp b \Rightarrow a/c$$

2.
$$a/c \wedge b/c \wedge a \perp b \Rightarrow ab/c$$

3.
$$p/bc \Rightarrow p/b \vee p/c$$

Důkaz.

1. Nechť a/bc, $a \perp b$. Chceme: a/c. Je $bc = aq$, kde q je celé číslo. Protože $a \perp b$, lze Euklidovým algoritmem určit celá čísla a', b' taková, že $1 = a'a+b'b$. Pak $c = c(a'a+b'b) = ca'a + cb'b = ca'a + bcb' = ca'a + aqb' = a(ca' + qb')$, $c = a(ca' + qb')$, a/c.

2. Nechť a/c, b/c, $a \perp b$. Chceme: ab/c. Je $c = aq$ pro nějaké celé číslo q. Pak b/aq, $a \perp b$. Dle již dokázané části 1 pak b/q. Takže $q = br$, kde r je celé číslo. Pak $c = abr$, ab/c.

3. Nechť p/bc. Chceme: $p/b \vee p/c$. Pokud p/b, jsme hotovi. Nechť tedy b není násobkem čísla p. Buď d kladné celé číslo, d/p, d/b. Protože p je prvočíslo, je $d = 1$ nebo $d = p$. Avšak nemůže být $d = p$, jelikož b není násobkem čísla p. Nutně tedy $d = 1$. Ukázali jsme: $p \perp b$. Dle již dokázané části 1 dostáváme p/c.

Jestliže m a n jsou celá čísla, $n > 0$, $m \perp n$, pak můžeme Euklidovým algoritmem určit celá čísla m', n' taková, že $m'm + n'n = 1$. Položme

$$m^{-1} \bmod n = m' \bmod n$$

Označme stručně $k = m^{-1} \bmod n$. Je $k = m' \bmod n = m' - n \left\lfloor \frac{m'}{n} \right\rfloor$, takže $m' = k + n \left\lfloor \frac{m'}{n} \right\rfloor$, $1 = \left(k + n \left\lfloor \frac{m'}{n} \right\rfloor\right) m + n'n = km + n \left\lfloor \frac{m'}{n} \right\rfloor m + n'n$, $1 - km = n \left(\left\lfloor \frac{m'}{n} \right\rfloor m + n' \right)$.
Platí tedy:

$$m^{-1} \bmod n \in \mathbb{Z},\ 0 \leq m^{-1} \bmod n < n,\ n/1 - (m^{-1} \bmod n)m$$

Poznámka: S využitím kongruencí, které zavedeme v části 4.3., lze místo $n/1 - (m^{-1} \bmod n)m$ psát $(m^{-1} \bmod n)m \equiv 1 \pmod{n}$.
Číslo $m^{-1} \bmod n$ se nazývá **multiplikativní inverze** čísla m modulo n.

Na závěr této části provedeme **analýzu složitosti Euklidova algoritmu**.

Pomocné tvrzení (1): *Nechť a, b jsou celá čísla, $0 < a < b$. Pak $b \bmod a < \frac{b}{2}$.*

Zdůvodnění: Rozlišíme 2 případy. (I) $a \leq \frac{b}{2}$ (II) $\frac{b}{2} < a$
ad (I): Stačí si uvědomit, že $b \bmod a < a$.
ad (II): Chceme: $b - a \left\lfloor \frac{b}{a} \right\rfloor < \frac{b}{2}$, tj. $\frac{b}{2} < a \left\lfloor \frac{b}{a} \right\rfloor$. Je $1 < \frac{b}{a}$, takže $1 \leq \left\lfloor \frac{b}{a} \right\rfloor$, $a \leq a \left\lfloor \frac{b}{a} \right\rfloor$. Protože $\frac{b}{2} < a$, je $\frac{b}{2} < a \left\lfloor \frac{b}{a} \right\rfloor$.

Nechť m, n jsou celá čísla, $0 < m < n$. Jeden krok Euklidova algoritmu znamená přechod od dvojice (m, n) ke dvojici (n_1, m), kde $n_1 = n \bmod m$. Dle Pomocného tvrzení (1) je $n_1 < \frac{n}{2}$. Předpokládejme, že $0 < n_1$. Pak máme $0 < n_1 < m$. Další krok Euklidova algoritmu od dvojice (n_1, m) přejde ke dvojici (m_1, n_1), kde $m_1 = m \bmod n_1$. Dle Pomocného tvrzení (1) je $m_1 < \frac{m}{2}$.
Zjistili jsme toto: Euklidův algoritmus po 2 krocích přejde od dvojice (m, n) ke dvojici (m_1, n_1), kde $m_1 < \frac{m}{2}$, $n_1 < \frac{n}{2}$.
Předpokládejme nyní, že m, n jsou celá čísla, $0 < m < n$. Předpokládejme ještě, že $n = 2^k$. Výpočet Euklidova algoritmu na vstupu (m, n) skončí, vzhledem k tomu, co jsme před chvílí zjistili, nejvýše po k dvojkrocích (protože $2^{k-k} = 2^0 = 1$), tedy nejvýše po $2k$ krocích. Počet rekurzivních kroků Euklidova algoritmu pro vstup (m, n) označme $T(n)$. Je tedy $T(n) \leq 2k = 2 \lg n$. Odvodili jsme tedy:
Jestliže $T(n)$ je počet rekurzivních kroků Euklidova algoritmu pro vstup (m, n), pak

$$T(n) \leq \mathcal{O}(\lg n)$$

Přesnější analýzu Euklidova algoritmu provedli Lucas a Lamé roku 1884, což byla pravděpodobně první hluboká analýza algoritmů.

Je poměrně silná souvislost mezi Euklidovým algoritmem a Fibonacciho čísly.

Pomocné tvrzení (2): *Nechť k, m, n jsou celá čísla, $k \geq 0$, $n > m > 0$. Jestliže $m \geq F_{k+1}$, $n \bmod m \geq F_k$, pak $n \geq F_{k+2}$.*

Zdůvodnění: Nejprve ukážeme, že $n \geq m + (n \bmod m)$. Je tedy třeba ukázat, že $n \geq m + n - m \left\lfloor \frac{n}{m} \right\rfloor$, tj. $m \left\lfloor \frac{n}{m} \right\rfloor \geq m$, tj. $\left\lfloor \frac{n}{m} \right\rfloor \geq 1$. To platí, protože $\frac{n}{m} > 1$. Máme tedy $n \geq m + (n \bmod m) \geq F_{k+1} + F_k = F_{k+2}$.

Věta 4.1.5.

1. *Jestliže m, n, k jsou celá čísla, $n > m > 0$, $k \geq 1$ a Euklidův algoritmus pro vstupní hodnoty m, n skončí po k rekurzivních krocích, pak $n \geq F_{k+2}$, $m \geq F_{k+1}$.*

2. *Jestliže m, n, k jsou celá čísla, $n > m > 0$, $k \geq 1$ a $m < F_{k+1}$, pak Euklidův algoritmus pro vstupní hodnoty m, n vyžaduje méně než k rekurzivních kroků.*

Důkaz.

1. Indukcí vzhledem ke k.

   $k = 1$: Nechť m, n jsou celá čísla, $n > m > 0$, a Euklidův algoritmus pro vstupní hodnoty m, n skončil po jednom rekurzivním kroku. Chceme: $n \geq F_3$, $m \geq F_2$, tj. $n \geq 2$, $m \geq 1$. To platí.

   $k > 1$: Nechť m, n jsou celá čísla, $n > m > 0$, a Euklidův algoritmus pro vstupní hodnoty m, n skončil po k rekurzivních krocích. Chceme: $n \geq F_{k+2}$, $m \geq F_{k+1}$. První krok výpočtu znamenal přechod od vstupní dvojice (m, n) ke dvojici $(n \bmod m, m)$. Potom výpočet skončil po $k - 1 > 0$ rekurzivních krocích. Také $m > n \bmod m > 0$. Dle indukčního předpokladu pak $m \geq F_{(k-1)+2}$, $n \bmod m \geq F_{(k-1)+1}$, tedy $m \geq F_{k+1}$, $n \bmod m \geq F_k$. Dle Pomocného tvrzení (2) pak $n \geq F_{k+2}$.

2. Nechť m, n, k jsou celá čísla, $n > m > 0$, $k \geq 1$ a $m < F_{k+1}$. Chceme: Euklidův algoritmus pro vstupní hodnoty m, n vyžaduje méně než k rekurzivních kroků. Předpokládejme, že Euklidův algoritmus skončí po l rekurzivních krocích, kde $l \geq k$. Podle části 1 je $m \geq F_{l+1}$. Ovšem $F_{l+1} \geq F_{k+1}$ (protože $k + 1 \geq l + 1$), což dává $m \geq F_{k+1}$, spor.

Například $F_{30} = 832040$, takže pro vstupní hodnoty m, n, kde $n > m > 0$ a $m < 832040$, vyžaduje Euklidův algoritmus méně než 30 rekurzivních kroků.

4.2 Prvočísla

Celé číslo p, $p > 1$, se nazývá **prvočíslo**, pokud má pouze dva kladné celočíselné dělitele, 1 a p; v opačném případě se nazývá **složené** číslo.

Zde jsou všechna prvočísla menší než 100:

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97$$

Prvočísla mají výsadní postavení mezi celými čísly (prvním dokladem tohoto faktu je následující věta). Prvočísla také hrají důležitou roli v počítačové vědě (kryptografie).

Věta 4.2.1. (**Základní věta aritmetiky**) *Každé celé číslo n, $n > 1$, má jednoznačný prvočíselný rozklad tvaru*

$$n = \prod_{1 \le i \le k} p_i^{e_i}$$

kde k je kladné celé číslo, p_i jsou prvočísla pro $i = 1, \ldots, k$, $p_i < p_{i+1}$ pro $i = 1, \ldots, k-1$, e_i jsou kladná celá čísla pro $i = 1, \ldots, k$.

Existuje nekonečně mnoho prvočísel. Dokonce existuje nekonečně mnoho prvočísel speciálního tvaru. Například platí

Věta 4.2.2. *Existuje nekonečně mnoho prvočísel tvaru $4k + 3$, kde k je celé číslo.*

Důkaz. Existuje aspoň jedno prvočíslo tvaru $4k + 3$, například 3. Předpokládejme, že existuje pouze konečně mnoho prvočísel tvaru $4k + 3$; nechť jsou to prvočísla $p_1, p_2, \ldots, p_n$ (n je kladné celé číslo). Položme $a = p_1 p_2 \ldots p_n$. Jistě a je kladné celé číslo. Uvažme číslo $4a - 1$. V prvočíselném rozkladu čísla $4a - 1$ se nevyskytuje číslo 2, jelikož číslo $4a - 1$ je liché. Předpokládejme, že v prvočíselném rozkladu čísla $4a - 1$ se vyskytuje prvočíslo tvaru $4k + 3$; nechť je to prvočíslo p_i (je $i \in \{1, \ldots, n\}$). Pak $4a - 1 = p_i q$, kde q je celé číslo. Pak $1 = 4p_1 \ldots p_i \ldots p_n - p_i q$. Vidíme, že $p_i / 1$. Pak $p_i \le 1$, spor. V prvočíselném rozkladu čísla $4a - 1$ se tedy vyskytují pouze prvočísla tvaru $4k + 1$. Součin několika takových čísel je číslo tvaru $4b + 1$, kde b je celé číslo. Pak $4a - 1 = 4b + 1$, $4(a - b) = 2$, $2(a - b) = 1$, 1 je sudé číslo, spor.

První známou metodu pro vyhledávání prvočísel navrhl Eratosthenes z Kyrény (276 – 194 př. n. l.).

Eratosthenovo síto:

Vstup: celé číslo n, $n \ge 2$

Výstup: množina P všech prvočísel menších nebo rovných číslu n

1. $P \leftarrow \emptyset$, $S \leftarrow \{2, 3, \ldots, n\}$
2. $p \leftarrow \min S$, $P \leftarrow P \cup \{p\}$, $S \leftarrow S - \{kp |\ k \in \mathbb{Z}^+\}$
3. Jestliže $S = \emptyset$, pak výpočet končí. Jinak jdi na krok číslo 2.

Největší známé prvočíslo v roce 2018 bylo $2^{82589933} - 1$. Má 24862048 číslic. Všechna známá velká prvočísla mají tvar $2^p - 1$, kde p je prvočíslo. Prvočísla tohoto tvaru se nazývají Mersennova prvočísla. Není známo, zda existuje nekonečně mnoho Mersennových prvočísel.

Další důležitou otázkou je, kolik prvočísel je mezi prvními n kladnými celými čísly; tento počet se značí $\pi(n)$. Základní odhad $\pi(n) = \Theta\left(\frac{n}{\ln n}\right)$ vyslovil již Gauss ve věku 15 let. Lepší odhad je

$$\frac{n}{\ln n} < \pi(n) \leq \frac{5}{4}\frac{n}{\ln n} \text{ pro } n \geq 114$$

Další informaci o rozložení prvočísel poskytuje následující věta, v níž ϕ je **Eulerova funkce**. Pro kladné celé číslo n je $\phi(n)$ rovno počtu všech celých čísel k takových, že $1 \leq k \leq n$ a $k \perp n$. Takže například pro prvočísla p, q, $p \neq q$, máme $\phi(p) = p - 1$ a $\phi(pq) = (p-1)(q-1)$.

Věta 4.2.3. (**Prvočíselná věta**) *Nechť b, c, n jsou celá čísla, $b \geq 1$, $n \geq 1$. Pak pro počet $\pi_{b,c}(n)$ všech prvočísel tvaru $bk + c$ menších nebo rovných číslu n platí*

$$\pi_{b,c}(n) \sim \frac{1}{\phi(b)}\frac{n}{\ln n}$$

Specielně, volbou $b = 1$, $c = 0$ dostaneme

$$\pi(n) \sim \frac{n}{\ln n}$$

Následující tabulka ukazuje, jak dobrý je odhad $\pi(n) = \frac{n}{\ln n}$.

n	10^1	10^2	10^3	10^4	10^5	10^7	10^{10}
$\pi(n)$	4	25	168	1229	9592	664579	455052511
$\frac{n}{\ln n}$	4	22	145	1086	8686	620421	434294482
$\frac{\pi(n)}{\frac{n}{\ln n}}$	0,921	1,151	1,160	1,131	1,104	1,071	1,047

(V tabulce jsou podíly $\frac{n}{\ln n}$ zaokrouhleny na celá čísla, u čísel $\frac{\pi(n)}{\frac{n}{\ln n}}$ jsou uvedeny první tři číslice za desetinnou čárkou.)

Důležitost prvočísel v kryptografii je dána tím, že umíme efektivně najít velká prvočísla, avšak nejsme schopni efektivně faktorizovat velké součiny prvočísel. Navíc, některé důležité výpočty lze provádět v polynomiálním čase, pokud argument je prvočíselný, ale zdají se být nezvládnutelné, pokud argumentem je libovolné celé číslo.

4.3 Aritmetika kongruencí

Pro libovolná celá čísla a, b, m, $m > 0$, klademe

$$a \equiv b \pmod{m} \Leftrightarrow a \bmod m = b \bmod m$$

Zápis $a \equiv b \pmod{m}$ čteme "a **je kongruentní s** b **modulo** m".

Předpokládejme, že $a \equiv b \pmod{m}$. Pak $a - m\left\lfloor \frac{a}{m} \right\rfloor = b - m\left\lfloor \frac{b}{m} \right\rfloor$, $a - b = m\left\lfloor \frac{a}{m} \right\rfloor - m\left\lfloor \frac{b}{m} \right\rfloor = m\left(\left\lfloor \frac{a}{m} \right\rfloor - \left\lfloor \frac{b}{m} \right\rfloor\right)$, $m/a - b$.

Předpokládejme naopak, že $m/a - b$. Pak $a - b = mq$, kde q je celé číslo. Pak $a \bmod m = a - m\left\lfloor \frac{a}{m} \right\rfloor = a - m\left\lfloor \frac{b}{m} + q \right\rfloor = a - m\left(\left\lfloor \frac{b}{m} \right\rfloor + q\right) = b + mq - m\left\lfloor \frac{b}{m} \right\rfloor - mq = b - m\left\lfloor \frac{b}{m} \right\rfloor = b \bmod m$.

Ukázali jsme, že platí

$$a \equiv b \pmod{m} \Leftrightarrow m/a - b$$

Nechť m je kladné celé číslo. Snadno lze ukázat, že pro všechna celá čísla a, b, c platí

$$a \equiv a \pmod{m}$$

$$a \equiv b \pmod{m} \Rightarrow b \equiv a \pmod{m}$$

$$a \equiv b \pmod{m} \wedge b \equiv c \pmod{m} \Rightarrow a \equiv c \pmod{m}$$

Je tedy relace kongruence modulo m ekvivalence na množině $\mathbb{Z}$. Tato ekvivalence indukuje rozklad množiny $\mathbb{Z}$, jehož třídy nazýváme **zbytkové třídy modulo** m a který značíme $\mathbb{Z}_m$. Je tedy $\mathbb{Z}_m$ množina všech zbytkových tříd modulo m.

Předpokládejme, že $a \equiv b \pmod{m}$, $a \perp m$. Ukážeme, že $b \perp m$. Nechť d je kladné celé číslo, d/b a d/m. Chceme: $d = 1$. Máme $m/a - b$, takže $a - b = mq$, kde q je celé číslo. Pak $a = b + mq$. Vidíme, že d/a (protože d/b, d/m). Takže d/a, d/m. Jelikož $a \perp m$, je $d = 1$.

Ukázali jsme: Jestliže nějaký prvek zbytkové třídy modulo m je nesoudělný s číslem m, pak všechny prvky té zbytkové třídy jsou nesoudělné s m. Symbol $\mathbb{Z}_m^*$ označuje množinu všech zbytkových tříd modulo m, jejichž prvky jsou nesoudělné s m. Je $\mathbb{Z}_m^* \subseteq \mathbb{Z}_m$.

S využitím definice kongruence a Tvrzení 4.1.4. lze snadno ukázat, že pro všechna celá čísla a, b, c, d, m, n, $m > 0$, $n > 0$, platí:

$$\begin{aligned}
a \equiv b \wedge c \equiv d \pmod{m} &\Rightarrow a + c \equiv b + d \pmod{m} \\
a \equiv b \wedge c \equiv d \pmod{m} &\Rightarrow a - c \equiv b - d \pmod{m} \\
a \equiv b \wedge c \equiv d \pmod{m} &\Rightarrow ac \equiv bd \pmod{m} \\
ad \equiv bd \pmod{m} &\Leftrightarrow a \equiv b \pmod{m} \text{ pro } d \perp m \\
ad \equiv bd \pmod{md} &\Leftrightarrow a \equiv b \pmod{m} \text{ pro } d > 0 \\
a \equiv b \pmod{mn} &\Leftrightarrow a \equiv b \pmod{m} \wedge a \equiv b \pmod{n} \text{ pokud } m \perp n
\end{aligned}$$

Vidíme, že kongrunce (se stejným modulem) lze sčítat, odčítat a násobit, obdobně jako rovnosti. Čtvrtá vlastnost říká, že násobení celým číslem nesoudělným s modulem je ekvivalentní úpravou kongruence.

Na ukázku dokážeme poslední vlastnost. Nechť a, b, m, n jsou celá čísla, $m > 0$, $n > 0$, $m \perp n$.

Předpokládejme, že $a \equiv b \pmod{mn}$. Chceme: $a \equiv b \pmod{m}$ a $a \equiv b \pmod{n}$. Máme $mn/a-b$. Takže $a-b = mnq$ pro nějaké celé číslo q. Vidíme, že $m/a-b$ a $n/a-b$. Tudíž $a \equiv b \pmod{m}$ a $a \equiv b \pmod{n}$.

Předpokládejme nyní, že $a \equiv b \pmod{m}$ a $a \equiv b \pmod{n}$. Chceme: $a \equiv b \pmod{mn}$. Víme: $m/a-b$, $n/a-b$, $m \perp n$. Dle Tvrzení 4.1.4. pak $mn/a-b$, a tedy $a \equiv b \pmod{mn}$.

Poslední vlastnost lze využít ke zjednodušení počítání s kongruencemi. Jestliže $\prod_{1 \le i \le k} p_i^{e_i}$ je prvočíselný rozklad kladného celého čísla m, pak

$$a \equiv b \pmod{m} \Leftrightarrow a \equiv b \pmod{p_i^{e_i}} \text{ pro } 1 \le i \le k$$

Kongruence modulo mocniny prvočísel jsou tedy stavebními kameny všech kongruencí modulo kladná celá čísla.

Věta 4.3.1. *Nechť c, d, m jsou celá čísla, $m > 0$. Jestliže číslo d není násobkem čísla $\gcd(c,m)$, pak lineární kongruence $cx \equiv d \pmod{m}$ nemá žádné řešení. Jestliže číslo d je násobkem čísla $\gcd(c,m)$, pak lineární kongruence $cx \equiv d \pmod{m}$ má přesně $k = \gcd(c,m)$ navzájem nekongruentních (modulo m) celočíselných řešení*

$$x_0\frac{d}{k}, x_0\frac{d}{k} + \frac{m}{k}, \ldots, x_0\frac{d}{k} + (k-1)\frac{m}{k}$$

kde x_0 je celočíselné řešení rovnice $cx + ym = \gcd(c,m)$, které může být získáno Euklidovým algoritmem.

Důkaz. Nechť číslo d není násobkem čísla $\gcd(c,m)$. Chceme: Lineární kongrunce $cx \equiv d \mod m$ nemá žádné řešení. Postupujme sporem. Předpoládejme, že celé číslo u je řešením kongruence $cx \equiv d \mod m$. Pak $cu \equiv d \mod m$, $m/cu-d$, $cu-d = mq$ pro nějaké celé číslo m. Pak $cu - mq = d$. Jelikož $\gcd(c,m)/c$ a $\gcd(c,m)/m$, máme $\gcd(c,m)/d$. To je spor. Nechť nyní číslo d je násobkem čísla $\gcd(c,m) = k$. Nechť x_0, y_0 jsou celá čísla, $cx_0 + y_0m = k$. Nyní je třeba dokázat následující tři věci:

1. Buř i celé číslo. Ukážeme, že $x_0\frac{d}{k}+i\frac{m}{k}$ je řešením kongruence $cx \equiv d \mod m$. Počítejme:

$$\begin{aligned} c(x_0\frac{d}{k}+i\frac{m}{k})-d &= cx_0\frac{d}{k}+y_0m\frac{d}{k}+ci\frac{m}{k}-y_0m\frac{d}{k}-d \\ &= (cx_0+y_0m)\frac{d}{k}+m(i\frac{c}{k}-y_0\frac{d}{k})-d \\ &= k\frac{d}{k}+m(i\frac{c}{k}-y_0\frac{d}{k})-d \\ &= d+m(i\frac{c}{k}-y_0\frac{d}{k})-d \\ &= m(i\frac{c}{k}-y_0\frac{d}{k}) \end{aligned}$$

   Máme

$$m \Big/ c\left(x_0\frac{d}{k}+i\frac{m}{k}\right)-d,\ c\left(x_0\frac{d}{k}+i\frac{m}{k}\right) \equiv d \mod m$$

2. Nechť i, j jsou celá čísla, $0 \le i < k$, $0 \le j < k$, $x_0\frac{d}{k}+i\frac{m}{k} \equiv x_0\frac{d}{k}+j\frac{m}{k} \pmod m$. Chceme: $i=j$. Je $0 \le i < k$, $-k < -j \le 0$, takže $-k < i-j < k$. Dále, $m/(x_0\frac{d}{k}+i\frac{m}{k})-(x_0\frac{d}{k}+j\frac{m}{k})$, $m/i\frac{m}{k}-j\frac{m}{k}$, $i\frac{m}{k}-j\frac{m}{k}=mq$ pro nějaké celé číslo q. Pak $im-jm=mqk$, $i-j=qk$. Celkem: $i-j$ je násobkem čísla k a $-k<i-j<k$. Z toho plyne, že $i-j=0$, $i=j$.

3. Nechť u je celé číslo, u je řešením kongruence $cx \equiv d \pmod m$. Chceme: $u \equiv x_0\frac{d}{k}+i\frac{m}{k} \pmod m$ pro nějaké $i \in \{0,1,\ldots,k-1\}$. Položme $v = x_0\frac{d}{k}$. Víme, že v je řešením kongruence $cx \equiv d \pmod m$. Pak $cu \equiv cv \pmod m$, $m/c(u-v)$, $k\frac{m}{k}/k\frac{c}{k}(u-v)$, $\frac{m}{k}/\frac{c}{k}(u-v)$. Je $k=\gcd(c,m)$. Z toho snadno plyne, že $\frac{m}{k} \perp \frac{c}{k}$. Dle Tvrzení 4.1.4. máme $\frac{m}{k}/u-v$. Pak $u-v=q\frac{m}{k}$ pro nějaké celé číslo q. Nechť $q=lk+i$, kde l, i jsou celá čísla, $0 \le i < k$. Pak $u-v=(lk+i)\frac{m}{k}=lm+i\frac{m}{k}$, $u-v-i\frac{m}{k}=lm$, $m/u-(v+i\frac{m}{k})$, $u \equiv v+i\frac{m}{k} \pmod m$, $u \equiv x_0\frac{d}{k}+i\frac{m}{k} \pmod m$; připomeňme ještě, že $i \in \{0,1,\ldots,k-1\}$.

Příklad 4.3.2. Vyřešíme lineární kongruenci $51x \equiv 9 \pmod{69}$. Je $c=51$, $d=9$, $m=69$. Euklidovým algoritmem vypočteme $k=\gcd(c,m)=\gcd(51,69)=\gcd(18,51)=\gcd(15,18)=\gcd(3,15)=\gcd(0,3)=3$. Platí: $\gcd(c,m)/d$, takže kongruence má přesně $k=3$ navzájem nekongruentních řešení. Pomocí Euklidova algoritmu najdeme celočíselné řešení rovnice $cx+ym=\gcd(c,m)$: $51\cdot(-4)+3\cdot 69=3$. Je tedy $x_0=-4$ a
$x_0\frac{d}{k}=-4\cdot\frac{9}{3}=-12$, $x_0\frac{d}{k}+\frac{m}{k}=-12+\frac{69}{3}=-12+23=11$, $x_0\frac{d}{k}+2\frac{m}{k}=-12+2\cdot 23=-12+46=34$.
Závěr: Lineární kongruence $51x \equiv 9 \pmod{69}$ má právě 3 navzájem nekongruentní řešení, a to $-12, 11, 34$; jestliže chceme řešení vyjádřit pouze kladnými celými čísly, pak můžeme vzít trojici

$$11, 34, 57$$

Existuje stará metoda pro řešení specielních systémů lineárních kongruencí. Následující výsledek je připisován čínskému matematikovi Sun Tsu (3. století).

Věta 4.3.3. (**Čínská věta o zbytcích**) *Nechť $m_1, \ldots, m_t$ jsou kladná celá čísla, $m_i \perp m_j$ pro všechna $i.j \in \{1, \ldots, t\}$, $i \neq j$, a $a_1, \ldots, a_t$ jsou celá čísla. Pak systém kongruencí*

$$x \equiv a_j \text{ pro } j = 1, \ldots, t$$

má řešení

$$x = \sum_{1 \leq i \leq t} a_i M_i N_i$$

kde $M = \prod_{1 \leq i \leq t} m_i$, $M_i = \frac{M}{m_i}$, a $N_i = M_i^{-1} \bmod m_i$, pro $i = 1, \ldots, t$; navíc uvedené řešení je jediné až na kongruenci modulo M, tj. $z \equiv x \pmod{M}$ pro každé další řešení z.

Důkaz. Zvolme $i, j \in \{1, \ldots, t\}$, $i \neq j$. Pak m_j / M_i, $m_j / a_i M_i N_i$, $a_i M_i N_i \equiv 0 \pmod{m_j}$. Dále, $M_j N_j \equiv 1 \pmod{m_j}$, takže $a_j M_j N_j \equiv a_j \pmod{m_j}$. Pak

$$x = \sum_{1 \leq i \leq t} a_i M_i N_i \equiv 0 + \cdots + 0 + a_j + 0 + \cdots + 0 = a_j \pmod{m_j}$$

Další část věty dokážeme pouze pro $t = 2$. Obecný případ se dokáže obdobně. Nechť tedy z je celé číslo, $z \equiv a_1 \pmod{m_1}$ a $z \equiv a_2 \pmod{m_2}$. Chceme: $z \equiv x \pmod{M}$ (zde $M = m_1 m_2$). Je $z \equiv x \pmod{m_1}$, $z \equiv x \pmod{m_2}$. Takže $m_1 / z - x$, $m_2 / z - x$. Ovšem $m_1 \perp m_2$. Podle Tvrzení 4.1.4. máme $m_1 m_2 / z - x$, $M / z - x$, $z \equiv x \pmod{M}$.

Čínská věta o zbytcích má řadu aplikací. Jednou z nich je následující modulární reprezentace čísel.

Nechť $m_1, \ldots, m_n$ jsou vzájemně nesoudělná kladná celá čísla. Nechť $M = m_1 \ldots m_n$. Pak každé celé číslo x, $0 \leq x < M$, má jednoznačnou reprezentaci

$$(x \bmod m_1, \ldots, x \bmod m_n)$$

Co přesně tím myslíme? Pro $i \in \{1, \ldots, n\}$ položme $A_i = \{0, \ldots, m_i - 1\}$. Dále položme $A = \{0, \ldots, M - 1\}$. Pak

$$x \mapsto (x \bmod m_1, \ldots, x \bmod m_n)$$

je vzájemně jednoznačné zobrazení (bijekce) množiny A na množinu $A_1 \times \cdots \times A_n$. Vskutku, buď $(a_1, \ldots, a_n) \in A_1 \times \cdots \times A_n$. Dle Čínské věty o zbytcích existuje celé číslo y, $y \equiv a_i \pmod{m_i}$ pro $i = 1, \ldots, n$. Položme $x = y \bmod M$. Pak $x \in A$. Je $x \equiv y \pmod{M}$, $M / x - y$. Pro $i = 1, \ldots, n$ máme m_i / M, takže $m_i / x - y$, $x \equiv y \pmod{m_i}$, $x \equiv a_i \pmod{m_i}$, $x \bmod m_i = a_i \bmod m_i$, $x \bmod m_i = a_i$. Je tedy $(x \bmod m_1, \ldots, x \bmod m_n) = (a_1, \ldots, a_n)$. Ještě nám zbývá

si uvědomit, že takové x je pouze jedno. Nechť tedy $z \in A$, $(z \bmod m_1, \ldots, z \bmod m_n) = (a_1, \ldots, a_n)$. Chceme $z = x$. Pro $i = 1, \ldots n$ máme $z \bmod m_i = a_i$, $z \bmod m_i = a_i \bmod m_i$, $z \equiv a_i \pmod{m_i}$. Podle Čínské zbytkové věty je $z \equiv x \pmod{M}$. Ovšem $0 \leq x < M$, $0 \leq z < M$, a proto $z = x$.

Uveďme příklad. Vezměme $m_1 = 2$, $m_2 = 3$, $m_3 = 5$. Pak 27 má reprezentaci $(1, 0, 2)$.

V kryptografických aplikacích je často potřeba **modulární umocňování**, tedy určit $a^b \bmod n$, kde a, b a n mohou být velmi velká celá čísla. Efektivní modulární umocňování je také potřebné pro efektivní testování prvočíselnosti. Pro zjednodušení modulárního umocňování je možné využít následující větu a její zobecnění. Oba výsledky také hrají důležitou roli v kryptografii.

Věta 4.3.4. (**Malá Fermatova věta, 1640**) *Jestliže p je prvočíslo a a je celé číslo, pak*

$$a^p \equiv a \pmod{p}$$

a pokud a není dělitelné číslem p, pak

$$a^{p-1} \equiv 1 \pmod{p}$$

Důkaz. Nejdříve vyřešíme případ $p = 2$. Chceme: $a^2 \equiv a \pmod 2$, tedy $2/a^2 - a$, $2/a(a-1)$; to platí, protože jedno z čísel a, $a - 1$ je sudé.
Pomocné tvrzení: Nechť k je celé číslo, $0 < k < p$. Pak $p/\binom{p}{k}$.
Zdůvodnění Pomocného tvrzení: Je $\binom{p}{k} = \frac{p(p-1)\ldots(p-k+1)}{k!}$, $\binom{p}{k} k(k-1)\ldots 1 = p(p-1)\ldots(p-k+1)$. Vidíme, že $p/\binom{p}{k} k(k-1)\ldots 1$. Protože p je prvočíslo, p dělí některé z čísel $1,\ldots,k-1$, k, $\binom{p}{k}$ (viz Tvrzení 4.1.4.). Ovšem $0 < 1 < \cdots < k - 1 < k < p$, takže $p/\binom{p}{k}$.
Konec zdůvodnění Pomocného tvrzení.
Nyní se zabývejme případem, kdy p je liché prvočíslo. Dokážeme nejprve indukcí vzhledem k a, že $a^p \equiv a \pmod p$ pro všechna nezáporná celá čísla a.
$a = 0$: Chceme: $0^p \equiv 0 \pmod p$, tj. $0 \equiv 0 \pmod p$. To platí.
$a \geq 0$. Indukční předpoklad: $a^p \equiv a \pmod p$. Chceme: $(a+1)^p \equiv a+1 \pmod p$. Připomeňme,

že pro všechna celá čísla k, $0 < k < p$, je $\binom{p}{k} \equiv 0 \pmod{p}$ (viz Pomocné tvrzení). Počítejme:

$$\begin{aligned}
(a+1)^p &= \sum_{0\le k\le p} \binom{p}{k} a^k \\
&= \binom{p}{0} a^0 + \binom{p}{p} a^p + \sum_{0<k<p} \binom{p}{k} a^k \\
&= 1 + a^p + \sum_{0<k<p} \binom{p}{k} a^k \\
&\equiv 1 + a^p + \sum_{0<k<p} 0 \cdot a^k \\
&= a^p + 1 + \sum_{0<k<p} 0 \\
&= a^p + 1 + 0 \\
&= a^p + 1 \\
&\equiv a + 1 \pmod{p}
\end{aligned}$$

Vezměme nyní celé číslo a, $a < 0$. Pak $-a > 0$ a dle již dokázaného je $(-a)^p \equiv -a \pmod{p}$, $-a^p \equiv -a \pmod{p}$, $a^p \equiv a \pmod{p}$.
Ve zbytku důkazu už bude p značit libovolné prvočíslo. Zbývá ještě dokázat toto: Pokud a je celé číslo, které není dělitelné prvočíslem p, pak $a^{p-1} \equiv 1 \pmod{p}$. Nechť tedy a je celé číslo, které není dělitelné číslem p. Pak $a \perp p$. Víme již, že $a \cdot a^{p-1} \equiv a \cdot 1 \pmod{p}$, takže $a^{p-1} \equiv 1 \pmod{p}$.

Ještě obecnější je **Eulerova věta**:

$$a^{\phi(m)} \equiv 1 \pmod{m} \text{ pokud } a \perp m$$

(a, m jsou celá čísla, $m > 0$)

Příklad 4.3.5. Vypočítáme $3^{1000} \bmod 19$. Číslo 19 je prvočíslo a 3 není násobkem čísla 19, takže dle Malé Fermatovy věty je $3^{18} \equiv 1 \pmod{19}$. Je $1000 = 18 \cdot 55 + 10$, takže

$$3^{1000} = (3^{18})^{55} \cdot 3^{10} \equiv 1^{55} \cdot 3^{10} = 3^{10} = (3^4)^2 \cdot 3^2 \equiv 5^2 \cdot 9 \equiv 6 \cdot 9 = 54 \equiv 16 \pmod{19}$$

Je tedy $3^{1000} \equiv 16 \pmod{19}$, takže $3^{1000} \bmod 19 = 16$.

5 Diskrétní odmocniny a logaritmy

5.1 Diskrétní odmocniny

Problém řešení kvadratických kongruencí

$$x^2 \equiv a \pmod{n}$$

neboli, jinými slovy, počítání diskrétních odmocnin modulo n, je velmi zajímavý a má mnoho aplikací.

Do této kapitoly patří mimo jiné témata: kvadratické zbytky, Legendreův symbol, Jacobiho symbol.

Látka této části je rozšiřující, budeme se jí věnovat jen v případě dostatku času. Výklad tedy bude ústní a na tabuli. Zájemci mohou najít poučení o této problematice například v knize [3], kapitola 1.8.1: Discrete Square Roots.

Kongruencím $x^2 \equiv a \pmod{n}$ se také věnují důkladně mnohé učebnice teorie čísel, například [5], kapitola 5: Quadratic Reciprocity. Ostatně, tuto knihu lze doporučit všem zájemcům o teorii čísel.

5.2 Diskrétní logaritmus

Jsou dána celá čísla a, b, n, $n > 1$. Chceme určit nezáporné celé číslo x takové, že $b^x \equiv a \pmod{n}$, pokud takové x existuje.

Může se stát, že takových x existuje víc, například $x = 4$ a $x = 10$ pro kongruenci $5^x \equiv 16 \pmod{21}$, nebo naopak žádné, například pro kongruenci $5^x \equiv 3 \pmod{21}$. (Předpokládejme, že existuje nezáporné celé číslo m splňující $5^m \equiv 3 \pmod{21}$. Pak $21/5^m - 3$, $5^m - 3 = 21q$ pro nějaké celé číslo q. Pak $5^m = 3 + 21q$, $5^m = 3 \cdot (1 + 7q)$, $3/5^m$. Jsou dvě možnosti: $m = 0$ nebo $m > 0$. První případ dává $3/1$, spor. Druhý případ dává $3/5$, spor.)

Důležitým případem je situace, kdy existuje tzv. **primitivní kořen modulo** n. Co to je? Jestliže g je celé číslo, $g \perp n$, pak dle Eulerovy věty (viz část 4.3) je $g^{\phi(n)} \equiv 1 \pmod{n}$. Pokud navíc $\phi(n)$ je nejmenší ze všech kladných celých čísel k splňujících $g^k \equiv 1 \pmod{n}$, pak je g primitivní kořen modulo n.

Připomeňme, že pro celá čísla c, d platí: Jestliže $c \equiv d \pmod{n}$, $c \perp n$, pak $d \perp n$. To jsme dokázali v části 4.3.

Dále, jestliže $c \perp n$, $d \perp n$, pak $cd \perp n$. Zdůvodnění: Předpokládejme, že $c \perp n$, $d \perp n$. Chceme: $cd \perp n$. Předpokládejme naopak, že čísla cd, n nejsou nesoudělná. Existuje tedy prvočíslo p, p/cd, p/n. Protože p je prvočíslo, máme p/c nebo p/d. Pokud p/c, jsou čísla c, n soudělná (víme, že p/n), spor. Obdobně dostaneme spor v případě p/d. Nutně tedy $cd \perp n$.

Nechť $P = \{k \in \mathbb{Z} |\ 0 \leq k < n,\ k \perp n\}$. Uvědomme si, že množina P má $\phi(n)$ prvků.

Nechť tedy g je primitivní kořen modulo n. Pak $P = \{g^0 \bmod n, g^1 \bmod n, \ldots, g^{\phi(n)-1} \bmod n\}$. Zdůvodnění:

- $\{g^0 \bmod n, g^1 \bmod n, \ldots, g^{\phi(n)-1} \bmod n\} \subseteq P$: Stačí ukázat, že pro každé nezáporné celé číslo i je $g^i \bmod n \in P$. Jistě $0 \leq g^i \bmod n < n$. Ještě musíme ukázat, že $g^i \bmod n \perp n$. Ovšem $g \perp n$, takže $g^i \perp n$ (součin čísel nesoudělných s n je číslo nesoudělné s n). Dále, $g^i \bmod n \equiv g^i \pmod{n}$, takže $g^i \bmod n \perp n$.

- $\{g^0 \bmod n, g^1 \bmod n, \ldots, g^{\phi(n)-1} \bmod n\} = P$: Víme, že $|P| = \phi(n)$. Stačí tudíž ukázat, že pro celá čísla i, j, $0 \leq i < j < \phi(n)$, je $g^i \bmod n \neq g^j \bmod n$. Předpokládejme naopak, že $g^i \bmod n = g^j \bmod n$. Pak $g^i \equiv g^j \pmod{n}$. Pak ovšem $1 \equiv g^{j-i} \pmod{n}$. Je $0 < j - i < \phi(n)$. To je spor.

Předpokládejme, že a je celé číslo, $a \perp n$. Pak kongruence

$$g^x \equiv a \pmod{n}$$

má právě jedno řešení v množině $\{0, 1, \ldots, \phi(n) - 1\}$. Je totiž $a \bmod n \in P$, takže $a \bmod n = g^i \bmod n$ pro nějaké $i \in \{0, 1, \ldots, \phi(n) - 1\}$; je $g^i \equiv a \pmod{n}$. Víme již, že pro $j \in \{0, 1, \ldots, \phi(n) - 1\}$, $j \neq i$, je $g^i \bmod n \neq g^j \bmod n$, takže nemůže být $g^j \equiv a \pmod{n}$.

Konečně se dostáváme k diskrétnímu logaritmu. Nechť g je primitivní kořen modulo n. Nechť a je celé číslo nesoudělné s n. Pak existuje přesně jedno celé číslo l splňující

$$0 \leq l < \phi(n) \wedge g^l \equiv a \pmod{n}$$

A toto číslo l se nazývá **diskrétní logaritmus** o základu g z čísla a (modulo n) a značí se $\log_g a$ (případně $\mathrm{dlog}_g a$). Jiný název, obvyklý v teorii čísel, je **index** čísla a vzhledem ke g (modulo n), označení: $index_{n,g}(a)$ nebo $\mathrm{ind}_g(a)$.

Již C. F. Gauss věděl, že primitivní kořen modulo n existuje právě tehdy, když n je některé z čísel 2, 4, p^i, $2p^i$, kde p je liché prvočíslo a i je kladné celé číslo.

Příklad 5.2.1. Uvedeme tabulku diskrétních logaritmů o základu 2 modulo 13.

a	1	2	3	4	5	6	7	8	9	10	11	12
$\log_2 a$	0	1	4	2	9	5	11	3	8	10	7	6

Problémem diskrétního logaritmu se nazývá následující úkol: Je dáno n (celé číslo větší než 1), g (primitivni kořen modulo n), a (celé číslo nesoudělné s n). Najdi celé číslo x, $0 \leq x < \phi(n)$, takové, že $g^x \equiv a \pmod{n}$ (tj. najdi $\log_g a$).

Není znám žádný efektivní deterministický algoritmus, který řeší problém diskrétního logaritmu. Tento fakt hraje důležitou roli v kryptografii a jejích aplikacích.

Dosud jsme hovořili především řečí teorie čísel. Na diskrétní logaritmus můžeme také nahlížet z hlediska algebry.

Připomeňme, že symbol $\mathbb{Z}_n^*$ označuje množinu všech zbytkových tříd modulo n, jejichž prvky jsou nesoudělné s n (viz část 4.3). Pro celé číslo c symbolem $[c]_n$ označme zbytkovou třídu

modulo n, v níž leží číslo c, tedy $[c]_n = \{d \in \mathbb{Z} |\ d \equiv c \pmod{n}\}$. Pak $\mathbb{Z}_n^* = \{[k]_n |\ k \in \mathbb{Z}, 0 \leq k < n, k \perp n\}$.

Víme, že pro celá čísla c, d, $c \perp n$, $d \perp n$, je $cd \perp n$. Předpis

$$[c]_n \cdot [d]_n = [cd]_n$$

korektně definuje operaci na množině $\mathbb{Z}_n^*$. To lze snadno dokázat. Také lze snadno ukázat, že $\mathbb{Z}_n^*$ s operací $\cdot$ je komutativní grupa, a to grupa řádu $\phi(n)$.

Celé číslo g, $g \perp n$, je primitivní kořen modulo n právě tehdy, když $[g]_n$ je **generátor** grupy $\mathbb{Z}_n^*$. Takže primitivní kořen modulo n existuje právě tehdy, když grupa $\mathbb{Z}_n^*$ je cyklická (a to je právě tehdy, když n je některé z čísel 2, 4, p^i, $2p^i$, kde p je liché prvočíslo a i je kladné celé číslo).

Pokud $[g]_n$ je generátor grupy $\mathbb{Z}_n^*$, je $\mathbb{Z}_n^* = \{[g]_n^0, [g]_n^1, \ldots, [g]_n^{\phi(n)-1}\}$ a pro celé číslo a, $a \perp n$, určit diskrétní logaritmus o základu g z čísla a znamená najít $k \in \{0, 1, \ldots, \phi(n) - 1\}$ splňující $[g]_n^k = [a]_n$.

6 Grafy

Ideje, metody a poznatky teorie grafů a grafové algoritmy mají rozsáhlé využití v řadě oblastí informatiky a je tedy třeba seznámit se také se základy teorie grafů.

Základní informace o grafech (především z hlediska využití v informatice) lze najít v knize [3], kapitola 2.4: Graphs.

Grafům, především stromům, se věnuje také Donald E. Knuth v knize [7] v části 2.3: Stromy. Nahlíží na ně především z hlediska programátora – část o stromech (jako důležitém typu nelineárních struktur) je zařazena v kapitole 2: Informační struktury. Nevyhýbá se však ani přístupu matematickému (část 2.3.4: Základní matematické vlastnosti stromů).

Knih (skript, učebnic,...) o teorii grafů je velké množství – v češtině a, samozřejmě, hlavně v angličtině.

Jiří Matoušek a Jaroslav Nešetřil jsou autory výborné knihy [9], jejíž značná část je věnována teorii grafů – jedná se o kapitoly 3: Grafy: úvod, 4: Stromy, 5: Rovinné kreslení grafů, 7: Počet koster. Tuto knihu lze jednoznačně doporučit všem, kteří se zajímají o teorii grafů a vůbec o diskrétní matematiku.

Doporučit lze také novou knihu [10].

Řada textů o teorii grafů je také volně přístupná na internetu, a to opět jak v češtině, tak hlavně v angličtině. Tyto texty si jistě dokážete vyhledat sami. Doporučím jen dva texty, a to výukový text [4] Petra Hliněného z Fakulty informatiky Masarykovy univerzity v Brně; text je primárně určen pro potřeby studentů informatických oborů na vysokých školách (je ale dobře přístupný i ne-informatikům). Doporučit mohu také text [8].

6.1 Základní ideje

Tato část má hlavně prezentovat základní pojmy teorie grafů (a také některá základní tvrzení). Zde je jejich seznam:

orientovaný a neorientovaný graf, vrcholy, hrany, incidence, stupně, sledy, tahy, cesty, cykly, souvislé komponenty, silně souvislé komponenty, vrcholová souvislost, hranová souvislost, isomorfismus, regularita, vrcholově symetrický graf, hranové symetrický graf, podgrafy, úplné grafy, bipartitní grafy, stromy, kostry grafu, rovinné grafy, Kuratowského věta, průměr grafu, multigrafy, hypergrafy.

Veškerou uvedenou látku ve stručném (avšak postačujícím) rozsahu najdete v knize [3], část 2.4.1: Basic Concepts.

Jestliže chcete využít česky psanou a snadno dostupnou literaturu, pak doporučuji výukový text [4], a to následující části: 1.1: Definice grafu, 1.2: Stupně vrcholů v grafu, 1.3: Podgrafy a isomorfismus, 1.4: Orientované grafy a multigrafy, 2.1: Spojení vrcholů, komponenty, 2.3: Vyšší stupně souvislosti, 3.1: Vzdálenost v grafu, 8.1: Rovinné kreslení grafu, 8.2: Eulerův vztah o počtu stěn, 8.3: Rozpoznávání rovinných grafů. Projdete-li důkladně uvedené kapitoly, naučíte se o dost víc, než je nezbytně potřeba. Z toho vyplývá, že je třeba se zaměřit na pochopení definovaných pojmů – totiž těch, které jsou v seznamu uvedeném výše. Z tvrzení se zaměřte především na věty 1.3, 1.4, 8.2 a její důsledky a na větu 8.8 (Kuratowského věta). Na konci každé části najdete úlohy k řešení. A na konci výkového textu je pak část IV: Klíč k řešení úloh.

Můžete také v knize [9] projít části 3.1: Pojem grafu; isomorfismus, 3.2: Podgrafy, souvislost, metrika a matice sousednosti, 3.4: Skóre grafu, 5.1: Kreslení do roviny a na další plochy, 5.3: Eulerův vztah. Neseznámíte se ovšem se všemi pojmy z našeho seznamu, na druhé straně však text jde více do hloubky. Opět platí, zaměřte se především na pochopení definovaných pojmů; z tvrzení a vět se zaměřte především na 3.4.1 (a důsledek), 3.4.3, 5.3.1, 5.3.2 a 5.3.3. Na konci každé části najdete cvičení, z nichž však některá jsou obtížnější či obtížná. Na konci knihy jsou pak návody ke cvičením (jen některým).

6.2 Reprezentace grafů

Nechť $G = (V, E)$ je graf. Připomeňme, že V je neprázdná množina, jejíž prvky se nazývají vrcholy (uzly) grafu. Graf je konečný, pokud množina V je konečná. Graf je nekonečný, pokud množina V je nekonečná. V případě orientovaného grafu je $E \subseteq V \times V$. V případě neorientovaného grafu je $E \subseteq \binom{V}{2}$, kde $\binom{V}{2}$ je množina všech dvouprvkových podmnožin množiny V. Prvky množiny E se nazývají hrany grafu. Jestliže G je orientovaný graf a $(u, v) \in E$, pak říkáme, že vrcholy u, v jsou incidentní s hranou (u, v), vrchol u je začátek hrany (u, v) a vrchol v je konec hrany (u, v), (u, v) je hrana z u do v.

Existují tři základní metody explicitní reprezentace konečného grafu $G = (V, E)$.

1. **Seznamy sousedů**

- pro orientovaný graf: Pro každý vrchol u je dán seznam $L[u]$ všech vrcholů v takových, že $(u, v) \in E$.
- pro neorientovaný graf: Pro každý vrchol u je dán seznam $L[u]$ všech vrcholů v takových, že $\{u, v\} \in E$.

2. **Matice sousednosti**

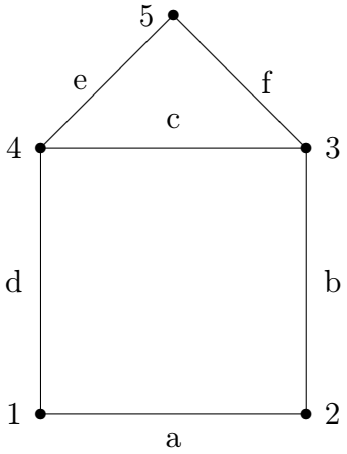
- pro orientovaný graf: Booleovská matice typu $|V| \times |V|$ s řádky a sloupci označenými vrcholy grafu G a s 1 na pozici (u, v) právě tehdy, když $(u, v) \in E$.
- pro neorientovaný graf: Booleovská matice typu $|V| \times |V|$ s řádky a sloupci označenými vrcholy grafu G a s 1 na pozici (u, v) právě tehdy, když $\{u, v\} \in E$.

3. **Matice incidence**

- pro orientovaný graf: Matice typu $|V| \times |E|$ s řádky označenými vrcholy grafu G a sloupci označenými hranami grafu G a s -1 na pozici (u, e), pokud u je začátek hrany e, s 1 na pozici (u, e), pokud u je konec hrany e a s 0 na pozici (u, e), pokud vrchol u není incidentní s hranou e.
- pro neorientovaný graf: Booleovská matice typu $|V| \times |E|$ s řádky označenými vrcholy grafu G a sloupci označenými hranami grafu G a s 1 na pozici (u, e) právě tehdy, když $u \in e$.

Příklad 6.2.1. Je dán neorientovaný graf G (viz Obrázek 2). Sestrojíme seznamy sousedů vrcholů grafu G, matici sousednosti grafu G (označíme ji A) a matici incidence grafu G (označíme ji I). Vrcholy uspořádáme v pořadí 1,2,3,4,5 a hrany v pořadí a, b, c, d, e, f.

Obrázek 2



$$L[1] = [2, 4],\ L[2] = [1, 3],\ L[3] = [2, 4, 5],\ L[4] = [1, 3, 5],\ L[5] = [3, 4]$$

$$A = \begin{pmatrix} 01010 \\ 10100 \\ 01011 \\ 10101 \\ 00110 \end{pmatrix}, \; I = \begin{pmatrix} 100100 \\ 110000 \\ 011001 \\ 001110 \\ 000011 \end{pmatrix}$$

Konečný graf $G = (V, E)$ může být určen seznamy sousedů celkové velikosti $\Theta(|E|)$, maticí sousednosti velikosti $\Theta(|V|^2)$ a maticí incidence velikosti $\Theta(|V| \cdot |E|)$. Seznamy sousedů jsou tudíž obecně nejvíc ekonomické reprezentace grafů.

Žádná z uvedených metod nemůže být použita pro popis nekonečných grafů (nebo i velmi velkých grafů). V takových případech je nutno použít jiné metody popisu grafů, například množina vrcholů či hran je specifikována nějakou formulí. Uvedeme dva příklady:

1. Orientovaný graf $G = (\mathbb{Z}, E)$, kde $E = \{(x, y) \in \mathbb{Z} \times \mathbb{Z} |\ x/y\}$; vrcholy grafu G jsou celá čísla, z x do y vede hrana právě tehdy, když číslo y je násobkem čísla x.

2. Nechť $\mathcal{G} = (C, \cdot, ^{-1}, 1)$ je grupa, $T \subseteq C$. Množina T se nazývá symetrická podmnožina grupy $\mathcal{G}$, pokud $1 \notin T$ a $g \in T \Rightarrow g^{-1} \in T$ (pro všechna $g \in T$). Cayleyho graf $G(\mathcal{G}, T)$ grupy $\mathcal{G}$ a její symetrické podmnožiny T je definován takto: $G(\mathcal{G}, T) = (C, E)$, kde $E = \{\{u, v\} \in \binom{C}{2} |\ \exists g \in T, ug = v\}$; $G(\mathcal{G}, T)$ je neorientovaný graf.

6.3 Párování a barvení

Do této části patří následující pojmy a věty:

párování, perfektní párování, Hallova věta, Tutteova věta, hranové barvení, Vizingova věta, vrcholové barvení, chromatické číslo.

Z anglicky psané literatury doporučuji knihu [3], část 2.4.3: Matchings and Colourings. Obsahuje dosti stručný, avšak pro naše účely postačující přehled.

Kde lze načerpat informace ve snadno dostupné a česky psané literatuře?

Tématu párování se věnuje text [8] v kapitole 6: Párování a pokrytí. Nás zajímají především části 6.1: Párování, 6.2: Párování v bipartitních grafech a 6.4: Úplné párování v obecných grafech. Věnujte se definicím, a to především párování v grafu (strana 65), perfektního párování (strana 66) a okolí množiny vrcholů (strana 67). Dále se věnujte větám 6.2 (Hallova věta) a 6.5 (Tutteova věta; důkaz můžete vynechat). Na konci každé podkapitoly najdete cvičení.

Základní poučení o barvení grafů doporučuji načerpat ve výukovém textu [4]. Jde především o to seznámit se s pojmy. V části 7.1: Barevnost grafu jsou definovány obarvení (vrcholové barvení) grafu a barevnost (chromatické číslo) grafu. V části 7.2: Variace na barevnost a jiné najdete definici hranového barvení a hranové barevnosti grafu. Seznamte se také s větou 7.11 (Vizingova věta; uvedena bez důkazu).

V textu [4] v části 8.4: Barvení map a rovinných grafů nebo v knize [9] v části 5.4: Barevnost mapy – problém 4 barev si přečtěte o slavném problému čtyř barev. Kdo má zájem nejen o matematické poznatky, ale také o jejich zdůvodnění, najde v [9] větu o 5 barvách včetně jejího důkazu (tvrzení 5.4.3).

6.4 Cestování grafem

Grafy jsou matematické objekty. V aplikacích vrcholy reprezentují procesy, procesory, města, firmy. Hrany reprezentují komunikační spoje, silnice. Řada aplikací a grafových algoritmů vyžaduje procházet graf nějakým systematickým a efektivním způsobem tak, abychom navštívili všechny vrcholy nebo všechny hrany. Existují různé postupy jak to udělat.

Přehledné a stručné pojednání o těchto metodách najdete v knize [3] v části 2.4.4: Graph Traversals.

Jako první uveďme eulerovský tah (eulerovský graf) a hamiltonovskou cestu a kružnici (hamiltonovský graf). Přečtěte si o tom v učebním textu [4] v části 2.4: Jedním tahem: Eulerovské grafy. Důležitá je především charakterizace eulerovských grafů (věta 2.9). Jako základní informace o hamiltonovských grafech zde postačí poznámka o hamiltonovských kružnicích na straně 120 v [9]; pokuste se v této knize též vyřešit cvičení 6 na straně 122. Vážní zájemci o problematiku eulerovských grafů si mohou v knize [9] přečíst také část 3.6: Algoritmus na kreslení grafu jedním tahem – jedná se však již o látku rozšiřující. Obdobně, zájemci o problematiku hamiltonovských grafů si mohou v knize [10] projít kapitolu 3: Hamiltonian Graphs – ta je však již dalekosáhlým rozšířením základní látky.

Dvě další obecné metody procházení grafu jsou prohledávání do šířky a prohledávání do hloubky. Jsou potřebné pro zpracovávání velkých grafů v počítači. Prostudujte ve výukovém textu [4] část 2.2: Prohledávání grafu. Ukázky prohledávání grafu do šířky a do hloubky najdete v části 2.5: Cvičení: Procházení grafů, souvislost a isomorfismus.

Cestování grafem dostává novou dimenzi, když ke každé hraně je přiřazeno nezáporné číslo, zvané délka (cena, váha) hrany. Hovoříme pak o váženém (ohodnoceném) grafu. Úkolem je pak najít nejlevnější (nejkratší) průchod grafem (samozřejmě, toto je volné vyjádření, které je třeba upřesnit).

První myšlenkou je najít minimální kostru grafu, tedy kostru s nejmenším součtem délek jejích hran.

Existuje několik jednoduchých algoritmů pro nalezení minimální kostry grafu. Asi nejznámější jsou Kruskalův algoritmus a Jarníkův (Primův) algoritmus. Seznamte se s nimi v textu [4] v části 5.1: Hledání minimální kostry nebo v knize [9] v částech 4.4: Problém minimální kostry a 4.5: Jarníkův algoritmus a Borůvkův algoritmus. Studium důkazů správnosti uvedených dvou algoritmů je již nad rámec toho, co budeme v tomto kursu potřebovat (smozřejmě, zájemci se s důkazy mohou seznámit); zaměřte se především na ukázky průběhu algoritmu (například v [4], část 5.5: Cvičení: Příklady o stromech, kostrách a hladovém postupu).

Modifikací problému hamiltonovské kružnice v ohodnocených grafech je problém obchodního cestujícího. Je dán úplný graf G, $V(G) = \{c_1, \ldots, c_n\}$ a vzdálenost $d(c_i, c_j)$ pro každou

dvojici vrcholů (v této souvislosti se obvykle nazývají města) c_i a c_j. Úkolem je najít hamiltonovskou kružnici v G s minimálním součtem vzdáleností vrcholů, tedy najít permutaci π množiny $\{1, \ldots, n\}$ jež minimalizuje hodnotu součtu

$$\sum_{1 \leq i \leq n-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(n)}, c_{\pi(1)})$$

Není znám žádný polynomiální algoritmus pro řešení tohoto problému, ale současně není dokázáno, že takový algoritmus neexistuje.

6.5 Stromy

Do této části patří následující pojmy:

strom, les, kořenový strom, hloubka stromu, uspořádaný strom, binární strom, reprezentace binárních stromů.

Postačující poučení najdete v knize [3] v části 2.4.5: Trees.

Z české (a snadno dostupné literatury) vám opět doporučuji výukový text [4]. Projděte důkladně celou část 4.1: Základní vlastnosti stromů; seznámíte se (případně zopakujete si) s pojmy strom, les a naučíte se jejich základní vlastnosti. Také důkladně projděte část 4.2: Kořenové stromy; seznámíte se s pojmy kořenový strom, uspořádaný strom. Nezapomeňte na cvičení na konci každé z uvedených podkapitol.

Binární strom a jeho reprezentaci vymezíme stručně (avšak dostatečně pro tento kurs) přímo zde.

Binární strom je kořenový strom, v němž každý vrchol má nejvýše dvě děti a ka každému vrcholu, kromě kořene, je přiřazeno číslo 1 nebo 2. V tomto případě můžeme hovořit o prvním nebo druhém dítěti, případně o levém nebo pravém dítěti. Všimněte si, že vrchol může mít pouze levé dítě či pouze pravé dítě. V **úplném binárním stromu** každý vrchol, kromě listů, má dvě děti.

Reprezentace binárních stromů: Binární strom s n vrcholy označenými celými čísly od 1 do n můžeme reprezentovat třemi uspořádanými n-ticemi (lineárními seznamy) $P[1, \ldots, n]$, $L[1, \ldots, n]$, $R[1, \ldots, n]$. Pro každý vrchol i, $P[i]$ obsahuje číslo rodiče vrcholu i, $L[i]$ obsahuje číslo levého dítěte a $R[i]$ obsahuje číslo pravého dítěte vrcholu i. S touto reprezentací binárního stromu lze každou z operací (a) jít k rodiči, (b) jít k levému dítěti, (c) jít k pravému dítěti, provádět v čase $\mathcal{O}(1)$.

Zájemcům o další poučení lze doporučit následující dva zdroje (upozorňuji však, že obsahují mnohem více informací (pojmů, tvrzení), než stanovuje sylabus tohoto kursu):

[9], kapitola 4: Stromy

[7], část 2.3: Stromy; Donald E. Knuth zde čtenáři předkládá velmi mnoho informací (v českém překladu zabírají 116 stran). Upozorňuji, že kořenový strom se v této knize nazývá orientovaný strom. Jak jsem uvedl již výše, Knuth nahlíží na stromy především z hlediska programátora – část o stromech (jako důležitém typu nelineárních struktur) je zařazena v

kapitole 2: Informační struktury. Nevyhýbá se však ani přístupu matematickému (část 2.3.4: Základní matematické vlastnosti stromů).

7 Základy teorie složitosti

V této části se máte naučit minimum základů teorie složitosti. Proč minimum? Prostě proto, že této problematice je v sylabu tohoto kursu věnováno málo časového prostoru.

Mnoho problémů, například v teorii grafů, je považováno za "těžké"z algoritmického úhlu pohledu. V této části budeme prezentovat minimální teoretický základ potřebný k pochopení obvyklé klasifikace "obtížnosti"problémů.

Problematika (v rozsahu, jaký v tomto kursu potřebujeme) je pěkně zpracována v knize [10], kapitola 2: A Glimpse at Complexity Theory.

Existuje velké množství knih a studijních materiálů pojednávajících o teorii složitosti (samozřejmě převážně v angličtině, ale mnohé také v češtině). Řada z nich je volně k dispozici na internetu a jistě si je snadno vyhledáte. Je tu však jeden problém: Výukové texty jsou často určeny pro kursy teorie složitosti, kterým je věnován celý semestr, ne-li celé dva semestry (například na Katedře kogiky Filosofické fakulty UK se v zimním semestru vyučuje Výpočetní složitost I a v letním semestru Výpočetní složitost II, vždy 2 hodiny přednášky týdně). Ukázkou takového textu je [11].

Rád bych se zmínil ještě o učebním textu [6] - je to česky psaný a volně přístupný učební text pro kursy teoretické informatiky. Teorii složitosti je věnována kapitola 8: Úvod do teorie složitosti a kapitola 9: Úvod do teorie složitosti – rozšiřující část. Samozřejmě, před studiem uvedených kapitol by bylo třeba seznámit se s látkou některých předchozích kapitol, jistě aspoň kapitoly 6: Úvod do teorie vyčíslitelnosti.

7.1 Některé třídy složitosti

Látka této části:

rozhodovací problém, třída **P**, třída **NP**.

Turingovy stroje jsou užitečným modelem výpočtů poskytujícím formální definici algoritmu. Nebudeme zde tento model prezentovat, budeme však předpokládat, že máme rozumné chápání toho, co je algoritmus (budeme tedy vycházet z Churchovy - Turingovy teze: ke každému algoritmu lze zkonstruovat ekvivalentní Turingův stroj). Naším cílem je podat neformální vymezení tříd **P** a **NP**.

Přečtěte si v knize [10] část 2.1: Some complexity classes.

7.2 Polynomiální redukce

Látka této části:

polynomiální redukce, **NP**-těžký problém, **NP**-úplný problém.

Naučíme se porovnávat rozhodovací problémy pomocí polynomiálních redukcí. To umožní definovat, kdy je rozhodovací problém **NP**-těžký a kdy je **NP**-úplný.

Přečtěte si v knize [10] část 2.2: Polynomial reductions.

7.3 Výpočetně obtížné problémy teorie grafů

Mnohé problémy teorie grafů jsou výpočetně obtížné. Cílem této části je prezentovat seznam několika **NP**-úplných problémů teorie grafů. Jde pouze o to registrovat fakt, že problém je **NP**-úplný, nikoli to dokazovat.

Přečtěte si v knize [10] část 2.3: More hard problems in graph theory.

Alternativně (nebo navíc) můžete v textu [11] projít část 3.3: **NP**-úplnost a část 3.4: Další příklady **NP**-úplných úloh. Najdete tam, jak názvy napovídají, příklady **NP**-úplných problémů (některé nespadají do teorie grafů) a to včetně důkazů, že opravdu jsou **NP**-úplné. Tyto důkazy můžete vynechat.

8 Sylabus rozdělený do týdnů

1. **Rekurence a základní metody řešení rekurencí**: substituční metoda, iterační metoda

2. **Redukce rekurencí na algebraické rovnice**: homogenní lineární rekurence, charakteristický polynom, charakteristické kořeny

3. **Speciální funkce**: funkce dolní a horní celé části, logaritmy, binomické koeficienty, binomická věta

4. **Asymptotika**: asymptotická hierarchie, $\mathcal{O}$-, Θ- a Ω- notace, relace mezi asymptotickými notacemi, manipulace s $\mathcal{O}$- notací, asymptotika a rekurence: analýza algoritmů rozděl a panuj

5. **Euklidův algoritmus a prvočísla**: Euklidův algoritmus: největší společný dělitel, nejmenší společný násobek, rozšířený Euklidův algoritmus, analýza Euklidova algoritmu; prvočísla: Základní věta aritmetiky, Eratosthenovo síto, Eulerova funkce ϕ, Prvočíselná věta

6. **Kongruence**: zbytkové třídy modulo n, řešení lineárních kongruencí, Čínská věta o zbytcích, modulární umocňování, Malá Fermatova věta, Eulerova věta

7. **Diskrétní logaritmy**: primitivní kořeny, diskrétní logaritmus

8. **Základní pojmy z teorie grafů**: orientovaný a neorientovaný graf, vrcholy, hrany, incidence, stupně, reprezentace grafů: seznamy sousedů, matice sousednosti, matice incidence

9. **Další vlastnosti grafů**: sledy, tahy, cesty, cykly, souvislé komponenty, silně souvislé komponenty, vrcholová souvislost, hranová souvislost, isomorfismus, regularita, vrcholově symetrický graf, hranově symetrický graf, podgrafy, úplné grafy, bipartitní grafy, stromy, kostry grafu, rovinné grafy, Kuratowského věta, průměr grafu, multigrafy, hypergrafy

10. **Párování a barvení v grafech**: párování, perfektní párování, Hallova věta, Tutteova věta, hranové barvení, Vizingova věta, vrcholové barvení, chromatické číslo

11. **Cestování grafem**: eulerovský tah, eulerovský graf, hamiltonovská cesta, hamiltonovský graf, prohledávání do šířky, prohledávání do hloubky, minimální kostra, Kruskalův algoritmus, Primův algoritmus, problém obchodního cestujícího

12. **Stromy**: strom, les, kořenový strom, hloubka stromu, uspořádaný strom, binární strom, reprezentace binárních stromů

13. **Základy teorie složitosti**: některé třídy složitosti: rozhodovací problém, třída **P**, třída **NP**; polynomiální redukce: **NP**-těžký problém, **NP**-úplný problém; výpočetně obtížné problémy teorie grafů

Reference

[1] Thomas H. Cormen, Charles Leiserson, Ronald L. Rivest, *Introduction to Algorithms*. The MIT Press – McGraw Hill, Cambridge, New York, 1990.

[2] Ronald L. Graham, Donald E. Knuth, Oren Patashnik, *Concrete Mathematics*. Addison – Wesley Publishing Company, 1989.

[3] Jozef Gruska, *Foundations of Computing*. International Thomson Computer Press, 1997.

[4] Petr Hliněný, *Základy teorie grafů pro (nejen) informatiky*. Fakulta Informatiky, Masarykova Univerzita, 2010.
`https://is.muni.cz/do/1499/el/estud/fi/js10/grafy/Grafy-text10.pdf`

[5] Kenneth Ireland, Michael Rosen, *A Classical Introduction to Modern Number Theory*. Springer, New York, 1990.

[6] Petr Jančar, *Teoretická informatika*. Vysoká škola báňská – Technická univerzita Ostrava, 2007 (a 2010).
`http://www.cs.vsb.cz/jancar/TEORET-INF/ti-text.2010-01-20.pdf`

[7] Donald E. Knuth, *Umění programování, 1. díl, Základní algoritmy*. Computer Press, a.s., Brno, 2008.

[8] Petr Kovář, *Teorie grafů*. 2020.
http://homel.vsb.cz/~kov16/files/skriptum_teorie_grafu_tisk.pdf

[9] Jiří Matoušek, Jaroslav Nešetřil, *Kapitoly z diskrétní matematiky*. Karolinum, Praha, 2002.

[10] Michel Rigo, *Advanced Graph Theory and Combinatorics*. ISTE Ltd and John Wiley & Sons, Inc., 2016.

[11] Petr Savický, *Výpočetní složitost I pro obor logika na FF UK*.
http://www.cs.cas.cz/~savicky/vyuka/vypsl/vypsl2016zs1.pdf

Martin Kuřil
Katedra matematiky
Přírodovědecká fakulta
Univerzita Jana Evangelisty Purkyně
Ústí nad Labem
martin.kuril@ujep.cz