Matematická analýza 1

Asymptotika & Složitost algoritmů

Tomáš Kalvoda[1], Pavel Paták[2]

[1]tomas.kalvoda@fit.cvut.cz, [2]pavel.patak@fit.cvut.cz

Katedra aplikované matematiky
Fakulta informačních technologií
České vysoké učení technické v Praze

21. února 2025
LS 2024/2025

Hlavní body

Hlavní výsledky této přednášky

- Další způsoby porovnávání chování posloupností pomocí Ω, ω, Θ a $\sim$.

- Ukázky popisu složitosti algoritmů.

Hlavní body

Ω

ω

Θ

Další asymptotická porovnání: proč?

- V jedné z předchozích přednášek jsme se seznámili s horními asymptotickými mezemi o a $\mathcal{O}$.

- Vztah $a_n = \mathcal{O}(n)$ splní jak $a_n = n$, tak i $a_n = \sqrt{n}$ nebo $a_n = 0$.

Další asymptotická porovnání: proč?

- V jedné z předchozích přednášek jsme se seznámili s horními asymptotickými mezemi o a $\mathcal{O}$.
- Vztah $a_n = \mathcal{O}(n)$ splní jak $a_n = n$, tak i $a_n = \sqrt{n}$ nebo $a_n = 0$.
- Při porovnávání chování posloupností bychom přirozeně chtěli mít možností **být přesnější**!
- Pro konkrétnost se v této přednášce už většinou omezíme pouze na posloupnosti, i když bychom mohli definovat tyto vztahy i pro funkce.

Dolní asymptotická mez: Ω

Definice (Asymptotická dolní mez Ω):

Mějme dvě posloupnosti $(a_n)_{n=1}^{\infty}$ a $(b_n)_{n=1}^{\infty}$. Řekneme, že **posloupnost** $(a_n)_{n=1}^{\infty}$ **je asymptoticky zdola omezená posloupností** $(b_n)_{n=1}^{\infty}$, symbolicky „$a_n = \Omega(b_n)$ pro $n \to \infty$“, právě když existují kladná konstanta $c \in \mathbb{R}$ a přirozené $N \in \mathbb{N}$ tak, že pro všechna $n \geq N$ platí

$$|a_n| \geq c \cdot |b_n|.$$

Dolní asymptotická mez: Ω

Definice (Asymptotická dolní mez Ω):

Mějme dvě posloupnosti $(a_n)_{n=1}^{\infty}$ a $(b_n)_{n=1}^{\infty}$. Řekneme, že **posloupnost** $(a_n)_{n=1}^{\infty}$ **je asymptoticky zdola omezená posloupností** $(b_n)_{n=1}^{\infty}$, symbolicky „$a_n = \Omega(b_n)$ pro $n \to \infty$“, právě když existují kladná konstanta $c \in \mathbb{R}$ a přirozené $N \in \mathbb{N}$ tak, že pro všechna $n \geq N$ platí

$$|a_n| \geq c \cdot |b_n|.$$

Ω je řecké velké písmenko *omega*. Z definice ihned plynou následující pozorování:

- $a_n = \Omega(b_n)$, právě když $b_n = \mathcal{O}(a_n)$.
- $a_n = \Omega(a_n)$.
- Pokud jsou členy posloupnosti $(b_n)_{n=1}^{\infty}$ nenulové a pokud $\lim_{n\to\infty} \frac{a_n}{b_n} > 0$, potom $a_n = \Omega(b_n)$.
- Vztah Ω je tranzitivní.

Dolní asymptotická mez: Ω

Příklad.

Platí $n = \Omega\left(\sqrt{n}\right)$:

Skutečně, pro $c = 1$ a každé přirozené n platí $n = \sqrt{n} \cdot \sqrt{n} \geq 1 \cdot \sqrt{n}$.

Příklad.

Platí $2^n = \Omega(n^2)$:

Skutečně: Protože

$$\lim_{n\to\infty} \frac{2^{n+1}/(n+1)^2}{2^n/n^2} = 2 \lim_{n\to\infty} \frac{1}{(1+\frac{1}{n})^2} = 2 \cdot \frac{1}{(1+0)^2} = 2 > 1,$$

pak podle podílového kritéria $\lim_{n\to\infty} \frac{2^n}{n^2} = +\infty$.

Dolní striktní asymptotická mez: ω

Definice (Striktně menší dolní mez ω):

Mějme dvě posloupnosti $(a_n)_{n=1}^{\infty}$ a $(b_n)_{n=1}^{\infty}$. Řekneme, že **posloupnost** $(a_n)_{n=1}^{\infty}$ **je asymptoticky zdola striktně omezená posloupností** $(b_n)_{n=1}^{\infty}$, symbolicky „$a_n = \omega(b_n)$ pro $n \to \infty$“, právě když pro každé kladné $c \in \mathbb{R}$ existuje $N \in \mathbb{N}$ tak, že pro všechna $n \geq N$ platí

$$|a_n| > c \cdot |b_n|.$$

Dolní striktní asymptotická mez: ω

Definice (Striktně menší dolní mez ω):

Mějme dvě posloupnosti $(a_n)_{n=1}^{\infty}$ a $(b_n)_{n=1}^{\infty}$. Řekneme, že **posloupnost $(a_n)_{n=1}^{\infty}$ je asymptoticky zdola striktně omezená posloupností** $(b_n)_{n=1}^{\infty}$, symbolicky „$a_n = \omega(b_n)$ pro $n \to \infty$“, právě když pro každé kladné $c \in \mathbb{R}$ existuje $N \in \mathbb{N}$ tak, že pro všechna $n \geq N$ platí

$$|a_n| > c \cdot |b_n|.$$

ω je malé řecké písmenko *omega*. Z definice ihned plynou následující pozorování:

- $a_n = \omega(b_n)$, právě když $b_n = o(a_n)$.
- Pokud $a_n = \omega(b_n)$, pak $a_n = \Omega(b_n)$.
- Pokud jsou členy posloupnosti $(b_n)_{n=1}^{\infty}$ nenulové a pokud $\lim_{n\to\infty} \frac{a_n}{b_n} = +\infty$, potom $a_n = \omega(b_n)$.
- ω je tranzitivní.

Asymptotická těsná mez: Θ

Definice (Asymptotická těsná mez Θ):

Mějme dvě posloupnosti $(a_n)_{n=1}^{\infty}$ a $(b_n)_{n=1}^{\infty}$. Řekneme, že **posloupnost** $(a_n)_{n=1}^{\infty}$ **je téhož řádu jako posloupnost** $(b_n)_{n=1}^{\infty}$, symbolicky „$a_n = \Theta(b_n)$ pro $n \to \infty$“, právě když existují kladné konstanty $c_1, c_2 \in \mathbb{R}$ a $N \in \mathbb{N}$ tak, že pro všechna $n \geq N$ platí

$$c_1|b_n| \leq |a_n| \leq c_2|b_n|.$$

Asymptotická těsná mez: Θ

Definice (Asymptotická těsná mez Θ):

Mějme dvě posloupnosti $(a_n)_{n=1}^{\infty}$ a $(b_n)_{n=1}^{\infty}$. Řekneme, že **posloupnost** $(a_n)_{n=1}^{\infty}$ **je téhož řádu jako posloupnost** $(b_n)_{n=1}^{\infty}$, symbolicky „$a_n = \Theta(b_n)$ pro $n \to \infty$“, právě když existují kladné konstanty $c_1, c_2 \in \mathbb{R}$ a $N \in \mathbb{N}$ tak, že pro všechna $n \geq N$ platí

$$c_1|b_n| \leq |a_n| \leq c_2|b_n|.$$

Θ je velké řecké písmenko *theta*. Z definice ihned plynou následující pozorování:

- $a_n = \Theta(b_n)$, právě když $b_n = \Theta(a_n)$.
- Tento vztah vlastně kombinuje $\mathcal{O}$ a Ω.
  Přesněji: $a_n = \Theta(b_n)$, právě když $a_n = \Omega(b_n)$ a $a_n = \mathcal{O}(b_n)$.
- Θ je tranzitivní.

Asymptotická těsná mez: Θ

Příklad.

Například platí (rozmyslete!):

- $n + \sin(n) = \Theta(n)$,
- $n + \sin(n) = \Theta(n + 10)$,
- $n + \sin(n) = \Theta(n - 10)$,
- $n + \sin(n) = \Theta(10n)$.

Už ale třeba není pravda, že

- $n + \sin(n) = \Theta(n^2)$.

Další asymptotická porovnání: komentář

- Vidíme, že nově zavedené vztahy ω, Ω a Θ jsou v podstatě odvozené od o a $\mathcal{O}$, kterým jsme věnovali více času.
- Může pomoci se na porovnávání **posloupností** pomocí ω, Ω, Θ, $\mathcal{O}$ a o dívat jako na analogii porovnávání **čísel**:

ω	$>$
Ω	$\geq$
Θ	$=$
$\mathcal{O}$	$\leq$
o	$<$

- Tento způsob porovnávání funkcí bývá často připisován **Edmundu Landauovi** (německý matematik, 1877 – 1938). Díky své obecnosti nachází využití v matematice, fyzice, informatice…

Asymptotická ekvivalence: $\sim$

Tento vztah definujeme i pro funkce, definice ale přirozeně zahrnuje i případ posloupností.

Definice (Asymptotická ekvivalence $\sim$):

Mějme dvě funkce f, g a bod $a \in \overline{\mathbb{R}}$ takový, že a je hromadným bodem množiny $D_f \cap D_g$ a existuje okolí V_a splňující[a] $(V_a \cap D_f) \setminus \{a\} = (V_a \cap D_g) \setminus \{a\}$.

Řekneme, že **funkce f je asymptoticky ekvivalentní funkci g pro x jdoucí k a**, symbolicky „$f(x) \sim g(x)$ pro $x \to a$“, právě když existuje okolí U_a bodu a a funkce u definovaná na U_a a splňující $\lim_{x \to a} u(x) = 1$ tak, že pro všechna $x \in U_a \cap D_f \cap D_g$ různé od a platí

$$f(x) = u(x)g(x).$$

[a] Definiční obory obou funkcí vypadají na okolí a stejně.

Na následujícím slidu rozebereme několik postřehů.

Asymptotická ekvivalence: $\sim$

- Tento asymptotický vztah $\sim$ je v jistém smyslu nejpřesnější.
- Vztah $\sim$ je symetrický, tranzitivní a reflexivní, jde skutečně o **ekvivalenci**.
- Pokud $a_n \sim b_n$ pro $n \to \infty$, pak i $a_n = \Theta(b_n)$ pro $n \to \infty$. Naopak nutně ne!
- Pokud platí $\lim_{x \to a} \frac{f(x)}{g(x)} = 1$, pak $f(x) \sim g(x)$ pro $x \to a$. Naopak nutně ne!
- Opačná implikace nutně neplatí, kvůli případným nulám. Nejnázorněji asi v tomto případě: $0 \sim 0$ pro $x \to a$ je pravda, ale limitu $\frac{0}{0}$ nemá vůbec smysl počítat.

Asymptotické vztahy: obecné poznámky

Uveďme několik notačních poznámek (platné i pro další symboly a i funkce):

- Někdy narazíte na zápis tvaru „$\mathcal{O}(a_n) = \mathcal{O}(b_n)$“. Tím má autor na mysli, že každá posloupnost $(c_n)_{n=1}^{\infty}$ splňující $c_n = \mathcal{O}(a_n)$, splňuje i $c_n = \mathcal{O}(b_n)$.

- Například platí $\mathcal{O}(n) = \mathcal{O}(n^2)$, nebo $o(n) = \mathcal{O}(n)$. Tento způsob zápisu může být potenciálně matoucí, nejde zde o skutečnou rovnost, vždy je potřeba ho interpretovat zleva-doprava.

- Občas se můžete setkat i se zápisem tvaru „$a_n = \mathcal{O}\left(n^{\mathcal{O}(b_n)}\right)$“. Tím se myslí, že ona posloupnost $(a_n)_{n=1}^{\infty}$ splňuje $a_n = \mathcal{O}(n^{c_n})$, kde $(c_n)_{n=1}^{\infty}$ je nějaká posloupnost splňující $c_n = \mathcal{O}(b_n)$.

Hlavní body

Ω

ω

Θ

Problémy a algoritmy

- **Výpočetní problém** P: úkol zpracovat/transformovat vstupní data I na výstupní data O s požadovanými vlastnostmi.
- **Algoritmus** A: výpočetní postup řešení problému P, čili posloupnost výpočetních kroků, která z vstupních dat vyprodukuje výstupní data požadovaných vlastností.
- **Instance problému**: problém s konkrétními daty potřebnými pro jeho řešení.

Problémy a algoritmy

- **Výpočetní problém** P: úkol zpracovat/transformovat vstupní data I na výstupní data O s požadovanými vlastnostmi.
- **Algoritmus** A: výpočetní postup řešení problému P, čili posloupnost výpočetních kroků, která z vstupních dat vyprodukuje výstupní data požadovaných vlastností.
- **Instance problému**: problém s konkrétními daty potřebnými pro jeho řešení.

Definujeme-li „velikost“ vstupních dat I, pak můžeme zkoumat chování daného algoritmu A z různých úhlů pohledu:

- Průměrný/minimální/maximální/řádový (asymptotický) počet „elementárních operací“, nutných k získání výstupu v závislosti na velikosti vstupu.
- Průměrná/minimální/maximální/řádová (asymptotická) „paměťová náročnost“, nutná k získání výstupu v závislosti na velikosti vstupu.

Problém třídění

- **Vstup**: množina $\{x_1, \ldots, x_n\}$ a úplné uspořádání $\leq$ mezi jejími prvky. Velikostí vstupu rozumíme jednoduše počet prvků vstupu.
- **Výstup**: uspořádaná n-tice $(x_{k_1}, \ldots, x_{k_n})$, kde $(k_1, \ldots, k_n)$ je permutace množiny $\{1, \ldots, n\}$ a platí $x_{k_1} \leq \cdots \leq x_{k_n}$.
- **Elementární operace**: porovnání dvou prvků pomocí $\leq$.

Bublinkový algoritmus (*Bubble sort*)

```
procedure bubbleSort(A : array of length n)
   for k in n-1 to 1 do
       sorted = true
       for i in 1 to k do
           if A[i] > A[i+1] then
               swap( A[i], A[i+1])
               sorted = false
           end if
       end for
       if sorted then return A
   end for
   return A
end procedure
```

Bublinkový algoritmus (*Bubble sort*)

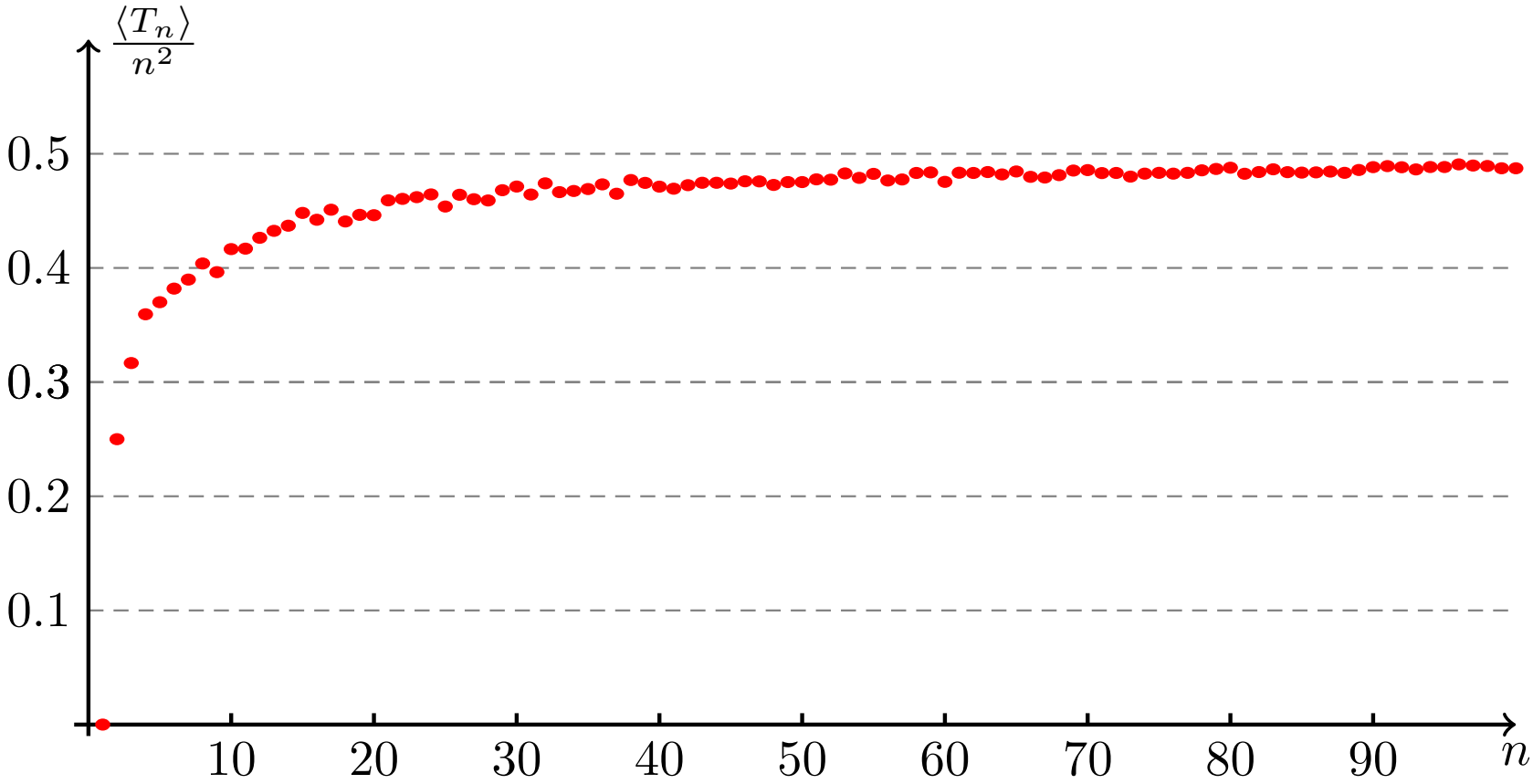
- Celkový počet porovnání může být nejhůře

$$(n-1)+(n-2)+\cdots+2+1=\sum_{k=1}^{n-1}k=\frac{n(n-1)}{2}.$$

- Pokud je na vstupu již setříděný seznam, pak algoritmus provede $(n-1)$ porovnání (nejlepší varianta).

Poznámka (Složitost *Bubble sort*):

Můžeme tedy shrnout, že pro operační složitost *Bubble sort* je třídy $\mathcal{O}(n^2)$ a $\Omega(n)$.



Poznámka (Ilustrace složitosti *Bubble sort*):

Pro každé $n \in \{1, 2, \ldots, 100\}$ jsme 20-krát setřídili náhodně permutovanou množinu $\{1, 2, \ldots, n\}$ a vypočetli tak střední hodnotu $\langle T_n \rangle$, počtu porovnání při třídění prvků množiny délky n. Na grafu je pak vynesen poměr $\frac{\langle T_n \rangle}{n^2}$.

Quick sort

Nechť je opět dán vstup $\{x_1, x_2, \ldots, x_n\}$.

Quick sort

Nechť je opět dán vstup $\{x_1, x_2, \ldots, x_n\}$.

- Vyberme náhodně prvek ze seznamu (nazývaný *pivot*).

Quick sort

Nechť je opět dán vstup $\{x_1, x_2, \ldots, x_n\}$.

- Vyberme náhodně prvek ze seznamu (nazývaný *pivot*).
- Prvky ze seznamu menší než pivot, dej do jednoho podseznamu a prvky větší do druhého podseznamu.

Quick sort

Nechť je opět dán vstup $\{x_1, x_2, \ldots, x_n\}$.

- Vyberme náhodně prvek ze seznamu (nazývaný *pivot*).
- Prvky ze seznamu menší než pivot, dej do jednoho podseznamu a prvky větší do druhého podseznamu.
- Dva kratší seznamy uspořádej podle velikosti.

Quick sort

Nechť je opět dán vstup $\{x_1, x_2, \ldots, x_n\}$.

- Vyberme náhodně prvek ze seznamu (nazývaný *pivot*).
- Prvky ze seznamu menší než pivot, dej do jednoho podseznamu a prvky větší do druhého podseznamu.
- Dva kratší seznamy uspořádej podle velikosti.
- Vezmi uspořádaný první seznam, za něj dej pivota a připoj uspořádaný druhý seznam.

Quick sort

Nechť je opět dán vstup $\{x_1, x_2, \ldots, x_n\}$.

- Vyberme náhodně prvek ze seznamu (nazývaný *pivot*).
- Prvky ze seznamu menší než pivot, dej do jednoho podseznamu a prvky větší do druhého podseznamu.
- Dva kratší seznamy uspořádej podle velikosti.
- Vezmi uspořádaný první seznam, za něj dej pivota a připoj uspořádaný druhý seznam.

Algoritmus probíhá rekurentně. Uspořádání dvouprvkového seznamu je jednoduché.

[Tony Hoare, 1961]

Quick sort

- Označme nyní $\langle T_n \rangle$ **průměrný počet porovnání** pro uspořádání seznamu délky n pomocí algoritmu *Quick sort*.

- Příští semestr ukážeme, že

$$\langle T_n \rangle = \mathcal{O}\big(n \ln(n)\big).$$

Quick sort

- Označme nyní $\langle T_n \rangle$ **průměrný počet porovnání** pro uspořádání seznamu délky n pomocí algoritmu *Quick sort*.

- Příští semestr ukážeme, že

$$\langle T_n \rangle = \mathcal{O}\big(n \ln(n)\big).$$

- Při nešťastné volbě pivotů se může stát, že k setřídění pomocí *Quick sort* bude potřeba n^2 porovnání.

- Vzhledem k důležitosti tohoto problému existuje celá řada dalších třídících algoritmů, například *Merge sort*, který netrpí poznámkou v předchozím bodu [John von Neumann, 1945].

Umocňování

- **Vstup**: reálné číslo α a přirozené číslo $N \in \mathbb{N}$.
- **Výstup**: Hodnota α^N.
- **Elementární operace**: aritmetická operace (sčítání, odčítání, násobení, dělení).
- Naivní implementace dle definice $\alpha^N = \underbrace{\alpha \cdot \alpha \cdots \alpha}_{N\times}$ vyžaduje přesně $N-1$ operací násobení. Řádově jde tedy o $\mathcal{O}(N)$ (resp. $\Omega(N)$ a $\Theta(N)$) algoritmus.
- Tyto poznámky platí třeba i pro násobení matic, nebo čísel v konečných tělesech, chápeme-li elementární operaci jako násobení příslušných objektů.

Umocňování: *Square-and-multiply*

- Typicky ale máme k dispozici binární reprezentaci čísla $N = (N_k N_{k-1} \dots N_1 N_0)_2$, kde $N_0, \dots, N_{k-1} \in \{0, 1\}$ a $N_k = 1$, tj.

$$N = \sum_{j=0}^{k} N_j 2^j.$$

Umocňování: *Square-and-multiply*

- Typicky ale máme k dispozici binární reprezentaci čísla $N = (N_k N_{k-1} \dots N_1 N_0)_2$, kde $N_0, \dots, N_{k-1} \in \{0, 1\}$ a $N_k = 1$, tj.

$$N = \sum_{j=0}^{k} N_j 2^j.$$

- Za tohoto předpokladu pak platí

$$\alpha^N = \alpha^{\sum_{j=0}^{k} N_j 2^j} = \prod_{j=0}^{k} \alpha^{N_j 2^j}.$$

Umocňování: *Square-and-multiply*

- Typicky ale máme k dispozici binární reprezentaci čísla $N = (N_k N_{k-1} \dots N_1 N_0)_2$, kde $N_0, \dots, N_{k-1} \in \{0, 1\}$ a $N_k = 1$, tj.

$$N = \sum_{j=0}^{k} N_j 2^j.$$

- Za tohoto předpokladu pak platí

$$\alpha^N = \alpha^{\sum_{j=0}^{k} N_j 2^j} = \prod_{j=0}^{k} \alpha^{N_j 2^j}.$$

- Pro výpočet α^N je pak tedy potřeba získat opakovaným umocňováním α^{2^k} – to je $k-1$ operací násobení – a násobit mezivýsledek pokud $N_j \neq 0$ – to je také nejhůře k operací násobení.
- Pro *square-and-multiply* algoritmus tak dostáváme operační složitost řádově $2k = \mathcal{O}(\log_2 N) = \mathcal{O}(\ln N)$, N.B.:

$$N \geq 2^k \Rightarrow \log_2 N \geq k.$$

Hlavní body

Ω

ω

Θ

3 Dodatek

Komentář

- V dalším semestru na tato témata složitosti navážeme při studování rekurencí.

- Na shledanou v příštím semestru u BI-MA2!