ASYMPTOTICKÁ SLOŽITOST

Karel Horák, Petr Ryšavý

24. února 2016

Katedra počítačů, FEL, ČVUT

ORGANIZACE

Organizace

8 programovacích úloh, dohromady 16 bodů

pro zápočet nutné získat alespoň 10 bodů

nutné vyřešit 9 z 10 testovacích dat, v rámci paměťového a časového limitu

9/10 odpovídá 1 bodu za úlohu, 10/10 odpovídá 2 bodům

kód musí být vlastní dílo, úlohy řešte včas

max 2 cvičení s neomluvenými absencemi

možnost navštěvovat ACM seminář

přednášky PO 14.30-16.00

PŘÍKLADY

Příklad 1

Nechť $f(n)$ a $g(n)$ jsou funkce definované na $\mathbb{N}$. Pokud $f(n) \in \mathcal{O}(g(n))$, pak:

1. $f(n) \leq g(n)$ pro všechna dostatečně velká n.
2. $\exists c > 0 : \exists n_0 : \forall n > n_0 : f(n) \leq c \cdot g(n)$
   ($c \in \mathbb{R}^+$, $n_0 \in \mathbb{N}$, $f(n) \geq 0$ a $g(n) \geq 0$)
3. $\lim_{n\to\infty} \frac{f(n)}{g(n)} \to c$, kde $c \in \mathbb{R}_0^+$ a $f(n), g(n) \in \mathbb{R}_0^+$
4. Funkce g roste asymptoticky rychleji než funkce f.
5. Funkce f a g jsou nezáporné.

Řešení 1

1. Neplatí, protože $n \in \mathcal{O}(n/2)$ ale $n > n/2$ pro všechna přirozená čísla.
2. Platí.
3. Platí. [`https://cw.fel.cvut.cz/wiki/_media/courses/a4b33alg/alg01_2010.pdf`, slide 13]
4. Neplatí. Funkce g může růst i asymptoticky stejně rychle jako funkce f.
5. Platí. Nezápornost je jednou z podmínek, abychom mohli tvrdit, že $f(n) \in \mathcal{O}(g(n))$.

Příklad 2

Nechť $f(n)$ a $g(n)$ jsou funkce definované na $\mathbb{N}$. Pokud $f(n) \in \Omega(g(n))$, pak:

1. $f(n) \geq g(n)$ pro všechna dostatečně velká n.
2. $\forall c > 0 : \exists n_0 : \forall n > n_0 : f(n) > c \cdot g(n)$
   ($c \in \mathbb{R}^+$, $n_0 \in \mathbb{N}$, $f(n) \geq 0$ a $g(n) \geq 0$)
3. $g(n) \in \mathcal{O}(f(n))$
4. $\lim_{n\to\infty} \frac{f(n)}{g(n)} \to \infty$, kde $f(n), g(n) \in \mathbb{R}_0^+$
5. Funkce f roste asymptoticky alespoň tak rychle jako funkce g.
6. Funkce f a g jsou kladné.

Pro zamyšlení: Existuje i třída malá omega, ve ktere jsou funkce, ktere rostou striktně rychleji. Zkuste vymyslet definici.

Řešení 2

1. Neplatí, protože $n/2 \in \Omega(n)$ ale $n/2 < n$ pro všechna přirozená čísla.
2. Neplatí. Protipříkladem je $f(n) = \frac{1}{2}n$, $g(n) = n$ a $c = 1$.
3. Platí.
4. Neplatí pokud $f(n) \in \Theta(g(n))$.
5. Platí.
6. Neplatí. Funkce f a g mohou být klidně nulové.

$f(n) \in \omega(g(n))$ právě tehdy, když
$\forall c > 0 : \exists n_0 : \forall n > n_0 : f(n) > c \cdot g(n)$
($c \in \mathbb{R}^+$, $n_0 \in \mathbb{N}$, $f(n) \geq 0$ a $g(n) \geq 0$).

Příklad 3

Nechť $f(n)$ a $g(n)$ jsou funkce definované na $\mathbb{N}$. Pokud $f(n) \in \Theta(g(n))$, pak:

1. $\exists c_1, c_2 > 0 : \exists n_0 : \forall n > n_0 : c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$
   ($c_1, c_2 \in \mathbb{R}^+$, $n_0 \in \mathbb{N}$, $f(n) \geq 0$ a $g(n) \geq 0$)
2. funkce f může růst rychleji než funkce g
3. funkce f může růst asymptoticky rychleji než funkce g
4. $f(n) \in \Omega(g(n)) \cap \mathcal{O}(g(n))$.
5. $g(n) \in \mathcal{O}(f(n))$
6. $\exists n \in \mathbb{N} : f(n) \neq g(n)$.

Řešení 3

1. Platí.
2. Platí. Například $2n$ roste rychleji než n, ale přitom roste asymptoticky stejně rychle.
3. Neplatí.
4. Platí. $f(n) \in \Theta(g(n))$ znamená, že $f(n) \in \Omega(g(n))$ a také, že $f(n) \in \mathcal{O}(g(n))$.
5. Platí.
6. Neplatí. Zvolme $f(n) = g(n)$. Pak $f(n) \in \Theta(g(n))$, ale neplatí závěr.

Příklad 4

Pro dvě spojité funkce $f(x)$ a $g(x)$ rostoucí na celém $\mathbb{N}$ platí $f(x) < g(x)$ pro každé $x \in \mathbb{N}$. To znamená

1. $f(x) \notin \Omega(g(x))$
2. $f(x) \notin \mathcal{O}(g(x))$
3. $f(x) \notin \Theta(g(x))$
4. je možné, že $f(x) \in \Omega(g(x))$
5. je možné, že $f(x) \in \mathcal{O}(g(x))$
6. $f(x)$ roste asymptoticky pomaleji než $g(x)$

Řešení 4

1. Neplatí. Protipříkladem je $f(x) = x$ a $g(x) = 2x$.
2. Neplatí. Protipříkladem je $f(x) = x$ a $g(x) = 2x$.
3. Neplatí. Protipříkladem je $f(x) = x$ a $g(x) = 2x$.
4. Platí. Příkladem je $f(x) = x$ a $g(x) = 2x$.
5. Platí, dokonce toto platí vždy. Lze to dokázat, pokud zvolíme např. $c = 1$ a $n_0 = 1$.
6. Neplatí. Protipříkladem je $f(x) = x$ a $g(x) = 2x$.

Příklad 5

Nechť $f(n)$ a $g(n)$ jsou asymptoticky nezáporné funkce. Za použití základní definice Θ dokažte, že

$$\max(f(n), g(n)) \in \Theta(f(n) + g(n)).$$

[`http://graphics.stanford.edu/courses/cs161-16-winter/Handouts/hmwk01.pdf`]

Řešení 5

Pro všechna n větší než 1 platí následující tvrzení:

$\max(f(n), g(n)) \leq f(n) + g(n)$ a
$2\max(f(n), g(n)) \geq f(n) + g(n)$.

Proto pro $c_1 = \frac{1}{2}$, $c_2 = 1$ a všechna $n \geq n_0 = 1$ platí

$$c_1(f(n) + g(n)) \leq \max(f(n), g(n)) \leq c_2(f(n) + g(n)),$$

což dokazuje, že $\max(f(n), g(n)) \in \Theta(f(n) + g(n))$.

Příklad 6

Seřaďte následující funkce podle asymptotické složitosti

$n \log_2 n$	$\log_2 n$
2^n	$\ln n$
1	$n!$
n	$3^n + 2^n$
$5n$	n^n
$\sqrt[3]{n^2}$	n^{1000}
$n \log_2 n + 7n$	3^n
$\log_4 n$	$2^{\log_2 n}$

Pokud je to málo, můžete zkusit vyřešit příklad 4 z `http://graphics.stanford.edu/courses/cs161-16-winter/Handouts/hmwk01.pdf`.

Řešení 6

1. 1
2. $\log_4 n$, $\log_2 n$, $\ln n$
3. $\sqrt[3]{n^2}$
4. n, $5n$, $2^{\log_2 n}$
5. $n \log_2 n$, $n \log_2 n + 7n$
6. n^{1000}
7. 2^n
8. 3^n, $3^n + 2^n$
9. $n!$
10. n^n

Příklad 7

Algoritmus A probírá postupně všechny prvky v dvourozměrném poli o velikosti $n \times n$ a s každým prvkem provádí další (nám neznámou) akci, jejíž složitost je $\Theta(\log_2(n))$. Jaká je celková asymptotická složitost algoritmu A a proč?

Řešení 7

Pro zpracování jednoho prvku potřebujeme nejméně $1 + c_1 \log_2(n)$ operací a nejvýše $1 + c_2 \log_2(n)$ operací pro vhodnou volbu c_1 a c_2 a dostatečně velká n. Pak platí, že celkový počet operací $f(n)$ je mezi $n^2(1 + c_1 \log_2(n))$ a $n^2(1 + c_2 \log_2(n))$. Pro dostatečně velká n, kde $\log_2(n) \geq 1$, je tedy

$$c_1 n^2 \log_2(n) \leq f(n) \leq (c_2 + 1) n^2 \log_2(n).$$

To znamená, že složitost algoritmu je

$$\Theta(n^2 \log_2 n).$$

Příklad 8

Uveďte příklad tří rostoucích funkcí reálné proměnné $f(x)$, $g(x)$ a $h(x)$, pro které současně platí všechny tři následující vztahy.

$$f(x) \notin \mathcal{O}(g(x))$$
$$g(x) \notin \Theta(h(x))$$
$$h(x) \notin \Omega(f(x))$$

Pokud taková trojice funkcí nemůže existovat, napište krátké zhodnocení proč.

Řešení 8

1. $f(x) = n^3$, $g(x) = n^2$, $h(x) = n$
2. $f(x) = n^3$, $g(x) = n$, $h(x) = n^2$

Příklad 9

Uveďte příklady tří rostoucích funkcí reálné proměnné $f(x)$, $g(x)$ a $h(x)$, pro které současně platí všechny tři následující vztahy.

$$f(x) \notin \mathcal{O}(g(x))$$
$$g(x) \notin \Omega(h(x))$$
$$h(x) \notin \Theta(f(x))$$

Pokud taková trojice funkcí nemůže existovat, napište krátké zhodnocení proč.

Řešení 9

1. $f(x) = n^3$, $g(x) = n$, $h(x) = n^2$
2. $f(x) = n^3$, $g(x) = n$, $h(x) = n^5$

Příklad 10

Jaká je složitost následujícího algoritmu v závislosti na N? Jaká je jeho asymptotická složitost?

```
// array has size N and contains positive integers
if(array.length == 0)
    return -1;
int val = array[0];
for(int i = 1; i < N; i++)
    if(array[i] < val)
        val = array[i];
return val;
```

Co tento algoritmus vypočte?

Řešení 10

Porovnání s nulou trvá jednu operaci. Projít pole trvá $N - 1$ operací. Celková složitost je N operací, tedy $\Theta(N)$.

Algoritmus počítá minimum z pole. Pokud je pole prázdné vrací -1.

Příklad 11

Jaká je asymptotická složitost následujícího algoritmu v závislosti na N?

```
// array has size N and contains positive integers
for (int i = 0; i < N - 1; i++)
    for (int j = 0; j < N - i - 1; j++)
        if(array[j] < array[j+1]){
            int tmp = array[j];
            array[j] = array[j+1];
            array[j+1] = tmp;
        }
```

K zamyšlení: Co tento algoritmus počítá?

Řešení 11

Vyhodnocení podmínky a prohození prvků ve vnitřním cyklu trvá $\mathcal{O}(1)$. Vnitřní cyklus proběhne $N - i - 1$ krát. Celkově tedy je třeba

$$\begin{aligned}
\sum_{i=0}^{N-2} N - i - 1 &= (N-1)\cdot(N-1) - \sum_{i=0}^{N-2} i \\
&= (N-1)\cdot(N-1) - (0 + 1 + 2 + \cdots + N - 2) \\
&= (N-1)\cdot(N-1) - \frac{(N-1)(N-2)}{2} \\
&= \frac{N\cdot(N-1)}{2} \\
&= \mathcal{O}(N^2)
\end{aligned}$$

operací.

Algoritmus řadí pole, nazývá se *bubble sort*. Více se o algoritmech řazení dozvíte na dalších přednáškách.

Příklad 12

Jaká je asymptotická složitost následujícího algoritmu v závislosti na N?

```
int[] a = new int[N];
for(i = 0; i < N; i++)
    a[i] = i;
for (i = 0; i < N; i++)
    while (a[i] > 0) {
        print(a[i]);
        a[i] = a[i]/2;      // integer division
    }
```

Řešení 12

Vytvoření pole trvá konstantní čas. Jeho naplnění hodnotami trvá N. Druhý cyklus proběhne N krát. Protože pro každé i je `a[i]` rovné i, proběhne vnořený while cyklus $log_2(i)$ krát. Celková složitost je tedy:

$$1 + N + 1 + \sum_{i=1}^{N} \log_2 i = \mathcal{O}(N \log N).$$

Příklad 13

Na výstup máme vypsat všechna kladná celá čísla, která jsou menší než dané číslo N a která ve svém binárním zápisu obsahují právě 3 jedničky.

Jaká bude asymptotická složitost efektivního algoritmu. Algoritmus lineární vůči N je neefektivní.

Příklad 14

Na obvodu kružnice jsou v libovolně v nepravidelných intervalech vyznačeny body očíslované po řadě za sebou 1, 2, ..., N. Určete počet všech takových trojúhelníků, jejichž vrcholy leží v očíslovaných bodech a které neobsahují střed kružnice jako svůj vnitřní bod. Navrhněte algoritmus a určete jeho asymptotickou složitost.

Řešte analogickou úlohu pro konvexní čtyřúhelníky.

Programovací úlohy k procvičení

`https://open.kattis.com/problems/tutorial`