

Semestrální projekt 1 – Y14TED

Asymptotická složitost algoritmů

Autor: Antonín DANĚK



<http://antonindanek.cz>



:-)

Osnova

Slide



2

- Co je to složitost algoritmu?
- Jak se počítá složitost algoritmu?
- Smysl přesného výpočtu složitosti algoritmu aneb Zjednodušení
- Zjednodušení aneb Asymptotická složitost algoritmů
- Ukázka – lineární versus kvadratický algoritmus
- Řád růstu funkcí
- Složitosti řadících algoritmů

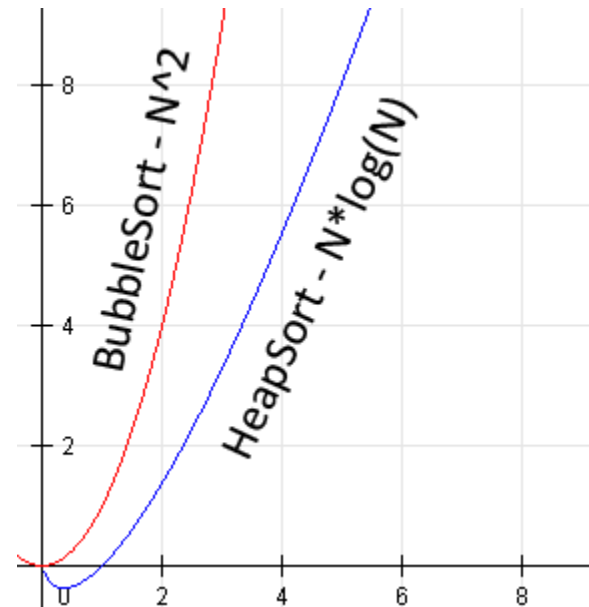
:-)

Co je to složitost algoritmu?

Slide

3

- Co je to algoritmus?
 - Postup práce (kuchařka, Java metoda)
 - Vlastnosti (konečnost)
- Dva algoritmy vykonají stejnou práci, ale trvají různě dlouho. Proč?
 - Mají různou složitost
 - Vykonávají různý počet operací



:-)

Jak se počítá složitost algoritmu?

Slide

4

- Složitost algoritmu se vyjadřuje počtem provedených základních operací.
 - Aritmetické operace
 - Podmínky (testy)
 - Přesuny v paměti

```
static int count=0;

for(int i=0; i<pole.length; i++) {
    if(pole[i]<=i) count++;
}
```

- Složitost tohoto kódu je: $1 + N + N + 0..N$
 - Nejhorší varianta: $3N + 1$
 - Nejlepší varianta: $2N + 1$

:)

Smysl přesného výpočtu složitosti algoritmu aneb Zjednodušení

„Čím výkonnější procesory nám hardwareáři dávají, tím méně softwareáři optimalizují programy pro rychlost.“

Slide

5

- Předpokládáme, že je **jedna operace** zabere procesoru **1 μ s (10⁻⁶ vteřiny)**.
- Ve skutečnosti je obvykle situace ještě několikrát lepší (4 jádrové 3GHz procesory).
 - » Program musí být pro běh na více jádrech optimalizovaný.
- **Milion operací proběhne za 1s**
 - Nemá smysl počítat jednotlivé operace.
 - Nemá smysl počítat ani 10ky, 100ky, 1000ce, 10 000ce (či 100 000ce) operací.

:)

Zjednodušení aneb Asymptotická složitost algoritmů

Slide

6

- Obvykle automaticky mluvíme o asymptotické složitosti
 - Zanedbáme operace nezávislé na datech
 - Zanedbáme dílčí jednodušší části algoritmu
 - Zanedbáme multiplikační konstanty

```
static int count=0;

for(int i=0; i<pole.length; i++) {
    if(pole[i]<=i) count++;
}
```

- Složitost tohoto kódu je: $1 + N + N + 0..N$
 - Nejhorší varianta: $3N + 1$
 - Nejlepší varianta: $2N + 1$
- Asymptotická složitost tohoto kódu je: N

:)

Ukázka – lineární versus kvadratický algoritmus - kvadratický

Slide

7

- Máme dvě pole. Program zjistí součet prvků prvního pole (**sumPole()**;) a do proměnné **count** uloží, kolikrát se takové číslo nachází v druhém poli.

POMALÉ (N²) :

```
for(int i=0; i<pole2.length; i++) {  
    if (sumPole(pole1)==pole2[i]) count++;  
}
```

Metoda **sumPole()**; projde celé první pole – její složitost je N.

- Náš cyklus prochází druhé pole a v každé iteraci volá tuto metodu. N krát tedy voláme metodu o složitosti N (Předpokládáme stejnou velikost obou polí.) → $N * N = N^2$

:)

Ukázka – lineární versus kvadratický algoritmus - lineární

- Máme dvě pole. Program zjistí součet prvků prvního pole (**sumPole()**;) a do proměnné **count** uloží, kolikrát se takové číslo nachází v druhém poli.

Slide

8

RYCHLÉ (N) :

```
int suma = sumPole(pole1);

for(int i=0; i<pole2.length; i++) {
    if(suma==pole2[i]) count++;
}
```

Zásadní rozdíl: Sumu prvního pole jsme předpočítali a uložili do proměnné. Metodu **sumPole()**; tedy voláme pouze jednou a asymptotická složitost tohoto algoritmu je N.

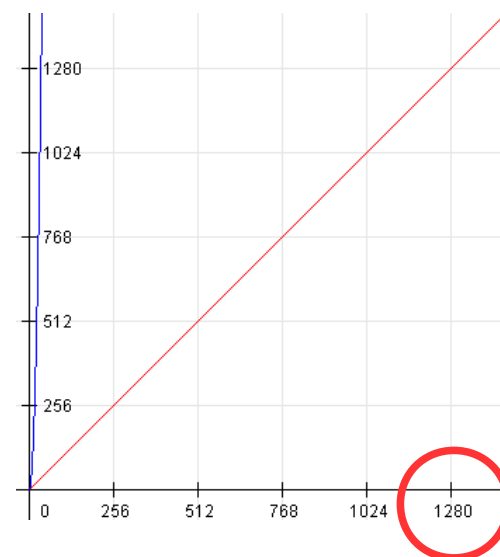
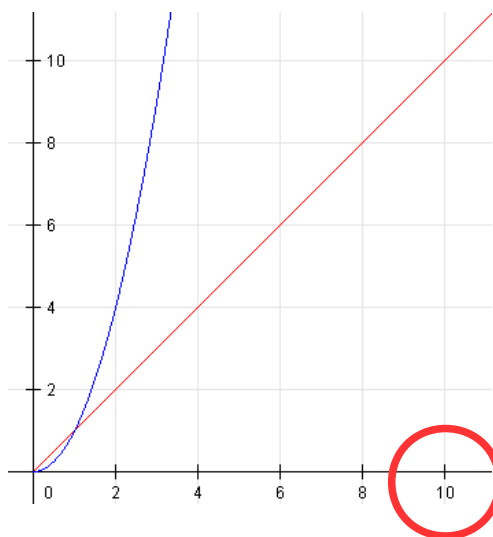
:)

Ukázka – lineární versus kvadratický algoritmus - výsledek

Slide

9

- Asymptotická složitost algoritmu by se dala vyložit jako **řád růstu času** v závislosti na datech.



- Pro milionové N:
 - už není parabola viditelná
 - „červený“ algoritmus proběhne za 1s
 - „modý“ algoritmus proběhne za 11 dní

:)

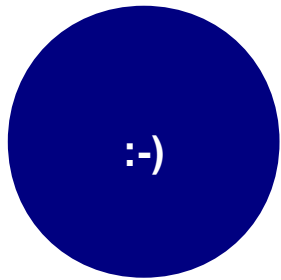
Řád růstu funkcí*

Slide

10

- Množina omega (Ω) – funkce které od určitého bodu rostou rychleji než $f(x)$ (nebo jejich násobek)
- Množina omikron (O) – funkce které od určitého bodu rostou pomaleji než $f(x)$ (nebo jejich násobek)
- Množina theta (Θ) – funkce které rostou stejně rychle jako $f(x)$

$N \in O(N^2)$	$N^2 \in \Omega(N)$
$\log(N) \in O(N)$	$N \cdot \log(N) \in \Omega(N)$



Tabulka rychlostí (jestliže jedna operace zabere procesoru 1 μ s)

- Ještě rychleji než 2^N roste $N!$

Slide

11

Vstupní data	Asymptotická složitost					
	1	N	$N \cdot \log(N)$	N^2	N^3	2^N
1	1 μ s	1 μ s	1 μ s	1 μ s	1 μ s	2 μ s
10	1 μ s	10 μ s	10 μ s	100 μ s	1ms	1ms
100	1 μ s	100 μ s	200 μ s	10ms	1s	3*10 ¹¹ dob ledových
1000	1 μ s	1ms	3ms	1s	16,6 min	... ehm
1000000	1 μ s	1s	6s	11,4 dne	31 709 let	

HeapSort

BubbleSort



:-)

Složitosti řadících algoritmů

Slide

12

- BubbleSort : N^2
 - InsertionSort : N^2
 - SelectionSort : N^2
 - QuickSort : N^2
 - MergeSort : $N \cdot \log(N)$
 - HeapSort : $N \cdot \log(N)$
-
- Velmi často potřebujeme seřadit nějaké prvky.
 - Je velmi vhodné, naučit se používat efektivní algoritmy.

:)

Konec

Slide

13

- Děkuji za pozornost.
- Máte-li dotaz – ptejte se.

- Zdroje: hlava, Wikipedia

