Jak porovnávat algoritmy

Algoritmus
Který je lepší?
Složitost
Doporučení

Jana Finkeová

Algoritmus

- program je tvořen algoritmy a datovými strukturami
  - algoritmus je pracovní postup
  - musí znát jak vypadají vstupní data
  - a jak má správně vypadat požadovaný výstup
- Kde se vzal název algoritmus?
  - v 9. století žil v Persii matematik, astronom a zeměpisec Al-Khwarizmi, který kolem roku 825 napsal knihu o počítání s indickými čísli
  - tento jeho spis byl ve 12. století přeložen do latiny pod názvem „Algoritmi de numero Indorum“
  - pojem algoritmus se ujal jako pojem pro metodu výpočtu (někteří tvrdí, že je to chyba v porozumění, někdo, že je to na počest autora)

Jana Finkeová

Datová struktura

- program je tvořen algoritmy a datovými strukturami
- datová struktura je způsob, jak si v počítači organizujeme a ukládáme data
  - příkladem může být reprezentace čísel v počítači
  - s datovými strukturami je spojena i volba algoritmu pro jejich zpracování a naopak

Jana Finkeová

Algoritmy

- existují základní problémy, které je často potřeba řešit
  - např. třídění, vyhledávání, porovnávání, fronta, zásobník…
- pro tyto problémy existují algoritmy, které se velmi často studují
- existují pro ně efektivní řešení v podobě ověřených algoritmů a k ni dobře navržených datových struktur
- velmi často je naším úkolem zvolit správný algoritmus
- jen výjimečně vytvořit algoritmus úplně nový
- jejich dobrá znalost patří k základním programátorským znalostem

Co budeme u algoritmů posuzovat? Co budeme měřit?

- čas
- prostor

Jana Finkeová

Který algoritmus je lepší?

- pro zadanou úlohu existuje obvykle více možných řešení
- podle jakých kritérií můžeme navržená řešení hodnotit
  - rychlost výpočtu: za jak dlouho získáme výsledky
  - paměťová náročnost: stačí operační paměť nebo budeme muset pracovat s daty na pevném disku?
  - doba, za jak dlouho program napíšeme: do tohoto času je potřeba započítat i odladění chyb
- první dvě kritéria jsou velmi důležitá a často se u jednotlivých algoritmů studují
- prvnímu kritériu se říká časová složitost a vypovídá o tom, jak dlouho algoritmus poběží v závislosti na velikosti vstupních dat
- druhému kritériu se říká prostorová nebo taky paměťová složitost a vypovídá o potřebné velikosti paměti pro provedení algoritmu v závislosti na velikosti vstupních dat

Jana Finkeová

Časová a paměťová složitost

spolu souvisí

příklad:

- někdy je možné dosáhnout zrychlení algoritmu tím, že si něco předpočítáme a uložíme např. do tabulky (typicky můžeme mít uschovaná předpočítaná prvočísla)
- program se potom jen podívá do tabulky a vybere správnou odpověď
- má to svá ALE: zvyšuje se tím obvykle paměťová složitost
- proto se často stává, že rychlejší algoritmy mají obvykle větší paměťovou složitost a naopak
- v dnešní době se klade větší důraz na časovou složitost, paměti bývá dost a není drahá

Jana Finkeová

Porovnávání algoritmů

- porovnávat algoritmy je možné jak prakticky, tak teoreticky
- teoretické porovnávání zahrnuje
  - odhad počtu kroků algoritmu
  - odhad velikosti potřebné paměti
- pro složité algoritmy není jednoduché odhady najít,
- pak přijde na řadu
- praktické porovnávání
  - porovnávané algoritmy naprogramovat
  - jenže tady už hodně záleží na programovacím jazyce
  - a na jeho verzi
  - a ještě více na programátorovi, i krásný algoritmus se dá .....
  - závisí na hardwaru a operačním systému
  - testování je potřeba provádět na stejném počítači a na stejných vstupních datech

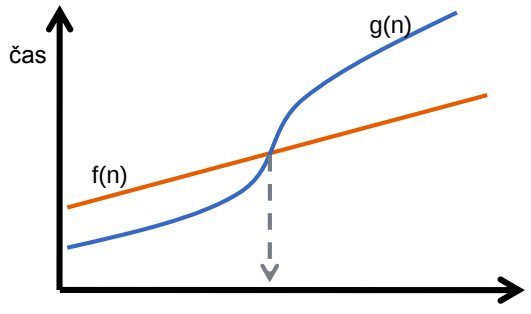
Jana Finkeová

Porovnávání algoritmů

- dalším krokem je, že musíme algoritmy porovnat na různě velkých datech
- může se stát, že jeden algoritmus je lepší pro malá data a druhý algoritmus pro větší data

Který vybrat?

- ve výsledném programu můžeme oba algoritmy zkombinovat, pro malá data použít první algoritmus, pro velká ten druhý



Jana Finkeová

Časová složitost

- Časová složitost algoritmu spuštěného na vstup *V* je počet kroků, které algoritmus provede. Časová složitost algoritmu je funkce *T*: N->N, kde *T(n)* je maximální počet kroků, který algoritmus provede na datech o velikosti *n* (pro všechny vstupy *V* o velikosti *n*)
  - velikost vstupu – musí být měřitelná a může ji např. představovat počet bitů, který je potřeba pro zápis vstupních dat do počítače
  - krok algoritmu – musí být definován, obvykle je to jedna operace či instrukce daného počítače,
    např. přiřazení, vyhodnocení podmínky, aritmetická operace, přístup do paměti, porovnání
    obvykle předpokládáme, že tyto základní operace vyžadují stejnou dobu (není to pravda)
- ještě jednou a trochu jinak
- Časová složitost je **funkce**, která vyjadřuje závislost počtu kroků algoritmu na velikosti vstupních dat.

Jana Finkeová

Časová složitost

Příklad:
```
int b;double a; double vysl=1;
a = ...; b = ...;
while (b > 0)
  {
     vysl = vysl * a;
     b = b - 1;
  }
Console.WriteLine(vysl);
```

- na čem závisí počet operací ?

*celkový počet operací = počet operací v jedné iteraci * b + 2*

pro b=5 5*3+2=17
pro b=50 =152
pro b=500 =1502

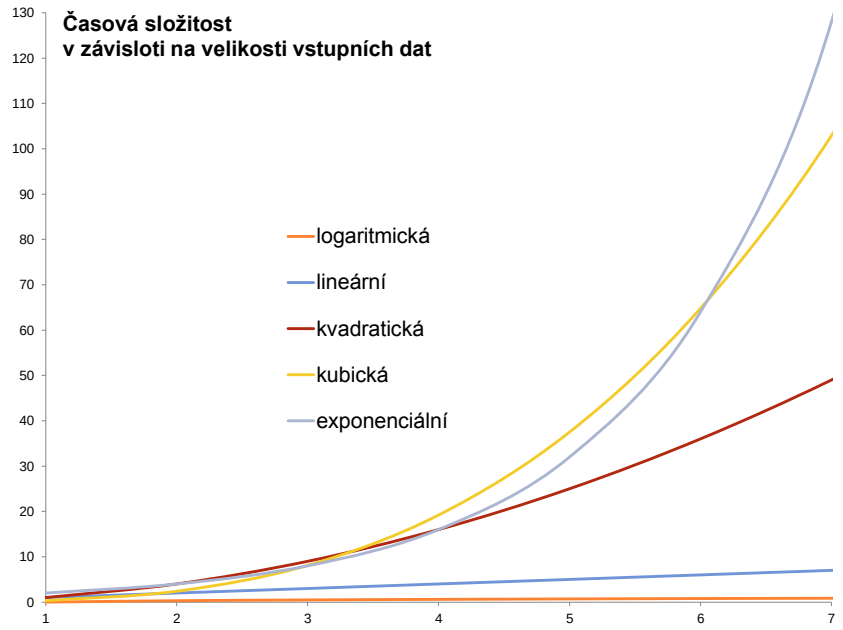
- O jakou závislost se jedná?

Jana Finkeová

Časová složitost

- Je-li velikost vstupu *n*, pak časové složitosti,
  - která je úměrná *n*, budeme říkat lineární
  - která je úměrná n^2, budeme říkat kvadratická
  - která je úměrná n^3, budeme říkat kubická
  - která je úměrná 2^n, budeme říkat exponenciální
  - která je úměrná *log(n)*, budeme říkat logaritmická
  - ...

Jana Finkeová



Jana Finkeová

Časová složitost

Odhad počtu operací pro různě velká vstupní data

	n = 10	n = 100	n = 1000	n = 1 000 000
log n	3,3	6,7	10	20
n	10	100	1000	10^6
n^2	100	10^4	10^6	10^{12}
n^3	1000	10^6	10^9	10^{18}
2^n	1024	1.10^{31}	1.10^{303}	$\sim \infty$

Odhad doby výpočtu pro různě velká vstupní data (10^9 operací za vteřinu)

	n = 10	n = 100	n = 1000	n = 1 000 000
log n	3,3 ns	6,7 ns	10 ns	20 ns
n	10 ns	100 ns	1 μs	1 ms
n^2	100 ns	10 μs	1 ms	16,5 min
n^3	1 μs	1 ms	1 s	31 let
2^n	1 μs	3.10^{14} let	3.10^{286} let	$\sim \infty$

Jak dlouho existuje planeta země?
Vyhněte se algoritmům s exponenciální časovou složitostí!

Časová složitost

- vyhněte se algoritmům s exponenciální časovou složitostí
- pokud si koupíte dvakrát rychlejší počítač, pak dokáže zpracovat dvakrát větší data jen v případě, že se jedná o algoritmus s lineární časovou složitostí

některé časové posloupnosti nemusí s velikostí vstupních dat vzrůstat rovnoměrně, ale mohou oscilovat

Příklad 1:

- algoritmus pro zjištění zda zadané číslo je prvočíslo
- postupně bude testovat, zda je dané číslo dělitelné 2, 3, ...
- pokud algoritmus **najde dělitelnost**, skončí a oznámí, že zadané číslo **není prvočíslo**
- jinak projde všechny možné dělitele a odpoví, že jde o prvočíslo
- pro každé sudé číslo skončí algoritmus hned v prvním kroku
- pro každé prvočíslo projde algoritmus všechny možné dělitele
- jaká je časová složitost?

Jana Finkeová

Časová složitost v nejlepším případě

- minimální počet operací
- to není moc užitečná informace

v našem příkladě by to odpovídalo číslu, které je dělitelné 2

Časová složitost v očekávaném případě

- průměrný počet operací pro danou úlohu
- není jednoduché rozhodnout, protože bychom potřebovali udělat pečlivý statistický rozbor toho, jak mohou vypadat vstupní data

museli bychom znát kolik prvočísel je mezi přirozenými čísli, jaký je jejich podíl

Časová složitost v nejhorším případě

- maximální počet kroků
- to je informace, kterou potřebujeme a je nám užitečná
- tu potřebujeme

v naší úloze to odpovídá situaci, kdy bylo zadáno prvočíslo

Časová složitost v nejhorším případě

Příklad 2: hledání minima

- máme pole hodnot
- máme najít nejmenší hodnotu
- postupně budeme procházet celé pole hodnot
- za krok algoritmu můžeme zvolit podmínku, kde testujeme prvek pole, zda je menší než minimální hodnota
- pouze testovací podmínka proběhne (n-1) krát
- proto je časová složitost $O(n)$

Příklad 3: sečtení prvků v matici

- máme matici n x n
- musíme projít všechna čísla v matici
- proto je časová složitost $O(n^2)$

Asymptotická časová složitost

- při určování asymptotické časové složitosti nás zajímá chování algoritmů na hodně velkých datech
- asymptotická časová složitost může být definována pomocí limity, kdy n -> ∞
- algoritmy, které se liší jen multiplikativní konstantou při použití na velkých datových souborech jaksi „splynou“
- zůstane propastný rozdíl mezi algoritmy, jejichž časovou složitost lze vyjádřit pomocí funkcí *log n, n a* 2^n
- ostatní algoritmy můžeme zařadit do stejné třídy

Jana Finkeová

Typické příklady

- O(1) skok na prvek v poli dle indexu
- O($\log_2$ N) vyhledání prvku v seřazeném poli metodou půlení intervalu
- O(N) vyhledání prvku v neseřazeném poli lineárním (sekvenčním) vyhledáváním
- O(N log N) seřazení pole prvků s definovanou nerovností (např. reálná čísla podle velikosti, např. algoritmem Mergesort)
- O(N^2) diskrétní Fourierova transformace (DFT)
- O(2^N) přesné řešení problému obchodního cestujícího

Jana Finkeová

Poučení

- pro vyřešení jedné a téže úlohy většinou existuje více algoritmů
- na nás je zvolit ten správný
- hledejte informace v literatuře