CZ.1.07/2.2.00/28.0041

Centrum interaktivních a multimediálních studijních opor pro inovaci výuky a efektivní učení

evropský sociální fond v ČR

EVROPSKÁ UNIE

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

OP Vzdělávání pro konkurenceschopnost

UNIVERSITAS MASARYKIANA BRUNENSIS

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Časť I

Návrh a analýza algoritmov

1. Analýza zložitosti, korektnosť
   - Lineárne vyhľadávanie
   - Triedenie vkladním
2. Asymptotická notácia

Vyhľadávanie prvku

Vstup postupnosť prvkov $(a_1, \ldots, a_n)$ a prvok x

Výstup index i taký, že $a_i = x$, resp. hodnota $\textsc{No}$ ak prvok x sa v postupnosti nevyskytuje

Linear Search

Procedure Linear Search

Vstup (A, n, x) // A: pole; n: počet prvkov v poli; x: hľadaná hodnota
Výstup index i taký, že $A[i] = x$ resp. hodnota No

```
answer ← No
for i = 1 to n do
    if A[i] = x then answer ← i fi
od
return answer
```

- zápis algoritmu: priraďovací príkaz, cyklus, podmienený príkaz
- vždy prehľadá celé pole = neefektívne

BETTER LINEAR SEARCH

Procedure BETTER LINEAR SEARCH

Vstup a Výstup rovnaké ako pre LINEAR SEARCH

```
1 for i = 1 to n do
2     if A[i] = x then return i fi
3 od
4 return NO
```

- prvý výskyt x ukončí prehľadávanie
- každý prechod cyklom znamená 2 testy: v riadku 1 testujeme, či $i \leq n$ a v riadku 2 testujeme, či $A[i] = x$
- stačí 1 test?

Sentinel Linear Search

Procedure Sentinel Linear Search

Vstup a Výstup rovnaké ako pre Linear Search

1 $last \leftarrow A[x]$
2 $A[n] \leftarrow x$
3 $i \leftarrow 1$
4 **while** $A[i] \neq x$ **do** $i \leftarrow i + 1$ **od**
5 $A[n] \leftarrow last$
6 **if** $i < n \vee A[n] = x$
7   **then return** i
8   **else return** No **fi**

- prvý výskyt x ukončí prehľadávanie
- sentinel pre prípad, že pole neobsahuje prvok x
- každý prechod cyklom znamená 1 test
- 2 testy na záver (riadok 6)

Časová zložitosť

- časová zložitosť je funkcia veľkosti vstupu
- vstupom je pole obsahujúce n prvkov a prvok x
- *búno* veľkosť vstupu je n (zanedbávame prvok x a veľkosť prvkov)
- každá aritmetická operácia vyžaduje konštantný čas (bez ohľadu na veľkosť prvkov)

Časová zložitosť LINEAR SEARCH

```
1 answer ← NO
2 for i = 1 to n do
3     if A[i] = x
4        then answer ← i fi od
5 return answer
```

- časová zložitosť kroku i je t_i
- krok 1 a 5 sa vykonajú len raz
- krok 2 sa vykoná $n+1$ krát
- krok 3 sa vykoná n krát
- krok 4 sa vykoná toľkokrát, aký je počet výskytov x v poli

časová zložitosť LINEAR SEARCH sa pohybuje medzi dolnou hranicou

$$t_1 + t_2 \cdot (n+1) + t_3 \cdot n + t_4 \cdot 0 + t_5$$

a hornou hranicou

$$t_1 + t_2 \cdot (n+1) + t_3 \cdot n + t_4 \cdot n + t_5$$

Časová zložitosť LINEAR SEARCH

- dolná aj horná hranica majú tvar $c \cdot n + d$, kde c a d sú konštanty nezávislé na n
- časová zložitosť LINEAR SEARCH je zdola ohraničená lineárnou funkciou premennej n a zároveň je aj zhora ohraničená lineárnou funkciou
- pre označenie takejto situácie používame špeciálnu notáciu $\Theta(n)$
- časová zložitosť LINEAR SEARCH je $\Theta(n)$

Časová zložitosť Better Linear Search

for $i = 1$ **to** n **do if** $A[i] = x$ **then return** i **fi od**
return No

- časová zložitosť je v najhoršom prípade $\Theta(n)$
- časová zložitosť je v najlepšom prípade $\Theta(1)$
- nie je pravda, že časová zložitosť je $\Theta(n)$ a zároveň nie je ani $\Theta(1)$

Časová zložitosť BETTER LINEAR SEARCH

```
for i = 1 to n do if A[i] = x then return i fi od
return No
```

- časová zložitosť je v najhoršom prípade $\Theta(n)$
- časová zložitosť je v najlepšom prípade $\Theta(1)$
- nie je pravda, že časová zložitosť je $\Theta(n)$ a zároveň nie je ani $\Theta(1)$
- časová zložitost BETTER LINEAR SEARCH je zhora ohraničená lineárnou funkciou; pre označenie takejto situácie používame notáciu $\mathcal{O}(n)$
- časová zložitosť BETTER LINEAR SEARCH je $\mathcal{O}(n)$
- časová zložitost BETTER LINEAR SEARCH je zdola ohraničená konštantnou funkciou; pre označenie takejto situácie používame notáciu $\Omega(1)$

spoločné označenie pre Θ, $\mathcal{O}$ a Ω je *asymptotická notácia*

Časová zložitosť Sentinel Linear Search

- procedúry Better Linear Search a Sentinel Linear Search majú rovnakú asymptotickú zložitosť $\mathcal{O}(n)$
- procedúra Sentinel Linear Search má lepší konštantný faktor

Korektnosť algoritmu

čiastočná korektnosť ak výpočet skončí, poskytne korektný výstup

úplnosť výpočet pre každú vstupnú inštanciu skončí

Korektnosť *iteratívneho* algoritmu

indukčné dokazovanie

- postupne analyzujeme všetky cykly, u vnorených cyklov začíname od cyklu najhlbšej úrovne
- pre každý cyklus určíme jeho invariant, ktorý platí počas celého výpočtu cyklu a vyjadruje efekt cyklu
- dokážeme, že invariant cyklu je pravdivý
- využitím invariantu
  - dokážeme konečnosť algoritmu
  - dokážeme správnosť vypočítaného výsledku

Platnosť invariantu cyklu

Inicializácia invariant je platný pred začiatkom cyklu

Iterácia ak invariant platí pred iteráciou cyklu, zostáva v platnosti aj pred nasledujúcou iteráciou

Ukončenie cyklus skončí, po ukončení platí invariant a garantuje požadovaný efekt cyklu

Korektnosť Better Linear Search

Invariant cyklu

Na začiatku každej iterácie cyklu platí, že ak prvok x sa nachádza v poli A, tak sa nachádza v časti poľa medzi pozíciami i a n.

Inicializácia Na začiatku je $i = 1$ a preto tvrdenie platí

Iterácia Predpokladajme, že na začiatku iterácie pre hodnotu i je prvok x na pozícii medzi i a n. Ak iterácia nevráti výslednú hodnotu, tak $A[i] \neq x$. Preto x musí byť na niektorej z pozícii medzi $i + 1$ a n a invariant zostáva v platnosti aj pred nasledujúcou iteráciou cyklu.

Ukončenie Cyklus skončí buď preto, že vráti výslednú hodnotu alebo preto, že $i > n$. Ak cyklus skončí vrátením hodnoty indexu i, správnosť vypočítaného výsledku je zrejmá. V opačnom prípade korektnosť vyplýva z (obráteného) invariantu: *Ak prvok x sa nenachádza na pozícii medzi i a n, tak prvok x sa nenachádza v poli A.* Pretože $i > n$, tak (obrátený) invariant platí a odpoveď No je správna.

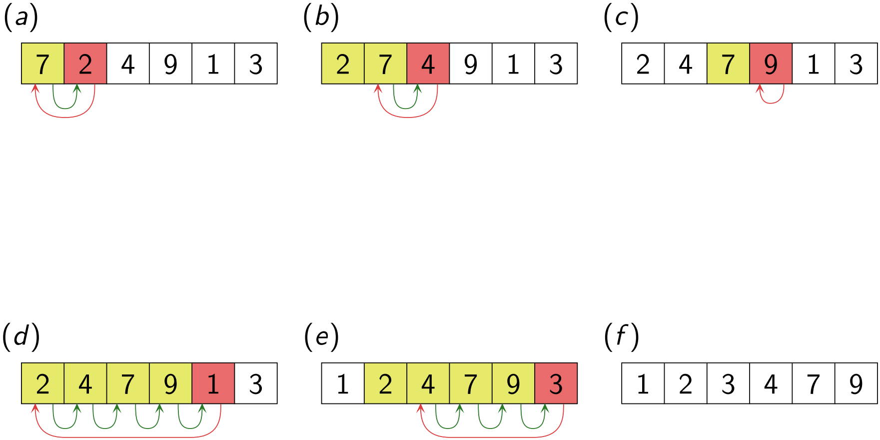
1. Analýza zložitosti, korektnosť
   - Lineárne vyhľadávanie
   - Triedenie vkladním
2. Asymptotická notácia

Problém triedenia

Vstup: postupnosť n čísel $(a_1, a_2, \ldots, a_n)$

Výstup: Permutácia (preusporiadanie) $(a'_1, a'_2, \ldots, a'_n)$ vstupnej postupnosti také, že $a'_1 \leq a'_2 \leq \ldots \leq a'_n$

Princíp triedenia vkladaním



Algoritmus

INSERTION SORT(A)

```
for j = 2 to A.length do
    key ← A[j]
    // Vlož A[j] do utriedenej postupnosti A[1 . . . j − 1]
    i ← j − 1
    while i > 0 ∧ A[i] > key do
        A[i + 1] ← A[i]
        i ← i − 1 od
    A[i + 1] ← key od
```

Korektnosť

Invariant cyklu

Na začiatku každej iterácie **for** cyklu (r. 1 - 8) obsahuje pole $A[1 \dots j-1]$ tie isté prvky ako na začiatku výpočtu, ale v utriedenom poradí.

Inicializácia Pred prvou iteráciou je $j = 2$ a tvrdenie platí.

Iterácia V tele cyklu sa prvky $A[j-1], A[j-2], A[j-3]$ atď posúvajú o jednu pozíciu doprava až kým sa nenájde vhodná pozícia pre prvok $A[j]$ (riadok 8). Pole $A[1 \dots j]$ preto na konci cyklu obsahuje tie isté prvky ako na začiatku výpočtu, ale v utriedenom poradí. Po zvýšením hodnoty j platí invariant aj pred nasledujúcou iteráciou cyklu.

Ukončenie Cyklus skončí ak $j > A.length = n$. Pretože v každej iterácii sa hodnota j zvyšuje o 1, musí platiť $j = n + 1$ a preto $A[1 \dots n]$ obsahuje pôvodnú postupnosť v utriedenom poradí.

Zložitosť

INSERTION SORT(A)	cena	počet
for $j = 2$ **to** $A.length$ **do**	c_1	n
$key \leftarrow A[j]$	c_2	$n-1$
//Vlož $A[j]$ do utriedenej postupnosti $A[1 \ldots j-1]$		
$i \leftarrow j - 1$	c_4	$n-1$
while $i > 0 \wedge A[i] > key$ **do**	c_5	$\sum_{j=2}^{n} t_j$
$A[i+1] \leftarrow A[i]$	c_6	$\sum_{j=2}^{n}(t_j - 1)$
$i \leftarrow i - 1$ **od**	c_7	$\sum_{j=2}^{n}(t_j - 1)$
$A[i+1] \leftarrow key$ **od**	c_8	$n-1$

$$
\begin{aligned}
T(n) = & c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n}(t_j - 1) \\
& + c_7 \sum_{j=2}^{n}(t_j - 1) + c_8(n-1)
\end{aligned}
$$

Zložitosť v najlepšom prípade

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1) \\ &= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8) \end{aligned}$$

- Zložitosť v najlepšom prípade môžeme vyjadriť ako funkciu $an + b$, kde a, b sú konštanty nezávislé na n.
- Zložitosť Insert Sort je $\Omega(n)$.

Zložitosť v najhoršom prípade

$$
\begin{aligned}
T(n) = {} & c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) \\
& + c_6\left(\frac{n(n+1)}{2}\right) + c_7\left(\frac{n(n+1)}{2}\right) + c_8(n-1) \\
= {} & \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right)n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2}\right)n \\
& - (c_2 + c_4 + c_5 + c_8)
\end{aligned}
$$

- Zložitosť v najhoršom prípade môžeme vyjadriť ako funkciu $an^2 + bn + c$, kde a, b, c sú konštanty nezávislé na n.
- Zložitosť INSERT SORT je $\mathcal{O}(n^2)$.

1. Analýza zložitosti, korektnosť

2. Asymptotická notácia
   - Θ notácia
   - $\mathcal{O}$ notácia
   - Ω notácia

Asymptotická notácia

- asymptotickú notáciu využívame pri určovaní časovej zložitosti algoritmov
- umožňuje abstrahovať od detailov skutočnej zložitosti
- využíva sa aj pri charakterizácií iných zložitostných kritérií ako napr. pamäťová zložitosť
- pozor: aj pri využití asymptotickej notácie musíme presne špecifikovať, či sa jedná o zložitosť v najhoršom, najlepšom, priemernom prípadne očakávanom prípade
- ak nie je uvedené inak, určujeme zložitosť v najhoršom prípade

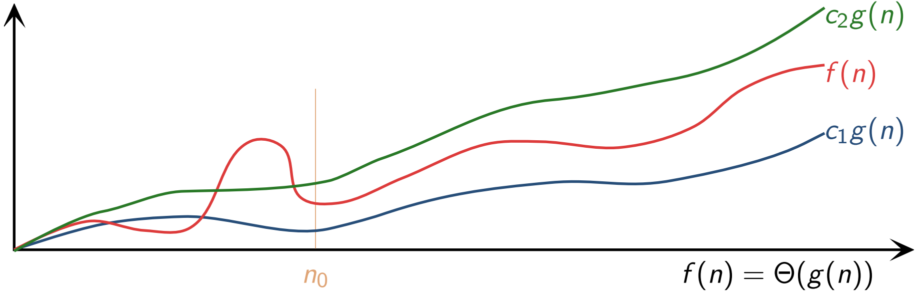
1. Analýza zložitosti, korektnosť
2. Asymptotická notácia
   - Θ notácia
   - $\mathcal{O}$ notácia
   - Ω notácia

Θ notácia

Definícia

Pre danú funkciu $g(n)$ označuje symbol $\Theta(g(n))$ *množinu* funkcií

$$\Theta(g(n)) = \{f(n) | \text{ existujú kladné konštantny } c_1, c_2 \text{ a } n_0 \text{ také, že } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ pre všetky } n \ge n_0$$



Θ notácia - príklad 1

- ukážeme, že $\frac{1}{2}n^2 - 3n = \Theta(n^2)$
- musíme nájsť **kladné** konštanty c_1, c_2 a n_0 také, že

$$c_1 n^2 \leq \frac{1}{2}n^2 - 3n \leq c_2 n^2 \tag{1}$$

  paltí pre všetky $n \geq n_0$
- po úprave dostávame

$$c_1 \leq \frac{1}{2} - \frac{3}{n} \leq c_2$$

- pravá nerovnosť platí pre každé $n \geq 1$ ak zvolíme $c_2 \geq 1/2$
- ľavá nerovnosť platí pre každé $n \geq 7$ ak zvolíme $c_1 \leq 1/14$
- ak zvolíme $c_1 = 1/14$, $c_2 = 1/2$ a $n_0 = 7$, tak platí požadovaná nerovnosť 1

Θ notácia - príklad 2

- ukážeme, že $6n^3 \neq \Theta(n^2)$
- predpokladajme, že existujú také konštanty c_2 a n_0, že pre všetky $n \geq n_0$ platí $6n^3 \leq c_2 n^2$
- potom musí platiť, že $n \leq c_2/6$, čo nie je možné, pretože c_2 je konštanta

Θ notácia - poznámky

- namiesto zápisu $f(n) \in \Theta(g(n))$ používame zápis $f(n) = \Theta(g(n))$ (*historické a praktické dôvody*)
- ak $f(n) = \Theta(g(n))$, tak funkcie $f(n)$ a $g(n)$ rastú *asymptoticky rovnako rýchlo*

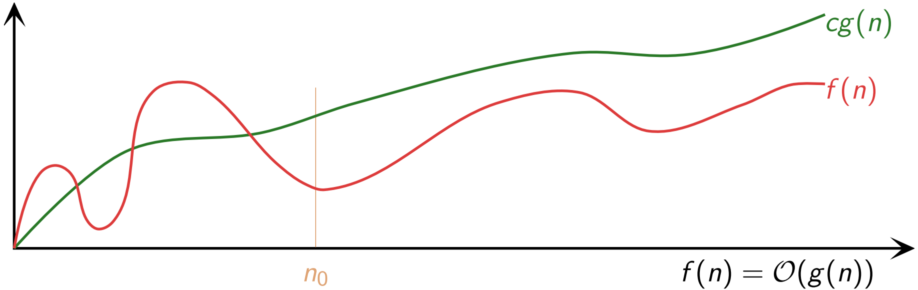
1. Analýza zložitosti, korektnosť
2. Asymptotická notácia
   - Θ notácia
   - $\mathcal{O}$ notácia
   - Ω notácia

$\mathcal{O}$ notácia

Definícia

Pre danú funkciu $g(n)$ označuje symbol $\mathcal{O}(g(n))$ *množinu* funkcií

$$\mathcal{O}(g(n)) = \{f(n) | \text{ existujú kladné konštantny } c \text{ a } n_0 \text{ také, že } 0 \leq f(n) \leq cg(n) \text{ pre všetky } n \geq n_0\}$$



$\mathcal{O}$ notácia - poznámky

- analogicky ako pre Θ notáciu používame zápis $f(n) = \mathcal{O}(g(n))$
- ak $f(n) = \mathcal{O}(g(n))$, tak funkcia $g(n)$ rastie *asymptoticky rýchlejšie* ako funkcia $f(n)$

- platí $\Theta(g(n)) \subseteq \mathcal{O}(g(n))$
- platí $n = \mathcal{O}(n^2)$

1. Analýza zložitosti, korektnosť

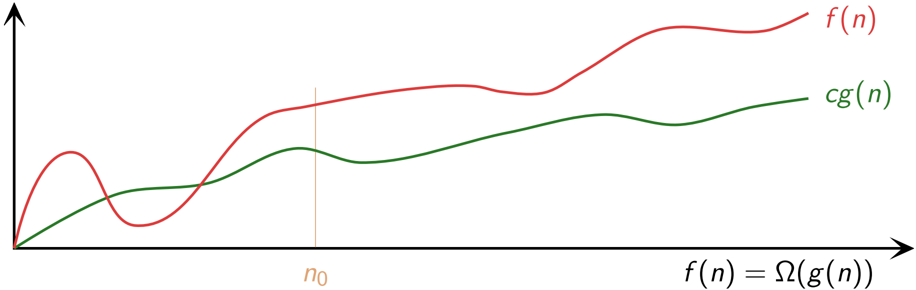
2. Asymptotická notácia
   - Θ notácia
   - $\mathcal{O}$ notácia
   - Ω notácia

Ω notácia

Definícia

Pre danú funkciu $g(n)$ označuje symbol $\Omega(g(n))$ *množinu* funkcií

$$\Omega(g(n)) = \{f(n) | \text{ existujú kladné konštantny } c \text{ a } n_0 \text{ také, že } 0 \leq f(n) \leq cg(n) \text{ pre všetky } n \geq n_0\}$$



Ω notácia - poznámky

- analogicky ako pre Θ notáciu používame zápis $f(n) = \Omega(g(n))$
- ak $f(n) = \Omega(g(n))$, tak funkcia $g(n)$ rastie *asymptoticky pomalšie* ako funkcia $f(n)$

- $f(n)) = \Theta(g(n))$ vtedy a len vtedy ak $f(n)) = \mathcal{O}(g(n))$ a zároveň $f(n)) = \Omega(g(n))$

Donald E. Knuth: *Big Omicron and big Omega and big Theta*.
ACM SIGACT, Volume 8 Issue 2, April-June 1976, pp. 18 - 24, ACM New York, NY, USA.

Asymptotická notácia - test

Nech $T(n) = \frac{1}{2}n^2 + 3n$. Ktoré z nasledujúcich tvrdení sú pravdivé?

- $T(n) = \mathcal{O}(n)$
- $T(n) = \Omega(n)$
- $T(n) = \Theta(n^2)$
- $T(n) = \mathcal{O}(n^3)$

Asymptotická notácia - test

Nech $T(n) = \frac{1}{2}n^2 + 3n$. Ktoré z nasledujúcich tvrdení sú pravdivé?

- $T(n) = \mathcal{O}(n)$
- $T(n) = \Omega(n)$ $\qquad n_0 = 1, c = 4$
- $T(n) = \Theta(n^2)$ $\qquad n_0 = 1, c_1 = 1, c_2 = 4$
- $T(n) = \mathcal{O}(n^3)$ $\qquad n_0 = 1, c = 1/2$

Asymptotická notácia - vlastnosti

tranzitivita

$$f(n) = \Theta(g(n)) \text{ a } g(n) = \Theta(h(n)) \text{ implikuje } f(n) = \Theta(h(n))$$
$$f(n) = \mathcal{O}(g(n)) \text{ a } g(n) = \mathcal{O}(h(n)) \text{ implikuje } f(n) = \mathcal{O}(h(n))$$
$$f(n) = \Omega(g(n)) \text{ a } g(n) = \Omega(h(n)) \text{ implikuje } f(n) = \Omega(h(n))$$

reflexivita

$$f(n) = \Theta(f(n)) \qquad \text{podobne pre } \mathcal{O} \text{ a } \Omega$$

symetria

$$f(n) = \Theta(g(n)) \text{ vtedy a len vtedy ak } g(n) = \Theta(f(n))$$

transpozícia

$$f(n) = \mathcal{O}(g(n)) \text{ vtedy a len vtedy ak } g(n) = \Omega(f(n))$$

poznámka: nie každá dvojica funkcií je asymptoticky porovnateľná

Používaná notácia

celá časť reálneho čísla x

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

modulárna aritmetika

$$a \mod n = a - n\lfloor a/n \rfloor$$

logaritmus

$$\log n = \log_2 n \quad \text{(dvojkový logaritmus)}$$
$$\ln n = \log_e n \quad \text{(prirodzený logaritmus)}$$