IB111 Programování a algoritmizace

Radek Pelánek
Zdeněk Říha

Programování a algoritmizace

- Úvod do programátorského a algoritmického stylu myšlení
- Obecné principy
  - použitelné v řadě programovacích jazyků
- Základní datové struktury a algoritmy
- Přehled existujících programovacích jazyků a jejich výhod/nevýhod

Datové struktury a algoritmy

- Datové struktury (seznamy, zásobníky, fronty, grafy, stromy)
- Grafové algoritmy (vzdálenosti, cesty)
- Řazení (bubblesort, insertsort, quicksort, mergesort)
- Jiné zajímavé problémy (teorie čísel, řetězce)
- Rekurze
- Hrubá síla, heuristiky

Přednáška a cvičení

- 2 hodiny přednášky
  - Nepovinné, UT 12 - 13:50, učebna B410
- 2 hodiny cvičení u počítače
  - Účast povinná, UT 14:00 – 15:50, učebna B311
  - Programování v C/C++ (MS Visual Studio)
    - Pro základní algoritmy je ovšem volba konkrétního jazyka relativně nepodstatná
    - Jde o algoritmy samotné, ne o mistrovství v C/C++
    - Více o programování v C/C++ viz. PB071, PB161

Ukončení předmětu

- Zkouška
  - Písemná, zkouší se principy, algoritmy, pojmy
  - NE programování na počítači
- Připuštění ke zkoušce – úkoly ze cvičení
  - Účast
  - Kód ze cvičení
  - Domácí úkoly
    - Po 10 bodech, požadováno celkem 75 % bodů.

Literatura

- Cormen, Thomas H. Introduction to algorithms [2nd ed.]. 2nd ed. Cambridge : MIT Press, 2001. xxi, 1180. ISBN 0-262-03293-7.
- Levitin, Anany. Introduction to the design & analysis of algorithms. 2nd ed. Boston : Addison-Wesley Publishing Company, 2007. xxiii, 562. ISBN 0321364139.
- Piotr Wróblewski, Algoritmy (Datové struktury a programovací techniky), Computer Press, prodejní kód: K0871, ISBN: 80-251-0343-9
- Materiály předmětu IB002

Algoritmus

- Algoritmus je návod/postup jak vyřešit určitý typ úlohy/problému
- Algoritmus převádí vstup na výstup
  - Například
    - Součet dvou čísel: vstupem jsou dvě čísla; výstupem je číslo představující jejich součet
    - Nalezení cesty v grafu: vstupem je graf, počáteční a koncový vrchol; výstupem je cesta
    - Řazení jmen souborů: vstupem je seznam jmen souborů v adresáři, výstupem je seznam setříděný podle abecedy

Co je algoritmus?

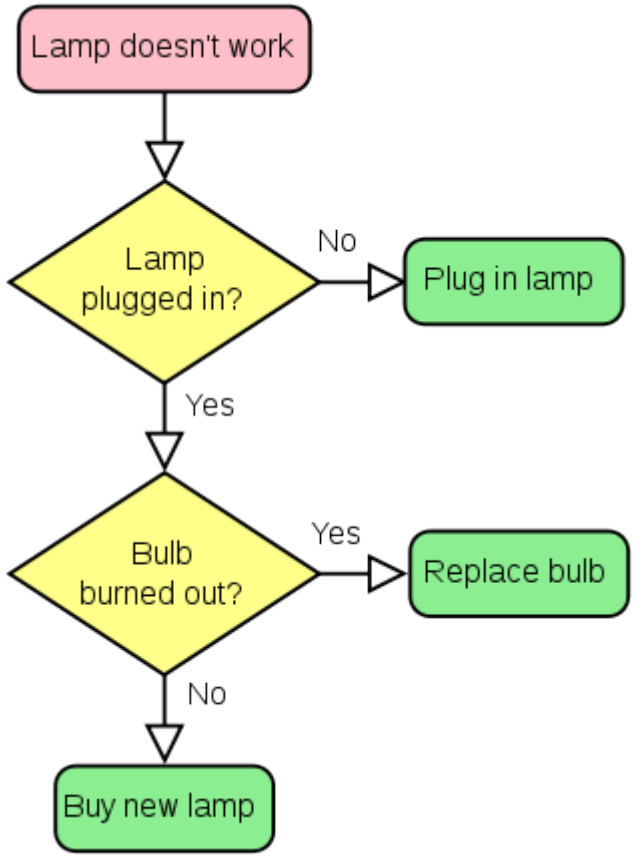
- Algoritmem může být recept, proces, metoda, procedura pokud je:
  - *Obecný* – neřeší jen konkrétní instanci problému (např. 1+1), ale obecnější třídu úloh (x+y)
  - *Srozumitelný* – popis algoritmu je dostatečně srozumitelný a přesný
  - *Konečný* – algoritmus musí skončit po určitém konečném počtu kroků
  - *Deterministický* – vždy je přesně a jednoznačně dané jak postupovat dále
  - *Má vstup a výstup* – je definováno co je validním vstupem a pro validní vstup algoritmus produkuje správný výstup

Historické aspekty

- Perský astronom a matematik
  Abū ‘Abd Allāh Muhammad ibn Mūsā al-Khwārizmī
- Roku 825 napsal dílo ‘Počítání s arabskými číslicemi ’
- Ve 12. století přeloženo do latiny jako ‘*Algoritmi de numero Indorum‘,* přičemž Algoritmi vzniklo přepisem autorova jména al-Khwārizmī bylo však chápáno jako množné číslo od ‘algorismus’ (způsob výpočtu).

Zápis algoritmu

- Algoritmus může být zapsán:
  - Přirozeným jazykem
  - Strukturovaným jazykem
  - Vývojovým diagramem (graficky)
  - Programovacím jazykem

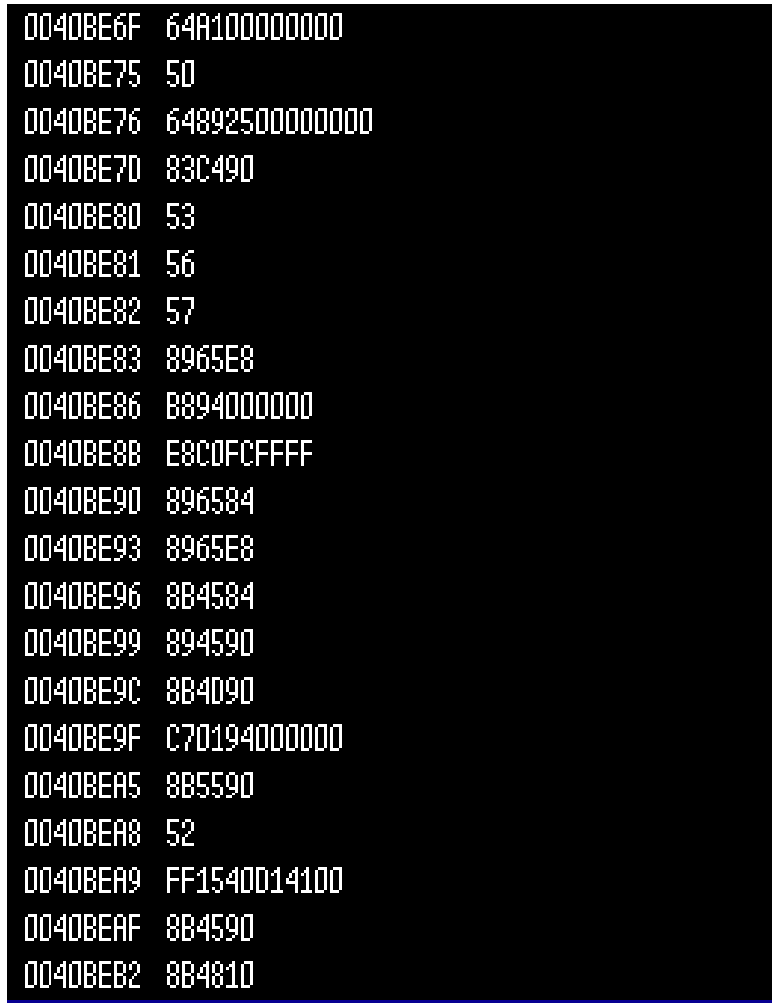


Programovací jazyky

- Strojový kód
- Assembler (jazyk symbolických adres)
- Jazyky vyšší úrovně (C/C++, Java, Pascal)
- Jazyky pro specifické aplikace (SQL, Matlab, CSS)
- Programování pomocí omezujících podmínek (Prolog)

Strojový kód

- Specifický pro určitý HW
  - Přímo instrukce procesoru
  - Instrukce skoku obsahují relativní nebo absolutní adresy
- Není lidsky čitelné



Assembler

- Příkazy odpovídají instrukcím procesoru
- Možnost používat návěstí, vkládat data
- Lidsky čitelné

```
; check that we really got 8 zeros and ones
MOV CX, 8
MOV SI, OFFSET s1
check_s:
        CMP [SI], 0
        JNE ok0
        MOV flag, 1     ; terminated.
        JMP convert
    ok0:
        CMP [SI], 'b'
        JNE ok1
        MOV flag, 1     ; terminated.
        JMP convert
    ok1:
        ; wrong digit? Not 1/0?
        CMP [SI], 31h
        JNA ok2
        JMP error_not_valid
    ok2:
        INC SI
    LOOP check_s
```

Jazyk vyšší úrovně

- Konstrukce vyšší úrovně
  - Výrazy
  - Cykly
  - Proměnné
- Pro programátory jednodušší

```
program mine(output);

var i : integer;

procedure print(var j: integer);

  function next(k: integer): integer;
  begin
    next := k + 1
  end;

begin
  writeln('The total is: ', j);
  j := next(j)
end;

begin
  i := 1;
  while i <= 10 do print(i)
end.
```

Jazyk pro specifické použití

- Například SQL (structured query language) pro databázové systémy
  - Odhlížíme od konkrétního uložení dat

```
SELECT Book.title,
       count(*) AS Authors
    FROM Book
       JOIN Book_author ON Book.isbn = Book_author.isbn
    GROUP BY Book.title;
```

Omezující podmínky

- Programování pomocí omezujících podmínek (např. Prolog)

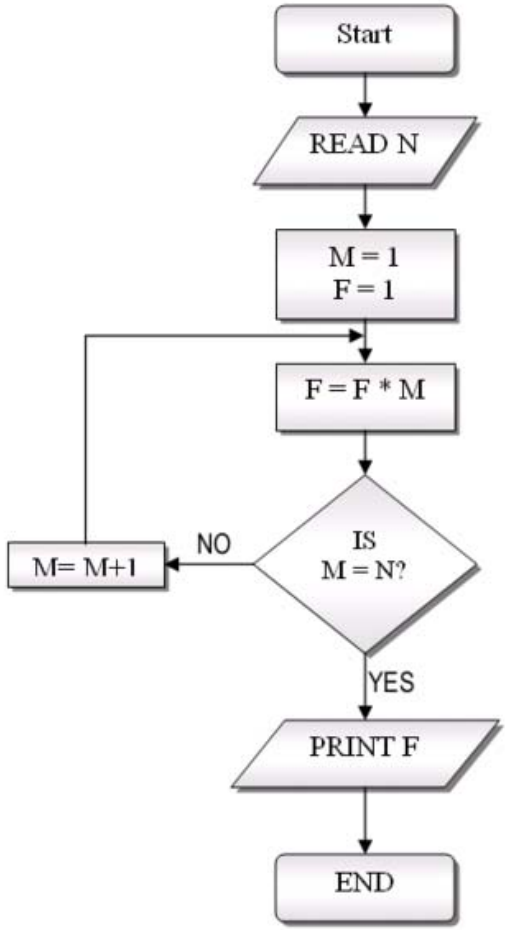
```
adjacent(1,2).          adjacent(2,1).    color(1,red,a).    color(1,red,b).
adjacent(1,3).          adjacent(3,1).    color(2,blue,a).   color(2,blue,b).
adjacent(1,4).          adjacent(4,1).    color(3,green,a).  color(3,green,b).
adjacent(1,5).          adjacent(5,1).    color(4,yellow,a). color(4,blue,b).
adjacent(2,3).          adjacent(3,2).    color(5,blue,a).   color(5,green,b).
adjacent(2,4).          adjacent(4,2).
adjacent(3,4).          adjacent(4,3).
adjacent(4,5).          adjacent(5,4).
```

```
conflict(R1,R2,Coloring) :-
   adjacent(R1,R2),
   color(R1,Color,Coloring),
   color(R2,Color,Coloring).
```

```
?- conflict(R1,R2,b).
R1 = 2    R2 = 4
?- conflict(R1,R2,b),color(R1,C,b).
R1 = 2    R2 = 4    C = blue
```

Vývojové diagramy (flowcharts)

- Začátek a konec
  - Kruh, zaokrouhlený obdélník
- Vstup a výstup
  - Kosodélník
- Příkaz
  - Obdélník
- Podmínka
  - Kosočtverec
- Šipky značí pořadí vykonávání



Pseudokód

- Neformální zápis využívající strukturovaný zápis jako u programovacích jazyků

Algorithm 1.7 BRUTE-FORCE PRIMALITYTEST
Input: A positive integer $n \geq 2$.

Output: *true* if n is prime and *false* otherwise.

```
1. s ← ⌊√n⌋
2. for j ← 2 to s
3.       if j divides n then return false
4. end for
5. return true
```

M. H. Alsuwaiyel: Algorithms, Design Techniques and Analysis.

Analýzy algoritmů

- Jak dobrý je daný algoritmus?
  - Je správný/korektní?
  - Časová náročnost
  - Paměťová náročnost
- Existuje lepší algoritmus?
  - Minimální náročnost
  - Optimalita

Rychlost algoritmů

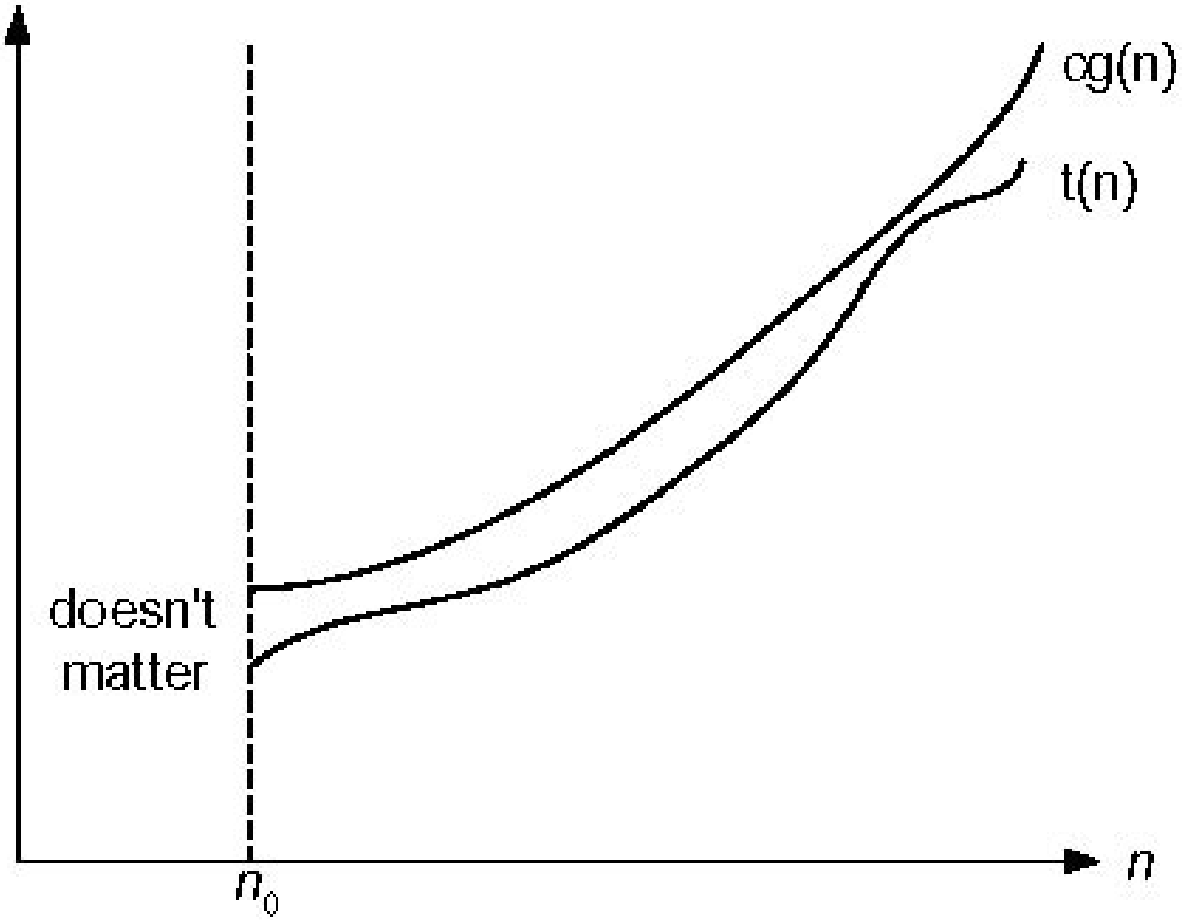
- Časová a paměťová náročnost algoritmu je obvykle závislá na velikosti vstupu.
- V závislosti na tom, co algoritmus provádí volíme základní operaci (např. porovnání, přiřazení) a náročnost vyjadřujeme v počtu těchto operací

Problém	*Velikost měříme podle*	*Základní operace*
Vyhledání klíče v seznamu n prvků	**Počet prvků v seznamu - n**	**Porovnání klíče**
Násobení dvou matic desetických čísel	**Velikost matic**	**Násobení desetiných čísel**
Výpoočet a^n	***n***	**Násobení desetiných čísel**
Grafové problémy	**Počet vrcholů nebo hran v grafu**	**Návštěva vrcholu nebo hrany**

Asymptotická složitost

- Asymptotická složitost je způsob jak porovnávat složitosti tak abychom mohli ignorovat konstantní faktory a malé vstupy
- Používáme 3 značení:
  - **O(g(n)): třída funkcí které neroste rychleji než g(n)**
  - Θ (g(n)): třída funkcí, která roste stejně rychle jako g(n)
  - Ω(g(n)): třída funkcí, která roste minimálně tak rychle jako g(n)

O(n)



Řádová složitost

1	Konstantní
log n	Logaritmická
n	Lineární
n log n	n log n
n^2	Kvadratická
n^3	Kubická
2^n	Exponenciální
$n!$	Faktoriál

Příklad délky běhu algoritmů

n	$\log n$	n	$n \log n$	n^2	n^3	2^n
8	3 nsec	0.01 μ	0.02 μ	0.06 μ	0.51 μ	0.26 μ
16	4 nsec	0.02 μ	0.06 μ	0.26 μ	4.10 μ	65.5 μ
32	5 nsec	0.03 μ	0.16 μ	1.02 μ	32.7 μ	4.29 sec
64	6 nsec	0.06 μ	0.38 μ	4.10 μ	262 μ	5.85 cent
128	0.01 μ	0.13 μ	0.90 μ	16.38 μ	0.01 sec	10^{20} cent
256	0.01 μ	0.26 μ	2.05 μ	65.54 μ	0.02 sec	10^{58} cent
512	0.01 μ	0.51 μ	4.61 μ	262.14 μ	0.13 sec	10^{135} cent
2048	0.01 μ	2.05 μ	22.53 μ	0.01 sec	1.07 sec	10^{598} cent
4096	0.01 μ	4.10 μ	49.15 μ	0.02 sec	8.40 sec	10^{1214} cent
8192	0.01 μ	8.19 μ	106.50 μ	0.07 sec	1.15 min	10^{2447} cent
16384	0.01 μ	16.38 μ	229.38 μ	0.27 sec	1.22 hrs	10^{4913} cent
32768	0.02 μ	32.77 μ	491.52 μ	1.07 sec	9.77 hrs	10^{9845} cent
65536	0.02 μ	65.54 μ	1048.6 μ	0.07 min	3.3 days	10^{19709} cent
131072	0.02 μ	131.07 μ	2228.2 μ	0.29 min	26 days	10^{39438} cent
262144	0.02 μ	262.14 μ	4718.6 μ	1.15 min	7 mnths	10^{78894} cent
524288	0.02 μ	524.29 μ	9961.5 μ	4.58 min	4.6 years	10^{157808} cent
1048576	0.02 μ	1048.60 μ	20972 μ	18.3 min	37 years	10^{315634} cent

Table 1.1 Running times for different sizes of input. "nsec" stands for nanoseconds, "μ" is one microsecond and "cent" stands for centuries.

M. H. Alsuwaiyel: Algorithms, Design Techniques and Analysis.

Příští přednáška

- Následuje cvičení v B311 v 14:00
- Příští přednáška 29.9.2009
  v B410 od 12:00