Euklidův algoritmus

Doprovodný materiál pro cvičení Programování I. NPRM044
Autor: Markéta Popelová
Datum: 31.10.2010

Euklidův algoritmus verze 1.0

Zadání: *Určete největšího společného dělitele dvou zadaných přirozených čísel* a_0 *a* b_0.

Vstup: a_0, b_0.

Výstup: NSD(a_0, b_0).

Algoritmus:

1. $a \leftarrow a_0, b \leftarrow b_0$. *(Do a dosaď* a_0 *a do b dosaď* b_0*.)*
2. Dokud $a \neq b$:
   2.1. Jestliže $a < b$:
      2.1.1. $a \leftrightarrow b$. *(Prohoď a a b.)*
   2.2. $a \leftarrow a - b$. *(Do a dosaď* $a - b$*.)*
3. Vrať a. *(Výsledek je a.)*

Důkaz konečnosti:

V každém kroku se $a + b$ zmenší o $\min(a, b)$, tedy vždy alespoň o 1. Jestliže jsme na začátku dostali 2 kladná (přirozená) čísla a odečítáme vždy menší od většího, tak nemůžeme získat číslo záporné. Nemůžeme získat ani 0, neboť to by se stalo jen tehdy, že by čísla byla stejná - v takovém případě bychom již skončili a vydali výsledek. Proto bude mít $a + b$ stále kladnou hodnotu. A jelikož se pokaždé alespoň o jedna sníží, tak proběhne nejvýše $a + b$ průchodů cyklem - a pak algoritmus skončí.

Důkaz správnosti:

Pozorování: Pro $a > b$ platí: $d \mid a \ \& \ d \mid b \Leftrightarrow d \mid a - b \ \& \ d \mid b$.

Důkaz. 1. $\Rightarrow$

Nechť d je společný dělitel čísel a i b. Pak existují přirozená čísla x a y, že

$$a = d \cdot x \quad \text{a} \quad b = d \cdot y.$$

Pak ale d je i dělitelem $a - b$, neboť

$$a - b = d \cdot x - d \cdot y = d \cdot (x - y).$$

A triviálně d dělí i b.

2. $\Leftarrow$

Nechť d je dělitel $a - b$ i b. Pak existují přirozená čísla x a y, že

$$a - b = d \cdot x \quad \text{a} \quad b = d \cdot y.$$

Pak d je dělitelem i a, neboť

$$a = a - b + b = d \cdot x + d \cdot y = d \cdot (x + y).$$

□

Důsledek: Pro $a > b$ platí $NSD(a, b) = NSD(a - b, b)$.

Důkaz. Jelikož všichni dělitelé a a b dělí i $a - b$ a b a naopak, tak i ten největší musí být stejný. Tedy $NSD(a, b) = NSD(a - b, b)$. □

Pozorování: Pro jakákoliv přirozená a, b platí: $NSD(a, b) = NSD(b, a)$.
Pozorování: Po celou dobu běhu algoritmu platí invariant $NSD(a_0, b_0) = NSD(a, b)$.

Důkaz. (Matematickou indukcí dle doby běhu algoritmu)

1. Na začátku algoritmu (před prvním průchodem cyklem) to triviálně platí. $NSD(a_0, b_0) = NSD(a_0, b_0)$
2. (IP) Nechť jsme na konci n-tého průchodu cyklem. Hodnoty v a a b si označme a a b. Nechť pro tyto a a b platí indukční předpoklad: $NSD(a_0, b_0) = (a, b)$.
3. (IK) Co se děje v následujícím průchodu cyklem? Nejdříve možná prohodíme a a b, to ale víme, že největší společný dělitel nemění. Neboli $NSD(a, b) = NSD(b, a)$. Pak ale jen do a uložíme $a - b$. A to jsme si taktéž ukázali, že největší společný dělitel nemění: $NSD(a, b) = NSD(a - b, b)$. Tedy největší společný dělitel čísel a, b před tímto průchodem cyklem je stejný jako před příštím průchodem cyklem.

□

Pozorování: $NSD(a, a) = a$.

Důsledek: Algoritmus vydá po konečně mnoha krocích správný výsledek.

Odhad paměťové složitosti:

Paměťová složitost je konstantní, neboť nám ke všem výpočtům si stačí pamatovat jen dvě čísla – aktuální a a b a možná jedna pomocná proměnná na výměnu. Tedy $M(n) \in \Theta(1)$.

Odhad časové složitosti:

Již při důkazu konečnosti jsme ukázali, že algoritmus uskuteční nejvýše $a + b$ průchodů cyklem. V jednom průchodu cyklem se provede konstantně mnoho operací (výměna dvou proměnných, odečtení a uložená výsledku). Tedy můžeme časovou složitost odhadhnout shora $T(n) \in \mathcal{O}(a + b)$.

Implementace v Pascalu:

```
program NSD1;
{Načte dvě čísla ze vstupu a vypíše na výstup jejich největšího společného dělitele.
 K tomu využívá euklidův algoritmus 1.0, tedy tak dlouho odečítám menší číslo od většího,
 až dostanu dvě stejná čísla, což je NSD zadaných čísel.}
var a,b,pom : integer;
begin
  read(a,b);
  while (a <> b) do
    begin
      if (a < b) then
        begin
          pom := a;
          a := b;
          b := pom;
        end;
      a := a-b;
    end;
  write(a);
end.
```

Euklidův algoritmus verze 2.0

Pozorování: Pokud bylo jedno číslo o mnoho větší než druhé, algoritmus zbytečně dlouho odečítá menší od většího. Těchto několik kroků by se dalo nahradit výpočtem zbytku po dělení. To přesně provádí operace modulo. Předpokládejme, že tato operace (např. pro čísla typu `integer`) trvá konstantně dlouho. Pak můžeme původní algoritmus zrychlit.

Zadání, vstup a výstup jsou stejné jako u Euklidova algoritmu verze 1.0. Změníme pouze algoritmus a implementaci + si dokážeme lepší časovou složitost.

Algoritmus:

1. $a \leftarrow a_0, b \leftarrow b_0$. *(Do a dosaď a_0 a do b dosaď b_0.)*
2. Jestliže $a < b$:
   2.1. $a \leftrightarrow b$. *(Prohoď a a b.)*
3. Dokud $b \neq 0$:
   3.1. $a \leftarrow a \mod b$. *(Do a dosaď $a \mod b$. Tím je určitě $a < b$.)*
   3.2. $a \leftrightarrow b$. *(Prohoď a a b, aby zase platilo $a \geq b$.)*
4. Vrať a. *(Výsledek je a.)*

Důkaz správnosti:

Rozmysleme si, že tento algoritmus oproti verzi 1.0 je jiný jen v tom, že několik průchodů cyklem s odečítáním nahradíme jedním dělením se zbytkem.

Pro představu (o kolik kroků si můžeme pomoci) se podívejme na tento příklad:

Nechť $a = 51$ a $b = 5$, tak hodnoty a, b po jednotlivých průchodech cyklem jsou:

$$46;5 \to 41;5 \to 36;5 \to 29;5 \to 26;5 \to 19;5 \to 16;5 \to 9;5 \to 6;5 \to 1;5 \to 1;1$$

Místo toho algoritmus verze 2.0 pro stejné vstupní hodnoty bude probíhat:

$$5;1 \to 1;0$$

A jelikož je jinak tento algoritmus stejný jako ten předchozí, je i tento algoritmus správný, tedy počítá největší společný dělitel dvou čísel. Změní se jen, jak dlouho mu to trvá. A na to se podívejme hned v následujícím odstavci.

Odhad časové a paměťové složitosti:

Pozorování: Jsme-li někde v průběhu Euklidova algoritmu verze 2.0 (na začátku průchodu cyklem), pak se během následujících dvou iterací a i b změnší alespoň dvakrát.

Důkaz. Označme si a_0, b_0 před iteracemi. Po první iteraci jejich hodnoty označme a_1, b_1. Po druhé iteraci je označme a_2, b_2.

Platí tedy

$$a_1 = b_0$$
$$b_1 = a_0 \mod b_0$$
$$a_2 = b_1$$
$$b_2 = a_1 \mod b_1.$$

Chceme ukázat, že

$$a_2 \leq \frac{a_0}{2} \text{ a } b_2 \leq \frac{b_0}{2}.$$

Nejdříve si uvědomme, že pro čísla $x > y$ pokud $z = x \mod y$, tak $x < y$ (zbytek po dělení y bude určitě menší než y).

Pak platí následující vztahy:

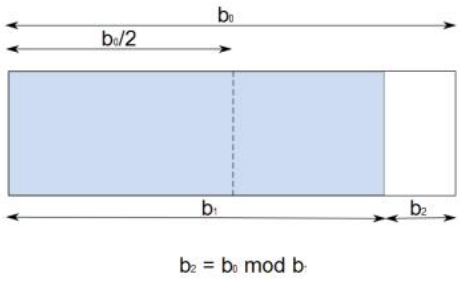
$$b_1 = a_0 \mod b_0 \Rightarrow b_1 < b_0,$$
$$b_2 = a_1 \mod b_1 = b_0 \mod b_1 \Rightarrow b_2 < b_1.$$

(a) Nechť platí $b_1 \leq \frac{b_0}{2}$. Pak

$$b_2 < b_1 \leq \frac{b_0}{2}.$$

(b) Jinak $b_1 > \frac{b_0}{2}$. Pak ale (plyne z obrázku)

$$b_2 < \frac{b_0}{2}.$$



Obdobně ukážeme, že

$$a_2 < \frac{a_0}{2}.$$

Z průběhu algoritmu víme, že

$$a_1 = b_0 < a_0 \Rightarrow a_1 < a_0,$$
$$a_2 = b_1 = a_0 \mod b_0 = a_0 \mod a_1 \Rightarrow a_2 < a_1.$$

Následuje úvaha analogická jako u b:

(a) Nechť platí $a_1 \leq \frac{a_0}{2}$. Pak

$$a_2 < a_1 \leq \frac{a_0}{2}.$$

(b) Jinak $a_1 > \frac{a_0}{2}$. Pak ale

$$a_2 < \frac{a_0}{2}.$$

□

Díky předchozímu tvrzení víme, že počet iterací Euklidova algoritmu verze 2.0 bude nejvýše

$$2 \cdot (\lceil \log_2 a \rceil + \lceil \log_2 b \rceil).$$

Proto platí, že asymptotická časová složitost tohoto vylepšeného algoritmu je $\mathcal{O}(\log a + \log b)$. Paměťová složitost je stejně jako u předchozího algoritmu konstantní.

Implementace v Pascalu:

```
program NSD2;

{Načte dvě čísla ze vstupu a vypíše na výstup jejich největšího společného dělitele.
 K tomu využívá euklidův algoritmus 2.0, tedy dlouhé odečítání z verze 1.0 nahradí
 operací modulo.}

var a,b,pom : integer;

begin
  read(a,b);
  if (a < b) then
    begin
      pom := a;
      a := b;
      b := pom;
    end;
  while ( b <> 0 ) do
    begin
      a := a mod b;
      pom := a;
      a := b;
      b := pom;
    end;
  write(a);
end.
```