PRESEK
List za mlade matematike, fizike, astronome in računalnikarje

ISSN 0351-6652
Letnik **21** (1993/1994)
Številka 2
Strani 116–121

Martin Juvan:

O EVKLIDOVEM ALGORITMU

Ključne besede: računalništvo, matematika, teorija števil, Evklidov algoritem.

Elektronska verzija: http://www.presek.si/21/1169-Juvan-Evklid.pdf

O EVKLIDOVEM ALGORITMU

Pri reševanju različnih problemov se večkrat srečamo z iskanjem največjega skupnega delitelja dveh naravnih ali včasih dveh celih števil. Da bomo problem dobro razumeli, si najprej oglejmo natančne definicije obravnavanih pojmov. Pravimo, da celo število a *deli* celo število b, če obstaja tako celo število k, da je $b = ka$. Številu a pravimo tudi *delitelj* števila b. Tako na primer 5 deli 235, saj je $235 = 5 \cdot 47$, medtem ko 7 ne deli 235, saj $\frac{235}{7}$ ni celo število. Prejšnji trditvi krajše zapišemo tudi takole: $5 \mid 235$ in $7 \nmid 235$. Neposredno iz definicije sledi, da imata števili a in $-a$ enake delitelje. Ker je po definiciji kvocient med številom in vsakim njegovim deliteljem celo število, so vsi delitelji neničelnega celega števila a zajeti med števili $-|a|, -|a|+1, \ldots, |a|-1, |a|$. Izjema je le število 0, ki je deljivo z vsakim celim številom, saj je $0 = 0 \cdot a$ za vsako celo število a.

Številu d, ki deli celi števili a in b, pravimo *skupni delitelj* števil a in b. Ker ima po gornjem razmisleku vsako neničelno celo število le končno mnogo deliteljev, imata poljubni različni celi števili a in b le končno mnogo skupnih deliteljev (saj je vsaj eno od obeh števil različno od nič). Ker pa vsaka končna neprazna množica celih števil vsebuje največje število, obstaja tudi *največje* število v (neprazni, saj števili 1 in -1 delita vsako drugo celo število) množici skupnih deliteljev števil a in b. To število imenujemo *največji skupni delitelj* števil a in b. V računalniških krogih (v katerih prevladuje angleško izrazoslovje) se je za največji skupni delitelj števil a in b uveljavila pisava $\gcd(a, b)$. To je kratica za izraz *greatest common divisor*, ki v angleščini pomeni največji skupni delitelj.

Končajmo z definicijami in poglejmo, kako izračunamo največji skupni delitelj dveh števil. Vzemimo poljubni naravni števili a in b ter ju razstavimo na prafaktorje:

$$a = p_1^{\alpha_1} \ldots p_k^{\alpha_k} \qquad \text{in} \qquad b = p_1^{\beta_1} \ldots p_k^{\beta_k},$$

kjer so $p_1, \ldots, p_k$ vsa različna praštevila, ki nastopajo v razcepu števila a, v razcepu števila b ali pa v razcepu obeh, eksponenti $\alpha_1, \ldots, \alpha_k$ in $\beta_1, \ldots, \beta_k$ pa so naravna števila ali število 0. Vrednost 0 moramo v takem zapisu dovoliti, saj nekateri prafaktorji lahko nastopajo le v razcepu enega od števil a ali b. Ponovimo, da je $a^0 = 1$ za vsako celo število a. Tako imamo

$$882 = 2^1 3^2 5^0 7^2 \qquad \text{in} \qquad 270 = 2^1 3^3 5^1 7^0.$$

Največji skupni delitelj števil a in b potem lahko zapišemo v obliki

$$\gcd(a, b) = p_1^{\min\{\alpha_1, \beta_1\}} \dots p_k^{\min\{\alpha_k, \beta_k\}}$$

V zgornjem primeru je torej

$$\gcd(882, 270) = 2^1 3^2 5^0 7^0 = 18 .$$

Vendar pa največjega skupnega delitelja dveh števil običajno ne iščemo tako, da obe števili razstavimo in izberemo skupne prafaktorje z ustreznimi eksponenti, ampak si raje pomagamo z Evklidovim algoritmom. Kot nam pove njegovo ime, so ga poznali že stari Grki. Omenjen je v sedmi knjigi Evklidovih Elementov. Algoritem temelji na naslednjih preprostih dejstvih:

- Za vsako naravno število a je $\gcd(a, 0) = a$.
- Naj bo $a = kb + r$, kjer je $0 \leq r < b$. Število a torej delimo z naravnim številom b, s k označimo celoštevilski kvocient, z r pa ostanek pri tem deljenju. Potem je $\gcd(a, b) = \gcd(b, r)$.

O veljavnosti prve trditve smo se že prepričali. Skicirajmo še dokaz druge. Pokazali bomo, da so skupni delitelji števil a in b ravno skupni delitelji števil b in r. Potem pa bo tudi največji skupni delitelj v obeh primerih enak. Naj bo d skupni delitelj števil a in b, torej je $a = k_1 d$ in $b = k_2 d$. Ker pa je $r = a - kb = (k_1 - kk_2)d$, število d deli tudi število r. Dokaz, da je vsak skupni delitelj števil b in r tudi skupni delitelj števil a in b, je skoraj enak, zato ga prepuščam bralcu.

Iskanje največjega skupnega delitelja z Evklidovim algoritmom ilustrirajmo na zgornjem primeru:

$$\begin{aligned} 882 &= 3 \cdot 270 + 72 \\ 270 &= 3 \cdot 72 + 54 \\ 72 &= 1 \cdot 54 + 18 \\ 54 &= 3 \cdot 18 + 0 \end{aligned}$$

Največji skupni delitelj števil 882 in 72 je tako zadnji neničelni ostanek, v našem zgledu je to seveda število 18.

Z nekaj znanja programiranja lahko Evklidov algoritem zapišemo kot kratek funkcijski podprogram v programskem jeziku pascal.

```
function GCD(a,b: integer): integer;
   { Izračuna največji skupni delitelj števil a in b z Evklidovim algoritmom. }
 var
   r: integer;
 begin
   while b<>0 do begin
     r:=a mod b;                                   { Izračunamo novi ostanek. }
     a:=b;                  { Prejšnje drugo število postane novo prvo število. }
     b:=r;                                  { Novi ostanek postane drugo število. }
   end;
   GCD:=abs(a);
 end; {GCD}
```

V rešitvi smo uporabili tudi v pascal vgrajeno funkcijo **mod**. Ta je tesno povezana s funkcijo **div**, ki jo bomo potrebovali kasneje. Klic *a* **mod** *b* vrne ostanek pri deljenju števila *a* s številom *b*. Predznak ostanka je enak predznaku števila *a*. Klic *a* **div** *b* vrne celoštevilski kvocient števil *a* in *b*. Kvocient je pozitiven, če imata števili *a* in *b* enak predznak, sicer pa je negativen. Na primer:

$$5 \ \textbf{mod}\ 3 = 5 \ \textbf{mod}\ -3 = 2 \text{ in } -5 \ \textbf{mod}\ 3 = -5 \ \textbf{mod}\ -3 = -2\,,$$
$$5 \ \textbf{div}\ 3 = -5 \ \textbf{div}\ -3 = 1 \text{ in } \quad 5 \ \textbf{div}\ -3 = -5 \ \textbf{div}\ 3 = -1\,.$$

Obe operaciji povezuje zveza $a = (a \ \textbf{div}\ b) \cdot b + a \ \textbf{mod}\ b$, ki velja za vsa cela števila.

Gornji algoritem je tudi zelo robusten. Za pravilno delovanje ni potrebno, da je število *a* po absolutni vrednosti večje od števila *b*. Če je število *b* večje, potem prva ponovitev zanke **while** poskrbi za zamenjavo vrednosti obeh spremenljivk. Prav tako ni nujno, da sta vrednosti obeh parametrov nenegativni. Če sta obe števili negativni, račun poteka enako kot pri pozitivnih podatkih, le da so tudi vsi ostanki negativni. Ker pa je rezultat funkcije GCD absolutna vrednost zadnjega neničelnega ostanka, ti predznaki ne vplivajo na končni rezultat. Če pa sta vhodna podatka različno predznačena, se tudi predznaki ostankov v algoritmu izmenjujejo.

Gornji podprogram lahko sprogramiramo tudi rekurzivno.

```
function GCDr(a,b: integer): integer;
{Rekurzivni Evklidov algoritem. }
 begin
   if b=0 then
     GCDr:=abs(a)
   else
     GCDr:=GCDr(b,a mod b);
 end; {GCDr}
```

Rekurzivna rešitev je elegantnejša, saj je skoraj dobesedno enaka matematičnemu opisu algoritma s prejšnje strani. Seveda pa ima tudi svoje slabosti. Tako vsak rekurzivni klic porabi nekaj dodatnega prostora in časa za oba parametra in "knjigovodstvo" ob klicu. Poskusi pokažejo, da je gornji rekurzivni podprogram približno dvakrat počasnejši od iterativnega.

In za kaj lahko uporabimo izračunani največji skupni delitelj? Recimo pri reševanju linearne diofantske enačbe z dvema neznankama. Tudi take enačbe so reševali že v antiki, svoje ime pa so dobile po matematiku Diofantu, ki je živel okoli leta 250. Teorija pravi, da je pri izbranih celih številih a, b in c enačba $ax + by = c$ rešljiva v celih številih natanko tedaj, ko $\gcd(a, b)$ deli c. In kako v tem primeru poiščemo kakšno rešitev? Pokazali bomo, da si tudi tu lahko pomagamo z Evklidovim algoritmom. Z njegovo pomočjo bomo poiskali taki celi števili x_0 in y_0, da bo veljalo

$$ax_0 + by_0 = \gcd(a, b)\,.$$

Oglejmo si postopek na prejšnjem primeru. Zadnji neničelni ostanek želimo zapisati kot celoštevilsko kombinacijo začetnih števil a in b. To dosežemo z zaporednim izražanjem ostankov od zadnjega proti prvemu in vstavljanjem dobljene zveze v prejšnjo vrstico Evklidove sheme. Tako imamo:

$$\begin{aligned}
18 &= 72 - 54 && 54 = 270 - 3 \cdot 72\\
&= 72 - (270 - 3 \cdot 72)\\
&= -270 + 4 \cdot 72 && 72 = 882 - 3 \cdot 270\\
&= -270 + 4 \cdot (882 - 3 \cdot 270)\\
&= 4 \cdot 882 - 13 \cdot 270
\end{aligned}$$

Vzamemo $x_0 = 4$ in $y_0 = -13$ in naloga je rešena.

Vendar pa je računanje s svinčnikom in papirjem naporno in zamudno, pa tudi napake se rade prikradejo v naše zapiske. Zato raje poskusimo napisati podprogram, ki bo to "zoprno" in zamudno delo opravil namesto nas. Tako dopolnjeni Evklidov algoritem bomo imenovali *razširjeni Evklidov algoritem*. Seveda pa moramo postopek sprogramirati tako, da bomo prepričani, da deluje pravilno, in da bomo znali to svoje prepričanje tudi ustrezno utemeljiti. Zato si najprej na prejšnjem primeru oglejmo nekoliko drugačen in bolj pregleden zapis razširjenega Evklidovega algoritma:

$$
\begin{aligned}
1 \cdot 882 + \quad 0 \cdot 270 &= 882 \\
0 \cdot 882 + \quad 1 \cdot 270 &= 270 \;/ \cdot 3 \\
1 \cdot 882 + (-3) \cdot 270 &= 72 \;/ \cdot 3 \\
(-3) \cdot 882 + \quad 10 \cdot 270 &= 54 \;/ \cdot 1 \\
4 \cdot 882 + (-13) \cdot 270 &= 18 \;/ \cdot 3 \\
(-15) \cdot 882 + \quad 49 \cdot 270 &= 0
\end{aligned}
$$

Prvi dve vrstici zgornje sheme trdita očitno resnico o obeh vhodnih podatkih. Nato z desnimi stranmi enakosti opravimo običajni Evklidov algoritem: izračunamo celoštevilski kvocient med zadnjima dvema desnima stranema, pomnožimo z njim zadnjo enačbo in jo odštejemo od predzadnje. Ko je desna stran nove enačbe enaka 0, se ustavimo. Predzadnja enačba je iskani zapis največjega skupnega delitelja kot celoštevilske kombinacije začetnih števil a in b. Mimogrede nam zadnja enačba razkrije še najmanjši skupni večkratnik začetnih števil a in b. V prejšnjem primeru je to $15 \cdot 882 = 49 \cdot 270 = 13230$. Da je to vedno res, naj se bralec prepriča sam.

Zgornje sheme pa ni težko zapisati v obliki podprograma v pascalu.

```
function razsGCD(a,b: integer; var x0,y0: integer): integer;
{Poišče največji skupni delitelj števil a in b ter njegov zapis kot}
{celoštevilsko kombinacijo števil a in b s koeficientoma x0 in y0.}
 var
   x1,y1,k,t: integer;
 begin
   x0:=1;  y0:=0;
   x1:=0;  y1:=1;
   while b<>0 do begin
     k:=a div b;
     t:=b;  b:=a-k*b;  a:=t;
     t:=x1;  x1:=x0-k*x1;  x0:=t;
     t:=y1;  y1:=y0-k*y1;  y0:=t;
   end;
   razsGCD:=a;
 end; {razsGCD}
```

Poskusimo še nekoliko analizirati Evklidov algoritem. Koliko deljenj moramo opraviti v najslabšem primeru, če z Evklidovim algoritmom iščemo največji skupni delitelj naravnih števil a in b? Odgovor na to vprašanje je povezan s Fibonaccijevimi števili. To je zaporedje števil $f_1, f_2, f_3, \ldots$, ki je določeno takole. Prva dva člena zaporedja sta enaka 1: torej $f_1 = f_2 = 1$, vsak nadaljnji člen pa je vsota predhodnih dveh: $f_{n+1} = f_n + f_{n-1}$ za

$n \geq 2$. Začetek zaporedja Fibonaccijevih števil je tako $1, 1, 2, 3, 5, 8, 13, \ldots$. Z indukcijo se zlahka prepričamo, da potrebujemo za izračun največjega skupnega delitelja števil f_{n+1} in f_n z Evklidovim algoritmom natanko n deljenj. Ko namreč opravimo prvo deljenje, je ostanek ravno število f_{n-1}. Za izračun največjega skupnega delitelja števil f_n in f_{n-1} po indukcijski predpostavki potrebujemo $n-1$ deljenj, skupaj torej natanko n. (Mimogrede tudi opazimo, da sta zaporedni Fibonaccijevi števili tuji.)

Pokažimo še nekakšen obrat zgornje trditve. Naj bosta a in b naravni števili, pri katerih je Evklidov algoritem potreboval natanko n deljenj za izračun njunega največjega skupnega delitelja. Privzeli bomo, da je $a > b$. Ostanke, ki se pojavijo v algoritmu, označimo z $r_1, r_2, \ldots, r_n = 0$, kvociente pa s $k_1, k_2, \ldots, k_n$. Posebej definiramo $r_{-1} = a$ in $r_0 = b$. Pri izračunu $\gcd(882, 270)$ je torej zaporedje ostankov $882, 270, 72, 54, 18, 0$, zaporedje kvocientov pa je $3, 3, 1, 3$. Ker so kvocienti in ostanki dobljeni z Evklidovim algoritmom, za $1 \leq i \leq n$ velja $r_{i-2} = k_i r_{i-1} + r_i$. Z matematično indukcijo bomo pokazali, da velja $r_i \geq f_{n-i}$ za $i = n-1, n-2, \ldots, 0, -1$. Ker je zaporedje ostankov strogo padajoče, vsi členi razen zadnjega pa so pozitivni, ocena velja pri $i = n-1$ in pri $i = n-2$. Opravimo še indukcijski korak. Prejšnji ostanek r_{i-2} je enak $k_i r_{i-1} + r_i$, to število pa po indukcijski predpostavki ni manjše od $f_{n-(i-1)} + f_{n-i} = f_{n-(i-2)}$. Pri oceni smo tudi upoštevali, da so vsi kvocienti k_i naravna števila, torej večji ali enaki 1. Ocene za velikost ostankov so s tem dokazane.

Če v izpeljane neenakosti vstavimo $i = -1$ in $i = 0$, vidimo, da je $a \geq f_{n+1}$ in $b \geq f_n$. Zaporedni Fibonaccijevi števili f_{n+1} in f_n torej tvorita najmanjši par različnih naravnih števil, pri katerih Evklidov algoritem potrebuje za izračun največjega skupnega delitelja n deljenj.

Označimo s φ razmerje zlatega reza, torej $\varphi = \frac{1+\sqrt{5}}{2}$. Pokažimo, da je $f_n \geq \varphi^{n-2}$. Oceno dokažemo z matematično indukcijo. Za $n = 1$ in $n = 2$ trditev očitno drži. Izpeljimo še indukcijski korak. Privzemimo, da je $f_n \geq \varphi^{n-2}$ in $f_{n-1} \geq \varphi^{n-3}$. Potem pa je tudi $f_{n+1} = f_n + f_{n-1} \geq \varphi^{n-2} + \varphi^{n-3} = \varphi^{n-3}(\varphi + 1) = \varphi^{n-3}\varphi^2 = \varphi^{n-1}$, saj je $\frac{1+\sqrt{5}}{2} + 1 = (\frac{1+\sqrt{5}}{2})^2$.

Gornji rezultat o številu deljenj, ki jih potrebuje Evklidov algoritem, lahko povemo tudi nekoliko drugače. Ker je $a \geq f_{n+1} \geq \varphi^{n-1}$, z logaritmiranjem dobimo $\log_\varphi a \geq n-1$. Število deljenj n, ki jih opravi Evklidov algoritem, je torej kvečjemu $1 + \log_\varphi a$.

Martin Juvan