

# Osnovni algoritmi teorije brojeva

---

Guskić, Ivona

Undergraduate thesis / Završni rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Department of Mathematics / Sveučilište Josipa Jurja Strossmayera u Osijeku, Odjel za matematiku**

Permanent link / Trajna poveznica: <https://um.nsk.hr/um:nbn:hr:126:851954>

Rights / Prava: [In copyright](#) / [Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2025-02-02**



**mathos**

Repository / Repozitorij:

[Repository of School of Applied Mathematics and Informatics](#)



Sveučilište J.J. Strossmayera u Osijeku  
Odjel za matematiku

Ivona Guskić  
**Osnovni algoritmi teorije brojeva**  
Završni rad

Osijek, 2019.

Sveučilište J.J. Strossmayera u Osijeku  
Odjel za matematiku

Ivona Guskić  
**Osnovni algoritmi teorije brojeva**  
Završni rad

Mentor: izv. prof. dr. sc. Ivan Matić

Osijek, 2019.

# Sadržaj

Sažetak	1
1 Uvod	2
2 Kineski teorem o ostacima	3
3 Računanje s velikim brojevima	5
3.1 Zbrajanje . . . . .	5
3.2 Oduzimanje . . . . .	5
3.3 Množenje . . . . .	5
3.4 Dijeljenje s ostatkom . . . . .	6
4 Euklidov algoritam	9
4.1 Osnovni algoritam . . . . .	9
4.2 Modifikacija Euklidovog algoritma . . . . .	10
5 Računanje modularnih inverza i algoritam za kineski teorem o ostacima	12
5.1 Modularni inverzi . . . . .	12
5.2 Algoritam za kineski teorem o ostacima . . . . .	12
Literatura	13

## **Sažetak**

U ovom završnom radu bavimo se algoritmima u teoriji brojeva. Navesti ćemo osnovne teoreme i dokaze potrebne za shvaćanje i implementaciju tih algoritama. Pokazati ćemo novi način zapisa brojeva u računalima kako bi se mogle odvijati osnovne aritmetičke operacije sa velikim cijelim brojevima. Na kraju ćemo dati efikasnu implementaciju Euklidovog algoritma i Kineskog teorema o ostacima u pseudojeziku.

## **Ključne riječi**

algoritmi, teorija brojeva, Euklidov algoritam, Kineski teorem o ostacima

## **Abstract**

In this paper we deal with algorithms in number theory. We will present basic theorems and proofs necessary for understanding and implementation of those algorithms. We will show a new way of storing numbers in a computer so we can do basic arithmetic with large integers. In the end, we will give an efficient implementation of Euclidean algorithm and of the Chinese remainder theorem in pseudocode.

## **Key words**

algorithms, number theory, Euclidean algorithm, Chinese remainder theorem

# 1 Uvod

Algoritmi teorije brojeva od velike su važnosti, ne samo za matematiku, nego i za programiranje. Posebno su značajni u raznim enkripcijski sustavima npr. RSA kriptosustav u kojem se primjenjuje Euklidov algoritam, koji ćemo u ovom radu prezentirati. Brojevi s kojima rade takvi algoritmi često se u računalima ne mogu zapisati kao brojevi pa je stoga bilo potrebno pronaći novi način zapisivanja brojeva i definirati aritmetiku na tom novom zapisu što ćemo pokazati u trećem poglavlju. U četvrtom poglavlju detaljno ćemo pokazati dva oblika Euklidovog algoritma, a na kraju rada prezentirati ćemo i algoritam za Kineski teorem o ostacima.

## 2 Kineski teorem o ostacima

**Teorem 2.1** (Kineski teorem o ostacima). *Neka su  $n_1, n_2, \dots, n_k$  u parovima relativno prosti prirodni brojevi te neka su  $a_1, a_2, \dots, a_k$  proizvoljni cijeli brojevi. Tada postoji rješenje sustava kongruencija*

$$a \equiv a_i \pmod{n_i} \quad (i = 1, \dots, k).$$

*Štoviše,  $a' \in \mathbb{Z}$  je rješenje ovog sustava kongruencija ako i samo ako je  $a \equiv a' \pmod{n}$ , gdje je  $n := \prod_{i=1}^k n_i$ .*

*Dokaz.* Da bi dokazali postojanje rješenja, prvo ćemo pokazati kako konstruirati cijele brojeve  $e_1, e_2, \dots, e_k$  takve da za  $i, j = 1, \dots, k$  vrijedi

$$e_j \equiv \begin{cases} 1 & \pmod{n_i} & i = j \\ 0 & \pmod{n_i} & i \neq j \end{cases}$$

Ako to uspijemo, tada za

$$a := \sum_{i=1}^k a_i e_i,$$

i za  $j = 1, \dots, k$  vrijedi

$$a \equiv \sum_{i=1}^k a_i e_i \equiv a_j \pmod{n_j},$$

jer su svi izrazi u sumi jednaki 0 modulo  $n_j$ , osim za izraz gdje je  $i = j$ , koji je kongruentan  $a_j$  modulo  $n_j$ .

Neka je  $n := \prod_{i=1}^k n_i$  kao u iskazu teorema i neka je  $n_j^* := n/n_i$  za  $j = 1, \dots, k$ , tj.  $n_i^*$  je produkt svih  $n_j$  ( $i \neq j$ ). Kako su  $n_1, \dots, n_k$  u parovima relativno prosti, slijedi da je za  $i = 1, \dots, k$   $(n_i, n_i^*) = 1$  pa možemo definirati  $t_i = (n_i^*)^{-1} \pmod{n_i}$  i  $e_i := n_i^* t_i$ . Vidi se da je  $e_i \equiv 1 \pmod{n_i}$ , dok za  $j \neq i$ , imamo  $n_i | n_j^*$ , pa je  $e_j \equiv 0 \pmod{n_i}$ , tj. uspješno smo konstruirali brojeve  $e_j$  kako je navedeno gore. Time smo dokazali postojanje rješenja sustava kongruencija. Ako je  $a \equiv a' \pmod{n}$ , onda zbog  $n_i | n$  za  $i = 1, \dots, k$  vidimo da je  $a' \equiv a \equiv a_i \pmod{n_i}$  tj.  $a'$  je rješenje sustava. Ostaje još pokazati da ako je  $a'$  rješenje sustava kongruencija, onda je  $a \equiv a_i \equiv a' \pmod{n_i}$  za  $i = 1, \dots, k$ . Kako su  $n_i$  u parovima relativno prosti slijedi  $n | (a - a')$  što je ekvivalentno sa  $a \equiv a' \pmod{n}$ .  $\square$

Kineski teorem o ostacima ima široku primjenu, a možemo ga izreći i na sljedeći način: Neka je  $I_m = \{0, \dots, m-1\}$  za prirodan broj  $m$ , neka su  $n_1, \dots, n_k$  u parovima relativno prosti i  $n := \prod_{i=1}^k n_i$ . Tada je preslikavanje

$$\begin{aligned} \tau : I_n &\rightarrow I_{n_1} \times \dots \times I_{n_k} \\ a &\mapsto (a \pmod{n_1}, \dots, a \pmod{n_k}) \end{aligned}$$

bijekcija.

Promotrimo primjer za  $n_1 = 3$  i  $n_2 = 5$  sljedećom tablicom.

$a$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$a \pmod{3}$	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2
$a \pmod{5}$	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4

Vidimo da kako  $a$  poprima vrijednosti od 0 do 14, parovi  $(a \pmod{3}, a \pmod{5})$  poprimaju sve vrijednosti  $(a_1, a_2)$  za  $a_1 \in \{0, 1, 2\}$  i  $a_2 \in \{0, \dots, 4\}$  i da je svaki par pogodan točno jednom. Algoritam za kineski teorem o ostacima pokazati ćemo kasnije, kada prođemo potrebne rezultate.



### 3 Računanje s velikim brojevima

Kako je memorija za spremanje brojeva u računalima ograničena, potrebno je definirati drugačiji način zapisivanja brojeva, a samim time i drugačiji način računanja za takve brojeve. Predstaviti ćemo ih kao vektor znamenaka u nekoj bazi  $B$ , zajedno s indikator varijablom koja će nam govoriti predznak broja, tj. za  $a \in \mathbb{Z}$  pisat ćemo

$$a = \pm \sum_{i=0}^{k-1} a_i B^i = \pm (a_{k-1} \dots a_1 a_0)_B,$$

gdje je  $0 \leq a_i \leq B$  za  $i = 0, \dots, k-1$ . Iz tehničkih razloga, često je dobro za  $B$  uzeti potenciju broja 2. U ovom radu, zbog jednostavnosti, baviti ćemo se algoritmima za računanje s velikim prirodnim brojevima. Uz male modifikacije ti algoritmi se mogu koristiti za računanje s proizvoljnim cijelim brojevima. Radi jednostavnosti zapisa prvo ćemo za  $x, y \in \mathbb{Z}$ ,  $y \neq 0$  definirati  $QuoRem(x, y)$  kao par kvocijenta i ostatka ( $\lfloor x/y \rfloor, x \pmod{y}$ )

#### 3.1 Zbrajanje

Neka su  $a = (a_{k-1} \dots a_0)_B$  i  $b = (b_{l-1} \dots b_0)_B$  prirodni brojevi i neka je bez smanjenja općenitosti  $k \geq l \geq 1$ . Tada reprezentaciju od  $a + b$  u bazi  $B$  možemo izračunati na sljedeći način:

```
carry ← 0
for i ← 0 to l - 1 do
    temp ← ai + bi + carry
    (carry, ci) ← QuoRem(temp, B)
end for
for i ← l to k - 1 do
    temp ← ai + carry
    (carry, ci) ← QuoRem(temp, B)
end for
ck ← carry
```

Primijetimo da je u svakoj iteraciji petlje vrijednost *carry* 0 ili 1, a vrijednost *temp* između 0 i  $2B - 1$ .

#### 3.2 Oduzimanje

Neka su  $a = (a_{k-1} \dots a_0)_B$  i  $b = (b_{l-1} \dots b_0)_B$  prirodni brojevi i neka je bez smanjenja općenitosti  $k \geq l \geq 1$ . Za računanje razlike  $c := a - b$  dovoljno je malo modificirati algoritam za zbrajanje. Svako pojavljivanje izraza  $a_i + b_i$  zamjenimo s  $a_i - b_i$ . Pri svakoj iteraciji vrijednost *carry* je ili 0 ili  $-1$ , a vrijednost *temp* je između  $-B$  i  $B - 1$ . Ako je  $a \geq b$  onda je  $c_k = 0$  inače je  $c_k = -1$  (i  $b - a = B^k - (c_{k-1} \dots c_0)_B$ ).

#### 3.3 Množenje

Neka su  $a = (a_{k-1} \dots a_0)_B$  i  $b = (b_{l-1} \dots b_0)_B$  prirodni brojevi,  $k \geq 1, l \geq 1$ . Produkt  $c := a \cdot b$  je oblika  $(c_{k+l-1} \dots c_0)_B$  i računamo ga na sljedeći način:

```
for i ← 0 to k + l - 1 do
    ci ← 0
```

```

end for
for  $i \leftarrow 0$  to  $k - 1$  do
  carry  $\leftarrow 0$ 
  for  $j \leftarrow 0$  to  $l - 1$  do
    temp  $\leftarrow a_i b_j + c_{i+j} + carry$ 
    (carry,  $c_{i+j}$ )  $\leftarrow QuoRem(temp, B)$ 
  end for
   $c_{i+l} \leftarrow carry$ 
end for

```

Primijetimo da je u svakom koraku algoritma vrijednost *carry* između 0 i  $B - 1$ , a vrijednost *temp* između 0 i  $B^2 - 1$ .

### 3.4 Dijeljenje s ostatkom

Neka su  $a = (a_{k-1} \dots a_0)_B$  i  $b = (b_{l-1} \dots b_0)_B$  prirodni brojevi,  $k \geq 1, l \geq 1$  i  $b_{l-1} \neq 0$ . Želimo izračunati  $q$  i  $r$  takve da je  $a = bq + r$  i  $0 \leq r < b$ . Pretpostavimo da je  $k \geq l$ , u suprotnom  $a < b$ , pa je  $q = 0$  i  $r = a$ . Kvocijent  $q$  će imati najviše  $m := k - l + 1$  znamenaka u bazi  $B$  ( $q = (q_{m-1} \dots q_0)_B$ ). Strategija kojom ćemo odrediti  $q$  i  $r$  je sljedeća:

```

r  $\leftarrow a$ 
for  $i \leftarrow m - 1$  down to 0 do
   $q_i \leftarrow \lfloor r/B^i b \rfloor$ 
   $r \leftarrow r - B^i \cdot q_i b$ 
end for

```

Lako se indukcijom pokaže da na početku svake iteracije petlje vrijedi  $0 \leq r < B^{i+1}b$  pa je svaki  $q_i$  između 0 i  $B - 1$ . Da bi gornju strategiju pretvorili u algoritam potrebno je pronaći efikasan način za računanje  $\lfloor r/B^i b \rfloor$ .

Promotrimo prvo specijalan slučaj  $l = 1$ . Tada kvocijent možemo odrediti pomoću sljedećeg teorema:

**Teorem 3.1.** *Neka su  $x$  i  $y$  cijeli brojevi takvi da*

$$0 \leq x = x'2^n + s, \quad 0 < y = y'2^n$$

*za neke cijele brojeve  $n, s, x', y'$  gdje je  $n \geq 0$  i  $0 \leq s < 2^n$ . Tada  $\lfloor x/y \rfloor = \lfloor x'/y' \rfloor$ .*

*Dokaz.* Iz

$$\frac{x}{y} = \frac{x'}{y'} + \frac{s}{y'2^n} \geq \frac{x'}{y'}$$

direktno slijedi  $\lfloor x/y \rfloor \geq \lfloor x'/y' \rfloor$ . Također imamo

$$\frac{x}{y} = \frac{x'}{y'} + \frac{s}{y'2^n} < \frac{x'}{y'} + \frac{1}{y'} \leq \left( \left\lfloor \frac{x'}{y'} \right\rfloor + \frac{y' - 1}{y'} \right) + \frac{1}{y'} \leq \left\lfloor \frac{x'}{y'} \right\rfloor + 1$$

Dakle, imamo  $x/y < \lfloor x'/y' \rfloor + 1$  pa je onda  $\lfloor x/y \rfloor \leq \lfloor x'/y' \rfloor$ . □

Prema ovom teoremu, sljedeći algoritam korektno računa kvocijent i ostatak (za slučaj  $l = 1$ ):

```

hi  $\leftarrow 0$ 
for  $i \leftarrow k - 1$  down to 0 do
  temp  $\leftarrow h_i \cdot B + a_i$ 

```

$(q_i, h_i) \leftarrow QuoRem(temp, b_0)$

**end for**

Kvocijent je vektor  $q = (q_{k-1} \dots q_0)_B$  a ostatak je  $h_i$ . U svakoj iteraciji petlje vrijednost  $h_i$  nalazi se između 0 i  $b_0 \leq B - 1$ , a vrijednost  $temp$  između 0 i  $B \cdot b_0 + (B - 1) \leq B^2 - 1$ . Time je riješen slučaj  $l = 1$ . U slučaju  $l \geq 1$  nije tako jednostavno odrediti znamenke  $q_i$  ali ih možemo procijeniti pomoću sljedećeg teorema:

**Teorem 3.2.** *Neka su  $x$  i  $y$  cijeli brojevi takvi da*

$$0 \leq x = x'2^n + s, 0 \leq y = y'2^n + t$$

za neke cijele brojeve  $n, s, t, x', y'$  gdje je  $n \geq 0, 0 \leq s < 2^n, 0 \leq t < 2^n$ . Neka je dodatno  $2y' \geq x/y$ . Tada vrijedi

$$\lfloor x/y \rfloor \leq \lfloor x'/y' \rfloor \leq \lfloor x/y \rfloor + 2.$$

Prvo ćemo predstaviti algoritam za dijeljenje s ostatkom koji radi uz pretpostavku da je  $b$  "normaliziran", tj. da je  $b_{l-1} \geq 2^{\omega-1}$  gdje je  $B = 2^\omega$ .

**for**  $i \leftarrow 0$  **to**  $k - 1$  **do**

$r_i \leftarrow a_i$

**end for**

$r_k \leftarrow 0$

**for**  $i \leftarrow k - l$  **down to** 0 **do**

$q_i \leftarrow \lfloor (r_{i+l}B + r_{i+l-1})/b_l - 1 \rfloor$

**if**  $q_i \geq B$  **then**  $q_i \leftarrow B - 1$

$carry \leftarrow 0$

**for**  $j \leftarrow 0$  **to**  $l - 1$  **do**

$temp \leftarrow r_{i+j} - q_i b_j + carry$

$(carry, r_{i+j}) \leftarrow QuoRem(temp, B)$

**end for**

$r_{i+l} \leftarrow r_{i+l} + carry$

**while**  $r_{i+l} < 0$  **do**  $carry \leftarrow 0$

**for**  $j \leftarrow 0$  **to**  $l - 1$  **do**

$temp \leftarrow r_{i+j} + b_i + carry$

$(carry, r_{i+j}) \leftarrow QuoRem(temp, B)$

**end for**

$r_{i+l} \leftarrow r_{i+l} + carry$

$q_i \leftarrow q_i - 1$

**end while**

**end for**

Primijetimo:

- (1) U 4. redu računamo  $q_i$ , koji je prema teoremu 3.2 veći ili jednak pravoj znamenci kvocijenta, a može biti veći najviše za 2
- (2) U 5. redu reduciramo  $q_i$  ako je očito prevelik.
- (3) U redovima 6-10 računamo

$$(r_{i+l} \dots r_i)_B \leftarrow (r_{i+l} \dots r_i)_B - q_i b$$

U svakoj iteraciji petlja vrijednost  $temp$  je između  $-(B^2 - B)$  i  $B - 1$ , a vrijednost  $carry$  između  $-(B - 1)$  i 0.

- (4) Ako je procijena  $q_i$  prevelika,  $r_{i+l}$  će imati negativnu vrijednost u 10. redu. Redovi 11-17 tada popravljaju tu pogrešku.

Pogledajmo sada općeniti slučaj, kada  $b$  ne mora biti normaliziran. Pomnožimo  $a$  i  $b$  vrijednošću  $2^{\omega'}$ , gdje je  $0 \leq \omega' < \omega$ , i dobijemo  $a' := a2^{\omega'}$  i  $b' := b2^{\omega'}$ , gdje je  $b'$  normaliziran. Tada izračunamo  $q$  i  $r'$  tako da je  $a' = b'q + r'$ , koristeći gore navedeni algoritam. Primijetimo da je  $q = \lfloor a'/b' \rfloor = \lfloor a/b \rfloor$  i  $r' = r2^{\omega'}$ , gdje je  $r = a \pmod{b}$ . Da dobijemo ostatak  $r$ , podijelimo  $r'$  sa  $2^{\omega'}$ .



## 4 Euklidov algoritam

### 4.1 Osnovni algoritam

Promotrimo sljedeći problem: za proizvoljna 2 prirodna broja  $a$  i  $b$ , treba odrediti njihov najveći zajednički djelitelj. To možemo odrediti pomoću Euklidovog algoritma na sljedeći način. Bez smanjenja općenitosti, pretpostavimo da je  $a \geq b \geq 0$ . Ako je  $b = 0$ , tada je očito  $NZD(a, 0) = a$ . Za  $b > 0$  izračunamo  $q$  i  $r$  kao  $(q, r) = QuoRem(a, b)$ , gdje je  $0 \leq r < b$ . Iz jednadžbe

$$a = bq + r,$$

lako se vidi da ako  $d \in \mathbb{Z}$  dijeli i  $a$  i  $b$  tada  $d$  također dijeli i  $r$ , analogno ako  $d$  dijeli i  $b$  i  $r$ , tada dijeli i  $a$ . Iz toga možemo zaključiti da je  $NZD(a, b) = NZD(b, r)$  i da dijeljenjem problem računanja  $NZD(a, b)$  možemo reducirati na problem  $NZD(b, r)$ .

**Teorem 4.1.** *Neka su  $a, b \in \mathbb{Z}, a \geq b \geq 0$ . Korištenjem teorema o dijeljenju s ostatkom, definiramo cijele brojeve  $r_0, \dots, r_l, q_1, \dots, q_l, l \geq 0$  na sljedeći način:*

$$\begin{aligned} a &= r_0 \\ b &= r_1 \\ r_0 &= r_1 q_1 + r_2 \quad (0 < r_2 < r_1) \\ &\vdots \\ r_{i-1} &= r_i q_i + r_{i+1} \quad (0 < r_{i+1} < r_i) \\ &\vdots \\ r_{l-2} &= r_{l-1} q_{l-1} + r_l \quad (0 < r_l < r_{l-1}) \\ r_{l-1} &= r_l q_l \quad (r_{l+1} = 0) \end{aligned}$$

*Dokaz.* Za  $i = 1 \dots l$  imamo  $r_{i-1} = r_i q_i + r_{i+1}$ , iz čega slijedi da su zajednički djelitelji od  $r_{i-1}$  i  $r_i$  jednaki kao zajednički djelitelji od  $r_i$  i  $r_{i+1}$  te je stoga  $NZD(r_{i-1}, r_i) = NZD(r_i, r_{i+1})$ . Iz toga slijedi

$$NZD(a, b) = NZD(r_0, r_1) = \dots = NZD(r_l, r_{l+1}) = NZD(r_l, 0) = r_l.$$

□

Promotrimo sljedeći primjer: neka je  $a = 100$  i  $b = 35$ . Tada po prethodnom teoremu jednostavno možemo izračunati sljedeće:

$i$	0	1	2	3	4
$r_i$	100	35	30	5	0
$q_i$		2	1	6	

Sada vidimo da je  $NZD(a, b) = r_3 = 5$ .

Shemu opisanu u teoremu 4.1 lako možemo pretvoriti u algoritam na sljedeći način:

```

r ← a
r' ← b
while r' ≠ 0 do
    r'' ← r (mod r')
    (r, r') ← (r', r'')
end while
d ← r
output d

```

## 4.2 Modifikacija Euklidovog algoritma

Neka su  $a$  i  $b$  cijeli brojevi i neka je  $d := NZD(a, b)$ . Iz Bezuotovog identiteta znamo da postoje cijeli brojevi  $s$  i  $t$  takvi da je  $as + bt = d$ . Brojeve  $s$  i  $t$  efikasno možemo izračunati modifikacijom Euklidovog algoritma. Sljedeći teorem definira vrijednosti koje će taj algoritam računati i daje važne činjenice o njima koje će biti bitne u primjeni algoritma.

**Teorem 4.2.** *Neka su  $a, b, r_0, \dots, r_{l+1}, q_1, \dots, q_l$  kao u teoremu 4.1. Definiramo cijele brojeve  $s_0, \dots, s_{l+1}, t_0, \dots, t_{l+1}$  na sljedeći način:*

$$\begin{aligned} s_0 &:= 1, & t_0 &:= 0, \\ s_1 &:= 0, & t_1 &:= 1, \\ s_{i+1} &:= s_i - 1 - s_i q_i, & t_{i+1} &:= t_{i-1} - t_i q_i \quad (i = 1, \dots, l) \end{aligned}$$

Tada vrijedi:

- (i) za  $i = 0, \dots, l + 1$  vrijedi  $as_i + bt_i = r_i$ , posebno,  $as_l + bt_l = \gcd(a, b)$
- (ii) za  $i = 0, \dots, l$  vrijedi  $s_i t_{i+1} - t_i s_{i+1} = (-1)^i$
- (iii) za  $i = 0, \dots, l + 1$  vrijedi  $NZD(s_i, t_i) = 1$
- (iv) za  $i = 0, \dots, l$  vrijedi  $t_i t_{i+1} \leq 0$  i  $|t_i| \leq |t_{i+1}|$ ; za  $i = 1, \dots, l$  vrijedi  $s_i s_{i+1} \leq 0$  i  $|s_i| \leq |s_{i+1}|$
- (v) za  $i = 1, \dots, l + 1$  vrijedi  $r_{i-1} |t_i| \leq a$  i  $r_{i-1} |s_i| \leq b$
- (vi) ako je  $a > 0$ , onda za  $i = 1, \dots, l + 1$  vrijedi  $|t_i| \leq a$  i  $|s_i| \leq b$ ; ako je  $a > 1$  i  $b > 0$ , onda  $|t_l| \leq a/2$  i  $|s_l| \leq b/2$ .

*Dokaz.* (i) se jednostavno pokaže indukcijom po  $i$ . Za  $i = 0, 1$  tvrdnja je očita. Za  $i = 2, \dots, l + 1$  imamo

$$\begin{aligned} as_i + bt_i &= a(s_{i-2} - s_{i-1}q_{i-1}) + b(t_{i-2} - t_{i-1}q_{i-1}) \\ &= (as_{i-2} + bt_{i-2}) - (as_{i-1} + bt_{i-1})q_{i-1} \\ &= r_{i-2} - r_{i-1}q_{i-1} \quad (\text{indukcijom}) \\ &= r_i. \end{aligned}$$

(ii) se također lako pokaže indukcijom po  $i$ . Za  $i = 0$  tvrdnja je očita. Za  $i = 1, \dots, l$  imamo

$$\begin{aligned} s_i t_{i+1} - t_i s_{i+1} &= s_i(t_{i-1} - t_i q_i) - t_i(s_{i-1} - s_i q_i) \\ &= -(s_{i-1} t_i - t_{i-1} s_i) \\ &= -(-1)^{i-1} \quad (\text{indukcijom}) \\ &= (-1)^i. \end{aligned}$$

(iii) direktno slijedi iz (ii).

Obje tvrdnje u (iv) se mogu pokazati indukcijom po  $i$ . Tvrdnja za  $t_i$  je očita za  $i = 0$ . Za  $i = 1, \dots, l$  imamo  $t_{i+1} = t_{i-1} - t_i q_i$ . Po pretpostavci indukcije  $t_i$  i  $t_{i-1}$  imaju različite predznake i  $|t_i| \geq |t_{i-1}|$ . Slijedi da je  $|t_{i+1}| = |t_{i-1}| + |t_i| q_i \geq |t_i|$  i da su predznaci od  $t_{i+1}$  i

$t_i$  različiti. Dokaz za tvrdnju o  $s_i$  je analogan, osim što indukcija počinje sa  $i = 1$ .

(v) Promotrimo sljedeće jednadžbe:

$$as_{i-1} + bt_{i-1} = r_{i-1}$$

$$as_0 + bt_i = r_i.$$

Oduzimanjem druge jednadžbe pomnožene s  $t_{i-1}$  od prve pomnožene sa  $t_i$  i primjenom (ii) dobijemo  $\pm a = t_i r_{i-1} - t_{i-1} r_i$ . iz činjenice da  $t_i$  i  $t_{i-1}$  imaju suprotne predznake slijedi

$$a = |t_i r_{i-1} - t_{i-1} r_i| = |t_i| r_{i-1} + |t_{i-1}| r_i \geq |t_i| r_{i-1}.$$

Analogno se pokaže nejednakost za  $s_i$ .

(vi) slijedi iz (v) i sljedećeg: ako je  $a > 0$ , onda je  $r_{i-1} > 0$  za  $i = 1, \dots, l + 1$ ; ako je  $a > 1$  i  $b > 0$ , onda je  $l > 0$  i  $r_{l-1} \geq 2$ .  $\square$

Nastavimo s primjerom za osnovni Euklidov algoritam. Iz  $q_i$  se mogu jednostavno izračunati  $s_i$  i  $t_i$ :

$i$	0	1	2	3	4
$r_i$	100	35	30	5	0
$q_i$		2	1	6	
$s_i$	1	0	1	-1	7
$t_i$	0	1	-2	3	-20

Vidimo da je  $NZD(a, b) = 5 = -a + 3b$ . Shemu teorema 4.2 za zadane  $a, b \in \mathbb{Z}, a \geq b \geq 0$  lako možemo pretvoriti u sljedeći algoritam:

```

r ← a, r' ← b
s ← 1, s' ← 0
t ← 0, t' ← 1
while r' ≠ 0 do
    q ← ⌊r/r'⌋
    r'' ← r (mod r')
    (r, s, t, r', s', t') ← (r', s', t', t'', s - s'q, t - t'q)
end while
d ← r
output d, s, t

```

## 5 Računanje modularnih inverza i algoritam za kineski teorem o ostacima

### 5.1 Modularni inverzi

Bitna primjena modificiranog Euklidovog algoritma je problem računanja multiplikativnih inverza u  $\mathbb{Z}_n$

**Teorem 5.1.** *Neka su  $n, b \in \mathbb{Z}, 0 \leq b < n$ . Ako su  $b$  i  $n$  relativno prosti, možemo odrediti  $b^{-1} \pmod{n}$ .*

*Dokaz.* Možemo pretpostaviti da je  $n > 1$ , jer ako je  $n = 1$ , onda je  $b = 0 = b^{-1} \pmod{n}$ . Pomoću modificiranog Euklidovog algoritma za  $n, b$  dobijemo brojeve  $d, s, t$  takve da je  $d = NZD(n, b)$  i  $ns + bt = d$ . Ako je  $d \neq 1$  onda  $b$  nema multiplikativni inverz modulo  $n$ . Inače, za  $d = 1$ ,  $t$  je multiplikativni inverz od  $b \pmod{n}$ ; iako je moguće da nije u skupu  $\{0, \dots, n-1\}$  kao što zahtjevamo. Po (iv) teorema 4.2, slijedi  $|t| \leq n/2 < n$ . Stoga, ako je  $t \geq 0$ , onda je  $b^{-1} \pmod{n}$  jednako  $t$ ; inače,  $b^{-1} \pmod{n}$  je jednako  $t + n$ .  $\square$

Primjer: Neka su dani cijeli brojevi  $a, b, n$  i  $0 \leq a < n, 0 \leq b < n$  i želimo izračunati rješenje  $z$  kongruencije  $az \equiv b \pmod{n}$  ili pokazati da takvo rješenje ne postoji. To možemo učiniti sljedećim algoritmom

```
d ← NZD(a, n)
if d ∤ b then output "no solution"
else
  a' ← a/d
  b' ← b/d
  n' ← n/d
  t ← (a')-1 (mod n)'
  z ← tb' (mod n)'
output z
```

### 5.2 Algoritam za kineski teorem o ostacima

Primijetimo da teorem 2.1 možemo preoblikovati u oblik koji je efikasniji za računanje.

**Teorem 5.2.** *Neka su  $n_1, \dots, n_k, a_1, \dots, a_k \in \mathbb{Z}$  i neka su  $n_i$  u parovima relativno prosti,  $n_i > 1, 0 \leq a_i < n_i$  za  $i = 1, \dots, k$ . Neka je  $n := \prod_{i=1}^k n_i$ . Tada možemo odrediti jedinstveni  $a \in \mathbb{Z}$  takav da je  $0 \leq a < n$  i  $a \equiv a_i \pmod{n_i}$  za  $i = 1, \dots, k$ .*

*Dokaz.* Algoritam je jednostavna implementacija dokaza teorema 2.1 dana na sljedeći način:

```
n ← ∏i=1k ni
for i ← 1 to k do
  ni' ← n/ni
  bi ← ni · (mod ni)
  ti ← bi-1 (mod ni)
  ei ← ni · ti
end for
a ← (∑i=1k aiei) (mod n)
```

$\square$



## Literatura

- [1] Victor Shoup: A Computational Introduction to Number Theory and Algebra, Cambridge University Press, 2005.