SVEUČILIŠTE U ZAGREBU
PMF – MATEMATIČKI ODJEL

Saša Singer

Aritmetički i algebarski algoritmi

Predavanja i vježbe

Zagreb, 2007.

Sadržaj

1. Klasični algoritmi za brojeve i polinome

Prirodni brojevi predstavljaju osnovnu brojevnu algebarsku strukturu. Zbog toga, na početku, analiziramo aritmetičke algoritme za prirodne brojeve. Iz tih algoritama možemo konstruirati aritmetičke algoritme za cijele i racionalne brojeve, oponašajući konstrukciju tih algebarskih struktura na bazi strukture prirodnih brojeva.

Sama algebarska struktura — skup objekata i operacije na tom skupu, definirane su apstraktno — aksiomatski.

Algoritmi za izvođenje aritmetičkih operacija postaju potrebni pri bilo kojoj konkretnoj realizaciji te apstraktne strukture. Ti algoritmi suštinski ovise o načinu prikazivanja brojeva. Zbog toga, prije realizacije aritmetičkih algoritama, moramo odabrati prikaz objekata — u ovom slučaju, prirodnih brojeva.

1.1. Pozicioni zapis prirodnih brojeva

Nama najbliži, uobičajeni prikaz, je tzv. pozicioni zapis brojeva u odabranoj bazi b. U tom zapisu broj opisujemo nizom njegovih znamenki. Na ovaj način zapisujemo brojeve u svakodnevnom životu — u bazi $b = 10$.

Osnovu za taj prikaz daje sljedeći jednostavni teorem.

Teorem 1.1.1.

Neka je $b \in \mathbb{N}$, $b \geq 2$, bilo koji prirodni broj. Za svaki prirodni broj $u \in \mathbb{N}$, postoje broj $n \in \mathbb{N}_0 = \mathbb{N} \cup \{0\}$, i brojevi $u_0, \ldots, u_n$, takvi da je

$$u = \sum_{i=0}^{n} u_i b^i. \tag{1.1.1}$$

Brojevi $u_0, \ldots, u_n$ mogu, općenito, biti i kompleksni. Uz dodatno ograničenje

$$u_i \in \mathbb{N}_0, \quad i = 0, \ldots, n,$$

i takozvanu **normalizaciju**

$$\begin{aligned} &0 \le u_i < b, \quad i = 0, \ldots, n-1, \\ &0 < u_n < b, \end{aligned} \tag{1.1.2}$$

prikaz (1.1.1) *je jednoznačan (jedinstven).*

Dokaz:

Provodimo ga za normalizirani prikaz (1.1.2), indukcijom po $u \in \mathbb{N}$.

Broj $1 \in \mathbb{N}$ očito ima jednoznačan prikaz oblika (1.1.1), jer je $b > 1$, pa mora biti $n = 0$ i $u_0 = 1$.

Neka je $u \ge 2$ i pretpostavimo da su svi brojevi $1, 2, \ldots, u-1$ jednoznačno prikazivi u obliku (1.1.1). Za broj u i odabrani b, koristimo Euklidov teorem o dijeljenju s ostatkom, koji kaže da postoje jedinstveni brojevi v, $u_0 \in \mathbb{N}_0$, takvi da je $0 \le u_0 < b$ i vrijedi

$$u = v \cdot b + u_0. \tag{1.1.3}$$

Broj v je kvocijent, a u_0 ostatak pri dijeljenju u s b.

Ako je $v = 0$, onda je $u_0 = u > 0$, pa vrijedi (1.1.2), s $n = 0$.

Ako je $v > 0$, onda je $v < u$ (zbog $b \ge 2$), pa po pretpostavci indukcije, broj v ima jednoznačan prikaz oblika

$$v = \sum_{j=0}^{m} v_j b^j, \tag{1.1.4}$$

za neki $m \in \mathbb{N}_0$ i neke $v_j \in \mathbb{N}_0$, $j = 0, \ldots, m$, i vrijedi

$$\begin{aligned} &0 \le v_j < b, \quad j = 0, \ldots, m-1, \\ &0 < v_m < b. \end{aligned} \tag{1.1.5}$$

Iz (1.1.3) i (1.1.4) izlazi

$$u = \left(\sum_{j=0}^{m} v_j b^j \right) b + u_0 = u_0 + \sum_{j=0}^{m} v_j b^{j+1} = u_0 + \sum_{i=1}^{m+1} v_{i-1} b^i.$$

Ako definiramo $n = m + 1$ i

$$u_i = v_{i-1}, \quad i = 1, \ldots, n,$$

dobivamo prikaz (1.1.1) broja u, a relacije (1.1.5) i $0 \le u_0 < b$ daju (1.1.2).

Dokaz se može provesti i direktno, ponovljenom primjenom Euklidovog teorema o dijeljenju s ostatkom. Stavimo $q_0 = u$, pa Euklidov teorem za brojeve q_i, b daje jednoznačno određene brojeve q_{i+1}, u_i, takve da je $0 \le u_i < b$ i vrijedi

$$q_i = q_{i+1} \cdot b + u_i, \quad i = 0, 1, \ldots. \tag{1.1.6}$$

Postupak ponavljamo sve dok prvi puta ne dobijemo $q_{n+1} = 0$. To se mora dogoditi u najviše konačno mnogo koraka, jer je $q_{i+1} < q_i$, za svaki i. Tada je $q_n > 0$, $q_{n+1} = 0$, pa je $u_n = q_n > 0$, što dokazuje (1.1.2), a relacije (1.1.6) su, zapravo, Hornerov algoritam za zapis (1.1.1). ■

Definicija 1.1.1.

Broj b zovemo **baza**, *a n je* **najviša potencija baze** *ili* **stupanj** *broja u u bazi b, što označavamo s*

$$\deg_b(u) = n.$$

Brojevi $u_0, \ldots, u_n$ su **znamenke** *broja u u bazi b, a znamenku u_n zovemo* **vodeća** *ili* **najznačajnija** *znamenka broja u.*

Ako vrijedi (1.1.2), *tj. ako su znamenke normalizirane, onda relaciju* (1.1.1) *zovemo* **pozicioni zapis broja** u **u bazi** b, *i skraćeno ju označavamo s*

$$u = (u_n \ldots u_0)_b, \tag{1.1.7}$$

ili

$$u = u_n \ldots u_0, \tag{1.1.8}$$

ako je iz konteksta jasno u kojoj bazi b se vrši prikaz.

Relacija (1.1.8) je, zapravo, naš uobičajeni prikaz brojeva s bazom $b = 10$.

Napomena 1.1.1. *Teorem* 1.1.1. *može se formulirati i mnogo općenitije — za realne i kompleksne brojeve u. Neke generalizacije za realne brojeve ćemo kasnije formulirati i koristiti. Niz drugih, koje uključuju i kompleksne baze opisane su u [Knuth, 1981. (poglavlje 4.1)].*

Napomena 1.1.2. *Za bilo koji cijeli broj $u \in \mathbb{Z}$ i prirodni broj $b \in \mathbb{N}$, Euklidov teorem daje egzistenciju i jedinstvenost kvocijenta q i ostatka r, uz $0 \leq r < b$, za koje vrijedi*

$$u = q \cdot b + r. \tag{1.1.9}$$

Kvocijent označavamo s

$$q = \lfloor u/b \rfloor = u \text{ div } b,$$

a ostatak s

$$r = u \bmod b.$$

U tim oznakama, relacija (1.1.9) *glasi*

$$u = \lfloor u/b \rfloor \cdot b + u \bmod b = (u \text{ div } b) \cdot b + u \bmod b. \tag{1.1.10}$$

Pozicioni zapis brojeva ima niz korisnih svojstava, koja sumiramo u sljedećoj tvrdnji, a kasnije koristimo za dokaz korektnosti aritmetičkih algoritama.

Propozicija 1.1.1.

Neka je $b \in \mathbb{N}$, $b \geq 2$, bilo koja odabrana baza i neka je $u \in \mathbb{N}$ broj dan normaliziranim pozicionim zapisom

$$u = \sum_{i=0}^{n} u_i b^i.$$

Za svaki $i \in \mathbb{N}_0$ vrijedi

$$u = (u \operatorname{div} b^i) \cdot b^i + u \bmod b^i,$$

gdje su

$$q_i = \lfloor u/b^i \rfloor = u \operatorname{div} b^i = \begin{cases} \sum_{j=i}^{n} u_j b^{j-i}, & za\ i = 0, \ldots, n, \\ 0, & za\ i > n, \end{cases} \tag{1.1.11}$$

brojevi iz relacije (1.1.6), *i*

$$u \bmod b^i = \begin{cases} \sum_{j=0}^{i-1} u_j b^j, & za\ i = 1, \ldots, n, \\ u, & za\ i > n. \end{cases} \tag{1.1.12}$$

Za $0 < i < n$, pozicioni prikazi "gornjeg" i "donjeg" dijela broja u obzirom na b^i imaju oblik

$$u \operatorname{div} b^i = (u_n \ldots u_i)_b, \quad u \bmod b^i = (u_{i-1} \ldots u_0)_b.$$

Postoje dva zgodna načina za "izdvajanje" i-te znamenke u_i iz broja u

$$u_i = \lfloor u/b^i \rfloor \bmod b = \left\lfloor \frac{u \bmod b^{i+1}}{b^i} \right\rfloor, \quad i = 0, \ldots, n. \tag{1.1.13}$$

Ovu relaciju za znamenke možemo zapisati i u obliku

$$u_i = (u \operatorname{div} b^i) \bmod b = (u \bmod b^{i+1}) \operatorname{div} b^i, \quad i = 0, \ldots, n.$$

Dokaz:

Direktan, korištenjem relacija (1.1.2) i Euklidovog teorema. ■

Odsada uvijek radimo s normaliziranim prikazom, osim ako posebno nije naglašeno suprotno. Tada je najviša potencija baze dana s

$$\deg_b(u) = \lfloor \log_b u \rfloor.$$

Broj znamenki u prikazu broja direktno određuje količinu memorije potrebne za prikaz broja u računalu i broj operacija u realizaciji aritmetičkih algoritama.

Definicija 1.1.2.

Duljina broja u **u bazi b**, *u oznaci* $\ell_b(u)$, *je broj znamenki u pozicionom prikazu broja* u *u bazi* b

$$\ell_b(u) = \deg_b(u) + 1 = \lfloor \log_b u + 1 \rfloor. \tag{1.1.14}$$

U opisu aritmetičkih algoritama koji rade s brojevima moramo prikazati podatke koji se sastoje od (konačnog) niza znamenki broja. Za taj prikaz koristimo apstraktnu strukturu podataka koju zovemo **niz**.

Definicija 1.1.3.

Niz u, *podataka istog tipa iz nekog skupa* $\mathcal{U}$, *je bilo koji element skupa*

$$\mathrm{Seq}(\mathcal{U}) = \bigcup_{k=0}^{\infty} \mathcal{U}^k.$$

Pošto je ova unija disjunktna, za bilo koji $u \in \mathrm{Seq}(\mathcal{U})$ *postoji jednoznačno određeni broj* $k \in \mathbb{N}_0$ *takav da je*

$$u \in \mathcal{U}^k,$$

pa je dobro definirana funkcija $\ell : \mathrm{Seq}(\mathcal{U}) \to \mathbb{N}_0$ *s*

$$\ell(u) = k \iff u \in \mathcal{U}^k.$$

Broj $\ell(u)$ *zovemo* **duljina niza** u.

Pretpostavljamo da je $\mathcal{U}^0 = \{\emptyset\}$. Ako je $\ell(u) = 0$, onda kažemo da je u **prazan niz**, što označavamo s

$$u = (\ ).$$

Ako je $\ell(u) > 0$, onda niz u označavamo na dva načina, prema potrebi:

(a) indeksima iz $\mathbb{N}_0$, i tada pišemo

$$u = (u_0, \ldots, u_n) \quad \text{ili} \quad u = (u_n, \ldots, u_0),$$

gdje je $n = \ell(u) - 1$, i definiramo **stupanj niza**

$$\deg(u) = n = \ell(u) - 1;$$

(b) indeksima iz $\mathbb{N}$, kada pišemo

$$u = (u_1, \ldots, u_n) \quad \text{ili} \quad u = (u_n, \ldots, u_1).$$

U ovom slučaju nema potrebe za definicijom stupnja niza.

U ovoj oznaci, niz u možemo interpretirati i kao bilo koji dobro uređeni konačni podskup skupa $\mathcal{U}$. Pri tome, uređaj prenosimo s početnih komada skupa $\mathbb{N}_0$ ili skupa $\mathbb{N}$, u prirodnom ili obratnom poretku.

Za neprazan niz u, podatke $u_0, \ldots, u_n$ (odnosno, $u_1, \ldots, u_n$) možemo smatrati elementima ili članovima niza.

Smatramo da je niz u zadan duljinom $\ell(u)$ (ili stupnjem $\deg(u)$, za prikaz (a)), i elementima niza, ako je $\ell(u) > 0$, tj. ako niz nije prazan.

Pri tome ne definiramo nikakve operacije za nizove. Jedine dozvoljene operacije su operacije definirane za osnovne podatke iz skupa $\mathcal{U}$.

Definicija niza i teorem 1.1.1. omogućavaju nam da svaki prirodni broj $u \in \mathbb{N}$ jednoznačno prikažemo nizom u' njegovih znamenki u odabranoj bazi b

$$u' = (u_n, \ldots, u_0). \tag{1.1.15}$$

Tada je $\ell(u') = \ell_b(u)$ i $\deg(u') = \deg_b(u)$. Ako je prikaz broja normaliziran, tj. vrijede relacije (1.1.2), onda za osnovni skup $\mathcal{U}$ možemo uzeti standardni skup ostataka modulo b

$$\mathbb{Z}_b = \{0, 1, \ldots, b-1\}.$$

Pridruživanje $u \mapsto u'$ je korektno definirana funkcija $\mathbb{N} \to \mathrm{Seq}(\mathbb{Z}_b)$, koja je injekcija, ali nije surjekcija (jer vodeća znamenka mora biti pozitivna).

Barem djelomično, vrijedi i obrat. Ako je $u' \in \mathrm{Seq}(\mathbb{N}_0)$ bilo koji neprazan niz oblika (1.1.15), i ako članovi niza zadovoljavaju relacije (1.1.2), onda je u' niz znamenki broja

$$u = \sum_{i=0}^{n} u_i b^i.$$

Zbog toga možemo identificirati broj u i niz njegovih znamenki u bazi b, oznakom

$$u = (u_n, \ldots, u_0),$$

što opravdava i raniju oznaku (1.1.7) za pozicioni zapis u bazi b.

U nizu algoritama koji koriste zbrajanja u petlji, javlja se potreba za brojem 0 pri inicijalizaciji. Zbog toga je vrlo korisno prirodnim brojevima dodati nulu, i prirodnu aritmetiku realizirati na skupu $\mathbb{N}_0$.

$\mathbb{N}_0$ ima i jaču algebarsku strukturu — monoida obzirom na zbrajanje, u odnosu na $\mathbb{N}$, koji je samo aditivna polugrupa.

Napomena 1.1.3. *Broj* 0 **nema** *normalizirani pozicioni prikaz, jer bi vodeća znamenka morala biti pozitivna, prema* (1.1.2).

Za nulu definiramo poseban pozicioni prikaz, kojeg onda realiziramo strukturom niza.

Definicija 1.1.4.

Broj $0 \in \mathbb{N}_0$ *prikazujemo* **praznim** *nizom znamenki*

$$0 = (\ )_b$$

u bilo kojoj bazi $b \geq 2$. *Duljina je prirodno definirana s*

$$\ell_b(0) = 0,$$

a najvišu potenciju (stupanj) definiramo s

$$\deg_b(0) = -1,$$

tako da i dalje vrijedi veza između duljine i stupnja broja iz (1.1.14).

Napomena 1.1.4. *Drugo moguće rješenje za prikaz nule je*

$$0 = (0)_b,$$

tj. tako da definiramo najvišu potenciju s $\deg_b(0) = 0$ *i* $u_0 = 0$. *Ovo rješenje koristimo u svakodnevnom životu, ali to, osim što narušava* (1.1.2), *dovodi i do niza nepotrebnih komplikacija pri realizaciji algoritama.*

Reprezentaciju brojeva $u \in \mathbb{N}_0$ nizom znamenki u nekoj bazi b je lako programski realizirati, uz uvjet da su znamenke u_i uvijek prikazive u računalu. Od osnovnih tipova podataka prikazivih u računalu, najzgodnije je znamenke prikazati **cijelim brojevima**, jer nam to kasnije omogućava da cjelobrojnu aritmetiku računala iskoristimo za izvođenje aritmetičkih operacija na znamenkama. Ovo je ključna ideja za razvoj aritmetičkih algoritama. U protivnom, morali bismo sami realizirati i osnovne operacije na znamenkama.

Označimo s *maxint* najveći prikazivi cijeli broj u računalu (u nekom izabranom tipu cijelih brojeva kojeg podržava arhitektura računala). Zahtjev prikazivosti znamenki broja u znači

$$u_i \leq maxint, \quad i = 0, \ldots, n.$$

Iz relacije (1.1.2) izlazi prvi uvjet na dopuštenu veličinu baze

$$b \leq maxint + 1. \tag{1.1.16}$$

Kasnije ćemo postaviti još strože zahtjeve na izbor baze b.

Pedantno govoreći, ograničenje (1.1.16) na bazu b osigurava samo to da je svaka pojedina znamenka u_i prikaziva. Da bismo broj u spremili u računalu, moramo moći spremiti **sve** njegove znamenke, tj. količina raspoložive memorije (koja je konačna), postavlja praktičnu gornju granicu na n. Ovo ograničenje na n ćemo, uglavnom,

ignorirati u razradi algoritama, u smislu da će algoritmi, u principu, raditi korektno za bilo koje duljine brojeva. Naravno, u praksi, tim algoritmima treba dodati kontrolu duljine brojeva.

Precizan izbor strukture podataka kojom ćemo realizirati pozicioni prikaz brojeva, odgađamo sve dok ne analiziramo pripadne aritmetičke algoritme. To nam daje dovoljnu općenitost da uočimo i analiziramo suštinu algoritama, neopterećenih detaljima strukture podataka. Strukturu podataka ćemo odabrati tako da ona omogući maksimalno efikasnu realizaciju ovih općih algoritama.

Pošto smo analizirali osnovne aritmetičke algoritme za prirodne i cijele brojeve, možemo razmotriti detaljnu realizaciju strukture podataka za prikaz brojeva.

Do sada smo koristili opću strukturu niza, koja dozvoljava nekoliko bitno različitih realizacija.

Ako **ne želimo** ograničiti maksimalnu duljinu niza (tj. raspon brojeva u aritmetici), prirodno je koristiti dinamičko raspolaganje memorijom. U svim algoritmima, znamenke brojeva obrađujemo sekvencijalno – rastuće ili padajuće po pripadnim potencijama baze. Zbog toga je dovoljno koristiti linearnu strukturu podataka – jednostruko ili dvostruko vezanu listu.

U Pascalu možemo definirati tip *mpnat* strukturom vezane liste

```
digitptr = ↑digitrec;
digitrec = record
             digit : integer;
             next : digitptr
           end;
mpnat    = record
             deg : integer;
             digits : digitptr
           end;
```

Predost ove realizacije je varijabilna duljina brojeva. To međutim, zahtijeva efikasno upravljanje memorijom, posebno pri kreiranju ili brisanju znamenki kod povećanja ili smanjenja duljine broja. Najčešće te operacije nisu efikasno izvedene, pa je aritmetika varijabilne duljine brojeva relativno spora.

Ograničimo li duljinu brojeva, možemo koristiti fiksne blokove memorije za prikaz brojeva, što omogućava vrlo brz pristup znamenkama. Tada u sve algoritme treba ugraditi kontrolu duljine brojeva, za osiguranje korektnosti algoritama.

Realizacija tipa *mpnat* poljem je, na primjer,

```
mpnat = record
          deg : integer;
          digit : array [0 .. maxdeg] of integer;
        end;
```

gdje je konstanta *maxdeg* najveći dozvoljeni stupanj broja.

Ova realizacija prikaza brojeva je znatno brža i jednostavnija, pa se vrlo često koristi. Osim toga, u nekim programskim jezicima (FORTRAN) nije standardno moguće dinamičko upravljanje memorijom.

1.1.1. Ovisnost složenosti o bazi

Za aritmetičke algoritme, duljina ulaznih brojeva je mjera veličine zadaće. Zbog toga, složenost aritmetičkih algoritama izražavamo kao funkciju duljine ulaznih brojeva. Međutim, duljina broja iz (1.1.14) ovisi o bazi, pa i složenost ovisi o bazi.

Da analiziramo tu ovisnost, potrebno je naći vezu između duljina brojeva u različitim bazama.

Teorem 1.1.2.

Duljine prirodnih brojeva u bilo koje dvije baze su kodominantne.

Vrijedi i jače — ako su b_1, $b_2 \geq 2$, bilo koje dvije baze, onda je

$$\ell_{b_2}(u) \sim \log_{b_2} b_1 \cdot \ell_{b_1}(u), \tag{1.1.17}$$

tj. duljine brojeva su asimptotski proporcionalne za velike brojeve.

Dokaz:

Očito je, po definiciji, da je duljina prirodnog broja $\ell_b(u)$ nenegativna rastuća funkcija za bilo koju bazu b.

Definiramo omjer duljina kao funkciju

$$f(u) = \frac{\ell_{b_2}(u)}{\ell_{b_1}(u)} = \frac{\lfloor \log_{b_2} u + 1 \rfloor}{\lfloor \log_{b_1} u + 1 \rfloor}.$$

Za $u \geq b_1$, koristeći relaciju $x - 1 < \lfloor x \rfloor \leq x, \ \forall x \in \mathbb{R}$, dobivamo

$$f(u) < \frac{\log_{b_2} u + 1}{\underbrace{\log_{b_1} u}_{\neq 0, \text{ jer } u>1}} = \log_{b_2} b_1 + \frac{1}{\log_{b_1} u}. \tag{1.1.18}$$

Kako je $u \geq b_1 \implies \log_{b_1} u \geq 1$, to je

$$f(u) < \log_{b_2} b_1 + 1. \tag{1.1.19}$$

Relacija (1.1.19) vrijedi i za $u < b_1$, jer je tada $\ell_{b_1}(u) = 1$, pa je

$$f(u) = \ell_{b_2}(u) \leq \log_{b_2} u + 1 < \log_{b_2} b_1 + 1.$$

S druge strane je

$$f(u) > \frac{\log_{b_2} u}{\log_{b_1} u + 1} = \log_{b_2} b_1 \cdot \frac{\log_{b_1} u}{\log_{b_1} u + 1}. \tag{1.1.20}$$

Za $u \geq b_1$ je

$$\frac{\log_{b_1} u}{\log_{b_1} u + 1} \geq \frac{1}{2},$$

što sa (1.1.19) daje

$$\frac{1}{2} \log_{b_2} b_1 < f(u) < \log_{b_2} b_1 + 1, \tag{1.1.21}$$

čim je $u \geq b_1$. To znači da je

$$\ell_{b_2}(u) = \Theta(\ell_{b_1}(u)),$$

pa su $\ell_{b_1}(u)$ i $\ell_{b_2}(u)$ kodominantne, po propoziciji A.2.1..

Iz relacija (1.1.18) i (1.1.20) dobivamo

$$\log_{b_2} b_1 \cdot \frac{\log_{b_1} u}{\log_{b_1} u + 1} < f(u) < \log_{b_2} b_1 + \frac{1}{\log_{b_1} u}.$$

Zbog $\log_{b_1} u \to \infty$ kad $u \to \infty$, odavde izlazi

$$\lim_{u \to \infty} f(u) = \log_{b_2} b_1,$$

što dokazuje (1.1.17). ■

Ovo je vrlo važan rezultat.

Dokazat ćemo da su složenosti svih algoritama koje ćemo promatrati, regularne funkcije duljine ulaznih brojeva.

Primjenom teorema B.1.3. dobivamo :

Složenost aritmetičkih algoritama ne ovisi bitno o bazi pozicionog prikaza.

Napomena 1.1.5. *Ovaj rezultat, naravno, treba shvatiti uvjetno, jer još nismo analizirali niti jedan aritmetički algoritam. Neovisnost složenosti o bazi dokazujemo posebno za svaki pojedini algoritam. Zbog toga, navedenu tvrdnju treba interpretirati kao zajednički izraz svih tih pojedinačnih rezultata.*

Složenost aritmetičkih algoritama ćemo izražavati kao funkciju duljine

$$\ell(u)$$

ulaznog broja u, za svaki ulazni broj. Sa stanovišta realizacije algoritma, to je naprosto broj osnovnih podataka prikazivih u računalu, potrebnih za reprezentaciju (kodiranje) broja u. To omogućava da se složenost analizira, ne vodeći računa o vrsti prikaza broja u. Opravdanje se može naći i u tom što pozicioni zapis nije jedini način prikaza brojeva. Ako je riječ o pozicionom zapisu, onda $\ell(u)$ može biti zamijenjen s $\ell_b(u)$ u odabranoj bazi b.

Kad pokažemo da je tako izražena složenost regularna funkcija, dobivamo naknadno opravdanje ove oznake, jer se složenost neće bitno izmijeniti ni za bilo koji drugi razumni prikaz (ili bazu).

1.2. Zbrajanje

Pozicioni zapis u bazi b izgleda kao pokušaj reprezentacije prirodnih brojeva polinomima nad prstenom $(\mathbb{Z}, +, \cdot)$ cijelih brojeva ili prstenom $(\mathbb{Z}_b, +_b, \cdot_b)$ ostataka modulo b.

Sličnost postoji, jer najviša potencija baze odgovara stupnju polinoma, a znamenke broja odgovaraju koeficijentima polinoma.

Aritmetičke operacije u pozicionom prikazu se očito svode na odgovarajuće operacije za znamenke — slično operacijama za koeficijente u polinomnoj aritmetici. Ta analogija vrijedi za zbrajanje, oduzimanje i množenje. Zahtjev normaliziranosti znamenki u rezultatu, narušava njihovu nezavisnost, jer dolazi do pojave prijenosa.

To pokazuje da aritmetika brojeva u pozicionom zapisu i polinomna aritmetika nisu izomorfne.

Bliskost struktura opravdava paralelno izlaganje osnovnih aritmetičkih algoritama u obje strukture.

Polinom $U(x)$, oblika

$$U(x) = \sum_{i=0}^{n} u_i x^i,$$

je zadan stupnjem $\deg(U) = n$ i nizom koeficijenata $(u_0, \ldots, u_n)$. Pretpostavit ćemo da su koeficijenti nenegativni, što osigurava da je

$$\deg(U + V) = \max\{\deg(U), \deg(V)\}.$$

Algoritam 1.2.1. (PADD1 — **Zbrajanje polinoma**)

Ulaz: Nizovi koeficijenata $(u_0, \ldots, u_n)$, $(v_0, \ldots, v_m)$ polinoma

$$U(x) = \sum_{i=0}^{n} u_i x^i, \quad V(x) = \sum_{i=0}^{m} v_i x^i,$$

uz pretpostavku da je $n = \deg(U) \geq \deg(V) = m$.

Izlaz: Niz koeficijenata $(w_0, \ldots, w_n)$ polinoma

$$W(x) = U(x) + V(x).$$

procedure PADD1
(U, V : *polinom* ;
var W : *polinom*) ;

begin
$\deg(W) := \deg(U)$;
for $i = 0$ **to** $\deg(V)$ **do**
$w_i := u_i + v_i$;
for $i = \deg(V) + 1$ **to** $\deg(U)$ **do**
$w_i := u_i$
end; { PADD1 }

Tip *polinom* ima strukturu niza.

Ovaj algoritam radi korektno, ako je svaki zbroj $u_i + v_i$ prikaziv u računalu. Ako realiziramo aritmetiku polinoma nad prstenom $(\mathbb{Z}_b, +_b, \cdot_b)$, onda $u_i + v_i$ treba reducirati u odabrani sistem ostataka modulo b. Ako uzmemo $\mathbb{Z}_b = \{0, 1, \ldots, b-1\}$, treba definirati

$$w_i = (u_i + v_i) \bmod b. \tag{1.2.1}$$

U tom slučaju polinom W može imati i manji stupanj od polinoma U. Taj fenomen ćemo pokazati kod oduzimanja.

Prostorna složenost ovog algoritma je

$$\mathrm{Compl}_S(\mathsf{PADD1}) = 2\ell(U) + \ell(V) + 4,$$

računajući $\ell(W) = \ell(U)$, prostor za spremanje duljine ili stupnja (jedno mjesto) za svaki od 3 polinoma, i prostor za pomoćnu varijablu i.

Aritmetička složenost je

$$\mathrm{Compl}_A(\mathsf{PADD1}) = \ell(V),$$

ako ne računamo kopiranje viših koeficijenata iz U u W, dok za vremensku složenost vrijedi

$$\mathrm{Compl}_T(\mathsf{PADD1}) \sim \ell(U).$$

Prostorna i vremenska složenost mogu se smanjiti na $\ell(U)+\ell(V)+3$, odnosno $\ell(V)$, ako rezultat W spremimo na mjesto polinoma U.

Kod zbrajanja brojeva, znamenke rezultata moraju biti normalizirane, pa je redukcija raspona oblika (1.2.1) nužna. Za razliku od aritmetike polinoma, ovdje i kvocijent $\lfloor (u_i + v_i)/b \rfloor$ treba uključiti u zbroj sljedeće dvije znamenke.

Algoritam 1.2.2. (NADD1 — **Zbrajanje prirodnih brojeva)**

Ulaz: Nizovi znamenki $(u_n, \ldots, u_0)$, $(v_m, \ldots, v_0)$ brojeva u i v u odabranoj bazi b. Pretpostavljamo da je

$$n = \deg_b(u) \geq \deg_b(v) = m. \tag{1.2.2}$$

Izlaz: Niz znamenki $(w_k, \ldots, w_0)$ broja

$$w = u + v$$

u bazi b.

```
procedure NADD1
        (       u, v : mpnat ;
          var w : mpnat) ;

  begin
    carry := 0 ;
    for i = 0 to deg(v) do
      begin
      w_i := (u_i + v_i + carry) mod b ;
      carry := ⌊(u_i + v_i + carry)/b⌋
      end ;
    for i = deg(v) + 1 to deg(u) do
      begin
      w_i := (u_i + carry) mod b ;
      carry := ⌊(u_i + carry)/b⌋
    end;
  if carry = 0 then
    deg(w) := deg(u)
  else
    begin
    deg(w) := deg(u) + 1 ;
```

```
    w_deg(w) := carry
  end
end ; { NADD1 }
```

Tip podataka *mpnat* je neka realizacija pozicionog prikaza prirodnih brojeva strukturom niza.

Dokažimo da ovaj algoritam radi korektno u egzaktnoj aritmetici, tj. da za **svaki** ulaz daje točan rezultat.

Propozicija 1.2.1.

Ako su brojevi $u, v \in \mathbb{N}$ *dani pozicionim zapisima u bazi* b

$$u = (u_n \dots u_0)_b, \quad v = (v_m \dots v_0)_b,$$

i ako je $n \geq m$*, onda algoritam* NADD1 *daje niz znamenki pozicionog zapisa broja* $u + v$ *u bazi* b

$$u + v = w = (w_k \dots w_0)_b,$$

gdje je $k = \deg_b(w) \in \{n, n+1\}$.

Dokaz:

Označimo $v_i = 0$, $i = m+1, \dots, n$, u slučaju da je $m < n$. U ovoj oznaci, obje petlje u algoritmu imaju identičnu strukturu, što omogućava zajedničku analizu.

Označimo sa $carry_{-1} = 0$ početnu vrijednost varijable *carry*, a sa $carry_i$, $i = 0, \dots, n$ vrijednost te varijable nakon i-tog prolaza kroz petlju. (i-ti prolaz = prolaz s kontrolnom vrijednošću i)

Tvrdimo da u svakom koraku algoritam daje korektan zbroj donjih dijelova ulaznih brojeva, tj. da je nakon i-tog koraka

$$carry_i \cdot b^{i+1} + \sum_{j=0}^{i} w_j b^j = \sum_{j=0}^{i} u_j b^j + \sum_{j=0}^{i} v_j b^j. \tag{1.2.3}$$

Rad algoritma možemo opisati rekurzivnim relacijama

$$\left.\begin{aligned} carry_{-1} &= 0 \\ w_i &= (u_i + v_i + carry_{i-1}) \bmod b \\ carry_i &= \lfloor (u_i + v_i + carry_{i-1})/b \rfloor \end{aligned}\right\} \quad i = 0, \dots, n. \tag{1.2.4}$$

Odavde je očito

$$0 \leq w_i < b, \quad i = 0, \dots, n. \tag{1.2.5}$$

Osim toga, relacije (1.2.4) možemo pisati i u obliku:

$$carry_i \cdot b + w_i = u_i + v_i + carry_{i-1} \tag{1.2.6}$$

za $i = 0, \ldots, n$.

Relaciju (1.2.3) dokazujemo indukcijom po i. Za $i = 0$, zbog $carry_{-1} = 0$, relacija (1.2.6) glasi

$$carry_0 \cdot b + w_0 = u_0 + v_0,$$

što je upravo (1.2.3) za $i = 0$.

Pretpostavimo da nakon $(i-1)$-og koraka vrijedi (1.2.3) sa $i-1$ umjesto i. Onda je

$$\begin{aligned} carry_i \cdot b^{i+1} + \sum_{j=0}^{i} w_j b^j &= (carry_i \cdot b + w_i) b^i + \sum_{j=0}^{i-1} w_j b^j = (\text{po (1.2.6)}) \\ &= (u_i + v_i + carry_{i-1}) b^i + \sum_{j=0}^{i-1} w_j b^j = \left(\begin{smallmatrix} \text{pretpostavka} \\ \text{indukcije} \end{smallmatrix} \right) \\ &= (u_i + v_i) b^i + \sum_{j=0}^{i-1} u_j b^j + \sum_{j=0}^{i-1} v_j b^j \\ &= \sum_{j=0}^{i} u_j b^j + \sum_{j=0}^{i} v_j b^j. \end{aligned}$$

Time je relacija (1.2.3) dokazana.

Ovu relaciju možemo, prema (1.1.12), pisati u obliku:

$$carry_i \cdot b^{i+1} + w \bmod b^{i+1} = u \bmod b^{i+1} + v \bmod b^{i+1},$$

za $i = 0, \ldots, n$. Za $i = n = \deg_b(u)$ dobivamo

$$carry_n \cdot b^{n+1} + w \bmod b^{n+1} = u + v. \tag{1.2.7}$$

Brojevi u, v su normalizirani, pa je

$$\begin{matrix} 0 \le u_i \le b-1, \\ 0 \le v_i \le b-1, \end{matrix} \quad i = 0, \ldots, n. \tag{1.2.8}$$

Zbog toga, iz (1.2.4) izlazi da je

$$0 \le carry_i \le 1, \quad i = -1, \ldots, n. \tag{1.2.9}$$

Dokaz je indukcijom po i. Za $i = -1$, to vrijedi direktno iz algoritma (1.2.4). Iz pretpostavke $0 \le carry_{i-1} \le 1$, za $i \ge 0$ i (1.2.8) slijedi

$$0 \le u_i + v_i + carry_{i-1} \le 2b - 1 < 2b, \tag{1.2.10}$$

pa je

$$0 \leq \lfloor (u_i + v_i + carry_{i-1})/b \rfloor < 2.$$

Po definiciji veličine $carry_i$ iz (1.2.4) izlazi (1.2.9).

Analizirajmo relaciju (1.2.7). Ako je $carry_n = 0$, onda algoritam postavlja

$$\deg_b(w) = \deg_b(u) = n,$$

pa iz (1.1.12) izlazi

$$w \bmod b^{n+1} = w.$$

Relacija (1.2.7) tada glasi

$$w = u + v.$$

Kako je $u_n > 0$, iz (1.2.6) sa $i = n$, slijedi

$$w_n = u_n + v_n + carry_{n-1} \geq u_n > 0.$$

Zajedno sa (1.2.5), to pokazuje da je

$$w = (w_n \dots w_0)_b$$

normalizirani prikaz broja $w = u + v$.

Ako je $carry_n > 0$, onda je $carry_n = 1$ i algoritam postavlja

$$\begin{aligned} \deg_b(w) &= n + 1 \\ w_{n+1} &= carry_n = 1, \end{aligned}$$

pa je

$$w = (w_{n+1} \dots w_0)_b$$

normalizirani prikaz broja $w = u + v$. ■

Da bi algoritam radio korektno i u aritmetici računala, sve operacije moraju biti egzaktno izvodive.

Pretpostavit ćemo da su osnovne aritmetičke operacije računala egzaktne, ako je, osim operanada i rezultat operacije egzaktno prikaziv, tj. nalazi se u rasponu između $-maxint$ i $maxint$.

Relacija (1.2.6) pokazuje da je $u_i + v_i + carry$ najveća vrijednost koju algoritam računa (w_i i novi $carry$ su manji ili jednaki toj vrijednosti). Prema (1.2.10) je

$$0 \leq u_i + v_i + carry \leq 2b - 1$$

što pokazuje da $2b - 1$ mora biti prikaziv. Time je dokazana

Propozicija 1.2.2.

Algoritam NADD1 *radi korektno u aritmetici računala, ako i samo ako baza b zadovoljava uvjet*

$$2b - 1 \leq maxint. \tag{1.2.11}$$

■

Ovo ograničenje je zbog $b \geq 2$, jače od ograničenja (1.1.16) koje garantira prikazivost podataka.

Napomena 1.2.1. *Algoritam se može preurediti tako da nakon svake pojedine operacije zbrajanja spremamo i analiziramo dobiveni rezultat. U tom slučaju se ograničenje (1.2.11) može oslabiti do*

$$2b - 2 \leq maxint.$$

To je i najblaže moguće ograničenje, ako zbroj pojedinačnih znamenki mora biti egzaktan. Moguće je, naravno, algoritam zbrajanja realizirati i uz ograničenje (1.1.16). Međutim, broj operacija i potrebno vrijeme rastu toliko, da sve ove modifikacije nisu opravdane.

Algoritam NADD1, u izloženom obliku, je vrlo neefikasan. Računanje znamenki rezultata i prijenosa sadrži cjelobrojno dijeljenje s ostatkom, a to je najsporija operacija u aritmetici računala.

Međutim, relacije (1.2.9) i (1.2.10) pokazuju da kvocijent može biti samo 0 ili 1, ovisno o tome da li je $u_i + v_i + carry < b$ ili ne. To znači da vrijednost kvocijenta (prijenosa) možemo otkriti jednim uspoređivanjem.

Druga petlja u algoritmu izvršava se samo ako je broj u dulji od v. Jedini njen posao je propagiranje prijenosa koji nastaje zbrajanjem donjeg dijela broja u i broja v, kroz gornji dio broja u. Tada je u relaciji (1.2.10) $v_i = 0$ za $i = m+1, \ldots, n$, što daje

$$0 \leq u_i + carry_{i-1} \leq b, \quad i = m+1, \ldots, n. \tag{1.2.12}$$

Uočimo da je tada

$$carry_i = 1 \iff u_i + carry_{i-1} = b,$$

što znači da mora biti $u_i = b - 1$ i $carry_{i-1} = 1$, tj. za $i > m$ je

$$carry_i = 1 \iff u_i = b - 1 \quad \text{i} \quad carry_{i-1} = 1. \tag{1.2.13}$$

Dakle, prijenos se propagira ako i samo ako je znamenka maksimalna.

S druge strane, ako za bilo koji i, $i > m$ dobijemo $carry_{i-1} = 0$, onda algoritam daje

$$w_i = u_i$$
$$carry_i = 0,$$

tj. od tog mjesta nadalje, prijenos ostaje 0 i znamenke broja u se kopiraju u znamenke broja w.

Ove rezultate pregledno formuliramo u obliku sljedeće tvrdnje.

Propozicija 1.2.3.

Ako u algoritmu NADD1 *vrijedi*

$$n = \deg_b(u) > \deg_b(v) = m,$$

onda je, u oznakama iz propozicije 1.2.1. , za svaki $i = m+1, \ldots, n$ *ispunjeno*

$$\begin{array}{llllll} carry_{i-1} = 1 & i & u_i = b-1 & \Longleftrightarrow & w_i = 0 & i \quad carry_i = 1 \\ carry_{i-1} = 1 & i & u_i < b-1 & \Longleftrightarrow & w_i = u_i + 1 & i \quad carry_i = 0 \\ carry_{i-1} = 0 & & & \Longleftrightarrow & w_i = u_i & i \quad carry_i = 0 \ . \end{array}$$

U svakom koraku nastupa točno jedan od tri slučaja na lijevoj strani. Pri tome drugi slučaj može nastupiti najviše jednom, kao prijelaz iz prvog u treći. ■

Potreba za ugrađivanjem ovih rezultata u algoritam ovisi o tome koliko često zbrajamo brojeve različitih duljina.

Razumno je pretpostaviti da se to relativno često događa, jer svako akumuliranje suma daje sve veće brojeve. Zato se isplati ugraditi ove rezultate u algoritam, iako to znatno povećava broj naredbi.

Time dobivamo novu verziju algoritma NADD1.

Algoritam 1.2.3. (NADD1 — **Zbrajanje prirodnih brojeva)**

Ulaz: Nizovi znamenki $(u_n, \ldots, u_0)$, $(v_m, \ldots, v_0)$ brojeva u i v u odabranoj bazi b, uz pretpostavku da je

$$n = \deg_b(u) \geq \deg_b(v) = m.$$

Izlaz: Niz znamenki $(w_k, \ldots, w_0)$ broja

$$w = u + v$$

u bazi b.

```
procedure NADD1
        (       u, v : mpnat ;
          var   w : mpnat) ;

  begin
```

```
carry := 0 ;
for i = 0 to deg(v) do
  begin
  temp := u_i + v_i + carry ;
  if temp < b then
    begin
    w_i := temp;
    carry := 0
    end
  else
    begin
    w_i := temp − b;
    carry := 1
    end ;
  end; { for }
if deg(u) > deg(v) then
  begin
  i := deg(v) + 1 ;
  if carry = 1 then
    begin
    while (u_i = b − 1) and (i < deg(u)) do
      begin
      w_i := 0 ;
      i := i + 1
      end;
    if u_i < b − 1 then
      begin
      w_i := u_i + 1;
      carry := 0
      end
    else
      w_i := 0 ;    { i = deg(u) }
    i := i + 1;
    end ; { if }
  for j = i to deg(u) do w_j := u_j ;
  end ;
if carry = 0 then
  deg(w) := deg(u)
else
  begin
  deg(w) := deg(w) + 1;
  w_deg(w) := 1
```

```
    end ;
end; { NADD1 }
```

Napomena 1.2.2. *Ovo je samo jedna od niza mogućih varijanti ovog algoritma. Na pr. varijabla carry se može potpuno eliminirati, dok izložena varijanta zbraja carry i onda kad znamo da je* $carry = 0$.

Analiza složenosti algoritma NADD1 je izrazito ovisna o detaljima realizacije. Rezultate o složenosti ćemo namjerno formulirati pomalo neprecizno, da osiguramo njihovu općenitost, tj. da oni vrijede za sve razumne varijante algoritma.

Prostorna složenost ovog algoritma je

$$\mathrm{Compl}_S(\mathsf{NADD1}) = 2\,\ell\,(u) + \ell\,(v) + c\ , \tag{1.2.14}$$

gdje je $c \leq 8$. Ova konstanta uključuje mogućnost da je $\ell\,(w) = \ell\,(u) + 1$, te prostor za duljine brojeva i pomoćne varijable *carry* , *temp* , i , j.

Aritmetička i vremenska složenost donekle ovise o broju prijenosa, čim koristimo činjenicu da je $carry \in \{\ 0\ ,\ 1\ \}$ za izbjegavanje dijeljenja s ostatkom. Zbog ovisnosti o ulaznim podacima, potrebno je naći gornju i donju granicu složenosti – u ovisnosti samo o veličini ulaza. Ta dva pojma posebno označavamo sa

$$\begin{aligned} \text{b – Compl} &= \text{složenost u najboljem slučaju} \\ &\quad\ (\text{best – case complexity}) \\ \text{w – Compl} &= \text{složenost u najgorem slučaju} \\ &\quad\ (\text{worst – case complexity}) \qquad . \end{aligned}$$

Pri tome je mnogo interesantnija složenost u najgorem slučaju.

Često je korisno znati i tzv. prosječnu složenost algoritma, u oznaci

$$\begin{aligned} \text{avg – Compl} &= \text{prosječna složenost} \\ &\quad\ (\text{average – case complexity}) \quad . \end{aligned}$$

Ova, naravno, ovisi o distribuciji ulazih podataka. Za njeno nalaženje, potrebno je pretpostaviti neki realan model ponašanja – distribucije podataka.

Napomena 1.2.3. *U analizi aritmetičke složenosti, promatrat ćemo samo aritmetičke operacije čiji rezultati ovise o ulaznim podacima.*

Nećemo računati promjene brojača u petljama, indeksiranje i adresiranje podataka, te kopiranje i pridjeljivanje vrijednosti (tj. spremanje podataka).

Sve to ćemo uključiti u vremensku složenost. Vodeći računa o različitom trajanju raznih operacija, vremensku složenost izražavamo samo redom veličine, bez faktora proporcionalnosti.

Uz ovaj dogovor, dobivamo sljedeće rezultate.

Propozicija 1.2.4.

Za aritmetičku složenost algoritma NADD1 *vrijedi :*

$$\begin{aligned} b - \mathrm{Compl}_A(\mathsf{NADD1}) &= 2\,\ell\,(v) \\ w - \mathrm{Compl}_A(\mathsf{NADD1}) &= 3\,\ell\,(v) + 1 \quad , \end{aligned} \tag{1.2.15}$$

a za vremensku :

$$\mathrm{Compl}_T(\mathsf{NADD1}) \sim \ell\,(u)\ . \tag{1.2.16}$$

Dokaz:

U najboljem slučaju su svi prijenosi jednaki 0, a u najgorem su svi jednaki 1. Zbog toga, u prvoj petlji obavljamo između $2\,\ell\,(v)$ i $3\,\ell\,(v)$ aritmetičkih operacija. Propozicija 1.2.3. pokazuje da drugi dio algoritma troši najviše jednu aritmetičku operaciju, kad nastupi drugi slučaj iz te propozicije.

Za vremensku složenost, rezultat je očit, jer moramo postaviti $\ell\,(w)$ znamenki rezultata, gdje je

$$\ell\,(u) \leq \ell\,(w) \leq \ell\,(u) + 1\ .$$

■

Napomena 1.2.4. *Algoritam* NADD1 *dozvoljava da rezultat w spremimo na mjesto ulaznog broja u ili v. Posebno je korisno spremanje w na mjesto broja u, pri zbrajanju niza brojeva u petlji. U tom slučaju otpada zadnja petlja u algoritmu, koja samo kopira znamenke. Tada je*

$$\mathrm{Compl}_S(\mathsf{NADD1}) = \ell\,(u) + \ell\,(v) + c$$

uz $c \leq 7$, a aritmetička i vremenska složenost ostaju kao u propoziciji 1.2.4. , iako je konstanta proporcionalnosti u (1.2.16) manja.

Napomena 1.2.5. *Ako eliminiramo zbrajanje prijenosa u slučaju kad znamo da je on jednak 0, prva petlja se može obaviti sa $\ell\,(v)$ zbrajanja znamenki ulaznih brojeva, pa je*

$$b - \mathrm{Compl}_A(\mathsf{NADD1}) = \ell\,(v)\ .$$

Za nalaženje prosječne složenosti, potrebno je naći prosječan broj pojava slučaja $carry = 1$, pri zbrajanju donjeg dijela broja u i broja v. Označimo taj broj pojava s K. Može se pokazati [Knuth, 1981. – str. 262–263] da je

$$K \approx \frac{1}{2}\,\ell\,(v)$$

uz pretpostavku da su znamenke brojeva uniformno distribuirane na $\{0, \ldots, b-1\}$. Tada je

$$\text{avg} - \text{Compl}_A(\mathsf{NADD1}) \approx \frac{5}{2}\,\ell\,(v) + \frac{1}{2}$$

(odnosno $2\,\ell\,(v) + 1/2$ u najboljoj varijanti), jer je vjerojatnost prijenosa 1 na mjestu $\deg(v)$ približno $1/2$. Prosječno vrijeme je proporcionalno s $\ell\,(u)$, osim u slučaju spremanja w na mjesto u, kad je proporcionalno s $\ell\,(v)$.

Pretpostavka $\deg(u) \geq \deg(v)$ iz algoritma NADD1, služi samo preglednosti algoritma i nije ozbiljno ograničenje, osim što narušava komutativnost zbrajanja.

Algoritam 1.2.4. (NADD – zbrajanje prirodnih brojeva)

Ulaz: Nizovi znamenki $(u_n, \ldots, u_0)$, $(v_m, \ldots, v_0)$ brojeva u i v u odabranoj bazi b.

Izlaz: Niz znamenki $(w_k, \ldots, w_0)$ broja

$$w = u + v$$

u bazi b.

procedure NADD (u , v : mpnat ; **var** w : mpnat);

begin
  if $\deg(u) \geq \deg(v)$ **then**
    NADD1 (u , v , w)
  else
    NADD1 (v , u , w)
end; { NADD }

Korektnost ovog algoritma je očita, a postignuta je i komutativnost zbrajanja.

Propozicija 1.2.5.

Za složenost algoritma NADD *vrijedi :*

$$\begin{aligned} b - \text{Compl}_A(\mathsf{NADD}) &= \min\{\,\ell\,(u)\ ,\ \ell\,(v)\,\} \\ w - \text{Compl}_A(\mathsf{NADD}) &= 3\,\min\{\,\ell\,(u)\ ,\ \ell\,(v)\,\} + 1 \end{aligned} \tag{1.2.17}$$

$$\text{Compl}_T(\mathsf{NADD}) \sim \max\{\,\ell\,(u)\ ,\ \ell\,(v)\,\}\ . \tag{1.2.18}$$

Vremenska složenost reda veličine $\min\{\,\ell\,(u)\ ,\ \ell\,(v)\,\}$ *dobiva se samo u slučaju prepisivanja rezultata na mjesto duljeg ulaznog broja.*

Dokaz:

Direktno iz propozicije 1.2.4. i napomene 1.2.5. . ■

Zbrajanje prirodnih brojeva možemo realizirati i na mnoge druge načine. Na pr. zbrajanje možemo početi od najviših znamenki, što je korisno kad paralelno učitavamo znamenke brojeva u uobičajenom redosljedu. Sve ove varijante imaju složenost sličnu onoj iz propozicije 1.2.5. .

1.3. Oduzimanje

Oduzimanje polinoma realiziramo uz iste pretpostavke kao i zbrajanje.

Algoritam 1.3.1. (PSUB1 – **oduzimanje polinoma)**

Ulaz: Nizovi koeficijenata $(u_0, \ldots, u_n)$ i $(v_0, \ldots, v_m)$ polinoma

$$U(x) = \sum_{i=0}^{n} u_i\, x^i \quad , \quad V(x) = \sum_{i=0}^{m} v_i\, x^i \quad ,$$

uz pretpostavku da je

$$n = \deg(U) \geq \deg(V) = m \ .$$

Izlaz: Niz koeficijenata $(w_0, \ldots, w_n)$ polinoma

$$W(x) = U(x) - V(x) \ ,$$

uz $k = \deg(W)$.

procedure PSUB1 (U , V : polinom ; **var** W : polinom);

begin
  $\deg(W) \leftarrow \deg(U)$;
  for $i \leftarrow 0$ **to** $\deg(V)$ **do**
    $w_i \leftarrow u_i - v_i$;
  if $(\deg(U) = \deg(V))$ **and** $(\deg(U) > 0)$ **then**
    begin
    while $(w_{\deg(W)} = 0)$ **and** $(\deg(W) > 0)$ **do**
      $\deg(W) \leftarrow \deg(W) - 1$;
    if $w_{\deg(W)} = 0$ **then**
      $\deg(W) \leftarrow -1$ { nul-polinom }
    end
  else
    for $i \leftarrow \deg(V) + 1$ **to** $\deg(U)$ **do**

$w_i \leftarrow u_i$
end; { PSUB1 }

Pretpostavka $n \geq m$ osigurava da je, u slučaju različitih duljina, polinom U dulji, pa njegove koeficijente treba kopirati u W, na kraju algoritma. U suprotnom, trebalo bi kopirati koeficijente $-v_i$ polinoma $-V$. Algoritam se lako proširuje do općeg algoritma PSUB za oduzimanje polinoma. Koeficijenti ne moraju biti nenegativni, jer uvijek postoji mogućnost da je W kraći od oba ulazna polinoma.

Do te pojave dolazi kod poništavanja vodećih koeficijenata ulaznih polinoma i to samo u slučaju kad su U i V iste duljine

$$\deg(W) < \deg(U) \implies \deg(U) = \deg(V) \ . \tag{1.3.1}$$

Korektnost algoritma je očita. Za prostornu složenost dobivamo

$$\mathrm{Compl}_S(\mathsf{PSUB1}) = 2\,\ell\,(U) + \ell\,(V) + 4 \ ,$$

jer računamo $\ell\,(V)$ koeficijenata polinoma W, čak i u slučaju da je $\deg(W) < \deg(U)$. Pažljivim uspoređivanjem stupnjeva i vodećih koeficijenata, to se može izbjeći i tada je

$$\mathrm{Compl}_S(\mathsf{PSUB1}) = \ell\,(U) + \ell\,(V) + \ell\,(W) + 4 \ .$$

Aritmetička složenost algoritma u ovom obliku je

$$\mathrm{Compl}_A(\mathsf{PSUB1}) = \ell\,(V) \ ,$$

ako ne računamo kopiranje vodećih koeficijenata iz U u W (za $n > m$), i smanjivanje $\deg(W)$. Uspoređivanjem vodećih koeficijenata, možemo postići

$$\mathrm{Compl}_A(\mathsf{PSUB1}) = \ell\,(W) \ .$$

Pošto moramo obraditi sve koeficijente duljeg polinoma, izlazi

$$\mathrm{Compl}_T(\mathsf{PSUB1}) \sim \ell\,(U) \ .$$

Kao kod zbrajanja, rezultat možemo spremiti na mjesto polinoma U, što analogno smanjuje i prostornu i vremensku složenost.

Za oduzimanje prirodnih brojeva, potrebna je jača pretpostavka

$$u \geq v \ ,$$

da rezultat $w = u - v$ bude također nenegativan. Algoritam ćemo realizirati tako da poziva proceduru ERROR, ukoliko otkrijemo da je $u < v$ na ulazu. Proceduru ERROR nećemo realizirati. Možemo smatrati da ona prekida izvršavanje programa.

I kod oduzimanja dolazi do pojave prijenosa, s tim da su prijenosi negativni.

Algoritam 1.3.2. (NSUB – **oduzimanje prirodnih brojeva**)

Ulaz: Nizovi znamenki $(u_n, \ldots, u_0)$, $(v_m, \ldots, v_0)$ brojeva u i v u odabranoj bazi b. Pretpostavljamo da je

$$u \geq v$$

što povlači i

$$n = \deg_b(u) \geq \deg_b(v) = m \ .$$

Izlaz: Niz znamenki $(w_k, \ldots, w_0)$ broja

$$w = u - v$$

u bazi b, sa $k = \deg_b(w)$.

procedure NSUB (u , v : mpnat ; **var** w : mpnat);

```
begin
    carry ← 0;
    for i ← 0 to deg(v) do
        begin
        w_i ← (u_i − v_i + carry) mod b;
        carry ← ⌊(u_i − v_i + carry)/b⌋
        end;
    for i ← deg(v) + 1 to deg(u) do
        begin
        w_i ← (u_i + carry) mod b;
        carry ← ⌊(u_i + carry)/b⌋
        end;
    if carry = 0 then
        begin
        deg(w) ← deg(u);
        if deg(w) ≥ 0 then
            begin
            while (w_deg(w) = 0) and (deg(w) > 0) do
                deg(w) ← deg(w) − 1;
            if w_deg(w) = 0 then
                deg(w) ← −1   { w = 0 }
            end
        end
    else
        ERROR   { rezultat negativan }
end;  { NSUB }
```

Bezuvjetno pretpostavljamo da je $n \geq m$ i samo tada kontroliramo $w \geq 0$. Na početku algoritma možemo dodati i kontrolu uvjeta $n \geq m$ i pozvati ERROR ako je $n < m$.

Ovaj algoritam je očita kombinacija algoritama NADD1 i PSUB1. Ovdje, međutim, može doći do skraćenja rezultata i kad ulazni brojevi nisu iste duljine, pa to treba uvijek provjeriti.

Propozicija 1.3.1.

Ako su brojevi u , $v \in \mathbb{N}_0$ *dani pozicionim zapisima u bazi* b

$$u = (u_n \dots u_0)_b \quad , \quad v = (v_m \dots v_0)_b \ ,$$

i ako je $u \geq v$, *onda algoritam* NSUB *daje niz znamenki pozicionog zapisa broja* $u - v$ *u bazi* b

$$u - v = w = (w_k \dots w_0)_b \ ,$$

gdje je $k = \deg_b(w) \in \{ -1, 0, \dots, n \}$.

Dokaz:

Potpuno analogan dokazu propozicije 1.2.1. za zbrajanje. U istim oznakama, rad algoritma je opisan rekurzivnim relacijama

$$\left.\begin{aligned} carry_{-1} &= 0 \\ w_i &= (u_i - v_i + carry_{i-1}) \bmod b \\ carry_i &= \lfloor (u_i - v_i + carry_{i-1})/b \rfloor \end{aligned}\right\} \quad i = 0, \dots, n \tag{1.3.2}$$

Za ostatak dokaza, dovoljno je u dokazu propozicije 1.2.1. , zamijeniti v_i sa $-v_i$, $i = 0, \dots, n$, jer se na isti način dobiva relacija (1.3.2) iz relacije (1.2.4).

Time dobivamo da algoritam u svakom koraku daje korektnu razliku donjih dijelova ulaznih brojeva

$$carry_i \cdot b^{i+1} + \sum_{j=0}^{i} w_j \, b^j = \sum_{j=0}^{i} u_j \, b^j - \sum_{j=0}^{i} v_j \, b^j \ , \tag{1.3.3}$$

za $i = 0, \dots, n$. Ovu relaciju možemo zapisati u obliku :

$$carry_i \cdot b^{i+1} + w \bmod b^{i+1} = u \bmod b^{i+1} - v \bmod b^{i+1} \ , \tag{1.3.4}$$

za $i = 0, \dots, n$.

Po analogiji s (1.2.7) dobivamo

$$carry_n \cdot b^{n+1} + w \bmod b^{n+1} = u - v \ . \tag{1.3.5}$$

Ovdje je

$$-1 \leq carry_i \leq 0 \ , \ i = -1, \ldots, n \tag{1.3.6}$$

jer je indukcijom :

$$-b \leq u_i - v_i + carry_{i-1} \leq b - 1 \tag{1.3.7}$$

za $i = 0, \ldots, n$.

Ako je $carry_n = -1$, onda zbog

$$w \bmod b^{n+1} < b^{n+1} \ ,$$

u (1.3.5) izlazi $u - v < 0$, pa algoritam korektno javlja grešku.

Za dobar rezultat nužno je $carry_n = 0$, pa (1.3.5) glasi

$$w \bmod b^{n+1} = u - v \ .$$

Kako algoritam postavlja $\deg(w)$ tako da je $\deg(w) \leq \deg(u) = n$, to po (1.1.12) slijedi $w = u - v$.

Ostatak algoritma samo korektno postavlja $\deg(w)$, tako da je $w_{\deg(w)} > 0$ za $w > 0$, odnosno $\deg(w) = -1$ za $w = 0$. ■

Ako je $b \leq maxint$, uočimo da, zbog relacije (1.3.7), algoritam radi korektno i u aritmetici računala.

Kao i kod zbrajanja, modificiramo algoritam tako da izbjegnemo cjelobrojno dijeljenje s ostatkom. Ovdje kvocijent (prijenos) može biti 0 ili -1, ovisno o tome da li je $u_i - v_i + carry \geq 0$ ili ne.

Analogno modificiramo i drugu petlju, korištenjem $v_i = 0$, $i = m+1, \ldots, n$. Prema relaciji (1.3.7), tada je

$$-1 \leq u_i + carry_{i-1} \leq b - 1 \ , \ i = m+1, \ldots, n \ . \tag{1.3.8}$$

Očito je, zbog $u_i \geq 0$, da je

$$carry_i = -1 \iff u_i + carry_{i-1} = -1 \ ,$$

što znači

$$carry_i = -1 \iff u_i = 0 \ \text{ i } \ carry_{i-1} = -1 \ . \tag{1.3.9}$$

Ako u relaciji (1.3.8), za bilo koji i , $i > m$, dobijemo $carry_{i-1} = 0$, onda algoritam daje

$$\begin{aligned} w_i &= u_i \\ carry_i &= 0 \ , \end{aligned}$$

tj. od tog mjesta nadalje, prijenos ostaje 0, i znamenke broja u se kopiraju u znamenke broja w.

Time je dokazana sljedeća tvrdnja.

Propozicija 1.3.2.

Ako u algoritmu NSUB *vrijedi*

$$n = \deg_b(u) > \deg_b(v) = m \ ,$$

onda je, u oznakama iz propozicije 1.3.1. , za svaki $i = m+1, \ldots, n$ *ispunjeno*

$$\begin{array}{llllll} carry_{i-1} = -1 & i & u_i = 0 & \Longleftrightarrow & w_i = b-1 & i \quad carry_i = -1 \\ carry_{i-1} = -1 & i & u_i > 0 & \Longleftrightarrow & w_i = u_i - 1 & i \quad carry_i = 0 \\ carry_{i-1} = 0 & & & \Longleftrightarrow & w_i = u_i & i \quad carry_i = 0 \end{array} \quad .$$

U svakom koraku nastupa točno jedan od tri slučaja na lijevoj strani. Drugi slučaj može nastupiti najviše jednom, kao prijelaz iz prvog u treći. ■

Ovim modifikacijama dobivamo novu verziju algoritma.

Algoritam 1.3.3. (NSUB – **oduzimanje prirodnih brojeva)**

Ulaz: Nizovi znamenki $(u_n, \ldots, u_0)$, $(v_m, \ldots, v_0)$ brojeva u i v u odabranoj bazi b. Pretpostavljamo da je

$$u \geq v \ .$$

U protivnom, algoritam završava pogreškom.

Izlaz: Niz znamenki $(w_k, \ldots, w_0)$ broja

$$w = u - v$$

u bazi b, sa $k = \deg_b(w)$.

```
procedure NSUB ( u , v : mpnat ; var w : mpnat);

begin
    if deg(u) < deg(v) then ERROR;
    carry ← 0;
    for i ← 0 to deg(v) do
        begin
        temp ← u_i − v_i + carry;
        if temp ≥ 0 then
            begin
            w_i ← temp;    carry ← 0
            end
        else
            begin
            w_i ← temp + b;    carry ← −1
            end;
        end; { for }
    if deg(u) > deg(v) then
        begin
        i ← deg(v) + 1;
        if carry = −1 then
            begin
            while (u_i = 0) and (i < deg(u)) do
                begin
                w_i ← b − 1;    i ← i + 1
                end;
            if u_i > 0 then
                begin
                w_i ← u_i − 1;    carry ← 0
                end
            else ERROR;  { i = deg(u)  i  carry = −1 }
            i ← i + 1;
            end; { if }
        for j ← i to deg(u) do w_j ← u_j;
        end;
    deg(w) ← deg(u);
    if deg(w) ≥ 0 then
        begin
        while (w_deg(w) = 0) and deg(w) > 0 do
            deg(w) ← deg(w) − 1;
        if w_deg(w) = 0 then deg(w) ← −1
        end
```

end; { NSUB }

Napomena 1.3.1. *Kao i kod zbrajanja, varijabla carry se može potpuno eliminirati. Osim toga, normalizaciona faza algoritma (postavljanje* $\deg(w)$*) može se izbjeći tako da pamtimo mjesto zadnje znamenke* $w_i > 0$ *, a znamenke* $w_{i+j} = 0$ *,* $j > 0$ *spremamo samo onda kad naiđemo na novu znamenku* $w_{i+j_0} > 0$ *,* $j_0 > 0$ *.*

Prostorna složenost ovog algoritma je

$$\mathrm{Compl}_S(\mathsf{NSUB}) = 2\,\ell\,(u) + \ell\,(v) + c\ ,$$

gdje je $c \leq 8$, kao u (1.2.14). Uz modifikaciju iz napomene 1.3.1. , može se postići

$$\mathrm{Compl}_S(\mathsf{NSUB}) = \ell\,(u) + \ell\,(v) + \ell\,(w) + c\ , \tag{1.3.10}$$

kao i kod oduzimanja polinoma.

Propozicija 1.3.3.

Za aritmetičku složenost algoritma NSUB *vrijedi :*

$$\begin{aligned} b\ -\ \mathrm{Compl}_A(\mathsf{NSUB}) &= 2\,\ell\,(v) \\ w\ -\ \mathrm{Compl}_A(\mathsf{NSUB}) &= 3\,\ell\,(v) + 1\ , \end{aligned} \tag{1.3.11}$$

a za vremensku :

$$\mathrm{Compl}_T(\mathsf{NSUB}) \sim \ell\,(u)\ . \tag{1.3.12}$$

Dokaz:

Za aritmetičku složenost rezultat je očit (kao u propoziciji 1.2.4.). Vremenska složenost je proporcionalna s $\ell\,(u)$, jer prolazimo sve znamenke broja u i za svaku od njih trošimo vrijeme ograđeno nekom konstantom. ■

I ovdje možemo rezultat w spremiti na mjesto broja u, što smanjuje prostornu složenost na $\ell\,(u) + \ell\,(v) + c$, dok aritmetička složenost ostaje ista, a vremenska je i dalje proporcionalna s $\ell\,(u)$, s manjom konstantom proporcionalnosti.

Napomena 1.3.2. *Ako eliminiramo zbrajanje prijenosa, kad znamo da je on jednak* 0*, dobivamo*

$$b\ -\ \mathrm{Compl}_A(\mathsf{NSUB}) = \ell\,(v)\ ,$$

kao za zbrajanje.

Uz pretpostavku da su znamenke brojeva uniformno distribuirane na skupu $\{\,0, \ldots, b-1\,\}$, prosječan broj K pojava slučaja $carry_i = -1$ pri oduzimanju donjeg dijela broja u i broja v, je

$$K \approx \frac{1}{2}\,\ell\,(v)\ .$$

Za prosječnu aritmetičku složenost dobivamo

$$\text{avg} - \text{Compl}_A(\mathsf{NSUB}) \approx \frac{5}{2}\,\ell\,(v) + \frac{1}{2}\ ,$$

odnosno $2\,\ell\,(v)+1/2$, u varijanti iz napomene 1.3.2. , kao i kod zbrajanja. Prosječno vrijeme je također proporcionalno s $\ell\,(u)$, osim u slučaju kad spremamo w na mjesto broja u. Tada je prosječno vrijeme proporcionalno s $\ell\,(v)$. Razlog tome je mala vjerojatnost ($\sim 1/b$) skraćenja rezultata.

Zanimljivo je pitanje, možemo li postići da aritmetička složenost oduzimanja bude proporcionalna s $\ell\,(w)$, za $\ell\,(w) < \ell\,(v)$, kao kod oduzimanja polinoma. To bi značilo da duljinu broja w treba naći uspoređivanjem znamenki brojeva u i v, uz najviše konstantan broj aritmetičkih operacija. Algoritam NSUB je tada lako modificirati tako da računa samo prvih $\ell\,(w)$ znamenki broja w sa $\Theta(\ell\,(w))$ aritmetičkih operacija.

Napomena 1.3.3. *Duljinu rezultata je lako odrediti u aritmetici polinoma, jer vrijedi relacija (1.3.1) i koeficijent w_i ovisi samo o u_i i v_i. U aritmetici brojeva, znamenka w_i, zbog (1.3.2), ovisi o svim prethodnim znamenkama u_j , v_j ,* $j = 0, \ldots, i$.

Za odgovor na ranije pitanje, potrebno je znati kada dolazi do skraćenja razlike $w = u - v$ obzirom na broj u. To znači da je

$$\deg(w) < \deg(u)\ .$$

U algoritmu NSUB i relaciji (1.3.2), tada je

$$w_i = 0\ \ ,\ \ i = \deg(w)+1, \ldots, \deg(u)\ . \tag{1.3.13}$$

Pogledajmo prvo, kada algoritam NSUB daje $w_i = 0$.

Propozicija 1.3.4.

U algoritmu NSUB*, odnosno relaciji (1.3.2), vrijedi :*

(a) *ako je* $i > m = \deg_b(v)$ *, onda je*

$$w_i = 0 \iff \left\{ \begin{array}{ll} & u_i = 0 \quad i \quad carry_{i-1} = 0 \\ ili & \\ & u_i = 1 \quad i \quad carry_{i-1} = -1 \end{array} \right. \tag{1.3.14}$$

$i\ \ w_i = 0 \implies carry_i = 0$.

(b) *ako je* $i \le m$ *, onda je*

$$w_i = 0\ \ i\ \ carry_i = 0 \iff \left\{ \begin{array}{ll} & u_i = v_i \quad\ \ i \quad carry_{i-1} = 0 \\ ili & \\ & u_i = v_i + 1 \quad i \quad carry_{i-1} = -1 \end{array} \right. \tag{1.3.15}$$

$$w_i = 0\ \ i\ \ carry_i = -1 \iff u_i = 0\ \ ,\ \ v_i = b-1\ \ i\ \ carry_{i-1} = -1\ . \tag{1.3.16}$$

Dokaz:

ad (a) : direktno iz propozicije 1.3.2. , jer, zbog $b \geq 2$, $w_i = 0$ može nastupiti samo u drugom i trećem slučaju iz te propozicije.

ad (b) : direktno iz relacije (1.3.2). ■

Uočimo da su ovdje popisane sve mogućnosti da je $w_i = 0$. Uz oznaku $v_i = 0$ za $i > m$, slučaj (a) se može zapisati kao prva mogućnost u (b).

Analizirajmo postupak za nalaženje stupnja $\deg(w)$ razlike $w = u - v$ prirodnih brojeva u i v. Pri tome $\deg(w)$ treba naći direktno iz znamenki brojeva u i v **bez računanja** broja w. Algoritam NSUB nalazi, naravno, i $\deg(w)$, no upravo taj algoritam želimo ubrzati.

Propozicija 1.3.4. daje kriterije za provjeru $w_i = 0$, na bazi znamenki u_i i v_i, pod uvjetom da znamo vrijednosti $carry_{i-1}$. Tada možemo provjeravati $w_i = 0$, **uzlazno** po i. Relacija (1.3.13) pokazuje da nalaženje $\deg(w)$ treba početi od najviših znamenki brojeva u i v, tj. treba provjeravati $w_i = 0$ **silazno** po i, sve dok ne ustanovimo $w_i \neq 0$.

Pretpostavimo da je $u \geq v$ (algoritam za uspoređivanje brojeva ćemo realizirati nešto kasnije). Tada znamo da algoritam NSUB završava s

$$carry_{\deg(u)} = 0 \ .$$

To omogućava primjenu rezultata propozicije 1.3.4. silazno po i, pod uvjetom da možemo provjeriti vrijednost $carry_{i-1}$ u svakom koraku.

Stoga prvo razrađujemo algoritam za nalaženje vrijednosti prijenosa $carry_i$ direktno uspoređivanjem znamenki brojeva u i v. Algoritam je baziran na sljedećoj tvrdnji.

Propozicija 1.3.5.

U algoritmu NSUB*, odnosno relaciji (1.3.2), za prijenose vrijedi :*

(a) *ako je* $i > m = \deg_b(v)$ *, onda je :*

$$\begin{aligned} u_i \geq 1 &\implies carry_i = 0 \\ u_i = 0 &\implies carry_i = carry_{i-1} \end{aligned} \tag{1.3.17}$$

(b) *ako je* $i \leq m$ *, onda je :*

$$\begin{aligned} u_i \geq v_i + 1 &\implies carry_i = 0 \\ u_i = v_i &\implies carry_i = carry_{i-1} \\ u_i < v_i &\implies carry_i = -1 \ . \end{aligned} \tag{1.3.18}$$

Dokaz:

Uz oznaku $v_i = 0$ za $i > m$, slučaj (a) se može zapisati kao prve dvije relacije iz (1.3.18). Zbog $u_i \geq 0$, treća relacija iz (1.3.18) vrijedi trivijalno za $i > m$.

Iz relacije (1.3.6) znamo da je $carry_i \in \{ -1 , 0 \}$, za $i = -1, \ldots, n$. Zbog toga

$$\begin{aligned} u_i \geq v_i + 1 \quad &\Longrightarrow \quad u_i - v_i + carry_{i-1} \geq 0 \\ u_i = v_i \quad &\Longrightarrow \quad u_i - v_i + carry_{i-1} = carry_{i-1} \\ u_i < v_i \quad &\Longrightarrow \quad u_i - v_i + carry_{i-1} < 0 \ . \end{aligned}$$

Kako je $carry_i = \lfloor (u_i - v_i + carry_{i-1})/b \rfloor$, dobivamo relacije (1.3.18). ■

Lijeve strane relacija (1.3.17) i (1.3.18) popisuju sve mogućnosti odnosa znamenki u_i i v_i. To omogućava direktno računanje $carry_i$, za $u_i \neq v_i$, odnosno iterativno računanje (silazno po i) za $u_i = v_i$.

Algoritam 1.3.4. (NSUBC – **prijenos kod oduzimanja)**

Ulaz: – Nizovi znamenki $(u_n, \ldots, u_0)$, $(v_m, \ldots, v_0)$ brojeva u i v u odabranoj bazi b, uz pretpostavku da je $u \geq v$;

– Indeks $i \in \{ -1, \ldots, \deg(u) \}$.

Izlaz: Vrijednost $carry$,

$$carry = carry_i \in \{ -1, 0 \} \ ,$$

prijenosa na mjestu i, prilikom oduzimanja brojeva u i v.

```
procedure NSUBC ( u , v : mpnat ;  i : integer ;  var  carry : integer);

begin
   carry ← 0;
   if  (i ≥ 0)  and  (i ≤ deg(u))  and  (deg(v) ≥ 0)  then
      begin
      while  (u_i = 0)  and  (i > deg(v))  do  i ← i − 1;
      if  i ≤ deg(v)  then
         begin
         while  (u_i = v_i)  and  (i > 0)  do  i ← i − 1;
         if  u_i < v_i  then  carry ← −1
         end
      end
end;  { NSUBC }
```

Korektnost algoritma je očita iz propozicije 1.3.5. i $carry_{-1} = 0$.

Ovaj algoritam ne zahtijeva niti jednu aritmetičku operaciju, u smislu napomene 1.2.3. , jer koristi samo modifikacije indeksa (adresa), uspoređivanje i spremanje vrijednosti.

Za vremensku složenost je očito

$$\mathrm{w} - \mathrm{Compl}_T(\mathsf{NSUBC}) \sim i \ , \tag{1.3.19}$$

jer algoritam može proći sve znamenke $u_i, \ldots, u_0$ da nađe $carry_i$. Prosječna vremenska složenost je konstantna, pošto slučajevi $u_i = 0$, odnosno $u_i = v_i$ imaju malu vjerojatnost ($\sim 1/b$), a u svim ostalim slučajevima, algoritam daje odgovor samo na osnovi znamenke u_i.

Problem nalaženja $\deg(w)$ je ovim, u principu, riješen. Pažljivom primjenom propozicije 1.3.5. možemo, međutim, izbjeći nalaženje $carry_{i-1}$ za gotovo sve $i = \deg(u), \ldots, \deg(w) + 1$, što znatno skraćuje algoritam. Za slučajeve iz propozicije 1.3.4. , relacija (1.3.18) daje :

$$\begin{aligned} u_i = v_i \quad &\Longrightarrow \quad carry_i = carry_{i-1} \\ u_i = v_i + 1 \quad &\Longrightarrow \quad carry_i = 0 \\ u_i = 0 \ \text{ i } \ v_i = b - 1 \quad &\Longrightarrow \quad carry_i = -1 \ . \end{aligned} \tag{1.3.20}$$

Analizirajmo sve slučajeve u kojima dolazi do kraćenja $\deg(w) < \deg(u)$, prilikom oduzimanja, uz pretpostavku $u \geq v$.

Označimo, kao u algoritmima,

$$n = \deg_b(u) \ \ , \ \ m = \deg_b(v) \ \ , \ \ k = \deg_b(w) \ .$$

Kraćenje i pretpostavka $u \geq v$ daju

$$w_n = 0 \ \ , \ \ carry_n = 0$$

u relaciji (1.3.2) i algoritmu NSUB.

Po propoziciji 1.3.4. , moguća su tri slučaja :

A $n = m$ i $u_n = v_n$,

B $n = m$ i $u_n = v_n + 1$,

C $n > m$ i $u_n = 1$,

zbog $u_n > 0$.

Ova tri slučaja su redom opisana sljedećim tvrdnjama.

Propozicija 1.3.6. (slučaj A)

Neka su u , $v \in \mathbb{N}$, $u \geq v$, *prirodni brojevi dani pozicionim zapisima u bazi* b

$$u = (u_n \dots u_0)_b \quad , \quad v = (v_m \dots v_0)_b$$

i neka je $n = m$. *Definiramo*

$$d = \max \{ \, i \;\; | \;\; u_i \neq v_i \;\; , \;\; 0 \leq i \leq n \, \} \;\; , \tag{1.3.21}$$

uz dogovor $d = -1$, *ako je* $u = v$.

Za broj $w = u - v$ *tada vrijedi :*

$$\deg_b(w) \leq d \; , \tag{1.3.22}$$

$$\deg_b(w) < d \iff u_d = v_d + 1 \;\; i \;\; carry_{d-1} = -1 \; , \tag{1.3.23}$$

$$w = u \bmod b^{d+1} - v \bmod b^{d+1} \; . \tag{1.3.24}$$

Dokaz:

Tvrdnja je trivijalna za $d = n$ i $d = -1$. Pretpostavimo stoga $0 \leq d < n$. Zbog $u \geq v$, vrijedi $carry_n = 0$. Kako je iz (1.3.21)

$$u_i = v_i \;\; , \;\; i = d+1, \dots, n \; , \tag{1.3.25}$$

propozicija 1.3.5. daje

$$carry_i = carry_{i-1} \;\; , \;\; i = d+1, \dots, n \; ,$$

što, sa $carry_n = 0$, povlači

$$carry_i = 0 \;\; , \;\; i = d, \dots, n \; . \tag{1.3.26}$$

Primjenom propozicije 1.3.4. , iz realcije (1.3.16) dobivamo

$$w_i = 0 \;\; , \;\; i = d+1, \dots, n \; ,$$

što dokazuje $\deg_b(w) \leq d$.

Kako je $carry_d = 0$, iz (1.3.16) slijedi

$$w_{d-1} = 0 \iff u_d = v_d + 1 \;\; \text{i} \;\; carry_{d-1} = -1 \; .$$

Time je i relacija (1.3.23) dokazana.

Relacija (1.3.25) daje

$$\lfloor u / b^{d+1} \rfloor = \lfloor v / b^{d+1} \rfloor \; .$$

Tada je, prema (1.1.10)

$$u = \lfloor u/b^{d+1} \rfloor \cdot b^{d+1} + u \bmod b^{d+1}$$
$$v = \lfloor v/b^{d+1} \rfloor \cdot b^{d+1} + v \bmod b^{d+1}$$

pa je, oduzimanjem

$$w = u - v = u \bmod b^{d+1} - v \bmod b^{d+1} \ .$$

■

Napomena 1.3.4. *Ako je* $d = -1$ *, tj.* $u = v$ *, onda uspoređivanjem znamenki dobivamo* $\deg_b(w) = -1$ *. Ako je* $d \geq 0$ *, po definiciji broja d slijedi* $u_d \neq v_d$ *. Kako je* $u \geq v$ *, to je nužno*

$$u_d > v_d \ .$$

Ako je i $u_d > v_d + 1$ *, onda zaključujemo da je* $\deg_b = d$ *.*

Relacija (1.3.23) pokazuje da je daljnje kraćenje moguće samo ako je $u_d = v_d + 1$ *. Definiramo li brojeve* u' *,* v' *sa*

$$u' = u \bmod b^{d+1} = (u_d \dots u_0)$$
$$v' = v \bmod b^{d+1} = (v_d \dots v_0)$$

(dozvoljeno je i $v_d = 0$ *), onda je*

$$w = u' - v'$$

sa $\deg_b(u') = d$ *i* $carry_d = 0$ *. Stupanj broja w određujemo iz brojeva* u' *i* v' *pri čemu su vodeće znamenke različite. Time je dalje kraćenje u slučaju A, svedeno na slučaj B (ili C).*

Propozicija 1.3.7. (slučaj B)

Neka su u *,* $v \in \mathbb{N}$ *,* $u \geq v$ *, prirodni brojevi dani pozicionim zapisima u bazi b*

$$u = (u_n \dots u_0)_b \quad , \quad v = (v_m \dots v_0)_b$$

i neka je $n = m$ *i* $u_n > v_n$ *. Za broj* $w = u - v$ *vrijedi :*

(a) *Ako je* $u_n > v_n + 1$ *, onda je*

$$\deg_b(w) = n \ .$$

(b) *Ako je* $u_n = v_n + 1$ *, definiramo*

$$c = \begin{cases} n \quad , \quad ako\ je \quad n = 0 \quad ili \quad n > 0 \quad i \quad (u_{n-1} \neq 0 \ ili \ v_{n-1} \neq b-1) \\ \\ \min \{\ j \geq 0 \ \mid \ u_i = 0 \quad i \quad v_i = b-1 \quad , \quad \forall i = j, \dots, n-1 \ \} \quad , \ inače \end{cases} \tag{1.3.27}$$

tj. c je najniže mjesto počev od kog vrijedi $u_i = 0$, $v_i = b-1$, $i = c, \ldots, n-1$.

Tada je :

$$c - 1 \leq \deg_b(w) \leq c \tag{1.3.28}$$

i

$$\begin{aligned} \deg_b(w) = c &\iff carry_{c-1} = 0 \\ \deg_b(w) = c-1 &\iff carry_{c-1} = -1 \end{aligned} \tag{1.3.29}$$

(tj. $\deg_b(w) = c + carry_{c-1}$ *).*

Dokaz:

ad (a) : Očito, jer je tada $w_n > 0$.

ad (b) : Po pretpostavci je $u_n = v_n + 1$ i vrijedi $carry_n = 0$, zbog $u > v$. Iz relacije (1.3.16) dobivamo

$$w_n = 0 \iff carry_{n-1} = -1 \ . \tag{1.3.30}$$

Ako je $c = n = 0$, onda je $carry_{-1} = 0$, pa je $w_0 = 1$ i $\deg_b(w) = 0 = c$, što dokazuje (1.3.29).

Ako je $c = n > 0$, onda je iz (1.3.30)

$$w_n > 0 \iff carry_{n-1} = 0 \ ,$$

što dokazuje prvu relaciju (1.3.29). Pretpostavimo stoga da je $w_n = 0$, tj. $carry_{n-1} = -1$. Da je i $w_{n-1} = 0$, zbog $carry_{n-1} = -1$, iz (1.3.16) izlazi $u_{n-1} = 0$ i $v_{n-1} = b-1$, tj. $c < n$, što je suprotno pretpostavci $c = n$.

Preostaje analizirati slučaj $c < n$. Tada je, po definiciji c :

$$u_i = 0 \ , \ v_i = b-1 \ , \text{ za } \ i = c, \ldots, n-1 \ . \tag{1.3.31}$$

Relacija (1.3.20) tada daje

$$carry_i = -1 \ , \ i = c, \ldots, n-1 \ , \tag{1.3.32}$$

tj. vrijedi

$$u_i = 0 \ , \ v_i = b-1 \ , \ carry_{i-1} = -1 \ , \ i = c+1, \ldots, n-1 \ .$$

Iz relacije (1.3.16) dobivamo

$$w_i = 0 \ , \quad i = c+1, \ldots, n-1 \ ,$$

a zbog $carry_{n-1} = -1$ je i $w_n = 0$ iz (1.3.30). Time je dokazano

$$\deg_b(w) \leq c \ . \tag{1.3.33}$$

Kako je i $carry_c = -1$, te $u_c = 0$, $v_c = b-1$, relacija (1.3.16) daje

$$w_c = 0 \iff carry_{c-1} = -1 \ , \tag{1.3.34}$$

što dokazuje prvu relaciju (1.3.29).

Zbog minimalnosti c, vrijedi

$$c = 0$$

ili

$$u_c > 0 \ \text{ ili } \ v_c < b-1 \ \text{ , za } \ c > 0 \ .$$

Ako je $c = 0$, onda je $carry_{-1} = 0$, pa (1.3.34) daje $w_0 > 0$ ($w_0 = 1$) i $\deg_b(w) = 0 = c$. Za $c > 0$, ako je $w_c = 0$, onda je $w_{c-1} > 0$ iz (1.3.16), istom argumentacijom kao u slučaju $c = n > 0$. ■

Propozicija 1.3.8. (slučaj C)

Neka su u , $v \in \mathbb{N}$, $u \geq v$, *prirodni brojevi dani pozicionim zapisima u bazi* b

$$u = (u_n \dots u_0)_b \ , \ v = (v_m \dots v_0)_b$$

i neka je $n > m$. *Za broj* $w = u - v$ *vrijedi :*

(a) *Ako je* $u_n > 1$, *onda je*

$$\deg_b(w) = n \ .$$

(b) *Ako je* $u_n = 1$, *definiramo*

$$c = \begin{cases} n \ , \ \textit{ako je} \ \ n > m+1 \ \ \textit{ili} \ \ n = m+1 \ \ i \ \ (u_{n-1} \neq 0 \ \textit{ili} \ v_{n-1} \neq b-1) \\ \\ \min \{ \ j \geq 0 \ \mid \ u_i = 0 \ \ i \ \ v_i = b-1 \ \ , \ \ \forall i = j, \dots, n-1 \ \} \ \ , \ \textit{inače} \end{cases} \tag{1.3.35}$$

Tada je :

$$c - 1 \leq \deg_b(w) \leq c \tag{1.3.36}$$

i

$$\begin{aligned} \deg_b(w) = c \quad &\iff \quad carry_{c-1} = 0 \\ \deg_b(w) = c-1 \quad &\iff \quad carry_{c-1} = -1 \end{aligned} \tag{1.3.37}$$

(tj. $\deg_b(w) = c + carry_{c-1}$ *).*

Dokaz:

ad (a) : Očito, jer je $w_n > 0$.

ad (b) : Zbog $u_n = 1$ i $carry_n = 0$, iz (1.3.14) izlazi

$$w_n = 0 \iff carry_{n-1} = -1 \ . \tag{1.3.38}$$

Ako je $c = n$, onda je

$$w_n > 0 \iff carry_{n-1} = 0 \ ,$$

što dokazuje prvu relaciju (1.3.37). Pretpostavimo da je $w_n = 0$. Ako je $n > m+1$, onda je nužno $w_{n-1} > 0$, jer $w_{n-1} = 0$ daje $carry_{n-1} = 0$, po (1.3.14) što je kontradikcija s (1.3.38).

Ako je $n = m+1$, onda je $w_{n-1} = 0$ sa $carry_{n-1} = -1$, po relaciji (1.3.16), daje $u_{n-1} = 0$, $v_{n-1} = b-1$, što je kontradikcija s $c = n$.

Dakle, ako je $c = n$ i $w_n = 0$, onda je nužno $w_{n-1} > 0$. Time je tvrdnja dokazana za slučaj $c = n$.

Ako je $c < n$, onda je $n = m+1$ i dokaz tvrdnje je isti kao u propoziciji 1.3.7. . ■

Uočimo da su slučajevi B i C identični, uz oznaku $v_i = 0$, $i = m+1, \ldots, n$.

Napomena 1.3.5. *Ako su vodeće znamenke brojeva u i v jednake (slučaj A), onda je potrebno provjeriti znamenke počev od mjesta d na kojem je*

$$u_d > v_d \ .$$

Primijenimo li propoziciju 1.3.7. na brojeve $u \bmod b^{d+1}$, $v \bmod b^{d+1}$, *sa* $n = d$, *dobivamo potpuni kriterij nalaženja* $\deg_b(w)$ *i za slučaj A.*

Ovi rezultati pokazuju da je $\deg_b(w)$ moguće naći s pogreškom najviše jednakom 1, koristeći samo uspoređivanje znamenki brojeva u i v. Za nalaženje točne vrijednosti potrebna je najviše jedna provjera prijenosa.

Time dobivamo sljedeća dva algoritma.

Algoritam 1.3.5. (NSUBD – **stupanj razlike**)

Ulaz: Nizovi znamenki $(u_n, \ldots, u_0)$, $(v_m, \ldots, v_0)$ brojeva u, $v \in \mathbb{N}_0$ u odabranoj bazi b, uz pretpostavku $u \geq v$.

Izlaz: Broj k – stupanj razlike $w = u - v$

$$k = \deg_b(w) \ .$$

procedure NSUBD (u , v : mpnat ; **var** k : integer);

```
begin
    k = deg(u);
    if  k ≥ 0  then
        begin
        if  deg(u) = deg(v)  then
            begin  { u i v iste duljine i pozitivni }
            while  (u_k = v_k)  and  (k > 0)  do  k ← k − 1;
            if  u_k = v_k  then    { k = 0 }
                begin
                k = −1;
                check ← false
                end
            else
                check ← (u_k = v_k + 1)
            end
        else
            check ← (u_k = 1)  and  (deg(v) ≥ 0)
        if  check  and  (k > 0)  then
            begin
            repeat
                k ← k − 1
            until  (u_k > 0)  or  (v_k < b − 1)  or  (k = 0);
            if  (u_k > 0)  or  (v_k < b − 1)  then
                begin
                k ← k + 1;
                NSUBC ( u , v , k , carry);
                k ← k + carry
                end
            end
        end
end;  { NSUBD }
```

Propozicija 1.3.9.

Algoritam NSUBD *korektno nalazi*

$$k = \deg_b(w) \tag{1.3.39}$$

za bilo koje u , $u \in \mathbb{N}_0$, $u \geq v$.

Dokaz:

Za $u = 0$, algoritam samo postavlja

$$k = \deg_b(u) = -1 \ ,$$

što je korektno, jer je tada i $v = 0$, zbog $u \geq v$ i $v \in \mathbb{N}_0$.

Pretpostavimo da je $u > 0$.

Ako je $\deg(u) = \deg(v)$, onda prvi dio algoritma provjerava slučaj A i postavlja $k = d$, gdje je d definiran relacijom (1.3.21). Ako je $d = -1$, onda je $u = v$ i $w = 0$. Algoritam zbog $check = false$ ne mijenja k, pa vrijedi (1.3.39). Za $k = d \geq 0$, algoritam mijenja k samo ako je $u_d = v_d + 1$ $(check = true)$, što odgovara rezultatima propozicije 1.3.6. .

Ako je $\deg(u) > \deg(v)$, do skraćenja može doći samo ako je $v > 0$ i $u_n = 1$, tj. kada je $check = true$. U protivnom, algoritam daje $k = \deg(u)$, pa vrijedi (1.3.39).

Drugi dio algoritma provjerava slučajeve B i C, s tim da je $u_k = v_k + 1$. Tada repeat petlja postavlja $k = c - 1$, ako je $c > 0$ i $k = c$ za $c = 0$. Ako je $c = 0$, onda je $\deg_b(w) = c$, jer je $carry_{-1} = 0$. Algoritam tada radi korektno, jer ne mijenja k.

Za $c > 0$, algoritam obavlja posljednju if naredbu i korigira k tako da je

$$k = c + carry_{c-1} \ .$$

Relacije (1.3.29) i (1.3.37) pokazuju korektnost algoritma. ■

Za aritmetičku složenost ovog algoritma vrijedi

$$\mathrm{Compl}_A(\mathsf{NSUBD}) \leq 1 \ , \tag{1.3.40}$$

jer je jedina aritmetička operacija, u smislu napomene 1.2.3. , računanje $v_k + 1$.

Algoritam prolazi redom znamenke

$$u_k \ , \ k = \deg(u), \ldots, \deg(w) \ ,$$

a provjera zadnjeg prijenosa može proći i sve preostale znamenke. Zbog (1.3.19) je

$$\mathrm{w} - \mathrm{Compl}_T(\mathsf{NSUBD}) \sim \ell\,(u) \ , \tag{1.3.41}$$

dok je prosječna vremenska složenost kostantna, kao i za NSUBC.

Ako izbacimo provjeru zadnjeg prijenosa, onda je

$$0 \leq k - \deg_b(w) \leq 1 \ .$$

Tako dobivamo algoritam za približno nalaženje stupnja broja $w = u - v$.

Algoritam 1.3.6. (NSUBD1 – **približni stupanj razlike**)

Ulaz: Nizovi znamenki $(u_n, \ldots, u_0)$, $(v_m, \ldots, v_0)$ brojeva u, $v \in \mathbb{N}_0$ u odabranoj bazi b, uz pretpostavku $u \geq v$.

Izlaz: Broj ka – približni stupanj razlike $w = u - v$ takav da je

$$0 \leq ka - \deg_b(w) \leq 1 \ . \tag{1.3.42}$$

procedure NSUBD1 (u , v : mpnat ; **var** ka : integer);

```
begin
    ka = deg(u);
    if ka ≥ 0 then
        begin
        if deg(u) = deg(v) then
            begin
            while (u_ka = v_ka) and (ka > 0) do ka ← ka − 1;
            check ← (u_ka = v_ka + 1);
            end
        else
            check ← (deg(u) = deg(v) + 1) and (u_ka = 1);
        if check and (ka > 0) then
            begin
            repeat
                ka ← ka − 1
            until (u_ka > 0) or (v_ka < b − 1) or (ka = 0);
            ka ← ka + 1;
            end
        end
end; { NSUBD1 }
```

Zbog kratkoće algoritma, vrijednost ka postavljamo približno i u nekim slučajevima kad je moguće naći egzaktnu duljinu, bez provjere prijenosa. Takvi slučajevi su $d = 0$ (znamo $k = -1$, a postavljamo $ka = 0$) i $c = 0$ (znamo $k = 0$, a postavljamo $ka = 1$).

Uz ove modifikacije, korektnost algortima je očita.

Aritmetička složenost je kao u (1.3.40), ali za vremensku složenost dobivamo

$$\mathrm{w-\ Compl}_T(\mathsf{NSUBD1}) \sim \ell\,(u) - \ell\,(w) \tag{1.3.43}$$

što je poboljšanje u odnosu na relaciju (1.3.41) za algoritam NSUBD.

Poznavanje duljine rezultata $w = u - v$, omogućava nam da oduzimanje obavimo za $\Theta(\ell\,(w))$ aritmetičkih operacija, kad je $\ell\,(w) < \ell\,(v)$.

Lako je modificirati algoritam NSUB tako da računa samo znamenke

$$w_i \ , \ i = 0, \ldots, k \ \ (\text{ili } ka) \ ,$$

gdje je $k = \deg(w)$ i ka približna vrijednost za $\deg(w)$ iz (1.3.42). Pošto je ka najviše za 1 veći od pravog stupnja, slučaj $w_{ka} = 0$ se lako otkriva i bez normalizacione faze. Time normalizaciona faza algoritma NSUB postaje nepotrebna.

Označimo sa NSUBM bilo koju varijantu tako modificiranog algoritma za oduzimanje brojeva $u \ , v \in \mathbb{N}_0$ u pozicionom zapisu, uz uvjet da se računaju samo znamenke $w_i \ , \ i = 0, \ldots, k \ (ka)$ rezultata.

Napomena 1.3.6. *Algoritam* NSUBM *može se realizirati u nastavku algoritma* NSUBD *ili* NSUBD1*, tako da računa znamenke w_i silazno po i, koristeći algoritam* NSUBC *za provjeru prijenosa.*

Važnost ovakve modifikacije opravdava sljedeći rezultat o složenosti.

Propozicija 1.3.10.

Za aritmetičku složenost algoritma NSUBM *može se postići*

$$\begin{aligned} b - \text{Compl}_A(\text{NSUBM}) &= \min \{ \ell(v) \ , \ell(w) \} \\ w - \text{Compl}_A(\text{NSUBM}) &= 3 \min \{ \ell(v) \ , \ell(w) \} + 2 \end{aligned} \tag{1.3.44}$$

dok za vremensku vrijedi

$$\text{Compl}_T(\text{NSUBM}) \sim \ell(u) \ . \tag{1.3.45}$$

Dokaz:

Ako je $\ell(v) \leq \ell(w)$, rezultat slijedi direktno iz propozicije 1.3.3. i napomene 1.3.2. , jer se postupak računanja pojedinačnih znamenki broja w ne mijenja.

Ako je $\ell(w) < \ell(v)$, onda je dovoljno naći $\ell(w)$ znamenki $w_0, \ldots, w_k$ rezultata, po istom postupku, a nalaženje $\deg(w)$ traži najviše jednu aritmetičku operaciju. ■

To pokazuje da je broj potrebnih aritmetičkih operacija proporcionalan **najkraćem** od brojeva u , v , w. Isto vrijedi i kod zbrajanja, jer je tamo $\ell(w) \geq \ell(u)$.

Teorem 1.3.1.

Neka su $u \ , v \in \mathbb{N}_0$ brojevi dani pozicionim zapisima u bazi b. Zbrajanje i oduzimanje brojeva u i v zahtijeva općenito barem $\ell(u+v)$ odnosno $\ell(u-v)$ aritmetičkih operacija :

$$\begin{aligned} \text{Compl}_A(u+v) &= \Omega(\ell(u+v)) \\ \text{Compl}_A(u-v) &= \Omega(\ell(u-v)) \quad . \end{aligned} \tag{1.3.46}$$

Dokaz:

Očit, jer općenito treba izračunati sve znamenke rezultata. ■

Propozicija 1.2.5. i propozicija 1.3.10. pokazuju da su izloženi algoritmi za zbrajanje i oduzimanje **optimalni** u broju aritmetičkih operacija, do na mali konstantni faktor, jer je

$$\begin{aligned} \mathrm{Compl}_A(\mathsf{NADD}) &= \mathcal{O}(\ell\,(u+v)) \\ \mathrm{Compl}_A(\mathsf{NSUB}) &= \mathcal{O}(\ell\,(u-v)) \quad . \end{aligned}$$

Kod oduzimanja, za $\ell\,(v) < \ell\,(w)$, može se još smanjiti broj operacija.

To znači da se Ω u relaciji (1.3.46) može zamijeniti sa Θ .

Vremenska složenost je proporcionalna s $\ell\,(u)$ na sekvencijalnim arhitekturama.

Korolar 1.3.1.

Zbrajanje i oduzimanje n – znamenkastih prirodnih brojeva u pozicionom zapisu u bazi b, moguće je obaviti sa

$$\mathcal{O}(n)$$

aritmetičkih operacija i to je općenito optimalno.

Kod oduzimanja smo uvijek pretpostavljali da je $u \geq v$. Provjeru tog uvjeta možemo obaviti sljedećim algoritmom.

Algoritam 1.3.7. (NCOMP – **uspoređivanje prirodnih brojeva)**

Ulaz: Nizovi znamenki $(u_n, \ldots, u_0)$, $(v_m, \ldots, v_0)$ brojeva u, $v \in \mathbb{N}_0$ u odabranoj bazi b.

Izlaz: Broj $sign$ definiran sa

$$sign = \mathrm{sign}(u - v)\ . \tag{1.3.47}$$

procedure NCOMP (u , v : mpnat ; **var** $sign$: integer);

begin
  if $\deg(u) > \deg(v)$ **then**
    $sign \leftarrow 1$
  else if $\deg(u) < \deg(v)$ **then**
    $sign \leftarrow -1$
  else if $\deg(u) > 0$ **then**
    begin

```
        i ← deg(u);
        while (u_i = v_i) and (i > 0) do i ← i − 1;
        if u_i > v_i then
            sign ← 1
        else if u_i < v_i then
            sign ← −1
        else
            sign ← 0
        end
    else { u = v = 0 }
        sign ← 0
end; { NCOMP }
```

Korektnost algoritma je očita. Primijetimo da je algoritam sličan algoritmu NSUBC za $i = n$.

Za složenost vrijedi

$$\mathrm{Compl}_A(\mathsf{NCOMP}) = 0$$

i

$$\mathrm{w-}\ \mathrm{Compl}_T(\mathsf{NCOMP}) \sim \ell\,(u)\ ,$$

za slučaj da je $u = v$, dok je prosječna vremenska složenost konstantna.

Često je korisna modifikacija ovog algoritma koja vraća i broj d definiran sa

$$d = \begin{cases} \min\,\{\ i\ \mid\ u_i \neq v_i\ \ ,\ \ 0 \leq i \leq n\ \} & ,\quad \text{za } u \neq v \\ -1 & ,\quad \text{za } u = v \end{cases}$$

Tada je za $u \neq v$:

$$\mathrm{sign}(u - v) = \mathrm{sign}(u_{\,d} - v_{\,d})\ ,$$

uz dogovor $v_i = 0\ \ ,\ \ i = m+1, \ldots, n$.

1.4. Množenje

Algoritmi za zbrajanje i oduzimanje su brzi algoritmi, jer njihova kompleksnost ovisi linearno o duljini brojeva (rezultata). Osim toga, oni koriste samo zbrajanje i oduzimanje pojedinih znamenki, tj. najbrže operacije aritmetike računala.

Pokazali smo i da su ti algoritmi optimalni, do na mali konstantni faktor.

Nažalost, ništa od toga ne vrijedi za množenje i dijeljenje. Optimalni algoritmi za izvođenje te dvije operacije nisu poznati.

Zbog toga dajemo niz algoritama za množenje sa sve boljim asimptotskim karakteristikama složenosti. Dijeljenje i ostale operacije koriste množenje, pa njihova složenost ovisi o odabranom algoritmu za množenje.

Prvo izlažemo "klasični" algoritam za množenje polinoma i brojeva.

Neka su $U(x)$, $V(x)$ polinomi oblika

$$U(x) = \sum_{i=0}^{n} u_i \, x^i \quad , \quad V(x) = \sum_{i=0}^{m} v_i \, x^i \quad ,$$

uz pretpostavku da su vodeći koeficijenti različiti od nule

$$u_n \neq 0 \quad , \quad v_m \neq 0 \quad ,$$

tj. $\deg(U) = n$, $\deg(V) = m$. Za produkt

$$W(x) = U(x) \cdot V(x) \tag{1.4.1}$$

znamo da je

$$\deg(W) = \deg(U) + \deg(V) \ ,$$

pa nema problema s nalaženjem stupnja, kao kod zbrajanja ili oduzimanja.

Označimo

$$W(x) = \sum_{i=0}^{k} w_i \, x^i \ ,$$

gdje je $k = n + m = \deg(W)$. Direktno iz zapisa (1.4.1) dobivamo

$$W(x) = \sum_{i=0}^{n} \sum_{j=0}^{m} u_i \cdot v_j \, x^{i+j} \ , \tag{1.4.2}$$

što daje sljedeći algoritam.

Algoritam 1.4.1. (PMUL – **množenje polinoma)**

Ulaz: Nizovi koeficijenata $(u_0, \ldots, u_n)$ i $(v_0, \ldots, v_m)$ polinoma $U(x)$, $V(x)$.

Izlaz: Niz koeficijenata $(w_0, \ldots, w_k)$ polinoma

$$W(x) = U(x) \cdot V(x) \ .$$

procedure PMUL (U , V : polinom ; **var** W : polinom);

begin
$\deg(W) \leftarrow \deg(U) + \deg(V)$;
for $i \leftarrow 0$ **to** $\deg(W)$ **do** $w_i \leftarrow 0$;

```
    for  i ← 0  to  deg(U)  do
        for  j ← 0  to  deg(V)  do
            w_{i+j} ← w_{i+j} + u_i · v_j;
end;  { PMUL }
```

Algoritam očito radi korektno, zbog (1.4.2). U aritmetici računala, algoritam radi korektno, ako su rezultati svih operacija egzaktno prikazivi.

Za prostornu složenost vrijedi

$$\mathrm{Compl}_S(\mathsf{PMUL}) = 2\,(\ell\,(U) + \ell\,(V)) + 4\ ,$$

računajući $\ell\,(W) = \ell\,(U) + \ell\,(V) - 1$. Ako petlju po i preuredimo tako da radi silazno od $\deg(U)$ do 0, onda koeficijente $w_0, \ldots, w_n$ možemo spremiti na mjesto $u_0, \ldots, u_n$, što smanjuje prostornu složenost za $\ell\,(U)$. Tada je potrebno izvršiti još neke modifikacije algoritma.

Aritmetička složenost algoritma je

$$\mathrm{Compl}_A(\mathsf{PMUL}) = \begin{cases} \ell\,(U)\cdot\ell\,(V) + 1 & \text{zbrajanja} \\ \ell\,(U)\cdot\ell\,(V) & \text{množenja} \end{cases},$$

računajući i zbrajanje stupnjeva. Ako ne razlikujemo aritmetičke operacije, onda je

$$\mathrm{Compl}_A(\mathsf{PMUL}) = 2\,\ell\,(U)\cdot\ell\,(V) + 1\ . \tag{1.4.3}$$

Za vremensku složenost je očito

$$\mathrm{Compl}_T(\mathsf{PMUL}) \sim \ell\,(U)\cdot\ell\,(V)\ .$$

Uočimo da ovaj algoritam **ne računa** koeficijente w_i jedan za drugim, već redom pribraja sumande za različite koeficijente.

Ako želimo redom računati koeficijente w_i , onda koristimo relaciju

$$w_i = \sum_{j=\max\{\,0\,,\,i-m\,\}}^{\min\{\,i\,,\,n\,\}} u_j \cdot v_{i-j} \quad ,\ \ i = 0, \ldots, n+m\ . \tag{1.4.4}$$

Za efikasnu realizaciju odgovarajućeg algoritma treba svakako izbjeći računanje granica sume u (1.4.4) za svaki i.

Ovisno o i, za granice vrijedi :

$$\max\{\,0\,,\,i-m\,\} = \begin{cases} 0 & ,\ \ \text{za}\ \ 0 \le i \le m \\ i-m & ,\ \ \text{za}\ \ m < i \le n+m\ , \end{cases} \tag{1.4.5}$$

$$\min\{\,i\,,\,n\,\} = \begin{cases} i & ,\ \ \text{za}\ \ 0 \le i < n \\ n & ,\ \ \text{za}\ \ n \le i \le n+m\ . \end{cases} \tag{1.4.6}$$

To pokazuje da treba razlikovati slučajeve $n \ge m$, odnosno $n \le m$.

(a) Ako je $n \geq m$, onda je :

i	$\max\{\ 0\ ,\ i-m\ \}$	$\min\{\ i\ ,\ n\ \}$
$0 \leq i \leq m$	0	i
$m < i < n$	$i-m$	i
$n \leq i \leq n+m$	$i-m$	n

(b) Ako je $n \leq m$, onda je :

i	$\max\{\ 0\ ,\ i-m\ \}$	$\min\{\ i\ ,\ n\ \}$
$0 \leq i < n$	0	i
$n \leq i \leq m$	0	n
$m < i \leq n+m$	$i-m$	n

Najjednostavnije je izabrati jedan od ta dva slučaja, a drugi svesti na izabrani zamjenom U i V. Sljedeći algoritam realizira slučaj (b).

Algoritam 1.4.2. (PMUL1 – **množenje polinoma)**

Ulaz: Nizovi koeficijenata $(u_0, \ldots, u_n)$ i $(v_0, \ldots, v_m)$ polinoma $U(x)$, $V(x)$, uz pretpostavku da je $n \leq m$.

Izlaz: Niz koeficijenata $(w_0, \ldots, w_k)$ polinoma

$$W(x) = U(x) \cdot V(x)\ .$$

procedure PMUL1 (U , V : polinom ; **var** W : polinom);

begin
  $\deg(W) \leftarrow \deg(U) + \deg(V)$;
  for $i \leftarrow 0$ **to** $\deg(W)$ **do** $w_i \leftarrow 0$;
  for $i \leftarrow 0$ **to** $\deg(U) - 1$ **do**
    for $j \leftarrow 0$ **to** i **do**
      $w_i \leftarrow w_i + u_j \cdot v_{i-j}$;
  for $i \leftarrow \deg(U)$ **to** $\deg(V)$ **do**
    for $j \leftarrow 0$ **to** n **do**
      $w_i \leftarrow w_i + u_j \cdot v_{i-j}$;
  for $i \leftarrow \deg(V) + 1$ **to** $\deg(W)$ **do**
    for $j \leftarrow i - \deg(V)$ **to** n **do**
      $w_i \leftarrow w_i + u_j \cdot v_{i-j}$
end; { PMUL1 }

Algoritam 1.4.3. (PMUL – **množenje polinoma)**

Ulaz: Nizovi koeficijenata $(u_0, \ldots, u_n)$ i $(v_0, \ldots, v_m)$ polinoma $U(x)$, $V(x)$.

Izlaz: Niz koeficijenata $(w_0, \ldots, w_k)$ polinoma

$$W(x) = U(x) \cdot V(x) \ .$$

procedure PMUL (U , V : polinom ; **var** W : polinom);

```
begin
    if deg(U) ≤ deg(V) then
        PMUL1 (U , V , W)
    else
        PMUL1 (V , U , W)
end; { PMUL }
```

Složenost ove varijante algoritma PMUL je ista kao i za prošlu varijantu.

Napomena 1.4.1. *Relacija (1.4.3) pokazuje da je za množenje dva polinoma stupnja n, potrebno*

$$\mathcal{O}(n^2)$$

aritmetičkih operacija. Korištenjem brze Fourierove transformacije (FFT), moguće je broj aritmetičkih operacija smanjiti na

$$\mathcal{O}(n \log n) \ .$$

Taj algoritam je brži od klasičnog tek za vrlo velike n, a maksimalno je efikasan samo za stupnjeve oblika

$$n = 2^k \ .$$

Brza Fourierova transformacija i pripadni algoritam za množenje polinoma su opisani u [Wilf, 1986.] , [Aho, Hopcroft, Ullman, 1976.] .

Množenje brojeva u pozicionom zapisu možemo realizirati direktno po ugledu na množenje polinoma, uz normalizaciju znamenki i propagiranje prijenosa.

Prvo analiziramo algoritam za množenje broja u pozicionom zapisu i jedne znamenke (odnosno jednodimenzionalnog broja) u odabranoj bazi. Taj algoritam je sam za sebe koristan, a može poslužiti za razvoj općeg algoritma za množenje prirodnih brojeva.

Algoritam 1.4.4. (NMULD – **množenje prirodnog broja znamenkom**)

Ulaz: Niz znamenki $(u_n, \ldots, u_0)$ broja u i znamenka v_0 u odabranoj bazi b.

Izlaz: Niz znamenki $(w_k, \ldots, w_0)$ broja

$$w = u \cdot v_0$$

u bazi b.

procedure NMULD (u : mpnat ; v_0 : digit ; **var** w : mpnat);

```
begin
  if v0 > 0 then
    begin
    carry ← 0;
    for i ← 0 to deg(u) do
      begin
      temp ← u_i · v0 + carry;
      w_i ← temp mod b;
      carry ← temp div b
      end
    if carry > 0 then
      begin
      deg(w) ← deg(u) + 1;
      w_deg(w) ← carry
      end
    else
      deg(w) ← deg(u)
    end
  else
    deg(w) ← −1
end; { NMULD }
```

Ovdje je *digit* tip koji služi za prikaz znamenki u pozicionom prikazu, na pr.

$$digit = 0 \ldots b1 \ ,$$

gdje je $b1 = b - 1$ konstanta.

Propozicija 1.4.1.

Ako je $u \in \mathbb{N}_0$ *broj dan pozicionim zapisom u bazi* b

$$u = (u_n \ldots u_0)_b$$

i ako je $v_0 \in \mathbb{N}_0$ *i*

$$0 \leq v_0 < b \ ,$$

onda algoritam NMULD *daje niz znamenki pozicionog zapisa broja* $u \cdot v_0$ *u bazi* b

$$u \cdot v_0 = w = (w_k \dots w_0)_b \ ,$$

gdje je $k = \deg_b(w) \in \{\ n\ ,\ n+1\ \}$ *, za* $v_0 > 0$ *i* $k = -1$ *, za* $v_0 = 0$ *.*

Dokaz:

Analogno ranijim dokazima korektnosti algoritama. Možemo pretpostaviti da je $v_0 > 0$, jer za $v_0 = 0$ algoritam postavlja $\deg(w) = -1$, što odgovara $w = 0$. Rad algoritma opisan je rekurzivnim relacijama :

$$\left.\begin{aligned} carry_{-1} &= 0 \\ w_i &= (u_i \cdot v_0 + carry_{i-1}) \bmod b \\ carry_i &= \lfloor (u_i \cdot v_0 + carry_{i-1})/b \rfloor \end{aligned}\right\} \quad i = 0, \dots, n \tag{1.4.7}$$

Zbog

$$carry_i \cdot b + w_i = u_i \cdot v_0 + carry_{i-1} \ \ , \ \ i = 0, \dots, n \ , \tag{1.4.8}$$

indukcijom dobivamo :

$$carry_i \cdot b^{i+1} + \sum_{j=0}^{i} w_j \, b^j = v_0 \cdot \sum_{j=0}^{i} u_j \, b^j \ , \tag{1.4.9}$$

za $i = 0, \dots, n$. To znači da algoritam u svakom koraku nalazi korektni produkt donjeg dijela broja u i znamenke v_0, što možemo zapisati u obliku

$$carry_i \cdot b^{i+1} + w \bmod b^{i+1} = v_0 \, (u \bmod b^{i+1})$$

za $i = 0, \dots, n$. Za $i = n$ izlazi

$$carry_n \cdot b^{n+1} + w \bmod b^{n+1} = v_0 \cdot u \ . \tag{1.4.10}$$

Tvrdimo da je

$$\begin{aligned} 0 \ \leq \ u_i \cdot v_0 + carry_{i-1} \ &\leq \ b^2 - b - 1 \\ 0 \ \leq \ carry_i \ &\leq \ b - 2 \end{aligned} \quad , \quad i = 0, \dots, n \tag{1.4.11}$$

Dokaz je indukcijom, iz $0 \leq u_i \leq b-1$ i $0 < v_o \leq b-1$. Tada je, zbog $carry_{-1} = 0$,

$$0 \ \leq \ u_i \cdot v_0 \ \leq \ (b-1)^2 \ = \ b^2 - 2b - 1 \ ,$$

pa vrijedi prva relacija (1.4.11) za $i = 1$.

Iz (1.4.7), dobivamo

$$0 \leq carry_0 \leq \left\lfloor \frac{(b-1)^2}{b} \right\rfloor = \left\lfloor b - 2 + \frac{1}{b} \right\rfloor .$$

Zbog $b \in \mathbb{N}$, $b \geq 2$, vrijedi

$$\left\lfloor b - 2 + \frac{1}{b} \right\rfloor = b - 2 ,$$

čime je dokazano (1.4.11) za $i = 0$.

Iz pretpostavke $0 \leq carry_{i-1} \leq b - 2$, dobivamo

$$0 \leq u_i \cdot v_0 + carry_{i-1} \leq (b-1)^2 + b - 2 = b^2 - b - 1 ,$$

a (1.4.7) daje

$$0 \leq carry_i \leq \left\lfloor b - 1 - \frac{1}{b} \right\rfloor = b - 2 ,$$

što dokazuje (1.4.11).

Ako je $carry_n = 0$, algoritam postavlja $\deg(w) = \deg(u)$, pa (1.4.10) daje

$$w = v_0 \cdot u .$$

Zbog $v_0 > 0$ i $u_n > 0$, iz (1.4.8) izlazi

$$w_n = u_n \cdot v_0 + carry_{n-1} \geq u_n > 0 ,$$

pa je $w = (w_n \ldots w_0)_b$ normalizirani prikaz broja $w = v_0 \cdot u$.

Ako je $carry_n > 0$, algoritam postavlja

$$\begin{aligned} \deg(w) &= \deg(u) + 1 \\ w_{n+1} &= carry_n , \end{aligned}$$

pa (1.4.10) pokazuje da je

$$w = (w_{n+1} \ldots w_0)_b$$

normalizirani prikaz broja $w = v_0 \cdot u$. ■

Prva relacija (1.4.11) daje zahtjev

$$b^2 - b - 1 \leq maxint , \tag{1.4.12}$$

potreban da algoritam radi korektno i u aritmetici računala.

Napomena 1.4.2. *Ako u algoritmu* NMULD *zamijenimo naredbu*

$$temp \leftarrow u_i \cdot v_0 + carry$$

sa

$$temp \leftarrow w_i + u_i \cdot v_0 + carry \ ,$$

onda će algoritam postaviti

$$w \leftarrow w + u \cdot v_0 \ ,$$

uz kontrolu stupnja ulaznog broja w *i eventualno propagiranje prijenosa. Zahtjev (1.4.12) tada glasi*

$$b^2 - 1 \leq maxint \ ,$$

jer je tada $0 \leq carry_i \leq b - 1$.

Prostorna složenost je očito

$$\mathrm{Compl}_S(\mathsf{NMULD}) = 2\,\ell\,(u) + c$$

sa $c \leq 6$, što se može smanjiti, jer w možemo spremiti na mjesto broja u.

Primijetimo da je ovdje nemoguće zamijeniti cjelobrojno dijeljenje s ostatkom, nekim brzim operacijama, zbog relacija (1.4.11).

Iz viših programskih jezika, operacije div i mod se vrše odvojeno i traju podjednako, dok arhitektura računala odmah daje kvocijent i ostatak. Zbog toga svaku od tih operacija brojimo posebno.

Algoritam uvijek obavlja isti broj operacija, pa su najgora, najbolja i prosječna aritmetička složenost jednake.

$$\mathrm{Compl}_A(\mathsf{NMULD}) = \begin{cases} \ell\,(u) & \text{zbrajanja} \\ \ell\,(u) & \text{množenja} \\ 2\,\ell\,(u) & \text{dijeljenja} \end{cases}$$

ili, ukupno

$$\mathrm{Compl}_A(\mathsf{NMULD}) = 4\,\ell\,(u) \ . \tag{1.4.13}$$

Vremenska složenost, također linearno ovisi o duljini broja u

$$\mathrm{Compl}_T(\mathsf{NMULD}) \sim \ell\,(u) \sim \ell\,(w) \ .$$

Sljedeći, vrlo važan, specijalan slučaj množenja, je množenje potencijom baze.

Algoritam 1.4.5. (NMULB – **množenje potencijom baze**)

Ulaz: Niz znamenki $(u_n, \ldots, u_0)$ broja u i potencija $p \in \mathbb{N}_0$, odabrane baze b.

Izlaz: Niz znamenki $(w_k, \ldots, w_0)$ broja

$$w = u \cdot b^p, \tag{1.4.14}$$

u bazi b.

```
procedure NMULB ( u : mpnat ;  p : integer ;  var  w : mpnat);

begin
    if  deg(u) ≥ 0  then
        begin
        deg(w) ← deg(u) + p;
        for  i ← deg(u)  downto  0  do
            w_{i+p} ← u_i;
        for  i ← 0  to  p − 1  do
            w_i ← 0
        end
    else
        deg(w) ← −1
end;  { NMULB }
```

Korektnost algoritma je trivijalna, jer se množenje svodi na pomak znamenki.

Prostorna složenost je

$$\mathrm{Compl}_S(\mathsf{NMULB}) = 2\,\ell\,(u) + p + c\ ,$$

sa $c \leq 4$, a faktor 2 se može eliminirati, ako w spremamo na mjesto broja u.

Aritmetička složenost je konstantna

$$\mathrm{Compl}_A(\mathsf{NMULB}) = 1\ ,$$

zbog zbrajanja stupnjeva.

Vremenska složenost je izrazito ovisna o izboru strukture podataka *mpnat*. Prikazom broja vezanom listom znamenki, može se izbjeći pomicanje (prva petlja). U najgorem slučaju je :

$$\mathrm{Compl}_T(\mathsf{NMULB}) \sim \ell\,(w) = \ell\,(u) + p\ .$$

Napomena 1.4.3. *Algoritam* NMULB *se rijetko koristi kao zaseban algoritam. Računanje broja* $w = u \cdot b^p$ *je obično dio nekog drugog algoritma. Tada je mnogo bolje*

koristiti direktno znamenke broja u, umjesto znamenki broja w, uz pažljivo indeksiranje, koje zamjenjuje eksplicitni pomak znamenki. Ovo može znatno smanjiti potrebno vrijeme, jer se izbjegava kopiranje znamenki.

Ovdje izloženi algoritam NMULB *služi samo za pojednostavljivanje zapisa nekih drugih algoritama, koji koriste množenja potencijom baze.*

Posljednja dva algoritma, zajedno sa zbrajanjem, omogućavaju direktnu realizaciju općeg algoritma za množenje. Naime, produkt

$$w = u \cdot v = \sum_{i=0}^{n} u_i \, b^i \;\cdot\; \sum_{j=0}^{m} v_j \, b^j$$

možemo zapisati u obliku

$$w = \sum_{j=0}^{m} (u \cdot v_j) \cdot b^j \tag{1.4.15}$$

i primijeniti algoritam sličan Hornerovoj shemi, što odgovara "ručnom" algoritmu množenja.

Algoritam 1.4.6. (NMUL0 – **množenje prirodnih brojeva)**

Ulaz: Nizovi znamenki $(u_n, \ldots, u_0)$, $(v_m, \ldots, v_0)$ brojeva u i v u odabranoj bazi b.

Izlaz: Niz znamenki $(w_k, \ldots, w_0)$ broja

$$w = u \cdot v$$

u bazi b, uz $k = \deg_b(w)$.

procedure NMUL0 (u , v : mpnat ; **var** w : mpnat);

begin
  $w \leftarrow 0$;
  for $j \leftarrow \deg(v)$ **downto** 0 **do**
    $w \leftarrow w \cdot b + u \cdot v_j$
end; { NMUL0 }

Ovdje, naravno, sve operacije treba obaviti na nizovima znamenki odgovarajućih brojeva. To znači da bi svaku operaciju trebalo zamijeniti pozivom odgovarajućeg algoritma. Pri tome nastaje problem sa spremanjem međurezultata. Tako "raspisani" algoritam NMUL0 izgleda ovako :

```
procedure NMUL0 ( u , v : mpnat ; var w : mpnat);

begin
    deg(w) ← −1;
    for j ← deg(v) downto 0 do
        begin
        NMULB (w , 1 , t1);          { t1 ← w · b }
        NMULD (u , vj , t2);         { t2 ← vj · u }
        NADD (t1 , t2 , w)           { w ← t1 + t2 }
        end
end; { NMUL0 }
```

Ovdje su t_1 , t_2 pomoćni brojevi tipa *mpnat*, što troši dodatnu memoriju i dodatno vrijeme, jer se sav posao, očito može obaviti i bez pomoćnih brojeva t_1 i t_2.

Osim toga, ne koristimo neka pogodna svojstva brojeva za smanjenje broja aritmetičkih operacija. Na primjer, u pozivu algoritma NADD ne koristimo činjenicu da je najniža znamenka broja $t_1 = w \cdot b$ jednaka nuli.

Napomena 1.4.4. *Katkad ćemo koristiti skraćeni zapis algoritma, pišući standardne oznake za aritmetičke operacije na brojevima, umjesto čitavog niza naredbi koji realizira te operacije na nizovima znamenki u pozicionom zapisu brojeva. Time dobivamo preglednost algoritma.*

Pri tome je zamjena operacija pozivom algoritma najlošiji način realizacije takvog algoritma.

U skraćenom zapisu pretpostavljamo da odgovarajuće operacije treba izvesti maksimalno efikasno – direktno na nizovima znamenki brojeva.

U tom smislu, pogodnije je koristiti direktno zapis (1.4.15), a ne Hornerovu shemu, jer smanjujemo broj pomaka i broj aritmetičkih operacija.

Algoritam 1.4.7. (NMUL – množenje prirodnih brojeva)

Ulaz: Nizovi znamenki $(u_n, \ldots, u_0)$, $(v_m, \ldots, v_0)$ brojeva u i v u odabranoj bazi b.

Izlaz: Niz znamenki $(w_k, \ldots, w_0)$ broja

$$w = u \cdot v$$

u bazi b.

Skraćeni zapis algoritma je :

```
procedure NMUL ( u , v : mpnat ; var w : mpnat);

begin
    w ← 0;
    for j ← 0 to deg(v) do
        w ← w + (u · v_j) · b^j
end; { NMUL }
```

Efikasna realizacija ovog skraćenog zapisa je :

```
procedure NMUL ( u , v : mpnat ; var w : mpnat);

begin
    if (deg(u) ≥ 0) and (deg(w) ≥ 0) then
        begin
        deg(w) ← deg(u) + deg(v);
        for i ← 0 to deg(u) do w_i ← 0;
        for j ← 0 to deg(v) do
            begin
            carry ← 0;
            for i ← 0 to deg(u) do
                begin
                temp ← w_{i+j} + u_i · v_j + carry;
                w_{i+j} ← temp mod b;
                carry ← temp div b
                end;
            w_{deg(u)+j+1} ← carry;
            end;
        if carry > 0 then
            deg(w) ← deg(u) + deg(v) + 1
        else
            deg(w) ← deg(u) + deg(v)
        end
    else
        deg(w) ← −1
end; { NMUL }
```

Uočimo sličnost između ovog algoritma i algoritma 1.4.1. za polinome, pogotovo ako u PMUL zamijenimo mjesta petlje po i i petlje po j.

Dokažimo da ovaj algoritam radi korektno u egzaktnoj aritmetici.

Propozicija 1.4.2.

Ako su brojevi $u\ ,\ v \in \mathbb{N}_0$ *dani pozicionim zapisima u bazi* b

$$u = (u_n \dots u_0)_b \quad , \quad v = (v_m \dots v_0)_b \ ,$$

onda algoritam NMUL *daje niz znamenki pozicionog zapisa broja* $u \cdot v$

$$u \cdot v = w = (w_k \dots w_0)_b \ ,$$

gdje je

$$k = \deg_b(w) \in \{\ n+m\ ,\ n+m+1\ \}\ .$$

Dokaz:

Algoritam očito radi korektno, ako je $u = 0$ ili $v = 0$, jer postavlja $\deg(w) = -1$ tj. $w = 0$.

Pretpostavimo stoga da je $u\ ,\ v > 0$.

Označimo sa $w_{(-1)}$ polazno stanje broja (niza) w u algoritmu, a sa $w_{(j)}$, $j = 0, \dots, m$, stanje broja w nakon izvršenja vanjske petlje algoritma s kontrolnom vrijednošću j.

Znamenke broja $w_{(j)}$ označavamo sa

$$w_{(j)} = (w_{n+j+1\,,\,j}\ , \dots ,\ w_{0\,,\,j})\ ,$$

tj.

$$w_{(j)} = \sum_{i=0}^{n+j+1} w_{i\,,\,j}\, b^i \quad , \quad j = -1, \dots, m\ , \tag{1.4.16}$$

pri čemu ovaj zapis ne mora biti normaliziran. Algoritam inicijalno postavlja

$$w_{i\,,\,-1} = 0 \quad , \quad i = 0, \dots, n \tag{1.4.17}$$

tj.

$$w_{(-1)} = 0\ .$$

Označimo vrijednosti varijable *carry*, tokom prolaza kroz vanjsku petlju s indeksom j, na sljedeći način: početna vrijednost je $carry_{j-1\,,\,j} = 0$, a vrijednost nakon prolaza kroz unutarnju petlju s indeksom i je $carry_{i+j\,,\,j}$, za $i = 0, \dots, n$.

Za $j = 0$, rad algoritma možemo opisati rekurzivnim relacijama

$$\begin{gathered} carry_{-1\,,\,0} = 0 \\ \left.\begin{aligned} w_{i\,,\,0} &= (u_i \cdot v_0 + carry_{i-1\,,\,0}) \bmod b \\ carry_{i\,,\,0} &= \lfloor (u_i \cdot v_0 + carry_{i-1\,,\,0})/b \rfloor \end{aligned}\right\} \quad i = 0, \dots, n \\ w_{n+1\,,\,0} = carry_{n\,,\,0} \quad , \end{gathered} \tag{1.4.18}$$

s tim da smo iskoristili relaciju (1.4.17).

Uspoređivanjem s (1.4.7), po propoziciji 1.4.1. izlazi da je

$$w_{(0)} = u \cdot v_0 \ . \tag{1.4.19}$$

Za $j \geq 1$, rad algoritma je opisan rekurzivnim relacijama :

$$w_{i\,,\,j} = w_{i\,,\,j-1} \ \text{ , za } \ i = 0, \ldots, j-1 \ , \tag{1.4.20}$$

jer algoritam ne mijenja ta mjesta, i

$$\begin{aligned} carry_{j-1\,,\,j} &= 0 \\ w_{i+j\,,\,j} &= (w_{i+j\,,\,j-1} + u_i \cdot v_j + carry_{i+j-1\,,\,j}) \bmod b \\ carry_{i+j\,,\,j} &= \lfloor (w_{i+j\,,\,j-1} + u_i \cdot v_j + carry_{i+j-1\,,\,j})/b \rfloor \\ &\text{za } \ i = 0, \ldots, n \end{aligned} \tag{1.4.21}$$

i prvi puta se postavlja znamenka

$$w_{n+j+1\,,\,j} = carry_{n+j\,,\,j} \ . \tag{1.4.22}$$

Iz relacija (1.4.21) je

$$\begin{aligned} carry_{i+j\,,\,j} \cdot b + w_{i+j\,,\,j} &= w_{i+j\,,\,j-1} + u_i \cdot v_j + carry_{i+j-1\,,\,j} \\ &\text{za } \ i = 0, \ldots, n \ . \end{aligned} \tag{1.4.23}$$

Odavde je indukcijom :

$$\begin{aligned} &carry_{i+j\,,\,j} \cdot b^{i+1} + \sum_{l=0}^{i} w_{l+j\,,\,j}\, b^l = \\ &= \sum_{l=0}^{i} w_{l+j\,,\,j-1}\, b^l + v_j \cdot \sum_{l=0}^{i} u_l\, b^l \ \text{ , za } \ i = 0, \ldots, n \ . \end{aligned}$$

Množenjem ove relacije s b^j dobivamo

$$\begin{aligned} &carry_{i+j\,,\,j} \cdot b^{i+j+1} + \sum_{l=j}^{i+j} w_{l\,,\,j}\, b^l = \\ &= \sum_{l=j}^{i+j} w_{l\,,\,j-1}\, b^l + (v_j \cdot \sum_{l=0}^{i} u_l\, b^l)\, b^j \ \text{ , za } \ i = 0, \ldots, n \ . \end{aligned} \tag{1.4.24}$$

Iz relacije (1.4.20) slijedi :

$$\sum_{l=0}^{j-1} w_{l\,,\,j}\, b^l = \sum_{l=0}^{j-1} w_{l\,,\,j-1}\, b^l$$

(tj. $w_{(j)} \bmod b^j = w_{(j-1)} \bmod b^j$).

Zbrajanjem ove relacije i relacije (1.4.24) izlazi

$$carry_{i+j\,,\,j} \cdot b^{i+j+1} + \sum_{l=0}^{i+j} w_{l\,,\,j}\, b^l =$$

$$= \sum_{l=0}^{i+j} w_{l\,,\,j-1}\, b^l + (v_j \cdot \sum_{l=0}^{i} u_l\, b^l)\, b^j \quad , \text{ za } \; i = 0, \ldots, n \; .$$

Odavde za $i = n$, korištenjem relacije (1.4.16) dobivamo

$$carry_{n+j\,,\,j} \cdot b^{n+j+1} + w_{(j)} \bmod b^{n+j} = w_{(j-1)} + (v_j \cdot u)\, b^j \; .$$

Iz (1.4.22) i (1.4.16) izlazi

$$w_{(j)} = w_{(j-1)} + (v_j \cdot u)\, b^j \quad , \;\; j = 1, \ldots, m \; . \tag{1.4.25}$$

Koristeći relaciju (1.4.19) kao bazu indukcije i (1.4.25) kao korak indukcije, izlazi da je

$$w_{(m)} = \sum_{j=0}^{m} (v_j \cdot u)\, b^j$$

ili

$$w_{(m)} = u \cdot v \; .$$

Očito, iz (1.4.18) i (1.4.21) za sve znamenke $w_{i\,,\,j}$ vrijedi

$$0 \le w_{i\,,\,j} < b \quad , \quad \begin{array}{l} j \;=\; 0, \ldots, m \\ i \;=\; 0, \ldots, n + j + 1 \; . \end{array} \tag{1.4.26}$$

Na samom kraju, algoritam provjerava zadnju vrijednost varijable *carry*, tj. vrijednost

$$w_{n+m+1\,,\,m} = carry_{n+m\,,\,m} \; .$$

Ako je ta vrijednost pozitivna, algoritam postavlja

$$\deg(w) = n + m + 1$$

i daje

$$w = w_{(m)} \; ,$$

što je korektni normalizirani rezultat.

Ako je $w_{n+m+1\,,\,m} = 0$, algoritam daje

$$\deg(w) = n + m \; ,$$

tj. postavlja

$$w = w_{(m)} \bmod b^{n+m} = w_{(m)} \; .$$

Treba još dokazati da je tada

$$w_{\deg(w)} = w_{n+m\,,\,m} > 0\ .$$

Koristeći relacije (1.4.11) za bazu indukcije ($j = 0$) i normaliziranost brojeva u, v, indukcijom po i, pa po j, izlazi u (1.4.21) :

$$\begin{gathered} 0 \ \le\ w_{i+j\,,\,j} + u_i \cdot v_j + carry_{i+j-1\,,\,j} \ \le\ b^2 - 1 \\ 0 \ \le\ carry_{i+j\,,\,j} \ \le\ b - 1 \end{gathered} \tag{1.4.27}$$

za sve $j = 0, \ldots, m$ i $i = 0, \ldots, n$.

Za $i = n$, $j = m$, ako je $carry_{n+m\,,\,m} = 0$, onda iz (1.4.23) izlazi

$$w_{n+m\,,\,m} = w_{n+m\,,\,m-1} + u_n \cdot v_m + carry_{n+m-1\,,\,m}\ .$$

Iz relacija (1.4.26) i (1.4.27), koristeći $u_n > 0$, $v_m > 0$ (jer u , $v > 0$), dobivamo

$$w_{n+m\,,\,m} \ge u_n \cdot v_m > 0\ .$$

Time je tvrdnja u potpunosti dokazana. ■

Relacija (1.4.27) pokazuje da je $w_{i+j} + u_i \cdot v_j + carry$ najveća vrijednost koju algoritam računa.

Algoritam radi korektno u aritmetici računala ako i samo ako je ta vrijednost uvijek egzaktno prikaziva, što znači da broj $b^2 - 1$ mora biti egzaktno prikaziv. Time je dokazana

Propozicija 1.4.3.

Algoritam NMUL *radi korektno u aritmetici računala, ako i samo ako baza b zadovoljava uvjet*

$$b^2 - 1 \le maxint\ . \tag{1.4.28}$$

■

Ovo je, do sada, najjači zahtjev na veličinu baze. Pokazat ćemo, kasnije, da je tada i dijeljenje korektno u aritmetici računala, tj. relacija (1.4.28) daje konačni zahtjev na bazu.

Napomena 1.4.5. *Kao i kod zbrajanja, algoritam se može preurediti tako da nakon svake pojedine operacije normaliziramo rezultat. Zahtjev (1.4.28) se tada može oslabiti do*

$$(b-1)^2 \le maxint\ ,$$

ako želimo da produkt znamenki bude egzaktno prikaziv. Tada se algoritam znatno usporava, a povećanje dozvoljenog raspona za bazu b je neznatno.

Za prostornu složenost je očito

$$\mathrm{Compl}_S(\mathsf{NMUL}) = 2\,(\ell\,(u) + \ell\,(v)) + c\ ,$$

sa $c \leq 7$, računajući i mogućnost da je $\ell\,(w) = \ell\,(u) + \ell\,(v)$, te prostor za duljine brojeva i pomoćne varijable.

Aritmetička složenost algoritma je:

$$\mathrm{Compl}_A(\mathsf{NMUL}) = \begin{cases} 2\,\ell\,(u)\cdot\ell\,(v) + 2 & \text{zbrajanja} \\ \ell\,(u)\cdot\ell\,(v) & \text{množenja} \\ 2\,\ell\,(u)\cdot\ell\,(v) & \text{dijeljenja} \end{cases}$$

ili, ukupno

$$\mathrm{Compl}_A(\mathsf{NMUL}) = 5\,\ell\,(u)\cdot\ell\,(v) + 2\ . \tag{1.4.29}$$

Za vremensku složenost vrijedi:

$$\mathrm{Compl}_T(\mathsf{NMUL}) \sim \ell\,(u)\cdot\ell\,(v)\ . \tag{1.4.30}$$

Aritmetička složenost se može smanjiti, ako umjesto inicijalizacije $w_i = 0$, $i = 0, \ldots, n$, postavimo direktno

$$(w_{n+1} \ldots w_0) = u \cdot v_0\ ,$$

kao u relaciji (1.4.19), koristeći algoritam NMULD. Ovo donosi uštedu od $\ell\,(u)$ zbrajanja, jer izbjegava zbrajanje nule.

Napomena 1.4.6. *Ako potpuno izbacimo inicijalizaciju, algoritam će računati*

$$w \leftarrow w + u \cdot v\ ,$$

tj. akumulirati produkte, s tim da pazimo na ulaznu duljinu broja w i umjesto naredbe $w_{\deg(u)+j+1} \leftarrow carry$, dodamo $carry$ na to mjesto i propagiramo dalje eventualni prijenos.

Na isti način kojim smo algoritam NMUL dobili iz algoritma 1.4.1. za polinome, možemo konstruirati algoritam za množenje brojeva na bazi algoritma 1.4.2. i 1.4.3. za polinome. Taj algoritam računa redom znamenke $w_0, \ldots, w_k$ produkta. Zbog duljine, algoritam nećemo navoditi. Njegova složenost jednaka je složenosti izloženog algoritma.

Ovaj klasični algoritam za množenje je i najbrži algoritam za relativno male duljine brojeva ($\ell\,(u), \ell\,(v)$ manje ili približno jednako 5). Za veće duljine postoje asimptotski mnogo brži algoritmi. Jedan niz takvih algoritma opisat ćemo malo kasnije, nakon algoritama za dijeljenje.

1.5. Dijeljenje s ostatkom

Pod pojmom dijeljenja u prirodnoj i cjelobrojnoj aritmetici podrazumijevamo uvijek dijeljenje s ostatkom na bazi Euklidovog teorema. "Pravo" dijeljenje (bez ostatka) možemo realizirati tek u strukturi polja, a za brojeve — to znači tek kod racionalnih brojeva.

Do sada smo uvijek paralelno promatrali operacije nad polinomima i brojevima. Klasični algoritmi za zbrajanje, oduzimanje i množenje imaju sličnu osnovnu strukturu, a algoritam za brojeve dobivamo, uglavnom, tako da polinomnom algoritmu dodamo normalizaciju znamenki. Isti princip možemo pokušati i za dijeljenje, ali, nažalost, kao što ćemo vidjeti, on **ne** vodi na dobar algoritam za dijeljenje brojeva. Da bismo to pokazali, idemo prvo napraviti algoritam za dijeljenje polinoma.

1.5.1. Dijeljenje polinoma

U standardnim prstenima polinoma nad Euklidskim domenama (to je prsten u kojem vrijedi Euklidov teorem za koeficijente), također, vrijedi i Euklidov teorem za polinome. To znači da za bilo koja dva polinoma U i V, postoje polinomi Q i R, takvi da je

$$U = Q \cdot V + R. \tag{1.5.1}$$

Ako još ograničimo stupanj ostatka tako da je strogo manji od stupnja divizora

$$\deg(R) < \deg(V),$$

(što ujedno znači i $V \neq 0$), onda su polinomi Q i R jedinstveni. Tada je Q kvocijent, a R ostatak.

Ako u domeni koeficijenata polinoma nemamo dijeljenje, kao, na primjer, u prstenu $\mathbb{Z}$ cijelih brojeva, onda divizor V mora imati vodeći koeficijent jednak 1. U protivnom, Q i R ne moraju postojati. Razlog tome postaje očit, čim pokušamo napraviti algoritam za nalaženje Q i R.

Taj algoritam dobivamo direktno iz osnovne relacije (1.5.1), uvrštavanjem koeficijenta polinoma. Označimo koeficijente ova četiri polinoma na sljedeći način

$$U(x) = \sum_{i=0}^{n} u_i x^i, \quad n = \deg(U), \qquad V(x) = \sum_{i=0}^{m} v_i x^i, \quad m = \deg(V),$$
$$Q(x) = \sum_{i=0}^{k} q_i x^i, \quad k = \deg(Q), \qquad R(x) = \sum_{i=0}^{l} r_i x^i, \quad l = \deg(R),$$

Osim toga, uz $\deg(R) < \deg(V)$, znamo da za stupanj kvocijenta Q vrijedi

$$\deg(U) = \begin{cases} \deg(U) - \deg(V), & \text{ako je } \deg(U) \geq 0, \\ \deg(R) = -1, & \text{ako je } \deg(U) = -1. \end{cases}$$

Dakle, ako je $U = 0$ (tj. $\deg(U) = -1$), onda nemamo što računati, jer je $Q = R = 0$. Pretpostavimo stoga da je $U \neq 0$, tj. $\deg(U) \geq 0$.

Kad uvrstimo koeficijente u (1.5.1), za vodeći koeficijent od U dobivamo

$$u_n = q_k \cdot v_m.$$

Ako je domena koeficijenata polje, onda odavde uvijek možemo izračunati q_k, jer je $v_m \neq 0$, čim je $V \neq 0$. U protivnom, ako nismo u polju, moramo uzeti $v_m = 1$ da bismo mogli "podijeliti" u ovoj relaciji, odakle slijedi i $q_k = u_n$. Iz relacija za ostale (niže) koeficijente od U, vidimo da možemo **silaznim** redom izračunati sve koeficijente polinoma Q, a ostatak R dobivamo na kraju.

Dakle, najjednostavniji algoritam dobivamo točnim oponašanjem "ručnog" dijeljenja polinoma. U zapisu tog algoritma dozvoljavamo bilo koji $v_m \neq 0$, kao da smo nad poljem.

begin

{ Stupanj kvocijenta. }

$\deg(Q) := \deg(U) - \deg(V)$;

{ Koeficijenti. }

$R := U$; { Kopiranje polinoma. }

for $j := \deg(Q)$ **downto** 0 **do**

begin

{ Podijeli vodećih m članova iz trenutnog R s V, da dobiješ sljedeći član od Q i nađi novi ostatak R, manjeg stupnja. }

$q_j := r_{j+\deg(V)}/v_m$;

$R(x) := R(x) - (q_j x^j) \cdot V(x)$;

{ Novi $r_{j+\deg(V)} = 0$, pa ga ne računamo. }

end ;

Sredi stupanj od R ;

end ;

U ovom algoritmu treba još samo razraditi sve polinomne operacije u petlje po koeficijentima.

Algoritam 1.5.1. (PDIV — **Dijeljenje polinoma s ostatkom**)

Ulaz: Nizovi koeficijenata $(u_0, \ldots, u_n)$, $(v_0, \ldots, v_m)$ polinoma

$$U(x) = \sum_{i=0}^{n} u_i x^i, \quad V(x) = \sum_{i=0}^{m} v_i x^i,$$

uz pretpostavku $m = \deg(V) \geq 0$. U protivnom, imamo grešku — dijeljenje s nulpolinomom.

Izlaz: Nizovi koeficijenata $(q_0, \ldots, q_k)$, $(r_0, \ldots, r_l)$ polinoma

$$Q(x) = \sum_{i=0}^{k} q_i x^i, \quad R(x) = \sum_{i=0}^{l} r_i x^i,$$

za koje vrijedi

$$U(x) = Q(x) \cdot V(x) + R(x).$$

```
procedure PDIV
        (       U, V : polinom ;
          var Q, R : polinom ) ;

  { Računa Q(x) i R(x), gdje je U(x) = Q(x)·V(x) + R(x),
    uz deg(R) < deg(V). }

begin

  if deg(V) ≥ 0 then
    begin
    if deg(U) ≥ 0 then
      begin
      deg(Q) := deg(U) − deg(V) ;
      for i := 0 to deg(U) do
        r_i := u_i ;
        { Ne treba postaviti deg(R) := deg(U). }
      for j := deg(Q) downto 0 do
        begin
        q_j := r_{j+deg(V)} / v_m ;
        for i := 0 to deg(V) − 1 do
          r_{j+i} := r_{j+i} − q_j · v_i ;
          { Ne treba: r_{j+deg(V)} := 0 ; deg(R) := deg(R) − 1 ; }
        end ;    { for j. }
        { Nađi deg(R). }
      deg(R) := deg(V) − 1 ;
      if deg(R) ≥ 0 then
        begin
        while (deg(R) > 0) and (r_{deg(R)} = 0) do
          deg(R) := deg(R) − 1 ;
```

```
        if r_{deg(R)} = 0 then    { R(x) = 0. }
          deg(R) := −1 ;
        end ;
      end    { if deg(U) ≥ 0. }
    else
      begin    { deg(U) = −1 ⟺ U(x) = 0. }
      deg(Q) := −1 ;
      deg(R) := −1 ;
      end ;    { deg(U) = −1. }
    end    { if deg(V) ≥ 0. }
  else
      { deg(V) = −1 ⟺ V(x) = 0. }
    ERROR ;
end ; { PDIV }
```

Za prostornu složenost u ovoj realizaciji vrijedi

$$\mathrm{Compl}_S(\mathsf{PDIV}) = 3\ell(U) + c = 2\ell(U) + \ell(V) + \ell(Q) + c',$$

jer je $\ell(Q) = \ell(U) - \ell(V) + 1$, a za spremanje R trošimo $\ell(U)$ koeficijenata, zbog inicijalnog kopiranja cijelog U u R.

U pažljivijoj realizaciji, za spremanje polinoma R dovoljno je $\ell(V) - 1$ koeficijenata, tako da zaista svaki puta "spuštamo" po jedan koeficijent. Tada bi trebalo i pomicati koeficijente od R, što može trošiti dodatno vrijeme. U tom slučaju, prostorna složenost se može smanjiti na

$$\mathrm{Compl}_S(\mathsf{PDIV}) = 2\ell(U) + \ell(V) + c = \ell(U) + 2\ell(V) + \ell(Q) + c'.$$

Bolje ne ide, jer stupanj (duljinu) od R možemo odrediti tek na samom kraju algoritma, a još nam treba i "radni komad" duljine barem $\ell(V) - 1$ (što je upravo R u našoj realizaciji).

Aritmetička složenost algoritma je

$$\mathrm{Compl}_A(\mathsf{PDIV}) = \begin{cases} \ell(Q) \cdot (\ell(V) - 1) + 1 & \text{zbrajanja i oduzimanja} \\ \ell(Q) \cdot (\ell(V) - 1) & \text{množenja} \\ \ell(Q) & \text{dijeljenja,} \end{cases}$$

ili, ukupno

$$\mathrm{Compl}_A(\mathsf{PDIV}) = \ell(Q) \cdot (2\ell(V) - 1) + 1$$

aritmetičkih operacija.

Za vremensku složenost, očito, vrijedi

$$\begin{aligned}\mathrm{Compl}_T(\mathsf{PDIV}) &\sim \ell(Q)\cdot\ell(V),\\ &\sim (\ell(U)-\ell(V))\cdot\ell(V).\end{aligned}$$

Jednostavnije rečeno, klasično dijeljenje polinoma stupnja $2n$ polinomom stupnja n (tako da kvocijent ima stupanj n), je **kvadratni** algoritam u n.

1.5.2. Dijeljenje brojeva iz dijeljenja polinoma

Bliskost aritmetike polinoma i aritmetike brojeva u pozicionom zapisu u odabranoj bazi b, počiva na činjenici da broj

$$u = \sum_{i=0}^{n} u_i\, b^i$$

možemo interpretirati kao vrijednost polinoma

$$U(x) = \sum_{i=0}^{n} u_i\, x^i$$

u točki

$$u = U(b)\ .$$

Ako je

$$v = \sum_{i=0}^{m} v_i\, b^i = V(b) \quad , \quad V(x) = \sum_{i=0}^{m} v_i\, x^i\ ,$$

onda je

$$\begin{aligned} u+v &= U(b)+V(b) &&= (U+V)(b)\\ u-v &= U(b)-V(b) &&= (U-V)(b)\\ u\cdot v &= U(b)\cdot V(b) &&= (U\cdot V)(b)\ . \end{aligned} \tag{1.5.2}$$

Zbog toga smo algoritme za brojeve dobivali direktno iz algoritama za polinome, uz dodatak normalizacije znamenki i propagiranje prijenosa, jer koeficijenti polinoma $U+V$, $U-V$, $U\cdot V$ ne moraju biti normalizirani.

Osnovu za cjelobrojno dijeljenje s ostatkom daje Euklidov teorem. Za brojeve u , $v \in \mathbb{N}$, postoje jednoznačno određeni brojevi q , $r \in \mathbb{N}_0$, $0 \le r < v$, takvi da je

$$u = q\cdot v + r\ . \tag{1.5.3}$$

Broj $q = \lfloor u/v \rfloor$ je kvocijent, a $r = u \bmod v$ ostatak.

Euklidov teorem važi i za polinome (na pr. nad $(\mathbb{Z}\ ,\ +\ ,\cdot\ ))$, pa za U, V postoje jednoznačno određeni polinomi Q, R, $\deg(R) < \deg(V)$, takvi da je

$$U(x) = Q(x) \cdot V(x) + R(x)\ .$$

Po analogiji s (1.5.2) vrijedi

$$U(b) = Q(b) \cdot V(b) + R(b)\ ,$$

što daje

$$u = Q(b) \cdot v + R(b)\ .$$

No, ovdje ne postoji jednostavna veza između brojeva q , r i $Q(b)$,$R(b)$ ili znamenki q , r i koeficijenata polinoma Q , R. Normalizacija koeficijenata i propagiranje prijenosa ne mora dati q i r. Zbog toga dijeljenje brojeva ne možemo dobiti iz dijeljenja polinoma.

1.5.3. Dijeljenje brojeva u svakodnevnom životu

Druga mogućnost je algoritamska simulacija procesa dijeljenja "na ruke". Određujemo jednu po jednu znamenku kvocijenta, počev od vodećih, a proces nastavljamo na ostatku prethodnog dijeljenja, uz tzv. "spuštanje" znamenki.

Taj postupak možemo zapisati u obliku algoritma.

Algoritam 1.5.2. (NDIV0 – **dijeljenje prirodnih brojeva s ostatkom**)

Ulaz: Nizovi znamenki $(u_n, \ldots, u_0)$, $(v_m, \ldots, v_0)$ brojeva u i v u odabranoj bazi b. Pretpostavljamo da je $v > 0$. U protivnom, algoritam završava greškom.

Izlaz: Nizovi znamenki $(q_k, \ldots, q_0)$, $(r_l, \ldots, r_0)$ brojeva

$$\begin{aligned} q &= \lfloor u/v \rfloor \quad , \quad k = \deg_b(q) \\ r &= u \bmod v \quad , \quad l = \deg_b(r) \end{aligned}$$

u bazi b.

procedure NDIV0 (u , v : mpnat ; **var** q , r : mpnat);

begin
  if $(\deg(v) \geq 0)$ **then**
    begin
    if $\deg(u) \geq \deg(v)$ **then**
      begin
      $i \leftarrow \deg(u) - \deg(v)$;

```
            u' ← ⌊u/b^i⌋;
            deg(q) ← i;
            loop
                q_i ← ⌊u'/v⌋;     { jednoznamenkasti kvocijent }
                r ← u' − q_i · v  { ostatak }
            exit if  i = 0;
                i ← i − 1;
                u' ← r · b + u_i  { pomak ostatka i spuštanje sljedeće znamenke }
            end;
            if  q_deg(q) = 0  then
                deg(q) ← deg(q) − 1
            end
        else   { u < v }
            begin
            deg(q) ← −1;   { q = 0 }
            r ← u
            end
        end
    else
        ERROR
end;  { NDIV0 }
```

Ovo je skraćeni zapis algoritma, i sve operacije treba obaviti na nizovima znamenki odgovarajućih brojeva.

Prije detaljne razrade algoritma, dokažimo njegovu korektnost.

Propozicija 1.5.1.

Ako su brojevi $u\ ,\ v \in \mathbb{N}_0\ \ v > 0$ *dani pozicionim zapisima u bazi* b

$$u = (u_n \dots u_0)_b \quad , \quad v = (v_m \dots v_0)_b\ ,$$

onda algoritam NDIV0 *daje nizove znamenki brojeva* $q\ ,\ r \in \mathbb{N}_0$

$$q = (q_k \dots q_0)_b \quad , \quad r = (r_l \dots r_0)_b\ ,$$

za koje vrijedi

$$u = q \cdot v + r \quad , \quad 0 \le r < v\ , \tag{1.5.4}$$

i tada je

$$\begin{aligned} k &= \deg_b(q) \in \{\ n-m\ ,\ n-m-1\ \} \\ l &= \deg_b(r) \in \{\ -1, 0, \dots, m\ \}\ . \end{aligned}$$

Dokaz:

Ako je $\deg(u) < \deg(v)$, onda je sigurno i $u < v$. Tada je, u (1.5.4)

$$q = 0 \ , \ r = u$$

i algoritam radi korektno.

Pretpostavimo da je $\deg(u) \geq \deg(v)$.

Označimo sa $u'_{(i)}$, $r_{(i)}$ vrijednosti varijabli (brojeva) u', r u prolazu kroz petlju s vrijednošću i , $i = n-m, \ldots, 0$.

Rad algoritma je opisan rekurzivnim relacijama

$$\begin{aligned} u'_{(n-m)} &= \lfloor u/b^{n-m} \rfloor \ , \\ \left.\begin{aligned} q_i &= \lfloor u'_{(i)}/v \rfloor \\ r_{(i)} &= u'_{(i)} - q_i \cdot v_i \end{aligned}\right\} & \quad i = n-m, \ldots, 0 \\ u'_{(i-1)} &= r_{(i)} \cdot b + u_{i-1} \ , \quad i = n-m, \ldots, 1 \ . \end{aligned} \tag{1.5.5}$$

Dokažimo da za brojeve

$$q = \sum_{i=0}^{n-m} q_i \, b^i \ , \ r = r_{(0)} \ . \tag{1.5.6}$$

vrijedi relacija (1.5.4).

Iz relacije (1.5.5) direktno izlazi

$$\left.\begin{aligned} u'_{(i)} &= q_i \cdot v + r_{(i)} \\ 0 &\leq r_{(i)} < v \end{aligned}\right\} \quad , \ i = n-m, \ldots, 0 \ . \tag{1.5.7}$$

Zbog

$$u'_{(i)} = r_{(i+1)} \cdot b + u_i \ , \ i = n-m-1, \ldots, 0 \ ,$$

dobivamo

$$r_{(i+1)} \cdot b + u_i = q_i \cdot v + r_{(i)} \ , \ i = n-m-1, \ldots, 0 \ . \tag{1.5.8}$$

Iz relacije (1.5.7) za $i = n-m$ i prve relacije u (1.5.5) izlazi

$$\lfloor u/b^{n-m} \rfloor = q_{n-m} \cdot v + r_{(n-m)} \ . \tag{1.5.9}$$

Množenjem svake od relacija (1.5.8) s b^i , i množenjem (1.5.9) s b^{n-m} , te zbrajanjem svih tako dobivenih relacija, izlazi

$$\begin{aligned} \lfloor u/b^{n-m} \rfloor \cdot b^{n-m} + \sum_{i=0}^{n-m-1} u_i \, b^i + \sum_{i=1}^{n-m} r_{(i)} \, b^i = \\ = \left(\sum_{i=0}^{n-m} q_i \right) \cdot v + \sum_{i=0}^{n-m} r_{(i)} \, b^i \ . \end{aligned}$$

Zbog

$$u = \lfloor u/b^{n-m} \rfloor \cdot b^{n-m} + \sum_{i=0}^{n-m-1} u_i\, b^i \ ,$$

dobivamo

$$u = q \cdot v + r_{(0)} \ .$$

Kako je $0 \le r_{(0)} < v$, izlazi $r = r_{(0)}$, čime je (1.5.4) dokazana. Relacija za $l = \deg_b(r)$ je očita, zbog $r < v$.

Preostaje dokazati da je $q_i = \lfloor u'/v \rfloor$ uvijek jednoznamenkasti kvocijent, tj. da za znamenke broja q vrijedi

$$0 \le q_i < b \ \ , \ \ i = n-m, \ldots, 0 \tag{1.5.10}$$

i relaciju za stupanj $k = \deg_b(q)$ kvocijenta.

Zbog $u'_{(n-m)} = \lfloor u/b^{n-m} \rfloor$ i $u < b^{n+1}$ (jer $n = \deg_b(u)$), vrijedi

$$0 < u'_{(n-m)} < b^{m+1} \ . \tag{1.5.11}$$

Kako je i $v \ge b^m$, jer je $m = \deg_b(v)$, to je

$$0 \le q_{n-m} = \lfloor u'_{(n-m)}/v \rfloor < \lfloor b^{m+1}/b^m \rfloor = b \ ,$$

što dokazuje (1.5.10) za $i = n-m$.

Iz relacije (1.5.7) slijedi

$$0 \le r_{(i)} \le v-1 \ \ , \ \ i = n-m, \ldots, 0 \ .$$

Zbog $0 \le u_{i-1} \le b-1$, $i = n-m, \ldots, 1$, iz (1.5.5) dobivamo :

$$\begin{aligned} 0 \le u'_{(i-1)} &= r_{(i)} \cdot b + u_{i-1} \\ &\le (v-1)\, b + b - 1 \\ &= v\, b - 1 \ \ , \text{ za } \ i = n-m, \ldots, 1 \ . \end{aligned} \tag{1.5.12}$$

Pošto je $q_i = \lfloor u'_{(i)}/v \rfloor$, za $i = n-m-1, \ldots, 0$, to, iz (1.5.13) slijedi

$$0 \le q_i \le \lfloor (v\, b - 1)/v \rfloor = \left\lfloor b - \frac{1}{v} \right\rfloor < b \ ,$$

za $i = n-m-1, \ldots, 0$. Time je (1.5.10) dokazana.

Zbog prepostavke $\deg(u) \ge \deg(v)$ je i $u > 0$, tj. $u_n > 0$. To znači da, na početku algoritma broj

$$u' = \lfloor u/b^{n-m} \rfloor$$

ima stupanj $\deg(u') = m = \deg(v)$. Ako je $u' \geq v$, onda je $q_{n-m} > 0$, i tada je

$$k = \deg_b(q) = n - m \ .$$

Ako je $u' < v$, onda je $q_{n-m} = 0$. Ako još vrijedi i $n = m$, tj. $u' = u$, onda je i $u < v$, pa je $q = 0$ i algoritam na kraju postavlja $\deg(q) = -1$ i $r = u' = u$, tj. radi korektno.

Ako je $n > m$, onda tvrdimo da je

$$q_{n-m-1} > 0 \ ,$$

tj $k = \deg_b(q) = n - m - 1$.

Zbog $q_{n-m} = 0$, algoritam u prvom prolazu daje

$$r_{(n-m)} = u'_{(n-m)} = \lfloor u/b^{n-m} \rfloor$$

i postavlja

$$u'_{(n-m-1)} = r_{(n-m)} \cdot b + u_{n-m-1} = \lfloor u/b^{n-m-1} \rfloor \ .$$

Zbog $u_n > 0$, to znači da je

$$\deg_b(u'_{(n-m-1)}) = m + 1 \ ,$$

što daje

$$u'_{(n-m-1)} > v$$

ili

$$q_{n-m-1} = \lfloor u'_{(n-m-1)}/v \rfloor \geq 1 \ .$$

Dakle, algoritam korektno postavlja stupanj kvocijenta q. ■

Napomena 1.5.1. *Ovaj dokaz pokazuje da je*

$$k = \deg_b(q) = n - m \iff u/b^{n-m} \geq v \ .$$

Zbog toga možemo odrediti $\deg(q)$ *prije petlje u algoritmu* NDIV0*, i tako izbjeći testiranje* $q_{\deg(q)} = 0$ *. Iza naredbe*

$$u' \leftarrow \lfloor u/b^i \rfloor$$

možemo uz provjeru $i = n - m > 0$ *, dodati naredbe*

if $u' < v$ **then**
begin
$i \leftarrow i - 1$;
$u' \leftarrow u' \cdot b + u_i \quad \{ = \lfloor u/b^i \rfloor \quad , \quad i = n - m - 1 \ \}$
end

Slučaj $i = 0$ *tj.* $n = m$ *treba posebno obraditi.*

Test $u' \geq v$ *se najlakše izvodi direktnim uspoređivanjem vodećih znamenki brojeva* u *i* v *(bez poziva algoritma* NCOMP*). Ova modifikacija komplicira algoritam, a ne donosi bitna poboljšanja. To pogotovo vrijedi u slučaju da množenje s* $q_i = 0$ *realiziramo efikasno – bez množenja.*

Zbog toga nećemo mijenjati algoritam NDIV0.

Algoritam sadrži četiri naredbe u kojima operacije treba izvesti na nizovima znamenki brojeva u pozicionom zapisu :

1. $u' \leftarrow \lfloor u/b^i \rfloor$ – dijeljenje potencijom baze, što se svodi na pomake (v. sljedeći algoritam).

2. $q_i \leftarrow \lfloor u'/v \rfloor$ – određivnje jednoznamenkastog kvocijenta. Ovu operaciju treba detaljno razraditi.

3. $r \leftarrow u' - q_i \cdot v$ – množenje broja znamenkom i oduzimanje brojeva (algoritmi NMULD, NSUB).

4. $u' \leftarrow r \cdot b + u_i$ – množenje bazom, što je pomak za jedno mjesto i dodavanje (kopiranje) znamenke u_i na najniže mjesto.

Sve operacije se izvode jednostavno i brzo, osim nalaženja jednoznamenkastog kvocijenta.

Uočimo da je algoritam najugodnije realizirati tako da brojeve r i u' spremamo na mjesto broja u, jer se u' formira iz znamenki broja u. Operacije 1. i 4. se tada mogu potpuno eliminirati. Algoritme za dijeljenje ćemo tako realizirati, s tim da na kraju postavljamo ostatak r i njegov stupanj.

Prije toga, formulirajmo algoritam za dijeljenje potencijom baze.

Algoritam 1.5.3. (NDIVB – **dijeljenje potencijom baze**)

Ulaz: Niz znamenki $(u_n, \ldots, u_0)$ broja u i potencija $p \in \mathbb{N}_0$, odabrane baze b.

Izlaz: Niz znamenki $(q_k, \ldots, q_0)$, $(r_l, \ldots, r_0)$ brojeva

$$q = \lfloor u/b^p \rfloor \ , \ r = u \bmod b^p$$

u bazi b, uz $k = \max\{ -1 , n - p \}$.

```
procedure NDIVB ( u : mpnat ;  p : integer ;  var  q , r : mpnat);

begin
    if  deg(u) ≥ p  then
        begin
        deg(q) ← deg(u) − p;
        deg(r) ← p − 1;
        while  (u_deg(r) = 0)  and  (deg(r) ≥ 0)  do
            deg(r) ← deg(r) − 1;
        if  u_deg(r) > 0  then
            for  i ← 0  to  deg(r)  do  r_i ← u_i
        else
            deg(r) ← −1;
        for  i ← 0  to  deg(q)  do  q_i ← u_{i+p};
        end
    else
        begin
        deg(q) ← −1;
        deg(r) ← deg(u);
        for  i ← 0  to  deg(r)  do  r_i ← u_i
        end
end;  { NDIVB }
```

Korektnost algoritma je očita, jer je

$$q = \sum_{i=0}^{n-p} u_{i+p}\, b^i \quad , \quad r = \sum_{i=0}^{p-1} u_i\, b^i \quad , \text{ za } \; n \geq p$$

i

$$q = 0 \quad , \quad r = u \quad , \text{ za } \; n < p \; .$$

U algoritmu prvo postavljamo stupanj broja r, provjeravanjem znamenki broja u, pa tek onda kopiramo znamenke. To omogućava da r spremimo na mjesto broja u, i tada se kopiranje znamenki može izbaciti.

I broj q je moguće spremiti na mjesto broja u (ali ne zajedno sa r), jer je smjer petlje propisno postavljen.

Prostorna složenost je

$$\mathrm{Compl}_S(\mathsf{NDIVB}) = \ell\,(u) + \ell\,(q) + \ell\,(r) + c \leq 2\,\ell\,(u) + c \; ,$$

sa $c \leq 5$, uz smanjenje za $\ell\,(q)$ odnosno $\ell\,(r)$, ako q ili r spremimo na mjesto broja u.

Aritmetička složenost je konstantna

$$\mathrm{Compl}_A(\mathsf{NDIVB}) = 1$$

zbog računanja stupnja.

Za vremensku složenost, u najgorem slučaju izlazi

$$\mathrm{Compl}_T(\mathsf{NDIVB}) \sim \ell\,(u)\ .$$

Ovaj algoritam, (kao i NMULB) se rijetko koristi kao zaseban algoritam.

Napomena 1.5.2. *Sve algoritme za dijeljenje ćemo realizirati tako da daju i kvocijent i ostatak. Sasvim jednostavno se dobivaju modificirani algoritmi koji vraćaju samo cjelobrojni kvocijent, pa te modifikacije nećemo posebno navoditi.*

1.5.4. Jednoznamenkasti kvocijent

Za potpunu realizaciju algoritma za dijeljenje, potrebno je konstruirati algoritam za nalaženje jednoznamenkastog kvocijenta – operacije $q_i \leftarrow \lfloor u'/v \rfloor$ u algoritmu NDIV0. Relacija (1.5.10) pokazuje da je rezultat uvijek jedna znamenka.

Zbog toga je broj u' najviše za 1 dulji od broja v, a može biti i kraći :

$$\deg_b(u') \leq \deg_b(v) + 1\ .$$

Ako broj v ima točno jednu znamenku, onda broj

$$q_i = \lfloor u'/v \rfloor$$

možemo izračunati korištenjem aritmetike računala. Tada je broj u' najviše dvoznamenkast, pa je

$$u' \leq b^2 - 1\ .$$

To znači da su brojevi u' i v egzaktno prikazivi u računalu, jer smo u propoziciji 1.4.3. postavili ograničenje $b^2 - 1 \leq maxint$ za realizaciju množenja. Primijetimo da je tada i ostatak r najviše jednoznamenkast.

Iz algoritma NDIV0 za jednoznamenkaste brojeve v, dobivamo sljedeći algoritam.

Algoritam 1.5.4. (NDIVD – **dijeljenje prirodog broja znamenkom**)

Ulaz: Niz znamenki $(u_n, \ldots, u_0)$ broja u i znamenka v_0 u odabranoj bazi b.

Izlaz: Niz znamenki $(q_k, \ldots, q_0)$ broja q i znamenka r_0

$$\begin{aligned} q &= \lfloor u/v_0 \rfloor \\ r_0 &= u \bmod v_0 \end{aligned}$$

u bazi b.

```
procedure NDIVD ( u : mpnat ;  v0 : digit ;  var  q : mpnat ;
                  var  r0 : digit);

begin
   if  v0 > 0  then
      begin
      if  deg(u) ≥ 0  then
         begin
         i ← deg(u);
         temp ← ui;
         deg(q) ← i;
         loop
            qi ← temp  div  v0;
            r0 ← temp  mod  v0   { = temp − qi · v }
         exit if  i ← 0;
            i ← i − 1;
            temp ← r0 · b + ui
         end;
         if  qdeg(q) = 0  then
            deg(q) ← deg(q) − 1
         end
      else   { u = 0 }
         begin
         deg(q) ← −1;
         r0 ← 0
         end
      end
   else
      ERROR
end;  { NDIVD }
```

Korektnost algoritma je direktna posljedica propozicije 1.5.1. , s tim da za znamenke dozvoljavamo vrijednost 0, pa nema provjere i postavljanja stupnja za r_0.

Za prostornu složenost, koristeći $\ell(q) \leq \ell(u)$, dobivamo :

$$\mathrm{Compl}_S(\mathsf{NDIVD}) = 2\,\ell(u) + c\ ,$$

sa $c \leq 6$, što možemo smanjiti na polovinu, jer q možemo spremiti na mjesto broja u.

Pomoćnu varijablu *temp* također možemo eliminirati i umjesto nje koristiti r_0.

Aritmetička složenost algoritma je

$$\mathrm{Compl}_A(\mathsf{NDIVD}) = \begin{cases} \ell\,(u) - 1 & \text{zbrajanja} \\ \ell\,(u) - 1 & \text{množenja} \\ 2\,\ell\,(u) & \text{dijeljenja} \end{cases} ,$$

što je ukupno

$$\mathrm{Compl}_A(\mathsf{NDIVD}) = 4\,\ell\,(u) - 2 \ . \tag{1.5.13}$$

Potrebno vrijeme je proporcionalno duljini broja u ili broja q

$$\mathrm{Compl}_T(\mathsf{NDIVD}) \sim \ell\,(u) \sim \ell\,(q) \ .$$

1.5.5. Opći slučaj jednoznamenkastog kvocijenta

Razmotrimo opći problem određivanja jednoznamenkastog kvocijenta

$$\begin{aligned} q &= \lfloor u/v \rfloor \\ 0 &\leq q < b \end{aligned} \quad , \tag{1.5.14}$$

bez ograničenja na $\deg_b(v)$.

U procesu dijeljenja "na ruke", broj q procjenjujemo na bazi prvih nekoliko znamenki brojeva u i v, koristeći iskustvo.

Ukoliko želimo izbjeći provjeru svih b mogućnosti za q (na pr. ponovljenim oduzimanjem broja v od broja u), moramo algoritmizirati procjenu i procijeniti njenu točnost.

Pretpostavimo da je $\deg_b(v) = m$ fiksan, jer je divizor u algoritmu NDIV0 uvijek isti broj v. Iz relacije (1.5.14) dobivamo da je

$$n = \deg_b(u) \leq m + 1 \ ,$$

s tim da se dividend u mijenja tokom rada algoritma NDIV0.

Za pojednostavljivanje zapisa, pretpostavimo da je broj u dan nizom znamenki

$$u = (u_{m+1} \ldots u_0)_b$$

tj.

$$u = \sum_{i=0}^{m+1} u_i\, b^i \ , \tag{1.5.15}$$

pri čemu vrijedi

$$0 \leq u_i < b \ , \ \ i = 0, \ldots, m+1 \ ,$$

ali zapis ne mora biti normaliziran u smislu stupnja, tj. dozvoljavamo da su vodeće znamenke jednake nuli.

Posebno može biti

$$u_{m+1} = 0 \ .$$

To nam znatno olakšava razmatranje, jer izbjegava stalno postavljanje i testiranje stupnja broja u' u algoritmu NDIV0.

Na početku tog algoritma lako je postaviti $u_{n+1} = 0$ i dalje raditi s tako postavljenom znamenkom.

Uz ovaj dogovor, uočimo da iz (1.5.14) slijedi

$$u/b < v \ \ , \ \ \lfloor u/b \rfloor < v \ , \tag{1.5.16}$$

jer je $q = \lfloor u/v \rfloor \leq b-1$ tj. $u/v < b$.

To u oznaci (1.5.15) znači

$$(u_{m+1}\, u_m \ldots u_1)_b < (v_m\, v_{m-1} \ldots v_0)_b \ . \tag{1.5.17}$$

Pretpostavljamo da je

$$b^2 - 1 \leq maxint \ ,$$

bez dodatnih ograničenja na veličinu baze b. To znači da možemo korektno zbrajati, oduzimati i množiti brojeve u pozicionom zapisu u bazi b. Uz to isto ograničenje treba naći jednoznamenkasti kvocijent q.

Napomena 1.5.3. *Uz to ograničenje, kvocijent q možemo naći egzaktno, direktno iz brojeva u i v (mašinskim dijeljenjem), samo ako je $m = 1$. To smo realizirali u algoritmu* NDIVD. *Za $m \geq 2$, brojevi u i v nisu egzaktno prikazivi u računalu ($v \geq b^2$), pa mašinsko dijeljenje u i v nije moguće.*

Ako želimo naći q, koristeći samo aritmetiku računala, u svakoj operaciji mogu učestvovati najviše po dvije znamenke brojeva u i v.

Najjednostavniji pristup ovom problemu je nalaženje približne vrijednosti $\widehat{q}$ za q na bazi vodećih znamenki brojeva u i v. Nakon toga nađemo ostatak

$$\widehat{r} = u - \widehat{q} \cdot v$$

i vršimo korekciju tog ostatka dodavanjem ili oduzimanjem broja v, sve dok ne postignemo da je

$$0 \leq \widehat{r} < v \ .$$

Usput korigiramo i kvocijent.

Ovo je tzv. metoda "podijeli i korigiraj" koju možemo algoritamski zapisati u sljedećm obliku.

Algoritam 1.5.5. (NDIVC – **jednoznamenkasti kvocijent**)

Ulaz: Nizovi znamenki $(u_{m+1}, \ldots, u_0)$, $(v_m, \ldots, v_0)$ brojeva u i v u odabranoj bazi b. Pri tome je v normaliziran, dok za znamenke broja u vrijedi

$$0 \leq u_i < b \ , \ i = 0, \ldots, m+1 \ .$$

To znači $\deg_b(v) = m$, $\deg_b(u) \leq m+1$. Pretpostavljamo, bez provjere, da vrijedi (1.5.16) :

$$u < v \cdot b \ .$$

Izlaz: Jednoznamenkasti kvocijent q

$$q = \lfloor u/v \rfloor \ ,$$

$0 \leq q < b$ i niz znamenki $(r_m, \ldots, r_0)$ ostatka

$$\begin{aligned} r &= u - q \cdot v \\ 0 &\leq r < v \quad , \end{aligned} \tag{1.5.18}$$

s tim da je

$$0 \leq r_i < b \ , \ i = 0, \ldots, m$$

i $\deg_b(r) \leq m$. Stupanj broja r ne postavljamo.

```
procedure NDIVC ( u , v : mpnat ; var q : digit ; var r : mpnat);

begin
    nađi približnu vrijednost q̂ kvocijenta ⌊u/v⌋ ;
    temp ← q̂ · v;
    if u ≥ temp then { q̂ ≤ q }
        begin
        r ← u − temp;
        if r ≥ v then { q̂ < q }
            repeat
                r ← r − v;
                q̂ ← q̂ + 1
            until r < v
        end
    else { q̂ > q }
        begin
        repeat
            temp ← temp − v;
            q̂ ← q̂ − 1
        until u ≥ temp;
        r ← u − temp
        end;
    q ← q̂
end; { NDIVC }
```

Ovdje je *temp* pomoćna varijabla tipa *mpnat*. Ona je potrebna ako želimo da algoritam radi uvijek s nenegativnim brojevima. U protivnom, ako dozvolimo i negativne brojeve, algoritam ima mnogo pregledniji zapis.

```
begin
    nađi približnu vrijednost q̂ kvocijenta ⌊u/v⌋ ;
    r ← u − q̂ · v;   { dozvoljeno je r < 0 }
    if r ≥ 0 then   { q̂ ≤ q }
        while r ≥ v do
            begin
            r ← r − v;
            q̂ ← q̂ + 1
            end
    else   { q̂ > q }
        repeat
            r ← r + v;
            q̂ ← q̂ − 1
        until r ≥ 0;
    q ← q̂
end;
```

Korektnost algoritma je očita, jer on postavlja q, r tako da vrijedi (1.5.18).

Složenost ovog algoritma direktno ovisi o tome koliko je $\widehat{q}$ dobra aproksimacija za q.

Propozicija 1.5.2.

Za aritmetičku složenost algoritma NDIVC *vrijedi*

$$\mathrm{Compl}_A(\mathsf{NDIVC}) = (c_1\,|q-\widehat{q}|+c_2)\,\ell\,(v)+c_3\,|q-\widehat{q}|+c_4\ , \tag{1.5.19}$$

gdje su c_1 , c_2 , c_3 , c_4 *konstante.*

Dokaz:

Analizirajmo varijantu koja koristi samo nenegativne brojeve, jer tada znamo složenosti pojedinih operacija. Za drugu varijantu vrijedi isti rezultat, korištenjem algoritma iz dijela 2.7. .

Procjenu kvocijenta $\widehat{q}$ određujemo na bazi nekog broja vodećih znamenki brojeva u i v, s najviše linearnim brojem aritmetičkih operacija

$$\mathrm{Compl}_A(\widehat{q}) = k_1\,\ell\,(v)+k_2\ .$$

Pokazat ćemo uskoro, da je $\widehat{q}$ moguće naći uz konstantan broj aritmetičkih operacija (iz fiksnog broja vodećih znamenki), pa je $k_1 = 0$.

Računanje $temp \leftarrow \widehat{q} \cdot v$ i $r \leftarrow u - temp$ (jer se obavlja u obje grane) zahtijeva

$$k_3\,\ell\,(v) + k_4$$

operacija. To slijedi iz relacije (1.4.13) za algoritam NMULD i relacije (1.3.11) za algoritam NSUB, jer je $\ell\,(temp) \leq \ell\,(v) + 1$.

Algoritam izvršava točno jednu od grana u if naredbi za korekciju kvocijenta i ostatka. Petlja unutar grane se izvršava točno $|q - \widehat{q}|$ puta i sadrži jedno oduzimanje i jedno uspoređivanje, neovisno o izboru grane.

Iz složenosti algoritama NSUB, NCOMP slijedi

$$\mathrm{Compl}_A(\text{korigiraj}) = (k_5\,\ell\,(v) + k_6)\,|q - \widehat{q}| \ .$$

Zbrajanjem složenosti je:

$$\mathrm{Compl}_A(\mathsf{NDIVC}) = (k_5\,|q - \widehat{q}| + k_1 + k_3)\,\ell\,(v) + k_6\,|q - \widehat{q}| + k_2 + k_4 \ .$$

Preciznijom analizom je $k_3 \leq 7$, $k_4 \leq 1$, $k_5 \leq 3$, $k_6 \leq 1$, što sa $k_1 = 0$ daje

$$\mathrm{Compl}_A(\mathsf{NDIVC}) \leq (3\,|q - \widehat{q}| + 7)\,\ell\,(v) + \ |q - \widehat{q}| + 1 + k_2 \ .$$

■

1.5.6. Procjena jednoznamenkastog kvocijenta

Preostaje razraditi algoritme za nalaženje približnog kvocijenta $\widehat{q}$ sa što boljim ocjenama za $|q - \widehat{q}|$.

Uočimo da je vrlo korisno naći takve $\widehat{q}$ da je predznak pogreške $q - \widehat{q}$ fiksan, neovisno o u i v. Ovisno o predznaku ove razlike, tada možemo potpuno eliminirati jednu od grana u algoritmu NDIVC.

Takve procjene ćemo posebno označiti :

$$\begin{array}{lll} \widehat{q}_U & - \text{ gornja ograda za } q \ , & \widehat{q}_U \geq q \quad \text{za} \quad \forall u \in \mathbb{N}_0 \ , \ \forall v \in \mathbb{N} \\ \widehat{q}_L & - \text{ donja ograda za } q \ , & \widehat{q}_L \leq q \quad \text{za} \quad \forall u \in \mathbb{N}_0 \ , \ \forall v \in \mathbb{N} \ . \end{array}$$

Posebno je ugodno naći donju ogradu $\widehat{q}_L$, jer su tada svi brojevi nenegativni i u drugoj varijanti algoritma, bez pomoćne varijable *temp* (otpada else grana).

Napomena 1.5.4. *Tako upotpunjeni algoritam* NDIVC *treba ugraditi direktno u algoritam* NDIV0*, umjesto naredbi*

$$\begin{aligned} q_i &\leftarrow \lfloor u'/v \rfloor \\ r &\leftarrow u' - q_i \cdot v \ . \end{aligned}$$

Približni kvocijent $\widehat{q}$ određujemo na bazi nekog broja vodećih znamenki brojeva u i v.

Definicija 1.5.1.

Približnu vrijednost $\widehat{q}$ kvocijenta $q = \lfloor u/v \rfloor$, određenu tako da ovisi o prvih s znamenki broja u i prvih t znamenki broja v, zovemo $(s\ ,\ t)$ **procjena** *za q.*

To znači da $\widehat{q}$ ovisi o

$$\begin{matrix} u_{m+1} & , & u_m & , & \dots & , & u_{m+2-s} \\ & & v_m & , & \dots & , & v_{m+1-t} \end{matrix} \quad ,$$

uz oznake (1.5.15).

Napomena 1.5.5. *Očito je $\widehat{q} = q$ za svaki u, v, ako i samo ako je $\widehat{q}$ određen na osnovu svih znamenaka brojeva u i v. To znači da je $\widehat{q}$ $(m+1\ ,\ m)$ procjena za q.*

Čim je $s < m+1$ ili $t < m$, onda je moguće naći primjere da je

$$|q - \widehat{q}| \geq 1 \ . \tag{1.5.20}$$

Dovoljno je promijeniti zadnju znamenku u_0 ili v_0, pa da se q promijeni za barem 1, a $\widehat{q}$ se neće promijeniti.

Obično se $(s\ ,\ t)$ procjena $\widehat{q}$ formira tako da je

$$\widehat{q} = \frac{U_s}{V_t} \cdot b^{t-s+1} \ , \tag{1.5.21}$$

gdje je

$$\begin{matrix} & U_s = \lfloor u/b^{m+2-s} \rfloor & \text{ili} & U_s = \lfloor u/b^{m+2-s} \rfloor + 1 \\ \text{i} & & & \\ & V_t = \lfloor v/b^{m+1-t} \rfloor & \text{ili} & V_t = \lfloor v/b^{m+1-t} \rfloor + 1 \end{matrix} \quad .$$

To znači da se U_s formira iz vodećih s znamenki broja u, uz zaokruživanje "na dolje" – odbacivanjem ostatka, ili " na gore" – dodavanjem 1. Analogno se V_t formira iz v. Ovakav način nalaženja $\widehat{q}$ znatno olakšava ocjenu greške.

Metoda se može još poboljšati ako U_s, V_t formiramo zaokruživanjem na bazi zadnje odbačene znamenke. Tada je $\widehat{q}$, zapravo $(s+1\ ,\ t+1)$ procjena za q (odnosno $(s+1\ ,\ t)$ ili $(s\ ,\ t+1)$ procjena, ako samo U_s ili samo V_t formiramo na poboljšani način).

Najčešće se takve procjene koriste sa

$$s = t + 1 \ ,$$

jer otpada potencija baze u (1.5.21).

U praksi se gotovo uvijek koriste $(2\ ,\ 1)$ procjene za q, jer se one mogu direktno izračunati aritmetikom računala. Katkada se takva procjena, provjerom popravlja u $(3\ ,\ 2)$ procjenu, što ćemo uskoro pokazati.

Sve procjene su bazirane na sljedeća dva algoritma za $(2\ ,\ 1)$ procjene.

Algoritam 1.5.6. (Cox i Luther (1961.), Stein (1964.))

$$\widehat{q}_L = \left\lfloor \frac{u_{m+1} \cdot b + u_m}{v_m + 1} \right\rfloor \ . \tag{1.5.22}$$

Za U_2 koristimo zaokruživanje odbacivanjem, a za V_1 dodavanjem 1.

Algoritam 1.5.7. (Pope i Stein (1960.))

$$\widehat{q}_U = \min \left\{ \left\lfloor \frac{u_{m+1} \cdot b + u_m}{v_m} \right\rfloor \ , \ b-1 \right\} \ . \tag{1.5.23}$$

Za U_2 i V_1 koristimo zaokruživanje odbacivanjem, dok minimum osigurava da je $\widehat{q}_U$ jednoznamenkast.

Složenosti oba algoritma su konstantne.

Za dalju analizu je korisna sljedeća tvrdnja.

Lema 1.5.1.

Neka su $s \in \mathbb{N}_0$, $t \in \mathbb{N}$ *bilo koji brojevi. Tada vrijedi*

$$\frac{s+1}{t} - 1 \leq \left\lfloor \frac{s}{t} \right\rfloor \leq \frac{s}{t} \ . \tag{1.5.24}$$

Dokaz:

Po Euklidovom teoremu je

$$s = k \cdot t + l \ ,$$

sa $k = \lfloor s/t \rfloor \geq 0$ i $0 \leq l \leq t-1$. Odavde je

$$\left\lfloor \frac{s}{t} \right\rfloor = k = \frac{s-l}{t} \geq \frac{s-(t-1)}{t} = \frac{s+1}{t} - 1 \ .$$

Druga nejednakost u (1.5.24) je očita, zbog $l \geq 0$. ■

Pokažimo prvo da su oznake $\widehat{q}_L$, $\widehat{q}_U$ za ove procjene opravdane.

Propozicija 1.5.3.

Za bilo koje $u \in \mathbb{N}_0$, $v \in \mathbb{N}$ *, takve da je*

$$q = \lfloor u/v \rfloor < b$$

vrijedi :

$$\widehat{q}_L \leq q \leq \widehat{q}_U \ , \tag{1.5.25}$$

tj. $\widehat{q}_L$ *je donja, a* $\widehat{q}_U$ *gornja ograda za* q*.*

Dokaz:

Prvo dokažimo $\widehat{q}_L \leq q$. Kako je očito

$$u_{m+1} \cdot b + u_m = \lfloor u/b^m \rfloor \leq u/b^m$$

i

$$v_m + 1 = \lfloor v/b^m \rfloor + 1 > v/b^m \ ,$$

to je

$$\frac{u_{m+1} \cdot b + u_m}{v_m + 1} < \frac{u/b^m}{v/b^m} = \frac{u}{v} \ .$$

Odavde, zbog monotonosti funkcije $\lfloor \ \rfloor$, izlazi

$$\widehat{q}_L = \left\lfloor \frac{u_{m+1} \cdot b + u_m}{v_m + 1} \right\rfloor \leq \left\lfloor \frac{u}{v} \right\rfloor = q \ .$$

Dokažimo da je $q \leq \widehat{q}_U$.

Ako je

$$\left\lfloor \frac{u_{m+1} \cdot b + u_m}{v_m} \right\rfloor \geq b - 1 \ ,$$

onda je, po definiciji, $\widehat{q}_U = b - 1$. No, zbog $q = \lfloor u/v \rfloor < b$, jer je q cijeli broj, slijedi $q \leq b - 1 = \widehat{q}_U$, pa tada vrijedi (1.5.25).

Preostaje razmotriti slučaj

$$\left\lfloor \frac{u_{m+1} \cdot b + u_m}{v_m} \right\rfloor < b - 1 \ . \tag{1.5.26}$$

Tada je svakako, po (1.5.23),

$$\widehat{q}_U = \left\lfloor \frac{u_{m+1} \cdot b + u_m}{v_m} \right\rfloor \ .$$

Iz leme 1.5.1. dobivamo

$$\widehat{q}_U \geq \frac{u_{m+1} \cdot b + u_m + 1}{v_m} - 1 \ .$$

Kako je

$$\begin{aligned} u_{m+1} \cdot b + u_m + 1 \; &> \; u/b^m \\ v_m \; &\leq \; v/b^m \; , \end{aligned}$$

to je

$$\widehat{q}_U > \frac{u/b^m}{v/b^m} - 1 = \frac{u}{v} - 1 \geq \left\lfloor \frac{u}{v} \right\rfloor - 1 \; .$$

To znači da je

$$\widehat{q}_U > q - 1 \; .$$

Pošto su $\widehat{q}_U$ i q cijeli brojevi, onda je

$$\widehat{q}_U \geq q \; .$$

■

Uočimo da pretpostavku (1.5.26) nismo koristili, što pokazuje da je uvijek

$$\left\lfloor \frac{u_{m+1} \cdot b + u_m}{v_m} \right\rfloor \geq q \; , \tag{1.5.27}$$

ali izraz na lijevoj strani može biti veći ili jednak b, čak i ako je $u/v < b$.

Napomena 1.5.6. *Donja ograda $\widehat{q}_L$ je prirodno definirana, jer brojnik smanjujemo u odnosu na u/b^m , a nazivnik povećavamo u odnosu na v/b^m . Analogna gornja ograda je*

$$\widehat{q}_{UU} = \left\lfloor \frac{u_{m+1} \cdot b + u_m + 1}{v_m} \right\rfloor$$

u kojoj brojnik povećavamo obzirom na u/b^m , a nazivnik smanjujemo obzirom na v/b^m .

Očito vrijedi

$$\widehat{q}_{UU} \geq q \; ,$$

međutim, $\widehat{q}_{UU}$ može biti dvoznamenkasta. Tada treba definirati $\widehat{q}_{UU} = b - 1$, kao za $\widehat{q}_U$. No i tada postoji mogućnost da je brojnik u $\widehat{q}_{UU}$ jednak b^2, tj. neprikaziv.

Kako je očito i

$$\widehat{q}_{UU} \geq \widehat{q}_U \; ,$$

to je $\widehat{q}_u$ bolja gornja ograda od $\widehat{q}_{UU}$. Zbog toga se koristi $\widehat{q}_U$, iako je manje prirodno definirana.

Sljedeća tvrdnja iz [Singer, Rogina, 1986.] daje ocjenu razlike tih dviju procjena.

Teorem 1.5.1.

Neka su $u \in \mathbb{N}_0$, $v \in \mathbb{N}$ *bilo koji brojevi. Tada je*

$$\widehat{q}_U - \widehat{q}_L \leq 1 + \frac{b-2}{v_m+1} \ . \tag{1.5.28}$$

Dokaz:

Razmotrimo dva slučaja, ovisno o vrijednosti $\widehat{q}_U$.

Ako je $\lfloor (u_{m+1} \cdot b + u_m)/v_m \rfloor \geq b-1$, onda je

$$u_{m+1} \cdot b + u_m \geq (b-1)\, v_m \ , \tag{1.5.29}$$

i po definiciji je $\widehat{q}_U = b-1$. Redom je

$$\begin{aligned} \widehat{q}_U - \widehat{q}_L &= b - 1 - \left\lfloor \frac{u_{m+1} \cdot b + u_m}{v_m + 1} \right\rfloor \leq \text{(Lema 1.5.1.)} \\ &= b - 1 - \left(\frac{u_{m+1} \cdot b + u_m + 1}{v_m + 1} - 1 \right) \leq (1.5.29) \\ &\leq b - \frac{(b-1)\, v_m + 1}{v_m + 1} = 1 + \frac{b-2}{v_m+1} \ . \end{aligned}$$

Ako je $\lfloor (u_{m+1} \cdot b + u_m)/v_m \rfloor < b-1$, onda je

$$u_{m+1} \cdot b + u_m < (b-1)\, v_m \ , \tag{1.5.30}$$

i

$$\widehat{q}_U = \left\lfloor \frac{u_{m+1} \cdot b + u_m}{v_m} \right\rfloor \ .$$

Tada je analogno :

$$\begin{aligned} \widehat{q}_U - \widehat{q}_L &\leq \frac{u_{m+1} \cdot b + u_m}{v_m} - \left(\frac{u_{m+1} \cdot b + u_m + 1}{v_m + 1} - 1 \right) \\ &= \frac{u_{m+1} \cdot b + u_m}{v_m\,(v_m+1)} + 1 - \frac{1}{v_m+1} < (1.5.30) \\ &< \frac{(b-1)\, v_m}{v_m\,(v_m+1)} + 1 - \frac{1}{v_m+1} = 1 + \frac{b-2}{v_m+1} \ . \end{aligned}$$

■

Relacija (1.5.28) pokazuje da ocjena razlike procjena $\widehat{q}_U$ i $\widehat{q}_L$, a to znači i ocjena pogreške svake od tih procjena, ovisi fundamentalno o vodećoj znamenci v_m divizora.

1.5.7. Normalizacija divizora

Da dobijemo što bolju ocjenu pogreške, treba osigurati što je moguće veću vrijednost za v_m.

Zbog toga, prije dijeljenja brojeva u i v u algoritmu NDIV0 vršimo tzv. normalizaciju divizora v. Brojeve u i v množimo povoljno odabranim normalizacionim faktorom d. Taj postupak je dovoljno obaviti samo jednom, jer je divizor v jedini divizor u algoritmu.

Dakle, na početak algoritma NDIV0 treba dodati naredbe

$$u \leftarrow u \cdot d$$
$$v \leftarrow v \cdot d$$

što ne mijenaja kvocijent, a na kraju algoritma treba ostatak podijeliti s d

$$r \leftarrow r/d \ .$$

Kvocijent r/d je cijeli broj, jer je r višekratnik broja d, prije dijeljenja.

Poželjno je da algoritam za nalaženje broja d bude što jednostavniji i da d ima što manji broj znamenki, po mogućnosti samo jednu.

Takva normalizacija je moguća.

Teorem 1.5.2.

Neka je $v \in \mathbb{N}$ *bilo koji broj dan pozicionim zapisom*

$$v = (v_m \dots v_0)_b$$

u bazi b.

Ako definiramo

$$d_1 = \left\lfloor \frac{b}{v_m + 1} \right\rfloor \qquad i \qquad v' = d_1 \cdot v \tag{1.5.31}$$

onda je $\deg_b(v') = \deg_b(v) = m$ *i za vodeću znamenku broja* v' *vrijedi*

$$\left\lfloor \frac{b}{2} \right\rfloor \le v'_m \le b - 1 \ . \tag{1.5.32}$$

Dokaz:

Kako je $v_m + 1 > v/b^m$, to je

$$v' = d_1 \cdot v \le \frac{b}{v_m + 1} \cdot v < \frac{v \cdot b}{v/b^m} = b^{m+1} \ ,$$

što pokazuje da je $\deg_b(v') \leq m$.

Odavde direktno slijedi i

$$v'_m = \lfloor v'/b^m \rfloor < b \ ,$$

pa je $v'_m \leq b-1$.

Zbog $v_m \leq b-1$ je $v_m + 1 \leq b$, pa je uvijek

$$d_1 \geq 1 \ ,$$

što daje $v' \geq v$, pa je $\deg_b(v') \geq m$. Time je pokazano da v i v' imaju isti stupanj, pa iz $v' \geq v$ slijedi i

$$v'_m \geq v_m \ , \tag{1.5.33}$$

što znači da normalizacija divizora zaista povećava vodeću znamenku.

Preostaje dokazati da je $v'_m \geq \lfloor b/2 \rfloor$.

Ako je $v_m \geq \lfloor b/2 \rfloor$ (tj. v je već normalizirani divizor), onda je iz (1.5.33) i

$$v'_m \geq \lfloor b/2 \rfloor \ .$$

Primijetimo da je tada $d_1 = 1$, jer je $v_m + 1 \geq \lfloor b/2 \rfloor + 1 > b/2$.

Pretpostavimo da je $1 \leq v_m < \lfloor b/2 \rfloor$. Zbog $v_m = \lfloor v/b^m \rfloor \leq v/b^m$ vrijedi

$$v'_m = \lfloor v'/b^m \rfloor = \lfloor d_1 \cdot v/b^m \rfloor \geq \lfloor d_1 \cdot v_m \rfloor = d_1 \cdot v_m$$

jer je $d_1 \cdot v_m$ cijeli broj. To pokazuje da je dovoljno dokazati

$$d_1 \cdot v_m \geq \lfloor b/2 \rfloor \ . \tag{1.5.34}$$

Očito je :

$$d_1 \cdot v_m = \left\lfloor \frac{b}{v_m + 1} \right\rfloor \cdot v_m > \left(\frac{b}{v_m + 1} - 1 \right) \cdot v_m \ .$$

Preuređenjem izraza na desnoj strani, ovu relaciju možemo zapisati u obliku :

$$d_1 \cdot v_m > \frac{b}{2} - 1 + \frac{(v_m - 1)\,(b/2 - 1 - v_m)}{v_m + 1} \ . \tag{1.5.35}$$

Zbog $v_m < \lfloor b/2 \rfloor$, jer je v_m cijeli broj, slijedi

$$v_m + 1 \leq \lfloor b/2 \rfloor < b/2$$

pa je

$$\frac{b}{2} - 1 - v_m > 0 \ .$$

Očito je $v_m - 1 \geq 0$, pa je, množenjem

$$(v_m - 1)\left(\frac{b}{2} - 1 - v_m\right) \geq 0 \ ,$$

što pokazuje da je zadnji član u (1.5.35) nenegativan. Zbog toga je :

$$d_1 \cdot v_m > \frac{b}{2} - 1 \geq \left\lfloor \frac{b}{2} \right\rfloor - 1 \ .$$

Pošto je $d_1 \cdot v_m$ cijeli broj, to je

$$d_1 \cdot v_m \geq \left\lfloor \frac{b}{2} \right\rfloor \ .$$

Time je relacija (1.5.34) dokazana. ■

Primijetimo da je normalizacioni faktor

$$d_1 = \left\lfloor \frac{b}{v_m + 1} \right\rfloor$$

najveći mogući jednoznamenkasti broj, određen samo iz vodeće znamenke v_m divizora v, takav da je

$$v' = d_1 \cdot v < b^{m+1} \ ,$$

tj. takav da je $d_1 \cdot v$ istog stupnja kao i v.

Definicija 1.5.2.

Divizor v koji zadovoljava uvjet

$$v_m \geq \lfloor b/2 \rfloor$$

zovemo **normalizirani divizor**.

Napomena 1.5.7. *U dokazu teorema 1.5.2. pokazali smo da je*

$$v_m \geq \lfloor b/2 \rfloor \implies d_1 = 1 \ ,$$

tj. normalizacija ne mijenja već normalizirani divizor. To znači da normalizaciju ima smisla primijeniti najviše jednom i da je

$$v_m \geq \lfloor b/2 \rfloor$$

najbolja donja ograda vodeće znamenke broja v uz normalizaciju (1.5.31). Ako želimo bolju donju ogradu, onda treba koristiti drugačije normalizacije.

Za normalizirani divizor v dobivamo mnogo bolju ocjenu pogreške u teoremu 1.5.1. .

Teorem 1.5.3.

Ako je v normalizirani divizor, onda je

$$\widehat{q}_U - \widehat{q}_L \leq \begin{cases} 2 & , \quad za \quad b \geq 5 \\ 1 & , \quad za \quad b \leq 4 \ . \end{cases} \tag{1.5.36}$$

Dokaz:

Zbog $v_m \geq \lfloor b/2 \rfloor$ izlazi

$$v_m + 1 \geq \frac{b+1}{2} \ ,$$

pa u teoremu 1.5.1. dobivamo :

$$\widehat{q}_U - \widehat{q}_L \leq 1 + 2 \cdot \frac{b-2}{b+1} = 3 - \frac{6}{b+1} \ .$$

Razlika $\widehat{q}_u - \widehat{q}_L$ je cijeli broj, što direktno daje (1.5.36). ■

Odavde direktno izlazi sljedeći rezultat [Knuth, 1981.].

Korolar 1.5.1.

Za dijeljenje normaliziranim divizorom vrijedi

$$\begin{array}{rcl} q & \leq \widehat{q}_U \leq & q+2 \\ q-2 & \leq \widehat{q}_L \leq & q \end{array} \ . \tag{1.5.37}$$

Dokaz:

Direktno iz teorema 1.5.2. i teorema 1.5.3. . ■

Napomena 1.5.7. pokazuje da su to i najbolje moguće ocjene pogreške uz normalizaciju faktorom d_1. Lako je konstruirati primjere u kojima je

$$|q - \widehat{q}| = 2$$

za jednu od procjena $\widehat{q} = \widehat{q}_U$ ili $\widehat{q} = \widehat{q}_L$. Uočimo da je tada druga procjena sigurno korektna.

To znači da su potrebna najviše 2 oduzimanja za korekciju kvocijenta i ostatka u algoritmu NDIVC.

Usporedimo li rezultat iz korolara 1.5.1. s onim iz napomene 1.5.5. , zaključujemo da najjednostavnijim procjenama $\widehat{q}_U$, $\widehat{q}_L$, uz najjednostavniju normalizaciju postižemo točnost koja je za samo 1 lošija od optimalne.

Napomena 1.5.8. *Zanimljivo je analizirati koliko često se postiže najveća pogreška*

$$|q - \widehat{q}| = 2$$

za procjene $\widehat{q}_U$, $\widehat{q}_L$.

Collins i Musser su pokazali da su vjerojatnosti za pojedine vrijednosti pogreške procjene $\widehat{q}_U$, *uz normalizacioni divizor, dane sa :*

$$\begin{aligned} \mathrm{P}\left(\widehat{q}_U = q\right) &\approx 0.67 \\ \mathrm{P}\left(\widehat{q}_U = q+1\right) &\approx 0.32 \\ \mathrm{P}\left(\widehat{q}_U = q+2\right) &\approx 0.01 \end{aligned} \tag{1.5.38}$$

[Collins, Mignotte, Winkler, 1982.]. Slično vrijedi i za $\widehat{q}_L$, *tj. pogreška* 2 *se javlja vrlo rijetko.*

To pokazuje da su procjene $\widehat{q}_U$, $\widehat{q}_L$ *vrlo efikasne za normalizirane divizore* v.

Uz nešto kompliciraniju normalizaciju, ocjena pogreške iz teorema 1.5.3. se može poboljšati do optimalne.

Korolar 1.5.2.

Ako su $u \in \mathbb{N}_0$, $v \in \mathbb{N}$ *bilo koji brojevi i ako je*

$$v_m \geq b - 2$$

onda je :

$$\widehat{q}_U - \widehat{q}_L \leq 1 \tag{1.5.39}$$

i

$$\begin{aligned} q \;\leq\; \widehat{q}_U \;&\leq\; q+1 \\ q-1 \;\leq\; \widehat{q}_L \;&\leq\; q \quad . \end{aligned} \tag{1.5.40}$$

Dokaz:

Direktni iz teorema 1.5.2. i teorema 1.5.3. . ■

To znači da je barem jedna procjena točna, ali ne znamo koja, ako su one različite. Ako su procjene jednake, onda su one uvijek točne (bez obzira na normalizaciju), zbog $\widehat{q}_L \leq q \leq \widehat{q}_U$.

Preostaje opisati normalizaciju kojom se postiže $v_m \geq b - 2$.

Teorem 1.5.4.

Neka je $v \in \mathbb{N}$ *bilo koji prirodni broj dan pozicionim zapisom*

$$v = (v_m \ldots v_0)_b$$

u bazi b. Pretpostavljamo da je v normaliziran

$$v_m \geq \lfloor b/2 \rfloor \ .$$

Definiramo

$$d_2(v) = \left\lfloor \frac{b^2 - 1}{v_m + 1} \right\rfloor \tag{1.5.41}$$

i

$$\begin{aligned} v' &= d_2(v) \cdot v \\ v'' &= d_2(v') \cdot v' \ . \end{aligned}$$

Onda je

$$\begin{aligned} \deg(v') &= m + 1 \\ \deg(v'') &= m + 2 \end{aligned}$$

i za vodeću znamenku broja v'' vrijedi

$$v''_{m+2} \geq b - 2 \ .$$

■

Ovaj rezultat nećemo dokazati.

Napomena 1.5.9. *Teorem 1.5.4. je generalizacija teorema 6. iz [Singer, Rogina, 1986.] i dokazuje se na isti način. U tom radu je $d_2(v)$ definiran sa*

$$d_2(v) = \left\lfloor \frac{b^2}{v_m + 1} \right\rfloor \ . \tag{1.5.42}$$

Faktor $d_2(v)$ iz (1.5.41) je nešto manji od ovog, ali je direktno izračunljiv aritmetikom računala. Ako je i $b^2 \leq maxint$, onda je bolje koristiti $d_2(v)$ iz (1.5.42).

Primijetimo da je normalizacioni faktor $d_2(v)$ uvijek dvoznamenkast, pa naloženje v'' zahtijeva oko 5 puta više aritmetičkih operacija od normalizacije samo faktorom d_1 iz teorema 1.5.2. . Zbog toga se ovaj postupak obično ne koristi.

Napomena 1.5.10. *Ako ipak koristimo dodatne normalizacije divizora, onda je idealno osigurati*

$$v_m = b - 1 \ ,$$

ako koristimo procjenu $\widehat{q}_L$, odnosno

$$v_m = 1 \quad , \quad v_{m-1} = 0 \quad ,$$

ako koristimo $\widehat{q}_U$. Tada se procjene $\widehat{q}_U$, $\widehat{q}_L$ dobivaju bez dijeljenja, direktno iz vodećih znamenki broja u, a pogreška je najviše 1. Niz takvih normalizacija opisan je u [Nandi, Krishnamurthy, 1967.] i [Knuth, 1981.].

Na bazi (2 , 1) procjena $\widehat{q}_L$, $\widehat{q}_L$, možemo formirati (2 , 2) procjenu, koristeći zaokruživanje, ovisno o vrijednosti znamenke v_{m-1}.

Algoritam 1.5.8. (Stein (1964.), Krishnamurthy (1965.))

$$\widehat{q}_{LU} = \begin{cases} \widehat{q}_L & , \quad \text{za} \quad v_{m-1} \geq b/2 \\ \widehat{q}_U & , \quad \text{za} \quad v_{m-1} < b/2 \ . \end{cases} \tag{1.5.43}$$

Za $\widehat{q}_{LU}$, naravno vrijede ocjene za $\widehat{q}_L$, odnosno za $\widehat{q}_U$. Mana ove procjene je u tome što pogreška

$$q - \widehat{q}_{LU}$$

može biti i pozitivna i negativna, ovisno o u i v. Prednost ove procjene je u tome što se može izbjeći normalizacija divizora u oko 50 % slučajeva, a da greška ne prelazi 1. Detaljna analiza ovog algoritma dana je u [Krishnamurthy, Nandi, 1967.].

U praksi se najčešće koristi sljedeći algoritam, baziran na $\widehat{q}_U$.

Algoritam 1.5.9. (Knuth, 1969.)

begin
  $\widehat{q} \leftarrow \widehat{q}_U$;
  if $m \geq 1$ **then**
    while $v_{m-1} \cdot \widehat{q} > (u_{m+1} \cdot b + u_m - \widehat{q} \cdot v_m) \cdot b + u_{m-1}$ **do**
      $\widehat{q} \leftarrow \widehat{q} - 1$;
  $\widehat{q}_{UK} \leftarrow \widehat{q}$
end;

Test u while naredbi je formuliran tako da je egzaktno izvodiv u aritmetici računala, a ekvivalentan je sa

$$(v_m\, b + v_{m-1})\, \widehat{q} > u_{m+1}\, b^2 + u_m\, b + u_{m-1} \ . \tag{1.5.44}$$

To pokazuje da algoritam korigira procjenu $\widehat{q}_U$ sve dok ne dobije

$$\widehat{q}_{UK} \leq \frac{u_{m+1}\, b^2 + u_m\, b + u_{m-1}}{v_m\, b + v_{m-1}} \ ,$$

što dokazuje da je

$$\widehat{q}_{UK} \leq \left\lfloor \frac{u_{m+1}\, b^2 + u_m\, b + u_{m-1}}{v_m\, b + v_{m-1}} \right\rfloor \ ,$$

tj. $\widehat{q}_{UK}$ je $(3\ ,\ 2)$ procjena za q. Algoritam 1.5.9. samo pokazuje kako se $\widehat{q}_{UK}$ može naći korištenjem aritmetike računala.

Može se pokazati [Knuth, 1981.] da vrijedi sljedeća ocjena.

Teorem 1.5.5.

Neka su $u \in \mathbb{N}_0$, $v \in \mathbb{N}$ *bilo koji brojevi i neka je divizor* v *normaliziran*

$$v_m \geq \lfloor b/2 \rfloor \ .$$

Onda je :

$$q \leq \widehat{q}_{UK} \leq q+1$$

i

$$\widehat{q}_{UK} > q \implies u \bmod v \geq \left(1 - \frac{2}{b}\right) v \ .$$

■

Druga tvrdnja pokazuje da je vjerojatnost da je $\widehat{q}_{UK} - q = 1$ približno jednaka $2/b$, tj. $\widehat{q}_{UK}$ je vrlo rijetko pogrešna procjena.

Detaljna analiza slučajeva u kojima je $\widehat{q}_{UK}$ pogrešna procjena, dana je u [Regener, 1984.]. Za normalizirane divizore je $\widehat{q}_U \leq q+2$, pa algoritam 1.5.9. korigira $\widehat{q}$ najviše 2 puta.

Sličan algoritam, ali znatno kompliciraniji omogućava korekciju $\widehat{q}_L$ u odgovarajuću $(3\ ,\ 2)$ procjenu sličnih svojstava.

1.5.8. Algoritam za dijeljenje brojeva

Zbog jednostavnosti, algoritam za dijeljenje prirodnih brojeva realiziramo korištenjem procjene $\widehat{q}_L$.

Algoritam 1.5.10. (NDIV – **dijeljenje prirodnih brojeva s ostatkom**)

Ulaz: Nizovi znamenki $(u_n, \ldots, u_0)$, $(v_m, \ldots, v_0)$ brojeva u i v u odabranoj bazi b. Pretpostavljamo da je $v > 0$. U protivnom, algoritam završava greškom.

Izlaz: Nizovi znamenki $(q_k, \ldots, q_0)$, $(r_l, \ldots, r_0)$ brojeva

$$\begin{aligned} q &= \lfloor u/v \rfloor \quad , \quad k = \deg_b(q) \\ r &= u \bmod v \quad , \quad l = \deg_b(r) \end{aligned}$$

u bazi b.

Algoritam je dan u skraćenom zapisu, s tim da su eksplicitno dani nizovi znamenki na koje se odnose operacije.

```
procedure NDIV ( u , v : mpnat ; var q , r : mpnat);

begin
  if deg(v) ≥ 0 then
    if deg(u) ≥ deg(v) then
      begin
      vlead1 ← v_deg(v) + 1;
        { normalizacija divizora }
      if v_deg(v) < b div 2 then
        begin
        d ← b div vlead1;
        (v_m, ..., v_0) ← d · (v_m, ..., v_0);
        (u_{n+1}, u_n, ..., u_0) ← d · (u_{n+1}, u_n, ..., u_0)
        end
      else
        u_{deg(u)+1} ← 0;
          { dijeljenje }
      deg(q) ← deg(u) − deg(v);
      for i ← deg(q) downto 0 do
        begin
        nu ← deg(v) + i;
          { u' = (u_{i+m+1}, u_{i+m}, ..., u_i) }
        q_i ← (u_{nu+1} · b + u_nu) div vlead1;   { q̂_L }
          { r ← u' − q_i · v  ,  r = (u_{i+m+1}, u_{i+m}, ..., u_i) }
        if q_i > 0 then
          (u_{i+m+1}, ..., u_i) ← (u_{i+m+1}, ..., u_i) − q_i (v_m, ..., v_0);
            { provjeri ostatak i korigiraj }
        loop
          ok ← ((u_{i+m+1}, ..., u_i) < (v_m, ..., v_0))
        exit if ok;
            { korekcija q_i ← q_i − 1 , r ← r − v }
          q_i ← q_i − 1;
          (u_{i+m+1}, ..., u_i) ← (u_{i+m+1}, ..., u_i) − (v_m, ..., v_0)
        end      { loop }
        end;     { for i }
      if q_deg(q) = 0 then
        deg(q) ← deg(q) − 1;
          { korigiraj ostatak r ← r/d i stupanj }
      if d > 1 then
        (r_m, ..., r_0) ← (u_{m+1}, ..., u_0)/d
      else
        (r_m, ..., r_0) ← (u_m, ..., u_0);
```

```
            deg(r) ← deg(w);
            while (r_deg(r) = 0) and (deg(r) > 0) do
                deg(r) ← deg(r) − 1;
            if r_deg(r) = 0 then
                deg(r) ← −1
            end    { deg(u) ≥ deg(v) }
        else
            begin
            deg(q) ← −1;
            deg(r) ← deg(v);
            (r_m, ..., r_0) ← (v_m, ..., v_0)
            end
    else
        ERROR
end; { NDIV }
```

Korektnost ovog algoritma je direktna posljedica korektnosti algoritma NDIV0 i analize metode "podijeli i korigiraj".

Analizirajmo složenost algoritma.

Algoritam je realiziran tako da se ostaci r formiraju direktno na mjestu ulaznog broja u. Zbog toga, za prostornu složenost vrijedi

$$\mathrm{Compl}_S(\mathsf{NDIV}) = 2\,\ell\,(u) + \ell\,(v) + c\ , \tag{1.5.45}$$

sa $c \leq 10$. Pri tome je uračunato da smo broj u produljili za jedno mjesto i da za broj q koristimo $\ell\,(u) - \ell\,(v) + 1$ mjesta, a za r koristimo $\ell\,(v)$ mjesta.

Algoritam se može preformulirati tako da ne koristi produljenje u, a za brojeve q i r koristi točno $\ell\,(q)$ odnosno $\ell\,(r)$ znamenki. Tada je

$$\mathrm{Compl}_S(\mathsf{NDIV}) = \ell\,(u) + \ell\,(v) + \ell\,(q) + \ell\,(r) + c\ .$$

Naravno, kvocijent i ostatak mogu se spremiti na mjesto broja u (q u gornji, a r u donji dio broja).

Aritmetička složenost je zbroj aritmetičkih složenosti pojedinih faza algoritma :

$$\begin{aligned} \mathrm{Compl}_A(\mathsf{NDIV}) \;&=\; \mathrm{Compl}_A(\text{normaliziraj}) \\ &+\; \mathrm{Compl}_A(\text{podijeli}) \\ &+\; \mathrm{Compl}_A(\text{ostatak})\ . \end{aligned} \tag{1.5.46}$$

Složenost normalizacije divizora i nalaženja pravog ostatka ovisi o tome da li je divizor v već normaliziran na ulazu ili ne. Ako je već normaliziran, onda je :

$$\begin{aligned} \text{b} - \mathrm{Compl}_A(\text{normaliziraj}) &= 0 \\ \text{b} - \mathrm{Compl}_A(\text{ostatak}) &= 0 \qquad . \end{aligned}$$

U protivnom, uz normalizaciju je

$$\text{w} - \text{Compl}_A(\text{normaliziraj}) = \begin{cases} \ell(v) - 1 & \text{zbrajanja} \\ \ell(v) & \text{množenja} \\ 2\,\ell(v) - 1 & \text{dijeljenja} , \end{cases}$$

ili, ukupno

$$\text{w} - \text{Compl}_A(\text{normaliziraj}) = 4\,\ell(v) - 2 .$$

Za ostatak vrijedi :

$$\text{w} - \text{Compl}_A(\text{ostatak}) = \begin{cases} \ell(v) - 1 & \text{zbrajanja} \\ \ell(v) - 1 & \text{množenja} \\ 2\,\ell(v) - 1 & \text{dijeljenja} , \end{cases}$$

ili

$$\text{w} - \text{Compl}_A(\text{ostatak}) = 4\,\ell(v) - 3 .$$

Ako stupanj ostatka odredimo prije dijeljenja s d, onda se $\ell(v)$ može zamijeniti sa $\ell(r)$.

Pretpostavimo li uniformnu distribuciju za vodeću znamenku v_m, vjerojatnost da normalizacija bude potrebna je

$$\frac{1}{b}\left(\left\lfloor \frac{b}{2} \right\rfloor - 1\right) \approx \frac{1}{2} .$$

Ako divizori nastaju u sklopu nekog drugog algoritma, onda, uz varijabilnu preciznost, male vodeće znamenke imaju veću vjerojatnost, jer je distribucija logaritamska [Knuth, 1981.] Tada raste vjerojatnost normalizacije.

Za složenost dijeljenja je

$$\begin{aligned} \text{Compl}_A(\text{dijeljenje}) = \sum_{i=0}^{\deg(q)} & (\text{Compl}_A(q_i) + \text{Compl}_A(r) \\ + & \ \text{Compl}_A(\text{korigiraj } q_i, r)) \end{aligned} \tag{1.5.47}$$

Očito je za nalaženje procjene q_i

$$\text{Compl}_A(q_i) = 3$$

(po jedno zbrajanje, množenje i dijeljenje).

Ostatak $r \leftarrow u' - q_i \cdot v$ računamo samo ako je $q_i > 0$ (vjerojatnost je oko $1 - 1/b \approx 1$), i tada je

$$\text{w} - \text{Compl}_A(r) = \begin{cases} 4\,\ell(v) & \text{zbrajanja} \\ \ell(v) & \text{množenja} \\ 2\,\ell(v) & \text{dijeljenja} , \end{cases}$$

ili

$$\mathrm{w}-\ \mathrm{Compl}_A(r) = 7\,\ell\,(v)\ .$$

Uz pretpostavku da se u $1/2$ slučajeva javlja prijenos kod oduzimanja, prosječan broj zbrajanja se smanjuje na $3\,\ell\,(v)$, a ukupni broj operacija na $6\,\ell\,(v)$.

Za jednu korekciju kvocijenta i ostatka je

$$\mathrm{w}-\ \mathrm{Compl}_A(\text{korigiraj}) = 3\,\ell\,(v)+1 \quad \text{zbrajanja},$$

a prosječno je potrebno $2\,\ell\,(v)$.

U najgorem slučaju, potrebne su 2 korekcije, dok je prosječan broj korekcija za procjenu $\widehat{q}_L$ oko $0.34 \approx 1/3$ (prema napomeni 1.5.8.).

Iz (1.5.47), izlazi da je, zbog $\ell\,(q) \le \ell\,(u) - \ell\,(v) + 1$,

$$\mathrm{w}-\ \mathrm{Compl}_A(\text{dijeljenje}) = (\ell\,(u)-\ell\,(v)+1)\cdot \begin{cases} 10\,\ell\,(v)+5 & \text{zbrajanja} \\ \ell\,(v) & \text{množenja} \\ 2\,\ell\,(v) & \text{dijeljenja}\ , \end{cases}$$

ili, ukupno

$$\mathrm{w}-\ \mathrm{Compl}_A(\text{dijeljenje}) = (\ell\,(u)-\ell\,(v)+1)\,(13\,\ell\,(v)+5)\ .$$

Za prosječan broj operacija je

$$\mathrm{avg}-\ \mathrm{Compl}_A(\text{dijeljenje}) \approx (\ell\,(u)-\ell\,(v)+1)\,(7\,\ell\,(v)+4)\ .$$

Zbrajanjem svih ovih rezultata, u (1.5.47) izlazi

$$\begin{aligned} \mathrm{w}-\ \mathrm{Compl}_A(\mathsf{NDIV}) &= (\ell\,(u)-\ell\,(v)+1)\,(13\,\ell\,(v)+5)+8\,\ell\,(v)-4 \\ \mathrm{avg}-\ \mathrm{Compl}_A(\mathsf{NDIV}) &\approx (\ell\,(u)-\ell\,(v)+1)\,(7\,\ell\,(v)+4)+4\,\ell\,(v)-2\ , \end{aligned}$$

dodajući jedno zbrajanje za *vlead*1.

Za vremensku složenost vrijedi:

$$\mathrm{Compl}_T(\mathsf{NDIV}) \sim (\ell\,(u)-\ell\,(v)+1)\cdot\ell\,(v)\ . \tag{1.5.48}$$

Ovaj algoritam za dijeljenje je vrlo dobar za male duljine brojeva. Za veće duljine postoje mnogo brži algoritmi, bazirani na iterativnim metodama za rješavanje jednadžbi. Jedan takav algoritam dajemo u sljedećem dijelu.

1.6. Asimptotski brzo množenje i dijeljenje

Razmotrimo pitanje optimalnosti algoritma NMUL, ili, općenito, bilo kojeg algoritma za množenje prirodnih brojeva.

Pošto nas zanima broj aritmetičkih operacija potrebnih za množenje 2 prirodna broja u pozicionom zapisu u bazi b, zgodno je uvesti oznaku za taj broj, kao funkciju duljine (broja znamenki) brojeva koje množimo.

Definicija 1.6.1.

Neka je MUL *bilo koji opći algoritam za množenje prirodnih brojeva* $u\,,\, v \in \mathbb{N}$, *u pozicionom zapisu u nekoj (može i fiksnoj) bazi* b. *Aritmetičku složenost algoritma, u ovisnosti o duljini brojeva* u *i* v, *označavamo sa*

$$\mathrm{Mul}(n,m) = \mathrm{Compl}_A(\mathsf{MUL})\ ,$$

ako je $\ell\,(u) = n$, $\ell\,(v) = m$. *Ako je* $n = m$, *onda skraćeno označavamo*

$$\mathrm{Mul}(n) = \mathrm{Mul}(n,n)\ .$$

Pri tome smatramo da je algoritam **opći**, ako radi korektno za **sve** brojeve $u\ ,\ v \in \mathbb{N}$, tj. za proizvoljne duljine brojeva.

Ovo je bitan zahtjev, jer ako fiksiramo duljinu brojeva (ili raspon brojeva), onda je problem trivijalan. Tada možemo unaprijed konstruirati "tablicu množenja" za danu duljinu (raspon), pa se množenje svodi na čitanje rezultata iz tablice.

Uočimo da je na sekvencijalnim arhitekturama, $\mathrm{Mul}(n,m)$, odnosno $\mathrm{Mul}(n)$ dobra mjera potrebnog vremena.

Funkcija Mul, naravno, ovisi o algoritmu. No, sa stanovišta složenosti, ona, zapravo, **karakterizira** algoritam za množenje.

Osim toga, ta funkcija je vrlo korisna za zapis složenosti svih algoritama koji koriste množenja. Zapis složenosti tada ne ovisi o izboru algoritma za množenje i jasno je vidljiv odnos bitne složenosti algoritma i složenosti aritmetike kojom se on realizira. To je potrebno zbog toga što ćemo dati niz algoritama za množenje s bitno različitim složenostima.

Uz ove oznake, relacije (1.4.29) i (1.4.30) za algoritam NMUL možemo zapisati u obliku

$$\begin{aligned} \mathrm{Mul}(n,m) &= 5\,n\,m + 2 \\ \mathrm{Mul}(n) &= 5\,n^{\,2} + 2 \quad , \end{aligned} \tag{1.6.1}$$

tj. broj operacija potreban za množenje n – znamenkastih prirodnih brojeva je proporcionalan s $n^{\,2}$.

Očito je za **bilo koji** opći algoritam ispunjeno

$$\mathrm{Mul}(n) = \Omega\,(n)\ , \tag{1.6.2}$$

jer ulaz sadrži $2\,n$ općenito nezavisnih znamenki, na osnovu kojih treba postaviti $2\,n$ znamenki rezultata.

Ova dva rezultata pokazuju da je $\mathrm{Mul}(n)$ uvijek "između" n i n^2 (barem asimptotski) tj, da ima prostora za bitno ubrzanje klasičnog algoritma.

Ubrzanju klasičnog algoritma pristupamo strategijom "podijeli pa vladaj", kojom zadaću reda veličine n svodimo na neki niz istovrsnih zadaća, ali manje veličine. Taj postupak primjenjujemo rekurzivno, sve dok veličina zadaće ne padne toliko da je problem lako rješiv.

Ilustrirajmo ovaj pristup.

Pretpostavimo, radi jednostavnosti, da je $\ell\,(u) = \ell\,(v) = n$ i da je n paran broj

$$n = 2\,k\ .$$

Tada brojeve

$$u = \sum_{i=0}^{n} u_i\,b^{\,i} \quad , \quad v = \sum_{i=0}^{m} v_i\,b^{\,i} \quad , \tag{1.6.3}$$

možemo zapisati u bazi $b^{\,n/2} = b^{\,k}$ u obliku

$$u = U_1\,b^{\,k} + U_0 \quad , \quad v = V_1\,b^{\,k} + V_0 \quad , \tag{1.6.4}$$

gdje je

$$U_1 \;=\; \sum_{i=0}^{k-1} u_{i+k}\,b^{\,i} \;=\; (u_{2k-1}\ldots u_k)_b$$

$$U_0 \;=\; \sum_{i=0}^{k} u_i\,b^{\,i} \;=\; (u_{k-1}\ldots u_0)_b \quad ,$$

tj. U_1 je "značajnija" ili "gornja" polovina, a U_0 je "donja" polovina broja U, i analogno

$$V_1 \;=\; \sum_{i=0}^{k-1} v_{i+k}\,b^{\,i} \;=\; (v_{2k-1}\ldots v_k)_b$$

$$V_0 \;=\; \sum_{i=0}^{k} v_i\,b^{\,i} \;=\; (v_{k-1}\ldots v_0)_b \quad ,$$

Očito su $U_i\ ,\ V_i\ ,\ i = 0,1$ znamenke brojeva u i v u bazi $b^{\,k}$, i zapis (1.6.4) je normaliziran u bazi $b^{\,k}$, ako je zapis (1.6.3) normaliziran u bazi b.

Produkt $w = u \cdot v$, promatran u bazi $b^{\,k}$, ima oblik

$$w = (U_1 \cdot V_1)\,b^{\,2k} + (U_0 \cdot V_1 + U_1 \cdot V_0)\,b^{\,k} + U_0 \cdot V_0\ . \tag{1.6.5}$$

Ovaj zapis ne mora biti normaliziran u bazi b^k. Navedene operacije i normalizaciju treba, naravno, provesti radeći na nizovima znamenki u bazi b.

Relacija (1.6.5) znači da je problem nalaženja produkta n – znamenkastih brojeva u i v, sveden na problem nalaženja produkata $n/2 = k$ – znamenkastih brojeva $(U_1 \cdot V_1\ ,\ U_0 \cdot V_1\ ,\ U_1 \cdot V_0\ ,\ U_0 \cdot V_0)$ uz zbrajanja i množenja potencijom baze (pomake).

Može se pokazati da relacija (1.6.5) ne poboljšava klasični algoritam, jer je

$$\mathrm{Mul}(n) = 4\ \mathrm{Mul}(n/2) + c_1\, n + c_2\ ,$$

što daje

$$\mathrm{Mul}(n) = \mathcal{O}(n^{\,2})$$

(v. lema 1.6.1. , u nastavku).

Uočimo li da je

$$U_0 \cdot V_1 + U_1 \cdot V_0 = (U_1 + U_0) \cdot (V_1 + V_0) - U_0 \cdot V_0 - U_1 \cdot V_1\ , \tag{1.6.6}$$

onda je dovoljno naći 3 (a ne više 4) produkta. $U_0 \cdot V_0\ ,\ U_1 \cdot V_1$ su produkti k – znamenkastih, a $(U_1 + U_0) \cdot (V_1 + V_0)$ je produkt najviše $k+1$ – znamenkastih brojeva.

Relacije (1.6.5) i (1.6.6) daju sljedeći algoritam, kojeg pišemo u skraćenom obliku.

Algoritam 1.6.1. (NMULK0 – **Karacuba (1962.)**)

begin
$t1 \leftarrow (U_1 + U_0) \cdot (V_1 + V_0)$;
$t2 \leftarrow U_0 \cdot V_0$;
$t3 \leftarrow U_1 \cdot V_1$;
$w \leftarrow t3 \cdot b^{\,2k} + (t1 - t2 - t3) \cdot b^{\,k} + t2$;
end;

Ako pretpostavljamo da naznačena 3 množenja obavljamo rekurzivno, istim ovim algoritmom, za aritmetičku složenost dobivamo

$$\mathrm{Mul}(n) \le 2\ \mathrm{Mul}\Big(\frac{n}{2}\Big) + \mathrm{Mul}\Big(\frac{n}{2} + 1\Big) + c_1\, n\ . \tag{1.6.7}$$

Član $c_1\, n$ predstavlja gornju ogradu broja operacija za zbrajanja i množenja potencijom baze, jer aritmetička složenost tih dviju operacija ovisi linearno o duljini brojeva.

Član $\mathrm{Mul}(n/2+1)$ opisuje složenost množenja $(U_1+U_0)\cdot(V_1+V_0)$, jer ta dva broja mogu imati $n/2+1$ znamenki. Ako inzistiramo na parnoj duljini brojeva, možemo dodati vodeće znamenke nula i zamijeniti $\mathrm{Mul}(n/2+1)$ sa $\mathrm{Mul}(n/2+2)$. Ova promjena ne utiče bitno na karakter relacije (1.6.7), jer vrijedi sljedeća tvrdnja.

Propozicija 1.6.1.

Neka je $t \in \mathbb{N}$ *bilo koja konstanta. Za bilo koji algoritam množenja prirodnih brojeva u pozicionom zapisu, postoji konstanta* $C \in \mathbb{N}$ *, takva da je za svaki* $n \in \mathbb{N}$

$$\mathrm{Mul}(n+t) \leq \mathrm{Mul}(n) + C\,n \ . \tag{1.6.8}$$

C ovisi o t, ali ne i o n.

Dokaz:

Neka su u , $v \in \mathbb{N}$ $n+t$ – znamenkasti brojevi. Zapišimo te brojeve u obliku

$$u = U_1\,b^t + U_0 \quad , \quad v = V_1\,b^t + v_0$$

gdje su U_1, V_1 n – znamenkasti, a U_0, V_0 t – znamenkasti brojevi.

Produkt $w = u \cdot v$ možemo zapisati u formi sličnoj relaciji (1.6.5)

$$w = (U_1 \cdot V_1)\,b^{2t} + (U_0 \cdot V_1 + U_1 \cdot V_0)\,b^t + U_0 \cdot V_0 \ . \tag{1.6.9}$$

Produkt $U_1 \cdot V_1$ zahtijeva $\mathrm{Mul}(n)$ aritmetičkih operacija.

Produkte $U_0 \cdot V_1$ i $V_1 \cdot U_0$ možemo naći, tako da t puta primijenimo algoritam NMULD za množenje n – znamenkastog broja jednom znamenkom, i rezultate zbrojimo, ili direktno algoritmom NMUL. U oba slučaja potrebno je $c_1\,n\,t$ aritmetičkih operacija, gdje je c_1 neka konstanta.

Produkt $U_0 \cdot V_0$ zahtijeva $c_2\,t^{\,2}$, tj. konstantan broj aritmetičkih operacija, neovisno o n.

Preostale operacije (zbrajanja i pomake) u (1.6.9) možemo obviti za najviše $c_3\,(n+t)$ aritmetičkih operacija.

Ukupno je :

$$\mathrm{Mul}(n+t) \leq \mathrm{Mul}(n) + c_1\,n\,t + c_2\,t^{\,2} + c_3\,(n+t) \ .$$

No, tada je za $\forall n \in \mathbb{N}$

$$c_1\,n\,t + c_2\,t^{\,2} + c_3\,(n+t) \leq (c_1\,t + c_2\,t^{\,2} + c_3\,(t+1)) \cdot n \ ,$$

pa postoji $C \in \mathbb{N}$ takav da vrijedi (1.6.7).

Ako fiksiramo donju granicu n_0 za n, onda se C može još bolje odabrati, a (1.6.7) vrijedi tada za sve $n \geq n_0$. To je korisno, u slučaju da za $n < n_0$ koristimo neki drugi algoritam množenja. ■

Primjenom ovog rezultata sa $t = 1$ na relaciju (1.6.7), za složenost algoritma Karacube izlazi

$$\mathrm{Mul}(n) \leq 3\ \mathrm{Mul}\Big(\frac{n}{2}\Big) + c_0\,n \ , \tag{1.6.10}$$

gdje je c_0 neka konstanta, koja ne ovisi o n. Relacija (1.6.10) vrijedi za $n > 1$, ako algoritam upotrebljavamo rekurzivno, sve dok ne dobijemo pojedinačne znamenke. Tada koristimo aritmetiku računala u algoritmu 1.6.1. , pa je

$$\mathrm{Mul}(1) \leq c_0' \ . \tag{1.6.11}$$

Odaberemi li, radi jednostavnosti,

$$c = \max \{ \ c_0 \ , \ c_0' \ \} \ \ ,$$

dobivamo rekurzivne relacije :

$$\begin{aligned} \mathrm{Mul}(1) \ &\leq \ c \\ \mathrm{Mul}(n) \ &\leq \ 3 \ \mathrm{Mul}\Big(\frac{n}{2} \Big) + c\, n \ . \end{aligned} \tag{1.6.12}$$

Sljedeća lema omogućava rješenje ovih relacija.

Lema 1.6.1.

Neka je $T : \mathbb{N} \longrightarrow \mathbb{R}$ *monotono rastuća funkcija i neka su* a , d , $c \in \mathbb{R}^+$ *konstante takve da je*

$$a > d \ .$$

Ako funkcija T *zadovoljava rekurzivne jednadžbe*

$$\begin{aligned} T(1) \ &= \ c \\ T(n) \ &= \ a \cdot T(n/d) + c \cdot n \quad , \ \ n = d^k \ , \ k > 0 \end{aligned} \tag{1.6.13}$$

onda je :

$$T(n) = \frac{c}{a-d} \ \Big[a \cdot n^{\log_d a} - d \cdot n \Big] \tag{1.6.14}$$

za $n = d^k$, $k > 0$ *i postoji konstanta* $C > 0$ *takva da je*

$$T(n) \leq C \cdot n^{\log_d a} \quad , \ \ \forall n \in \mathbb{N} \ . \tag{1.6.15}$$

Dokaz:

Iz rekurzivnih relacija (1.6.13), indukcijom se lako dokazuje

$$T(d^k) = \frac{c}{a-d} \ \Big[a \cdot a^k - d \cdot d^k \Big] \ .$$

Kako je

$$a^k = d^{k \cdot \log_d a} = (d^k)^{\log_d a} \ ,$$

za $n = d^k$ izlazi (1.6.14). Relacija (1.6.15) je direktna posljedica monotonosti funkcije i pretpostavke $a > d$. ■

Funkcija Mulje očito monotono rastuća, jer brojeve duljine između $n/2$ i n možemo dopuniti vodećim nulama, do duljine n. Zbog toga, iz relacija (1.6.12) i leme 1.6.1. , za algoritam Karacube izlazi

$$\mathrm{Mul}(n) < C \cdot n^{\log_2 3} \quad , \quad n \in \mathbb{N} \tag{1.6.16}$$

što je znatno ubrzanje obzirom na klasični algoritam, jer je

$$\log_2 3 \approx 1.585 \ .$$

Napomena 1.6.1. *Preciznija analiza aritmetičke složenosti ovakvih rekurzivnih algoritama je izrazito ovisna o detaljima realizacije, posebno o realizaciji rekurzije. Zbog toga smo proveli samo analizu reda veličine funkcije* $\mathrm{Mul}(n)$ *.*

Algoritam Karacube se može generalizirati tako da brojeve u i v rastavljamo na $r+1$ dijelova, a ne samo na 2 dijela.

Pretpostavimo, radi jednostavnosti, da je $n = (r+1)\,k$, gdje je $r \in \mathbb{N}$ bilo koji fiksan broj.

Brojeve u i v možemo zapisati u bazi b^k u obliku

$$u = \sum_{i=0}^{r} U_i\, b^{ki} \quad , \quad v = \sum_{i=0}^{r} V_i\, b^{ki} \quad , \tag{1.6.17}$$

gdje su U_i , V_i , k – znamenkasti brojevi, za $i = 0, \ldots, r$.

Definiramo polinome $U(x)$, $V(x)$ sa

$$U(x) = \sum_{i=0}^{r} U_i\, x^i \quad , \quad V(x) = \sum_{i=0}^{r} V_i\, x^i \quad ,$$

i njihov produkt

$$W(x) = U(x) \cdot V(x) = \sum_{i=0}^{2r} W_i\, x^i \ .$$

Pošto je $u = U(b^k)$, $v = V(b^k)$, onda je

$$w = u \cdot v = W(b^k) \ . \tag{1.6.18}$$

Ako nađemo koeficijente W_i , $i = 0, \ldots, 2r$ polinoma $W(x)$, onda je lako izračunati

$$w = \sum_{i=0}^{2r} W_i\, b^{ki} \ , \tag{1.6.19}$$

koristeći zbrajanja i množenja potencijom baze. Ta faza zahtijeva broj operacija proporcionalan s $n = (r+1)\,k$.

Primijetimo da je polinom W potpuno određen bilo koeficijentima, bilo vrijednostima u $2r+1$ točaka.

Odaberimo točke $0, 1, \ldots, 2r$. Tada je

$$W(i) = U(i) \cdot V(i) \quad , \quad i = 0, \ldots, 2r \ .$$

Iz ovih podataka, koristeći Newtonov oblik interpolacionog polinoma (razmak točaka je 1)

$$W(x) = \sum_{i=0}^{2r} \frac{1}{i!} \Delta^i W(0) \cdot \prod_{j=0}^{i-1} (x - j) \ , \tag{1.6.20}$$

lako nalazimo koeficijente W_i Hornerovom shemom i polinomnom aritmetikom. Još je lakše $w = W(b^k)$ direktno izračunati iz (1.6.20).

Algoritam 1.6.2. (NMULR0 – **"podijeli, pa vladaj"**)

begin
{ $U(i)$, $V(i)$ }
for $i \leftarrow 0$ **to** $2r$ **do**
begin
$U(i) \leftarrow U_r$;
for $j \leftarrow r-1$ **downto** 0 **do**
$U(i) \leftarrow U(i) \cdot i + U_j$;
$V(i) \leftarrow V_r$;
for $j \leftarrow r-1$ **downto** 0 **do**
$V(i) \leftarrow V(i) \cdot i + V_j$;
end;
{ $W(i) = U(i) \cdot V(i)$ }
for $i \leftarrow 0$ **to** $2r$ **do**
$W(i) \leftarrow U(i) \cdot V(i)$
{ *tablica konačnih razlika* $\Delta^i W(0)$ }
for $i \leftarrow 1$ **to** $2r$ **do**
for $j \leftarrow 2r$ **downto** i **do**
$W(j) \leftarrow (W(j) - W(j-1))$;
{ $w = W(b^k)$ }
$w \leftarrow W(2r)$;
for $i \leftarrow 2r-1$ **downto** 0 **do**
$w \leftarrow (w \cdot b^k - w \cdot i)/(i+1) + W(i)$
end;

Naravno, procjenu broja aritmetičkih operacija $\mathrm{Mul}(n)$, za $n = (r+1)\,k$, vršimo uz pretpostavku da za množenja $W(i) = U(i) \cdot V(i)$ koristimo rekurzivno ovaj isti algoritam.

Prvo treba odrediti duljinu brojeva $U(i)$, $V(i)$.

Pošto je

$$U(i) = \sum_{j=0}^{r} U_j\, i^j \quad , \quad i = 0, \ldots 2r \ ,$$

gdje su U_j k – znamenkasti brojevi, to je $U_j < b^k$, $j = 0, \ldots, r$. Zbog toga je

$$U(i) < b^k \cdot \sum_{j=0}^{r} i^j \ .$$

Znamenke U_j (u bazi b^k) su nenegativne, pa brojevi $U(i)$ očito rastu po i.

Zbog toga je za $i = 0, \ldots, 2r$

$$U(i) \le U(2r) < b^k \cdot \frac{(2r)^{\,r+1} - 1}{2r - 1} \ .$$

Broj r je konstanta, neovisna o n, pa postoji konstanta $t \in \mathbb{N}$, takva da je

$$U(i) < b^{k+t} \quad , \quad i = 0, \ldots, 2r \ . \tag{1.6.21}$$

Isto vrijedi i za $V(i) \quad , \quad i = 0, \ldots, 2r$.

Zbog toga, nalaženje $U(i)$, $V(i) \quad , \quad i = 0, \ldots, 2r$, Hornerovom shemom, koristi operacije s najviše $k + t$ – znamenkastim brojevima.

Brojevi $i = 0, \ldots, 2r$, očito imaju konstantnu duljinu ($\ll t$) i možemo čak pretpostaviti da je njihova duljina jednaka 1 (iako to nije bitno). Zbog toga svaka operacija za nalaženje $U(i)$, $V(i)$ zahtijeva najviše

$$c_1 \, (k + t)$$

aritmetičkih operacija.

Takvih operacija je ukupno $(2r + 1) \cdot 2r$, pa je

$$\mathrm{Compl}_A(\ U(i)\ ,\ V(i)\ ,\ \ i = 0, \ldots, 2r\ ) \leq c_1 \cdot 2r \cdot (2r + 1) \cdot (k + t) \ . \tag{1.6.22}$$

Za $2r + 1$ produkata $W(i) = U(i) \cdot V(i)$, potrebno je ukupno

$$(2r + 1)\ \mathrm{Mul}(k + t)$$

operacija. Koristeći propoziciju 1.6.1. , dobivamo

$$\mathrm{Compl}_A(\ W(i)\ ,\ \ i = 0, \ldots, 2r\ ) \leq (2r + 1) \cdot [\,\mathrm{Mul}(k) + c_2 \cdot k\,] \ . \tag{1.6.23}$$

Brojevi $W(i)$ imaju najviše $2\,(k + t) + 1$ znamenku, direktno iz (1.6.21). Nalaženje tablice konačnih razlika zahtijeva $r \cdot (2r + 1)$ oduzimanja, pa je

$$\mathrm{Compl}_A(\ \Delta^i W(0)\ ,\ \ i = 0, \ldots, 2r\ ) \leq c_2 \cdot r \cdot (2r + 1) \cdot (2k + 2t + 1) \ . \tag{1.6.24}$$

Može se pokazati da prilikom oduzimanja ne dobivamo negativne brojeve.

Računanje broja w koristi samo pomake, zbrajanja, te množenja i dijeljenja znamenkama (ili brojevima konstantne duljine). Zbog toga je

$$\mathrm{Compl}_A(w) \leq c_3 \cdot 2r \cdot (2k + 2t + 1) \ . \tag{1.6.25}$$

Zbrajanjem relacija (1.6.22) – (1.6.25) izlazi

$$\mathrm{Mul}((r + 1)\,k) \leq (2r + 1)\ \mathrm{Mul}(k) + c\,(r + 1)\,k \ ,$$

za neku konstantu c. Zbog toga je

$$\mathrm{Mul}(n) \leq (2r + 1)\ \mathrm{Mul}\Big(\ \frac{n}{r + 1}\ \Big) + c \cdot n \ .$$

Ako konstantu c odaberemo tako da je

$$\mathrm{Mul}(1) \leq c \ ,$$

lema 1.6.1. daje

$$\mathrm{Mul}(n) \leq C \cdot n^{\log_{r+1}(2r+1)} \ .$$

Kako je

$$\log_{r+1}(2r+1) < 1 + \log_{r+1} 2 \ ,$$

dobivamo ocjenu aritmetičke složenosti algoritma 1.6.2.

$$\mathrm{Mul}(n) \leq C \cdot n^{1+\log_{r+1} 2} \ . \tag{1.6.26}$$

Ovdje je konstanta C ovisna samo o r i bazi b.

Zbog

$$\log_{r+1} 2 \to 0 \ \text{ za } \ r \to \infty \ ,$$

relacija (1.6.26) dokazuje sljedeći teorem.

Teorem 1.6.1.

Za bilo koji fiksni $\varepsilon > 0$, postoji algoritam množenja, takav da je broj aritmetičkih operacija potrebnih za množenje dva n – znamenkasta broja u bazi b, dan sa

$$\mathrm{Mul}(n) \leq C\,(\varepsilon\ ,\ b) \cdot n^{\,1+\varepsilon} \quad , \ \ \forall n \in \mathbb{N} \ , \tag{1.6.27}$$

gdje je $C(\varepsilon\ ,\ b)$ neka konstanta koja ovisi samo o ε i b, a ne ovisi o n. ■

Napomena 1.6.2. *Iz analize složenosti algoritma* NMULR0*, očito je da konstanta proporcionalnosti C vrlo brzo raste, kad povećamo broj r (stupanj polinoma U, V). Zbog toga je taj algoritam efikasan samo za ogromne duljine n. Za sve razumne duljine n (prikazive u računalu), algoritam je efikasan samo za vrlo male brojeve r (r manje ili približno jednako 5), a i tada n mora biti velik.*

Za sve praktične primjene dovoljni su klasični algoritmi i algoritam Karacube.

Strategija "podijeli, pa vladaj", korištenjem polinomne reprezentacije i interpolacije, može se za teoretske potrebe, još poboljšati.

Treba dozvoliti da stupanj r varira sa n, tako da biramo sve veće vrijednosti za r, kad n raste.

Birajući $r \approx \sqrt{n}$, može se postići (algoritam Toom – Cook, v. [Knuth, 1981.])

$$\mathrm{Mul}(n) \leq C \cdot n^{\,1+3.5/\sqrt{\lg n}} \quad , \ \ n \in \mathbb{N} \ ,$$

a uz male modifikacije i

$$\mathrm{Mul}(n) \leq C \cdot n^{\,1+\sqrt{2/\lg n}} \cdot \log n \quad , \ \ n \in \mathbb{N} \ .$$

Napomena 1.6.3. *Najbrži poznati algoritam za množenje je algoritam Schönhage – Strassen (v. [Aho, Hopcroft, Ullman, 1976.]) za koji vrijedi*

$$\mathrm{Mul}(n) \leq C \cdot n \cdot \log n \cdot \log\log n \quad , \quad n \in \mathbb{N} \ .$$

Algoritam je baziran na sličnoj ideji, uz korištenje brze Fourierove transformacije i modularne aritmetike.

Najbolja donja ograda za **bilo koji** algoritam množenja n – znamenkastih brojeva na klasičnim arhitekturama je

$$\mathrm{Mul}(n) = \Omega\left(n \cdot \log n / \log\log n\right) \ .$$

Zadnja dva rezultata daju povod za široko rasprostranjeno vjerovanje da je

$$\mathrm{Mul}(n) = \Theta\left(n \cdot \log n\right)$$

za optimalni algoritam množenja.

Pokažimo još da se dijeljenje prirodnih brojeva može obaviti podjednako brzo kao i množenje.

Kvocijent brojeva u , $v \in \mathbb{N}_0$, $v > 0$, možemo zapisati u obliku

$$\frac{u}{v} = u \cdot \frac{1}{v} \ , \tag{1.6.28}$$

pa je dovoljno naći recipročnu vrijednost $1/v$ broja v i rezultat pomnožiti s u.

Osnovni problem ovog pristupa je u tome, što $1/v$ nije više cijeli broj i može imati beskonačni prikaz u bazi b. Zbog toga treba naći neku dovoljno točnu aproksimaciju za $1/v$.

Pošto nas zanima cjelobrojni kvocijent

$$q = \left\lfloor \frac{u}{v} \right\rfloor \ ,$$

aproksimacija za $1/v$ mora biti toliko točna da omogući nalaženje broja q ili njegove dobre procjene $\widehat{q}$.

Ako je $\ell\left(u\right) = n$, $\ell\left(v\right) = m$, i ako nađemo aproksimaciju w za $1/v$ s točnošću ε

$$\frac{1}{v} = w + \varepsilon$$

gdje je

$$|\varepsilon| \leq b^{-n} \ , \tag{1.6.29}$$

onda je

$$\left| \frac{u}{v} - u \cdot w \right| \leq u \cdot |\varepsilon| \leq 1 \ . \tag{1.6.30}$$

Stavimo li

$$\widehat{q} = \lfloor u \cdot w' \rfloor$$

onda je

$$|q - \widehat{q}| \leq 1 \ . \tag{1.6.31}$$

Za nalaženje w koristimo Newtonovu metodu za rješavanje jednadžbe

$$\frac{1}{x} - v = 0 \ .$$

Pretpostavimo da možemo naći dovoljno dobru startnu aproksimaciju $w_{(0)}$ za $1/v$. Sljedeće aproksimacije su definirane rekurzivnom relacijom

$$w_{(k+1)} = w_{(k)} \cdot (2 - v \cdot w_{(k)}) \ \ , \ \ k \geq 0 \ . \tag{1.6.32}$$

Označimo sa $e_{(k)}$ pogrešku k – te aproksimacije

$$e_{(k)} = \frac{1}{v} - w_{(k)} \ \ , \ \ k \geq 0 \ .$$

Iz (1.6.32) direktno izlazi

$$e_{(k+1)} = v \cdot e_{(k)}^2 \ \ , \ \ k \geq 0 \ . \tag{1.6.33}$$

Ako za pogrešku $e_{(k)}$ osiguramo da je

$$|e_{(k)}| \leq b^{-(m+d)} \ , \tag{1.6.34}$$

onda je iz (1.6.33), zbog $v < b^m$

$$|e_{(k+1)}| \leq b^{-(m+2\,d)} \ . \tag{1.6.35}$$

Zbog $b^{m-1} \leq v < b^m$, za $1/v$ je

$$b^{-m} < \frac{1}{v} \leq b^{-(m-1)} \ , \tag{1.6.36}$$

pa (1.6.34) i (1.6.35) pokazuju da svaki sljedeći korak udvostručuje broj točnih znamenki, uz pretpostavku **egzaktnog** izvođenja operacija u (1.6.32).

Za praktičnu realizaciju, brojeve $w_{(k)}$ moramo prikazati nekim konačnim nizom znamenki u bazi b. Osim toga, poželjno je čitav proces realizirati korištenjem aritmetike prirodnih brojeva.

Zbor relacije (1.6.36), broj $1/v$ ima sljedeći prikaz u bazi b

$$\frac{1}{v} = b^{-(m-1)} \cdot \sum_{i=0}^{\infty} v'_i \, b^{-i}$$

(generalizacija teorema 1.1.1. za realne brojeve), gdje je

$$0 \le v'_i < b \quad , \quad i \in \mathbb{N}_0$$

i vrijedi :

$$b^{-m} < \frac{1}{v} < b^{-(m-1)} \iff v'_0 = 0 \ \text{ i } \ v'_1 > 0$$
$$\frac{1}{v} = b^{-(m-1)} \iff v'_0 = 1 \ \text{ i } \ v'_i = 0 \quad , \ \forall i \in \mathbb{N} \ .$$

Pretpostavimo da je unaprijed zadana preciznost p i da tražimo aproksimaciju w' za $1/v$ takvu da je

$$b^{m-1+p} \cdot w' = \left\lfloor \frac{b^{m-1+p}}{v} \right\rfloor = \sum_{i=0}^{p} v'_i \, b^{p-i} = \sum_{i=0}^{p} w_i \, b^i \tag{1.6.37}$$

($w_i = v'_{p-i}$). Uočimo da je $\ell(w') = p$, osim u slučaju $v = b^{m-1}$, kada je $\ell(w') = p+1$.

Ovo pokazuje da u relaciji (1.6.32) lako prelazimo na cijele brojeve. Tu relaciju pomnožimo s b^{m-1+p} i definiramo

$$w'_{(k)} = b^{m-1+p} \cdot w_{(k)} \ .$$

Dobivamo rekurzivnu relaciju

$$w'_{(k+1)} = 2\, w'_{(k)} - \frac{v}{b^{m-1+p}} \cdot \left(w'_{(k)}\right)^2 \quad , \ k \ge 0 \ , \tag{1.6.38}$$

u kojoj sve operacije možemo obaviti koristeći cjelobrojnu aritmetiku.

Ovaj postupak ima manu da u ranoj fazi algoritma svih p (ili $p+1$) znamenki broja $w_{(k)}$ sudjeluje u operacijama, iako ih je većina netočna.

Drugim riječima, ne isplati se cijelo vrijeme provoditi postupak (1.6.38) u zadanoj točnosti.

Druga mogućnost je da preciznost p povećavamo tokom iteracija, dupliranjem p u svakoj sljedećoj iteraciji. U tom slučaju, i od broja v treba koristiti samo dio vodećih znamenki u (1.6.38). Pri tome treba paziti da ne dođe do gubitka točnosti.

Pretpostavimo, radi jednostavnosti, da je $b = 2$ i da je broj m potencija broja 2

$$m = 2^l$$

i da je tražena točnost $p = m$. Kasnije ćemo opisati kako izbjeći ova ograničenja.

Osnovna ideja algoritma je da u k – tom koraku, u relaciji (1.6.38), koristimo točnost $p_k = 2^k$ i samo vodećih p_k znamenki broja v.

Algoritam 1.6.3. (NREC – **recipročni prirodni broj**)

Ulaz : *niz znamenki* $(v_{m-1}, \ldots, v_0)$ *prirodnog broja* v *u bazi* 2, *uz pretpostavku da za* $m = \ell(v)$ *vrijedi*

$$m = 2^l$$

za neki $l \in \mathbb{N}_0$.

Izlaz : *niz znamenki* $(w_{r-1}, \ldots, w_0)$ *broja*

$$w = \lfloor 2^{2m-1}/v \rfloor$$

gdje je $r = \ell(w) \in \{\, m \,,\, m+1 \,\}$.

procedure NREC *(v : mpnat ;* **var** *w : mpnat);*

begin
  $m \leftarrow \deg(v) + 1$; { $\ell(v)$ }
  $(w_1 \,,\, w_0) \leftarrow (1 \,,\, 0)$;
  $p \leftarrow 1$;
  if $m > 1$ **then**
    repeat
      $p \leftarrow 2 \cdot p$; { $p = 2^k$ }
        { *nađi* $w \leftarrow \lfloor 2^{2p-1} / \lfloor v/2^{m-p} \rfloor \rfloor$ }
      $(s_{2p-1}, \ldots, s_0) \leftarrow (w_{p/2}, \ldots, w_0) \cdot 2^{3p/2} -$
          $-(w_{p/2}, \ldots, w_0)^2 \cdot (v_{m-1}, \ldots, v_{m-p})$;
      $(w_p, \ldots, w_0) \leftarrow (s_{2p-1}, \ldots, s_{p-1})$;
        { *popravak* }
      for $i \leftarrow 2$ **downto** 0 **do**
        if $((w_p, \ldots, w_0) + 2^i) \cdot (v_{m-1}, \ldots, v_{m-p}) \le 2^{2p-1}$ **then**
          $(w_p, \ldots, w_0) \leftarrow (w_p, \ldots, w_0) + 2^i$
    until $p = m$;
  if $w_m = 0$ **then**
    $\deg(w) \leftarrow \deg(v)$
  else
    $\deg(w) \leftarrow \deg(v) + 1$
end; { NREC }

Teorem 1.6.2.

Algoritam NREC *nalazi broj* $w \in \mathbb{N}$ *takav da je*

$$w = \lfloor 2^{2m-1}/v \rfloor \,, \tag{1.6.39}$$

tj. vrijedi

$$\begin{aligned} v \cdot w &= 2^{2m-1} - r \\ 0 \le r &< v \qquad . \end{aligned} \tag{1.6.40}$$

Dokaz:

Dokažimo da algoritam nalazi niz brojeva

$$w_{(k)} = \left\lfloor \frac{2^{2p-1}}{\lfloor v/2^{m-p} \rfloor} \right\rfloor \quad , \quad k = 0, \ldots, l \ , \tag{1.6.41}$$

pri čemu je $w = w_{(0)}$ na početku algoritma, i $w = w_{(k)}$ na kraju petlje u kojoj je $p = 2^k$, $k = 1, \ldots, l$.

Pošto algoritam vraća $w = w_{(l)}$, zbog $m = 2\,l$, iz (1.6.41) direktno izlazi (1.6.39).

Dokaz relacije (1.6.41) provodimo indukcijom po k. Za $k = 0$ je $p = 1$, pa je

$$\lfloor v/2^{m-1} \rfloor = v_{m-1} = 1 \ ,$$

jer je $v_{m-1} > 0$ i $b = 2$. Algoritam postavlja $w = (1 \ , \ 0) = 2$, pa vrijedi (1.6.41).

Pretpostavimo da (1.6.41) vrijedi za $k < l$, i označimo $p = 2^{k+1}$. Također označimo, za lakši zapis

$$\begin{aligned} v_{(k)} &= \lfloor v/2^{m-p/2} \rfloor = (v_{m-1}, \ldots, v_{m-p/2}) \\ v_{(k+1)} &= \lfloor v/2^{m-p} \rfloor = (v_{m-1}, \ldots, v_{m-p}) \quad , \end{aligned}$$

pa je

$$v_{(k+1)} = v_{(k)} \cdot 2^{p/2} + v'_{(k)}$$

sa

$$v'_{(k)} = (v_{m-p/2-1}, \ldots, v_{m-p}) \ .$$

Po pretpostavci indukcije, na početku koraka petlje s $p = 2^{k+1}$, vrijedi

$$w_{(k)} = \left\lfloor \frac{2^{p-1}}{\lfloor v/2^{m-p/2} \rfloor} \right\rfloor$$

ili

$$w_{(k)} = \left\lfloor \frac{2^{p-1}}{v_{(k)}} \right\rfloor = (w_{p/2}, \ldots, w_0) \ .$$

To znači

$$\begin{aligned} v_{(k)} \cdot w_{(k)} &= 2^{p-1} - r_{(k)} \\ 0 \le r_{(k)} &< v_{(k)} \quad . \end{aligned} \tag{1.6.42}$$

Prva naredba petlje računa broj

$$s = 2^{3p/2} \cdot w_{(k)} - v_{(k+1)} \cdot w_{(k)}^2 \ . \tag{1.6.43}$$

Kako je $v_m = 1$, to je $v_{(k)} \ge 2^{p/2-1}$, pa je $w_{(k)} \le 2^{p/2}$, što pokazuje da je

$$s \le 2^{3p/2} \cdot 2^{p/2} - v_{(k+1)} \cdot w_{(k)}^2 < 2^{2p} \ .$$

To znači da algoritam egzaktno računa broj $s = (s_{2p-1}, \ldots, s_0)$.

Promatramo produkt $s \cdot v_{(k+1)}$.

$$\begin{aligned} s\, v_{(k+1)} &= s \cdot (v_{(k)}\, 2^{p/2} + v'_{(k)}) = (1.6.43) \\ &= v_{(k)}\, w_{(k)}\, 2^{2p} + v'_{(k)}\, w_{(k)}\, 2^{3p/2} - (v_{(k)}\, w_{(k)}\, 2^{p/2} + v'_{(k)}\, w_{(k)})^2 \end{aligned}$$

Primjenom relacije (1.6.42) dobivamo

$$s\, v_{(k+1)} = 2^{3p-2} - (2^{p/2}\, r_{(k)} - v'_{(k)}\, w_{(k)})^2 \ .$$

Odavde je, dijeljenjem s 2^{p-1}

$$2^{2p-1} = \frac{s\, v_{(k+1)}}{2^{p-1}} + t \tag{1.6.44}$$

gdje je

$$t = 2^{-(p-1)} \cdot (2^{p/2}\, r_{(k)} - v'_{(k)}\, w_{(k)})^2 \ .$$

Iz relacije (1.6.42) izlazi $r_{(k)} < 2^{p/2}$, jer je i $v_{(k)} < 2^{p/2}$. Vrijedi i $w_{(k)} \leq 2^{p/2}$ i $v'_{(k)} < 2^{p/2}$, pa je

$$|2^{p/2}\, r_{(k)} - v'_{(k)}\, w_{(k)}| < 2^p$$

što daje

$$0 \leq t < 2^{p+1} \ . \tag{1.6.45}$$

Algoritam postavlja

$$w = \lfloor s/2^{p-1} \rfloor \ .$$

Iz (1.6.44), zbog $t \geq 0$ izlazi :

$$\begin{aligned} 2^{2p-1} &\geq \frac{s\, v_{(k+1)}}{2^{p-1}} \geq v_{(k+1)} \cdot \left\lfloor \frac{s}{2^{p-1}} \right\rfloor \\ &> v_{(k+1)} \cdot \left(\frac{s}{2^{p-1}} - 1 \right) = 2^{2p-1} - v_{(k+1)} - t \ , \end{aligned}$$

što sa $v_{(k+1)} < 2^p$ i $t < 2^{p+1}$, daje

$$2^{2p-1} \geq v_{(k+1)} \cdot w > 2^{2p-1} - 2^{p+1} - 2^p \ .$$

To znači da je

$$v_{(k+1)} \cdot w = 2^{2p-1} - r' \ ,$$

gdje je

$$0 \leq r' < 2^{p+1} + 2^p \ .$$

Kako je $v_{(k+1)}\, 2^{p-1}$, to je

$$0 \leq r' < 6\, v_{(k+1)} \ .$$

Odavde slijedi, da korekcijom

$$w_{(k+1)} = w + c \tag{1.6.46}$$

sa $c < 6$ možemo postići

$$v_{(k+1)} \cdot w_{(k+1)} = 2^{2p-1} - r_{(k+1)}$$
$$0 \leq r_{(k+1)} < v_{(k+1)} \quad .$$

Algoritam, na kraju, upravo vrši korekciju (1.6.46).

Time je dokazana relacija (1.6.41). ■

Primijetimo da je algoritam rekurzivan, i da vrijednosti $w_{(k)}$ ne izlaze direktno Newtonovom metodom, zbog korekcija.

Označimo sa $\mathrm{Rec}(m)$ aritmetičku složenost ovog algoritma. Ona, naravno, ovisi o složenosti algoritma za množenje kojeg koristimo.

Za nalaženje $w = w_{(l)}$ potrebno je naći $w_{(l-1)}$ i obaviti zadnji prolaz kroz petlju u algoritmu, sa $p = m = 2^l$.

Nalaženje broja s zahtijeva kvadriranje broja $w_{(l-1)}$ i množenje sa v, te zbrajanja i pomak, pa je

$$\mathrm{Compl}_A(s) \leq \mathrm{Mul}\Big(\frac{m}{2} + 1\Big) + \mathrm{Mul}(m+2) + c_1\, m \ .$$

Nalaženje broja w iz s se obavlja pomakom, a faza korekcije zahtijeva najviše 3 množenja, uz nekoliko zbrajanja i uspoređivanja.

Pažljivom realizacijom, korekcija se može obaviti samo jednim množenjem, uz nešto dodatnih zbrajanja i oduzimanja.

Zbog toga je, korištenjem propozicije 1.6.1. ,

$$\mathrm{Rec}(m) \leq \mathrm{Rec}\Big(\frac{m}{2}\Big) + \frac{5}{2}\ \mathrm{Mul}(m) + c\, m \ .$$

Ako konstantu c odaberemo tako da je

$$\mathrm{Rec}(1) \leq c + \frac{5}{2}\ \mathrm{Mul}(1) \ ,$$

onda je

$$\mathrm{Rec}(m) \leq \sum_{k=0}^{l} \frac{5}{2}\ \mathrm{Mul}(2^k) + c\, 2^k \ .$$

Pretpostavimo da je

$$\mathrm{Mul}\Big(\frac{1}{2}\, n\Big) \leq \frac{1}{2}\ \mathrm{Mul}(n) \quad , \quad \forall n \geq 2 \ ,$$

što odgovara tome da $\mathrm{Mul}(n)$ raste barem linearno. Tada je

$$\mathrm{Rec}(m) \leq 5\ \mathrm{Mul}(m) + c\,m \quad , \quad \forall m \in \mathbb{N}\ ,$$

a lako se vidi da postoji konstanta c', takva da je

$$\mathrm{Rec}(m) \leq c'\ \mathrm{Mul}(m) \quad , \quad \forall m \in \mathbb{N}\ .$$

Može se pokazati da je i

$$\mathrm{Rec}(m) = \Theta\left(\mathrm{Mul}(m)\right)\ .$$

Napomena 1.6.4. *U algoritmu* NREC *smo pretpostavili da je* $\ell\,(v) = m = 2^l$ *, i da je tražena preciznost* $p = m$ *. Ako duljina* m *nije potencija broja* 2*, onda nađemo* $m_1 = 2^l$ *takav da je*

$$m_1/2 < m \leq m_1\ .$$

Definiramo $v_1 = 2^{\,m_1 - m} \cdot v$ *i nađemo aproksimaciju* w_1 *za* $1/v$ *algoritmom* NREC*. Broj*

$$w = \lfloor w_1/2^{\,m_1-m} \rfloor$$

je tražena aproksimacija za $1/v$ *s preciznošću* $p = m$ *.*

Ako je tražena preciznost $p \neq m$ *, onda definiramo*

$$v = \lfloor v \cdot 2^{\,m_2-m} \rfloor$$

gdje je m_2 *potencija broja* 2*, takva da je*

$$\frac{m_2}{2} < p \leq m_2$$

Analogno, nađemo i aproksimaciju za w_2 *za* $1/v_2$ *algoritmom* NREC *i broj*

$$w = \lfloor w_2/2^{m_2-p} \rfloor$$

je tražena točnost aproksimacija s preciznošću p*.*

Napomena 1.6.5. *Algoritam* NREC *se može generalizirati na baze* $b > 2$ *, ali je tada korekcija nešto kompliciranija.*

Za cjelobrojno dijeljenje, relacija (1.6.29) pokazuje da je potrebno $1/v$ naći s precizošću $p = n - m + 1$.

To pokazuje da složenost cjelobrojnog dijeljenja linearno ovisi o složenosti potrebnih množenja, pa funkcija Mul karakterizira aritmetičke algoritme. Naime, množenje ima bitno veću složenost od zbrajanja i oduzimanja, a dijeljenje ima podjednaku složenost kao množenje.

1.7. Cjelobrojna aritmetika

Cijeli broj $u \in \mathbb{Z}$ najjednostavnije je prikazati u obliku

$$u = \text{sign}(u) \cdot \text{abs}(u) \tag{1.7.1}$$

gdje je

$$\text{sign}(u) = \begin{cases} 1 & , \text{ za } u > 0 \\ 0 & , \text{ za } u = 0 \\ -1 & , \text{ za } u < 0 \end{cases} ,$$

i $\text{abs}(u) \in \mathbb{N}_0$. Apsolutnu vrijednost $\text{abs}(u)$ prikazujemo u pozicionom zapisu u bazi b.

Duljinu $\ell\,(u)$ cijelog broja $u \in \mathbb{Z}$, definiramo sa

$$\ell\,(u) = \ell\,(\text{abs}(u)) \ .$$

Aritmetičke operacije na skupu $\mathbb{Z}$, svodimo na aritmetičke operacije na skupu $\mathbb{N}_0$, provjerom predznaka cijelih brojeva.

Za realizaciju algoritama pretpostavljamo da je tip *mpint* neka realizacija zapisa (1.1.2), na pr.

```
mpint = record
            sign : integer;  { ili −1 . . 1 }
            abs  : mpnat
        end;
```

Algoritam 1.7.1. (IADD – **zbrajanje cijelih brojeva**)
Ulaz : *cijeli brojevi u, v u pozicionom zapisu u bazi b.*
Izlaz : *cijeli broj*

$$w = u + v$$

u pozicionom zapisu u bazi b.

```
procedure IADD ( u , v : mpint ; var w : mpint);

begin
  if sign(u) = sign(v) then
    begin
    NADD (abs(u) , abs(v) , abs(w));
    sign(w) ← sign(u)
    end
  else
```

```
    begin
    NCOMP (abs(u) , abs(v) , sign);
    if sign = 1 then
       begin
       NSUB (abs(u) , abs(v) , abs(w));
       sign(w) ← sign(u)
       end
    else if sign = −1 then
       begin
       NSUB (abs(v) , abs(u) , abs(w));
       sign(w) ← sign(v)
       end
    else
       begin
       sign(w) ← 0;
       deg(abs(w)) ← −1
       end
    end
end; { IADD }
```

Korektnost algoritma je trivijalna, a za složenost, iz propozicija 1.2.5. i 1.3.10. , slijedi

$$\begin{aligned} \text{b} - \text{Compl}_A(\mathsf{IADD}) &= \min \{ \ell(u) , \ell(v) , \ell(w) \} \\ \text{w} - \text{Compl}_A(\mathsf{IADD}) &= 3 \min \{ \ell(u) , \ell(v) , \ell(w) \} + 2 \quad . \end{aligned}$$

Oduzimanje cijelih brojeva možemo realizirati analogno zbrajanju, ili koristeći promjenu predznaka

$$u - v = u + (-v) \ .$$

Algoritam 1.7.2. (ISUB – **oduzimanje cijelih brojeva**)
Ulaz : *cijeli brojevi u, v u pozicionom zapisu u bazi b.*

Izlaz : *cijeli broj*

$$w = u - v$$

u pozicionom zapisu u bazi b.

```
procedure ISUB ( u , v : mpint ; var w : mpint);

begin
   sign(v) = − sign(v);
   IADD (u , v , w)
end; { ISUB }
```

Očito je

$$\mathrm{Compl}_A(\mathsf{ISUB}) = \mathrm{Compl}_A(\mathsf{IADD}) \ .$$

Isti rezultat izlazi i za realizaciju ISUB analognu algoritmu IADD.

Množenje i dijeljenje se realiziraju trivijalno.

Algoritam 1.7.3. (IMUL – **množenje cijelih brojeva**)
Ulaz : *cijeli brojevi u, v u pozicionom zapisu u bazi b.*

Izlaz : *cijeli broj*

$$w = u \cdot v$$

u pozicionom zapisu u bazi b.

procedure IMUL *(u , v : mpint ;* **var** *w : mpint);*

begin
  $\mathrm{sign}(w) = \mathrm{sign}(u) \cdot \mathrm{sign}(v)$;
  NMUL $(\mathrm{abs}(u)\ ,\ \mathrm{abs}(v)\ ,\ \mathrm{abs}(w))$
end; { IMUL }

Za složenost vrijedi

$$\mathrm{Compl}_A(\mathsf{NMUL}) = \mathrm{Compl}_A(\mathsf{NMUL}) \ ,$$

jer se $\mathrm{sign}(w)$ može odrediti samo uspoređivanjem.

Algoritam 1.7.4. (IDIV – **dijeljenje cijelih brojeva s ostatkom**)
Ulaz : *cijeli brojevi u, v u pozicionom zapisu u bazi b, uz pretpostavku $v \neq 0$.*

Izlaz : *cijeli brojevi q, r u pozicionom zapisu u bazi b, takvi da je*

$$u = q \cdot v + r$$

tj. $q = \lfloor u/v \rfloor$, a za r vrijedi

$$\begin{aligned} 0 \ &\leq r < |v| \quad , \ za \ \ u \geq 0 \\ -|v| \ &< r \leq \ 0 \quad , \ za \ \ u > 0 \ \ . \end{aligned}$$

procedure IDIV *(u , v : mpint ;* **var** *q , r : mpint);*

begin
  $\mathrm{sign}(q) = \mathrm{sign}(u) \cdot \mathrm{sign}(v)$;
  $\mathrm{sign}(r) = \mathrm{sign}(u)$;
  NDIV $(\mathrm{abs}(u)\ ,\ \mathrm{abs}(v)\ ,\ \mathrm{abs}(q)\ ,\ \mathrm{abs}(r))$
end; { IDIV }

Očito je

$$\mathrm{Compl}_A(\mathsf{IDIV}) = \mathrm{Compl}_A(\mathsf{NDIV}) \ .$$

Napomena 1.7.1. *Cijele brojeve možemo prikazati i na druge načine. Često se koristi tzv. komplement notacija, koja zahtijeva fiksnu duljinu brojeva. Negativni brojevi se prikazuju na isti način kao i pozitivni, a za svaku grupu je rezervirana polovina raspona brojeva. Ovaj prikaz odgovara modularnoj aritmetici i pogodan je za zbrajanje i oduzimanje. Za množenje i dijeljenje je mnogo pogodniji pozicioni zapis, u kom se predznak broja pamti posebno.*

Druga mogućnost je korištenje pozicionog prikaza iz teorema 1.1.1. , s tim da negativni brojevi imaju vodeću ili sve znamenke negativne. Posljednja varijanta je realizirana u [Collins, Mignotte, Winkler, 1982.].

Pošto smo analizirali osnovne aritmetičke algoritme za prirodne i cijele brojeve, možemo razmotriti detaljnu realizaciju strukture podataka za prikaz brojeva.

Do sada smo koristili opću strukturu niza, koja dozvoljava nekoliko bitno različitih realizacija.

Ako **ne želimo** ograničiti maksimalnu duljinu niza (tj. raspon brojeva u aritmetici), prirodno je koristiti dinamičko raspolaganje memorijom. U svim algoritmima, znamenke brojeva obrađujemo sekvencijalno – rastuće ili padajuće po pripadnim potencijama baze. Zbog toga je dovoljno koristiti linearnu strukturu podataka – jednostruko ili dvostruko vezanu listu.

U Pascalu možemo definirati tip *mpnat* strukturom vezane liste :

```
digitptr = ↑ digitrec;
digitrec = record
               digit : integer;
               next  : digitptr
           end;
mpnat    = record
               deg   : integer;
               digits : digitptr
           end;
```

Predost ove realizacije je varijabilna duljina brojeva. To međutim, zahtijeva efikasno upravljanje memorijom, posebno pri kreiranju ili brisanju znamenki kod povećanja ili smanjenja duljine broja. Najčešće te operacije nisu efikasno izvedene, pa je aritmetika varijabilne duljine brojeva relativno spora.

Ograničimo li duljinu brojeva, možemo koristiti fiksne blokove memorije za prikaz brojeva, što omogućava vrlo brz pristup znamenkama. Tada u sve algoritme treba ugraditi kontrolu duljine brojeva, za osiguranje korektnosti algoritama.

Realizacija tipa *mpnat* poljem je na. pr.

```
mpnat =  record
             deg    : integer;
             digit  : array [0..maxdeg] of integer;
         end;
```

gdje je konstanta *maxdeg* najveći dozvoljeni stupanj broja.

Ova realizacija prikaza brojeva je znatno brža i jednostavnija, pa se vrlo često koristi. Osim toga, u nekim programskim jezicima (FORTRAN) nije standardno moguće dinamičko upravljanje memorijom.

1.8. Racionalna aritmetika

Racionalne brojeve $q \in \mathbb{Q}$ prikazujemo u obliku uređenog para

$$q = (q_1, q_2) \tag{1.8.1}$$

gdje su

$$\begin{aligned} q_1 &= \text{num}(q) \in \mathbb{Z} \\ q_2 &= \text{denom}(q) \in \mathbb{N} \end{aligned}$$

brojnik i nazivnik broja q

$$q = \frac{q_1}{q_2} \ .$$

Brojeve q_1 i q_2 prikazujemo u pozicionom zapisu u odabranoj bazi b.

Da bi zapis (1.8.1) bio jednoznačan, q_1 i q_2 moraju biti neskrativi, tj.

$$\gcd(q_1, q_2) = 1 \ . \tag{1.8.2}$$

Zbog toga u zapisu $q = (q_1, q_2)$ uvijek pretpostavljamo da vrijedi (1.8.2). Uočimo da je to i najekonomičniji prikaz, jer tada brojnik i nazivnik imaju najmanje duljine.

Sve operacije u racionalnoj aritmetici moraju davati neskrativi rezultat.

Analizirajmo, prvo, algoritam za nalaženje najveće zajedničke mjere.

Algoritam 1.8.1. (IGCD – **najveća zajednička mjera)**
Ulaz : *cijeli brojevi* $u \in \mathbb{Z}$ *i prirodni broj* $v \in \mathbb{N}$ *u pozicionom zapisu u bazi* b.
Izlaz : *prirodni broj* w

$$w = \gcd(u, v)$$

u pozicionom zapisu u bazi b.

```
procedure IGCD ( u : mpint   v : mpnat ;  var  w : mpnat);

begin
        { Euklidov algoritam }
   w ← abs(u);
   while  v > 0  do
      begin
      r ← w  mod  v;
      w ← v;
      v ← r
      end
end;  { IGCD }
```

Korektnost algoritma se lako dokazuje iz Euklidovog teorema.

Jedina operacija u algoritmu je cjelobrojno dijeljenje s ostatkom (algoritam NDIV).

Zbog toga složenost bitno ovisi o broju dijeljenja u algoritmu IGCD. Može se pokazati da za taj broj D dijeljenja u Euklidovom algoritmu vrijedi

$$D \leq \lceil \log_{\phi}(\sqrt{5}\, N) \rceil - 2 \ ,$$

gdje je $N = \max \{ u \ , \ v \}$ i

$$\phi = (1 + \sqrt{5})/2 \ ,$$

ili

$$D \leq (\log_{\phi} 2) \cdot \ell\,(M) + 2 \ , \tag{1.8.3}$$

gdje je $M = \min \{ u \ , \ v \}$ [Collins, Mignotte, Winkler, 1982.].

Oba rezultata pokazuju da je broj dijeljenja proporcionalan duljini ulaznih brojeva.

Aritmetička složenost svakog pojedinog dijeljenja je ovisna o duljini brojeva w i v u tom času, a te duljine sa stalno smanjuju tokom algoritma. Pokazali smo da za složenost jednog dijeljenja vrijedi

$$\begin{aligned} \mathrm{Compl}_A(w \bmod v) &\approx c \cdot \ell\,(v) \cdot [\ell\,(w) - \ell\,(v) + 1] \\ &\approx c \cdot \ell\,(v) \cdot \ell\left(\left\lfloor \frac{w}{v} \right\rfloor \right) \ . \end{aligned}$$

Označimo s $w_{(i)}$, $v_{(i)}$, $r_{(i)}$ vrijednosti brojeva w, v, r u trenu izvršavanja naredbe $r \leftarrow w \bmod v$, tokom i – tog prolaza kroz petlju.

Tada je za $i = 1, \ldots, D$

$$\mathrm{Compl}_A(w_{(i)} \bmod v_{(i)}) \approx c \cdot \ell\,(v_{(i)}) \cdot \ell\left(\left\lfloor \frac{w_{(i)}}{v_{(i)}} \right\rfloor \right) \ .$$

Algoritam postavlja

$$\begin{array}{rcl} w_{(i+1)} & = & v_{(i)} \\ v_{(i+1)} & = & r_{(i)} \end{array} \quad , \quad i = 1, \ldots, D-1 \ . \tag{1.8.4}$$

Koristeći $\ell\,(r_{(i)}) \leq \ell\,(v_{(i)}) \leq \ell\,(v_{(1)}) = \ell\,(v)$, gdje je v ulazna vrijednost, dobivamo

$$\begin{aligned} \mathrm{Compl}_A(\mathsf{IGCD}) \ &= \ \sum_{i=1}^{D} \mathrm{Compl}_A(w_{(i)} \bmod v_{(i)}) \\ &\leq \ c \cdot \ell\,(v) \cdot \sum_{i=1}^{D} \ \ell\ \left(\left\lfloor \frac{w_{(i)}}{v_{(i)}} \right\rfloor \right) \ . \end{aligned}$$

Očito je iz (1.8.4)

$$\sum_{i=1}^{D} \ \ell\ \left(\left\lfloor \frac{w_{(i)}}{v_{(i)}} \right\rfloor \right) \leq \ell\ \left(\left\lfloor \frac{w_{(1)}}{v_{(D)}} \right\rfloor \right) \ .$$

Kako je $\ell\,(w_{(1)}) = \ell\,(u)$, $v_{(D)} = w$, gdje je $w = \gcd(u, v)$, to je

$$\mathrm{Compl}_A(\mathsf{IGCD}) \leq c \cdot \ell\,(v) \cdot [\ell\,(u) - \ell\,(w) + 1] \ . \tag{1.8.5}$$

Pri tome treba uzeti da je $\ell\,(u) \geq \ell\,(v)$, jer, u protivnom, prvi korak algoritma zamjenjuje $w = |u|$ i v.

Gornja ograda u relaciji (1.8.5) sa može dostići, pa je

$$\begin{aligned} \mathrm{w} - \ \mathrm{Compl}_A(\mathsf{IGCD}) \ &\approx \ c \cdot \min \{\ \ell\,(u)\ ,\ \ell\,(v)\ \} \cdot \\ &\cdot \ [\max \{\ \ell\,(u)\ ,\ \ell\,(v)\ \} - \ell\,(w) + 1] \ . \end{aligned} \tag{1.8.6}$$

Detaljna analiza Euklidovog algoritma i nekih drugih algoritama za nalaženje najveće zajedničke mjere je dana u [Knuth, 1981.].

Za realizaciju aritmetičkih algoritama, pretpostavljamo da je tip *mprat* neka realizacija zapisa (1.8.1), na pr.

```
mprat = record
            num   : mpint;
            denom : mpnat
        end;
```

Algoritme za racionalnu aritmetiku formuliramo u skraćenom obliku, naznačavanjem operacija za brojnike i nazivnike. Za precizan zapis, trebali bi realizirati aritmetičke algoritme za prirodno-cjelobrojnu aritmetiku, kojoj je jedan operand tipa *mpnat*, a drugi tipa *mpint*. Realizacija tih algoritama je sasvim jednostavna, pa ih nećemo posebno navoditi.

Algoritam 1.8.2. (QADD – **zbrajanje racionalnih brojeva)**

Ulaz : *racionalni brojevi* $u = (u_1\ ,\ u_2)$ $v = (v_1\ ,\ v_2)$ *u pozicionom zapisu u bazi b.*

Izlaz : *racionalni broj* $w = (w_1\ ,\ w_2)$

$$w = u + v$$

u pozicionom zapisu u bazi b.

procedure QADD *(u , v : mprat* **var** *w : mprat);*

```
begin
        { pribrojnik = 0 ? }
   if u = 0 then
      w ← v
   else if v = 0 then
      w ← u
   else
      begin
      d ← gcd(u2, v2);
      if d = 1 then
         begin
         w1 ← u1 · v2 + v1 · u2;
         w2 ← u2 · v2
         end
      else
         begin
               { elimininiraj zajednički faktor nazivnika }
         u'2 ← u2/d;
         v'2 ← v2/d;
         w'1 ← u1 · v'2 + v1 · u'2;
         w'2 ← u2 · v'2;
         if w'1 = 0 then
            w ← (0 , 1)
         else { skrati sumu }
            begin
            e ← gcd(w'1, d);
            if e = 1 then
               begin
               w1 ← w'1;
               w2 ← w'2
               end
            else
               begin
```

```
                w1 ← w'1/e;
                w2 ← w'2/e
                end
            end
        end
    end
end;  { QADD }
```

Algoritam je dug, jer pedantno provjeravamo eventualne zajedničke faktore, i izbjegavamo dijeljenja, ako je faktor 1. Osim toga, sve nule posebno tretiramo.

Korektnost algoritma je direktna posljedica relacije

$$\frac{u_1}{u_2} + \frac{v_1}{v_2} = \frac{u_1 \cdot v_2 + v_1 \cdot u_2}{u_2 \cdot v_2}$$

i neskrativosti ulaznih brojeva.

Ako je $d = \gcd(u_2, v_2)$ i $u'_2 = u_2/d$, $v'_2 = v_2/d$, onda je

$$\frac{u_1}{u_2} + \frac{v_1}{v_2} = \frac{u_1 \cdot v'_2 + v_1 \cdot u'_2}{u'_2 \cdot d \cdot v'_2}$$

s tim da je $u' \cdot d \cdot v' = u_2 \cdot v'_2$. Tada je

$$\gcd(u_1 \cdot v'_2 + v_1 \cdot u'_2, u_2 \cdot v'_2) = \gcd(u_1 \cdot v'_2 + v_1 \cdot u'_2, d) \ .$$

Ove relacije maksimalno smanjuju veličinu brojeva u algoritmu, a time i složenost.

Pošto koristimo množenja i dijeljenja, složenost nije linearno ovisna o duljini brojeva. Može se pokazati da je

$$\mathrm{w} - \mathrm{Compl}_A(\mathsf{QADD}) \approx \ell\,(m) \cdot m \cdot n^{-1} \cdot \mathrm{Mul}(n) \ , \tag{1.8.7}$$

gdje je

$$\begin{aligned} m &= \max \{\ \ell\,(u)\ ,\ \ell\,(v)\ \} \\ n &= \min \{\ \ell\,(u)\ ,\ \ell\,(v)\ \} \end{aligned}$$

[Collins, Mignotte, Winkler, 1982.].

Pri tome je duljina $\ell\,(q)$, racionalnog broja q definirana sa

$$\ell\,(q) = \max \{\ \ell(q_1)\ ,\ \ell\,(q_2)\ \} \ .$$

Algoritam 1.8.3. (QSUB – **oduzimanje racionalnih brojeva)**

Ulaz : *racionalni brojevi* $u = (u_1\ ,\ u_2)$ $v = (v_1\ ,\ v_2)$ *u pozicionom zapisu u bazi b.*

Izlaz : *racionalni broj* $w = (w_1\ ,\ w_2)$

$$w = u - v$$

u pozicionom zapisu u bazi b.

procedure QSUB *(u , v : mprat* **var** *w : mprat);*

begin
sign$(u_1) \leftarrow -$ sign(u_1)*;*
QUADD *(u , v , w)*
end; { QSUB }

Očito je

$$\mathrm{Compl}_A(\mathsf{QSUB}) = \mathrm{Compl}_A(\mathsf{QADD})\ .$$

Algoritam za nalaženje produkta $w = u \cdot v$ je baziran na relaciji

$$\frac{u_1}{u_2} \cdot \frac{v_1}{v_2} = \frac{u_1 \cdot v_1}{u_2 \cdot v_2}\ ,$$

s tim da treba skratiti u_1 i v_2, te v_1 i u_2, ako je to moguće. Nakon toga kraćenje je nemoguće.

Algoritam 1.8.4. (QMUL – množenje racionalnih brojeva)
Ulaz : *racionalni brojevi* $u = (u_1\ ,\ u_2)$ $v = (v_1\ ,\ v_2)$ *u pozicionom zapisu u bazi b.*

Izlaz : *racionalni broj* $w = (w_1\ ,\ w_2)$

$$w = u \cdot v$$

u pozicionom zapisu u bazi b.

```
procedure QMUL ( u , v : mprat  var  w : mprat);

begin
   if (u = 0) or (v = 0) then
      w ← (0 , 1)
   else
      begin
      d1 ← gcd(u1, v2);
      d2 ← gcd(v1, u2);
      if d1 = 1 then
         begin
         u'1 ← u1;
         v'2 ← v2
         end
      else
         begin
         u'1 ← u1/d1;
         v'2 ← v2/d1
         end;
      if d2 = 1 then
         begin
         v'1 ← v1;
         u'2 ← u2
         end
      else
         begin
         v'1 ← v1/d2;
         u'2 ← u2/d2
         end;
        { produkt }
      w1 ← u'1 · v'1;
      w2 ← u'2 · v'2
      end
end; { QMUL }
```

Kao i kod zbrajanja, vrijedi

$$\mathrm{Compl}_A(\mathsf{QMUL}) \approx \ell\,(m) \cdot m \cdot n^{-1} \cdot \ \mathrm{Mul}(n)\ .$$

Dijeljenje racionalnih brojeva je trivijalno, jer je za $q = (q_1\ ,\ q_2)\ \ ,\ \ q_1 \neq 0$

$$\frac{1}{q} = (\mathrm{sign}(q_1) \cdot q_2,\ \mathrm{abs}(q_1))\ .$$

Algoritam 1.8.5. (QDIV – **dijeljenje racionalnih brojeva**)
Ulaz : *racionalni brojevi* $u = (u_1\ ,\ u_2)\quad v = (v_1\ ,\ v_2)$ *u pozicionom zapisu u bazi b, uz pretpostavku* $v \neq 0$.

Izlaz : *racionalni broj* $w = (w_1\ ,\ w_2)$

$$w = \frac{u}{v}$$

u pozicionom zapisu u bazi b.

procedure QDIV *(u , v : mprat* **var** *w : mprat);*

begin
sign(v'_1) $\leftarrow$ sign(v_1)*;*
abs(v'_1) $\leftarrow v_2$*;*
$v'_2 \leftarrow$ abs(v_1)*;*
QMUL *(u , v′ , w)*
end; { QDIV }

Očito je

$$\mathrm{Compl}_A(\mathsf{QDIV}) = \mathrm{Compl}_A(\mathsf{QMUL})\ .$$

Ovo pokazuje da sve aritmetičke operacije imaju sličnu vremensku složenost.

Za razliku od prirodne i cjelobrojne aritmetike, i uspoređivanje može zahtijevati 2 moženja, zbog

$$\frac{u_1}{u_2} < \frac{v_1}{v_2} \iff u_1 \cdot v_2 < u_2 \cdot v_1$$

uz korištenje uspoređivanja cijelih brojeva.

1.9. Realna aritmetika

1.9.1. Prikaz realnih brojeva

Za realne brojeve vrijedi sljedeća generalizacija teorema 1.1.1. , koju dajemo bez dokaza.

Teorem 1.9.1.

Neka je $b \in \mathbb{N}\ ,\ b \geq 2$ *bilo koji prirodni broj. Za svaki pozitivni realni broj* $u \in \mathbb{R}^+$ *, postoje* $n \in \mathbb{Z}$ *i niz brojeva* $u_i\ ,\ i \in \mathbb{Z}\ ,\ i \leq n$ *, takvi da je :*

$$u\ =\ \sum_{i=-\infty}^{n} u_i\, b^i \tag{1.9.1}$$

Uz dodatna ograničenja na niz u_i ; $u_i \in \mathbb{N}$, $\forall i$, *te*

$$0 \le u_i < b \quad , \quad i \in \mathbb{Z} \ , \ i < n$$
$$0 < u_n < b$$

i ako za svaki $i_0 \in \mathbb{Z}$, $i_0 < n$ *postoji* $i \in \mathbb{Z}$ $i < i_0$, *takav da je* $u_i < b-1$, *onda je prikaz (1.9.1) jednoznačan.* ■

Posljednje ograničenje ne dozvoljava da, počev od nekog mjesta i_0, sve znamenke u_i , $i \le i_0$ budu jednake $b-1$.

Za negativne brojeve treba, na pr. dodati predznak reprezentaciji (1.9.1), dok za nulu treba definirati prikladan izraz.

Reprezentacija (1.9.1) se očito ne može koristiti u praksi, jer niz znamenki u_i može sadržavati beskonačno monogo pozitivnih znamenki.

Zbog toga, u realnoj aritmetici, za prikaz brojeva uzimamo konačan broj znamenki u (1.9.1). Posljedice toga su dalekosežne, jer dobiveni skup prikazivih brojeva ne mora biti zatvoren obzirom na aritmetičke operacije. To rezultira pojavom grešaka zaokruživanja, ako rezultat svake aritmetičke operacije mora biti prikaziv.

Pošto realiziramo aritmetiku visoke točnosti, fenomenu grešaka zaokruživanja posvećujemo samo najnužniju pažnju, dovoljnu za realizaciju aritmetičkih algoritama.

Postoje dva standardna načina skraćenja prikaza (1.9.1) u konačnu sumu.

(a) Fiksiramo donju granicu u toj sumi, na pr. na $p \in \mathbb{Z}$. Prikazivi brojevi tada imaju oblik

$$u = \sum_{i=p}^{n} u_i \, b^i \ .$$

Ovo je tzv. prikaz s fiksnom točkom u bazi b ("fixed point" prikaz), jer se točka u pripadnom pozicionom zapisu nalazi p mjesta ispred zadnje znamenke.

Pripadnu aritmetiku možemo nazvati **aritmetika apsolutne točnosti**, jer je apsolutna pogreška odozgo ograđena, pri aproksimaciji bilo kog broja nekim prikazivim brojem.

Pomnožimo li sve prikazive brojeve s 2^p, dobivamo upravo skup prirodnih brojeva. Zbog toga se pripadna aritmetika lako realizira, uz neki odabrani algoritam zaokruživanja za množenje i dijeljenje.

Ovaj prikaz se vrlo često realizira tako da je i gornja granica n fiksna, obično je $n = -1$. Tada je

$$u = \sum_{i=p}^{-1} u_i \, b^i = (0.u_{-1} \dots u_p) \ , \tag{1.9.2}$$

pa za prikazive brojeve vrijedi $0 < u < 1$.

(b) Fiksiramo broj znamenki u prikazu (1.9.1) na $p \in \mathbb{N}$. Prikazivi brojevi su oblika

$$u = \sum_{i=n-p+1}^{n} u_i\, b^i = b^{n+1} \cdot \sum_{i=-p}^{-1} u_i\, b^i \tag{1.9.3}$$

Ovaj se prikaz obično zove prikaz s pomičnom točkom u bazi b ("floating point"). Uočimo li da pamtimo najznačajnije znamenke broja, u pripadnoj aritmetici je relativna pogreška odozgo ograđena, pri aproksimaciji bilo kog broja prikazivim brojem. Zbog toga je mnogo pogodniji naziv **aritmetika relativne točnosti** (odosno prikaz relativne točnosti).

Uočimo sličnost između prikaza (1.9.2) i (1.9.3). Jedina razlika je u varijabilnoj potenciji baze za (1.9.3).

Realnu aritmetiku ćemo realizirati za "floating point" prikaz, pošto on uključuje i prikaz (1.9.2), ako stavimo $n = -1$.

Da izbjegnemo rad s negativnim ideksima, definirajmo normalizirani "floating point" prikaz na sljedeći način.

Definicija 1.9.1.

Neka je $u \in \mathbb{R}$, $u \neq 0$ *bilo koji broj i neka je* $b \in \mathbb{N}$, $b \geq 2$ *odabrana baza prikaza. Tada postoje jednoznačno određeni brojevi* $s_u \in \{ -1 , 1 \}$, $m_u \in \mathbb{R}$, $e_u \in \mathbb{Z}$ *takvi da je*

$$u = s_u\, m_u\, b^{e_u} \tag{1.9.4}$$

i

$$\frac{1}{b} \leq m_u < 1 \ .$$

Ovaj prikaz zovemo **normalizirani** *prikaz broja* u *u bazi* b*.*

Pri tome je s_u **predznak** *broja* u*,* m_u *je* **mantisa***, a* e_u **eksponent** *baze za broj* u*.*

U praktičnoj realizaciji za broj m koristimo konačni broj znamenki, tj. prikaz oblika (1.9.2). Taj broj znamenki mantise je obično fiksan za sve brojeve i zadan je unaprijed kao konstanta n. Tada m_u prikazujemo u obliku

$$m_u = \sum_{i=1}^{n} u_i\, b^{-i}$$

sa

$$\begin{aligned} 0 \ &< \ u_1 \ < \ b \\ 0 \ &\leq \ u_i \ < \ b \quad , \quad i = 2, \ldots, n \ . \end{aligned} \tag{1.9.5}$$

i m prikazujemo nizom njegovih znamenki u bazi b.

$$(u_1, \ldots, u_n) \ .$$

Za eksponent, u praksi, možemo uzeti prikazivi cijeli broj u računalu, tj. za prikazivi broj u je

$$-maxint \leq e \leq maxint \ .$$

Broj $0 \in \mathbb{R}$ prikazujemo u obliku

$$s_0 = 0 \ \ , \ \ m_0 = 0 = \underbrace{(0, \ldots, 0)}_{n \text{ puta}} \ \ , \ \ e_0 = 0 \ .$$

Tip *mpreal* kojim realiziramo taj prikaz, možemo definirati sa

```
digits  =  array [1..n] of integer;
mpreal  =  record
               sign  : integer;  { s }
               digit : digits;   { m }
               bexp  : integer;  { e }
           end;
```

Aritmetiku takvih brojeva realiziramo tako da svaka operacija uvijek daje normalizirani rezultat, tj. vrijedi (1.9.5), ako rezultat nije nula. Pri tome koristimo zaokruživanje na osnovu svih izračunljivih znamenki mantise rezultata (tzv. dvostruki akumulator – v. [Wilkinson, 1963.]). Tada se može pokazati da zaokruženi rezultat $fl\,(u \circ v)$ zadovoljava relaciju.

$$\begin{aligned} fl\,(u \circ v) &= (u \circ v)\,(1 + \varepsilon) \\ |\varepsilon| &< b^{-n+1} \end{aligned} \quad , \tag{1.9.6}$$

tj. ima malu relativnu pogrešku.

Pedantnije zaokruživanje nije potrebno, jer ionako radimo u aritmetici visoke točnosti, pa faktor $1/2$ kojeg daje "pravo" zaokruživanje u relaciji (1.9.6), nema bitnog utjecaja.

Također, pretpostavimo da je $b^2 - 1 \leq maxint$.

Neka su u i v brojevi oblika (1.9.4) i pretpostavimo da je $u \geq v > 0$. Tada je

$$u = m_u\, b^{e_u} \quad , \quad v = m_v\, b^{e_v} \tag{1.9.7}$$

Za zbrajanje znamenki mantisa, prvo treba poravnati točku u bazi b.

$$u + v = (m_u + m_v \cdot b^{e_v - e_u}) \cdot b^{e_u} \ ,$$

s tim da rezultat ne mora biti normaliziran, jer je

$$m_u + m_v \cdot b^{e_v - e_u} < 2 \ .$$

Normalizaciju provodimo množenjem potencijom baze, tj. pomakom.

Algoritam 1.9.1. (ADD1 – **zbrajanje pozitivnih realnih brojeva**)
Ulaz : *nizovi znamenki mantisa* $(u_1, \ldots, u_n) = m_u$, $(v_1, \ldots, v_n) = m_v$ *i eksponenti baze* e_u, e_v *brojeva* u, v, *uz pretpostavku* $u \geq v > 0$.

Izlaz : *niz znamenki mantise* $(w_1, \ldots, w_n) = m_w$ *i eksponent baze* e_w *broja*

$$w = m_w\, b^{e_w} = fl\,(u + v) \ .$$

```
procedure ADD1 ( m_u , m_v : digits ;  e_u , e_v : integer ;
                 var m_w : digits ; var e_w : integer);

begin
  e_w ← e_u;
  d ← e_u − e_v;
  if d < n then
      { zbroji mantise }
    temp ← u_n + v_{n−d};
    for i ← n − 1 downto d + 1 do
      if temp < b then
        begin
        w_{i+1} ← temp;
        temp ← u_i + v_{i−d}
        end
      else  { temp ≥ b }
        begin
        w_{i+1} ← temp − 1;
        temp ← u_i + v_{i−d} + 1
        end;
    ok ← (temp < b);
    if ok then
      w_d ← temp
    else
      begin
      w_d ← temp − b;
      if d > 1 then
        begin
        d ← d − 1;
        while (u_d = b − 1) and (d > 1) do
          begin
          w_d ← 0;
          d ← d − 1
          end;
        ok ← (u_d < b − 1);
        if ok then
```

```
        w_d ← u_d + 1
    else
        w_1 ← 0
    end
end;
```

```
      if ok then
         for i ← 1 to d − 1 do w_i ← u_i
      else
         begin { shift }
         for i ← n downto 2 do w_i ← w_{i−1};
         w_1 ← 1;
         e_w ← e_w + 1
         end
      end
   else
      for i ← 1 to n do w_i ← u_i   { d ← n + 1 }
end; { ADD1 }
```

Za složenost dobivamo iste rezultate kao i za algoritam NADD1, uz $\ell(u) = n$, $\ell(v) = \max\{0\ ,\ n-d\}$ (v. (1.2.17), (1.2.18), prop. 1.2.5.).

Ako su u i v dani sa (1.9.7), onda je

$$u - v = (m_u - m_v \cdot b^{e_v - e_u}) \cdot b^{e_u} \ .$$

Rezultat opet ne mora biti normaliziran, ako dolazi do kraćenja.

Algoritam 1.9.2. (SUB1 – **oduzimanje pozitivnih realnih brojeva)**

Ulaz : *nizovi znamenki mantisa* $(u_1, \ldots, u_n) = m_u$, $(v_1, \ldots, v_n) = m_v$ *i eksponenti baze* e_u, e_v *brojeva* u, v, *uz pretpostavku* $u > v > 0$.

Izlaz : *niz znamenki mantise* $(w_1, \ldots, w_n) = m_w$ *i eksponent baze* e_w *broja*

$$w = m_w\, b^{e_w} = fl\,(u - v) \ .$$

```
procedure SUB1 ( m_u , m_v : digits ;  e_u , e_v : integer ;
                 var m_w : digits ; var e_w : integer);

begin
  e_w ← e_u;
  d ← e_u − e_v;
  if d ≥ n + 2 then
    for i ← 1 to n do w_i ← u_i
  else
    begin
    if d = n + 1 then
      t ← −1
    else if d ≥ 2 then
      begin
      i ← −d + 2;
      while (v_i = 0) and (i < n) do
        i ← i − 1;
      if v_i > 0 then
        t ← −v_{n−d+1} − 1
      else
        t ← −v_{n−d+1}
      end
    else if d = 1 then
      t ← −v_n
    else
      t ← 0;
    carry ← (t < 0);
    if carry then
      t ← t + b;
    if d < n then
      begin
      temp ← u_n − v_{−d};
      if carry then temp ← temp − 1;
      for i ← n − 1 downto d + 1 do
        if temp ≥ 0 then
          begin
          w_{i+1} ← temp;      temp ← u_i − v_{i−d}
          end
        else
          begin
          w_{i+1} ← temp + b;      temp ← u_i − v_{i−d} − 1
          end;
```

```
        carry ← (temp < 0);
        if not carry then
            w_d ← temp
        else
            w_d ← temp + b;
        end;
    if carry then { d ≥ 1 }
        begin
        if d > 1 then
            begin
            d ← d − 1;
            while (u_d = 0) and (d > 1) do
                begin
                w_d ← b − 1;
                d ← d − 1
                end;
            end;
        u_d ← u_d − 1;
        end;
    if (d = 1) and (w_d = 0) then
        begin
        repeat
            d ← d + 1
        until w_d ≠ 0;
        d ← d − 1;
        for i ← 1 to n − d do
            w_i ← w_{i+d};
        w_{d+1} ← t;
        for i ← d + 2 to n do w_i ← 0;
        e_w ← e_w − d
        end
    else
        for i ← 1 to d − 1 do w_i ← u_i
    end
end; { SUB1 }
```

Ovaj algoritam je vrlo kompliciran zato što prijenose računa samo onda kada je to zaista potrebno. Na primjer, za slučaj $d \geq 2$, direktno nalazimo prijenos na prvom mjestu iza zadnje znamenke mantise m_u. Također moramo pamtiti znamenku t koja nastaje na tom mjestu, zbog eventualnog kraćenja. Korektnost algoritma izlazi iz razmatranja vezana uz algoritam NSUB. Aritmetička složenost je jednaka složenosti algoritma NSUB, uz $\ell(v) = \max\{ 0 , n - d \}$.

Da izbjegnemo ograničenje pozitivnosti, potrebno je prvo usporediti brojeve.

Algoritam 1.9.3. (COMP – **uspoređivanje realnih brojeva**)
Ulaz : *realni brojevi* u, v *u normaliziranom zapisu.*

Izlaz : *broj* $sign$ *definiran sa :*

$$sign = \text{sign}(u - v) \ .$$

```
procedure COMP ( u , v : mpreal ; var sign : integer);

begin
   if s_u > s_v then
      sign ← 1
   else if s_u < s_v then
      sign ← −1
   else if s_u ≠ 0 then
      begin
      if e_u > e_v then
         sign ← 1
      else if e_u < e_v then
         sign ← −1
      else
         begin
         i ← 1;
         while (u_i = v_i) and (i < n) do i ← i − 1;
         if u_i > v_i then
            sign ← 1
         else if u_i < v_i then
            sign ← −1
         else sign ← 0;
      if s_u < 0 then sign ← −sign
      end
   else
      sign ← 0
end; { COMP }
```

Algoritam 1.9.4. (ADD – **zbrajanje realnih brojeva**)
Ulaz : *realni brojevi* u, v *u normaliziranom zapisu.*

Izlaz : *realni broj*

$$w = fl\,(u + v)$$

u normaliziranom zapisu.

```
procedure ADD ( u , v : mpreal ; var w : mpreal);

begin
   if s_u = 0 then
      w ← v
   else if s_v = 0 then
      w ← u
   else if s_u = s_v then
      begin
      COMP (u , v , sign);
      if sign ≥ 0 then
         ADD1 (m_u , m_v , e_u , e_v , m_w , e_w)
      else
         ADD1 (m_v , m_u , e_v , e_u , m_w , e_w);
      s_w ← 1
      end
   else
      begin
      COMP (u , v , sign);
      if sign > 0 then
         begin
         SUB1 (m_u , m_v , e_u , e_v , m_w , e_w);
         s_w ← s_u
         end
      else if sign < 0 then
         begin
         SUB1 (m_v , m_u , e_v , e_u , m_w , e_w);
         s_w ← s_v
         end
      else
         s_w ← 0
      end
end; { ADD }
```

Oduzimanje realiziramo ili slično zbrajanju, ili promjenom znaka broja v i zbrajanjem, kao u algoritmu ISUB.

Ako su u, v brojevi ublika (1.9.4) i ako je $u \neq 0$ i $v \neq 0$, onda je

$$u \cdot v = s_u \cdot s_v \cdot (m_u \cdot m_v) \cdot b^{e_u + e_v} \ .$$

I ovdje treba paziti na normalizaciju, jer je

$$1/b^2 \leq m_u \cdot m_v < 1 \ .$$

Za spremanje znamenki broja $m_u \cdot m_v$ koristimo akumulator – broj duljine $2n$, oblika

$$acc = (0.acc_1, \ldots, acc_{2n}) \ ,$$

kojeg možemo realizirati kao globalnu varijablu *acc* za sve operacije. Koristimo klasični algoritam za množenje.

Algoritam 1.9.5. (MUL – **množenje realnih brojeva**)
Ulaz : *realni brojevi u, v u normaliziranom zapisu.*
Izlaz : *realni broj*

$$w = fl\,(u \cdot v)$$

u normaliziranom zapisu.

```
procedure MUL ( u , v : mpreal ; var w : mpreal);

begin
  s_w ← s_u · s_v;
  if s_w ≠ 0 then
    begin
    e_w ← e_u + e_v;
    for i ← 1 to 2n do acc_i ← 0;
    for j ← n downto 1 do
      begin
      carry ← 0;
      for i ← n downto 1 do
        begin
        temp ← acc_{i+j} + u_i · v_j + carry;
        acc_{i+j} ← temp mod b;
        carry ← temp div b
        end
      acc_{i+j-1} ← carry
      end;
    if acc_1 > 0 then
      for i ← 1 to n do
        w_i ← acc_i
    else
      begin
      e_w ← e_w − 1;
      for i ← 1 to n do
        w_i ← acc_{i+1}
      end
    end
end; { MUL }
```

Korektnost algoritma je trivijalna, a složenost je kao za algoritam NMUL, uz $\ell(u) = \ell(v) = n$.

Za vrlo velike duljine mantisa možemo koristiti i brže algoritme za množenje mantisa. Algoritmi za množenje prirodnih brojeva se trivijalno modificiraju za množenje brojeva u zapisu s fiksnom točkom, jer treba samo korektno transformirati indekse znamenki.

Kod dijeljenja realnih brojeva u , $v \in \mathbb{R}$, $v \neq 0$ u zapisu (1.9.4), dobivamo

$$\frac{u}{v} = \frac{s_u}{s_v} \left(\frac{m_u}{m_v} \right) \cdot b^{e_u - e_v}$$

uz

$$\frac{1}{b} < \frac{m_u}{m_v} < b \ ,$$

pa treba paziti na normalizaciju.

Za nalaženje m_u/m_v možemo koristiti metodu "podijeli i korigiraj" u varijanti za brojeve u zapisu s fiksnom točkom. Tu varijantu dobivamo direktno iz algoritma NDIV, ako stavimo

$$\begin{aligned} M_u &= b^{2n} \cdot m_u \\ M_v &= b^n \cdot m_v \end{aligned}$$

pa su M_u, M_v očito prirodni brojevi.

Njihov kvocijent

$$M_w = \lfloor M_u / M_v \rfloor$$

ima n ili $n+1$ znamenku, pa definiramo

$$m_w = \begin{cases} M_w / b^n & , \text{ ako } \ \ell(M_w) = n \\ \lfloor M_w / b \rfloor / b^n & , \text{ ako } \ \ell(M_w) = n+1 \end{cases}$$

Algoritam dajemo u skraćenom zapisu.

Algoritam 1.9.6. (DIV – **dijeljenje realnih brojeva)**

Ulaz : *realni brojevi u, v u normaliziranom zapisu, uz pretpostavku* $v \neq 0$.

Izlaz : *realni broj*

$$w = fl(u/v)$$

u normaliziranom zapisu.

procedure DIV *(u , v : mpreal ;* **var** *w : mpreal);*

begin

```
   if  s_v = 0  then
      ERROR
   else  if  s_u = 0  then
      s_w ← 0
   else
      begin
      s_w ← s_u · s_v;
      e_w ← e_u − e_v;
      M_u ← b^{2n} · m_u;
      M_v ← b^n · m_v;
      M_w ← ⌊M_u/M_v⌋;
      if  ℓ(M_w) = n  then
         m_w ← M_w/b^n
      else
         begin
         m_w ← ⌊M_w/b⌋/b^n;
         e_w ← e_w + 1
         end
      end
end;  { DIV }
```

Složenost ovog algoritma je jednaka složenosti odabranog algoritma NDIV za nalaženje cjelobrojnog kvocijenta. Množenje i dijeljenje potencijom baze se svode na pomake. Najefikasnije je algoritam realizirati korištenjem akumulatora, tako da na početku stavimo

$$acc \leftarrow m_u$$

i sve operacije izvodimo direktno u akumulatoru, koristeći normalizirani divizor m_v.

Primijetimo da ovdje nije potrebno naći ostatak, pa otpada dijeljenje ostatka normalizacionim faktorom.

Druga mogućnost za realizaciju dijeljenja mantisa, je korištenje Newtonove metode za nalaženje $1/m_v$. Za razliku od algoritma NREC, ovdje koristimo Newtonovu metodu bez dodatnih korekcija. Kako je

$$1 < \frac{1}{m_v} \leq b$$

to je za prikaz aproksimacija $w_{(k)}$ za $1/m_v$ potrebno predvidjeti još dvije vodeće znamenke.

Pretpostavimo da početna aproksimacija $w_{(0)}$ ima p_0 točnih znamenki desno od točke u bazi b

$$\left| \frac{1}{m_v} - w_{(0)} \right| < b^{-p_0}.$$

Takva aproksimacija se lako nalazi korištenjem aritmetike računala. Iz relacije (1.6.33) izlazi da aproksimacija $w_{(1)}$ po Newtonovoj metodi ima barem $2p_0$ točnih znamenki, uz egzaktnu aritmetiku.

Ako aproksimacija $w_{(k)}$ ima p_k točnih znamenki iza točke u bazi b, izlazi da je dovoljno u relaciji

$$w_{(k+1)} = w_{(k)} \cdot (2 - m_v \cdot w_{(k)})$$

sve operacije izvesti s točnošću od $2p_k$ znamenki. Tada i $w_{(k)}$ ima točnost od približno $2p_k$ znamenki iza točke u bazi b. Kako je Newtonova metoda samokorigirajuća, to ne narušava kvadratnu konvergenciju.

Ove iteracije provodimo koristeći akumulator, sve dok ne dobijemo $p_k > n$. Broj potrebnih koraka metode je

$$k \approx \lg \frac{n}{p_0} \ ,$$

a složenost ovog postupka je proporcionalna s $\mathrm{Mul}(n)$, zbog varijabilne preciznosti.

A. Složenost i asimptotsko ponašanje funkcija

A.1. Složenost algoritama

Algoritam je postupak za rješenje nekog problema. Za uspoređivanje algoritama potrebno je uvesti funkcije koje opisuju ponašanje algoritma u ovisnosti o veličini zadaće — ulazom zadanog konkretnog problema. Takve funkcije zovemo **složenost algoritma**.

Ovi pojmovi mogu se i precizno definirati u okviru teorije automata i algoritama. U praksi, složenost algoritma znači količinu sredstava (resursa) koje algoritam koristi pri izvršavanju na određenoj arhitekturi računala. U analizi aritmetičkih algoritama obično nas zanimaju tri osnovne vrste složenosti:

prostorna složenost = količina potrebne memorije (za podatke) u nekim osnovnim jedinicama. Najčešća jedinica u analizi aritmetičkih algoritama je riječ = memorija potrebna za prikaz brojeva u aritmetici računala;

vremenska složenost = potrebno vrijeme za izvođenje algoritma, u nekim osnovnim jedinicama. Obično se koristi osnovni ciklus računala ili prosječno trajanje osnovnih instrukcija;

aritmetička složenost = broj osnovnih aritmetičkih operacija računala potreban za izvođenje algoritma.

Ove pojmove posebno označavamo s

$$\begin{aligned} \mathrm{Compl}_S &= \text{prostorna složenost}, \\ \mathrm{Compl}_T &= \text{vremenska složenost}, \\ \mathrm{Compl}_A &= \text{aritmetička složenost}, \end{aligned}$$

i koristimo ih kao funkcije definirane na algoritmima. Njihove vrijednosti izražavamo u terminima veličine ulaznih podataka algoritma. Veličina podataka je broj znakova potrebnih za "razumno" kodiranje tih podataka.

Prostorna, a posebno vremenska složenost ovise o jedinicama u kojima ih mjerimo. Promjena jedinica, međutim, mijenja složenost za konstantni faktor, pa nema utjecaja na uspoređivanje algoritama.

Napomena A.1.1. *Prostorna složenost, osim ulaznih podataka, uključuje i sve međurezultate koji se javljaju tijekom izvođenja algoritma. Može se pokazati da je ona uvijek bitno manja od vremenske, odnosno, aritmetičke složenosti. Zbog toga ćemo prostornoj složenosti posvetiti mnogo manje pažnje i obično je nećemo navoditi.*

Napomena A.1.2. *Aritmetička složenost gotovo ne ovisi o arhitekturi računala, za razliku od vremenske, pa je mnogo pogodnija za opću analizu aritmetičkih algoritama. Na sekvencijalnim arhitekturama, te dvije složenosti su sličnog reda veličine, dok paralelno izvođenje aritmetičkih operacija može bitno smanjiti vremensku složenost.*

Ovisnost vremenske složenosti o jedinicama je samo dodatni razlog da koristimo aritmetičku složenost za analizu aritmetičkih algoritama.

A.2. Relacije asimptotskog ponašanja funkcija

Algoritme uspoređujemo, uglavnom, po njihovom ponašanju za velike probleme. Za analizu tog asimptotskog ponašanja uvodimo sljedeće oznake.

Definicija A.2.1.

Neka su f, $g : D \to \mathbb{R}$ realne funkcije na odozgo neograničenom podskupu $D \subseteq \mathbb{R}$.

(a) *f je* **manjeg reda veličine** *od g, ili f* **raste sporije** *od g, u oznaci*

$$f(x) \in o(g(x)) \quad (x \to \infty),$$

ako postoji

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} \quad \textit{i jednak je } 0.$$

(b) *f* **nije većeg reda veličine** *od g, ili f* **ne raste brže** *od g, u oznaci*

$$f(x) \in O(g(x)) \quad (x \to \infty),$$

ako $\exists C \in \mathbb{R}$ i $\exists x_0 \in D$, takvi da je

$$|f(x)| < C\,|g(x)|, \quad \forall x > x_0.$$

(c) *f* **je istog reda veličine** *kao i g, ili f* **raste istom brzinom** *kao i g, u oznaci*

$$f(x) \in \Theta(g(x)) \quad (x \to \infty),$$

ako postoje realne konstante $c_1 > 0$, $c_2 > 0$, i $\exists x_0 \in D$, takvi da je

$$c_1 \, |g(x)| < |f(x)| < c_2 \, |g(x)|, \quad \forall x > x_0.$$

(d) *f i g su* **asimptotski jednake**, *u oznaci*

$$f(x) \sim g(x) \quad (x \to \infty),$$

ako je

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = 1.$$

(e) *f* **nije manjeg reda veličine** *od g, ili f* **raste barem jednako brzo** *kao i g, u oznaci*

$$f(x) \in \Omega(g(x)) \quad (x \to \infty),$$

ako **nije** *$f(x) \in o(g(x))$. Uz pretpostavku da je $g(x) \neq 0$ za sve dovoljno velike x, to znači da postoje $\varepsilon > 0$ i niz $x_n \in D$, $n \in \mathbb{N}$, $x_n \to \infty$, takvi da je*

$$|f(x_n)| > \varepsilon \, |g(x_n)|, \quad \forall n \in \mathbb{N}.$$

(f) *f je* **većeg reda veličine** *od g, ili f* **raste brže** *od g, u oznaci*

$$f(x) \in \omega(g(x)) \quad (x \to \infty),$$

ako **nije** *$f(x) \in O(g(x))$. Uz pretpostavku da je $g(x) \neq 0$ za sve dovoljno velike x, to znači da postoje nizovi $\varepsilon_n > 0$, $x_n \in D$, $n \in \mathbb{N}$, takvi da*

$$\varepsilon_n \to \infty, \quad x_n \to \infty,$$

i da je

$$|f(x_n)| \geq \varepsilon_n \, |g(x_n)|, \quad \forall n \in \mathbb{N}.$$

Napomena A.2.1. *Nas zanima ponašanje složenosti aritmetičkih operacija za velike brojeve. Zbog toga su relacije asimptotskog ponašanja u definiciji* A.2.1. *definirane oko točke ∞, iako se definicija može poopćiti na bilo koje gomilište domene.*

Kod navođenja asimptotskih relacija, oznaku $(x \to \infty)$ najčešće ćemo ispuštati, podrazumjevajući ponašanje za velike argumente.

Napomena A.2.2. *Lako se dokazuje da su relacije asimptotskog ponašanja Θ i $\sim$ relacije ekvivalencije na skupu $\mathbb{R}^D$ svih funkcija $f : D \to \mathbb{R}$.*

Na skupu klasa ekvivalencije $\mathbb{R}^D/_\Theta$ relacije o i O su relacije parcijalnog uređaja koje odgovaraju relacijama "manje", odnosno, "manje ili jednako".

Propozicija A.2.1.

(a) *Ako je* $f(x) \in \Theta(g(x))$*, onda je i* $g(x) \in \Theta(f(x))$*, tj. relacija* Θ *je simetrična.*

(b) *Ako je* $f(x) \in O(g(x))$ *i* $g(x) \in O(f(x))$*, onda je* $f(x) \in \Theta(g(x))$*, tj. relacija* O *je antisimetrična na* $\mathbb{R}^D/_\Theta$*.*

(c) *Ako je* $f(x) \in \Theta(g(x))$ *i ako konstante* c_1*,* c_2 *možemo odabrati proizvoljno bliske, tj.* $\forall \varepsilon > 0$*,* $\exists c_1(\varepsilon), c_2(\varepsilon) > 0$ *i* $\exists x_0(\varepsilon) \in D$ *da je*

$$c_2(\varepsilon) - c_1(\varepsilon) < \varepsilon$$

i

$$c_1(\varepsilon)\,|g(x)| < |f(x)| < c_2(\varepsilon)\,|g(x)|, \quad \forall x > x_0(\varepsilon),$$

onda su $|f|$ *i* $|g|$ **asimptotski proporcionalne***, tj.* $\exists c > 0$ *takav da je*

$$|f(x)| \sim c\,|g(x)|.$$

Dokaz:

(a) Zbog $c_1, c_2 > 0$, iz

$$c_1\,|g(x)| < |f(x)| < c_2\,|g(x)|$$

slijedi

$$\frac{1}{c_2}\,|g(x)| < |f(x)| < \frac{1}{c_1}\,|g(x)|.$$

(b) Po definiciji, $\exists C_1, C_2 \in \mathbb{R}$ i $\exists x_1, x_2 \in D$ takvi da vrijedi

$$\begin{aligned} |f(x)| &< C_1\,|g(x)|, \quad \forall x > x_1, \\ |g(x)| &< C_2\,|f(x)|, \quad \forall x > x_2. \end{aligned}$$

Očito je $C_1, C_2 > 0$ i stavimo $x_0 = \max\{x_1, x_2\}$, pa je

$$\frac{1}{C_2}\,|g(x)| < |f(x)| < C_1\,|g(x)|, \quad \forall x > x_0.$$

(c) Kako je $c_1(\varepsilon) < c_2(\varepsilon)$ za $\forall \varepsilon > 0$, to postoji

$$c = \sup\{c_1(\varepsilon) \mid \varepsilon > 0\},$$

i očito je $c > 0$. Zbog $c_2(\varepsilon) - c_1(\varepsilon) < \varepsilon$, vrijedi i

$$c = \inf\{c_2(\varepsilon) \mid \varepsilon > 0\}.$$

Iz pretpostavke

$$c_1(\varepsilon) < \left|\frac{f(x)}{g(x)}\right| < c_2(\varepsilon), \quad \forall x > x_0(\varepsilon),$$

izlazi

$$c_1(\varepsilon) - c < \left|\frac{f(x)}{g(x)}\right| - c < c_2(\varepsilon) - c, \quad \forall x > x_0(\varepsilon).$$

Kako je $c_2(\varepsilon) - c < \varepsilon$ i $c - c_1(\varepsilon) < \varepsilon$, to je

$$-\varepsilon < \left|\frac{f(x)}{g(x)}\right| - c < \varepsilon, \quad \forall x > x_0(\varepsilon),$$

ili

$$\left|\left|\frac{f(x)}{g(x)}\right| - c\right| < \varepsilon, \quad \forall x > x_0(\varepsilon),$$

što znači

$$\lim_{x\to\infty}\left|\frac{f(x)}{g(x)}\right| = c,$$

ili

$$\lim_{x\to\infty}\frac{|f(x)|}{c\,|g(x)|} = 1.$$

■

Prirodno je očekivati da bilo koja mjera složenosti nekog algoritma bude nenegativna i da raste s veličinom zadaće, pa će funkcije f i g iz definicije biti nenegativne i gotovo uvijek rastuće, bar za velike argumente.

Za nenegativne funkcije, relacije iz propozicije A.2.1. vežu direktno funkcije, a ne samo njihove apsolutne vrijednosti. To opravdava sljedeću definiciju.

Definicija A.2.2.

Ako su f i g nenegativne funkcije i ako je $f(x) \in \Theta(g(x))$, onda kažemo da su funkcije f i g **kodominantne**.

Složenost algoritma bit će opisana funkcijom f, koja je, obično, kompliciranog oblika i često ju je nemoguće egzaktno odrediti za sve veličine zadaća.

Relacije asimptotskog ponašanja $\sim$ i Θ omogućavaju nam da za velike argumente izdvojimo tzv. dominantni član, kojeg ćemo opisati funkcijom g. Taj dominantni dio složenosti se najčešće lagano nalazi i reprezentativan je za ponašanje algoritma, što olakšava uspoređivanje algoritama. Pri tome je relacija Θ nepreciznija, jer daje red veličine složenosti, uz neki raspon konstantnog faktora.

Relacije o i O koristimo kad nas zanima tzv. "najgora" složenost (složenost u najgorem slučaju), tj. kad želimo pokazati da je složenost f nekog algoritma, manjeg (ili najviše istog) reda veličine kao i funkcija g, ne vodeći računa o konstantnim faktorima. Funkcija g služi tada kao gornja ograda složenosti.

Relacije ω i Ω koristimo rjeđe — kad dajemo donju ogradu složenosti nekog problema (što za algoritme nije interesantno).

Pošto ćemo raditi uglavnom s nenegativnim funkcijama, označimo

$$\mathbb{R}^+ = \{c \in \mathbb{R} \mid c > 0\} \qquad \text{i} \qquad \mathbb{R}_0^+ = \mathbb{R}^+ \cup \{0\}.$$

Sljedeća definicija uvodi klasifikaciju nenegativnih funkcija prema njihovom asimptotskom ponašanju (rastu) za velike argumente.

Definicija A.2.3.

Neka je $f : D \to \mathbb{R}_0^+$ nenegativna funkcija na odozgo neograničenom podskupu $D \subseteq \mathbb{R}$, i pretpostavimo da f neograničeno raste

$$\lim_{x\to\infty} f(x) = \infty.$$

(a) *f je* **blagog rasta**, *ako f raste sporije od bilo koje pozitivne potencije, tj. za $\forall \varepsilon > 0$ je*

$$f(x) \in o(x^\varepsilon).$$

(b) *f je* **polinomnog rasta**, *ako f raste sličnom brzinom kao i neka pozitivna potencija, tj. $\exists a_1, a_2 > 0$ takvi da je*

$$f(x) \in \Omega(x^{a_1}) \qquad i \qquad f(x) \in O(x^{a_2}).$$

(c) *f je* **blagog eksponencijalnog rasta**, *ako f raste brže od bilo koje potencije, ali sporije od bilo koje eksponencijalne funkcije c^x, za $c > 1$. To znači da je za $\forall a > 0$*

$$f(x) \in \Omega(x^a)$$

i za $\forall \varepsilon > 0$

$$f(x) \in o((1+\varepsilon)^x).$$

(d) *f je* **eksponencijalnog rasta**, *ako postoje $c_1, c_2 > 1$ takvi da je*

$$f(x) \in \Omega(c_1^x) \qquad i \qquad f(x) \in O(c_2^x)$$

(e) *f je* **nadeksponencijalnog rasta**, *ako f raste brže od bilo koje eksponencijalne funkcije, tj. za $\forall c > 1$ je*

$$f(x) \in \Omega(c^x).$$

Napomena A.2.3. *U definiciji* A.2.3., *svagdje gdje stoji O, može stajati o, i obratno. Isto tako, svagdje gdje stoji Ω, može stajati ω, i obratno.*

Ova klasifikacija funkcija prilagođena je primjeni u analizi složenosti algoritama. Zbog toga nas ne zanimaju posebno funkcije ponašanja $f(x) \in O(1)$ (ograničene) ili $f(x) \in o(1)$ (po volji male) za velike argumente, koje su od interesa u nekim drugim područjima matematike.

Napomena A.2.4. *Za funkcije f za koje je $f(x) \in o(1)$, tj. vrijedi $\lim_{x\to\infty} f(x) = 0$, može se definirati analogna klasifikacija* **pada** *za velike argumente. Ova klasifikacija se dobiva direktno iz definicije* A.2.3., *primjenom na funkciju $1/f$.*

Za naše primjene dovoljna je sljedeća definicija.

Definicija A.2.4.

Funkcija $f : D \to \mathbb{R}_0^+$ je **blagog ponašanja**, *ako za svaki $\varepsilon > 0$ vrijedi*

$$f(x) \in o(x^\varepsilon) \qquad i \qquad \frac{1}{f(x)} \in o(x^\varepsilon),$$

tj. ako je f i blagog rasta i blagog pada.

Primijetimo da ova definicija uključuje funkcije blagog rasta i blagog pada, ali, na primjer, i sve funkcije oblika $f(x) \in \Theta(1)$.

Za praksu su najvažniji algoritmi najviše eksponencijalne složenosti. Efikasnim (ili brzim) možemo smatrati one algoritme čija složenost blago raste, ili raste polinomno s malom potencijom a_2 (svakako $a_2 \leq 4$), u ovisnosti o veličini zadaće.

Primjer A.2.1. *Sljedeće funkcije su karakteristični netrivijalni primjeri klasifikacije iz definicije* A.2.3.*:*

(*a*) $f(x) = \log x$ *je blagog rasta,*

(*b*) $f(x) = x \log x$ *je polinomnog rasta,*

(*c*) $f(x) = x^{\log x}$ *je blagog eksponencijalnog rasta,*

(*d*) $f(x) = x\, 2^x$ *je eksponencijalnog rasta,*

(*e*) $f(x) = x^x$ *je nadeksponencijalnog rasta.*

Primjer A.2.2. *Očito su potencije $f(x) = x^a$, za $a > 0$, trivijalni primjeri funkcija polinomnog rasta. Međutim, primijetimo da polinomni rast funkcije f* **ne znači** *da je $f(x) \in \Theta(x^a)$ za neki a.*

Neka je $a > 0$. Za funkciju

$$f(x) = x^a \log x$$

vrijedi

$$f(x) \in \Omega(x^{a_1}), \quad \forall a_1 \leq a,$$

jer

$$\frac{f(x)}{x^{a_1}} = x^{a-a_1} \log x \to \infty, \quad za \quad x \to \infty,$$

zbog $a - a_1 \geq 0$, *ali i*

$$f(x) \in o(x^{a_2}), \quad \forall a_2 > a,$$

jer

$$\frac{f(x)}{x^{a_2}} = \frac{\log x}{x^{a_2-a}} \to 0, \quad za \quad x \to \infty,$$

zbog $a_2 - a > 0$.

Dakle, $f(x) = x^a \log x$ *je polinomnog rasta, ali ne postoji potencija s kojom bi* f *bila kodominantna.*

Primjer A.2.3. *Nije teško konstruirati primjer funkcije* f *čije asimptotsko ponašanje "varira" između* **dvije različite** *potencije, tj. vrijedi*

$$f(x) \in \Omega(x^{a_1}), \quad f(x) \in O(x^{a_2}),$$

za neke eksponente $a_1 < a_2$, *s tim da su ove asimptotske ocjene najbolje moguće (tj. dostižne).*

Na primjer, uzmimo $a_1 = 1$ *i* $a_2 = 2$. *Funkciju* $f : \mathbb{R}^+ \to \mathbb{R}^+$ *definiramo po komadima — intervalima duljine* 1, *tako da alterniramo "linearne" i "kvadratne" komade,*

$$f(x) = \begin{cases} x, & za\ x \in (2n-2, 2n-1] \\ x^2, & za\ x \in (2n-1, 2n] \end{cases}, \quad n \in \mathbb{N}.$$

Očito, za svaki $x \geq 1$ *vrijedi* $x \leq f(x) \leq x^2$, *s tim da se obje ocjene mogu dostići za proizvoljno velike* x *(donja za neparne, a gornja za parne prirodne brojeve). Dakle, vrijedi*

$$f(x) \in \Omega(x), \quad f(x) \in O(x^2).$$

Naravno, f *je prekidna funkcija, pa nije bilo teško postići ovakvo asimptotsko ponašanje. No, isto ponašanje se može postići i neprekidnom, čak beskonačno glatkom funkcijom.*

Napomenimo još da se takve funkcije f, čije asimptotsko ponašanje varira između funkcija f_1 i f_2 bitno različtih redova veličina, katkad javljaju u analizi složenosti aritmetičkih algoritama. Najpoznatiji primjer je klasični brzi Fourierov algoritam (FFT) za računanje vrijednosti polinoma stupnja n u svim n-tim korijenima iz jedinice, čija složenost varira između $n \log n$ (na primjer, za $n = 2^k$) i n^2 (za proste brojeve n).

B. Regularne funkcije složenosti

B.1. Ovisnost složenosti o veličini zadaće

Već smo rekli da složenost algoritma izražavamo kao funkciju veličine zadaće, odnosno, veličine ulaznih podataka algoritma.

Međutim, veličina podataka nužno ovisi o načinu kodiranja tih podataka. Zbog toga, i složenost algoritma ovisi o načinu kodiranja.

Najjednostavniji primjer kodiranja podataka je pozicioni zapis brojeva u nekoj bazi. Veličina podatka je, u tom slučaju, broj znamenki određenog broja u toj bazi. No, broj znamenki ovisi o bazi, pa i složenost svih aritmetičkih algoritama ovisi o bazi izabranoj za pozicioni prikaz.

Naš sljedeći zadatak je analiza ovisnosti složenosti algoritma o načinu kodiranja podataka. Posebno nas zanima uz koje uvjete složenost ne ovisi bitno o načinu kodiranja.

Naravno, potrebno je uvesti neke pretpostavke na kodiranja za koja uspoređujemo složenost. Pošto analiziramo aritmetičke algoritme, naši podaci će uvijek biti brojevi. Možemo pretpostaviti da radimo s nenegativnim brojevima, jer kôd predznaka uvijek možemo smatrati dijelom kôda broja. Zbog toga, kodiranje možemo interpretirati kao funkciju g na nekom podskupu $D \subseteq \mathbb{R}_0^+$, koja danom broju pridružuje duljinu njegovog kôda.

Razumno je pretpostaviti da je g nenegativna funkcija (jer je riječ o duljini kôda). Kako nas zanimaju velike zadaće, pretpostavljamo da je domena D odozgo neograničena, a prirodno je očekivati i

$$x \to \infty \implies g(x) \to \infty.$$

Duljina kôda g je, obično, i monotono rastuća (ili, barem, nepadajuća) funkcija, ali ni to nije nužno.

Dva kodiranja možemo smatrati **usporedivim**, ako su pripadne duljine kôdova g_1 i g_2 kodominantne. U praksi je, obično, ispunjen i jači uvjet, da su g_1 i g_2 asimptotski proporcionalne funkcije, tj. $\exists c > 0$ takav da je

$$g_1(x) \sim c\, g_2(x). \tag{B.1.1}$$

Ako je f složenost nekog algoritma, zanima nas da li su (i uz koje uvjete) funkcije $f \circ g_1$ i $f \circ g_2$ asimptotski proporcionalne, i kako su tada vezane konstante asimptotske proporcionalnosti.

Na konstantu c iz relacije (B.1.1) nećemo postavljati nikakve dodatne uvjete, jer za razna kodiranja dobivamo i razne konstante asimptotske proporcionalnosti. Zbog toga, želimo da $f \circ g_1$ i $f \circ g_2$ budu asimptotski proporcionalne za **bilo koji** $c > 0$.

Za preciznu formulaciju tvrdnje, potrebne su sljedeće definicije.

Definicija B.1.1.

Funkcija $f : D \to \mathbb{R}_0^+$ je **asimptotski homogena u točki** $c > 0$*, ako je f asimptotski pozitivna, tj. $\exists M \in D$ takav da za $\forall x \in D$, $x > M \implies f(x) > 0$, i ako postoji broj $K(c) \in \mathbb{R}^+$ takav da je*

$$f(cx) \sim K(c)\, f(x) \quad (x \to \infty). \tag{B.1.2}$$

Ako je f asimptotski homogena u **svakoj točki** $c > 0$*, onda kažemo da je f* **asimptotski homogena na** $\mathbb{R}^+$.

Tada je dobro definirana funkcija $K_f : \mathbb{R}^+ \to \mathbb{R}$ s

$$K_f(c) = \lim_{x \to \infty} \frac{f(cx)}{f(x)}, \quad c > 0. \tag{B.1.3}$$

Ako je K_f neprekidna funkcija na $\mathbb{R}^+$, onda je f **neprekidno asimptotski homogena**.

Pretpostavljamo da je skup D oblika (a, ∞) ili $[a, \infty)$, za neki $a \geq 0$.

Funkciju K_f često ćemo skraćeno označavati s K, ako je jasno na koju funkciju f se ona odnosi.

Primjer B.1.1. *Za svaku funkciju f je očito $K(1) = 1$. Ako je $f(x) = x^a$, $a \in \mathbb{R}$, onda je f neprekidno asimptotski homogena i vrijedi*

$$K(c) = c^a, \quad \forall c > 0, \tag{B.1.4}$$

jer je

$$\frac{f(cx)}{f(x)} = \frac{(cx)^a}{x^a} = c^a.$$

Kasnije ćemo pokazati da relacija (B.1.4) karakterizira sve neprekidno asimptotski homogene funkcije.

Svojstvo (neprekidne) asimptotske homogenosti omogućava da se faktor u argumentu zamijeni faktorom uz funkcijsku vrijednost.

Pošto radimo u okolini točke ∞, potreban je još i određeni uvjet neprekidnosti funkcije.

Definicija B.1.2.

Funkcija $f : D \to \mathbb{R}$ je **asimptotski uniformno relativno neprekidna**, *ako vrijedi*

$$\begin{aligned} &\forall \varepsilon > 0, \quad \exists \delta > 0, \quad \exists M \in D, \quad \forall \gamma \in \mathbb{R}, \quad \forall x \in D, \\ &x > M \quad i \quad |\gamma| < \delta \implies |f((1+\gamma)\,x) - f(x)| < \varepsilon\,|f(x)|. \end{aligned} \tag{B.1.5}$$

Ovaj uvjet znači da "mala" relativna pogreška u argumentu rezultira "malom" relativnom pogreškom funkcije, za sve dovoljno velike argumente, i to uniformno. Uočimo da je klasična neprekidnost zapravo neprekidnost u smislu apsolutne pogreške.

Lema B.1.1.

Neka je $f : D \to \mathbb{R}_0^+$ asimptotski homogena funkcija na $\mathbb{R}^+$. Ako je f i asimptotski uniformno relativno neprekidna funkcija, onda

(a) *za svaki $c \in \mathbb{R}^+$ vrijedi*

$$\begin{aligned} &\forall \varepsilon > 0, \quad \exists \delta > 0, \quad \exists M \in D, \quad \forall \gamma \in \mathbb{R}, \quad \forall x \in D, \\ &x > M \quad i \quad |\gamma| < \delta \implies |f((c+\gamma)x) - f(cx)| < \varepsilon\,|f(x)|, \end{aligned} \tag{B.1.6}$$

(b) *f je* **neprekidno** *asimptotski homogena funkcija, tj. K_f je neprekidna funkcija.*

Dokaz:

ad (a): Odaberimo proizvoljni $c \in \mathbb{R}^+$. Pošto je f asimptotski homogena na $\mathbb{R}^+$, iz relacije (B.1.2) za odabrani c slijedi

$$\begin{aligned} &\forall \varepsilon_0 > 0, \quad \exists M_0 \in D, \quad \forall x \in D, \\ &x > M_0 \implies \left| \frac{f(cx)}{K(c)\,f(x)} - 1 \right| < \varepsilon_0, \end{aligned}$$

ili

$$(1-\varepsilon_0)\,K(c) < \frac{f(cx)}{f(x)} < (1+\varepsilon_0)\,K(c).$$

Odaberimo $\varepsilon_0 < 1$. Zbog $K(c) > 0$, dobivamo da je $f(cx)/f(x) > 0$, odakle slijedi

$$1 < \frac{f(x)}{f(cx)} \cdot (1+\varepsilon_0)\,K(c). \tag{B.1.7}$$

Za proizvoljni $\varepsilon > 0$ definiramo

$$\varepsilon_1 = \frac{\varepsilon}{(1+\varepsilon_0)\,K(c)}. \tag{B.1.8}$$

Zbog asimptotske uniformne relativne neprekidnosti funkcije f, iz (B.1.5), za taj ε_1 postoje $\delta_1 > 0$ i $M_1 \in D$, takvi da za $\forall \gamma_1 \in \mathbb{R}$ i $\forall x \in D$ vrijedi

$$x > M_1 \quad \text{i} \quad |\gamma_1| < \delta_1 \implies |f((1+\gamma_1)\,x) - f(x)| < \varepsilon_1\,|f(x)|. \tag{B.1.9}$$

Kako je D odozgo neograničen, postoji $\widetilde{M}_1 \in D$ takav da je

$$\widetilde{M}_1 \geq \frac{M_1}{c}.$$

Tada $x > \widetilde{M}_1 \implies cx > M_1$, pa iz relacije (B.1.9) izlazi

$$|f((1+\gamma_1)\,cx) - f(cx)| < \varepsilon_1\,|f(cx)|. \tag{B.1.10}$$

Definirajmo $M = \max\{M_0, \widetilde{M}_1\}$, tako da za $x > M$ vrijede relacije (B.1.7) i (B.1.10). Množenjem tih dviju relacija i uvažavanjem $f(cx) > 0$ dobivamo

$$|f((1+\gamma_1)\,cx) - f(cx)| < \varepsilon_1\,(1+\varepsilon_0)\,K(c)\cdot f(x). \tag{B.1.11}$$

Ako definiramo $\delta = c\,\delta_1$ i $\gamma = c\,\gamma_1$, onda je $\delta > 0$ i $|\gamma_1| < \delta_1 \iff |\gamma| < \delta$. Iz (B.1.11), zbog definicije ε_1 iz (B.1.8), izlazi

$$|f((c+\gamma)\,x) - f(cx)| < \varepsilon\,f(x),$$

što je tražena relacija (B.1.6), jer je $f(x) > 0$.

ad (b) : Treba dokazati da je K neprekidna funkcija u svakoj točki $c > 0$. Za proizvoljni $c > 0$, promatramo vrijednost $|K(c+\gamma) - K(c)|$. Očito je

$$\begin{aligned} |K(c+\gamma) - K(c)| \leq &\left| K(c+\gamma) - \frac{f((c+\gamma)\,x)}{f(x)} \right| \\ &+ \left| \frac{f((c+\gamma)\,x) - f(cx)}{f(x)} \right| + \left| \frac{f(cx)}{f(x)} - K(c) \right|. \end{aligned} \tag{B.1.12}$$

Odaberimo $\varepsilon > 0$. Iz relacije (B.1.6) odredimo δ i M, pa za $x > M$ i bilo koji γ, $|\gamma| < \delta$, vrijedi

$$\left| \frac{f((c+\gamma)\,x) - f(cx)}{f(x)} \right| < \varepsilon. \tag{B.1.13}$$

Za preostala dva člana u (B.1.12), zbog asimptotske homogenosti funkcije f na $\mathbb{R}^+$, postoje $M_1, M_2 > 0$, takvi da vrijedi

$$x > M_1 \implies \left| \frac{f(cx)}{f(x)} - K(c) \right| < \varepsilon \tag{B.1.14}$$

i

$$x > M_2 \implies \left| \frac{f((c+\gamma)\,x)}{f(x)} - K(c+\gamma) \right| < \varepsilon. \tag{B.1.15}$$

Pri tome M_2 ovisi o γ. Za bilo koji γ, $|\gamma| < \delta$, stavimo

$$M_0 = \max\{M, M_1, M_2\},$$

pa za $x > M_0$ vrijede sve tri relacije (B.1.13), (B.1.14), (B.1.15). Time u (B.1.12) dobivamo

$$|K(c+\gamma) - K(c)| < 3\varepsilon,$$

za svaki γ, $|\gamma| < \delta$, što pokazuje da je funkcija K neprekidna u točki c. ■

Prvi rezultat o kompoziciji funkcija je sljedeći teorem.

Teorem B.1.1.

Neka su $g_1, g_2 : D \to \mathbb{R}_0^+$ asimptotski proporcionalne funkcije

$$g_1(x) \sim c\, g_2(x), \tag{B.1.16}$$

uz $c > 0$, takve da $x \to \infty \implies g_1(x) \to \infty$, $g_2(x) \to \infty$. (Uočimo da je dovoljno da samo jedna od ove dvije funkcije, recimo g_2, bude neograničena u beskonačnosti.)

Neka je $f : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ asimptotski homogena funkcija u točki c.

Ako f zadovoljava uvjet (B.1.6) *za dani c, onda su funkcije $f \circ g_1$ i $f \circ g_2$ asimptotski proporcionalne i konstanta proporcionalnosti je konstanta iz asimptotske homogenosti funkcije f*

$$(f \circ g_1)(x) \sim K_f(c)\,(f \circ g_2)(x). \tag{B.1.17}$$

Dokaz:

Relacija asimptotske homogenosti (B.1.2) znači da za $\forall \varepsilon > 0$, $\exists M_1 > 0$, tako da

$$x > M_1 \implies \left| \frac{f(cx)}{f(x)} - K(c) \right| < \varepsilon. \tag{B.1.18}$$

Da dokažemo (B.1.17), promatramo razliku

$$\left| \frac{f(g_1(x))}{f(g_2(x))} - K(c) \right|$$

za dovoljno velike x. Ovu razliku rastavljamo i ocjenjujemo u obliku

$$\left| \frac{f(g_1(x))}{f(g_2(x))} - K(c) \right| \le \left| \frac{f(g_1(x)) - f(c\,g_2(x))}{f(g_2(x))} \right| + \left| \frac{f(c\,g_2(x))}{f(g_2(x))} - K(c) \right|. \tag{B.1.19}$$

Fiksirajmo proizvoljni $\varepsilon > 0$. Zbog $g_2(x) \to \infty$, za $x \to \infty$, postoji $\widetilde{M}_1 \in D$ takav da $x > \widetilde{M}_1 \implies g_2(x) > M_1$. Iz (B.1.18) izlazi

$$\left| \frac{f(c\, g_2(x))}{f(g_2(x))} - K(c) \right| < \varepsilon. \tag{B.1.20}$$

Ostaje ocijeniti prvi član iz (B.1.19). Iz uvjeta (B.1.6) nađimo δ i M za odabrani ε. Analogno, postoji $\widetilde{M} \in D$, takav da $x > \widetilde{M} \implies g_2(x) > M$. Iz (B.1.6), dijeljenjem s $|f(g_2(x))| > 0$, dobivamo

$$x > \widetilde{M} \quad \text{i} \quad |\gamma| < \delta \implies \frac{|f((c+\gamma)\, g_2(x)) - f(c\, g_2(x))|}{|f(g_2(x))|} < \varepsilon. \tag{B.1.21}$$

Funkcije g_1 i g_2 su asimptotski proporcionalne, pa za nađeni δ, postoji $M_2 \in D$ takav da

$$x > M_2 \implies \left| \frac{g_1(x)}{g_2(x)} - c \right| < \delta,$$

ili

$$(c-\delta)\, g_2(x) < g_1(x) < (c+\delta)\, g_2(x).$$

No, onda za svaki $x > M_2$, postoji γ $(= \gamma(x))$, takav da je $|\gamma| < \delta$ i

$$g_1(x) = (c+\gamma)\, g_2(x). \tag{B.1.22}$$

Definirajmo $M_0 = \max\{\widetilde{M}, \widetilde{M}_1, M_2\}$. Za $x > M_0$, iz (B.1.21) i (B.1.22) izlazi

$$\frac{|f(g_1(x)) - f(c\, g_2(x))|}{|f(g_2(x))|} < \varepsilon,$$

što, zajedno s (B.1.20), u (B.1.19) daje

$$\left| \frac{f(g_1(x))}{f(g_2(x))} - K(c) \right| < 2\varepsilon,$$

za $x > M_0$. Time je relacija (B.1.17) dokazana. ■

Napomena B.1.1. *Uvjeti u teoremu* B.1.1. *osiguravaju prenošenje asimptotske proporcionalnosti funkcija g_1 i g_2 na funkcije $f \circ g_1$ i $f \circ g_2$, za* **fiksni** *c u* (B.1.16). *Zato su pretpostavke na funkciju f formulirane u točki c. Ako je f asimptotski homogena na nekoj okolini točke c u $\mathbb{R}^+$, onda uvjet* (B.1.6) *u točki c osigurava i neprekidnost funkcije K_f u točki c.*

Pošto želimo da relacija (B.1.17) vrijedi za **bilo koje** asimptotski proporcionalne funkcije g_1 i g_2, tj. za **bilo koji** $c > 0$, izlazi ovaj teorem.

Teorem B.1.2.

Neka je $f : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ asimptotski homogena funkcija na $\mathbb{R}^+$. Ako je f asimptotski uniformno relativno neprekidna funkcija, onda za **bilo koje dvije** *asimptotski proporcionalne funkcije $g_1,\ g_2 : D \to \mathbb{R}_0^+$, takve da $x \to \infty \implies g_2(x) \to \infty$, su i funkcije $f \circ g_1$ i $f \circ g_2$ asimptotski proporcionalne i vrijedi*

$$(f \circ g_1)(x) \sim K_f\left(\lim_{x\to\infty} \frac{g_1(x)}{g_2(x)}\right)(f \circ g_2)(x). \tag{B.1.23}$$

Dokaz:

Stavimo $c = \lim\limits_{x\to\infty} \dfrac{g_1(x)}{g_2(x)} > 0$. Lema B.1.1. daje uvjet (B.1.6) u toj točki c, a teorem B.1.1. daje relaciju (B.1.23). ■

U ovom slučaju su uvjeti (B.1.5) i (B.1.6) ekvivalentni, jer se (B.1.5) dobiva iz (B.1.6) za $c = 1$. Lema B.1.1. garantira da je funkcija K_f neprekidna.

Ovaj teorem, međutim, ne daje konstruktivan odgovor na pitanje za koje složenosti f se čuva asimptotska proporcionalnost mjera veličine zadaća.

Primijetimo još da se asimptotska homogenost se lako provjerava, za razliku od uvjeta (B.1.5). Bilo bi zgodno pronaći nešto jednostavniji uvjet, pa makar i uz blage dodatne pretpostavke.

Već smo rekli da je za složenost f prirodno očekivati monotonost, bar za velike argumente.

Definicija B.1.3.

Funkcija $f : D \to \mathbb{R}$ na odozgo neograničenom skupu $D \subseteq \mathbb{R}$ je **asimptotski monotona**, *ako je f monotona za velike argumente. Preciznije, ako $\exists M \in D$, takav da za svaki $x, y \in D$ vrijedi*

$$y > x > M \implies f(y) \geq f(x),$$

onda je f **asimptotski monotono rastuća**, *odnosno, ako vrijedi*

$$y > x > M \implies f(y) \leq f(x),$$

onda je f **asimptotski monotono padajuća**.

Uz pretpostavku asimptotske monotonosti vrijedi sljedeći rezultat o kompoziciji funkcija.

Teorem B.1.3.

Neka je $f : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ asimptotski monotona, neprekidno asimptotski homogena funkcija

$$f(cx) \sim K_f(c)\, f(x), \quad \forall c > 0.$$

Ako su $g_1,\ g_2 : D \to \mathbb{R}_0^+$ *asimptotski proporcionalne funkcije*

$$g_1(x) \sim c\, g_2(x),$$

uz $c > 0$, *i ako* $x \to \infty \implies g_2(x) \to \infty$, *onda su funkcije* $f \circ g_1$ *i* $f \circ g_2$ *asimptotski proporcionalne i vrijedi*

$$(f \circ g_1)(x) \sim K_f(c)\,(f \circ g_2)(x). \tag{B.1.24}$$

Dokaz:

Neprekidnost funkcije $K = K_f$ u točki c znači da za $\forall \varepsilon > 0$, $\exists \delta > 0$, tako da

$$|\gamma| < \delta \implies |K(c+\gamma) - K(c)| < \varepsilon. \tag{B.1.25}$$

Fiksirajmo $\varepsilon > 0$ i odaberimo $\gamma = \delta/2$. Iz asimptotske proporcionalnosti funkcija g_1 i g_2, za odabrani γ postoji $M_1 \in D$ takav da

$$x > M_1 \implies \left| \frac{g_1(x)}{g_2(x)} - c \right| < \gamma,$$

ili

$$(c-\gamma)\, g_2(x) \le g_1(x) \le (c+\gamma)\, g_2(x). \tag{B.1.26}$$

Pretpostavimo da je f asimptotski monotono rastuća. Tada $\exists M > 0$ takav da

$$M < y_1 < y_2 \implies f(y_1) \le f(y_2).$$

Zbog $x \to \infty \implies g_2(x) \to \infty$, postoji $M_2 \in D$ takav da

$$x > M_2 \implies (c-\gamma)\, g_2(x) > M.$$

Stavimo li $\widetilde{M} = \max\{M_1, M_2\}$, onda za $x > \widetilde{M}$, iz (B.1.26) izlazi

$$f((c-\gamma)\, g_2(x)) \le f(g_1(x)) \le f((c+\gamma)\, g_2(x)). \tag{B.1.27}$$

Kako g_2 neograničeno raste, asimptotska homogenost funkcije f u točkama $c-\gamma$, $c+\gamma$ daje da postoje $M_3, M_4 \in D$ takvi da

$$x > M_3 \implies \left| \frac{f((c-\gamma)\, g_2(x))}{f(g_2(x))} - K(c-\gamma) \right| < \varepsilon,$$

$$x > M_4 \implies \left| \frac{f((c+\gamma)\, g_2(x))}{f(g_2(x))} - K(c+\gamma) \right| < \varepsilon,$$

što daje

$$K(c-\gamma) - \varepsilon < \frac{f((c-\gamma)\, g_2(x))}{f(g_2(x))}$$

$$\frac{f((c+\gamma)\, g_2(x))}{f(g_2(x))} < K(c+\gamma) + \varepsilon.$$

Ako je $x > M_0 = \max\{\widetilde{M}, M_3, M_4\}$, onda iz (B.1.27) izlazi

$$K(c-\gamma) - \varepsilon < \frac{f(g_1(x))}{f(g_2(x))} < K(c+\gamma) + \varepsilon. \tag{B.1.28}$$

Pošto smo γ odabrali tako da vrijedi (B.1.25), dobivamo

$$\begin{aligned} -\varepsilon <& K(c-\gamma) - K(c), \\ & K(c+\gamma) - K(c) < \varepsilon, \end{aligned}$$

što u (B.1.28) daje

$$K(c) - 2\varepsilon < \frac{f(g_1(x))}{f(g_2(x))} < K(c) + 2\varepsilon,$$

tj.

$$x > M_0 \implies \left| \frac{f(g_1(x))}{f(g_2(x))} - K(c) \right| < 2\varepsilon.$$

Time je tvrdnja (B.1.24) dokazana. Sasvim analogno se dokaz provodi i za asimptotski monotono padajuće funkcije. ■

Ova tvrdnja se mnogo lakše primjenjuje, jer se oba uvjeta mogu relativno jednostavno provjeriti. Uočimo da smo pretpostavku o asimptotskoj homogenosti funkcije f pojačali zahtijevajući neprekidnost funkcije K_f.

Ako je f složenost nekog algoritma, onda je neprekidnost funkcije K_f vrlo prirodan zahtjev. To znači da "mala" relativna razlika u duljini kodova zadaće daje "malu" relativnu promjenu složenosti algoritma (barem za velike zadaće).

Drugi uvjet — asimptotske monotonosti, još je prirodniji, jer znači da složenost raste s duljinom zadaće, za dovoljno velike zadaće.

Ovo opravdava sljedeću definiciju.

Definicija B.1.4.

Složenost f nekog algoritma je **regularna**, *ako je f asimptotski monotono rastuća i neprekidno asimptotski homogena funkcija.*

Prema teoremu B.1.3., ako je složenost algoritma regularna, onda ona ne ovisi bitno o načinu kodiranja ulaznih podataka, tj. promjena kodiranja mijenja složenost najviše za konstantni faktor.

Teorem B.1.3. može se dobiti i kao direktna posljedica teorema B.1.2., jer vrijedi sljedeći rezultat.

Propozicija B.1.1.

Neka je $f : D \to \mathbb{R}_0^+$ asimptotski monotona funkcija. Ako je f neprekidno asimptotski homogena, onda je f i asimptotski uniformno relativno neprekidna funkcija.

Dokaz:

Kako je f neprekidno asimptotski homogena, to je funkcija K_f iz (B.1.3) definirana na nekom intervalu $(1-\delta_0, 1+\delta_0)$, s $\delta_0 > 0$, i neprekidna u točki $c = 1$. (Vrijedi i obrat, vidi napomenu B.2.1.). Dakle

$$\exists \delta_0 > 0, \quad \forall \varepsilon > 0, \quad \forall \gamma \in \mathbb{R}, \quad \exists M_{\varepsilon,\gamma} \in D, \quad \forall x \in D,$$
$$|\gamma| < \delta_0 \quad \text{i} \quad x > M_{\varepsilon,\gamma} \implies \left| \frac{f((1+\gamma)\,x)}{f(x)} - K_f(1+\gamma) \right| < \frac{\varepsilon}{2},$$

i

$$\forall \varepsilon > 0, \quad \exists \delta_1 > 0, \quad \forall \gamma \in \mathbb{R},$$
$$|\gamma| < \delta_1 \implies |K_f(1+\gamma) - K_f(1)| < \frac{\varepsilon}{2}.$$

Fiksirajmo $\varepsilon > 0$ i stavimo $\delta_2 = \min\{\delta_0, \delta_1\}$. Kombinacijom ove dvije tvrdnje, zbog $K_f(1) = 1$, izlazi

$$\forall \varepsilon > 0, \quad \exists \delta_2 > 0, \quad \forall \gamma \in \mathbb{R}, \quad \exists M_{\varepsilon,\gamma} \in D, \quad \forall x \in D,$$
$$|\gamma| < \delta_2 \quad \text{i} \quad x > M_{\varepsilon,\gamma} \implies \left| \frac{f((1+\gamma)\,x)}{f(x)} - 1 \right| < \varepsilon. \tag{B.1.29}$$

Općenito, $M_{\varepsilon,\gamma}$ ovisi i o γ.

Odaberimo $\delta = \delta_2/2$. Ako je f asimptotski monotono rastuća, onda postoji $M_0 \in D$, takav da za $\forall x \in D$ i $\forall \gamma \in \mathbb{R}$ vrijedi

$$x > M_0 \quad \text{i} \quad |\gamma| < \delta \implies f((1-\delta)\,x) \le f((1+\gamma)\,x) \le f((1+\delta)\,x). \tag{B.1.30}$$

Relacija (B.1.29) za $\gamma = -\delta$ daje

$$x > M_{\varepsilon,-\delta} \implies 1 - \varepsilon < \frac{f((1-\delta)\,x)}{f(x)}, \tag{B.1.31}$$

dok za $\gamma = \delta$ izlazi

$$x > M_{\varepsilon,\delta} \implies \frac{f((1+\delta)\,x)}{f(x)} < 1 + \varepsilon. \tag{B.1.32}$$

Ako stavimo $M_\varepsilon = \max\{M_0, M_{\varepsilon,-\delta}, M_{\varepsilon,\delta}\}$, onda iz (B.1.30), dijeljenjem s $f(x) > 0$, i (B.1.31), (B.1.32) dobivamo da za $\forall \gamma \in \mathbb{R}$, $|\gamma| < \delta$, vrijedi

$$x > M_\varepsilon \implies \left| \frac{f((1+\gamma)\,x)}{f(x)} - 1 \right| < \varepsilon,$$

pa M_ε ne ovisi o γ. Dakle, pokazali smo

$$\forall \varepsilon > 0, \quad \exists \delta > 0, \quad \exists M_\varepsilon \in D, \quad \forall \gamma \in \mathbb{R}, \quad \forall x \in D,$$
$$x > M_\varepsilon \quad \text{i} \quad |\gamma| < \delta \implies \left| \frac{f((1+\gamma)\,x)}{f(x)} - 1 \right| < \varepsilon,$$

što je asimptotski uniformna relativna neprekidnost funkcije f. Za asimptotski monotono padajuće funkcije dokaz je analogan. ■

Napomena B.1.2. *Obrat ove tvrdnje* **ne vrijedi** *ni za asimptotski monotone funkcije. Obrat vrijedi, na primjer, uz uvjet da je funkcija*

$$\frac{f((1+\gamma)\,x)}{f(x)}$$

asimptotski monotona za svaki $\gamma \in (1-\delta, 1+\delta)$*, za neki* $\delta > 0$*.*

Ostaje vidjeti koje funkcije zadovoljavaju uvjet neprekidne asimptotske homogenosti.

B.2. Neprekidno asimptotski homogene funkcije

Razrada općih uvjeta koji garantiraju neprekidnu asimptotsku homogenost prelazi okvire ovog rada. Za sve funkcije interesantne u analizi složenosti aritmetičkih algoritama, to svojstvo ćemo direktno dokazati.

Analizirajmo osnovna svojstva neprekidno asimptotski homogenih funkcija.

Propozicija B.2.1.

Neka je $f : D \to \mathbb{R}_0^+$ *neprekidno asimptotski homogena funkcija i*

$$K_f(c) = \lim_{x\to\infty} \frac{f(cx)}{f(x)}. \tag{B.2.1}$$

Tada je za svako $c_1,\ c_2 > 0$

$$K_f(c_1c_2) = K_f(c_1) \cdot K_f(c_2), \tag{B.2.2}$$

i postoji $a \in \mathbb{R}$*, takav da je*

$$K_f(c) = c^a, \quad \forall c > 0. \tag{B.2.3}$$

Dokaz:

Po definiciji (B.2.1) je

$$K_f(c_1c_2) = \lim_{x\to\infty} \frac{f(c_1c_2x)}{f(x)}.$$

Zbog $x \to \infty \implies c_2x \to \infty$, vrijedi i

$$\begin{aligned} K_f(c_1c_2) = \lim_{x\to\infty} \frac{f(c_1c_2x)}{f(x)} &= \lim_{x\to\infty} \left(\frac{f(c_1c_2x)}{f(c_2x)} \cdot \frac{f(c_2x)}{f(x)} \right) \\ &= \lim_{x\to\infty} \frac{f(c_1c_2x)}{f(c_2x)} \cdot \lim_{x\to\infty} \frac{f(c_2x)}{f(x)} \\ &= \lim_{c_2x\to\infty} \frac{f(c_1c_2x)}{f(c_2x)} \cdot \lim_{x\to\infty} \frac{f(c_2x)}{f(x)} = K_f(c_1) \cdot K_f(c_2), \end{aligned}$$

što dokazuje (B.2.2). Funkcija K_f je neprekidni multiplikativni homomorfizam na $\mathbb{R}^+$, pa postoji $a \in \mathbb{R}$ takav da vrijedi (B.2.3). ■

Napomena B.2.1. *Relacija* (B.2.2) *vrijedi i bez pretpostavke neprekidnosti funkcije K_f. To pokazuje da, ako je K_f definirana na bilo kojem intervalu pozitivne mjere, na primjer, $(1-\delta, 1+\delta)$, uz $\delta > 0$, onda je K_f definirana na cijelom $\mathbb{R}^+$.*

Za rezultat (B.2.3) *dovoljna je neprekidnost funkcije K_f u jednoj točki (na primjer, $c = 1$), što daje neprekidnost K_f na cijelom $\mathbb{R}^+$.*

Prema primjeru B.1.1., relacija (B.2.3) vrijedi za potenciju x^a. Ovaj rezultat sugerira da se neprekidno asimptotski homogene funkcije ne mogu "previše" razlikovati od neke potencije.

Ako je $K_f(c) = c^a$, definirajmo funkciju $g : D \to \mathbb{R}_0^+$ s

$$g(x) = \frac{f(x)}{x^a}, \quad x \in D. \tag{B.2.4}$$

Tada je, za svaki $c > 0$

$$\frac{g(cx)}{g(x)} = \frac{\dfrac{f(cx)}{(cx)^a}}{\dfrac{f(x)}{x^a}} = \frac{1}{c^a}\,\frac{f(cx)}{f(x)},$$

pa postoji $\lim\limits_{x\to\infty} \dfrac{g(cx)}{g(x)}$ i jednak je 1. To je motivacija za sljedeću definiciju.

Definicija B.2.1.

Funkcija $g : D \to \mathbb{R}_0^+$, na odozgo neograničenom skupu $D \subseteq \mathbb{R}$ je **asimptotski invarijantna na konstante**, *ako je g asimptotski pozitivna i ako za svaki $c > 0$ vrijedi*

$$g(cx) \sim g(x).$$

Očito je g neprekidno asimptotski homogena, jer je

$$K_g(c) = 1, \quad \forall c > 0.$$

Time dobivamo ovaj teorem **reprezentacije neprekidno asimptotski homogenih funkcija**.

Teorem B.2.1.

Funkcija $f : D \to \mathbb{R}_0^+$ je neprekidno asimptotski homogena, ako i samo ako postoje $a \in \mathbb{R}$ i funkcija $g : D \to \mathbb{R}_0^+$ asimptotski invarijantna na konstante, takvi da je

$$f(x) = x^a\, g(x), \quad \forall x \in D. \tag{B.2.5}$$

Ovaj prikaz je jedinstven.

Dokaz:

Egzistenciju prikaza (B.2.5) smo već dokazali, definicijom funkcije g u (B.2.4). Obrat je očit, kao i jedinstvenost, jer potencija a iz (B.2.5) jednoznačno određuje funkciju K_f. ■

Ovaj teorem opravdava sljedeću definiciju.

Definicija B.2.2.

Broj $a \in \mathbb{R}$ iz prikaza (B.2.5) *zovemo* **stupanj asimptotske homogenosti** *funkcije f i označavamo s*

$$\deg_H(f) = a.$$

Primijetimo da je

$$\deg_H(f) = \log_c K_f(c), \quad \forall c > 0. \tag{B.2.6}$$

Napomena B.2.2. *Funkcija g je asimptotski invarijantna na konstante ako i samo ako je*

$$\deg_H(g) = 0. \tag{B.2.7}$$

Sljedeća propozicija opisuje ponašanje klase neprekidno asimptotski homogenih funkcija obzirom na razne aritmetičke operacije.

Propozicija B.2.2.

Neka su f, f_1, $f_2 : D \to \mathbb{R}_0^+$ neprekidno asimptotski homogene funkcije i neka su α, $C \in \mathbb{R}$, $C > 0$, bilo koji brojevi.

Tada su sljedeće funkcije, također, neprekidno asimptotski homogene i za pripadne stupnjeve asimptotske homogenosti vrijede sljedeće relacije:

(a) $C \cdot f$,

$$\deg_H(C \cdot f) = \deg_H(f),$$

(b) f^α, *uz restrikciju domene na područje asimptotske pozitivnosti funkcije f, ako je $\alpha < 0$,*

$$\deg_H(f^\alpha) = \alpha \cdot \deg_H(f), \tag{B.2.8}$$

(c) $f_1 \cdot f_2$,

$$\deg_H(f_1 \cdot f_2) = \deg_H(f_1) + \deg_H(f_2). \tag{B.2.9}$$

Dokaz:

ad (a): Očito.

ad (b): Za $\forall c > 0$ vrijedi

$$\lim_{x\to\infty}\left(\frac{f(cx)}{f(x)}\right)^{\alpha} = \left(\lim_{x\to\infty}\frac{f(cx)}{f(x)}\right)^{\alpha},$$

jer limes na desnoj strani postoji i pozitivan je. Odavde je

$$K_{f^\alpha}(c) = \Big(K_f(c)\Big)^{\alpha}, \quad \forall c > 0.$$

ad (c): Za $\forall c > 0$ vrijedi

$$\lim_{x\to\infty}\frac{(f_1\cdot f_2)(cx)}{(f_1\cdot f_2)(x)} = \lim_{x\to\infty}\frac{f_1(cx)}{f_1(x)}\cdot\lim_{x\to\infty}\frac{f_2(cx)}{f_2(x)},$$

jer oba limesa na desnoj strani postoje. To znači

$$K_{f_1\cdot f_2}(c) = K_{f_1}(c)\cdot K_{f_2}(c), \quad \forall c > 0.$$

Relacije za stupnjeve izlaze direktno iz (B.2.6). ■

Slične tvrdnje vrijede i za klasu funkcija asimptotski invarijantnih na konstante i za klasu regularnih funkcija složenosti.

Korolar B.2.1.

Neka su f, f_1, $f_2 : D \to \mathbb{R}_0^+$ funkcije asimptotski invarijantne na konstante i neka su α, $C \in \mathbb{R}$, $C > 0$, bilo koji brojevi.

Tada su i funkcije $C\cdot f$, f^α, $f_1\cdot f_2$ asimptotski invarijantne na konstante.

Dokaz:

Direktno iz propozicije B.2.2. i napomene B.2.2. (iskoristimo (B.2.7) za stupanj asimptotske homogenosti). ■

Korolar B.2.2.

Neka su funkcije složenosti f, f_1, $f_2 : D \to \mathbb{R}_0^+$ regularne i neka su α, $C \in \mathbb{R}$, $\alpha \geq 0$, $C > 0$, bilo koji brojevi.

Tada su i funkcije $C\cdot f$, f^α, $f_1\cdot f_2$ regularne.

Dokaz:

Funkcije f, f_1, f_2 su asimptotski monotono rastuće, pa su takve i funkcije $C\cdot f$, f^α, $f_1\cdot f_2$. Iz propozicije B.2.2. odmah slijedi regularnost ovih funkcija. ■

Tvrdnja (a) iz propozicije B.2.2. za množenje konstantom može se proširiti i na asimptotsku proporcionalnost.

Propozicija B.2.3.

Neka su f_1, $f_2 : D \to \mathbb{R}_0^+$ asimptotski proporcionalne funkcije

$$f_2(x) \sim C\, f_1(x), \tag{B.2.10}$$

uz $C > 0$.

Ako je f_1 neprekidno asimptotski homogena funkcija, onda je i f_2 neprekidno asimptotski homogena, i

$$\deg_H(f_2) = \deg_H(f_1). \tag{B.2.11}$$

Dokaz:

Relacija (B.2.10) znači

$$\lim_{x\to\infty} \frac{f_2(x)}{f_1(x)} = C,$$

pa za bilo koji fiksni $c > 0$ vrijedi i

$$\lim_{x\to\infty} \frac{f_2(cx)}{f_1(cx)} = C.$$

Odavde je f_2 asimptotski pozitivna i vrijedi

$$\lim_{x\to\infty} \frac{f_2(cx)}{f_2(x)} = \lim_{x\to\infty} \frac{f_1(cx)}{f_1(x)} \cdot \frac{\lim\limits_{x\to\infty} \dfrac{f_2(cx)}{f_1(cx)}}{\lim\limits_{x\to\infty} \dfrac{f_2(x)}{f_1(x)}} = \lim_{x\to\infty} \frac{f_1(cx)}{f_1(x)},$$

jer svi limesi na desnoj strani postoje. Zbog toga je i f_2 neprekidno asimptotski homogena i vrijedi

$$K_{f_2}(c) = K_{f_1}(c), \quad \forall c > 0,$$

što dokazuje (B.2.11). ■

Ovaj rezultat, naravno, vrijedi i za funkcije asimptotski invarijantne na konstante. Za regularne funkcije, ovaj rezultat vrijedi samo ako je funkcija f_1 asimptotski **strogo** rastuća.

Primijetimo da propozicija B.2.2. ne uključuje zbrajanje. Za formulaciju odgovarajućeg rezultata potreban je sljedeći teorem.

Teorem B.2.2.

Neka je $f : D \to \mathbb{R}_0^+$ neprekidno asimptotski homogena funkcija. Za svaki $a \in \mathbb{R}$ vrijedi

$$\deg_H(f) > a \implies f(x) \in \omega(x^a). \tag{B.2.12}$$

Dokaz:

Pretpostavka $\deg_H(f) > a$ znači

$$K(c) > c^a, \quad \forall c > 0.$$

Odaberimo bilo koji $c > 1$. Tada postoji $\varepsilon > 0$ takav da je i

$$(1-\varepsilon)\, K(c) > c^a. \tag{B.2.13}$$

Zbog asimptotske homogenosti funkcije f u točki c, postoji $M \in D$ takav da za $\forall x \in D$ vrijedi

$$x > M \implies \left| \frac{f(cx)}{K(c)\, f(x)} - 1 \right| < \varepsilon. \tag{B.2.14}$$

Odaberimo proizvoljni $x_0 \in D$ takav da je $x_0 > M$. Iz (B.2.14) slijedi

$$\frac{f(cx_0)}{f(x_0)} > (1-\varepsilon)\, K(c). \tag{B.2.15}$$

Definiramo niz $x_n \in D$ s

$$x_n = c^n\, x_0, \quad n \in \mathbb{N}. \tag{B.2.16}$$

Zbog $c > 1$, očito $x_n \to \infty$ kad $n \to \infty$, i $x_n > M$, $\forall n \in \mathbb{N}$, pa relacija (B.2.14) vrijedi za svaki x_n

$$\frac{f(cx_n)}{f(x_n)} > (1-\varepsilon)\, K(c), \quad \forall n \in \mathbb{N}. \tag{B.2.17}$$

Produkt prvih $n-1$ relacija (B.2.17) i relacije (B.2.15), zbog definicije niza x_n iz (B.2.16), daje

$$\frac{f(x_n)}{f(x_0)} > [(1-\varepsilon)\, K(c)]^n, \quad \forall n \in \mathbb{N}.$$

Odavde izlazi

$$f(x_n) > f(x_0)\, [(1-\varepsilon)\, K(c)]^n = f(x_0)\, \frac{[(1-\varepsilon)\, K(c)]^n}{x_n^a} \cdot x_n^a$$

$$= f(x_0)\, \frac{[(1-\varepsilon)\, K(c)]^n}{c^{an}\, x_0^a} \cdot x_n^a = \frac{f(x_0)}{x_0^a} \left[\frac{(1-\varepsilon)\, K(c)}{c^a} \right]^n \cdot x_n^a.$$

Definiramo li

$$\varepsilon_n = \frac{f(x_0)}{x_0^a} \left[\frac{(1-\varepsilon)\, K(c)}{c^a} \right]^n,$$

iz (B.2.13) izlazi $\varepsilon_n \to \infty$.

Dakle, postoje nizovi $x_n \in D$, $\varepsilon_n > 0$, $n \in \mathbb{N}$, takvi da $x_n \to \infty$, $\varepsilon_n \to \infty$ i

$$f(x_n) > \varepsilon_n\, x_n^a.$$

Po definiciji A.2.1.(f), to znači da je

$$f(x) \in \omega(x^a),$$

što dokazuje implikaciju (B.2.12). ■

Napomena B.2.3. *Ovaj teorem vrijedi i uz mnogo slabije uvjete. Iz dokaza se vidi da je dovoljna asimptotska homogenost u jednoj točki $c > 1$, pa da vrijedi*

$$\lim_{x\to\infty} \frac{f(cx)}{f(x)} > c^a \implies f(x) \in \omega(x^a).$$

Korolar B.2.3.

Neka je $f : D \to \mathbb{R}_0^+$ neprekidno asimptotski homogena funkcija. Za svaki $a \in \mathbb{R}$ vrijedi

$$f(x) \in O(x^a) \implies \deg_H(f) \leq a. \tag{B.2.18}$$

Dokaz:

Tvdrnja slijedi direktno iz teorema B.2.2., jer je (B.2.18) obrat po kontrapoziciji relacije (B.2.12). ■

Propozicija B.2.4.

Neka su f_1, $f_2 : D \to \mathbb{R}_0^+$ neprekidno asimptotski homogene funkcije. Ako postoji bar jedan od limesa

$$\lim_{x\to\infty} \frac{f_1(x)}{f_2(x)}, \quad \lim_{x\to\infty} \frac{f_2(x)}{f_1(x)},$$

onda je i $f_1 + f_2$ neprekidno asimptotski homogena i vrijedi

$$\deg_H(f_1 + f_2) = \max\{\deg_H(f_1), \deg_H(f_2)\}. \tag{B.2.19}$$

Dokaz:

Pretpostavimo da postoji

$$\lim_{x\to\infty} \frac{f_2(x)}{f_1(x)} = A. \tag{B.2.20}$$

Za bilo koji $c > 0$, vrijedi

$$\frac{(f_1+f_2)(cx)}{(f_1+f_2)(x)} = \frac{f_1(cx)+f_2(cx)}{f_1(x)+f_2(x)} = \frac{f_1(cx)}{f_1(x)} \cdot \frac{1+\dfrac{f_2(cx)}{f_1(cx)}}{1+\dfrac{f_2(x)}{f_1(x)}}.$$

Prijelazom na limes izlazi

$$\lim_{x\to\infty} \frac{(f_1+f_2)(cx)}{(f_1+f_2)(x)} = K_{f_1}(c) \cdot \frac{1+A}{1+A} = K_{f_1}(c),$$

što daje

$$K_{f_1+f_2}(c) = K_{f_1}(c),$$
$$\deg_H(f_1 + f_2) = \deg_H(f_1).$$

Da dokažemo (B.2.19), treba pokazati

$$\deg_H(f_1) \geq \deg_H(f_2). \tag{B.2.21}$$

Ako je $A > 0$ u (B.2.20), onda su f_1 i f_2 asimptotski proporcionalne, pa propozicija B.2.3. daje

$$\deg_H(f_1) = \deg_H(f_2),$$

što dokazuje (B.2.19). (Tada postoje oba limesa iz izreke teorema i dovoljno je da jedna od funkcija bude neprekidno asimptotski homogena.)

Ako je $A = 0$, to znači da je

$$f_2(x) \in o(f_1(x)),$$

ili

$$\frac{f_2(x)}{f_1(x)} \in o(1).$$

Kako je tada i funkcija f_2/f_1 neprekidno asimptotski homogena (propozicija B.2.2.), primjenom korolara B.2.3. s $a = 0$ izlazi

$$\deg_H\left(\frac{f_2}{f_1}\right) \leq 0. \tag{B.2.22}$$

Iz relacija (B.2.8) i (B.2.9) slijedi

$$\deg_H\left(\frac{f_2}{f_1}\right) = \deg_H(f_2) - \deg_H(f_1),$$

što, zajedno s (B.2.22), dokazuje (B.2.21). ■

Propozicija B.2.4., također, vrijedi za funkcije asimptotski invarijantne na konstante i za regularne složenosti.

Ovaj rezultat pokazuje da na asimptotsko ponašanje takvih funkcija utječe samo dominantni član.

B.3. Regularne složenosti

Teorem reprezentacije neprekidno asimptotski homogenih funkcija pokazuje da se njihovo proučavanje svodi na proučavanje funkcija asimptotski invarijantnih na konstante.

Pošto nas zanimaju regularne funkcije $f : D \to \mathbb{R}_0^+$, u njihovoj reprezentaciji

$$f(x) = x^{\deg_H(f)}\, g(x),$$

potencija $\deg_H(f)$ mora biti nenegativna. To izlazi direktno iz definicije funkcije K_f, jer je f asimptotski monotono rastuća funkcija. Funkcija g ne mora biti asimptotski monotono rastuća, osim za $\deg_H(f) = 0$.

Tipični reprezentanti klase funkcija asimptotski invarijantnih na konstante, koji se najčešće koriste u analizi složenosti algoritama, dani su sljedećom definicijom.

Definicija B.3.1.

Neka je $b \in \mathbb{R}$, $b > 1$, bilo koji realan broj. Niz funkcija f_n, $n \in \mathbb{N}$, definiran rekurzivno relacijama

$$\begin{aligned} f_1(x) &= \log_b x, \\ f_n(x) &= \log_b f_{n-1}(x), \quad n \geq 2, \end{aligned} \tag{B.3.1}$$

zovemo **ponovljeni** *ili* **iterirani logaritmi** *(po bazi b). Funkciju*

$$f_n(x) = \underbrace{\log_b \cdots \log_b}_{n\ puta} x$$

*zovemo n-**ti ponovljeni (iterirani) logaritam** (po bazi b).*

Smatramo da je f_n definirana na skupu D_n na kojem postiže **nenegativne** *vrijednosti, tj. $f_n : D_n \to \mathbb{R}_0^+$.*

Bazu b logaritama iz relacije (B.3.1) najčešće nećemo označavati. Ovako definirani niz funkcija f_n, $n \in \mathbb{N}$, katkad je korisno dopuniti funkcijom $f_0 : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ definiranom s

$$f_0(x) = x.$$

(Logaritam po bazi b se primijenjuje "nula" puta.)

Propozicija B.3.1.

Ponovljeni logaritmi $f_n : D_n \to \mathbb{R}_0^+$, $n \in \mathbb{N}$, su regularne funkcije, asimptotski invarijantne na konstante, tj. vrijedi

$$\deg_H(f_n) = 0, \quad \forall n \in \mathbb{N}. \tag{B.3.2}$$

Dokaz:

Provodimo ga indukcijom po n. Za bilo koji $c > 0$ je

$$\frac{f_1(cx)}{f_1(x)} = \frac{\log_b(cx)}{\log_b x} = 1 + \frac{\log_b c}{\log_b x}.$$

Kako $x \to \infty \implies \log_b x \to \infty$, to je

$$\lim_{x\to\infty} \frac{f_1(cx)}{f_1(x)} = 1, \quad \forall c > 0,$$

što dokazuje (B.3.2) za $n = 1$. Monotonost funkcije f_1 je očita, pa je f_1 regularna.

Primijetimo da i za derivacije po x vrijedi

$$\lim_{x\to\infty} \frac{f_1'(cx)}{f_1'(x)} = 1, \quad \forall c > 0,$$

jer je $f_1'(cx) = f_1'(x)$, $\forall x \geq 1$ i $\forall c > 0$.

Pretpostavimo da je tvrdnja dokazana za funkciju f_{n-1}, za neki $n \geq 2$, i da vrijedi

$$\lim_{x\to\infty} \frac{f_{n-1}(cx)}{f_{n-1}(x)} = \lim_{x\to\infty} \frac{f_{n-1}'(cx)}{f_{n-1}'(x)} = 1, \tag{B.3.3}$$

za svaki $c > 0$.

Iz $f_{n-1}(x) \to \infty$ za $x \to \infty$, slijedi i $f_n(x) \to \infty$, direktno iz rekurzivne definicije. Zbog toga, sljedeći limes računamo L'Hospitalovim pravilom.

$$\begin{aligned}\lim_{x\to\infty} \frac{f_n(cx)}{f_n(x)} &= (\text{L'Hospital}) = \lim_{x\to\infty} \frac{f_n'(cx)}{f_n'(x)} = (B.3.1)\\ &= \lim_{x\to\infty} \frac{\ln b \cdot \dfrac{f_{n-1}'(cx)}{f_{n-1}(cx)}}{\ln b \cdot \dfrac{f_{n-1}'(x)}{f_{n-1}(x)}} = \frac{\lim\limits_{x\to\infty} \dfrac{f_{n-1}'(cx)}{f_{n-1}'(x)}}{\lim\limits_{x\to\infty} \dfrac{f_{n-1}(cx)}{f_{n-1}(x)}} = (B.3.3) = 1.\end{aligned}$$

Monotonost funkcije f_n je očita, pa je tvrdnja dokazana. ■

Teorem B.3.1.

Neka je $f : D \to \mathbb{R}_0^+$ asimptotski strogo rastuća funkcija. Neka je F funkcija oblika

$$F(x) = x^{a_0} (\log x)^{a_1} (\log\log x)^{a_2} \cdots (\underbrace{\log \cdots \log x}_{n\ puta})^{a_n}, \tag{B.3.4}$$

za neki $n \in \mathbb{N}_0$ i neke brojeve $a_0, a_1, \ldots, a_n \in \mathbb{R}$, koji nisu svi jednaki 0, uz $a_k > 0$, gdje je $k = \min\{i \mid a_i \neq 0\}$.

Ako je f asimptotski proporcionalna funkciji F, onda je f regularna funkcija i

$$\deg_H(f) = a_0. \tag{B.3.5}$$

Dokaz:

Lako se pokazuje da je F asimptotski strogo monotono rastuća funkcija. Iz propozicija B.2.2. i B.3.1. slijedi neprekidna asimptotska homogenost funkcije F i

$$\deg_H(F) = a_0.$$

Na kraju, iz propozicije B.2.3. slijede regularnost funkcije f i relacija (B.3.5). ■

Ovo je podloga za vrlo važan rezultat u analizi složenosti aritmetičkih algoritama, jer **gotovo svi aritmetički algoritmi imaju složenost asimptotski proporcionalnu nekoj funkciji oblika (B.3.4)**. Posljedica toga je neovisnost složenosti o načinu prikaza ulaznih brojeva, kad jednom pokažemo da su duljine brojeva u različitim bazama asimptotski proporcionalne.

U teoremu B.2.2. i korolaru B.2.3. uspostavili smo vezu između stupnja asimptotske homogenosti funkcije i reda veličine njenog rasta. Pokazali smo da

$$f(x) \in O(x^{a_1}) \implies \deg_H(f) \leq a_1. \tag{B.3.6}$$

Pošto je i $1/f$ neprekidno asimptotski homogena, gornjom ogradom rasta funkcije $1/f$ izlazi donja ograda za $\deg_H(f)$

$$\frac{1}{f(x)} \in O(x^{-a_2}) \implies \deg_H(f) \geq a_2. \tag{B.3.7}$$

Propozicija B.3.2.

Ako je $f : D \to \mathbb{R}_0^+$ neprekidno asimptotski homogena funkcija i ako je f blagog ponašanja, onda je

$$\deg_H(f) = 0,$$

tj. f je asimptotski invarijantna na konstante.

Dokaz:

Po definiciji A.2.4., jer je f blagog ponašanja, za $\forall \varepsilon > 0$ vrijedi

$$f(x) \in o(x^{\varepsilon}), \quad \frac{1}{f(x)} \in o(x^{-\varepsilon}),$$

pa je i

$$f(x) \in O(x^{\varepsilon}), \quad \frac{1}{f(x)} \in O(x^{-\varepsilon}).$$

Iz relacija (B.3.6), (B.3.7) izlazi

$$-\varepsilon \leq \deg_H(f) \leq \varepsilon, \quad \forall \varepsilon > 0,$$

što dokazuje tvrdnju. ■

Mnogo zanimljivije je pitanje obrata ove tvrdnje i obrata tvrdnji teorema B.2.2. i korolara B.2.3.

Napomena B.3.1. *Potpuni obrat tvrdnje teorema* B.2.2. **ne vrijedi**. *Za neprekidno asimptotski homogenu funkciju f, prema* (B.2.12)*, taj bi obrat glasio*

$$f(x) \in \omega(x^a) \implies \deg_H(f) > a.$$

Za funkciju $f(x) = x^a \log x$, očito je $f(x) \in \omega(x^a)$, ali je $\deg_H(f) = a$.

Zbog toga, u obratu treba dozvoliti i jednakost za stupanj asimptotske homogenosti. Takav obrat dokazujemo uz pojačane pretpostavke na funkciju f.

Teorem B.3.2.

Neka je $f : D \to \mathbb{R}_0^+$ **regularna** *funkcija. Za svaki $a \in \mathbb{R}$ vrijedi*

$$f(x) \in \Omega(x^a) \implies \deg_H(f) \geq a. \tag{B.3.8}$$

Dokaz:

Uočimo da je dovoljno pokazati da je

$$K(c) \geq c^a,$$

za neki $c > 0$.

Pretpostavka $f(x) \in \Omega(x^a)$ znači da $\exists \varepsilon_0 > 0$ i $\exists x_n \in D$, $n \in \mathbb{N}$, takvi da $x_n \to \infty$ i da je

$$f(x_n) \geq \varepsilon_0 \, x_n^a. \tag{B.3.9}$$

Bez smanjenja općenitosti, možemo pretpostaviti da je x_n rastući niz, jer $x_n \to \infty$.

Odaberimo proizvoljne $c > 1$, $0 < \varepsilon < 1$. Iz pretpostavke asimptotske homogenosti funkcije f u točki c, slijedi da postoji $M \in D$ takav da

$$x > M \implies \left| \frac{f(cx)}{K(c)\, f(x)} - 1 \right| < \varepsilon,$$

ili

$$(1 - \varepsilon)\, K(c) < \frac{f(cx)}{f(x)} < (1 + \varepsilon)\, K(c).$$

Kako je $c > 1$, to je i $c^{k-1}x > M$, $\forall k \in \mathbb{N}$, pa je

$$(1 - \varepsilon)\, K(c) < \frac{f(c^k x)}{f(c^{k-1}x)} < (1 + \varepsilon)\, K(c). \tag{B.3.10}$$

Zbog $\varepsilon < 1$ i $K(c) > 0$ je $(1 - \varepsilon)\, K(c) > 0$. Za bilo koji $m \in \mathbb{N}$, množenjem relacija (B.3.10) za $k = 1, \ldots, m$, izlazi

$$[(1 - \varepsilon)\, K(c)]^m < \frac{f(c^m x)}{f(x)} < [(1 + \varepsilon)\, K(c)]^m, \tag{B.3.11}$$

za svaki $x > M$ i $\forall m \in \mathbb{N}$.

Zbog $x_n \to \infty$, postoji $n_0 \in \mathbb{N}$ takav da je $x_{n_0} > M$. Niz x_n je rastući, pa

$$n \geq n_0 \implies x_n > M.$$

Odaberimo fiksni $n > n_0$ i označimo

$$x_{n+k} = c^{\alpha_k} x_n, \quad \forall k \in \mathbb{N},$$

što daje niz realnih brojeva $\alpha_k > 0$ i $\alpha_k \to \infty$ (jer $x_{n+k} \to \infty$, za $k \to \infty$).

Primjenom relacije (B.3.9) na x_{n+k} dobivamo

$$f(c^{\alpha_k} x_n) \geq \varepsilon_0 \, c^{a\alpha_k} x_n^a. \tag{B.3.12}$$

Označimo

$$f(x_n) = \varepsilon_0 \, c^{\alpha} \, x_n, \tag{B.3.13}$$

pa iz (B.3.9) izlazi $\alpha \geq 0$, jer je $c > 1$. Iz relacija (B.3.12) i (B.3.13) izlazi da za svaki $k \in \mathbb{N}$ vrijedi

$$\frac{f(c^{\alpha_k} x_n)}{f(x_n)} \geq c^{a\alpha_k - \alpha}. \tag{B.3.14}$$

Funkcija f je asimptotski monotono rastuća, pa ako M odaberemo dovoljno velik, onda vrijedi

$$f(c^{\lceil \alpha_k \rceil} x_n) \geq f(c^{\alpha_k} x_n), \tag{B.3.15}$$

za svaki $k \in \mathbb{N}$. Kako je $\lceil \alpha_k \rceil$ prirodan broj, primjenom relacije (B.3.11) s $m = \lceil \alpha_k \rceil$ i $x = x_n$, dobivamo

$$\frac{f(c^{\lceil \alpha_k \rceil} x_n)}{f(x_n)} < [(1+\varepsilon)\, K(c)]^{\lceil \alpha_k \rceil},$$

što, zajedno s (B.3.14) i (B.3.15), daje

$$[(1+\varepsilon)\, K(c)]^{\lceil \alpha_k \rceil} > c^{a\alpha_k - \alpha},$$

ili

$$K(c) > \frac{1}{1+\varepsilon} \cdot c^{(a\alpha_k - \alpha)/\lceil \alpha_k \rceil}. \tag{B.3.16}$$

Brojevi a i α su konstante, i $\alpha_k \to \infty$, pa na limesu $k \to \infty$ izlazi

$$\frac{a\alpha_k - \alpha}{\lceil \alpha_k \rceil} \to a,$$

što u (B.3.16) daje

$$K(c) \geq \frac{c^a}{1+\varepsilon}.$$

Pošto je $\varepsilon > 0$ proizvoljan, to pokazuje

$$K(c) \geq c^a,$$

za odabrani c, što smo i htjeli dobiti. ■

Primijetimo da u tvrdnji (B.3.8) stoji Ω, a ne ω, što je nešto blaži uvjet, suglasan dozvoljenoj jednakosti $\deg_H(f) \geq a$ na desnoj strani.

Napomena B.3.2. *U ovom teoremu, kao i u teoremu* B.2.2.*, dovoljna je asimptotska homogenost u jednoj točki $c > 1$, pa da vrijedi*

$$f(x) \in \Omega(x^a) \implies \lim_{x\to\infty} \frac{f(cx)}{f(x)} \geq c^a,$$

uz uvjet da je f asimptotski monotono rastuća.

Dodatna pretpostavka na funkciju f (asimptotska uzlaznost, u ovom slučaju), potrebna je zbog toga što uvjet $f(x) \in \Omega(x^a)$ predstavlja mnogo slabiji zahtjev na funkciju od, na primjer, uvjeta $1/f(x) \in O(x^{-a})$ iz relacije (B.3.7), iako je zaključak isti.

Međutim, i ta dodatna pretpostavka može se oslabiti do asimptotski uniformne relativne neprekidnosti.

Teorem B.3.3.

Neka je $f : D \to \mathbb{R}_0^+$ neprekidno asimptotski homogena i asimptotski uniformno relativno neprekidna funkcija. Za svaki $a \in \mathbb{R}$ vrijedi

$$f(x) \in \Omega(x^a) \implies \deg_H(f) \geq a.$$

Dokaz:

Asimptotski uniformna relativna neprekidnost funkcije f znači

$$\begin{aligned} &\forall\varepsilon_0 > 0, \quad \exists\delta_0 > 0, \quad \exists M_0 \in D, \quad \forall\gamma_0 \in \mathbb{R}, \quad \forall x \in D, \\ &\quad x > M_0 \quad \text{i} \quad |\gamma_0| < \delta_0 \implies \left|\frac{f((1+\gamma_0)\,x)}{f(x)} - 1\right| < \varepsilon_0. \end{aligned} \tag{B.3.17}$$

Fiksiramo proizvoljni $\varepsilon_0 > 0$ i nađemo pripadne δ_0 i M_0. Zatim, **definiramo** c

$$c = 1 + \delta_0 > 1. \tag{B.3.18}$$

Nastavak dokaza, koji koristi asimptotsku homogenost funkcije f, je isti kao u dokazu teorema B.3.2., s tim da, za izabrani $0 < \varepsilon < 1$, odaberemo

$$x_n > \max\{M_0, M\}.$$

Dobivamo relaciju (B.3.14)

$$\frac{f(c^{\alpha_k}x_n)}{f(x_n)} \geq c^{a\alpha_k-\alpha}, \quad \forall k \in \mathbb{N}.$$

Vrijedi

$$c^{\lceil\alpha_k\rceil}x_n = c^{\lceil\alpha_k\rceil-\alpha_k} \cdot c^{\alpha_k}x_n.$$

Kako je $0 \leq \lceil\alpha_k\rceil - \alpha_k < 1$, $\forall k \in \mathbb{N}$, to je, prema (B.3.18)

$$1 \leq c^{\lceil\alpha_k\rceil-\alpha_k} < c = 1 + \delta_0.$$

Zbog $c^{\alpha_k}x_n > M_0$, $\forall k \in \mathbb{N}$, primjenom relacije (B.3.17) s $\gamma_0 = c^{\lceil\alpha_k\rceil-\alpha_k}$ i $x = c^{\alpha_k}x_n$, dobivamo

$$f(c^{\lceil\alpha_k\rceil}x_n) > (1-\varepsilon_0)\, f(c^{\alpha_k}x_n),$$

za svaki $k \in \mathbb{N}$. Relacija (B.3.11) s $m = \lceil\alpha_k\rceil$ daje

$$K(c) > \frac{(1-\varepsilon_0)^{1/\lceil\alpha_k\rceil}}{1+\varepsilon} \cdot c^{(a\alpha_k-\alpha)/\lceil\alpha_k\rceil}. \tag{B.3.19}$$

Iz $\alpha_k \to \infty$ za $k \to \infty$, izlazi i

$$(1-\varepsilon_0)^{1/\lceil\alpha_k\rceil} \to 1,$$

jer je ε_0 fiksna konstanta. Na kraju, iz (B.3.19) slijedi

$$K(c) \geq \frac{c^a}{1+\varepsilon},$$

za proizvoljni $\varepsilon > 0$, što dokazuje tvrdnju. ■

Zbog propozicije B.1.1., teorem B.3.2. se može dobiti i kao direktna posljedica ovog teorema. Odmah izlazi i sljedeći rezultat.

Korolar B.3.1.

Neka je $f : D \to \mathbb{R}_0^+$ neprekidno asimptotski homogena i asimptotski uniformno relativno neprekidna funkcija. Za svaki $a \in \mathbb{R}$ vrijedi

$$\deg_H(f) < a \implies f(x) \in o(x^a).$$

Dokaz:

Direktno iz tvrdnje teorema B.3.3., obratom po kontrapoziciji. ■

Kombinacijom ovih rezultata dobivamo striktnu vezu između stupnja asimptotske homogenosti i rasta regularne (asimptotski uniformno relativno neprekidne) funkcije.

Teorem B.3.4.

Neka je $f : D \to \mathbb{R}_0^+$ neprekidno asimptotski homogena i asimptotski uniformno relativno neprekidna funkcija. Tada je

$$\deg_H(f) = a,$$

ako i samo ako za svaki $\varepsilon > 0$ vrijede relacije

$$f(x) \in o(x^{a+\varepsilon}), \quad \frac{1}{f(x)} \in o(x^{-a+\varepsilon}). \tag{B.3.20}$$

Dokaz:

Iz definicije asimptotski uniformno relativno neprekidne funkcije je očito da je i $1/f$ takva funkcija.

Ako je $\deg_H(f) = a$, onda propozicija B.2.2. daje $\deg_H(1/f) = -a$. Tada je

$$\deg_H(f) < a + \varepsilon, \quad \deg_H(1/f) < -a + \varepsilon, \quad \forall \varepsilon > 0,$$

pa korolar B.3.1. daje relacije (B.3.20).

Obratno, ako je $f(x) \in o(x^{a+\varepsilon})$, za svaki $\varepsilon > 0$, onda je očito i $f(x) \in O(x^{a+\varepsilon})$, pa iz (B.2.18) slijedi

$$\deg_H(f) \leq a + \varepsilon, \quad \forall \varepsilon > 0. \tag{B.3.21}$$

Analogno, iz $1/f(x) \in o(x^{-a+\varepsilon})$ slijedi

$$\deg_H(1/f) \leq -a + \varepsilon,$$

što, zbog $\deg_H(1/f) = -\deg_H(f)$, daje

$$\deg_H(f) \geq a - \varepsilon, \quad \forall \varepsilon > 0. \tag{B.3.22}$$

Relacije (B.3.21) i (B.3.22) daju $\deg_H(f) = a$. ■

Druga od relacija (B.3.20) može se zamijeniti i slabijim zahtjevom

$$f(x) \in \omega(x^{a-\varepsilon}), \quad \forall \varepsilon > 0.$$

Napomena B.3.3. *Stavimo li $a = 0$ u teoremu* B.3.4.*, dobivamo djelomični obrat propozicije* B.3.2.*, jer relacije* (B.3.20) *znače tada da je f blaga funkcija.*

Dokaz teorema B.3.4. mogli smo provesti i korištenjem teorema reprezentacije, pa bi tada dovoljno bilo dokazati tvrdnju za $a = 0$.

Za regularne funkcije, također, vrijedi teorem B.3.4., s tim da je $a \geq 0$, zbog asimptotske uzlaznosti. Ta tvrdnja ima sljedeći oblik.

Korolar B.3.2.

Ako je $f : D \to \mathbb{R}_0^+$ regularna funkcija, onda je

$$\deg_H(f) = a \quad (\geq 0),$$

ako i samo ako za svaki $\varepsilon > 0$ vrijedi

$$f(x) \in o(x^{a+\varepsilon}), \quad \frac{1}{f(x)} \in o(x^{-a+\varepsilon}).$$

Dokaz:

Direktno iz propozicije B.1.1. i teorema B.3.4. ■

Uočimo da dokaz ne ide direktno iz teorema B.3.3., jer $1/f$ nije regularna funkcija, pošto asimptotski monotono pada.

U terminima klasifikacije rasta funkcija, korolar B.3.2. može se i ovako izreći.

Teorem B.3.5.

Neka je $f : D \to \mathbb{R}_0^+$ regularna funkcija. Tada je f **najviše poliomnog rasta** *i vrijedi*

$$f \text{ je blagog rasta} \iff \deg_H(f) = 0,$$
$$f \text{ je polinomnog rasta} \iff \deg_H(f) > 0.$$

Dokaz:

Pretpostavimo da je f nadpolinomnog rasta, tj. da vrijedi

$$f(x) \in \Omega(x^a), \quad \forall a > 0.$$

Teorem B.3.1. tada daje

$$\deg_H(f) \geq a, \quad \forall a > 0,$$

što je nemoguće, ako je f regularna, odnosno pokazuje da vrijedi

$$\frac{f(cx)}{f(x)} \to \infty, \quad \forall c > 1. \tag{B.3.23}$$

■

Relacija (B.3.23) dokazuje i sljedeću tvrdnju.

Korolar B.3.3.

Asimptotski monotono rastuća funkcija nadpolinomnog rasta nije asimptotski homogena niti u jednoj točki $c \neq 1$.

Ova dva rezultata su izuzetno važna u analizi složenosti algoritama, jer imamo sljedeći zaključak:

Složenost algoritma ne ovisi bitno o veličini zadaće, samo ako je najviše polinomnog rasta.

Primjer B.3.1. *Funkcija* $f(x) = x^{\log x}$ *je blagog eksponencijalnog rasta. Za* $c > 0$ *je*

$$f(cx) = (cx)^{\log(cx)} = (cx)^{\log c + \log x} = c^{\log c}\, x^{2\log c}\, x^{\log x},$$

pa je

$$\frac{f(cx)}{f(x)} = c^{\log c}\, x^{2\log c} = (cx^2)^{\log c}.$$

Dobivamo

$$\lim_{x\to\infty} \frac{f(cx)}{f(x)} = 0, \quad za \quad 0 < c < 1,$$

tj. $f(cx) \in o(f(x))$*, za* $c < 1$*, i*

$$\lim_{x\to\infty} \frac{f(cx)}{f(x)} = \infty, \quad za \quad c > 1,$$

tj. $f(x) \in o(f(cx))$*. Dakle, za* $c \neq 1$*, funkcije* $f(cx)$ *i* $f(x)$ *nisu istog reda veličine.*

Literatura

[1] Aho, A.V., Hopcroft, J.E., Ullman, J.E.: *The Design and Analysis of Computer Algorithms*, Addison–Wesley, Reading, 1976.

[2] Borodin, A., Munro, I.: *The Computational Complexity of Algebraic and Numeric Problems*, American Elsevier, New York, 1975.

[3] Brent, R.P.: *A FORTRAN Multiple–Precision Arithmetic Package*, ACM Transactions on Mathematical Software, vol. 4, No. 1 (1978), 57–70.

[4] Brent, R.P.: *ALGORITHM 524, MP, A FORTRAN Multiple–Precision Arithmetic Package*, ACM Transactions on Mathematical Software, vol. 4, No. 1 (1978), 71–81.

[5] Brent, R.P.: *Fast Multiple–Precision Evaluation of Elementary Functions*, Journal of the Association for Computing Machinery, vol. 23, No. 2 (1976), 242–251.

[6] Collins, G.E., Mignotte, M., Winkler, F.: *Arithmetic in Basic Algebraic Domains*, U: Computer Algebra — Symbolic and Algebraic Computation, ed. by B. Buchberger, G.E. Collins, R. Loos, Springer–Verlag, Wien, 1982, (ruski prijevod, Mir, Moskva, 1986, 237–276.)

[7] Gonnet, G.H.: *Handbook of Algorithms and Data Structures*, Addison–Wesley, Reading, 1984.

[8] Knuth, D.E.: *The Art of Computer Programming, Vol.1 — Fundamental Algorithms (3rd ed.)*, Addison–Wesley, Reading, 1997.

[9] Knuth, D.E.: *The Art of Computer Programming, Vol.2 — Seminumerical Algorithms (3rd ed.)*, Addison–Wesley, Reading, 1998.

[10] Krishnamurthy, E.V., Nandi, S.K.: *On the Normalization Requirement of Divisor in Divide–and–Correct Methods*, Communications of the ACM, vol. 10, No. 12 (1967), 809–813.

[11] Nandi, S.K., Krishnamurthy, E.V.: *A Simple Technique for Digital Division*, Communications of the ACM, vol. 10, No. 5 (1967), 299–301.

[12] Regener, E.: *Multiprecision Integer Division Examples Using Arbitrary Radix*, ACM Transactions on Mathematical Software, vol. 10, No. 3 (1984), 325–328.

[13] Singer, S., Rogina, M.: *Quotient estimation in multiprecision division algorithms*, U: V Conference on Applied Mathematics, Ljubljana, September 2–5, 1986, ed. by Z. Bohte, Ljubljana, 1986, 133–138.

[14] Wilf, H.S.: *Algorithms and Complexity*, Prentice–Hall, Englewood Cliffs, 1986.

[15] Wilkinson, J.H.: *Rounding Errors in Algebraic Processes, Notes on Applied Science No. 32*, Her Majesty's Stationery Office, London; Prentice–Hall, Englewood Cliffs, 1963.

Uvod

Većina modernih digitalnih računala ima vrlo ograničene aritmetičke mogućnosti, barem iz perspektive matematičara.

Dozvoljene su, tj. procesorski realizirane samo četiri osnovne aritmetičke operacije i to svega za dvije kategorije objekata:

- brojevi u zapisu s fiksnom točkom ("fixed point"),
- brojevi u zapisu s pomičnom točkom (što je pomalo nespretan prijevod engleskog termina "floating point").

Prva vrsta objekata uglavnom odgovara cijelim brojevima u nekom rasponu, a pripadna aritmetika je, u stvari, aritmetika u prstenu ostataka modulo neki broj.

Za mnoge primjene potrebna je realizacija aritmetike u polju realnih brojeva. Da bi se povećao raspon prikazivih brojeva, posebno se pamti tzv. razlomljeni dio (ili mantisa), a posebno eksponent odabrane baze b, pa broj ima oblik

$$s \cdot m \cdot b^e,$$

gdje je s predznak, m je obično ograničen s

$$\frac{1}{b} \leq m < 1,$$

(osim za broj 0), a e je eksponent — cijeli broj. Ovo je takozvani normalizirani prikaz s pomičnom točkom. (Pomična točka, kao termin, potječe od toga što se, zapravo, u m pamti najznačajniji dio broja — neovisan o njegovoj stvarnoj veličini, a eksponent regulira stvarnu veličinu.)

Svaki dio u prikazu broja pamti se u obliku pozicionog zapisa u bazi b. Iz tehničkih razloga najčešća baza je $b = 2$, jer je najlakše realizirati memorijski element sa samo dva različita stabilna stanja, a toliko je dovoljno za pamćenje znamenki (0 ili 1) u binarnom sistemu.

Broj znamenki koje se pamte, je konačan i unaprijed fiksiran za svaki dio u prikazu broja. To ograničava raspon prikazivih brojeva (preko eksponenta) i skup prikazivih brojeva postaje diskretan (zbog konačnosti mantise). Osim toga,

to dovodi do nužne pojave grešaka zaokruživanja. Naime, skup prikazivih brojeva nije više zatvoren obzirom na osnovne aritmetičke operacije, pa umjesto egzaktnih operacija moramo obavljati približne, kako bi i rezultat operacije bio prikaziv. Dobivena algebarska struktura je vrlo komplicirana, i niz uobičajenih aksioma za polje realnih, odn. racionalnih brojeva više ne vrijedi. U detaljnu analizu te strukture nećemo ulaziti.

U većini primjena, najčešće je ukupni efekt grešaka zaokruživanja zanemariv i ovakva realizacija aritmetičkih operacija sasvim zadovoljava.

Osim osnovnih aritmetičkih operacija, sve ostale operacije i funkcije se realiziraju programski, svođenjem na osnovne, najčešće uz korištenje odgovarajućih aproksimacija ovisnih o karakteristikama aritmetike računala.

Realizacija ostalih algebarskih struktura, na pr. konačnih grupa, polja, polinoma, redova potencija i sl., mora se obaviti programski.

Osim toga, i u nekim numeričkim primjenama potrebno je izaći van granica aritmetike podržane arhitekturom računala, zbog zahtjeva na veću točnost operacija i (ili) veći raspon prikazivih brojeva.

Takvu aritmetiku nazivamo **aritmetikom visoke točnosti**.

U ovom radu promatramo algoritme za **programsku** realizaciju takve aritmetike na brojevima. Takvi algoritmi su obično bazirani na svojstvima objekata u drugačijim algebarskim strukturama, pa ćemo katkad promatrati i aritmetiku polinoma.

Osnovni cilj rada je pregledna formulacija i analiza niza algoritama u aritmetici visoke točnosti, i to na mašinski neovisnom nivou, onoliko koliko je to moguće. To znači da su izloženi algoritmi namijenjeni korištenju iz viših programskih jezika (FORTRAN, Pascal) sa zahtjevom na maksimalnu prenosivost s jednog računala na drugo. Zbog toga ne koristimo posebna svojstva aritmetike nekog računala, već je naglasak na efikasnoj formulaciji algoritma na globalnom matematičkom nivou. Naravno, za povećanje efikasnosti moguće je realizirati te algoritme i na mašinskom nivou — nivou osovnih instrukcija računala.

Rad se sastoji iz tri poglavlja.

U prvom poglavlju analiziramo opća svojstva funkcija kojima izražavamo složenost algoritama i problema. Posebna pažnja je posvećena problemu neovisnosti složenosti o načinu kodiranja ulaznih podataka. Taj problem je vrlo važan za analizu aritmetičkih algoritama, pošto broj zapisujemo (kodiramo) nizom znamenki u određenoj bazi. Fundamentalni rezultat je da složenost ne ovisi bitno o načinu kodiranja, ako i samo ako je najviše polinomnog rasta u ovisnosti o veličini zadaće (duljini ulaza). Primjenjen na pozicioni zapis brojeva u nekoj bazi, taj rezultat pokazuje da složenost aritmetičkih algoritama ne ovisi bitno o izboru baze.

Drugo poglavlje je posvećeno formulaciji i analizi algoritama za izvođenje četiri osnovne aritmetičke operacije u visokoj točnosti. Pokazujemo da zbrajanje i oduzimanje možemo optimalno realizirati. Aritmetička složenost algoritama za zbrajanje i oduzimanje je linearna. Zatim analiziramo klasične algoritme za množenje i dijeljenje. Posebna pažnja je posvećena efikasnoj realizaciji dijeljenja. Problem optimalne realizacije množenja i dijeljenja još uvijek nije riješen. Zbog toga dajemo niz, asimptotski sve bržih, algoritama za množenje i pokazujemo da su množenje i dijeljenje podjednako komplicirane operacije.

Izloženi rezultati pokazuju da su klasični algoritmi najbolji za relativno male duljine brojeva. Na kraju, primjenjujemo aritmetiku prirodnih brojeva za konstrukciju aritmetičkih algoritama za cijele i racionalne brojeve.

U trećem poglavlju izlažemo algoritme za "floating point" aritmetiku visoke točnosti.

Pri izboru prikaza brojeva i pripadnih aritmetičkih algoritama, naglasak je stavljen na praktično primjenjive postupke na klasičnim arhitekturama računala.

Svjesno je izostavljena analiza asimptotski najbržih poznatih algoritama, baziranih na brzoj Fourierovoj transformaciji, jer njihova brzina dolazi do izražaja tek za ogromne brojeve — daleko van granica prikazivosti u računalima.

Modularna aritmetika je, također, izostavljena, jer se u njoj vrlo teško realizira dijeljenje i uspoređivanje brojeva, a nije prikladna ni za realizaciju realne aritmetike.

Koristim ovu priliku da se najtoplije zahvalim svojem voditelju, prof. Emilu Coffou, koji je svojim ogromnim iskustvom, strpljivošću i savjetima pridonio da ovaj rad poprimi svoj konačni oblik.

Saša Singer

Sažetak

Osnovni cilj ovog rada je pažljiva analiza algoritama za izvođenje četiri osnovne aritmetičke operacije u visokoj točnosti. Brojeve prikazujemo u normaliziranom pozicionom zapisu s bazom b.

U prvom poglavlju razmatramo opća svojstva funkcija složenosti. Njih koristimo za dokaz neovisnosti složenosti aritmetičkih algoritama o izboru baze.

U drugom poglavlju formuliramo i analiziramo algoritme za cjelobrojnu i racionalnu aritmetiku.

Zadnje poglavlje je posvećeno realnoj aritmetici visoke točnosti.

Summary

The main purpose of this work is to make a careful study of algorithms for performing four basic arithmetic operations in high precision. Numbers are represented in the normalized radix b notation.

In the first chapter, general properties of complexity functions are considered. These are used to show the independence of complexity of arithmetical algorithms on the choice of radix b.

In the second chapter, integer and rational arithmetic algorithms are formulated and analyzed.

The final chapter deals with "floating point" high precision algorithms.

Biografija

Rođen sam u Zagrebu 5. 7. 1957. godine, gdje sam završio osnovnu školu i Matematičku gimnaziju. Godine 1975. upisao sam studij matematike na Prirodoslovno–matematičkom fakultetu Sveučilišta u Zagrebu. Diplomirao sam 1981. godine na smjeru inženjer matematike, struka praktična matematika i informatika. Iste godine upisao sam postdiplomski studij Prirodnih znanosti Sveučilišta u Zagrebu iz područja matematike.

Od 1. 2. 1982. obavljam dužnosti asistenta iz područja Numerička matematika i informatika na OOUR-u Matematički odjel Prirodoslovno–matematičkog fakulteta u Zagrebu.

Uže područje mog znanstvenog interesa je razrada i efikasna primjena numeričkih i nenumeričkih algoritama na računalima. Iz tog područja objavio sam jedan znanstveni i nekoliko stručnih radova. Od 1982. godine do danas učestvovao sam na nekoliko znanstvenih skupova unutar zemlje.