Analiza Evklidovega algoritma

Matjaž Urlep, 27004378

24. februar 2005

Povzetek

Evklidov algoritem, ki kot rezultat vrne največji skupni delitelj dveh celih števil, je v svoji osnovni obliki poznan že več kot 2000 let. Po kratkem zgodovinskem uvodu in opisu nekaterih možnih izboljšav in nadgradenj, sledi natančnejša časovna analiza osnovnega algoritma in njegove izboljšave, to je binarne metode. Na koncu si bomo pogledali še, kako bi lahko izboljšave prenesli na Evklidov algoritem na polinomih.

Kazalo

1. Uvod

Evklidov algoritem je bil prvič zapisan v njegovih Elementih okoli leta $300 BC$. Sicer obstajajo dvomi, ali je bil Evklid res njegov avtor, ali pa je bil poznan že prej, a dejstvo ostaja, da ga več kot 2000 let ni uspel nihče izpopolniti. Do tega je prišlo šele leta 1961, ko je J. Stein razvil binarno metodo. Pri tem je računsko zahtevno deljenje zamenjal s precej lažjim odštevanjem.

Med tem je prišlo do drugačnih sprememb algoritma, ki so se kazale predvsem v nadgradnji na razširjeni Evklidov algoritem, s pomočjo katerega dobimo celi števili x in y, ki rešita enačbo $ax + by = \gcd(a, b)$, ki je zelo uporaben tudi v moderni kriptografiji. Poleg tega se je v dobi računalnikov pojavilo vprašanje kako izboljšati algoritem, ko gre za velika števila. Tudi tu je imela kriptografija svojo vlogo, saj vemo, da se pri modernih računalnikih težko pogovarjamo o varnih kriptosistemih brez velikih števil. Leta 1938 je D.H.Lehmer hitrostno precej izpopolnil Evklidov algoritem za velika števila, ko je opazil, da lahko s pomočjo vodilnih števk precej zmanjšamo število operacij z velikimi števili.

V drugem poglavju bomo spoznali kaj največji skupni delitelj sploh je in kako ga lahko izračunamo. V tretjem poglavju bomo opisali osnovni Evklidov algoritem, kot ga je poznal že Evklid. V četrtem poglavju ga vzamemo pod drobnogled in se s pomočjo modelov lotimo njegove časovne analize, kjer obdelamo tako najslabši kot povprečni primer. V petem poglavju si pogledamo kako deluje razširjeni Evklidov algoritem. V šestem poglavju spoznamo binarno metodo, ki jo nato v sedmem poglavju časovno analiziramo. V osmem poglavju spoznamo, kako lahko z Lehmerjevo metodo Evklidov algoritem priredimo za velika števila.

2. Največji skupni delitelj

Evklidov algoritem je postopek za iskanje največjega skupnega delitelja. Tako si bomo najprej pogledali kaj največji skupni delitelj sploh je in nekaj njegovih lastnosti, ki jih bomo potrebovali kasneje. Iskanje največjega skupnega delitelja najdemo na različnih področjih.

Definicija. Naj bosta m in n celi števili. Njun *največji skupni delitelj* je največje naravno število, ki deli tako m kot n. Označimo ga z $\gcd(m, n)$, kar pride iz angleškega *greatest common divisor*. V posebnem primeru rečemo, da je $\gcd(0, 0) = 0$. Če je največji skupni delitelj enak 1, rečemo, da sta m in n *tuji si števili*.

Največji skupni delitelj lahko dobimo tako, da vsako od števil m in n razcepimo na prafaktorje, nato pa skupne prafaktorje, skupaj z manjšo od potenc, zmnožimo v željeni rezultat. Vendar razcepljanje na prafaktorje hitro predstavlja časovno zelo zahteven problem, zato takega načina v praksi ne uporabljamo. Da dobimo hitrejši postopek, se lahko poslužimo naslednjih lastnosti operatorja gcd, ki so takorekoč očitne.

Lema 1. *Za operator* gcd *velja*

$$\gcd(m,n) = \gcd(n,m),$$

$$\gcd(m,n) = \gcd(-m,n),$$

$$\gcd(m,0) = |m|,$$

$$\gcd(m,n) = \gcd(m,n-qm) \text{ za } \forall q \in \mathbb{Z},$$

$$\gcd(mk,nk) = k\gcd(m,n).$$

□

Predzadnja lastnost, z ustrezno izbranim q, nas pripelje do *Evklidovega algoritma.* S pomočjo zadnje lastnosti pa bomo prišli do izboljšave Evklidovega algoritma, imenovane *binarna metoda.* Preden si natančneje ogledamo obe metodi, si poglejmo še en zanimiv izrek, ki nam zagotavlja, da bomo imeli v več kot 60% primerih opravka s tujimi si števili.

Izrek 2. *Naj bosta m in n dve naključno izbrani celi števili. Potem je verjetnost, da je* $\gcd(m,n) = 1$ *enaka* $6/\pi^2 \approx 0.6079271$.

Dokaz. Naj bo p verjetnost, da je $\gcd(m,n) = 1$ in k poljubno naravno število. Potem je verjetnost, da je $\gcd(m,n) = k$, enaka p/k^2. To lahko utemeljimo tako, da rečemo, da je $\gcd(m,n) = k$ natanko takrat, ko je $\gcd(m/k, n/k) = 1$ in sta števili m in n večkratnika števila k. Če sedaj seštejemo verjetnosti po vseh k, dobimo

$$1 = \sum_{k\geq 1} \frac{p}{k^2} = p\sum_{k\geq 1} \frac{1}{k^2} = p\frac{\pi^2}{6}.$$

Za zadnjo enakost uporabimo znano vsoto (glej [1, str. 75]). Torej je res $p = 6/\pi^2$. □

3. Osnovni Evklidov algoritem

Poglejmo si kako je Evklidov algoritem definiran. Za dani celi števili m in n poišče njun največji skupni delitelj. Na vsakem koraku se zmanjša število n, pri tem pa $\gcd(m, n)$ ostane nespremenjen.

Algoritem 1. *(Evklid)*

```
dokler je m != 0 ponavljaj
   t = n mod m;
   n = m;
   m = t;
vrni n;
```

Pravilnost delovanja algoritma sledi po lemi 1.. Za lažjo predstavo si poglejmo na primeru kako izračunamo $\gcd(1982, 2004)$:

$$\gcd(1982, 2004) = \gcd(22, 1982) = \gcd(2, 22) = \gcd(0, 2) = 2.$$

Časovna zahtevnost Evklidovega algoritma za števili m in n na Knuthovem računalniku MIX je $19T + 6$ procesorskih operacij, kjer je T število potrebnih deljenj, ki ga bomo ocenili v naslednjem poglavju. Kako dobimo koeficienta 19 in 6, si lahko bralec pogleda v [2, str. 320].

4. Analiza osnovnega Evklidovega algoritma

Analiza bo tekla po naslednjih korakih. V prvem podpoglavju bomo definirali nekaj pojmov, ki jih bomo uporabljali pri analizi. V drugem podpoglavju bomo poiskali časovno najslabši primer in ga analizirali. Nazadnje bomo v tretjem podpoglavju najprej preko zveznega, nato pa še preko diskretnega modela analizirali povprečni primer.

4.1. Definicije

Najprej si poglejmo povezavo med Evklidovim algoritmom in verižnimi ulomki. Pri tem se bomo omejili na ulomke oblike

$$\frac{1}{x_1}, \frac{1}{x_1 + \frac{1}{x_2}}, \ldots,$$

kjer so x_i naravna števila.

Definicija. Za lažji zapis vpeljimo oznako

$$/x_1, \ldots, x_n/ = \begin{cases} \cfrac{1}{x_1 + \cfrac{1}{\ddots \cfrac{}{x_{n-1} + 1/x_n}}}, & \text{za } n > 0, \\ 0, & \text{za } n = 0. \end{cases}$$

Pri tem seveda velja $x_n \neq 0$.

Primer. Poglejmo si, kateri so prvi trije takšni verižni ulomki

$$/x_1/ = \frac{1}{x_1},$$

$$/x_1, x_2/ = \frac{1}{x_1 + 1/x_2} = \frac{x_2}{x_1 x_2 + 1},$$

$$/x_1, x_2, x_3/ = \frac{1}{x_1 + \cfrac{1}{x_2 + 1/x_3}} = \frac{x_2 x_3 + 1}{x_1 x_2 x_3 + x_1 + x_3}.$$

Ta oblika nas spodbudi, da vpeljemo polinome n-tih spremenljivk, s katerimi se je ukvarjal že Euler.

Definicija. Vzemimo produkt $x_1 x_2 \cdots x_n$ in iz njega črtajmo nič ali več disjunktnih parov zaporednih spremenljivk $x_i x_{i+1}$. Sedaj seštejmo dobljene produkte in dobimo polinome n-tih spremenljivk.

Prvih nekaj polinomov se glasi

$$Q_0 = 1,$$

$$Q_1(x_1) = x_1,$$

$$Q_2(x_1, x_2) = x_1 x_2 + 1,$$

$$Q_3(x_1, x_2, x_3) = x_1 x_2 x_3 + x_1 + x_3.$$

Za $n \geq 2$ velja naslednja rekurzivna zveza polinomov n-tih spremenljivk

$$Q_n(x_1, \ldots, x_n) = x_1 Q_{n-1}(x_2, \ldots, x_n) + Q_{n-2}(x_3, \ldots, x_n).$$

Opomba. Iz zgornje zveze z indukcijo sledi, da je število členov v dobljeni vsoti ravno $(n+1)$-vo Fibonaccijevo število F_n. Prva dva polinoma imata namreč ravno en člen, za nadaljnje polinome pa rekurzivna zveza pravi, da se število členov povečuje kot v Fibonaccijevem zaporedju.

Z indukcijo lahko hitro preverimo, da velja

$$/x_1, \ldots, x_n/ = \frac{Q_{n-1}(x_2, \ldots, x_n)}{Q_n(x_1, \ldots, x_n)}. \tag{1}$$

Velja namreč

$$\frac{Q_{n-1}(x_2, \ldots, x_n)}{Q_n(x_1, \ldots, x_n)} = \frac{Q_{n-1}(x_2, \ldots, x_n)}{x_1 Q_{n-1}(x_2, \ldots, x_n) + Q_{n-2}(x_3, \ldots, x_n)} =$$

$$= \frac{1}{x_1 + \dfrac{Q_{n-2}(x_3, \ldots, x_n)}{Q_{n-1}(x_2, \ldots, x_n)}} \overset{\text{I.P.}}{=} \frac{1}{x_1 + /x_2, \ldots, x_n/} = /x_1, \ldots, x_n/.$$

□

Rekurzivno formulo iz definicije lahko zapišemo tudi kot

$$Q_n(x_1, \ldots, x_n) = x_n Q_{n-1}(x_1, \ldots, x_{n-1}) + Q_{n-2}(x_1, \ldots, x_{n-2}).$$

Tako dobimo, če ponovimo prejšnji dokaz z novo rekurzivno formulo, da velja

$$Q_n(x_1, \ldots, x_n) = Q_n(x_n, \ldots, x_1).$$

Še pomembnejša za našo analizo pa bo naslednja lastnost, katere ki jo lahko dokažemo s pomočjo indukcije po n.

Trditev 3. *Za polinome Q_n, kjer je $n \geq 1$, velja naslednja zveza*

$$Q_n(x_1, \ldots, x_n)Q_n(x_2, \ldots, x_{n+1}) - Q_{n+1}(x_1, \ldots, x_{n+1})Q_{n-1}(x_2, \ldots, x_n) = (-1)^n.$$

□

Poglejmo si, kako lahko poljubno realno število X iz intervala $[0, 1)$ zapišemo s pomočjo verižnih ulomkov. Postavimo $X_0 = X$, nato pa za vsak $n \geq 0$, pri katerem $X_n \neq 0$, definirajmo

$$A_{n+1} = \left\lfloor \frac{1}{X_n} \right\rfloor,$$

$$X_{n+1} = \frac{1}{X_n} - A_{n+1}.$$

V primeru, da je $X_n = 0$ vrednosti A_{n+1} in X_{n+1} nista definirani in postopek se ustavi. Dobili smo verižni ulomek za X, ki je oblike $X = /A_1, \ldots, A_n/$. Če pa je $X_n \neq 0$, imamo $0 \leq X_{n+1} < 1$ in naravno število A_{n+1}.

Iz te definicije sledi

$$X = X_0 = \frac{1}{A_1 + X_1} = \frac{1}{A_1 + 1/(A_2 + X_2)} = \cdots$$

in tako

$$X = /A_1, \ldots, A_{n-1}, A_n + X_n/, \qquad \text{za } n \geq 1.$$

Če je $X_n \neq 0$ za vsak n imamo opravka z iracionalnimi števili, ki jih zapišemo kot

$$X = \lim_{n\to\infty} /A_1, \ldots, A_n/.$$

Iz rekurzivne zveze polinomov Q dobimo, da X leži med $/A_1, \ldots, A_n/$ in $/A_1, \ldots, A_n + 1/$.

Število X je racionalno natanko tedaj, ko obstaja tak n, da je $X_n = 0$. Vsako racionalno število lahko namreč zapišemo kot ulomek p/q. Ker smo na intervalu $[0, 1)$, velja $p < q$. Sedaj lahko preoblikujemo ulomek

$$\frac{p}{q} = \frac{1}{\frac{q}{p}} = \frac{1}{k + \frac{q - kp}{p}},$$

kjer je k kvocient, ki ga dobimo pri deljenju števila q s številom p. Jasno je, da je $q - kp < p$. Ta postopek nato ponovimo na ulomku $(q - kp)/p$ in tako dalje. Ker dobivamo v števcu vsakič manjše naravno število, se mora postopek končati v končno korakih. Konča se namreč, ko v števcu dobimo vrednost 0. V drugo smer je razmislek trivialen. Vsak končen verižni ulomek lahko namreč preoblikujemo v navaden ulomek, torej v neko racionalno število. □

Če je X racionalno število, lahko njegov verižni ulomek lepo povežemo z Evklidovim algoritmom. Naj bo $X = v/u$, kjer za naravni števili u in v velja $u > v \geq 0$. Pojdimo še enkrat skozi postopek generiranja verižnega ulomka. Postavimo $U_0 = u$, $V_0 = v$ in $X_n = V_n/U_n$, za vsako naravno število n. Potem velja

$$\begin{aligned} A_{n+1} &= \left\lfloor \frac{1}{X_n} \right\rfloor = \left\lfloor \frac{U_n}{V_n} \right\rfloor, \\ X_{n+1} &= \frac{1}{X_n} - A_{n+1} = \frac{U_n}{V_n} - \left\lfloor \frac{U_n}{V_n} \right\rfloor = \frac{U_n \bmod V_n}{V_n}. \end{aligned} \tag{2}$$

Če sedaj postavimo

$$U_{n+1} = V_n \qquad \text{in} \qquad V_{n+1} = U_n \bmod V_n,$$

res velja $X_n = V_n/U_n$ za vsako naravno število n. Poleg tega lahko vidimo, da sta zgornji enačbi ravno identični operacijama v enem koraku Evklidovega algoritma.

Primer. Vzemimo zopet naš prejšnji primer, ko smo iskali $\gcd(2004, 1982)$. Dobimo $\frac{1982}{2004} = /1, 90, 11/$, kar pomeni, da bo Evklidov algoritem potreboval 3 deljenja, koeficienti pa se bodo glasili zaporedoma 1, 90 in 11.

Opomba. Zadnji koeficient v razvoju racionalnega števila iz intervala $(0, 1)$ je A_n in je vedno večji ali enak 2 za $n \geq 1$. To sledi iz dejstva, da je X_{n-1} vedno manjši od 1.

Naj bo sedaj $X = v/u$ racionalno število. Potem za neko naravno število n velja $X = /A_1, \ldots, A_n/$ in iz (1) sledi

$$\frac{v}{u} = \frac{Q_{n-1}(A_2, \ldots, A_n)}{Q_n(A_1, \ldots, A_n)}.$$

Trditev 3. nam zagotavlja, da sta si polinoma $Q_{n-1}(A_2, \ldots, A_n)$ in $Q_n(A_1, \ldots, A_n)$ tuja, torej je

$$u = Q_n(A_1, \ldots, A_n)\ d \qquad \text{in} \qquad v = Q_{n-1}(A_2, \ldots, A_n)\ d,$$

kjer smo z d označili $\gcd(u, v)$.

4.2. Analiza najslabšega primera

Ugotoviti želimo koliko največ deljenj je potrebnih, da se Evklidov algoritem konča. Na to vprašanje nam odgovori naslednji izrek.

Izrek 4. *Naj bo $n \geq 1$ naravno število. Naj bosta $u > v > 0$ najmanjši taki naravni števili, da potrebuje Evklidov algoritem natanko n deljenj. Potem je $u = F_{n+1}$ in $v = F_{n+2}$, kjer smo s F_i označili i-ti člen Fibonaccijevega zaporedja.*

Dokaz. Iz (1) sledi $u = Q_n(A_1, \ldots, A_n)d$, kjer so $A_1, \ldots, A_n$ naravna števila. Vemo že, da je $A_n \geq 2$. Ker želimo, da bi bil u čim manjši, vzamemo $d = 1$, $A_1 = A_2 = \ldots = A_{n-1} = 1$ in $A_n = 2$. Že prej smo omenili, da je število členov v Q_n ravno $(n+1)$-vo Fibonaccijevo število F_n, v našem primeru pa so vsi členi razen zadnjega enaki 1, torej imamo za 1 premaknjeno Fibonaccijevo zaporedje, t.j. zaporedje z enako rekurzivno formulo, ki se začne s členoma $Q_0 = 1$, $Q_1 = 2$. Tako je res $u = F_{n+1}$. Podobno dobimo še $v = F_{n+2}$. □

S pomočjo zgornjega izreka lahko dokažemo naslednjo lemo o maksimalnem številu deljenj v Evklidovem algoritmu.

Lema 5. *Naj za naravni števili u in v velja $0 \leq u, v < N$. Potem je število deljenj v Evklidovem algoritmu največ $\lceil \log_\varphi(\sqrt{5}N) \rceil - 2$, kjer smo s $\varphi = \frac{\sqrt{5}+1}{2}$ označili zlati rez.*

Dokaz. Prejšnji izrek nam pravi, da bomo imeli največ korakov, t.j. n, takrat, ko bomo imeli opravka z dvema zaporednima Fibonaccijevima številoma F_n in F_{n+1}, kjer je n največje tako naravno število, da velja $F_{n+1} < N$. To pa pomeni, da je $\varphi^{n+1}/\sqrt{5} < N$ oziroma $n < \log_\varphi(\sqrt{5}N) - 1$. Torej je res $n \leq \lceil \log_\varphi(\sqrt{5}N) \rceil - 2$. □

4.3. Analiza povprečnega primera

Zvezni model. Poiskati želimo povprečno število deljenj v Evklidovem algoritmu, ki smo ga označili s T. Poglejmo si, kako se obnaša Evklidov algoritem na parih naravnih števil (u, v), če fiksiramo $v = N$. Vemo že, da je potek tesno povezano z računanjem verižnega ulomka za število$X \in \{0/N, 1/N, \ldots, (N-1)/N\}$. Če je število N zelo veliko, vidimo, da nas pravzaprav zanimajo verižni ulomki naključnega realnega števila iz intervala $[0, 1)$. Zato definirajmo porazdelitveno funkcijo

$$F_n(x) = P(X_n \leq x) \qquad \text{za } 0 \leq x \leq 1,$$

kjer je n naravno število in je X_n definiran z (2). Po definiciji verižnih ulomkov vemo, da je $F_0(x) = x$ in

$$F_{n+1}(x) = \sum_{k\geq 1} P\Big(k \leq \frac{1}{X_n} \leq k + x\Big) = \sum_{k\geq 1} P\Big(\frac{1}{k+x} \leq X_n \leq \frac{1}{k}\Big),$$

$$F_{n+1}(x) = \sum_{k\geq 1} \left(F_n\Big(\frac{1}{k}\Big) - F_n\Big(\frac{1}{k+x}\Big)\right).$$

Če bo za porazdelitev $F_0(x), F_1(x), \ldots$ veljalo $\lim_{n\to\infty} F_n(x) = F(x)$, bomo dobili zvezo

$$F(x) = \sum_{k\geq 1} \left(F\Big(\frac{1}{k}\Big) - F\Big(\frac{1}{k+x}\Big)\right).$$

Ena od funkcij, ki zadošča dobljeni zvezi je $F(x) = \log_b(1+x)$, pri poljubni logaritemski bazi $b > 1$. Velja namreč

$$\sum_{k\geq 1} \left(\log_b\Big(1+\frac{1}{k}\Big) - \log_b\Big(1+\frac{1}{k+x}\Big)\right) = \sum_{k\geq 1} \log_b \frac{(k+1)(k+x)}{k(k+x+1)} = \log_b\left(\prod_{k\geq 1} \frac{(k+1)(k+x)}{k(k+x+1)}\right),$$

$$\log_b\left(\prod_{k\geq 1} \frac{(k+1)(k+x)}{k(k+x+1)}\right) = \log_b \frac{2(x+1)}{1(x+2)} \cdot \frac{3(x+2)}{2(x+3)} \cdots = \log_b(1+x).$$

Iz pogoja $F(1) = 1$, dobimo $b = 2$. Torej vzamemo $F(x) = \log_2(1 + x)$. Nas pa zanima, kako se obnaša $F_n(x) - \log_2(1 + x) =: R_n(\log_2(1 + x))$.

Naj bo G poljubna funkcija, definirana na intervalu $[0, 1]$. Definirajmo operator S kot

$$SG(x) = \sum_{k\geq 1} \Big(G\Big(\frac{1}{k}\Big) - G\Big(\frac{1}{k+x}\Big)\Big).$$

To bi na naši porazdelitveni funkciji pomenilo, da je $F_{n+1}(x) = SF_n(x)$ in posledično $F_n(x) = S^n F_0(x)$. Operator S je linearen, ker je $S(kG) = k(SG)$ in $S(G_1 + G_2) = SG_1 + SG_2$. Če je G odvedljiva funkcija in je njen prvi odvod omejen, dobimo zvezo

$$(SG)'(x) = \sum_{k\geq 1} \frac{1}{(k+x)^2} G'\Big(\frac{1}{k+x}\Big).$$

Torej je tudi SG odvedljiva in je njen prvi odvod omejen. Označimo $H = SG$. Naj bo $g(x) = (1 + x)G'(x)$ in $h(x) = (1 + x)H'(x)$. Potem je

$$h(x) = \sum_{k\geq 1} \frac{1+x}{(k+x)^2}\Big(1 + \frac{1}{k+x}\Big)^{-1} g\Big(\frac{1}{k+x}\Big) = \sum_{k\geq 1} \Big(\frac{k}{k+1+x} - \frac{k-1}{k+x}\Big) g\Big(\frac{1}{k+x}\Big).$$

To pa je linearen operator na $g(x)$, ki ga označimo z U. Torej je

$$Ug(x) = \sum_{k\geq 1} \Big(\frac{k}{k+1+x} - \frac{k-1}{k+x}\Big) g\Big(\frac{1}{k+x}\Big).$$

Spet se pokaže, da če je funkcija g odvedljiva in je njen prvi odvod omejen, potem isto velja tudi za operator U. Tako lahko definiramo še tretji linearni operator V kot $V(g') = -(Ug)'$,

$$V\varphi(x) = \sum_{k\geq 1} \Big(\frac{k}{(k+1+x)^2} \int_{1/(k+1+x)}^{1/(k+x)} \varphi(t)dt + \frac{1+x}{(k+x)^3(k+1+x)} \varphi\Big(\frac{1}{k+x}\Big)\Big).$$

Poskusimo sedaj to povezati z našim problemom. Recimo, da je

$$F_n(x) = \log_2(1 + x) + R_n(\log_2(1 + x))$$

in

$$f_n(x) = (1 + x)F_n'(x) = \frac{1}{\ln 2}(1 + R_n'(\log_2(1 + x))).$$

Potem je

$$f_n'(x) = \frac{R_n''(\log_2(1 + x))}{(\ln 2)^2(1 + x))}.$$

Nadalje imamo zaradi $F_n = S^n F_0$ tudi $f_n = U^n f_0$ in $f_n' = (-1)^n V^n f_0'$. Tako dobimo

$$(-1)^n R_n''(\log_2(1 + x)) = (1 + x)(\ln 2)^2 V^n f_0'(x).$$

Sedaj upoštevajmo še, da je $F_0(x) = x$ in zato $f_0(x) = 1+x$ ter $f_0'(x) = 1$. Radi bi pokazali, da operator V^n na konstantni funkciji vrne funkcijo z zelo majhnimi vrednostmi. To bi pomenilo, da je $|R_n''(x)|$ zelo majhna za $0 \le x \le 1$. Odtod pa bi sledilo, da je tudi funkcija $R_n(x)$ zelo majhna, ker imamo $R_n(0) = R_n(1) = 0$. Z interpolacijo namreč dobimo

$$R_n(x) = -\frac{x(1-x)}{2} R_n''(\xi(x)),$$

za neko funkcijo $\xi(x)$, za katero velja $0 \le \xi(x) \le 1$, ko $0 \le x \le 1$.

Te zaključke moramo torej potrditi s tem, da pokažemo, da V^n res pretvori konstantne funkcije v funkcije z majhnimi vrednostmi. Opazimo, da je V pozitiven operator, saj je za $\varphi(x) \ge 0$ tudi $V\varphi(x) \ge 0$. Torej iz $\varphi_1(x) \le \varphi_2(x)$ sledi $V\varphi_1(x) \le V\varphi_2(x)$ za vsak x. To lastnost pa lahko izkoristimo, če poiščemo tako funkcijo $\varphi(x)$, za katero lahko natančno izračunamo $V\varphi(x)$. Najprej poiščimo tako funkcijo g, da bomo znali enostavno izračunati Ug. Če upoštevamo vse funkcije, ki so definirane za $x \ge 1$ in ne le na intervalu $[0, 1]$, dobimo

$$SG(x+1) - SG(x) = G\Big(\frac{1}{1+x}\Big) - \lim_{k\to\infty} G\Big(\frac{1}{k+x}\Big) = G\Big(\frac{1}{1+x}\Big) - G(0),$$

za zvezne funkcije G. Ker je $U((1+x)G') = (1+x)(SG)'$, imamo

$$\frac{Ug(x)}{x+1} - \frac{Ug(x+1)}{x+2} = \Big(\frac{1}{x+1} - \frac{1}{x+2}\Big) g\Big(\frac{1}{x+1}\Big).$$

Če si izberemo $Ug(x) = (x+1)^{-1}$, dobimo, da je $g(x) = 1 + x - (1+x)^{-1}$. Naj bo sedaj $\varphi(x) = g'(x) = 1 + (1+x)^{-2}$. Potem je $V\varphi(x) = (1+x)^{-2}$. Za tako izbrano funkcijo $\varphi(x)$ dobimo

$$\frac{1}{5}\,\varphi \le V\varphi \le \frac{1}{2}\,\varphi.$$

Ker sta V in φ pozitivni, lahko na zgornji neenačbi izračunamo V od vsakega člena in dobimo $\frac{1}{25}\varphi \le \frac{1}{5}V\varphi \le V^2\varphi \le \frac{1}{2}V\varphi \le \frac{1}{4}\varphi$. Induktivno dobimo po $n-1$ korakih

$$5^{-n}\varphi \le V^n\varphi \le 2^{-n}\varphi.$$

Poglejmo sedaj kaj naredi operator V na konstantni funkciji $\chi(x) = 1$. Za $0 \le x \le 1$ velja $\frac{5}{4}\chi \le \varphi \le 2\chi$ in tako

$$\frac{5}{8}\,5^{-n}\chi \le \frac{1}{2}\,5^{-n}\varphi \le \frac{1}{2}\,V^n\varphi \le V^n\chi \le \frac{4}{5}\,V^n\varphi \le \frac{4}{5}\,2^{-n}\varphi \le \frac{8}{5}2^{-n}\chi.$$

Ker je $f_0'(x) = 1$, je

$$V^n\chi(x) = V^n f_0'(x) = \frac{(-1)^n R_n''(\log_2(1+x))}{(1+x)(\ln 2)^2}.$$

Tako dobimo

$$\frac{5}{8}(\ln 2)^2 5^{-n} \leq (-1)^n R_n''(x) \leq \frac{16}{5}(\ln 2)^2 2^{-n}.$$

S tem smo zaključili dokaz naslednjega izreka.

Izrek 6. *Za porazdelitev $F_n(x)$ velja $\lim_{n\to\infty} F_n(x) = \log_2(1+x) + \mathcal{O}(2^{-n})$.*

Diskretni model. Dosedanje ugotovitve veljajo za vsa realna števila $0 \leq x \leq 1$. Ker pa je realno število le z verjetnostjo 0 tudi racionalno, bomo morali dobljene rezultate malo dopolniti, da bo izrek (6.) še vedno veljal.

Lema 7. *Naj bodo $I_1, I_2, \ldots$ in $J_1, J_2, \ldots$ paroma disjunktni podintervali intervala $[0,1)$. Naj bo še*

$$\mathcal{I} = \bigcup_{k\geq 1} I_k, \qquad \mathcal{J} = \bigcup_{k\geq 1} J_k, \qquad \mathcal{K} = [0,1] \setminus (\mathcal{I} \cup \mathcal{J}).$$

Predpostavimo, da ima $\mathcal{K}$ mero 0. Naj bo P_n množica $\{0, 1/n, \ldots, (n-1)/n\}$. Potem je

$$\lim_{n\to\infty} \frac{||\mathcal{I} \cap P_n||}{n} = \mu(\mathcal{I}),$$

kjer je $\mu(\mathcal{I})$ Lebesgue-ova mera množice $\mathcal{I}$.

Dokaz. Definirajmo $\mathcal{I}_N = \bigcup_{1\leq k\leq N} I_k$ in podobno $\mathcal{J}_N = \bigcup_{1\leq k\leq N} J_k$. Pri danem $\varepsilon > 0$ obstaja dovolj veliko naravno število N, da velja $\mu(\mathcal{I}_N) + \mu(\mathcal{J}_N) \geq 1 - \varepsilon$. Definirajmo še

$$\mathcal{K}_N = \mathcal{K} \cup \bigcup_{k>N} I_k \cup \bigcup_{k>N} J_k.$$

Če je $\mathcal{I}$ poljuben interval od a do b, je $\mu(\mathcal{I}) = b - a$ in

$$n\mu(\mathcal{I}) - 1 \leq ||\mathcal{I} \cap P_n|| \leq n\mu(\mathcal{I}) + 1.$$

Ker je $||\mathcal{I}_N \cap P_n|| + ||\mathcal{J}_N \cap P_n|| + ||\mathcal{K}_N \cap P_n|| = n$, dobimo

$$n\mu(\mathcal{I}_N) - N \leq ||\mathcal{I}_N \cap P_n|| \leq n\mu(\mathcal{I}_N) + N$$

in

$$n\mu(\mathcal{J}_N) - N \leq ||\mathcal{J}_N \cap P_n|| \leq n\mu(\mathcal{J}_N) + N.$$

Tako imamo

$$\mu(\mathcal{I}) - \frac{N}{n} - \varepsilon \leq \mu(\mathcal{I}_N) - \frac{N}{n} \leq \frac{||\mathcal{I}_N \cap P_n||}{n} \leq \frac{||\mathcal{I}_N \cap P_n|| + ||\mathcal{K}_N \cap P_n||}{n} =$$

$$= 1 - \frac{||\mathcal{J}_N \cap P_n||}{n} \leq 1 - \mu(\mathcal{J}_N) + \frac{N}{n} \leq \mu(\mathcal{I}) + \frac{N}{n} + \varepsilon.$$

Ker to velja za poljuben ε in poljuben n, dobimo

$$\lim_{n\to\infty} \frac{||\mathcal{I}_N \cap P_n||}{n} = \mu(\mathcal{I}).$$

□

Porazdelitev kvocientov. Zdaj lahko združimo ugotovitve izreka (6.) in leme (7.) v dokazu naslednjega izreka.

Izrek 8. *Naj bosta n in k naravni števili in naj bo $p_k(a, n)$ verjetnost, da je $(k+1)$-vi kvocient A_{k+1} v Evklidovem algoritmu enak a, če je $v = n$ in je u izbran naključno. Potem velja*

$$\lim_{n\to\infty} p_k(a, n) = F_k\Big(\frac{1}{a}\Big) - F_k\Big(\frac{1}{a+1}\Big),$$

kjer je $F_k(x) = P(X_k \leq x)$.

Dokaz. Označimo z $\mathcal{I}$ množico vseh X-ov na intervalu $[0, 1)$ za katere je $A_{k+1} = a$. To je unija disjunktnih intervalov. Naj bo še $\mathcal{J}$ množica vseh X-ov za katere je $A_{k+1} \neq a$. Torej lahko uporabimo lemo (7.). Pri tem s $\mathcal{K}$ označimo množico vseh X-ov, kjer A_{k+1} ni definiran. Vidimo lahko, da je $F_k(1/a) - F_k(1/(a+1))$ verjetnost, da je $1/(a+1) < X_k \leq 1/a$. To pa je ravno $\mu(\mathcal{I})$, ki predstavlja verjetnost, da je $A_{k+1} = a$. □

Posledica 9. *V enem koraku Evklidovega algoritma nastopi koeficient a z verjetnostjo približno*

$$\log_2\Big(1 + \frac{1}{a}\Big) - \log_2\Big(1 + \frac{1}{a+1}\Big) = \log_2\Big(\frac{(a+1)^2}{(a+1)^2 - 1}\Big).$$

Primer. Kvocient 1 nastopa v približno $\log_2(\frac{4}{3}) \approx 41,5\%$ primerih, kvocient 2 v približno $\log_2(\frac{9}{8}) \approx 17,0\%$ primerih, kvocient 3 pa samo še v približno $\log_2(\frac{16}{15}) \approx 9,3\%$ primerih.

Opomba. Obnašanje, kot ga predvidevajo zgornji izsledki, lahko pričakujemo le pri koeficientih A_k, kjer je k dovolj majhen v primerjavi s številom deljenj v Evklidovem algoritmu.

Poskusimo sedaj pogledati, kako se obnašajo zadnji kvocienti v Evklidovem algoritmu. Najbolje bo, da si pogledamo na primeru.

Primer. Poglejmo si verižne ulomke, katerih imenovalec je 29.

$$\begin{array}{llllllllllll}
\frac{1}{29} & = & /29/ & \frac{2}{29} & = & /3,1,1,1,2/ & \frac{3}{29} & = & /1,1,14/ & \frac{4}{29} & = & /1,3,7/ \\
\frac{5}{29} & = & /14,2/ & \frac{6}{29} & = & /3,4,2/ & \frac{7}{29} & = & /1,1,4,3/ & \frac{8}{29} & = & /1,3,1,5/ \\
\frac{9}{29} & = & /9,1,2/ & \frac{10}{29} & = & /2,1,9/ & \frac{11}{29} & = & /1,1,2,2,2/ & \frac{12}{29} & = & /1,4,1,4/ \\
\frac{13}{29} & = & /7,4/ & \frac{14}{29} & = & /2,1,1,1,3/ & \frac{15}{29} & = & /1,1,1,1,1,3/ & \frac{16}{29} & = & /1,6,4/ \\
\frac{17}{29} & = & /5,1,4/ & \frac{18}{29} & = & /2,2,2,2/ & \frac{19}{29} & = & /1,1,1,9/ & \frac{20}{29} & = & /1,8,1,2/ \\
\frac{21}{29} & = & /4,1,5/ & \frac{22}{29} & = & /2,4,3/ & \frac{23}{29} & = & /1,2,4,2/ & \frac{24}{29} & = & /1,13,2/ \\
\frac{25}{29} & = & /4,7/ & \frac{26}{29} & = & /2,14/ & \frac{27}{29} & = & /1,2,1,1,1,2/ & \frac{28}{29} & = & /1,28/
\end{array}$$

Vidimo lahko, da je zadnji kvocient res vedno večji ali enak 2. Verižni ulomki v zadnjih dveh stolcih imajo posebno povezavo z verižnimi ulomki v prvih dveh stolpcih

$$1 - /x_1, \ldots, x_n/ = /1, x_1 - 1, x_2, \ldots, x_n/.$$

Opazimo lahko, da imamo v prvih dveh stolpcih simetrijo - če nastopi verižni ulomek $/x_1, \ldots, x_n/$, nastopi tudi $/x_n, \ldots, x_1/$. Preverimo lahko tudi kako dobro smo ocenili pojavitve posameznih kvocientov. Izkaže se, da se 1 pojavi v $\frac{39}{96} \approx 40,6\%$, 2 v $\frac{21}{96} \approx 21,9\%$ in 3 v $\frac{8}{96} \approx 8,3\%$ primerih. Te vrednosti se dokaj dobro skladajo s prejšnjimi ugotovitvami.

Število korakov Evklidovega algoritma. Vrnimo se sedaj k ocenjevanju povprečnega števila deljenj, pri $v = n$, ki smo ga označili s T_n. V Knuthovi knjigi (glej [2, str. 353]) najdemo nekaj primerov T_n pri izbranih n. Pa si jih poglejmo.

n	=	95	96	97	98	99	100	101	102	103	104	105
T_n	=	5,0	4,4	5,3	4,8	4,7	4,6	5,3	4,6	5,3	4,7	4,6
n	=	996	997	998	999	1000	1001		...	9999	10000	10001
T_n	=	6,5	7,3	7,0	6,8	6,4	6,7		...	8,6	8,3	9,1
n	=	49999	50000	50001			...			99999	100000	100001
T_n	=	10,6	9,7	10,0			...			10,7	10,3	11,0

Vidimo lahko, da je T_n opazno višji glede na sosede, kadar je $n \in \{97, 101, 103, 997, 49999\}$ praštevilo in opazno nižji, kadar ima $n \in \{96, 1000, 50000\}$ veliko deliteljev. To lahko enostavno pojasnimo. Naj bo $\gcd(u, v) = d$. Potem se Evklidov algoritem na u/d in v/d obnaša praktično enako kot na u in v. Tako je pri v-jih, ki imajo veliko število deliteljev, velika izbira u-jev, za katere se $v = n$ obnaša kot bi bil manjši. Poskusimo definirati novo vrednost, ki jo bomo označili s τ_n, in bo pomenila povprečno število deljenj, ko je $v = n$ in je u tuj n. Torej je

$$\tau_n = \frac{1}{\varphi(n)} \sum_{\substack{0 \le m < n \\ \gcd(m,n)=1}} T(m, n),$$

kjer je $\varphi(n)$ Eulerjeva φ-funkcija, ki pove koliko deliteljev ima število n, $T(m,n)$ pa predstavlja število deljenj v Evklidovem algoritmu na številih m in n. Torej lahko izrazimo T_n kot

$$T_n = \frac{1}{n}\sum_{d\backslash n}\varphi(d)\tau_d.$$

Poglejmo si še tabelo vrednosti τ_n pri istih n kot v prejšnjem primeru.

n	=	95	96	97	98	99	100	101	102	103	104	105
τ_n	=	$5,4$	$5,3$	$5,3$	$5,6$	$5,2$	$5,2$	$5,4$	$5,3$	$5,4$	$5,3$	$5,6$
n	=	996	997	998	999	1000	1001	...		9999	10000	10001
τ_n	=	$7,2$	$7,3$	$7,3$	$7,3$	$7,3$	$7,4$	...		$9,21$	$9,21$	$9,22$
n	=	49999	50000	50001			...			99999	100000	100001
τ_n	=	$10,58$	$10,57$	$10,59$			...			$11,170$	$11,172$	$11,172$

Takoj lahko vidimo, da je τ_n lepše porazdeljen in tako bolj primeren za analizo. Izkaže se, da lahko precej dobro ocenimo τ_n kot

$$\tau_n \approx 0,843\ln n + 1,47.$$

Poskusimo pojasniti takšno obnašanje. Spomnimo se, da smo definirali $X_k = V_k/U_k$. Poglejmo si kakšna je vrednost $X_0\cdots X_{t-1}$, kjer je t število deljenj.

$$X_0\cdots X_{t-1} = \frac{V_0}{U_0}\,\frac{V_1}{U_1}\cdots\frac{V_{t-1}}{U_{t-1}} = \frac{V_{t-1}}{U_0} = \frac{1}{U}.$$

Pri tem smo upoštevali, da je $U_{k+1} = V_k$ in privzeli, da sta $U = U_0$ in $V = V_0$ tuji si števili. Naj bo še $V < U$. Potem je

$$\ln X_0 + \cdots + \ln X_{t-1} = -\ln U.$$

Ker poznamo približno distribucijo $X_0, X_1, \ldots$, lahko ocenimo

$$t := T(U,V) = T(V,U) - 1.$$

Naj bo X_0 realno število, enakomerno porazdeljeno na intervalu $[0,1)$. Potem je povprečna vrednost $\ln X_n$ enaka

$$\int_0^1 \ln x F_n'(x)dx = \int_0^1 \frac{\ln x f_n(x)}{1+x}dx.$$

Prej smo pokazali, da je $F_n = \log_2(1+x) + \mathcal{O}(2^{-n})$. Tako imamo

$$f_n(x) = (1+x)F_n'(x) = (1+x)\Big(\frac{1}{(1+x)\ln 2} + \mathcal{O}(2^{-n})\Big) = \frac{1}{\ln 2} + \mathcal{O}(2^{-n}).$$

Tako lahko nadalje ocenimo povprečno vrednost $\ln X_n$ kot

$$\int_0^1 \frac{\ln x f_n(x)}{1+x} dx \approx \frac{1}{\ln 2} \int_0^1 \frac{\ln x}{1+x} dx = -\frac{\pi^2}{12 \ln 2}.$$

Dobimo

$$-\frac{\pi^2 t}{12 \ln 2} \approx -\ln U.$$

Oziroma povedano drugače

$$t \approx \frac{12 \ln 2}{\pi^2} \ln U = 0,842765913 \ln U.$$

Konstanta $12 \ln 2/\pi^2$ pa se zelo dobro ujema s konstanto $0,843$, s katero smo prej ocenili τ_n. Tako je zelo verjetno, da velja

$$\tau_n \approx \frac{12 \ln 2}{\pi^2} \ln n + 1,47,$$

ko gre $n \to \infty$. Knuth v svoji knjigi preoblikuje približek za T_n v

$$T_n \approx \frac{12 \ln 2}{\pi^2} \Big(\ln n - \sum_{d \backslash n} \frac{\Lambda(d)}{d} \Big) + 1,47,$$

kjer je $\Lambda(n)$ von Mongoldt-ova funkcija, definirana kot

$$\Lambda(n) = \begin{cases} \ln p, & \text{če je } n = p^r, \text{ kjer je } p \text{ praštevilo in je } r \geq 1, \\ 0, & \text{sicer.} \end{cases}$$

Primer. Izračunajmo koliko se vrednost za T_{99} izračunana po tej formuli razlikuje od prave, ki je približno $4,7$.

$$T_{99} \approx \frac{12 \ln 2}{\pi^2} \left(\ln 99 - \frac{\ln 3}{3} - \frac{\ln 3}{9} - \frac{\ln 11}{11} \right) + 1,47 \approx 4,7.$$

Torej smo vsaj na decimalko natančno zadeli pravo vrednost.

Sedaj lahko izračunamo tudi povprečno število deljenj, če sta u in v enakomerno porazdeljena med 1 in N

$$\frac{1}{N} \sum_{1 \leq n \leq N} T_n = \frac{12 \ln 2}{\pi^2} \ln N + \mathcal{O}(1).$$

Kar se lepo sklada z rezultati, ki jih dobimo računsko, ko smo dobili, da je povprečno število deljenj

$$\frac{12 \ln 2}{\pi^2} \ln N + 0,06.$$

5. Razširjen Evklidov algoritem

Osnovni Evklidov algoritem, ki smo ga dodobra spoznali v prejšnjem podpoglavju, lahko enostavno razširimo tako, da dobimo poleg največjega skupnega delitelja tudi rešitvi u_1 in u_2 diofantske enačbe

$$uu_1 + vu_2 = \gcd(u, v).$$

Algoritem 2. *Razširjeni Evklidov algoritem*

```
u_1 = 1, u_3= u;
v_1 = 0, v_3 = v;
dokler je v_3 > 0
   q = u_3 / v_3;
   t_1 = u_1 - v_1 q;
   t_3 = u_3 - v_3 q;
   u_1 = v_1;
   u_3 = v_3;
   v_1 = t_1;
   v_3 = t_3;
u_2 = (u_3 - u * u_1) / v;
vrni u_1, u_2, u_3;
```

Opomba. V spremenljivki u_3 se skriva največji skupni delitelj števil u in v.

Primer. Poglejmo si kako bi opisani algoritem tekel na številih $u = 2004$ in $v = 1982$.

q	u_1	u_3	v_1	v_3
/	1	2004	0	1982
1	0	1982	−90	22
90	1	22	991	2
11	−90	2	1	0

V zadnji vrstici je v_3 enak 0, zato se algoritem ustavi. Sedaj potrebujemo izračunati še u_2.

$$u_2 = \frac{u_3 - u \cdot u_1}{v} = \frac{2 + 2004 \cdot 90}{1982} = 91.$$

Tako smo dobili rešitev

$$-90 \cdot 2004 + 91 \cdot 1982 = 2 = \gcd(2004, 1982).$$

Jasno je, da ima zanka v razširjenem Evklidovem algoritemu enako število korakov, kot v osnovnem algoritmu. Torej velja tudi ista ocena za število deljenj. Razlika je le v koeficientih, ki sta v tem primeru nekoliko višja. Ta razširitev osnovnega Evklidovega algoritma je zelo uporabna tudi v kriptografiji. Z njeno pomočjo lahko hitro računamo inverze elementov po nekem modulu. Takrat namreč rešujemo enačbo

$$uu^{-1} \equiv 1 \pmod{v},$$

ki jo lahko zapišemo tudi drugače kot

$$uu^{-1} + kv = 1.$$

Ker inverz elementa u po modulu v obstaja le, če sta si u in v tuji si števili, lahko zapišemo tudi

$$uu^{-1} + kv = \gcd(u, v).$$

Tako dobimo ravno diofantsko enačbo, ki jo znamo rešiti z opisanim algoritmom.

6. Binarna metoda

Binarna metoda predstavlja malo drugačen pristop k iskanju največjega skupnega delitelja dveh celih števil. Sestavljajo jo operacije odštevanje, pomik v desno in preverjanje sodosti. Osnova za nastanek metode je naslednja lema.

Lema 10. *Za naravni števili m in n veljajo naslednje lastnosti:*

- *Če sta m in n sodi števili, potem je* $\gcd(m, n) = \gcd(m/2, n/2)$*;*
- *Če je m sodo in n liho število, je* $\gcd(m, n) = \gcd(m/2, n)$*;*
- *Če sta m in n lihi števili, je $|m - n|$ sodo in velja* $|m - n| < \max(m, n)$.

Če povežemo točke iz leme 10., dobimo naslednji algoritem:

Algoritem 3. *Binarna metoda*

```
k = 0;
dokler sta m in n sodi števili
   k = k + 1;
```

```
   m = m / 2;
   n = n / 2;
če je m lih potem t = -n sicer t = m;
dokler t != 0
   dokler je t sod
      t = t / 2;
   če je t > 0 potem m = t sicer n = -t;
   t = m - n;
vrni m * 2 ^ k;
```

Primer. Poglejmo kako teče binarna metoda pri računanju največjega skupnega delitelja števil 2004 in 1982. Prva zanka naredi en korak. Dobimo $k = 1$, $m = 1002$ in $n = 991$. Postavimo še $t = m = 1002$. Korake druge zanke bomo zapisali v tabeli, kjer bomo notranjo zanko izvajali kar v zadnjem stolpcu, korake pa bomo ločili z vejico.

m	n	t
1002	991	$1002, 501$
501	991	$-490, -245$
501	245	$256, 128, 64, 32, 16, 8, 4, 2, 1$
1	245	$-244, -122, -61$
1	61	$-60, -30, -15$
1	15	$-14, -7$
1	7	$-6, -3$
1	3	$-2, -1$
1	1	0

Tako smo dobili rešitev

$$\gcd(2004, 1982) = m \cdot 2^k = 1 \cdot 2^1 = 2.$$

V tabeli zgornjega primera lahko opazimo, da smo že v četrti vrstici dobili $m = 1$. Takrat bi že lahko končali z računanjem, saj je jasno, da bo nadaljnje računanje privedlo do $n = 1$, medtem ko bo m ostal ves čas enak 1. Takšno prekinitev bi lahko dosegli z enim odločitvenim stavkom. Vprašanje pa je, v koliko primerih pride do takšne situacije, da m postane 1. To se zgodi natanko takrat, ko je največji skupni delitelj potenca števila 2. V dokazu izreka 2. smo videli, da je verjetnost, da je $\gcd(m, n) = 2^k$ enaka $p/4^k$. Če

seštejmo te verjetnosti po vseh k, dobimo

$$\sum_{k=0}^{\infty} \frac{p}{4^k} = p \sum_{k=0}^{\infty} \frac{1}{4^k} = \frac{4p}{3} = \frac{8}{\pi^2} \approx 0.8105695.$$

Torej se to zgodi v kar 81% primerih.

7. Analiza binarne metode

V tem razdelku si bomo natančneje pogledali časovno zahtevnost binarne metode. Na prvi pogled bi lahko rekli, da porabi binarna metoda več korakov in imeli bi prav. Vendar pa moramo upoštevati, da pri tem ne uporabljamo nobenega deljenja, ki predstavlja računsko zahtevnejšo operacijo od odštevanja in desnega pomika, s katerima smo ga zamenjali. Tako se pokaže, da je povprečna časovna zahtevnost osnovnega Evklidovega algoritma $11,1k+7$, če vzamemo m in n iz intervala $1 \leq m, n < 2^k$, medtem ko ima binarna metoda povprečno časovno zahtevnost $8,8k+5$. Še občutnejša je razlika pri najslabši časovni zahtevnosti, kjer imamo v prvem primeru $26,8k+19$, v drugem pa $12,8k+8$. Pri tem velja omeniti, da je govora o operacijah na Knuthovem računalniku MIX.

Z A označimo število deljenj z 2 na začetku algoritma Torej $A = k$. Če je m po deljenju z 2 še vedno sod, je $B = 1$, sicer je $B = 0$. Število odštevanj označimo s C, število preostalih deljenj z 2 pa z D. Kolikokrat je $t > 0$ pa označimo z E. Tako dobimo, da je časovna zahtevnost binarne metode na Knuthovem računalniku MIX

$$9A + 2B + 6C + 3D + E + 13.$$

Za lažjo obravnavo problema bomo začeli s poenostavljenim modelom. Predpostavili bomo, da sta m in n lihi števili, in da je $m > n$. Prav tako bomo privzeli, da je $\lfloor \log_2 m \rfloor = u$ in $\lfloor \log_2 n \rfloor = v$. V binarni metodi imamo operacijo $m-n$, katere rezultat nato pomikamo v desno dokler ne dobimo lihe številke m', ki zamenja m. Tako lahko pričakujemo, da bo pri naključno izbranih m in n v približno polovici primerov $m' = (m-n)/2$, v približno četrtini primerov $m' = (m-n)/4$ in tako naprej. Tako dobimo

$$\lfloor \log_2 m' \rfloor = u - k - r,$$

kjer je k število števk, ki jih izgubimo zaradi desnih pomikov, r pa je $\lfloor \log_2 m \rfloor - \lfloor \log_2 (m-n) \rfloor$, torej število mest, ki jih izgubimo na levi zaradi odštevanja. Seveda je jasno, da bo $r \geq 1$ le v primeru, ko bo $u = v$. Sicer bo $r \leq 1$. Zaradi enostavnosti privzemimo, da je $r = 0$, kadar je $u \neq v$ in $r = 1$ sicer.

Naj par (u, v) predstavlja točko v koordinatnem sistemu. To pomeni, da nas binarna metoda ponese v točko (u', v), če je $m > n$, v (u, v'), če je $m < n$, oziroma se konča, če je $m = n$. Grobo lahko ocenimo, kakšna je verjetnost, da bomo prišli iz točke (u, v) v poljubno drugo točko. V primeru, da je $u > v$, bomo prišli v točko $(u-w, v)$ z verjetnostjo 2^{-w} za $w < u$ in v točko $(0, v)$ z verjetnostjo 2^{1-u}. V primeru, da je $u < v$ so verjetnosti simetrične prvemu primeru. Preostane nam še primer, ko je $u = v > 0$. Tudi tukaj so verjetnosti enake, razlika je le v verjetnosti, da zadanemo točko $(0, v)$ oziroma $(u, 0)$, ki je 2^{-u}. V zadnjem primeru pa se lahko zgodi tudi, da se algoritem konča. Verjetnost za to je 2^{1-u}. Hitro lahko preverimo, da nam vsota verjetnosti v vsakem od treh primerov da 1. Te ocene verjetnosti smo naredili predvsem zato, ker bomo s tem lahko naš model zelo dobro analizirali. Zavedati pa se je treba, da pri majhnih razlikah v dolžinah m in n pride do hitrejših konvergenc.

Analizo zgornjega modela lahko prestavimo na reševanje rekurzivnih enačb

$$A_{uu} = a + \frac{1}{2}A_{u(u-2)} + \cdots + \frac{1}{2^{u-1}}A_{u0} + \frac{b}{2^{u-1}}, \qquad \text{za } u \geq 1;$$

$$A_{uv} = c + \frac{1}{2}A_{(u-1)v} + \cdots + \frac{1}{2^{u-1}}A_{1v} + \frac{1}{2^{u-1}}A_{0v}, \qquad \text{za } u > v \geq 0;$$

$$A_{uv} = A_{vu}, \qquad \text{za } v > u \geq 0.$$

Iščemo torej matriko A in parametre a, b, c in A_{00}.

Ker je matrika A simetrična, lahko v nadaljnjih računih privzamemo, da je $u \geq v$. Za $u > v$ dobimo

$$A_{(u+1)v} = c + \sum_{k=1}^{u} 2^{-k}A_{(u+1-k)v} + 2^{-u}A_{0v}.$$

Vzemimo iz vsote prvi člen in popravimo indeks k. Dobimo

$$A_{(u+1)v} = c + \frac{1}{2}A_{uv} + \sum_{k=1}^{u-1} 2^{-k-1}A_{(u-k)v} + 2^{-u}A_{0v}.$$

Opazimo, da se ista vsota skupaj z zadnjim členom pojavi v definiciji za A_{uv} in dobimo

$$A_{(u+1)v} = c + \frac{1}{2}A_{uv} + \frac{1}{2}(A_{uv} - c).$$

Sedaj le še seštejmo člene skupaj v

$$A_{(u+1)v} = \frac{1}{2}c + A_{uv}.$$

Podobno lahko z indukcijo po k pokažemo, da je $A_{(u+k)v} = \frac{1}{2}ck + A_{uv}$. Posebej dobimo še $A_{10} = c + A_{00}$ in tako $A_{u0} = \frac{1}{2}c(u+1) + A_{00}$ za $u > 0$.

Preostane nam še izračun $A_{(u+1)u}$.

$$A_{(u+1)u} = c + \sum_{k=1}^{u+1} 2^{-k} A_{(u+1-k)u} + 2^{-u-1} A_{0u}.$$

Ponovno is vsote izločimo prvi člen in popravimo indeks k. Dobimo

$$A_{(u+1)u} = c + \frac{1}{2}A_{uu} + \sum_{k=1}^{u} 2^{-k-1} A_{(u-k)u} + 2^{-u-1} A_{0u}.$$

Upoštevajmo prejšnji rezultat, da je $A_{(u-k)u} = A_{(u-k)(u+1)} - c/2$. Dobimo

$$A_{(u+1)u} = c + \frac{1}{2}A_{uu} + \sum_{k=1}^{u} \Big(2^{-k-1}(A_{(u-k)(u+1)} - c/2)\Big) + 2^{-u-1} A_{0u}.$$

Opazimo, da se podobna vsota pojavi v definiciji za A_{uu}

$$A_{(u+1)u} = c + \frac{1}{2}A_{uu} + \frac{1}{2}\big(A_{(u+1)(u+1)} - a - 2^{-u}b\big) - \frac{1}{4}c(1-2^{-u}) + 2^{-u-1}\big(\frac{1}{2}c(u+1) + A_{00}\big).$$

Na koncu zapišimo še malo lepše

$$A_{(u+1)u} = \frac{1}{2}(A_{uu} + A_{(u+1)(u+1)}) + \frac{3}{4}c - \frac{1}{2}a + 2^{-u-1}(c - b + A_{00}) + u2^{-u-2}c.$$

Za lažjo berljivost označimo A_{uu} z A_u. Z nekaj premetavanja enačb dobimo naslednjo zvezo

$$A_{u+1} = \frac{3}{4}A_u + \frac{1}{4}A_{u-1} + \alpha + 2^{-u-1}(\beta + (u+2)\gamma), \qquad u \geq 2,$$

kjer je $\alpha = \frac{1}{4}a + \frac{7}{8}c$, $\beta = A_0 - b - \frac{3}{2}c$ in $\gamma = \frac{1}{2}c$.

Tako smo iz matrike pridelali vektor. Z uporabo rodovne funkcije $G(z) = A_0 + A_1 z + A_2 z^2 + \cdots$ dobimo enačbo

$$\Big(1 - \frac{3}{4}z - \frac{1}{4}z^2\Big)G(z) = a_0 + a_1 z + a_2 z^2 + \frac{\alpha}{1-z} + \frac{\beta}{1-z/2} + \frac{\gamma}{(1-z/2)^2},$$

kjer so a_0, a_1 in a_2 konstante odvisne od A_0, A_1 in A_2. Ker je $(1 - \frac{3}{4}z - \frac{1}{4}z^2) = (1 + z/4)(1-z)$, lahko enačbo s pomočjo parcialnih ulomkov preoblikujemo v

$$G(z) = b_0 + b_1 z + \frac{b_2}{(1-z)^2} + \frac{b_3}{1-z} + \frac{b_4}{(1-z/2)^2} + \frac{b_5}{1-z/2} + \frac{b_6}{1+z/4}.$$

Z nekaj računanja določimo koeficiente $b_0, \ldots, b_6$ in dobimo rešitev

$$A_{uu} = u\Big(\frac{1}{5}a + \frac{7}{10}c\Big) + \Big(\frac{16}{25}a + \frac{2}{5}b - \frac{23}{50}c + \frac{3}{5}A_{00}\Big) + 2^{-u}\Big(-\frac{1}{3}cu + \frac{2}{3}b - \frac{1}{9}c - \frac{2}{3}A_{00}\Big)$$
$$+\Big(-\frac{1}{4}\Big)^u\Big(-\frac{16}{25}a - \frac{16}{15}b + \frac{16}{225}c + \frac{16}{15}A_{00}\Big) + \frac{1}{2}c\delta_{u0};$$

$$A_{uv} = \frac{1}{2}uc + v\Big(\frac{1}{5}a + \frac{1}{5}c\Big) + \Big(\frac{6}{25}a + \frac{2}{5}b + \frac{7}{50}c + \frac{3}{5}A_{00}\Big) + 2^{-v}\Big(\frac{1}{3}c\Big)$$
$$+\Big(-\frac{1}{4}\Big)^v\Big(-\frac{6}{25}a - \frac{2}{5}b + \frac{2}{75}c + \frac{2}{5}A_{00}\Big), \qquad u > v.$$

Sedaj lahko pogledamo, kako se naš model obnaša. Naj bosta m in n lihi števili. Povprečno število odštevanj, ki smo ga označili s spremenljivko C, dobimo tako, da v rekurzivne enačbe, ki smo jih nastavili na začetku, vstavimo $a = 1$, $b = 0$, $c = 1$ in $A_{00} = 1$, je torej

$$C = \frac{1}{2}u + \frac{2}{5}v + \frac{49}{50} - \frac{1}{5}\delta_{uv}.$$

Pri tem smo izpustili člene, ki gredo z rastjo v hitro proti 0. Povprečno število primerov, ko je $\gcd(m, n) = 1$, dobimo tako, da vstavimo $a = 0$, $b = 0$, $c = 0$ in $A_{00} = 1$. Torej je verjetnost, da sta si števili tuji približno $\frac{3}{5}$. Poglejmo si sedaj še kolikokrat povprečno gre algoritem čez diagonalni element $A_{u'u'}$. To dobimo tako, da postavimo $a = 1$, $b = 0$, $c = 0$ in $A_{00} = 0$. To se torej zgodi približno $\frac{1}{5}v + \frac{6}{25} + \frac{2}{5}\delta_{uv}$ krat. Sedaj lahko določimo spremenljivko D, to je število desnih pomikov. Ker smo rekli, da je r vedno ali 0 ali 1, mora veljati zveza

$$\text{število desnih pomikov} + \text{število sekanj diagonale} + 2\lfloor \log_2 \gcd(m, n) \rfloor = u + v.$$

Povprečna vrednost $\lfloor \log_2 \gcd(m, n) \rfloor$ je v našem modelu $\frac{4}{5}$, tako dobimo

$$D = u + v - \Big(\frac{1}{5}v + \frac{6}{25} + \frac{2}{5}\delta_{uv}\Big) - \frac{4}{5} = u + \frac{4}{5}v - \frac{46}{25} - \frac{2}{5}\delta_{uv}.$$

Knuth v svoji knjigi pravi, da je z empiričnimi poskusi za $29 \le u, v \le 37$ dobil približka

$$C \approx \frac{1}{2}u + 0,203v + 1,9 - 0,4(0,6)^{u-v},$$
$$D \approx u + 0,41v - 0,5 - 0,7(0,6)^{u-v}.$$

Tako se pokaže, da se napaka pri naših predpostavkah, ko smo postavili poenostavljen sistem pokaže pri koeficientih, ki nastopajo pri vrednosti v, kateri se izkažejo kot previsoki. Če pa bi računali namesto na intervalu $2^u \le m < 2^{u+1}$, $2^v \le n < 2^{v+1}$, na intervalu $1 \le m, n < 2^N$, bi dobili približka

$$C \approx 0,70N + \mathcal{O}(1), \qquad D \approx 1,41N + \mathcal{O}(1),$$

kar bi se lepo skladalo z rezultati, ki jih je dobil Knuth z empiričnimi poskusi. Vrednosti spremenljivk A, B in E dobimo precej enostavno. Tako je $A = \frac{1}{3}$, $B = \frac{1}{3}$ in $E = 0,35N - 0,4$. Skupna povprečna časovna zahtevnost binarne metode je torej

$$9A + 2B + 6C + 3D + E + 13 = 8,8N + 5.$$

8. Evklidov algoritem na velikih številih

Kadar imamo opravka z velikimi števili, imamo radi, da je čim več operaciji enostavnih. Tako nas pri Evklidovem algoritmu takoj zmoti deljenje, ki je na velikih številih precej zamudno. Seveda lahko ta problem poskušamo ovreči, češ da nam posledica 9. zagotavlja, da je v več kot $99,8\%$ primerih koeficient, ki ga bomo dobili manjši od 1000. Če pa že dobimo velik koeficient, to pomeni, da bosta števili, ki bosta nastopali v naslednjem koraku precej manjši. A kljub temu bomo tukaj poskusili pogledati, kako bi lahko računanje še pohitrili. Prva ideja bi bila, da uporabimo binarno metodo. Tukaj vemo, da imamo opravka le z odštevanjem in desnim zamikom. Ta ideja ni slaba. Vendar pa nas navede na drugo idejo, ki bila pri manjših številih več ali manj neuporabna. Če pogledamo malo bolj natančno, vidimo, da se binarna metoda ukvarja predvsem s tem, kakšni sta števili na zadnjih mestih. Naravno se je vprašati, ali bi si lahko kako pomagali s tem, da vemo kakšni sta števili na prvih nekaj mestih. Na to pa je dal odgovor D. H. Lehmer, ki je podal metodo za računanje le s prvimi nekaj ciframi velikih števil. Tako se znebimo precej operacij na velikih številih. Poglejmo si kako ta metoda deluje.

Primer. Recimo, da želimo izračunati največji skupni delitelj števil $u = 19822004$ in $v = 10000000$ na računalniku, ki zna delati osnovne operacije na največ štirimestnih številih. Naj bo $u' = 1982$, $v' = 1001$, $u'' = 1983$ in $v'' = 1000$. Potem velja

$$\frac{u'}{v'} < \frac{u}{v} < \frac{u''}{v''}.$$

Kvocient u/v predstavlja zaporedje kvocientov, ki jih dobimo v zaporednih korakih Evklidovega algoritma. Če izvajamo Evklidov algoritem vzporedno na parih (u', v') in (u'', v''), dokler ne dobimo dveh različnih kvocientov, je jasno, da bo enako zaporedje kvocientov pripadalo tudi paru (u, v). Poglejmo si, kako to deluje na našem primeru:

u'	v'	q'	u''	v''	q''
1982	1001	1	1983	1000	1
1001	981	1	1000	983	1
981	20	49	983	17	57

Prva dva kvocienta sta v obeh primerih enaka, torej morata biti prava. Tako vemo, da bi Evklidov algoritem na paru (u, v) potekal takole:

$$\begin{array}{ccc} u & v & q \\ u_0 & v_0 & 1 \\ v_0 & u_0 - v_0 & 1 \\ u_0 - v_0 & 2v_0 - u_0 & ? \end{array}$$

Za kvocient označen z ? pa vemo da leži nekje med 49 in 57. Torej se lahko znebimo deljenj na velikih številih v prvih dveh korakih tako, da jih zamenjamo z izračunom števil $u = u_0 - v_0$ in $v = 2v_0 - u_0$. V našem primeru dobimo $u = 9822004$ in $v = 177996$. Sedaj bi računanje nadaljevali z $u' = 982$, $v' = 18$, $u'' = 983$ in $v'' = 17$.

Izkaže se, da je število korakov Evklidovega algoritma, ki jih lahko naenkrat nadomestimo proporcionalno številu cifer števil na katerih lahko izvajamo enostavne operacije. V našem primeru, ko smo imeli štirimestna števila, bi povprečno pridobili 5 korakov, pri desetmestnih številih pa 12 korakov.

Algoritem 4. *Lehmerjeva metoda*

```
dokler je v prevelik,
  u1 = floor(u / b^k), kjer je k cim manjsi in je u1 < b^p;
  v1 = floor(v / b^k);
  a = 1;
  b = 0;
  c = 0;
  d = 1;
  q = floor((u1 + a) / (v1 + c));
  če je q enak floor((u1 + b) / (v1 + d)), potem
    t = a - q * c;
    a = c;
    c = t;
    t = b - q * d;
    b = d;
    d = t;
    t = u1 - q * v1;
    u1 = v1;
```

```
      v1 = t;
    sicer
      če je b = 0, potem
        t = u mod v;
        u = v;
        v = t;
      sicer
        t = a * u + b * v;
        w = c * u + d * v;
        u = t;
        v = w;

izračunaj gcd(u, v) z osnovnim algoritmom;
```

Pri izvajanju tega algoritma dobimo mimogrede tudi koeficiente v verižnem ulomku števila u/v.

Kljub temu, da je ta algoritem bolj kompliciran, kot binarna metoda, se izkaže za hitrejšega. Seveda pa lahko z idejo, ki smo jo imeli tukaj, izpopolnimo tudi binarno metodo. Tako postane binarna metoda zopet hitrejša. Prav tako ima zgornji algoritem tud to lastnost, da se ga da dokaj enostavno popraviti v razširjeni Evklidov algoritem.

9. Zaključek

Ves čas smo se ukvarjali z raznimi izvedbami Evklidovega algoritma na številih. Znano pa je, da lahko Evklidov algoritem uporabimo tudi za iskanje največjega skupnega delitelja dveh polinomov.

Polinome si lahko približno predstavljamo kot števila. Če imamo polinom s koeficienti iz $GF(n)$ in neznanko x nadomestimo z n, dobimo število. Tako bi v primeru $n = 2$ dobili polinome s koeficienti 0 ali 1, ki bi jih lahko predstavili kot števila v dvojiškem sistemu. Tako pa lahko takoj pomislimo, da bi na njih uporabili binarno metodo. Ali se kaj spremeni? Se! Pomik v desno pri številih pomeni deljenje z 2. Tako je jasno, da bo pri polinomih to pomenilo deljenje z x. Kaj pa odštevanje? Ali lahko odšejemo polinom x od polinoma x^2 in pri tem ohranimo zahtevo, da morajo biti koeficienti v $GF(2)$. Kot števili bi ta polinoma predstavljala števili 2 in 4. Njuna razlika bi torej bila 2, kar pa bi pomenilo polinom x. Jasno je, da v splošnem velja $x^2 - x \neq x$. Ker smo v tem primeru vzeli

polinome s koeficienti v $GF(2)$, lahko odštevanje nadomestimo s seštevanjem. Tudi tu pridemo v težave. Vendar lahko opazimo, da bi v obeh primerih dobili pravilna rezultata, če bi namesto odštevanja oz. seštevanja uporabili operacijo xor (strogi logični ali). Torej lahko takšne polinome lepo rešujemo z binarno metodo. Pri splošnem n pa ne moremo rešiti težav z odštevanjem na takšen način, torej se je potrebno polinomov lotiti drugače.

Literatura

[1] D. E. Knuth, *The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 2nd ed.*, Reading MA, Addison-Wesley, 1969

[2] D. E. Knuth, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms, 2nd ed.*, Reading MA, Addison-Wesley, 1981

[3] E. W. Weisstein, *Greatest Common Divisor*, MathWorld - A Wolfram Web Resource, http://mathworld.wolfram.com/GreatestCommonDivisor.html

[4] E. W. Weisstein, *Euclidean Algorithm*, MathWorld - A Wolfram Web Resource, http://mathworld.wolfram.com/EuclideanAlgorithm.html

[5] E. R. Berlekamp, *Algebraic Coding Theory*, Aegan Park Press, 1984