Faktorizacija velikih prirodnih brojeva

Laća, Marija

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Marija Laća

FAKTORIZACIJA VELIKIH PRIRODNIH BROJEVA

Diplomski rad

Voditelj rada:
prof. dr. sc. Andrej Dujella

Zagreb, srpanj 2019.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Hvala roditeljima na omogućenom bezbrižnom obrazovanju i bezuvjetnoj podršci u svemu. Hvala sestrama i najbližima, posebno Fabiju i Katarini.

Sadržaj

Uvod

U algoritamskoj teoriji brojeva važan je problem faktorizacije brojeva, tj. rastava složenog prirodnog broja u umnožak manjih prirodnih brojeva. Ako su ti cijeli brojevi dodatno ograničeni na proste brojeve, taj proces nazivamo rastavljanje na proste faktore. Prosti ili prim brojevi su jedan od fundamentalnih pojmova u matematici. To su brojevi koji su djeljivi samo s brojem 1 i sa samim sobom. Postoje deseci važnih primjena prostih brojeva, ali jedna od najvažnijih je svakako primjena u modernoj računalnoj sigurnosti. Naglim razvojem računarstva u 20. stoljeću javila se potreba za što bržim i efikasnijim algoritmima za faktorizaciju velikih brojeva. Algoritam je konačan slijed dobro definiranih naredbi za ostvarenje zadatka. U kriptografiji se uvijek slijedi temeljno pravilo: algoritam ne mora biti tajan, ali ključ mora biti. Generiranje prostih brojeva potrebno je za gotovo sve algoritme korištene za generiranje javnog ključa. Naprimjer, kod RSA ili Rabinovog kriptosustava trebamo pronaći proste brojeve p i q za izračun javnog ključa $N = p \cdot q$. Ideja korištenja velikih prostih brojeva u kriptografiji zasniva se na činjenici da nije poznat dovoljno brz algoritam koji bi u razumnom vremenu faktorizirao dovoljno velik broj.

U ovom radu proučit ćemo nekoliko testova prostosti i algoritama za faktorizaciju velikih brojeva. U prvom poglavlju komentirat ćemo učestalost prostih brojeva i što su to Mersennovi prosti brojevi. Za razliku od algoritama faktorizacije, testovi prostosti nam uglavnom ne daju faktore, već samo potvrdu je li broj složen ili prost. U drugom poglavlju obradit ćemo nekoliko najpoznatijih testova prostosti. Osnovni način za provjeru prostosti je pokusno dijeljenje, odnosno dijeljenje broja n sa svim brojevima do $\sqrt{n}$. Većina naprednih algoritama za testiranje prostosti koristi obrat malog Fermatovog teorema. Jedan od najpoznatijih testova prostosti je Fermatov test koji nas može uvjeriti u složenost broja, ali nikada ne može dokazati da je broj prost. Postoje složeni brojevi za koje će Fermatov test uvijek vratiti *vjerojatno prost*, a nazivaju se Carmichaelovi brojevi. Modificirana verzija Fermatovog testa, Miller-Rabinov test, izbjegava taj problem složenih brojeva. Glavna literatura za obradu testova prostosti u drugom poglavlju bila je [5].

Traženje faktora velikog složenog broja je zahtjevna računalna operacija. Kod složenosti algoritama razmatra se vremenska složenost, a *L-notacija* se koristi za mjerenje složenosti algoritama koji računaju faktore nekog broja N.

U trećem poglavlju obrađeno je 6 algoritama za faktorizaciju velikih prirodnih brojeva.

- **Pokusno dijeljenje** - osnovni algoritam za pronalaženje faktora nekog broja s kojim smo se susreli kada smo spominjali testiranje prostosti.
- **Fermatova razlika kvadrata** - zasniva se na jednostavnom triku kod algoritama faktoriziranja, odnosno pronalasku dva broja čiji su kvadrati kongruentni modulo N-u.
- **Pollardova rho metoda** - jedan od poznatijih algoritama, a razvio ga je John Pollard.
- **Metoda** $p-1$ - bazira se na malom Fermatovom teoremu.
- **CFRAC (*Metoda verižnog razlomka*)** - bazira se na činjenici da svaki realan broj možemo zapisati u obliku verižnog razlomka. On je jedan od bržih algoritama za faktorizaciju.
- **QS (*Metoda kvadratnog sita*)** - vjerojatno najbrža metoda za faktorizaciju cijelih brojeva koji imaju između 80 i 100 decimalnih znamenki.

Za obradu navedenih algoritama korišteno je 2. poglavlje iz [5] i 5. i 8. poglavlje iz [1]. Diplomski rad napravljen je u sklopu aktivnosti Projekta KK.01.1.1.01.0004 - Znanstveni centar izvrsnosti za kvantne i kompleksne sustave te reprezentacije Liejevih algebri.

Poglavlje 1

Prosti brojevi

Prirodni broj veći od 1 koji je djeljiv samo sa samim sobom i brojem 1 naziva se prost broj. Prvih nekoliko prostih brojeva su 2, 3, 5, 7, 11, 13, 17, 19, 23 i 29. Brojevi koji imaju više od dva faktora zovu se složeni brojevi. Broj 1 nije niti prost niti složen broj. Za brojeve a i b kažemo da su relativno prosti ako je najveći zajednički djelitelj brojeva a i b jednak 1. Za svaki prosti broj p postoji prosti broj p' takav da je $p' > p$. Ovaj matematički dokaz demonstrirao je grčki matematičar Euklid još u 3. st. pr. Kr.

Dokaz. Pretpostavimo da ima konačno mnogo prostih brojeva i da je najveći prosti broj p_n. Promatrajmo broj $N = p_1p_2p_3...p_n + 1$. Kako je N veći od svih prostih brojeva, on sam ne može biti prost. Iz toga slijedi da je N složen, što znači da ga dijeli jedan od prostih brojeva. Neki prost broj P dijeli N i broj $p_1p_2p_3...p_n$, budući da je on jedan od njegovih faktora. To znači da $P \mid p_1p_2p_3...p_n$ i $P \mid p_1p_2p_3...p_n + 1$, pa $p \mid 1$, što je kontradikcija. □

Iz dokaza slijedi da ne postoji najveći prosti broj, te da prostih brojeva ima beskonačno mnogo. Kako se skup prirodnih brojeva $\mathbb{N}$ nastavlja, prosti brojevi postaju manje učestali i teže ih je pronaći u razumnom vremenskom roku.

1.1 Prosti brojevi kroz povijest

Do prije 100-ak godina ljudi su računali proste brojeve i objavljivali tablice istih. Računali su proste brojeve i proste faktore brojeva od 1 do neke granice. Erastoten je prvi objavio takvu tablicu prije više od 2500 godina. Cataldi je 1603. godine objavio tablicu faktora za brojeve od 1 do 750. Chernac je 1811. godine objavio tablicu faktora za brojeve do 1 020 000. Godinu kasnije Burckhardt je objavio tablicu za sljedećih milijun brojeva, a za sljedećih milijun tri godine kasnije. Crelle je tablicu nadopunio za brojeve do pet milijuna, ali njegove tablice imale su previše grešaka da bi se objavile. Zadnja tablica faktora, ona

od D. N. Lehmera, objavljena je 1909. i sadržavala je faktore za brojeve do 10 017 000. Pet godina nakon objavio je tablicu svih prostih brojeva do 10 006 721.

1.2 Mersennovi prosti brojevi

Mersennovi brojevi su brojevi oblika $2^n - 1$ gdje je n prirodan broj. Podskup Mersennovih brojeva su Mersennovi prosti brojevi oblika $2^p - 1$ gdje je p prost broj. Naziv su dobili po francuskom matematičaru, teologu, filozofu i teoretičaru glazbe Marinu Mersennu. U svojoj knjizi Cogita Physico-Mathematica iznio je tvrdnju da je broj $2^p - 1$ prost za $p = 2, 3, 5, 7, 13, 17, 19, 31, 67, 127, 257$, a za sve druge prirodne brojeve manje od 257 složen. Kasnije provjere pokazale su da je Mersenne pogriješio i da su brojevi m_{67} i m_{257} složeni, a m_{61}, m_{89} i m_{107} prosti. Savršeni brojevi su oni koji su jednaki zbroju svih svojih djelitelja, a svi Mersennovi prosti brojevi povezani su sa savršenim brojevima oblika

$$2^{p-1}(2^p - 1).$$

Teorem 1.2.1. *Ako je $M(n)$ Mersennov prosti broj, onda je*

$$m = 2^{n-1} \cdot M(n)$$

savršen broj.

Lema 1.2.2. $x^k - 1 = (x - 1)(1 + x + x^2 + ... + x^{k-1})$

Dokaz. Dokaz ovog *teorema 1.2.1* zasniva se na *lemi 1.2.2.* Ako je $M(n)$ prost, onda su djelitelji broja

$$m = 2^{n-1} \cdot M(n)$$

$1, 2, 3, ..., 2^{n-1}, M(n), 2 \cdot M(n), 4 \cdot M(n), ..., 2^{n-2} \cdot M(n)$. Suma svih djelitelja od m je:

$$\begin{aligned} &(1 + 2 + 4 + ... + 2^{n-1}) + (1 + 2 + 3 + ... + 2^{n-2}) \cdot M(n) \\ &= (2^n - 1) + (2^{n-1} - 1) \cdot (2^n - 1) \\ &= 2^{n-1} \cdot (2^n - 1) = m. \end{aligned}$$

□

Teorem 1.2.3. *Ako je n složen, tada je i $M(n) = 2^n - 1$ složen.*

Dokaz. Neka je $n = a \cdot b$, a i b su veći od 1. Koristeći *lemu 1.2.2.* imamo

$$\begin{aligned} M(n) &= 2^{a \cdot b} - 1 \\ &= (2^a)^b - 1 \\ &= (2^a - 1) \cdot (1 + 2^a + 2^{2a} + ... + 2^{(b-1) \cdot a}). \end{aligned}$$

Kako je svaki od tih faktora veći od 1, $M(n)$ je složen broj. □

GIMPS ili *Great Internet Mersenne Prime Search* je projekt u kojem volonteri mogu besplatno koristiti program za traženje Mersennovih prostih brojeva. Osnovan je 1996. godine, a do sada je u sklopu projekta pronađeno 17 Mersennovih prostih brojeva, od čega je čak 15 najvećih poznatih prostih brojeva. U trenutku pisanja ovog rada najveći poznati prosti broj bio je $2^{82\,589\,933} - 1$. Taj broj ima preko 24 milijuna znamenaka, a pronašao ga je Patrick Laroche, član GIMPS-a, krajem 2018. godine.

$\mathbf{2^p - 1}$	**Broj znamenaka**
$2^2 - 1$	1
$2^3 - 1$	1
$2^5 - 1$	2
$2^7 - 1$	3
.	.
.	.
.	.
$2^{57\,885\,161} - 1$	17 425 170
$2^{74\,207\,281} - 1$	22 338 618
$2^{77\,232\,917} - 1$	23 249 425
$2^{82\,589\,933} - 1$	24 862 048

Poglavlje 2

Testovi prostosti

Testiranje broja za prostost može obaviti vrlo brzo koristeći vrlo jednostavan kod, ali s algoritmom koji ima mogućnost pogreške. Ponavljanjem tog algoritma možemo smanjiti vjerojatnost pogreške na bilo koju vrijednost koju trebamo. Neke od naprednijih tehnika za testiranje prostosti dat će nam potvrdu prostosti, a je li promatrani broj uistinu prost možemo provjeriti nekim trećim algoritmom. Provjeravanje ćemo izvršavati pomoću testova prostosti. To su kriteriji za koje vrijedi da ako ih n ne zadovolji, onda je n sigurno složen, dok ako ih zadovolji, n može biti prost, ali i ne mora. Što više testova n zadovolji, to je veća vjerojatnost da je n uistinu prost.

Napomena 2.0.1. *Najveća zajednička mjera je najveći cijeli broj s kojim su djeljivi elementi nekog skupa, a označavat ćemo je oznakom* $\mathrm{NZD}(a, b)$*, gdje su a i b elementi skupa prirodnih brojeva.*

Lema 2.0.2. (Euklidova lema)
Neka je p prost broj te neka su a i b cijeli brojevi takvi da $p \mid a \cdot b$*. Tada* $p \mid a$ *ili* $p \mid b$*.*

Dokaz. Neka p ne dijeli jednog od brojeva a, b. Možemo pretpostaviti da $p \nmid a$. Trebamo dokazati da tada $p \mid b$. Kako p ne dijeli cijeli broj a, a jedini djelitelji prostog broja p su 1 i p, slijedi da je $\mathrm{NZD}(a, p) = 1$. Znamo da postoje cijeli brojevi x, y takvi da je $a \cdot x + p \cdot y = 1$. Množenjem s b dobivamo $a \cdot b \cdot x + p \cdot b \cdot y = b$. Kako p dijeli $a \cdot b$, slijedi da p dijeli b, što je i trebalo dokazati. □

Teorem 2.0.3. (Osnovni teorem aritmetike)
Za svaki prirodni broj n >1 postoje prosti brojevi $p_1, p_2, ..., p_k$ *i prirodni brojevi* $\alpha_1, \alpha_2, ..., \alpha_k$ *tako da je*

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \cdot ... \cdot p_k^{\alpha_k}$$

Nadalje, takva faktorizacija broja n jedinstvena je do na poredak faktora p_i*.*

Dokaz. Neka je n prirodan broj veći od 1 te neka su $n = p_1 p_2 \cdots p_k$ i n = $q_1 q_2 \cdots q_t$ dva prikaza broja n u obliku produkta prostih brojeva. Tada vrijedi $p_1 p_2 \cdots p_k = q_1 q_2 \cdots q_t$, pa $p_1 \mid q_1 q_2 \cdots q_t$. Iz leme 2.0.2, odnosno njezinog poopćenja koje se lako dokaže matematičkom indukcijom, znamo da $p_1 \mid q_i$ za neki i. Kako su p_1 i q_i oba prosti, slijedi $p_1 = q_i$. Permutiranjem faktora $q_1, q_2, ..., q_t$ možemo uzeti da je $i = 1$ te nakon kraćenja dobivamo $p_2 \cdots p_k = q_2 \cdots q_t$. Sličnim zaključivanjem dobivamo $p_2 = q_2, p_3 = q_3, ..., p_k = q_k$, te $k = 1$. Time je teorem u potpunosti dokazan. □

Definicija 2.0.4. *Neka je* $\pi : [2, +\infty\rangle \to \mathbb{N}$. *Funkcija* $\pi(X)$ *vraća broj prostih brojeva* p *takvih da je* $p \leq X, X \in [2, +\infty\rangle$. *Odnosno,*

$$\pi(X) = card\{p \in P \mid p \leq X\}$$

Jedan od prvih velikih problema matematike 19. stoljeća bio je razumjeti ponašanje te funkcije, odnosno dobiti informaciju o njenom asimptotskom ponašanju.

Tablica 2.0.5. *Vrijednosti funkcije* $\pi(X)$

X	$\pi(X)$
10	4
100	25
1 000	168
10 000	1 229
100 000	9 592
1 000 000	78 498
10 000 000	664 579
100 000 000	5 761 455
1 000 000 000	50 847 534
10 000 000 000	455 052 511
100 000 000 000	4 118 054 813
...	

Teorem 2.0.6. (Teorem o prostim brojevima)
Neka je $X \in [2, +\infty\rangle$. *Funkciju* $\pi(X)$ *tada možemo aproksimirati:*

$$\pi(X) \approx \frac{X}{\ln X}$$

odnosno,

$$\lim_{x \to \infty} \frac{\pi(X)}{\frac{X}{\ln X}} = 1$$

Dokaz. Opsežan dokaz može se naći u [4]. □

To znači da su prosti brojevi poprilično "gusti". Naprimjer, prostih brojeva manjih od $2^{1\,024}$ ima oko $2^{1\,014}$. Teorem o prostim brojevima omogućava nam i da procijenimo vjerojatnost da je nasumično odabrani broj prost. Ukoliko je p nasumično odabran broj, onda vjerojatnost da je on prost iznosi otprilike

$$\frac{1}{\ln p}$$

Dakle, nasumično odabran p duljine 1 024 bita, bit će prost s vjerojatnošću

$$\approx \frac{1}{\ln p} \approx \frac{1}{709}$$

Odnosno, u prosjeku trebamo odabrati 354 neparna broja veličine $2^{1\,024}$ prije nego li nađemo neki koji je prost. Iz toga slijedi da je korisno generirati velike proste brojeve, dokle god možemo efikasno primijeniti test prostosti.

2.1 Pokusno dijeljenje

Najjednostavniji test za provjeru prostosti je obično dijeljenje. Testiramo sve brojeve između 2 i $\sqrt{p}$ i provjeravamo jesu li oni djelitelji broja p. Ako niti jedan broj nije djelitelj, slijedi da je p prost broj. Ukoliko neki broj dijeli p, tada smo našli jednog od djelitelja složenog broja p. Slijedi da pokusno dijeljenje ima prednost da ili utvrdi je li p uistinu prost, ili utvrđuje netrivijalan faktor broja p u slučaju da je p složen. Međutim, testiranje prostosti pomoću pokusnog dijeljenja je loša strategija. U najgorem slučaju, ako je p uistinu prost broj, algoritam zahtijeva $\sqrt{p}$ koraka do kraja izvršavanja, što je vrlo neisplativa, odnosno eksponencijalna složenost algoritma. Ako je p složen broj, algoritam daje potvrdu da je broj složen. Unatoč svojim manama, ova metoda je idealan izbor za jako malene brojeve.

2.2 Fermatov test

Definicija 2.2.1. *Ako za cijele brojeve a, b te prirodan broj m vrijedi da m dijeli razliku a − b onda to možemo zapisati u sljedećem obliku:*

$$a \equiv b \pmod{m}.$$

Ovaj zapis zovemo kongruencijom, broj m njezinim modulom, a zapis čitamo a je kongruentno s b modulo m. Naravno, zapis je ekvivalentan ovima: $m \mid (a - b)$ ili $a - b = mk$, gdje je k cijeli broj.

Većina naprednih algoritama za testiranje prostosti koristi obrat malog Fermatovog teorema.

Teorem 2.2.2. (Mali Fermatov teorem) *Neka je p prost i $a \in \mathbb{N}$ takav da $p \nmid a$. Tada vrijedi*

$$a^{p-1} \equiv 1 \pmod{p}.$$

Dokaz. Pretpostavimo da je a nije djeljiv s p. Promotrimo prvih $p-1$ višekratnika broja a:

$$a, 2a, 3a, ..., (p-2)a, (p-1)a.$$

Neka su $r, s \in \{1, 2, 3, ..., p-1\}$. Pretpostavimo da ra i sa imaju isti ostatak pri dijeljenju s p, pa vrijedi $r \equiv s \pmod{p}$. Višekratnici broja a su kongruentni s $1, 2, 3, ..., p-1$ u nekom redoslijedu. Pomnožimo li sve ove ostatke dobijemo:

$$a \cdot 2a \cdot 3a \cdot ... \cdot (p-1)a \equiv 1 \cdot 2 \cdot 3 \cdot ... \cdot (p-1) \pmod{p}$$

Odnosno:

$$a^{(p-1)} \cdot (p-1)! \equiv (p-1)! \pmod{p}$$

Podijelimo li obe strane s $(p-1)!$ dobit ćemo:

$$a^{p-1} \equiv 1 \pmod{p}$$

čime je teorem u potpunosti dokazan. □

Ukoliko p $\nmid$ a, onda su brojevi p i a relativno prosti, tj.$(p, a) = 1$. Prisjetimo se, tada vrijedi

$$a^{p-1} \equiv 1 \pmod{p} \Leftrightarrow a^p \equiv a \pmod{p}.$$

Također, ukoliko $p \mid a$, onda imamo $a^p \equiv a \equiv 0 \pmod{p}$. Zato tvrdnju malog Fermatovog teorema možemo i ovako izreći: Za svaki prost broj p i svaki prirodan broj a vrijedi

$$a^p \equiv a \pmod{p}.$$

Modularno potenciranje može se izvesti vrlo efikasno koristeći metodu *kvadriraj i množi*. Algoritam *kvadriraj i množi* koristimo za računanje modularnog potenciranja. Neka je $\sum_{i=0}^{t} k_i 2^i$ binarni zapis broja k, gdje je $k_i \in \{0, 1\}$. Tada je

$$a^k = \prod_{i=0}^{t} a^{k_i 2^i} = (a^{2^0})^{k_0} (a^{2^1})^{k_1} \cdots (a^{2^t})^{k_t}.$$

Algoritam 2.2.3. *Kvadriraj i množi*

Ulaz: $a \in \mathbb{Z}_n = \{0, 1, ..., n-1\}$, cijeli broj $0 \leq k \leq n$ u binarnom zapisu $k = \sum_{i=0}^{t} k_i 2^i$

$b \leftarrow 1$
ako $k = 0$ **vrati** b
inače $A \leftarrow a$
ako $k_0 = 1$ **onda** $b \leftarrow a$
za $i = 1$ do t
    $A \leftarrow A^2 \pmod{n}$
    ako $k_i = 1$ **onda** $b \leftarrow Ab \pmod{n}$
vrati b
Izlaz: $b \equiv a^k \pmod{n}$

Činjenica da je računanje $a^{n-1} \mod n$ jako brza operacija daje nam vrlo brzi test prostosti kojeg zovemo Fermatov test za bazu a. Fermatov test nas može uvjeriti u složenost broja, ali nikada ne može dokazati da je broj prost.

Algoritam 2.2.4. *Fermatov test prostosti*

za $i = 0$ **do** $i = k - 1$ **izvedi** {
    odaberi $a \in \{2, ..., n-1\}$
    $b \leftarrow a^{n-1} \mod n$
    ako $b \neq 1$ **onda vrati** (složen, a)
    }
vrati vjerojatno prost

Pogledajmo na primjeru $n = 11 \cdot 31 = 341$ i uzmimo bazu $a = 2$. Imamo:

$$a^{n-1} \equiv 2^{340} \equiv 1 \pmod{341}$$

ali n očito nije prost broj. U tom slučaju kažemo da je n Fermatov pseudoprost broj u bazi 2. Postoji beskonačno mnogo pseudoprostih brojeva u bilo kojoj bazi. Ako algoritam vrati *(složen, a)*, tada znamo da je n sigurno složen broj, a a je svjedok njegove složenosti. Ako algoritam vrati *vjerojatno prost*, onda znamo da je n ili prost ili tzv. vjerojatno prost broj. Uzmimo za primjer broj

$$n = 43\,040\,357$$

gdje je n složen broj, s jednim svjedokom u bazi 2 budući da vrijedi

$$2^{n-1} \equiv 9\,888\,212 \pmod{n}.$$

Kao još jedan primjer uzmimo

$$n = 2^{192} - 2^{64} - 1,$$

za kojeg algoritam vraća *vjerojatno prost* budući da mu ne može naći svjedoka sa složenost. Zapravo je n prost, pa to i nije iznenađujuće. Ipak, postoje složeni brojevi za koje će Fermatov test prostosti uvijek vratiti *vjerojatno prost* za svaki a relativno prost s n. Ti brojevi nazivaju se Carmichaelovi brojevi, a kako bi stvar bila još gora, ima ih beskonačno mnogo. Prva tri su 561, 1105 i 1729. Trenutno najveći otkriveni Carmichaelov broj ima čak 60 351 znamenku. Charmichaelovi brojevi su uvijek neparni, uvijek imaju barem tri prosta faktora i nisu djeljivi s niti jednim kvadratom prirodnog broja većeg od 1. Također vrijedi da ako p dijeli Carmichaelov broj N, tada $p - 1$ dijeli $N - 1$. Iako ne možemo zaključiti da je neki broj prost ako prođe Fermatov test za neku bazu, provodeći ga možemo smanjiti vjerojatnost da broj bude lažno prost. Ponovno promatrajući broj 341 iz primjera primjetit ćemo da je za bazu 2 on pseudoprost, dok za bazu 3 nije.

$$3^{340} \equiv 56 \pmod{341}.$$

Ako promatramo skup prirodnih brojeva manjih od 10^{16} uočit ćemo da u tom skupu ima oko $2.7 \cdot 10^{14}$ prostih brojeva i čak $246\,683 \approx 2.4 \cdot 10^5$ Carmichaelovih brojeva. Iz toga slijedi da su Carmichaelovi brojevi rijetki, ali ne toliko da bi bili potpuno zanemareni. Zato se koriste i drugi kriteriji, koji ne dopuštaju takve propuste, kao npr. Eulerov kriterij ili jaki test prostosti.

Teorem 2.2.5. (Eulerov test) *Ako je N neparan prost broj i a relativno prost s N, onda vrijedi*

$$a^{(N-1)/2} \equiv \pm 1 \pmod{N}.$$

Teorem 2.2.6. (Jaki test prostosti) *Ako je N neparan prost broj i $N - 1 = 2^s t$, t neparan, te ako a nije djeljiv s N, onda vrijedi (samo) jedno od sljedećeg:*

$$\begin{cases} a^t \equiv 1 \pmod{N}, \\ a^{2^i t} \equiv -1 \pmod{N} \quad \text{za neki } i, 0 \leq i \leq s-1. \end{cases}$$

2.3 Miller-Rabinov test

Zbog postojanja Carmichaelovih brojeva obično se Fermatov test izbjegava. Ipak, postoji modificirana verzija Fermatovog testa, odnosno Miller-Rabinov test, koji izbjegava problem složenih brojeva za koje ne postoji svjedok za prostost. To ne znači da je lako naći svjedoka za svaki složeni broj, već da svjedok mora postojati. Dodatno, Miller-Rabinov test ima vjerojatnost $1/4$ da će utvrditi da je složeni broj prost, za svaku nasumično odabranu bazu a, pa ponavljanjem testa dolazimo do značajnog smanjenja vjerojatnosti pogreške.Drugim riječima, ovaj test se bazira na uzastopnom slučajnom odabiru baze za testiranje prostosti.

Algoritam 2.3.1. *Miller-Rabinov test*

Zapiši $n - 1 = 2^s \cdot m$, takav da je m neparan
za $j = 0$ **do** $j = k - 1$ **izvedi** {
odaberi $a \in \{2, ..., n - 2\}$
$b \leftarrow a^m \mod n$
ako $b \neq 1$ i $b \neq (n - 1)$ **onda vrati** (složen, a)
}
vrati vjerojatno prost

Ako je n složen broj, onda je vrijednost a u algoritmu Miller-Rabinov svjedok složenosti broja n. Poznato je da Generalizirana Riemannova hipoteza (GRH) povlači da uvijek postoji Miller-Rabinov svjedok a za složenost od n tako da vrijedi:

$$a \leq O((\log n)^2).$$

2.4 Dokazi prostosti

Do sada smo samo pronalazili svjedoke za složenost broja i takve brojeve smo mogli promatrati kao dokaz složenosti. Uz to, dobili smo brojeve za koje smo pretpostavljali da su vjerojatno prosti, ali ne i brojeve koji su *sigurno* prosti. Vjerojatnost da je *vjerojatno prost* broj složen je prema Miller-Rabinovom testu na 20 baza oko 2^{-40} što je u praksi prihvatljivo, no teoretski gledano to bi mogao biti problem. Drugim riječima, želimo pronaći brojeve koji su sigurno prosti, a ne *vjerojatno prosti*. Postoje algoritmi čiji rezultati daju svjedoke za prostost nekog broja. Takvi svjedoci zovu se dokazi prostosti. U praksi su takvi programi korišteni samo kada smo poprilično sigurni da je broj kojeg testiramo uistinu prost. Drugim riječima, taj broj je već prošao Miller-Rabinov test za dovoljan broj baza i sada nam je preostalo pronaći broj koji je dokaz prostosti. Najuspješniji algoritam za dokazivanje prostosti je ECCP (*Elliptic Curve Primality Prover*). ECCP algoritam ne garantira uvijek rezultat, odnosno pronalazak svjedoka, pa čak i kada je unos uistinu prost broj. Ako je ulaz složeni broj, nije sigurno da će se algoritam uopće zaustaviti u nekom trenutku. Kako ECCP radi sa polinomnom vremenskom složenošću, prilično je učinkovit. Dokazi o prostosti koje on proizvodi mogu se deterministički provjeriti još i brže.

2.5 AKS algoritam

Dugo godina postavljalo se pitanje je li moguće stvoriti algoritam za testiranje prostosti s determinističkim polinomnim vremenom izvršavanja za sve ulaze, bez ikakvih pretpostavki o istima. Pozitivan odgovor na to pitanje dali su Agrawal, Kayal i Saxena 2002. godine.

Oni su razvili AKS test prostosti koristeći poopćeni Fermatov test. Pitamo se jesu li dva polinoma stupnja n jednaka. Korištenje ovog osnovnog teorema, čiji je dokaz relativno jednostavan, bio je veliki napredak. Algoritam ćemo prikazati u nastavku. U algoritmu koristimo notaciju $F(X) \pmod{G(X), n}$ za označavanje uzimanja redukcije od $F(X)$ modulo $G(X)$ i modulo n.

Teorem 2.5.1. *Cijeli broj $n \geq 2$ je prost ako i samo ako kongruencija*

$$(X - a)^n \equiv (X^n - a) \pmod{n}$$

vrijedi za neki cijeli broj a relativno prost sa n.

Definicija 2.5.2. *Neka je $x \in \mathbb{R}$. Tada broj $\lfloor x \rfloor = \max\{m \in \mathbb{Z} : m \leq x\}$ zovemo* ***najveće cijelo od x*** *ili* ***cijeli dio realnog broja x.***

Algoritam 2.5.3. *AKS algoritam*

ako $n = a^b$ za neke cijele brojeve a i b **onda vrati** "složen"
Nađi najmanji r takav da je red od n modulo r veći od $(\log n)^2$.
ako $\exists a \leq r$ takav da $1 < \text{NZD}(a, n) < n$ **onda vrati** "složen"
ako $n \leq r$ **onda vrati** "složen"
za $a = 1$ **do** $\lfloor \sqrt{\phi(r)} \cdot \log n \rfloor$ **izvedi** {
ako $(X + a)^n \neq X^n + a \pmod{X^r - 1, n}$ **onda vrati** "složen"
}
vrati prost

Poglavlje 3

Metode faktorizacije

Traženje faktora velikog složenog broja je zahtjevna računalna operacija. Kod složenosti algoritama razmatra se vremenska složenost.

O-notacija se koristi za opisivanje ponašanja funkcije kada argument teži određenoj vrijednosti ili beskonačnosti. U računalnoj znanosti, veliko O oznaka se koristi za klasificiranje algoritama prema tome kako se njihovo vrijeme ili prostorni zahtjevi povećavaju porastom ulazne veličine. Oznaka $f(n) = O(g(n))$ znači da postoji konstanta C takva da je $| f(n) | \leq Cg(n)$, za sve dovoljno velike n.

o-notacija se koristi kao gornja granica složenosti algoritma. Oznaka $f(n) = o(g(n))$ znači da je $\lim_{n\to\infty} f(n)/g(n) = 0$.

L-notacija je asimptotski zapis analogan O-notaciji kao $L_n[\alpha, \beta]$ za vrijednost n koja teži u beskonačnost. Ova notacija obično se koristi za mjerenje složenosti algoritama koji računaju faktore broja N.

$$L_N(\alpha, \beta) = \exp((\beta + o(1)(\log N)^{\alpha}(\log \log N)^{1-\alpha})$$

gdje je
$L_N(0, \beta) = (\log N)^{\beta+o(1)}$ polinomno vrijeme,
$L_N(1, \beta) = N^{\beta+o(1)}$ eksponencijalno vrijeme.

U nekom smislu, funkcija $L_N(\alpha, \beta)$ interpolira između polinomnog i eksponencijalnog vremena izvršavanja. Ne zaboravimo da je množenje, inverzni algoritam od faktorizacije, jako jednostavna operacija koja zahtijeva manje vremena od $O(L_N(0, 2))$.

Postoje razne metode za faktorizaciju brojeva oblika $N = p \cdot q$. Neke od poznatijih su:

- **Pokusno dijeljenje** ili *Trial Division* - dijelimo N sa svim prostim brojevima do $\sqrt{N}$ i provjeravamo jesu li oni faktori broja N. Složenost je $L_N(1, 1)$ i iz toga slijedi da je ovo eksponencijalno složen algoritam.

- **Metoda s eliptičkom krivuljom** ili *Elliptic Curve Method* - jako dobra metoda ukoliko je faktor $p < 2^{50}$, a složenost ovog algoritma je $L_p(1/2, c)$, za neku konstantu c. Ovo je trenutno treći najbrži algoritam za faktorizaciju velikih prirodnih brojeva. Više o ovoj metodi u [2].
- **Kvadratno sito** ili *Quadratic Sieve* - ovo je vjerojatno najbrža metoda za faktorizaciju cijelih brojeva koji imaju između 80 i 100 decimalnih znamenki. Složenost je $L_N(1/2, 1)$.
- **Sito polja brojeva** ili *Number Field Sieve* - ovo je trenutno najuspješnija metoda za brojeve koji imaju više od 100 decimalnih znamenki. Pomoću ove metode faktorizirani su brojevi veličine $100^{155} \approx 2^{512}$ i ima složenost $L_N(1/3, 1.923)$ Više o ovoj metodi u [5].

Metode faktorizacije obično su podjeljene u dva dijela: metode "Mračnog doba" i moderne metode. U nastavku ćemo se upoznati s nekoliko metoda iz oba dijela.

3.1 Pokusno dijeljenje

Pokusno dijeljenje ili *Trial Division* je osnovni algoritam za pronalaženje faktora nekog broja s kojim smo se već susreli kada smo spominjali testiranje prostosti. Pretpostavimo da je N broj kojeg želimo faktorizirati. Lako zaključimo da je složenost u najgorem slučaju, odnosno, kada N zaista je prost, iznosi

$$O(\sqrt{N}) = O(2^{(\log_2 N)/2})$$

Složenost je potencija broja N, pa slijedi da je složenost algoritma eksponencijalna.

Algoritam 3.1.1. *Faktoriziranje pomoću pokusnog dijeljenja*

Ulaz: Broj N kojeg želimo faktorizirati
$a \leftarrow []$ *#inicijaliziramo praznu listu a*
$f \leftarrow 2$ *#prvi mogući faktor*
dok $n > 1$ i $f < \sqrt{N}$ **radi:**
    ako $N\%f == 0$: *#ako je ostatak pri dijeljenju 0*
        dodaj f u listu a
        podijeli N sa f
    inače f povećaj za 1
vrati a
Izlaz: Lista netrivijalnih faktora prirodnog broja N

Iako zbog složenosti možda i nije najsretniji izbor, pokusno dijeljenje ne treba potpuno zanemariti. Ova metoda je najčešći odabir za faktorizaciju brojeva manjih od 10^{12}. Algoritam ne mora provjeravati složene brojeve pri izvršavanju, a jedan način za preskakanje nekih složenih brojeva je provjeravanje svakog drugog, zatim četvrtog broja, i tako do $\sqrt{N}$. Zbog ovog trika algoritam će preskakati višekratnike brojeva 2 i 3. Ova tehnika zove se *kotač* i može se proširiti na preskakanje višekratnika od 2, 3 i 5 dodavanjem razlika između uzastopnih klasa ostataka relativno prostih do 30.

Primjer 3.1.2. Klase ostataka relativno proste s 30 su:

$$1, 7, 11, 13, 17, 19, 23, 29.$$

Nakon pokusnog dijeljenja s 2, 3 i 5, treba dodati uzastopno $7-5=2, 11-7=4, 13-11=2, 17-13=4, 19-17=2, 23-19=4, 29-23=6, 31-29=2, 37-31=6, 2, 4, ...$, do prethodnog djelitelja da bi se formirao sljedeći. Ovaj kotač ima $\phi(30)=8$ "zubaca". Kotači koji preskaču višekratnike prvih osam ili deset prostih brojeva korišteni su na računalima kako bi ubrzali pokusno dijeljenje. Alternativno, može se računati tablica prostih brojeva i podijeliti broj kandidat samo s tim prostim brojevima. Možemo koristiti kvadratne ostatke kako bismo ubrzali pokusnu podjelu preskakanjem nekih osnovnih brojeva koji ne mogu biti djelitelji broja N.

3.2 Fermatova razlika kvadrata

Osnovni trik kod algoritama faktoriziranja, poznat već stoljećima, je da nađemo dva broja čiji su kvadrati kongruentni modulo N-u, tako da vrijedi

$$x^2 \equiv y^2 \pmod{N},$$

jer tada imamo da je

$$x^2 - y^2 \equiv (x-y)\cdot(x+y) \equiv 0 \pmod{N}.$$

Ako je $N = p\cdot q$, onda imamo četiri moguća slučaja:

1. p dijeli $x-y$ i q dijeli $x+y$
2. p dijeli $x+y$ i q dijeli $x-y$
3. i p i q dijele $x-y$, ali ni p ni q ne dijele $x+y$

4. i p i q dijele $x + y$, ali ni p ni q ne dijele $x - y$

Svi ovi slučajevi imaju jednaku vjerojatnost $1/4$. Ako izračunamo

$$d = \text{NZD}(x - y, N),$$

tada prva četiri slučaja dijelimo su slučajeve:

1. $d = p$
2. $d = q$
3. $d = N$
4. $d = 1$.

Kako se svi ovi slučajevi pojavljuju s jednakom vjerojatnošću, vidimo da ćemo pronaći netrivijalne faktore od N s vjerojatnošću $1/2$. Jedini problem je kako naći brojeve x i y takve da vrijedi $x^2 \equiv y^2 \pmod{N}$?

Primjer 3.2.1. *Neka je* $N = 200\,819$.

Imamo $\lfloor \sqrt{200\,819} \rfloor + 1 = 449$. Sada je $449^2 - 200\,819 = 782$, a to nije kvadrat prirodnog broja. Pokušamo li sa $x = 450$ imamo $450^2 - 200\,819 = 1681 = 41^2$, pa je

$$200\,819 = 450^2 - 41^2 = (450 + 41)(450 - 41) = 491 \cdot 409.$$

3.3 Pollardova rho metoda

Najpoznatije ime na području faktorizacije u kasnom 20. stoljeću bio je John Pollard. Gotovo svi važni napretci u faktorizaciji nastali su njegovom zaslugom, a razvio je nekoliko poznatih algoritam kao: metoda $p - 1$, Pollardova Rho metoda i Sito polja brojeva.

Neka je N složen broj kojeg želimo faktorizirati i neka je p nepoznati prosti faktor od N. Neka je $f(x)$ ireducibilan polinom nad $\mathbb{Z}$. U praksi se najčešće koristi $x^2 + 1$ ili neki sličan polinom. Počevši s x_0, kreiramo niz pomoću rekurzivne definicije člana x_i:

$$x_i \equiv f(x_{i-1}) \mod N$$

Primjer 3.3.1. *Ako je* $x_0 = 2$, $f(x) = x^2 + 1$ *i* $N = 1133$, *naš niz izgleda ovako:*

$$\begin{aligned} x_0 &= 2 \\ x_1 &= 5 \\ x_2 &= 26 \\ x_3 &= 677 \\ x_4 &= 598 \\ x_5 &= 710 \\ x_6 &= 1\,049 \\ &\dots \end{aligned}$$

Neka je

$$y_i \equiv x_i \mod p.$$

Ako odaberemo $p = 11$, onda je niz y_i-ova:

$$\begin{aligned} y_0 &= 2 \\ y_1 &= 5 \\ y_2 &= 4 \\ y_3 &= 6 \\ y_4 &= 4 \\ y_5 &= 6 \\ y_6 &= 4 \\ &\dots \end{aligned}$$

Kako je $x_i \equiv f(x_{i-1}) \pmod N$, y_i je kongruentan $f(y_{i-1}) \mod p$. Imamo konačno mnogo klasa kongruencije, pa ćemo sigurno doći do:

$$y_i = y_j,$$

za neki par (i, j). Kada se to dogodi, nastavit ćemo "kružiti" pa za sve pozitivne t dolazimo do jednakosti

$$y_{i+t} = y_{j+t}.$$

Naš niz y_i-ova izgleda kao krug s repom, drugim riječima, podsjeća na grčko slovo *rho*, ρ, po čemu je algoritam i dobio ime.
Ako je y_i jednak y_j, onda

$$x_i \equiv x_j \pmod p,$$

pa p dijeli $x_i - x_j$. Šanse da x_i i x_j nisu jednaki su jako velike, pa je u tom slučaju

$$\mathrm{NZD}(N, x_i - x_j)$$

netrivijalni djelitelj broja N.

Primjer 3.3.2. *Pollardovom Rho metodom faktorizirat ćemo* $N = 25\,279$ *s vrijednošću* $x_0 = 1$ *i funkcijom* $f(x) = x^2 + 1$.

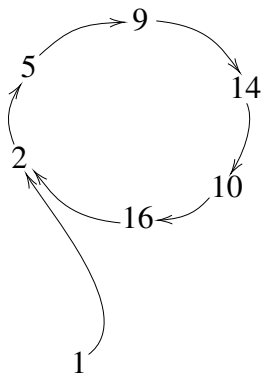
Koristeći jednakost $x_i = f(x_{i-1}) \mod N$ dobijemo prvih 13 članova x_i.

$$\begin{aligned} x_0 &= 1 \\ x_1 &= 2 \\ x_2 &= 5 \\ x_3 &= 26 \\ x_4 &= 677 \\ x_5 &= 3\,308 \\ x_6 &= 22\,337 \\ x_7 &= 9\,947 \\ &\ldots \end{aligned}$$

NZD($x_{12} - x_6, N$) = NZD(20331-22337, 25279) = 17. Slijedi da je $p = 17$ pa imamo y_i-ove zadane formulom $y_i = x_i \mod p$:

$$\begin{aligned} y_0 &= 1 \\ y_1 &= 2 \\ y_2 &= 5 \\ y_3 &= 9 \\ y_4 &= 14 \\ y_5 &= 10 \\ y_6 &= 16 \\ y_7 &= 2 \\ &\ldots \end{aligned}$$

... što možemo prikazati i dijagramom.



3.4 Metoda $p-1$

Metoda $p-1$ bazirana je na Malom Fermatovom teoremu koji tvrdi da za prosti broj p, koji ne dijeli a, vrijedi $a^{p-1} \equiv 1 \pmod{p}$. Iz toga slijedi $a^L \equiv 1 \pmod{p}$ za bilo koji višekratnik L od $p-1$. Ako uz to $p \mid N$, onda p dijeli NZD$(a^L - 1, N)$. Naravno, ne možemo izračunati $a^L \mod p$ zato što je p nepoznat prost faktor od N, ali možda možemo izračunati $a^L \mod N$. Pollardova ideja je da odjednom isprobava mnogo potencijalnih prostih faktora p od N .

Definicija 3.4.1. *Prirodan broj n je y-gladak ako su svi njegovi prosti brojevi $\leq y$. De Bruijnova funkcija, $\psi(x, y)$, nam daje broj y-glatkih brojeva između 1 i x.*

Ako je $p-1$ *B-gladak*, odnosno, ako je najveći prosti faktor od $p-1$ manji ili jednak B, tada će $p-1$ dijeliti L ako je L umnožak svih prostih brojeva $\leq B$, svaki od njih ponovljen dovoljno puta. Ako prosti broj $q \leq B$ dijeli $p-1$, tada q ne može dijeliti $p-1$ više od $\log_y p - 1 = (\log p / \log q) - 1$ puta. Ovaj broj je gornja granica za *dovoljan broj puta* da q dijeli L. Ipak, veliki prosti brojevi rijetko dijele veliki nasumično odabran broj nekoliko puta. Kako bi olakšali posao, odabrat ćemo granicu B i definirati L kao najmanjeg zajedničkog djelitelja prirodnih brojeva do B. Može se pokazati da vrijedi $L = \prod q^e$, gdje q ide kroz sve proste brojeve $\leq B$, i za svaki q, q^e je najveća potencija od q koja je $\leq B$. Najčešće je B jako velik, pa je L zaista ogroman, pa nema smisla da ga računamo. Kada god se q^e formira, računa se i $a = a^{q^e} \mod N$. Ovo je prva faza algoritma:

Algoritam 3.4.2. *Prva faza $p-1$ algoritma:*
Pronađi sve proste brojeve $p_1 = 2, p_2 = 3, p_3, ..., p_k \leq B$

Ulaz: Pozitivan složeni broj N kojeg ćemo faktorizirati i granica B

$a \leftarrow 2$
za $i = 1$ **do** k **izvedi** {
$e \leftarrow \lfloor (\log B)/\log p_i \rfloor$
$f \leftarrow p_i^e$
$a \leftarrow a^f \mod N$
}
$g \leftarrow$ NZD$(a-1, N)$
ako $(1 < g < N)$ **ispiši** "g dijeli N"
inače odustani

Izlaz: Pravi faktor g od N, inače odustani

Prosti brojevi se lako mogu generirati pomoću Erastotenovog sita. Eksponenciranje se računa pomoću algoritma za brzo eksponenciranje. Računanje najvećeg zajedničkog djelitelja bi se trebalo odvijati svakih nekoliko tisuća iteracija u **for** petlji, umjesto samo jednom na kraju algoritma. **For** petlja se nastavlja ako $g = 1$. Ako je $g = 1$ na kraju, možemo ili odustati ili ići na drugu fazu algoritma. Ako je $g = N$, tada su otkriveni svi prosti djelitelji p od N. Ali čak niti ovaj algoritam neće funkcionirati u slučaju da $p - 1$ ima istog najvećeg prostog djelitelja q za svakog prostog faktora p od N.

Primjer 3.4.3. Pollardova $p - 1$ metoda ne uspijeva faktorizirati $1247 = 29 \cdot 43$ zato što $29 - 1 = 2^2 \cdot 7$ i $43 - 1 = 2 \cdot 3 \cdot 7$. Oni imaju isti najveći prosti faktor 7, pa će i 29 i 43 biti otkriveni tijekom iste iteracije **for** petlje. To jest, a će postati $1 \mod N$ kada $i = 4$.

Primjer 3.4.4. Baillie je 1986. godine otkrio da je 16-znamenkasti prosti djelitelj $p = 1\,256\,132\,134\,125\,569$ Fermatovog broja $F_{12} = 2^{4096} + 1$ koristeći Pollardovu $p - 1$ metodu i granicu $B = 30\,000\,000$. Baillie je bio uspješan zato što je najveći prosti faktor od $p - 1$ manji od B:

$$p - 1 = 2^{14} \cdot 7^2 \cdot 53 \cdot 29\,521\,841.$$

Algoritam ima i drugu fazu u kojoj biramo drugu granicu $B_2 > B$ i tražimo faktor p od N, za kojeg vrijedi da najveći faktor od $p - 1$ iznosi $\leq B_2$, a za drugi najveći faktor od $p - 1$ vrijedi $\leq B$. Na kraju prve faze prikazane u Algoritmu 3.1.4. a ima vrijednost $2^L \pmod N$. Neka su $q_1 < q_2 < ... < q_t$ prosti brojevi između B i B_2. Ideja je da uspješno izračunamo $2^{L_{q_i}} \pmod N$ i onda $\text{NZD}(2^{L_{q_i}} - 1, N)$ za $1 \leq i \leq k$. Prva potencija $2^{L_{q_i}} \pmod N$ se računa izravno kao $a^{q_1} \pmod N$. Razlika između $q_{i+1} - q_i$ su parni brojevi mnogo manji od q_i. Unaprijed se izračuna $2^{Ld} \pmod N$ za $d = 2, 4, ...$ do nekoliko stotina. Za naći $2^{L_{q_{i+1}}} \pmod N$ od $2^{L_{q_i}} \pmod N$ moramo pomnožiti s $2^{Ld} \pmod N$, gdje je $d = q_{i+1} - q_i$.

Primjer 3.4.5. Brent je 1984. godine pronašao 31-znamenkasti prosti faktor $p = 49\,858\,990\,580\,788\,843\,054\,012\,690\,078\,841$ od $N = 2^{977} - 1$.

Kako vrijedi:

$$p - 1 = 2^3 \cdot 5 \cdot 13 \cdot 19 \cdot 977 \cdot 1\,231 \cdot 4\,643 \cdot 74\,941 \cdot 1\,045\,397 \cdot 11\,535\,449$$

morao je koristiti $B \geq 1\,045\,397$ i $B_2 \geq 11\,535\,449$.

Kada Pollardova $p - 1$ metoda vrati faktor, testira se je li on uistinu prost. Kada se algoritam tek razvijao i počeo koristiti, ta provjera se nije uvijek izvršavala. Nekada između 1978. i 1981. godine, "prosti" faktor $1\,223\,165\,341\,640\,099\,735\,851$ od broja $6^{175} - 1$ uvršten je u Cunninghamovu tablicu prostih brojeva. Kasnije, 1986. godine, Atkin je uspješno faktorizirao taj broj kao produkt dva prosta 11-znamenkasta prosta broja.

3.5 CFRAC

Izraz oblika

$$x = q_0 + \cfrac{1}{q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + ...}}}$$

naziva se *verižni razlomak*, a možemo ga zapisati i u obliku $x = [q_0; q_1, q_2, q_3, ...]$. Vrijednosti q_i su cijeli brojevi za neki i, i pozitivni kad je $i > 0$. Jednostavan verižni razlomak može biti i konačan:

$$x = \frac{m_i}{n_i} = q_0 + \cfrac{1}{q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + ... + \cfrac{1}{q_k}}}}$$

i tada se označava kao $x = [q_0; q_1, q_2, q_3, ..., q_k]$. Brojevi q_i se nazivaju parcijalni kvocijenti u oba slučaja.
Vrijedi:

$$\begin{array}{ll} m_o = q_0 & n_0 = 1 \\ m_1 = q_0 q_1 + 1 & n_1 = q_1 \\ m_k = q_k m_{k-1} + m_{k-2} & n_k = q_k n_{k-1} + n_{k-2} \end{array}$$

Svaki realan broj x može se proširiti do verižnog razlomka koristeći sljedeći algoritam:

Algoritam 3.5.1. *Verižni razlomak*

Ulaz: realni broj x

$i \leftarrow 0$
$q_0 \leftarrow \lfloor x \rfloor$
$x \leftarrow x - q_0$
dok $x > 0$ **izvodi** {
$i \leftarrow i + 1$
$q_i \leftarrow \lfloor 1/x \rfloor$
$x \leftarrow x - q_i$
}

Izlaz: $x = [q_0; q_1, q_2, q_3, ...]$ je verižni razlomak broja x

Ako je broj kojeg želimo prikazati u obliku verižnog razlomka korijen prirodnog broja k koji nije potpun kvadrat, tada je njegov razvoj u verižni razlomak periodičan. Odnosno:

$$\sqrt{k} = [q_0; \overline{q_1, q_2, ..., q_{r-1}, 2q_0}]$$

i vrijedi $q_i = q_{r-i}$ za $i = 1, 2, ..., r-1$. Npr. $\sqrt{7} = [2; \overline{1, 1, 4}]$. U tom slučaju, razvoj u verižni razlomak može se dobiti primjenom sljedeće metode:

$$\begin{aligned} s_o &= 0 & t_0 &= 1 \\ s_{i+1} &= q_i t_i - s_i & t_{i+1} &= \frac{k - s_{i+1}^2}{t_i} \\ q_i &= \lfloor \frac{s_i + \sqrt{k}}{t_i} \rfloor & i &\geq 0 \end{aligned}$$

Uz navedene oznake vrijedi:

$$m_i^2 - k n_i^2 = (-1)^{i+1} t_{i+1}, i \geq 0, \tag{1.1}$$

$$s_i < \sqrt{k}, t_i < 2\sqrt{k}.$$

Uvedimo oznaku

$$t_i^* = (-1)^i t_i.$$

Pretpostavimo da smo našli produkt $t^*_{l_1+1} \cdot t^*_{l_2+1} \cdot t^*_{l_3+1} \cdots t^*_{l_j+1}$ koji je potpun kvadrat npr. jednak z^2. Tada iz (1.1) slijedi

$$m_{l_1}^2 m_{l_2}^2 \cdots m_{l_j}^2 \equiv z^2 \pmod{k}. \tag{1.2}$$

U jednakosti (1.2) svaki od m_{l_i} možemo zamijeniti s $m_{l_i} \pmod{k}$ ili s njegovim najmanjim ostatkom po apsolutnoj vrijednosti. Promotrimo sada brojeve NZD$(m_{l_1} \cdots m_{l_j} - z, k)$ i NZD$(m_{l_1} \cdots m_{l_j} + z, k)$. To su faktori od k. Ako je barem jedan od njih različit od k, uspješno smo našli netrivijalan faktor broja k. Metoda verižnog razlomka ili (*CFRAC - Continued Fraction Factoring Algorithm*) je jedan od bržih algoritama za faktorizaciju. Korištene formule mogu se naći u [3].

Primjer 3.5.2. *Metodom verižnog razlomka faktorizirat ćemo broj* $n = 55$.

Razvoj od $\sqrt{55}$ u verižni razlomak:

i	0	1	2	3	4
s_i	0	7	5	5	7
t_i	1	**6**	5	**6**	1
q_i	7	2	2	2	14
$m_{i_{pravi}}$	7	15	37	89	1283
m_i	7	15	37	34	18.

Uočimo kako je $t_1^* \cdot t_3^* = 36 = 6^2$. Stoga je

$$m_0^2 m_2^2 = (7 \cdot 37)^2 \equiv 259^2 \equiv 39^2 \equiv 6^2 \pmod{55}.$$

Sada računamo

$$\text{NZD}(39 + 6, 55) = 5 \text{ i } \text{NZD}(39 - 6, 55) = 11.$$

Pronašli smo faktore broja 55 i zaista vrijedi $55 = 5 \cdot 11$.

Primjer 3.5.3. *Metodom verižnog razlomka faktorizirat ćemo broj* $n = 9073$.

Razvoj od $\sqrt{9073}$ u verižni razlomak:

i	0	1	2	3	4	5
s_i	0	95	49	90	92	82
t_i	1	**48**	139	7	87	**27**
q_i	95	3	1	26	2	6
$m_{i_{pravi}}$	95	286	381	10192	20765	134782
m_i	95	286	381	1119	2619	16833.

Uočimo kako je $t_1^* \cdot t_5^* = 1296 = 36^2$. Stoga je

$$m_0^2 m_4^2 = (95 \cdot 2619)^2 \equiv 3\,834^2 \equiv 36^2 \pmod{9\,073}.$$

Sada računamo

$$\text{NZD}(3\,834 + 36, 9\,073) = 43 \text{ i } \text{NZD}(3\,834 - 36, 9\,073) = 211.$$

Pronašli smo faktore broja 9 073 i zaista vrijedi $9\,073 = 43 \cdot 211$.

3.6 QS

U ovom poglavlju ćemo opisati Metodu kvadratnog sita ili *QS - Quadratic sieve*. Jednu od prvih implementacija kvadratnog sita proveo je Gerver 1982. godine na 47-znamenkastom broju. Prvi korak, računanje kongruencija, trajao je 7 minuta, a treći korak, Gaussova eliminacija, trajao je oko 6 minuta. Iako se takav vremenski period čini razuman, samo provjeravanje brojeva kroz sito trajalo je preko 70 sati. Objasnit ćemo princip ove metode na primjeru. Faktorizirajmo broj $n = 4\,999\,486\,012\,441$. Računat ćemo s bazom od 30 prostih brojeva i deset tisuća vrijednosti za r.

$$\lfloor \sqrt{4\,999\,486\,012\,441} \rfloor = 2\,235\,953.$$

Umjesto da uzmemo $r = 2\,235\,953$ i sljedećih 10 000 brojeva, uzet ćemo

$$r = 2\,230\,953 + i, 1 \leq i \leq 10\,000,$$

jer *okružuju* korijen broja n. Zbog takvog odabira vrijednosti r vrijednost funkcije $f(x) = r^2 - n$ je bliže 0, a što je vrijednost $f(x)$ manja, veća je vjerojatnost da ćemo n uspjeti faktorizirati. Naravno, zbog takvog odabira dobit ćemo i negativne vrijednosti, ali možemo izlučiti -1 iz njih i tretirati -1 kao prvi prosti broj od njih 30 u bazi prostih brojeva.

Definicija 3.6.1. *Za cijeli broj n kažemo da je kvadratni ostatak modulo p ako postoji cijeli broj x takav da je*

$$x^2 \equiv n \pmod{p}.$$

Za svaki prosti broj p u bazi, n je kvadratni ostatak modulo p. Kako je n kongruentan s 1 modulo 8, možemo tretirati 8 kao element baze faktora koja dijeli $r^2 - n$ kada god je r neparan. U teoriji brojeva **Legendreov simbol** je multiplikativna funkcija za koju vrijedi:

$$(n/p) = \begin{cases} 0 & \text{ako } p \mid n \\ 1 & \text{ako } p \nmid n \text{ i } n \text{ je kvadratni ostatak modulo } p \\ -1 & \text{ako } p \nmid n \text{ i } n \text{ nije kvadratni ostatak modulo } p \end{cases}$$

Sada tražimo još 28 prostih brojeva koji zadovoljavaju $(n/p) = +1$. U ovom slučaju to su:

3	19	59	163	229	277	359
5	31	16	181	241	311	367
7	43	67	193	263	331	389
17	47	107	197	271	349	397

Za svaki ovaj broj moramo riješiti kongruenciju $n \equiv t^2 \pmod{p}$. Sada gledamo najmanje moguće rješenje t, ili ostatak pri dijeljenju s p ili $p - t$.

1	1	14	38	7	39	171
1	5	16	19	18	39	125
2	19	6	86	101	65	69
3	18	32	14	22	52	50

Umjesto da računamo logaritam apsolutne vrijednosti od $f(2\,230\,953 + i)$ deset tisuća puta, počet ćemo s nultim vektorom kojem dodajemo $\log p$ kada p dijeli odgovarajući $f(r)$. Ako je nakon "prosijavanja" kroz sito ostatak manji od kvadrata najvećeg broja u bazi prostih brojeva, onda je taj broj prost. Primijetimo da je r neparan kada je i paran, pa je prvi korak dodati $\log 8$ svakoj drugoj vrijednosti. Kako je 2 230 953 djeljiv s 3, dodajemo $\log 3$ prvoj vrijednosti, pa svakoj trećoj, zatim drugoj vrijednosti i ponovno svakoj trećoj.

Modulo 5, r je kongruentan s $3 + i$, takav da je r kongruentan s 1 ili 4 ako i samo ako je i kongruentan s 3 ili 1 modulo 5. Za sve takve, dodajemo $\log 5$ odgovarajućem unosu. Slično je r kongruentan s 2 ili 5 modulo 7 ako i samo ako je i kongruentan s 5 ili 1. Takvim unosima dodajemo $\log 7$. Sito nam daje čak 39 kompletnih faktorizacija u ovoj bazi prostih faktora:

$$
\begin{array}{rrl}
i = & 243; & f(r) = 5^2 \cdot 7 \cdot 17 \cdot 107 \cdot 241 \cdot 277 \\
& 484; & 2^4 \cdot 3^3 \cdot 7 \cdot 19 \cdot 31 \cdot 47 \cdot 241 \\
& 649; & 3 \cdot 7^2 \cdot 31 \cdot 59 \cdot 197 \cdot 367 \\
& 1548; & 2^5 \cdot 5 \cdot 7 \cdot 19 \cdot 31 \cdot 67 \cdot 349 \\
& 1755; & 7 \cdot 19 \cdot 31 \cdot 43 \cdot 263 \cdot 311 \\
& 2336; & 2^3 \cdot 3 \cdot 5 \cdot 7 \cdot 193 \cdot 271^2 \\
& 2878; & 2^3 \cdot 3^2 \cdot 5 \cdot 7 \cdot 17 \cdot 19 \cdot 43 \cdot 271 \\
& 2916; & 2^4 \cdot 5 \cdot 19 \cdot 43 \cdot 359 \cdot 397 \\
& 3218; & 2^7 \cdot 3 \cdot 5^2 \cdot 7 \cdot 17 \cdot 19 \cdot 367 \\
& 3292; & 2^8 \cdot 3^3 \cdot 17 \cdot 181 \cdot 359 \\
& & \\
& 3394; & 2^5 \cdot 3 \cdot 17 \cdot 43 \cdot 263 \cdot 389 \\
& 4094; & 2^3 \cdot 3^2 \cdot 19 \cdot 67 \cdot 193 \cdot 229 \\
& 4340; & 2^4 \cdot 3 \cdot 17 \cdot 43 \cdot 241 \cdot 349 \\
& 4476; & 2^6 \cdot 5^2 \cdot 17 \cdot 277 \cdot 311 \\
& 4630; & 2^3 \cdot 3 \cdot 59 \cdot 67 \cdot 107 \cdot 163 \\
& 4686; & 2^3 \cdot 5 \cdot 17^2 \cdot 331 \cdot 367 \\
& 4786; & 2^6 \cdot 3^2 \cdot 5 \cdot 7 \cdot 197 \cdot 241 \\
& 4793; & 3 \cdot 5^2 \cdot 7 \cdot 31 \cdot 163 \cdot 349 \\
& 4866; & 2^5 \cdot 5 \cdot 7 \cdot 47 \cdot 59 \cdot 193 \\
& 4901; & 3 \cdot 5^5 \cdot 7 \cdot 17 \cdot 397 \\
& & \\
& 5038; & 2^3 \cdot 3^2 \cdot 5 \cdot 7 \cdot 193 \cdot 349 \\
& 5043; & 5^2 \cdot 17 \cdot 47 \cdot 59 \cdot 163 \\
& 5101; & 3^3 \cdot 5^2 \cdot 7 \cdot 19 \cdot 47 \cdot 107 \\
& 5445; & 17 \cdot 61^2 \cdot 163 \cdot 193 \\
& 5506; & 2^5 \cdot 3^3 \cdot 5 \cdot 31 \cdot 61 \cdot 277 \\
& 5683; & 3 \cdot 5 \cdot 17^3 \cdot 181 \cdot 229 \\
& 5840; & 2^3 \cdot 3^3 \cdot 19 \cdot 43 \cdot 61 \cdot 349 \\
& 6506; & 2^4 \cdot 3^2 \cdot 5 \cdot 67 \cdot 359 \cdot 389 \\
& 6550; & 2^3 \cdot 3i2 \cdot 7 \cdot 17^2 \cdot 181 \cdot 263 \\
& 7780; & 2^4 \cdot 3 \cdot 17 \cdot 19 \cdot 43 \cdot 47 \cdot 397
\end{array}
$$

8216; $2^3 \cdot 3^4 \cdot 5 \cdot 7 \cdot 17 \cdot 163 \cdot 229$
8528; $2^3 \cdot 3 \cdot 5 \cdot 17 \cdot 67 \cdot 331 \cdot 349$
8678; $2^3 \cdot 3 \cdot 5 \cdot 7 \cdot 31 \cdot 43 \cdot 61 \cdot 241$
8726; $2^3 \cdot 3 \cdot 5^2 \cdot 17 \cdot 43 \cdot 193 \cdot 197$
8908; $2^5 \cdot 3^3 \cdot 5 \cdot 47 \cdot 277 \cdot 311$
9226; $2^4 \cdot 3 \cdot 5^2 \cdot 31^2 \cdot 47 \cdot 349$
9378; $2^5 \cdot 5 \cdot 7 \cdot 17 \cdot 47 \cdot 61 \cdot 359$
9763; $3^2 \cdot 5 \cdot 7^2 \cdot 17 \cdot 31 \cdot 59 \cdot 311$
9908; $2^4 \cdot 3^2 \cdot 5 \cdot 19 \cdot 31 \cdot 197 \cdot 263$

Sada koristimo Gaussovu eliminaciju kako bi pronašli umnožak $f(r)$-ova koji je potpun kvadrat. Potpun kvadrat je broj dobiven kvadriranjem prirodnog broja. Uz svaki i za kojeg možemo faktorizirati $f(2230953 + i)$ možemo povezati 30 binarnih znamenaka. Svaki stupac odgovara jednom od 30 prostih brojeva u bazi faktora. Znamenku 1 zapisat ćemo na odgovarajućoj poziciji ako se prosti faktor pojavljuje u neparnoj potenciji, a 0 ako se pojavljuje u parnoj. Kako imamo 39 faktoriziranih brojeva, imamo matricu s 39 redaka i 30 stupaca. Kako bi pregledno pratili koja kombinacija $f(r)$-ova nam daje potpun kvadrat, kreirat ćemo još jednu matricu 39×39. Prvih 10 redaka naših matrica izgledaju ovako:

```
000000010010000010000000110001    100000000000000000000000000...
000000000010000000001011010101    010000000000000000000000000...
001000000000100000010010000101    001000000000000000000000000...
000010000000000001000011011011    000100000000000000000000000...
000000100100000000000111010001    000010000000000000000000000...
000000000000010000000000011111    000001000000000000000000000...
000000001000000000000101111011    000000100000000000000000000...
110100000000000000000101001001    000000010000000000000000000...
001000000000000000000001110111    000000001000000000000000000...
000100000000001000000000100101    000000000100000000000000000...
.                                 .
.                                 .
.                                 .
```

Počevši s prvim stupcem na lijevoj strani, nalazimo prvi niz s jedinicom u tom stupcu i dodamo ga po modulu 2 svakom sljedećem nizu s jedinicom u tom istom stupcu. Zatim eliminiramo prvi redak s jedinicom u prvom stupcu. Ostalih 38 redaka sada imaju nulu u prvom stupcu. Postupak se ponavlja tako da nađemo prvi redak s jedinicom u drugom stupcu, dodajemo ga po modulu 2 svakom sljedećem s jedinicom na istoj poziciji itd. Postupak ponavljamo dok ne dobijemo redak kojemu je prvih 30 znamenaka 0. Zadnjih

39 znamenaka ovog retka govore nam koji su od naših početnih 39 stringova bili zbrojeni. Prvi string, odnosno redak, koji ima prvih 30 znamenaka 0 je:

$$...000000000 \qquad 000000000000100010001000000000000100000.$$

Jedinice koje se nalaze u stupcima 13, 17, 21 i 34 govore nam da je umnožak odgovarajućih $f(2230953 + i)$ jednak potpunom kvadratu. Te vrijednosti i su 4340, 4786, 5038 i 8726. Pomnožimo li ih dobijemo:

$$(2230953 + 43409)^2 \times (2230953 + 4786)^2 \times (2230953 + 5038)^2 \times (2230953 + 8726)^2$$

$$\equiv 2^{16} \times 3^6 \times 5^4 \times 7^2 \times 17^2 \times 43^2 \times 193^2 \times 241^2 \times 349^2 \pmod{4\,999\,486\,012\,441}.$$

Sada imamo kongruenciju tipa

$$x^2 \equiv y^2 \pmod{n}.$$

Računamo x i y modulo n. U našem slučaju korijen obje strane modulo 4 999 486 012 441 je isti: 2 497 001 218 533, odnosno, vrijedi $\text{NZD}(x - y, n) = 1$ ili n. Nastavimo li s Gaussovom eliminacijom, sljedeći string koji ima prvih 30 znamenaka 0 je:

$$...000000000 \qquad 101000010100001110010000111000110000000.$$

Ovaj put vrijednosti x i y su:

$$x = 3\,665\,300\,235\,664; \qquad y = 915\,847\,468\,484.$$

$\text{NZD}(x - y, n) = 999,961$ pa imamo faktorizaciju:

$$4\,999\,486\,012\,441 = 999\,961 \times 4\,999\,681.$$

Bibliografija

[1] D. M. Bressoud, *Factorization and Primality Testing*, Springer-Verlag, 1989.

[2] A. Dujella, *Eliptičke krivulje u kriptografiji*, skripta

[3] A. Dujella, *Uvod u teoriju brojeva*, skripta

[4] G. H. Hardy, E. M.Wright, *An Introduction to the Theory of Numbers*, Oxford University Press, Oxford, 2008.

[5] N. P. Smart, *Cryptography Made Simple*, Springer, 2016.

[6] S. S. Wagstaff, Jr., *The joy of factoring*, AMS, 2013.

Sažetak

Ovaj rad fokusira se na proste brojeve i faktorizacijske algoritme velikih prirodnih brojeva. Prosti ili prim brojevi su jedan od fundamentalnih pojmova u matematici. To su brojevi koji su djeljivi samo s brojem 1 i sa samim sobom. Algoritam je konačan slijed dobro definiranih naredbi za ostvarenje zadatka. Faktorizacija velikih prirodnih brojeva i pronalazak velikih prostih brojeva iznimno je bitan u kriptografiji. Još u 3. stoljeću pr. Kr. Euklid je dokazao da ne postoji najveći prosti broj te da prostih brojeva ima beskonačno mnogo. Kako se skup prirodnih brojeva nastavlja, prosti brojevi postaju sve manje učestali, te je pronalazak velikih prostih brojeva postao zanimacija mnogih. U prvom poglavlju komentirat ćemo učestalost prostih brojeva i što su to Mersennovi prosti brojevi.

U drugom poglavlju upoznat ćemo se s raznim testovima prostosti. Naivni način za provjeru prostosti broja je pokusno dijeljenje, a obrađeni su i Fermatov i Miller-Rabinov test.

Ako prirodan broj n ne prođe neki od testova prostosti, onda znamo da je n sigurno složen broj. Iako su korisni za provjeru prostosti, takvi testovi nam najčešće ne daju niti jedan netrivijalni faktor od n. Postavlja se pitanje kako naći netrivijalni faktor velikog složenog broja i koji algoritam je najbrži za pronalazak takvih faktora. To se smatra teškim problemom i na njegovoj težini su zasnovani neki od najvažnijih kriptosustava s javnim ključem. U trećem poglavlju obradit ćemo 6 algoritama za faktorizaciju prirodnih brojeva.

Summary

This paper focuses on prime numbers and factorisation algorithms of large numbers. Prime numbers have only two factors, number 1 and itself, and are one of the fundamental terms in mathematics. Algorithm is a process or set of rules to be followed in calculations or other problem-solving operations. Factorization of large numbers and finding large prime numbers is extremely important in cryptography. In 3rd century BC, Euclid proved that the largest prime number doesn't exist, and that there are infinite number of primes. As the set of natural numbers continues, they become less frequent, so finding new prime numbers has become a mission for many people. In chapter 1, we will discuss frequency of prime numbers and what are Mersenne's prime numbers.

In chapter 2, we will talk about primality testing. Most basic way to check if a number is prime is by trial division. Fermat's primality test and Miller-Rabin's test is also discussed in this paper.

If a number n doesn't pass one of the primality tests, then we know for sure that n is a composite number. Even though they are useful for primality testing, these tests rarely give us a non-trivial factor of n. How can we find factors of large numbers, and what is the fastest algorithm to do that? That is considered a hard problem and therefore some of the most important cryptosystems with public key are based on the complexity of factorization. In chapter 3 we will discuss six factorisation algorithms.

Životopis

Marija Laća rođena je 18. veljače 1995. godine u Šibeniku. Nakon završene osnovne glazbene škole Ivana Lukačića i osnovne škole Juraj Dalmatinac, upisuje matematički smjer gimnazije Antuna Vrančića u Šibeniku. Prirodoslovno-matematički fakultet upisala je 2013. godine, a 2017. godine završila je preddiplomski sveučilišni studij Matematika, nastavnički smjer. Obrazovanje nastavlja na diplomskom smjeru Matematika i informatika u sklopu kojeg odrađuje metodičku praksu matematike i informatike u osnovnoj školi Trnsko i XV. gimnaziji.