Pollardove metode za faktorizaciju

Gaćina, Martina

Master's thesis / Diplomski rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Science / Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:217:897856

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Martina Gaćina

POLLARDOVE METODE ZA FAKTORIZACIJU

Diplomski rad

Zagreb, veljača, 2022.

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Martina Gaćina

POLLARDOVE METODE ZA FAKTORIZACIJU

Diplomski rad

Voditelj rada:
prof. dr. sc. Andrej Dujella

Zagreb, veljača, 2022.

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. _____

Svima koji su me trpili i Dariji.

Sadržaj

Uvod

U teoriji brojeva, cjelobrojna faktorizacija je rastav složenog cijelog broja u produkt manjih cijelih brojeva. Ako su ti brojevi dodatno ograničeni na proste faktore, onda ovaj proces nazivamo rastav broja na proste faktore. Faktorizacija cijelih brojeva je bila predmet proučavanja već kod starogrčkih matematičara. Oni su dokazali da se svaki prirodan broj može rastaviti na produkt prostih faktora koji se dalje ne mogu rastaviti na cijele brojeve veće od 1. Štoviše, takva faktorizacija je jedinstvena do na poredak prostih faktora. O tome nam govori osnovni teorem aritmetike. Pronalazak nekog od prostih faktora složenog broja smatra se vrlo teškim problemom. Zbog toga nailazimo na široku primjenu faktorizacije u kriptografiji. Sigurnost RSA kriptosustava, jednog od najpopularnijih i najšire korištenih kriptosustava, zasniva se na teškoći faktorizacije velikih prirodnih brojeva. Postoji znatan broj algoritama za faktorizaciju, no za dovoljno velike brojeve ne postoji poznati efikasni algoritam. Najjednostavniji, ali i vremenski najzahtjevniji algoritam za faktorizaciju je *probno dijeljenje*. Ideja probnog dijeljenja je provjeriti s kojim brojevima manjim od n je n djeljiv. Lako je vidjeti da je zapravo dovoljno provjeravati brojeve manje od $\sqrt{n}$. Jasno je da ova metoda nije efikasna za velike brojeve. Međutim, za brojeve nekog određenog oblika postoje algoritmi koji jesu efikasni. Kada budemo razmatrali složenost algoritama za faktorizaciju, uglavnom ćemo pričati o vremenskoj složenosti, odnosno o očekivanom broju operacija koje algoritam izvodi. Za oznaku vremenske složenosti koristimo notaciju veliko O.

Definicija 0.0.1. *Neka su $f, g\colon \mathbb{N} \to \mathbb{R}^+$ dvije funkcije. Kažemo da je funkcija g asimptotska gornja međa za funkciju f ako postoji $c > 0$ i $n_0 \in \mathbb{N}$ tako da za svaki $n \geq n_0$ vrijedi $f(n) \leq cg(n)$. Oznaka: $f(n) = O(g(n))$.*

Metode faktorizacije dijelimo na opće i na specijalne. Kod općih metoda očekivani broj operacija ovisi samo o veličini broja n, a kod specijalnih ovisi i o svojstvima faktora od n. Probno dijeljenje je primjer jedne opće metode faktorizacije. Složenost algoritma za probno dijeljenje, u slučaju kada provjeravamo sve brojeve manje od n, je $O(n)$. Ako provjeravamo za brojeve manje od $\sqrt{n}$, tada je složenost $O(\sqrt{n})$. Primijetimo da je ovo vremenska složenost u najgorem slučaju, odnosno kada je n prost. Trenutno, najbrži algoritmi za faktorizaciju trebaju $\exp\left(O\left((\log n)^{\frac{1}{3}}(\log\log n)^{\frac{2}{3}}\right)\right)$ operacija.

John M. Pollard je britanski matematičar koji je izumio neke algoritme za faktorizaciju velikih brojeva i za izračun diskretnih logaritama. U ovom diplomskom radu proučit ćemo dvije Pollardove metode za faktorizaciju i neke njihove varijante. Također, istražit ćemo i jednu Pollardovu metodu za izračun diskretnih logaritama. U poglavlju 1 ćemo se prisjetiti nekih bitnih tvrdnji vezanih za kongruencije koje će nam biti potrebne za proučavanje metoda za faktorizaciju i računa diskretnih logaritama [5]. Metoda kojom ćemo se baviti u poglavlju 2 je takozvana Pollardova ρ metoda za faktorizaciju. Ova metoda spada u specijalne metode faktorizacije i vremenska složenost joj ovisi o najmanjem prostom faktoru broja kojeg želimo faktorizirati. Analizirat ćemo i dvije varijante Pollardove ρ metode koje smanjuju broj operacija potrebnih za pronalazak prostog faktora. Prva varijanta, Floydova varijanta (potpoglavlje 2.1), koristi Floydovu metodu za pronalaženje ciklusa, a druga, Brentova (potpoglavlje 2.3), koristi Brentovu metodu za pronalaženje ciklusa. Također, vidjet ćemo na koji način možemo dodatno smanjiti broj operacija potrebnih u algoritmima (potpoglavlje 2.2). Za potrebe pisanja ovog poglavlja većinom ću pratiti knjigu [10]. U poglavlju 3 ćemo obraditi Pollardovu ρ metodu za izračun diskretnih logaritama, analognu Pollardovoj ρ metodi za faktorizaciju. Problem diskretnog logaritma se također smatra teškim problemom te nailazimo na njegove česte primjene u kriptografiji. Sigurnost mnogih kriptosustava, kao što su Diffie-Helmanov protokol razmjene ključeva, algoritam digitalnog potpisa i Schnorr shema potpisa, se temelji na problemu diskretnog logaritma. Kao glavna literatura za obradu Pollardove ρ metode za diskretne logaritme korištena je [3]. U poglavlju 4 ćemo opisati Polladovu $p-1$ metodu za faktorizaciju. Ova metoda također spada u specijalne metode za faktorizaciju, a uspjeh joj ovisi o glatkoći broja $p-1$, gdje je p prosti faktor kojeg tražimo. U najgorem slučaju, kada je broj $\frac{p-1}{2}$ prost, ova metoda nije ništa bolja od metode probnog dijeljenja. Postoje mnoge varijante Pollardove $p-1$ metode kao što su $p+1$ metoda faktorizacije i faktorizacija pomoću eliptičkih krivulja, no mi ćemo se baviti samo drugom fazom ove metode. Opisat ćemo takozvanu varijaciju velikog prostog broja (potpoglavlje 4.1) i njezino unaprjeđenje. Za obradu Pollardove $p-1$ metode korištena je literatura [3], [4], [6] i [10].

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

Uvod u kongruencije

Pollardove metode za faktorizaciju i za izračun diskretnog logaritma se temelje na kongruencijama. U ovom poglavlju ćemo se prisjetiti što je to kongruencija i nekih pojmova koji će nam biti potrebni za lakše razumijevanje daljnjeg teksta. Navedene teoreme i propozicije ćemo samo iskazati, a dokaze možemo pronaći u [5].

Definicija 1.0.1. *Neka su a i b cijeli brojevi i neka je m prirodan broj $\neq 0$. Kažemo da je a kongruentan b modulo m i pišemo $a \equiv b \pmod{m}$ ako m dijeli razliku $a - b$. U protivnom, kažemo da a nije kongruentan b modulo m i pišemo $a \not\equiv b \pmod{m}$.*

Jer je razlika $a - b$ djeljiva s m ako i samo ako je djeljiva s $-m$, možemo promatrati samo pozitivne module. U daljnjem tekstu pretpostavljamo da je m pozitivan cijeli broj.

Teorem 1.0.2 (Eulerov teorem)**.** *Ako su a i m relativno prosti, onda je $a^{\varphi(m)} \equiv 1 \pmod{m}$, gdje je $\varphi(m)$ = broj relativno prostih brojeva s m manjih od m.*

Iz Eulerovog teorema direktno slijedi iduća tvrdnja za proste brojeve.

Teorem 1.0.3 (Mali Fermatov teorem)**.** *Neka je p prost broj i $a \in \mathbb{Z}$ takav da vrijedi $\gcd(a, p) = 1$, tj. p ne dijeli a. Tada vrijedi*

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

Nadalje, za svaki cijeli broj a vrijedi $a^p \equiv a \pmod{p}$.

U modularnoj aritmetici, cijeli brojevi iz skupa $\{0, 1, \ldots, m-1\}$ relativno prosti s m tvore grupu s množenjem modulo m. Tu grupu nazivamo *multiplikativna grupa cijelih brojeva modulo m* i označavamo s $\mathbb{Z}_m^*$. Ekvivalentno, na elemente ove grupe možemo gledati kao na klase ekvivalencije, poznate kao ostaci modulo m koji su relativno prosti s m. Otuda drugo ime grupa primitivnih klasa ostataka modulo m.

Neka je $a \in \mathbb{Z}_m^*$ za neki $m \in \mathbb{N}$. Znamo da je $\mathbb{Z}_m^*$ konačan, pa onda ne mogu svi $a, a^2, a^3, \ldots$ modulo m biti različiti. Dakle, postoje $i, j \in \mathbb{N}$, $i > j$ takvi da vrijedi $a^i \equiv a^j \pmod{m}$, odnosno $a^{i-j} \equiv 1 \pmod{m}$.

Definicija 1.0.4. *Neka su a i m relativno prosti prirodni brojevi. Kažemo da je $d \in \mathbb{N}$ multiplikativni red (ili samo red) od a modulo m ako je d najmanji prirodni broj za koji vrijedi $a^d \equiv 1 \pmod{m}$ i to označujemo sa $\mathrm{ord}_m a$. Ako je $d = \mathrm{ord}_m a$, također kažemo da a pripada eksponentu d modulo m.*

Propozicija 1.0.5. *Neka je d red od a modulo m. Tada za $k \in \mathbb{N}$ vrijedi $a^k \equiv 1 \pmod{m}$ ako i samo ako d dijeli k. Posebno, $d|\varphi(m)$.*

Iz Propozicije 1.0.5 i Teorema 1.0.3 slijedi da za svaki višekratnik k od $p-1$ vrijedi $a^k \equiv 1 \pmod{p}$. Ovaj rezultat će nam biti krucijalan za ideju Pollardove $p-1$ metode za faktorizaciju.

Definicija 1.0.6. *Kažemo da je a primitivni korijen modulo m ako je svaki broj g koji je relativno prost s m kongruentan s potencijom broja a modulo m. Drugim riječima, a je primitivni korijen modulo m ako za svaki cijeli broj g relativno prost s m postoji neki cijeli broj k za koji je $a^k \equiv g \pmod{m}$.*

Dakle, $a \in \mathbb{Z}_m^*$ je primitivni korijen modulo m ako njegove potencije generiraju cijeli $\mathbb{Z}_m^*$. Grupa $\mathbb{Z}_m^*$ ima $\varphi(m)$ elemenata. Promotrimo niz potencija nekog elementa $a \in \mathbb{Z}_m^*$

$$1, a, a^2, a^3, \ldots \pmod{m}. \tag{1.1}$$

Već smo rekli kako se taj niz u nekom trenutku mora početi ponavljati. Eulerov teorem 1.0.2 nam kaže da će se to sigurno dogoditi nakon $\varphi(m)$ elemenata, odnosno period niza (1.1) je $\varphi(m)$. Primijetimo da je minimalan period tog niza zapravo red od a modulo m. Sada dolazimo do jedne karakterizacije primitivnog korijena: a je primitivni korijen modulo m ako i samo ako je red od a modulo m jednak $\varphi(m)$.

Definicija 1.0.7. *Neka je a primitivni korijen modulo m i neka je g relativno prost s m. Kažemo da je $k \in \mathbb{N}$ indeks ili diskretni logaritam od g po bazi a modulo m ako vrijedi $a^k \equiv g \pmod{m}$.*

Poglavlje 2

Pollardova ρ metoda za faktorizaciju

Pollardova ρ metoda ili Pollardova druga metoda faktorizacije spada u klasu vjerojatnosnih algoritama zvanu Monte Carlo. Monte Carlo algoritmi su algoritmi koji uvijek daju odgovor, ali on ne mora biti točan. Metoda je najuspješnija za velike brojeve koji imaju barem jedan mali prosti faktor.

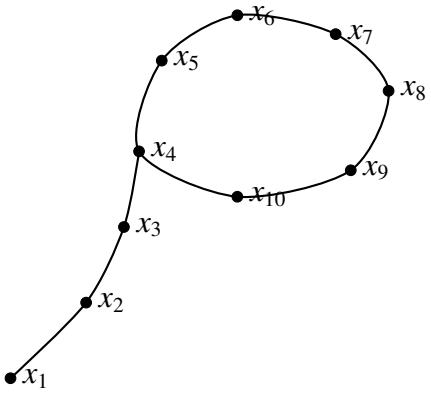
Neka je $n \in \mathbb{N}$ broj koji želimo faktorizirati i neka je $p \in \mathbb{N}$ jedan njegov nepoznat prosti faktor. Ideja Pollardove ρ faktorizacije se sastoji od sljedećih koraka:

1. Konstruiramo niz (x_i) cijelih brojeva koji je periodičan modulo p.
2. Nađemo period, tj. nađemo i i j takve da je $x_i \equiv x_j \pmod{p}$.
3. Odredimo faktor p.

Pogledajmo prvo kako bismo konstruirali niz iz koraka 1 za proizvoljan $m \in \mathbb{Z}$. Uzmimo neko slučajno preslikavanje F i promotrimo sljedeći rekurzivno definiran niz modulo m:

$$x_i \equiv F(x_{i-1}, x_{i-2}, \ldots, x_{i-s}) \pmod{m}, \quad i \in \mathbb{N}, \tag{2.1}$$

gdje su početne vrijednosti $x_1, x_2, \ldots, x_s$ zadane, s je konstanta koja ne ovisi o i i $i > s$. Sve vrijednosti od F uzimamo modulo m, stoga svaki x_i može poprimiti samo konačno mnogo različitih vrijednosti, točnije m. Dakle, postoji najviše m^s različitih nizova od s uzastopnih brojeva $x_{i-1}, x_{i-2}, \ldots, x_{i-s}$ pa nakon $m^s + 1$ koraka rekurzije se moraju ponoviti dva ista niza. Označimo ih sa $x_{j-1}, x_{j-2}, \ldots, x_{j-s}$ i $x_{k-1}, x_{k-2}, \ldots, x_{k-s}$. Jer vrijednosti niza (2.1) ovise samo o prethodnih s brojeva, onda će vrijednosti x_j i x_k dobivene iz dva jednaka niza biti jednake. Sada promotrimo nizove $x_j, x_{j-1}, \ldots, x_{j-s+1}$ i $x_k, x_{k-1}, \ldots, x_{k-s+1}$. Jasno je da su i ta dva niza identična te da će i vrijednosti x_{j+1} i x_{k+1} također biti jednake i tako dalje. Zaključujemo da je niz (x_i) periodičan modulo m, osim možda njegovog početnog dijela. Zbog ovoga je metoda i dobila ime ρ metoda. Naime, aperiodični dio niza podsjeća na "rep" grčkog slova ρ, a period na okrugli dio.



Slika 2.1: Ilustracija niza u obliku slova ρ

Primjer 2.0.1 (Pellovi brojevi)**.** *Pellovi brojevi su zadani rekurzijom:*

$$P_n := \begin{cases} 0, & n = 0 \\ 1, & n = 1 \\ 2P_{n-1} + P_{n-2}, & \textit{inače} \end{cases},$$

tj. $P(n) = 2P(n-1) + P(n-2), \quad P(0) = 0, \quad P(1) = 1$. *Prvih nekoliko članova niza Pellovih brojeva modulo* 5 *su*

$$0, 1, 2, 0, 2, 4, 0, 4, 3, 0, 3, 1, 0, 1, 2, 0, \ldots \pmod 5.$$

Vidimo da se niz počinje ponavljati nakon 12. *člana i da nema početnog aperiodičnog dijela, već se niz ponavlja od samog početka.*

Još jedan primjer sa slavnim Fibonaccijevim brojevima možemo vidjeti u knjizi Prime Numbers and Computer Methods for Factorization [10].

Pronalaženje perioda može biti vrlo mukotrpan zadatak, pogotovo ako je period jako dugačak. Potrebno je pronaći gdje se niz od s uzastopnih elemenata počinje ponavljati. U najgorem slučaju, niz će se početi ponavljati tek nakon m^s+1 koraka rekurzije, gdje su m i s kao i u (2.1). Zbog velike količine posla, ograničavamo se na najjednostavniji slučaj: kada x_i ovisi samo o svom prethodniku x_{i-1}. U ovom slučaju se niz počinje ponavljati čim se jedan element niza ponovi, a to će se dogoditi u najviše $m+1$ koraka. Ovime smo si uvelike olakšali zadatak jer sada moramo samo uspoređivati svaki novi element s prethodnima, a ne cijeli niz elemenata.

Sada smo vidjeli na koji način možemo konstruirati niz cijelih brojeva periodičan modulo m za proizvoljan m, no kako konstruirati niz periodičan modulo p ako je p nepoznat? Zapravo ćemo prvo konstruirati niz (x_i) periodičan modulo n čije vrijednosti možemo

izračunati. Neka je $F\colon \mathbb{Z}_n \to \mathbb{Z}_n$ preslikavanje i $x_0 \in \mathbb{Z}_n$ proizvoljan. Konstruiramo niz (x_i) kao

$$x_0, \quad x_1 = F(x_0), \quad x_2 = F(F(x_0)), \ldots.$$

Odnosno, $x_{i+1} = F(x_i)$. Nadalje, definiramo $f\colon \mathbb{Z}_p \to \mathbb{Z}_p$ kao $f(x) = F(x) \pmod p$. Faktor p sada možemo izračunati koristeći Euklidov algoritam za pronalaženje najvećeg zajedničkog djelitelja brojeva $x_i - x_j \pmod n$ i n za sve i, j. Označimo $d = \gcd(x_i - x_j, n)$. Ako je $x_i \equiv x_j \pmod p$, onda $p|(x_i - x_j)$ i $p|n$ pa će i d biti djeljiv s p. U suprotnom ćemo za d dobivati trivijalne vrijednosti $d = 1$ ili $d = n$.

2.1 Floydova varijanta Pollardove ρ metode

Pronalaženje perioda na opisan način i računanje $d = gcd(x_i - x_j, n)$ za sve i, j je vremenski zahtjevno i neefikasno. Ako je period jako velik nije moguće pamtiti sve prethodne elemente (npr. nekoliko milijuna) i uspoređivati ih te računati d dok ne nađemo $d > 1$. Tu nam stiže u pomoć Floydov algoritam za pronalaženje ciklusa koji kaže da je dovoljno provjeriti je li $x_i \equiv x_{2i} \pmod p$. Ako vrijedi $x_i \equiv x_{2i} \pmod p$ za neki i, onda je niz očito periodičan od x_{2i} nadalje, možda i ranije.

Propozicija 2.1.1. *Neka je (x_i) niz cijelih brojeva periodičan modulo p, $p \in \mathbb{N}$. Tada postoji $k \geq 1$ takav da vrijedi $x_k \equiv x_{2k} \pmod p$.*

Dokaz. Jer je niz (x_i) periodičan modulo p, onda postoje $i, j \in \mathbb{N}$, $i < j$ t. d. vrijedi $x_i \equiv x_j \pmod p$. Jasno je da za svaki $k \geq i$ i za svaki $s \in \mathbb{N}$ vrijedi $x_k \equiv x_{k+s(j-i)} \pmod p$. Ako za k uzmemo neki višekratnik od $j - i$, tj. $k = l(j - i), l \in \mathbb{N}$, onda imamo

$$x_k \equiv x_{k+s(j-i)} \equiv x_{l(j-i)+s(j-i)} \equiv x_{(l+s)(j-i)} \pmod p. \tag{2.2}$$

Jer (2.2) vrijedi za svaki $s \in \mathbb{N}$, onda vrijedi i za $s = l$ pa imamo

$$x_k \equiv x_{(l+s)(j-i)} \equiv x_{2l(j-i)} \equiv x_{2k} \pmod p. \tag{2.3}$$

Dakle, postoji $k \geq 1$ za koji vrijedi (2.3). □

Sada kada to znamo, možemo računati samo $d_k = \gcd(x_{2k} - x_k, n)$ za $k = 1, 2, 3, \ldots$ i nakon nekog vremena ćemo sigurno pronaći $d_k > 1$. No, kako uspoređivati x_k i x_{2k} te računati d_k bez da pamtimo sve x_k? Jednostavno računamo paralelno vrijednosti od x_k i x_{2k}.

Promatrajući ovaj algoritam možemo uočiti nekoliko stvari koje bi učinile algoritam efikasnijim. Prvo vidimo da niz (x_i) moramo dvaput računati, stoga bismo trebali za f uzeti funkciju koja se može jednostavno izračunati. Najjednostavniji izbor bi bio linearni

Algoritam 1 Floydova varijanta Pollardove ρ metode

Ulaz: funkcija f, n i $x_0 \in \mathbb{Z}_n$

```
x ← x_0
y ← x_0
while 1 do
    x ← f(x)                          ▷ x = x_k
    y ← f(f(y))                       ▷ y = x_2k
    d ← gcd(y − x, n)
    if d > 1 then break
    end if
end while
```

Izlaz: d

polinom, no ispada da on ne proizvodi dovoljno slučajne brojeve. Sljedeći izbor bi bio kvadratni polinom oblika $f(x) = x^2 + a \pmod{p}$. U praksi se pokazalo da je kvadratni polinom sasvim dobar izbor, ali samo za $a \neq 0, 2$, iako to još nije dokazano.

Primjer 2.1.2. *Faktorizirajmo broj* $n = 655703$. *Za generiranje niza pseudoslučajnih brojeva uzmimo funkciju* $f(x) = x^2 + 1 \pmod{n}$. *Počnimo s inicijalnom vrijednošću* $x_0 = 2$. *Račun je prikazan u tablici 2.1. Nakon* 15. *koraka algoritma, otkrili smo netrivijalni faktor* 191 *broja n. Odgovarajući drugi faktor je* 3433, *odnosno* $655703 = 191 \times 3433$. *Oba ova faktora su prosta, dakle u potpunosti smo faktorizirali broj* 655703.

Sada se možemo pitati koliko dugo će trebati kako bi se niz (x_i) počeo ponavljati. Odnosno, koliko koraka k algoritma 1 će trebati kako bismo pronašli dva cijela broja međusobno kongruentna modulo p. Ako se prisjetimo tzv. problema rođendana, odmah je jasno da je naš odgovor usko povezan s tim "paradoksom". Rođendanski problem se bavi pitanjem koliko ljudi treba nasumično odabrati kako bi se postigla određena vjerojatnost da dvoje ljudi ima rođendan na isti datum. Vjerojatnost da svih k osoba ima rođendan na različite datume je

$$\left(1 - \frac{1}{365}\right)\left(1 - \frac{2}{365}\right)\left(1 - \frac{3}{365}\right)\cdots\left(1 - \frac{k-1}{365}\right). \tag{2.4}$$

Ako odaberemo skup od 366 ljudi i izbacimo prijestupni datum 29. veljače iz računa, vjerojatnost će biti 100%. Vjerojatnost je 50% već za skupinu od 23 ljudi, zbog čega se ovaj problem ponekad naziva i *paradoksom rođendana*. Ako generaliziramo (2.4), dobit ćemo odgovor na naše pitanje. Vjerojatnost da je k slučajno odabranih brojeva međusobno nekongruentno je

$$\left(1 - \frac{1}{p}\right)\left(1 - \frac{2}{p}\right)\left(1 - \frac{3}{p}\right)\cdots\left(1 - \frac{k-1}{p}\right) \approx \left(1 - \frac{k}{2p}\right)^{k-1} \approx e^{-k(k-1)/2p} \approx e^{-k^2/2p}. \tag{2.5}$$

x_k	x_{2k}	$\gcd(x_{2k} - x_k, n)$
$x_0 = 2$	$x_0 = 2$	-
$x_1 = 5$	$x_2 = 26$	$\gcd(21, n) = 1$
$x_2 = 26$	$x_4 = 458330$	$\gcd(458304, n) = 1$
$x_3 = 677$	$x_6 = 24854$	$\gcd(24177, n) = 1$
$x_4 = 458330$	$x_8 = 217757$	$\gcd(-240573, n) = 1$
$x_5 = 130197$	$x_{10} = 56788$	$\gcd(-73409, n) = 1$
$x_6 = 24854$	$x_{12} = 617749$	$\gcd(592895, n) = 1$
$x_7 = 49091$	$x_{14} = 422247$	$\gcd(373156, n) = 1$
$x_8 = 217757$	$x_{16} = 10727$	$\gcd(-207030, n) = 1$
$x_9 = 292902$	$x_{18} = 367343$	$\gcd(74441, n) = 1$
$x_{10} = 56788$	$x_{20} = 333429$	$\gcd(276641, n) = 1$
$x_{11} = 129591$	$x_{22} = 394807$	$\gcd(265216, n) = 1$
$x_{12} = 617749$	$x_{24} = 371192$	$\gcd(-246557, n) = 1$
$x_{13} = 582329$	$x_{26} = 75538$	$\gcd(-506791, n) = 1$
$x_{14} = 422247$	$x_{28} = 577172$	$\gcd(154925, n) = 1$
$x_{15} = 326280$	$x_{30} = 628251$	$\gcd(301971, n) = 191$

Tablica 2.1: Faktorizacija broja 655703 pomoću Floydove varijante (Primjer 2.1.2)

Vjerojatnost (2.5) će biti približno $1/2$ za $k \approx 1.177\sqrt{p}$. Zaista,

$$\begin{aligned} e^{-k^2/2p} &= 1/2 \quad \Big/ \ln \\ \frac{k^2}{2p} &= \ln 2 \\ k &= \sqrt{2p \ln 2} \\ k &\approx 1.177\sqrt{p}. \end{aligned}$$

Analogno, vjerojatnost će biti približno 0.99 za $k \approx 3.035\sqrt{p}$. Dakle, očekivan broj koraka je puno manji od p, što je pomalo iznenađujuć rezultat. Pod pretpostavkom da je niz (x_i) slučajan, možemo zaključiti da je složenost algoritma $O(\sqrt{p})$. Znamo da n ima prosti faktor $\leq \sqrt{n}$, tj. $p \leq \sqrt{n}$ pa je složenost u najgorem slučaju $O(\sqrt[4]{n})$. Vidimo da složenost ovisi o najmanjem prostom faktoru od n, stoga ovu metodu možemo svrstati u specijalne metode za faktorizaciju. Još jednu važnu činjenicu koju moramo primijetiti je to da algoritam 1 ne mora uvijek dati rješenje. Naime, može se dogoditi da je $\gcd(x_{2i} - x_i, n) = n$. Tada moramo ponoviti cijeli postupak koristeći neku drugu funkciju f za generiranje pseudoslučajnih brojeva.

2.2 Računanje najvećeg zajedničkog djelitelja

Svaki korak algoritma 1 se sastoji od 3 računanja vrijednosti funkcije f i jednog računanja najvećeg zajedničkog djelitelja, no računanje najvećeg zajedničkog djelitelja nas najviše košta. Nauštrb jednog modularnog množenja, možemo odgađati računanje najvećeg zajedničkog djelitelja te ga obavljati nakon nekog određenog broja koraka. U svakom koraku akumuliramo produkt

$$\prod_{j=1}^{i}(x_{2j} - x_j) \pmod{n}, \tag{2.6}$$

a Euklidov algoritam primjenjujemo samo povremeno. Ako su svi $x_{2j} - x_j$ relativno prosti s n, onda će i gcd produkta i n također biti 1. S druge strane, čim je jedan $\gcd(x_{2j} - x_j, n) > 1$, onda će i gcd produkta i n biti veći od 1. Već smo spomenuli mogućnost da algoritam ne uspije pronaći prosti faktor od n, a sada smo ovom modifikacijom tu vjerojatnost dodatno povećali. Naime, ako n ima više od 2 prosta faktora, može se dogoditi da se u produktu akumulira više od jednog pa nam najveći zajednički djelitelj produkta (2.6) i n neće biti traženi prosti faktor od n. U tom slučaju treba pamtiti prethodne vrijednosti produkta te x_k i x_{2k}, gdje je k korak u kojem smo posljednji put računali gcd. Sada treba ponoviti postupak od tog koraka nadalje, ali s češćim računanjem najvećeg zajedničkog djelitelja. Ako niti ovako ne uspijemo dobiti rješenje, cijeli algoritam treba ponoviti s nekom drugom funkcijom f.

2.3 Brentova varijanta Pollardove ρ metode

Richard Brent je 1980. godine objavio varijantu Pollardove ρ metode koja dodatno smanjuje broj računanja najvećeg zajedničkog djelitelja. Metoda umjesto Floydovog algoritma za pronalaženje ciklusa koristi Brentovu metodu za pronalaženje ciklusa. Neka je τ najmanji period niza $(x_i) \pmod{p}$, gdje je p prosti faktor od n kojeg tražimo. Računamo niz $(x_i) \pmod{n}$ dok ne naiđemo na i oblika $2^k - 1$ za $k = 0, 1, 2, \ldots$. Prvi takav i je 0 za $k = 0$. Sada izračunamo $d = \gcd(x_1 - x_0, n)$ te na taj način provjeravamo je li $\tau = 1$. Ako smo dobili $d > 1$, onda smo gotovi i otkrili smo da je period jednak 1. U suprotnom računamo $d = \gcd(x_3 - x_1, n)$. Ako smo sada dobili da je $d > 1$, period je 2. Inače, računamo $\gcd(x_j - x_3, n)$ za $j = 6, 7$ te provjeravamo je li $\tau = 3, 4$, itd. Općenitije, računamo $\gcd(x_j - x_{2^k-1}, n)$ za $k \in \mathbb{N}$ i $3 \cdot 2^{k-1} \leq j \leq 2^{k+1} - 1$. U tablici 2.2 možemo vidjeti nabrojano nekoliko početnih vrijednosti koje se računaju u Brentovoj varijanti. Za određeni $k \in \mathbb{N}$ zapravo tražimo vrijednost perioda τ u rasponu $2^{k-1} + 1 \leq \tau \leq 2^k$. Iz algoritma 2 vidimo da posljednju vrijednost x_j u petlji koristimo dalje za računanje najvećeg zajedničkog djelitelja kao $x_{2^{k+1}-1}$. Jer smo do tada već istražili sve mogućnosti za $\tau \leq 2^k$, onda je dovoljno računati gcd od $j = (2^{k+1} - 1) + 2^k + 1 = 2^{k+1} + 2^k = 3 \cdot 2^k$.

x_0, x_1	$\gcd(x_1 - x_0, n)$		
x_2, x_3	$\gcd(x_3 - x_1, n)$		
x_4, x_5, x_6, x_7	$\gcd(x_6 - x_3, n)$		$\gcd(x_7 - x_3, n)$
$x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}$	$\gcd(x_{12} - x_7, n)$	$\ldots$	$\gcd(x_{15} - x_7, n)$
	$\vdots$		

Tablica 2.2: Vrijednosti u Brentovoj varijanti

Primjer 2.3.1. *Faktorizirajmo broj* $n = 655703$ *iz primjera 2.1.2 Brentovom varijantom Pollardove* ρ *metode. Kao i u spomenutom primjeru, za generiranje niza pseudoslučajnih brojeva uzmimo funkciju* $f(x) = x^2 + 1 \pmod{n}$ *i početnu vrijednost* $x_0 = 2$*. Račun možemo vidjeti u tablici 2.3. Ovaj put smo dobili prosti faktor* 3433 *te faktorizaciju* $n = 191 \times 3433$*. Vidimo da smo sada morali* 12 *puta računati najveći zajednički djelitelj brojeva, dok smo u primjeru 2.1.2 računali* 15 *puta. Dakle, Brentova varijanta se za ovaj primjer pokazala bržom od Floydove varijante Pollardove* ρ *metode.*

Algoritam 2 Brentova varijanta Pollardove ρ metode

Ulaz: funkcija f, n i $x_0 \in \mathbb{Z}_n$

$x \leftarrow x_0$
$y \leftarrow f(x_0)$
$t \leftarrow 1$ ▷ u t spremamo vrijednosti 2^{k-1}
$d \leftarrow \gcd(y - x, n)$
while $d = 1$ **do**
  $x \leftarrow y$ ▷ u x spremamo vrijednosti x_{2^k-1}
  for $i = 1, 2, \ldots, t$ **do** ▷ računamo elemente niza bez gcd
    $y \leftarrow f(y)$
  end for
  for $i = 1, 2, \ldots, , t$ **do** ▷ računamo elemente niza sa gcd
    $y \leftarrow f(y)$
    $d \leftarrow \gcd(y - x, n)$
    if $d > 1$ **then break**
    end if
  end for
  $t \leftarrow 2t$
end while

Izlaz: d

Jasno je da metoda radi za niz $(x_i) \pmod{p}$ koji je periodičan od samog početka. No

k	2^{k-1}	$2^k - 1$	x_{2^k-1}	j	x_j	$\gcd(x_j - x_{2^k-1}, n)$
0	0	0	2	1	5	1
1	1	1	5	2	26	/
				3	677	1
2	2	3	677	4	458330	/
				5	130197	/
				6	24854	1
				7	49091	1
3	4	7	49091	8	217757	/
				9	292902	/
				10	56788	/
				11	129591	/
				12	617749	1
				13	582329	1
				14	422247	1
				15	326280	1
4	8	15	326280	16	10727	/
				17	320505	/
				18	367343	/
				19	480765	/
				20	333429	/
				21	454392	/
				22	394807	/
				23	161496	/
				24	371192	1
				25	629475	1
				26	75538	1
				27	61939	3433

Tablica 2.3: Faktorizacija broja 655703 pomoću Brentove varijante (Primjer 2.3.1)

ako imamo početni aperiodični dio duljine a, onda početne vrijednosti od τ provjeravamo za dio niza koji nije periodičan. Naime, vrijednost $\gcd(x_j - x_{2^k-1}, n)$ će biti različita od 1 ako i samo ako je $x_j \equiv x_{2^k-1} \pmod{p}$. Kada bi vrijedilo $x_j \equiv x_{2^k-1} \pmod{p}$ za neki k takav da je $2^k - 1 < a$, onda bi niz bio periodičan modulo p od mjesta $2^k - 1$ pa nadalje, što je u kontradikciji s tim da je aperiodični dio niza duljine a. Dakle, u slučaju da je $\tau < a$, zapravo nećemo pronaći τ jer ćemo ga tražiti u aperiodičnom dijelu niza. No ovo nam neće predstavljati problem jer je dovoljno pronaći neki višekratnik od τ nakon što niz postane periodičan. Prednost Brentove metode nad Floydovom je ta što se manji broj puta računa

najveći zajednički djelitelj brojeva, ali i ne trebamo dvaput računati vrijednosti od x_i. Zbog toga bi Brentova varijanta trebala biti oko 25% brža. Također, i na ovu varijantu možemo primijeniti način računanja najvećeg zajedničkog djelitelja koji smo vidjeli u potpoglavlju 2.2.

Najveći uspjeh Pollardove ρ metode, odnosno modifikacije Brentove varijante, bio je faktorizacija osmog Fermatovog broja $F_8 = 2^{2^8} + 1$. Uspješno je pronađen prosti faktor 1238926361552897. Drugi prosti faktor od F_8 ima 63 znamenke. Vidimo da je ova metoda bila dobar izbor jer je faktor 1238926361552897 puno manji od drugog faktora.

Poglavlje 3

Pollardova ρ metoda za diskretne logaritme

Pollard je također izumio i ρ metodu za izračun diskretnih logaritama analognu ρ metodi za faktorizaciju velikih brojeva. Opišimo prvo općeniti problem diskretnog logaritma.

Definicija 3.0.1 (Problem diskretnog logaritma)**.** *Neka je G ciklička grupa reda n i neka je g neki njezin generator. Tada se svaki element h te grupe može prikazati kao $h = g^k$ za neki cijeli broj k. Cijeli broj k zovemo diskretni logaritam od h po bazi g. Problem diskretnog logaritma se bavi problemom izračuna broja k iz G, g i h.*

Mi ćemo se baviti problemom diskretnog logaritma za grupu $\mathbb{Z}_p^*$ reda p, gdje je p prosti broj veći od 3. Dakle, elementi te grupe su cijeli brojevi iz skupa $\{1, 2, \ldots, p-1\}$.

Napomena 3.0.2. *U teoriji brojeva se za diskretni logaritam uglavnom koristi naziv index kako ne bi došlo do zabune između običnih logaritama i diskretnih logaritama.*

Analogno kao i u Pollardovoj ρ metodi za faktorizaciju velikih prirodnih brojeva, i ovdje ćemo generirati niz (x_i) pseudoslučajnih brojeva pomoću nekog preslikavanja $F\colon \mathbb{Z}_p^* \to \mathbb{Z}_p^*$. Niz definiramo rekurzivno sa $F(x_i) = x_{i+1}$ i nekom inicijalnom vrijednošću $x_0 \in \mathbb{Z}_p^*$. Niz $x_0, x_1, x_2, \ldots$ predstavlja slučajnu šetnju u grupi $\mathbb{Z}_p^*$. Slučajna šetnja je neki naizgled slučajni proces koji opisuje put koji se sastoji od niza slučajnih koraka na nekom matematičkom prostoru, u ovom slučaju grupi $\mathbb{Z}_p^*$. Jer je $\mathbb{Z}_p^*$ konačna grupa reda p, nakon nekog vremena će doći do ponavljanja u nizu. Taj događaj se zove *kolizija*.

Podijelimo grupu $\mathbb{Z}_p^*$ na 3 dijela S_1, S_2 i S_3 približno jednake veličine tako da vrijedi $\mathbb{Z}_p^* = S_1 \cup S_2 \cup S_3$. Neka je $x_0 = 1$ početna vrijednost niza. Pollard je preslikavanje

$F\colon \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ definirao kao:

$$x_{i+1} = F(x_i) = \begin{cases} h \cdot x_i \pmod p, & x_i \in S_1 \\ x_i^2 \pmod p, & x_i \in S_2 \\ g \cdot x_i \pmod p, & x_i \in S_3 \end{cases}. \tag{3.1}$$

Vrijednosti funkcije F, odnosno vrijednost elementa x_{i+1}, ovise o tome u kojem od skupova S_1, S_2 i S_3 se nalazi element x_i pa niz (x_i) možemo smatrati "slučajnim". Grupu $\mathbb{Z}_p^*$ možemo podijeliti na primjer na trećine, odnosno na skupove $S_1 = \{x \in \mathbb{Z}_p^* \colon 0 < x < \frac{1}{3}p\}$, $S_2 = \{x \in \mathbb{Z}_p^* \colon \frac{1}{3}p < x < \frac{2}{3}p\}$ i $S_3 = \{x \in \mathbb{Z}_p^* \colon \frac{2}{3}p < x < p\}$. Primijetimo da su elementi niza oblika $x_i = h^{a_i} g^{b_i} \pmod p$. Za inicijalne vrijednosti nizova (a_i) i (b_i) uzmimo $a_0 = 0$ i $b_0 = 0$. Niz (a_i) računamo kao:

$$a_{i+1} = \begin{cases} a_i + 1 \pmod{(p-1)}, & x_i \in S_1 \\ 2a_i \pmod{(p-1)}, & x_i \in S_2 \\ a_i \pmod{(p-1)}, & x_i \in S_3 \end{cases}. \tag{3.2}$$

Slično, niz (b_i) računamo kao:

$$b_{i+1} = \begin{cases} b_i \pmod{(p-1)}, & x_i \in S_1 \\ 2b_i \pmod{(p-1)}, & x_i \in S_2 \\ b_i + 1 \pmod{(p-1)}, & x_i \in S_3 \end{cases}. \tag{3.3}$$

Čim za neke $i, j \in \mathbb{N}$, $i < j$ vrijedi $x_i = x_j$, došli smo do kolizije. Dakle, vrijedi $h^{a_i} g^{b_i} = h^{a_j} g^{b_j}$. Zbog paradoksa rođendana, očekujemo da će doći do kolizije nakon $O(\sqrt{p-1})$ iteracija. Jer je g generator grupe Z_p^*, onda vrijedi $h \equiv g^k \pmod p$ pa imamo $(g^k)^{a_i} \cdot g^{b_i} = (g^k)^{a_j} \cdot g^{b_j}$. Primijetimo da je g generator grupe $\mathbb{Z}_p^*$ ako i samo ako je g primitivni korijen modulo p. Dvije potencije su kongruentne modulo n ako i samo ako su njihovi eksponenti kongruentni modulo red baze. Jer je g primitivni korijen modulo p, onda znamo da je red od g jednak $p-1$. Iz toga slijedi

$$a_i \cdot k + b_i \equiv a_j \cdot k + b_j \pmod{(p-1)}.$$

Nadalje, imamo

$$(a_i - a_j)k \equiv b_j - b_i \pmod{(p-1)}. \tag{3.4}$$

Teorem 3.0.3. *Neka su a i m prirodni brojevi te neka je b cijeli broj. Kongruencija $ax \equiv b \pmod m$ ima rješenja ako i samo ako $d = \gcd(a, m)$ dijeli b. Ako je ovaj uvjet zadovoljen, onda ta kongruencija ima točno d rješenja modulo m.*

Dakle, ako vrijedi da $d = \gcd(a_i - a_j, p-1)$ dijeli $b_j - b_i$, onda možemo riješiti kongruenciju (3.4), odnosno izračunati diskretni logaritam k. Ako je $a_i - a_j$ relativno prost s $p-1$, onda postoje brojevi $u, v \in \mathbb{Z}$ takvi da vrijedi

$$(a_i - a_j)u + (p-1)v = 1.$$

Brojeve u i v možemo pronaći pomoću Euklidovog algoritma. Sada vrijedi

$$(a_i - a_j)u \equiv 1 \pmod{(p-1)},$$

pa broj k možemo izračunati kao

$$k \equiv u(b_j - b_i) \pmod{(p-1)}.$$

Nadalje, ako je najveći zajednički djelitelj d od $a_i - a_j$ i $p-1$ veći od 1 i d dijeli $b_j - b_i$, onda rješavamo kongruenciju

$$\frac{a_i - a_j}{d} k \equiv \frac{b_j - b_i}{d} \pmod{(p-1)/d} \tag{3.5}$$

umjesto početne kongruencije. Jer su brojevi $\frac{a_i - a_j}{d}$ i $\frac{p-1}{d}$ relativno prosti, onda kongruencija (3.5) ima točno 1 rješenje i možemo ju riješiti kao što je prethodno opisano. Kongruencija (3.4) ima d rješenja, a mi ćemo rješavanjem (3.5) dobiti jedno od njih. Neka je k' jedno rješenje kongruencije (3.4). Ostala rješenja su dana sa $k = k' + n\frac{p-1}{d}$ za $n \in \mathbb{Z}$, a sva međusobno neekvivalentna rješenja dobijemo za $n = 0, 1, \ldots, d-1$.

Kako bismo pronašli ciklus u nizu (x_i) koristit ćemo se Floydovom metodom za pronalaženje ciklusa. Isto kao i kod Pollardove ρ metode za faktorizaciju, mogli smo koristiti i Brentovu metodu za pronalaženje ciklusa. Međutim, koncept za računanje najvećeg zajedničkog djelitelja iz odjeljka 2.2 se ne može direktno primijeniti na Pollardovu ρ metodu za diskretne logaritme.

U svakom i-tom koraku algoritma računamo x_i, a_i i b_i te paralelno s tim računamo i x_{2i}, a_{2i} i b_{2i}. Zapravo računamo nizove (x_i) i (x_{2i}) te provjeravamo jesu li za neki i te dvije vrijednosti jednake. Ako dobijemo da vrijedi $x_i = x_{2i}$, znači da smo došli do kolizije, odnosno pronašli smo ciklus u nizu (x_i). Za račun elemenata nizova (x_i) i (x_{2i}) su nam potrebne 3 operacije modularnog množenja u svakom koraku algoritma.

Primjer 3.0.4. *Neka je $p = 9239$ i uzmimo grupu Z_p^*. Element $g = 19$ te grupe je primitivni korijen modulo p. Izračunajmo diskretni logaritam od $h = 107$ po g koristeći Pollardovu ρ metodu za račun diskretnih logaritama. Dakle, trebamo pronaći broj $0 < k < p-1$ takav da vrijedi $h \equiv g^k \pmod{p}$. Koristit ćemo Floydovu metodu za pronalaženje ciklusa.*

Generiramo niz (x_i) *pomoću* (3.1) *te paralelno računamo i niz* (x_{2i})*. Za skupove* S_1*,* S_2 *i* S_3 *ćemo uzeti*

$$S_1 = \{x \in \mathbb{Z}_p^* \colon 0 < x < \frac{1}{3}p \approx 3079.67\},$$
$$S_2 = \{x \in \mathbb{Z}_p^* \colon \frac{1}{3}p \approx 3079.67 < x < \frac{2}{3}p \approx 6159.34\} \; i$$
$$S_3 = \{x \in \mathbb{Z}_p^* \colon \frac{2}{3}p \approx 6159.34 < x < p\}$$

i	$x_i = h^{a_i} g^{b_i}$	a_i	b_i	$2i$	$x_{2i} = h^{a_{2i}} g^{b_{2i}}$	a_{2i}	b_{2i}
1	107	1	0	2	2210	2	0
2	2210	2	0	4	1973	6	0
3	5495	3	0	6	1383	7	1
4	1973	6	0	8	7560	9	1
5	7853	7	0	10	7190	18	4
6	1383	7	1	12	8670	18	6
7	157	8	1	14	7088	18	8
8	7560	9	1	16	2546	36	18
9	5055	9	2	18	344	74	36
10	7190	18	4	20	6427	75	37
11	7264	18	5	22	2145	76	38
12	8670	18	6	24	9216	77	39
13	7667	18	7	26	936	77	41
14	7088	18	8	28	8893	78	42
15	5326	18	9	30	7985	79	43
16	2546	36	18	32	6399	158	88
17	4491	37	18	34	655	159	89
18	344	74	36	36	2114	320	178
19	9091	75	36	38	8638	642	356
20	6427	75	37	40	4775	642	358
21	2006	75	38	42	4404	1284	717
22	2145	76	38	44	5454	2569	1434
23	7779	77	38	46	7074	1038	5736
24	9216	77	39	48	2331	2076	2236
25	8802	77	40	50	8555	2077	2237

Tablica 3.1: Račun diskretnog logaritma od 107 po bazi 19 modulo 9239 pomoću Pollardove ρ metode (Primjer 3.0.4) - 1. dio

i	$x_i = h^{a_i} g^{b_i}$	a_i	b_i	$2i$	$x_{2i} = h^{a_{2i}} g^{b_{2i}}$	a_{2i}	b_{2i}
26	936	77	41	52	7096	4154	4476
27	7762	78	41	54	212	8308	8954
28	8893	78	42	56	6990	7380	8670
29	2665	78	43	58	7074	5522	8104
30	7985	79	43	60	2331	1806	6972
31	3891	79	44	62	8555	1807	6973
32	6399	158	88	64	7096	3614	4710
33	1474	158	89	66	212	7228	184
34	655	159	89	68	6990	5220	368
35	5412	160	89	70	7074	1202	738
36	2114	320	178	72	2331	2404	1478
37	4462	321	178	74	8555	2405	1479
38	8638	642	356	76	7096	4810	2960
39	7059	642	357	78	212	382	5922
40	4775	642	358	80	6990	766	2606
41	8012	1284	716	82	7074	1532	5214
42	4404	1284	717	84	2331	3064	1192
43	2555	2568	1434	86	8555	3065	1193
44	5454	2569	1434	88	7096	6130	2388
45	5775	5138	2868	90	212	3022	4778
46	7074	1038	5736	92	6990	6046	318
47	5060	1038	5737	94	7074	2854	638
48	2331	2076	2236	96	2331	5708	1278

Tablica 3.2: Račun diskretnog logaritma od 107 po bazi 19 modulo 9239 pomoću Pollardove ρ metode (Primjer 3.0.4) - 2. dio

U tablicama 3.1 i 3.2 možemo vidjeti izračunate vrijednosti za nizove (x_i), (a_i) *i* (b_i). *Nizove* (a_i) *i* (b_i) *računamo pomoću* (3.2) *i* (3.3). *Paralelno smo računali i vrijednosti od* (x_{2i}), (a_{2i}) *te* (b_{2i}). *Za* $i = 48$ *smo došli do kolizije, tj. dobili smo da vrijedi* $x_{48} = x_{96}$. *Dakle, niz* (x_i) *je periodičan od elementa* x_{48}, *možda i ranije. Sada imamo*

$$x_{48} = h^{a_{48}} \cdot g^{b_{48}} \equiv h^{a_{96}} \cdot g^{b_{96}} = x_{96} \pmod{p}.$$

Odnosno, kada uvrstimo $h \equiv g^k \pmod{p}$

$$g^{k \cdot a_{48}} \cdot g^{b_{48}} \equiv g^{k \cdot a_{96}} \cdot g^{b_{96}} \pmod{p}.$$

Sada trebamo riješiti kongruenciju

$$k(a_{48} - a_{96}) \equiv b_{96} - b_{48} \pmod{(p-1)}, \text{ tj.}$$
$$k(2076 - 5708) \equiv 1278 - 2236 \pmod{9238}$$
$$3632k \equiv 958 \pmod{9238}.$$

Jer brojevi 3632 *i* 9238 *nisu relativno prosti, onda moramo prvo riješiti kongruenciju*

$$1816k \equiv 479 \pmod{4619}.$$

Sada postoje brojevi $u, v \in \mathbb{Z}$ *takvi da vrijedi*

$$1816u + 4619v = 1.$$

Provedimo Euklidov algoritam

$$\begin{aligned} 4619 &= 1816 \cdot 2 + 987 \\ 1816 &= 987 \cdot 1 + 829 \\ 987 &= 829 \cdot 1 + 158 \\ 829 &= 158 \cdot 5 + 39 \\ 158 &= 39 \cdot 4 + 2 \\ 39 &= 2 \cdot 19 + 1 \\ 2 &= 1 \cdot 2 \end{aligned}$$

i	-1	0	1	2	3	4	5	6
q_i			2	1	1	5	4	19
x_i	1	0	1	-1	2	-11	46	-885
y_i	0	1	-2	3	-5	28	-117	2251

Tablica 3.3: Prošireni Euklidov algoritam

Dakle, imamo

$$4619 \cdot (-885) + 1816 \cdot 2251 = 1.$$

Sada vrijedi

$$1816 \cdot 2251 \equiv 1 \pmod{4619}$$

pa za k imamo

$$k \equiv 479 \cdot 2251 \equiv 2002 \pmod{4619}.$$

Sva neekvivalentna rješenja početne kongruencije dobijemo za $n = 0, 1$. *Imamo* $k_0 = k = 2002$ *i* $k_1 = k + 4619 = 6621$. *Dakle,* k_0 *i* k_1 *su kandidati za diskretni logaritam od* 107 *po* 19. *Za* k_0 *imamo*

$$g^{k_0} = 19^{2002} \equiv 9132 \not\equiv 107 \pmod{9239}.$$

Za k_1 *imamo*

$$g^{k_1} = 19^{6621} \equiv 107 \pmod{9239}.$$

Dakle, $k_1 = 6621$ *je diskretni logaritam od* 107 *po* 19 *modulo* 9239.

Poglavlje 4

Pollardova $p-1$ metoda za faktorizaciju

Pollardova $p-1$ metoda također spada u Monte Carlo klasu algoritama. Metoda se temelji na Malom Fermatovom teoremu.

Neka je $n \in \mathbb{N}$ broj kojeg želimo faktorizirati i neka je $p \in \mathbb{N}$ neki njegov nepoznati prosti faktor. U poglavlju 1 smo rekli da za svaki višekratnik m od $p-1$ vrijedi $a^m \equiv 1 \pmod{p}$. Ako uspijemo pronaći neki višekratnik od $p-1$, onda možemo pomoću Euklidovog algoritma za pronalaženje najvećeg zajedničkog djelitelja izračunati faktor p. Dakle, ako za neki $m \in \mathbb{N}$ dobijemo da je $\gcd(a^m - 1, n)$ netrivijalan, onda smo pronašli neki faktor p od n. Sada se opet postavlja slično pitanje, kako pronaći višekratnik broja $p-1$ ako nam je p nepoznat? Mogli bismo provjeravati redom je li najveći zajednički djelitelj brojeva $a^m - 1$ i n veći od 1 za sve $m < p-1$, no bilo bi brže provjeravati je li n djeljiv prostim brojevima dok ne dođemo do p. Kada bismo znali za brojeve kojeg oblika je dovoljno računati najveći zajednički djelitelj, onda bismo bili u prednosti nad metodom probnog dijeljenja. Ideja Pollardove $p-1$ metode leži u tome da možemo jednostavno pronaći višekratnik od $p-1$ u slučaju kada se faktorizacija od $p-1$ sastoji samo od malih prostih faktora.

Definicija 4.0.1. *Za prirodan broj m kažemo da je B-gladak ako za svaki prosti faktor p od m vrijedi $p \leq B$.*

Problem se sada svodi na pronalaženje svih brojeva s malim prostim faktorima i njihovih višekratnika. Mogli bismo promatrati brojeve $k!$, ali tada već za $k = 8$ imamo $8! = 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 2^7 \cdot 3^2 \cdot 5 \cdot 7$ pa bismo radi gomilanja potencija malih prostih brojeva mogli preskočiti $p-1$ koji je djeljiv nekom malom potencijom prostog broja. Dakle, moramo dodatno pretpostaviti da su i sve potencije prostih brojeva $\leq B$. Bolji pristup bi bio promatrati umnožak prostih brojeva $\leq k$ u koji svaki put kada prođemo neku potenciju prostog broja akumuliramo samo prosti broj p umjesto njegove potencije. Na taj način zapravo računamo najmanji zajednički višekratnik brojeva do k. Neka je $M(k)$ najmanji zajednički

višekratnik brojeva $\leq k$. Računamo niz $M(1) = 1, M(2) = 2, M(3) = 6, M(4) = 12, \ldots$. Niz možemo definirati rekurzivno sa

$$M(k+1) := \begin{cases} pM(k), & \text{ako } k+1 = p^{\alpha},\ p \text{ prost} \\ M(k), & \text{inače} \end{cases}. \tag{4.1}$$

Za bazu a najčešće možemo uzeti broj 2. Ako je n neparan broj, onda sigurno nije djeljiv s

Algoritam 3 Pollardova $p-1$ metoda

Ulaz: broj n, granica B

Odaberi bazu a
$b \leftarrow 2$ ▷ u b spremamo vrijednosti $1, 2, \ldots, B$
$d \leftarrow 1$
while $b \leq B$ **do**
  $m \leftarrow M(b)$ ▷ najmanji zajednički višekratnik brojeva $1, 2, \ldots, b$
  $d \leftarrow \gcd(a^m - 1, n)$
  if $d > 1$ **then break**
  end if
  $b \leftarrow b + 1$
end while

Izlaz: d

2 pa mora vrijediti $\gcd(a, p) = 1$ za svaki p prosti faktor od n. Ako je n paran, onda odmah znamo da je 2 jedan prosti faktor od n pa niti ne trebamo provoditi algoritme za faktorizaciju kako bismo ga pronašli. Primijetimo da smo u Algoritmu 3 mogli samo računati najveći zajednički djelitelj tek nakon što izračunamo $M(B)$, no tada je veća vjerojatnost da nam algoritam neće dati rješenje, tj. da ćemo kao rezultat dobiti da je najveći zajednički djelitelj upravo n. Postoje 2 načina kako metoda može biti bezuspješna. Prvi slučaj je kada algoritam završi, a nismo pronašli niti jedan najveći zajednički djelitelj veći od 1. Taj slučaj nam ukazuje na to da brojevi $p-1$, za p proste faktore od n, nisu B-glatki. Tada možemo pokušati ponovno pokrenuti algoritam, ali s većom granicom B.

Primjer 4.0.2. *Neka je $n = 13493$ i $B = 10$. Jer je n neparan, možemo odabrati $a = 2$. Pokušajmo faktorizirati broj n pomoću Pollardove $p-1$ metode. Račun možemo vidjeti u tablici 4.1. Vidimo da za $B = 10$ i $a = 2$ nismo uspjeli pronaći niti jedan prosti faktor p od n. Pogledajmo koliku granicu B bismo trebali uzeti kako bismo uspjeli dobiti da je najveći zajednički djelitelj različit od 1. Dakle, trebali smo uzeti $B \geq 13$ kako bismo uspjeli dobiti prosti faktor $p = 131$ od n. Drugi kofaktor je 103, tj. dobili smo faktorizaciju broja $n = 103 \times 131$. Označimo prosti faktor 131 s p, a 103 s q. Dakle, imamo $n = p \cdot q$.*

b	$M(b)$	$a^{M(b)}$	$\gcd(a^{M(b)}-1, n)$
2	2	4	$\gcd(4-1, n) = 1$
3	6	64	$\gcd(64-1, n) = 1$
4	12	4096	$\gcd(4096-1, n) = 1$
5	60	1748	$\gcd(1748-1, n) = 1$
6	60	1748	$\gcd(1748-1, n) = 1$
7	420	2551	$\gcd(2551-1, n) = 1$
8	840	3975	$\gcd(3975-1, n) = 1$
9	2520	11608	$\gcd(11608-1, n) = 1$
10	2520	11608	$\gcd(11608-1, n) = 1$

Tablica 4.1: Faktorizacija broja 13493 pomoću Pollardove $p-1$ metode s granicom $B = 10$ (Primjer 4.0.2)

b	$M(b)$	$a^{M(b)}$	$\gcd(a^{M(b)}-1, n)$
11	27720	6874	$\gcd(6874-1, n) = 1$
12	27720	6874	$\gcd(6874-1, n) = 1$
13	360360	787	$\gcd(8778-1, n) = 131$

Tablica 4.2: Faktorizacija broja 13493 pomoću Pollardove $p-1$ metode s granicom $B = 13$ (Primjer 4.0.2)

Promotrimo sada faktorizacije brojeva $p-1$ i $q-1$.

$$p - 1 = 130 = 2 \cdot 5 \cdot 13$$
$$q - 1 = 102 = 2 \cdot 3 \cdot 17$$

Vidimo da je $p-1 = 130$ 13-gladak, a $q-1 = 102$ nije. Zbog toga smo za granicu $B = 13$ uspjeli pronaći prosti faktor 131.

Drugi slučaj u kojem algoritam može ne dati rješenje je kada dobijemo da je najveći zajednički djelitelj upravo n. Jasno je da u ovom slučaju povećavanje granice B neće rezultirati uspjehom. Pretpostavimo da broj n ima faktorizaciju $n = p \cdot q$. Jer je $\gcd(a^m - 1, n) = n$, onda vrijedi $n | a^m - 1$. Iz toga zaključujemo da vrijedi i $p | a^m - 1$ te $q | a^m - 1$, odnosno $a^m \equiv 1 \pmod{p}$ i $a^m \equiv 1 \pmod{q}$. Zapišimo m kao $m = j(q-1) + k$, gdje su j i k cijeli brojevi i $0 \leq k < q - 1$. Sada po Malom Fermatovom teoremu imamo

$$a^m = a^{j(q-1)+k} = a^{j(q-1)} \cdot a^k \equiv a^k \pmod{q}.$$

Jer je $a^m \equiv 1 \pmod{q}$, onda je i $a^k \equiv 1 \pmod{q}$. Ovo se može dogoditi iz dva razloga.

1. Ako je $k \neq 0$ može se dogoditi da je $a^k \equiv 1 \pmod{q}$ iako su a i q relativno prosti. Na primjer za $a = 3$, $k = 3$ i $q = 13$ imamo $3^3 = 27 \equiv 1 \pmod{13}$. Ovo se ne može dogoditi ako je a primitivni korijen modulo q. Jer ima jako puno primitivnih korijena modulo q, a čak i ako a nije primitivni korijen, vjerojatnost da se ovo dogodi je jako mala pa bi odabir neke druge baze a trebao riješiti problem i dati rješenje.

2. Ako je $k = 0$, onda za svaki a vrijedi da je $a^k \equiv 1 \pmod{q}$, osim kada je a neki višekratnik od q. Dakle, iz ove situacije se možemo izvući jedino ako za a "pogodimo" neki višekratnik od q, a za to je jako mala vjerojatnost.

Primjer 4.0.3. *Faktorizirajmo broj $n = 44287$ pomoću Pollardove $p-1$ metode. Neka je granica $B = 12$. Za bazu a uzmimo broj 2. Pogledajmo račun u tablici 4.3. Završili*

b	$M(b)$	$a^{M(b)}$	$\gcd(a^{M(b)} - 1, n)$
2	2	4	$\gcd(4-1, n) = 1$
3	6	64	$\gcd(64-1, n) = 1$
4	12	4096	$\gcd(4096-1, n) = 1$
5	60	1295	$\gcd(1295-1, n) = 1$
6	60	1295	$\gcd(1295-1, n) = 1$
7	420	5508	$\gcd(5508-1, n) = 1$
8	840	1469	$\gcd(1469-1, n) = 1$
9	2520	25536	$\gcd(25536-1, n) = 1$
10	2520	25536	$\gcd(25536-1, n) = 1$
11	27720	1	$\gcd(1-1, n) = 44287$

Tablica 4.3: Faktorizacija broja 44287 pomoću Pollardove $p-1$ metode s granicom $B = 12$ i bazom $a = 2$ (Primjer 4.0.3)

smo račun, a nismo dobili niti jedan prosti faktor broja 44287. *U zadnjem retku tablice 4.3 za $b = 11$ smo dobili* $\gcd(0, 44287) = 44287$, *a u retku prije za $b = 10$ imamo* $\gcd(25535, 44287) = 1$. *Dakle, za bazu* 2 *Pollardova $p-1$ metoda nam ne daje rješenje pa trebamo pokušati s nekom drugom bazom a. Ako uzmemo $a = 3$ za $b < 11$ ćemo dobiti da je* $\gcd(a^{M(b)} - 1, n) = 1$, *a za $b = 11$ dobijemo* $\gcd(3^{27720} - 1, 44287) = \gcd(1 - 1, 44287) = 44287$. *Dakle, niti za $a = 3$ nismo uspjeli dobiti neki prosti faktor od n. Isti rezultat ćemo dobiti ako za a uzmemo neki broj manji od* 12. *Tek za $a = 12$ Pollardova $p-1$ metoda daje prosti faktor* 661 *kao rezultat. Pogledajmo postupak u tablici 4.4. Drugi kofaktor je* 67, *dakle dobili smo faktorizaciju* $44287 = 67 \times 661$. *Označimo opet proste faktore od n s p i q, tj. $p = 67$ i $q = 661$. Promotrimo faktorizaciju brojeva $p-1$ i $q-1$.*

$$p - 1 = 66 = 2 \cdot 3 \cdot 11$$
$$q - 1 = 660 = 2^2 \cdot 3 \cdot 5 \cdot 11$$

b	$M(b)$	$a^{M(b)}$	$\gcd(a^{M(b)}-1, n)$
2	2	144	$\gcd(144-1, n) = 1$
3	6	18755	$\gcd(18755-1, n) = 1$
4	12	22671	$\gcd(22671-1, n) = 1$
5	60	11899	$\gcd(11899-1, n) = 661$

Tablica 4.4: Faktorizacija broja 44287 pomoću Pollardove $p-1$ metode s granicom $B = 12$ i bazom $a = 12$ (Primjer 4.0.3)

Vidimo da su oba broja B-glatka za $B = 11$*, naravno i za svaki* $B \geq 11$*. Zbog toga smo dobili da je* $\gcd(a^{M(b)}-1, n) = n$ *čim smo u najmanji zajednički višekratnik brojeva ubacili prosti faktor* 11*. Za* $M(11)$ *vrijedi* $p-1|M(11)$ *i* $q-1|M(11)$ *pa ne možemo otkriti niti jedan od prostih faktora broja n. Primijetimo da smo umjesto* $M(11)$ *uzeli* $M(11)/5 = 27720/5 = 5544$*, onda bismo skoro za svaku bazu a dobili prosti faktor* 67 *jer je broj* 5544 *višekratnik od* $p - 1 = 66$*, ali ne i od* $q - 1 = 660$*.*

4.1 Varijacija velikog prostog broja

Vidjeli smo da ako za izlaz algoritma 3 dobijemo da je $\gcd(a^{M(B)}-1, n) = 1$, onda $p-1$ nije B-gladak, za p prosti faktor od n. Međutim, može biti da je $p - 1 = uv$, gdje je u B-gladak, a v je neki prosti broj veći od B. Tada v zovemo *velikim prostim* djeliteljem od $p - 1$. U ovom slučaju možemo dodati tzv. drugu fazu našem algoritmu, umjesto da samo pokrenemo algoritam ispočetka s većom granicom B, te na taj način dobiti netrivijalni faktor od n.

Neka je $B' > B$ takav da vrijedi $v \leq B'$ i neka su $p_{i+1}, p_{i+2}, \ldots, p_{i'}$ svi prosti brojevi veći od B i manji ili jednaki B'. Kada završi prva faza, tj. kada Algoritam 3 završi, već imamo izračunato $a^{M(B)} \pmod n$. Označimo taj rezultat s c. Jer je u B-gladak, onda znamo da vrijedi $u|M(B)$. U drugoj fazi više ne računamo najmanji zajednički višekratnik brojeva, već računamo redom $c^{p_{i+1}} \pmod n$, $c^{p_{i+2}} \pmod n$, itd. dok ne dobijemo da je najveći zajednički djelitelj veći od 1 ili ne iscrpimo sve proste brojeve $\leq B'$. Mogli smo računati i $c^{p_{i+1}}$, $(c^{p_{i+1}})^{p_{i+2}}$, itd. te na taj način dobiti i v koji je produkt prostih brojeva većih od B i manjih ili jednakih B'. Vidimo da smo na ovaj način puno uštedjeli na vremenu, nego da smo ponovno pokretali algoritam s granicom B' jer za proste brojeve $\leq B$ u obzir uzimamo samo one potencije prostih brojeva koje su $\leq B$, a za proste brojeve između B (neuključivo) i B' (uključivo) uzimamo samo eksponente jednake 1. U praksi se najčešće za granicu B uzima $B \leq 10^6$, a za B' broj $\leq 10^8$. Ako do tada ne uspijemo pronaći neki prosti faktor od n, trebamo pokušati s nekom drugom metodom.

Primjer 4.1.1. *Pokušajmo faktorizirati broj* $n = 980051$ *pomoću Pollardove* $p-1$ *metode. Za granicu* B *uzmimo broj* 20*, a za bazu* a *uzmimo* 2*. Račun vidimo u tablici 4.5. Nismo*

b	$M(b)$	$a^{M(b)}$	$\gcd(a^{M(b)}-1, n)$
2	2	4	$\gcd(4-1, n) = 1$
3	6	64	$\gcd(64-1, n) = 1$
4	12	4096	$\gcd(4096-1, n) = 1$
5	60	720081	$\gcd(720081-1, n) = 1$
6	60	720081	$\gcd(720081-1, n) = 1$
7	420	426549	$\gcd(426549-1, n) = 1$
8	840	521404	$\gcd(521404-1, n) = 1$
9	2520	968344	$\gcd(968344-1, n) = 1$
10	2520	968344	$\gcd(968344-1, n) = 1$
11	27720	747002	$\gcd(747002-1, n) = 1$
12	27720	747002	$\gcd(747002-1, n) = 1$
13	360360	34286	$\gcd(34286-1, n) = 1$
14	360360	34286	$\gcd(34286-1, n) = 1$
15	360360	34286	$\gcd(34286-1, n) = 1$
16	720720	448647	$\gcd(448647-1, n) = 1$
17	12252240	929454	$\gcd(929454-1, n) = 1$
18	12252240	929454	$\gcd(929454-1, n) = 1$
19	232792560	933964	$\gcd(933964-1, n) = 1$
20	232792560	933964	$\gcd(933964-1, n) = 1$

Tablica 4.5: Prva faza Pollardove $p-1$ metode (Primjer 4.1.1)

uspjeli pronaći prosti faktor pomoću prve faze Pollardove $p-1$ *metode pa ćemo provesti i drugu fazu. Za granicu* B' *uzmimo* 100*. U tablici 4.6 vidimo račun za drugu fazu. Uspjeli smo dobiti prosti faktor* 997*. Imamo faktorizaciju* $n = 997 \times 983$*. Označimo* $p = 997$ *i* $q = 983$*. Pogledajmo faktorizaciju od* $p-1$ *i* $q-1$*.*

$$p - 1 = 996 = 2^2 \cdot 3 \cdot 83$$
$$q - 1 = 982 = 2 \cdot 491$$

Vidimo da niti $p-1$ *niti* $q-1$ *nisu* 20*-glatki, stoga nismo uspjeli dobiti nijedan prosti faktor pomoću prve faze Pollardove* $p-1$ *metode. Zapišimo* $p-1$ *kao* $p-1 = uv$*, gdje je* $u = 2^2 \cdot 3$*, a* $v = 83$*. Sada imamo da je* u 20*-gladak, a* v *je prosti broj veći od* 20*. Odnosno,* v *je veliki prosti djelitelj od* $p-1$*. Zbog toga, i jer* $q-1$ *nije* 100*-gladak (ima prosti faktor* 491 *koji je veći od* 100*), smo pomoću druge faze uspjeli faktorizirati broj* 980051*.*

p_i	$a^{M(B)\cdot p_i}$	$\gcd(a^{M(B)\cdot p_i}-1, n)$
23	418131	$\gcd(418131-1, n)=1$
29	32577	$\gcd(32577-1, n)=1$
31	582392	$\gcd(582392-1, n)=1$
37	465931	$\gcd(465931-1, n)=1$
41	877881	$\gcd(877881-1, n)=1$
43	398790	$\gcd(398790-1, n)=1$
47	146237	$\gcd(146237-1, n)=1$
53	219650	$\gcd(219650-1, n)=1$
59	646383	$\gcd(646383-1, n)=1$
61	466783	$\gcd(466783-1, n)=1$
67	863030	$\gcd(863030-1, n)=1$
71	494697	$\gcd(494697-1, n)=1$
73	975870	$\gcd(975870-1, n)=1$
79	50042	$\gcd(50042-1, n)=1$
83	475570	$\gcd(475570-1, n)=997$

Tablica 4.6: Druga faza Pollardove $p-1$ metode (Primjer 4.1.1)

Unaprjeđenje druge faze

Kao i u varijaciji velikog prostog broja, i ovdje nam treba posljednji rezultat prve faze. Označimo ga s $Q = a^{M(B)} \pmod n$. Nadalje, neka su $p_{i+1}, p_{i+2}, \ldots, p_{i'}$ prosti brojevi veći od granice B i manji ili jednaki od B'. Računamo tablicu s vrijednostima $Q^{p_{i+2}-p_{i+1}} \pmod n$, $Q^{p_{i+3}-p_{i+2}} \pmod n, \ldots, Q^{p_{i'}-p_{i'-1}} \pmod n$. Jer su razlike dva uzastopna prosta broja male i malobrojne, ova tablica će biti mala i možemo ju jednostavno izračunati. Sada računamo $a^{M(B)\cdot p_{i+1}}, a^{M(B)\cdot p_{i+2}}, \ldots, a^{M(B)\cdot p_{i'}}$ tako da sukcesivno množimo svaki Q uzastopnim razlikama prostih brojeva iz izračunate tablice. Na ovaj način smo zamijenili skupocjeno modularno potenciranje s jednostavnijim modularnim množenjem. Inicijaliziramo Q sa $Q = Q^{p_{i+1}} \pmod n$ i $P = 1$. Sada iteriramo po razlikama. Prva razlika je $d_1 = Q^{p_{i+2}-p_{i+1}} \pmod n$. Računamo $Q = Q \cdot Q^{d_1} \pmod n$ te $P = P \cdot (Q-1)$ itd. U svakom koraku računamo i $\gcd(P, n)$ dok ne dobijemo neki najveći zajednički djelitelj veći od 1. Ako smo dobili da je $\gcd > 1$, onda smo pronašli neki faktor od n.

Primjer 4.1.2. *Faktorizirajmo broj* $n = 980051$ *iz primjera 4.1.1 koristeći unaprijeđenu drugu fazu Pollardove* $p-1$ *metode. Uzmimo iste granice* $B = 20$ *i* $B' = 100$ *kao i u primjeru 4.1.1. Provodimo samo drugu fazu, prvu možemo vidjeti u tablici 4.5. Iz prve faze imamo* $Q = 933964$*. Proste brojeve* $> B$ *i* $\leq B'$ *i razlike uzastopnih prostih brojeva možemo vidjeti u tablici 4.7.*

Izračunajmo vrijednosti Q^d*, za* $d = 2, 4, 6, 8$ *i zapišimo u tablicu 4.8. Inicijalizi-*

p_i	$p_i - p_{i-1}$
23	
29	$29-23=6$
31	$31-29=2$
37	$37-31=6$
41	$41-37=4$
43	$43-41=2$
47	$47-43=4$
53	$53-47=6$
59	$59-53=6$
61	$61-59=2$
67	$67-61=6$
71	$71-67=4$
73	$73-71=2$
79	$79-73=6$
83	$83-79=4$
89	$89-83=6$
97	$97-89=8$

Tablica 4.7: Razlike uzastopnih prostih brojeva > 20 i ≤ 100

d	$Q^d \pmod n$
2	$933964^2 = 241052$
4	$933964^4 = 803016$
6	$933964^6 = 699924$
8	$933964^8 = 340296$

Tablica 4.8: Vrijednosti od $Q^d \pmod n$

ramo $Q = 933964^{23} \equiv 418131 \pmod n$ *i* $P = 1$. *Sada možemo jednostavno provesti unaprijeđenu drugu fazu koristeći tablicu 4.8. Račun vidimo u tablici 4.9. Uspjeli smo faktorizirati broj* $n = 980051$ *i dobili smo faktorizaciju* $n = 997 \times 983$.

Možemo još dodatno ubrzati drugu fazu Pollardove $p-1$ metode. Kao što smo već rekli, računanje najvećeg zajedničkog djelitelja je zahtjevna operacija, a mi ju obavljamo u svakom koraku algoritma. Kako bismo uštedjeli na vremenu, možemo odgađati računanje najvećeg zajedničkog djelitelja te ga računati periodično, odnosno nakon nekog određenog broja koraka. Ako na takav način dobijemo da je gcd = 1, onda znamo da bi i svaki najveći zajednički djelitelj između isto bio jednak 1 pa ih ne treba računati. Ako dobijemo da je gcd = n, onda smo u međuvremenu naišli na sve proste faktore od n te se trebamo vratiti

d	$Q = Q \cdot Q^d$	$P = P \cdot (Q-1)$	$\gcd(P,n)$
6	32577	32576	1
2	582392	141958	1
6	465931	809052	1
4	877881	578704	1
2	398790	340078	1
4	146237	918515	1
6	219650	542528	1
6	646383	444978	1
2	466783	612111	1
6	863030	494097	1
4	494697	149959	1
2	975870	104102	1
6	50042	397117	1
4	475570	706873	997

Tablica 4.9: Unaprijeđena druga faza Pollardove $p-1$ metode (Primjer 4.1.2)

na zadnju izračunatu vrijednost gcd te ponoviti algoritam računajući gcd u svakom koraku.

Pomoću Pollardove $p-1$ metode uspješno su pronađeni prosti faktori 12. Fermatovog broja te Mersenneovih brojeva $2^{257}-1$ i $2^{977}-1$. Za pronalazak 16-znamenkastog prostog faktora $p = 1256132134125569$ Fermatovog broja $F_{12} = 2^{2^{12}}-1$ dovoljna je bila samo prva faza Pollardove $p-1$ metode, dok je za pronalazak 31-znamenkastog faktora $p = 4985899058078884305401269007884 1$ Mersenneovog broja $2^{977}-1$ korištena i druga faza. Bitno je naglasiti da se "prosti" faktori koje vrati algoritam za Pollardovu $p-1$ faktorizaciju moraju testirati jesu li uistinu prosti. Naime, nekoliko godina se smatralo da je broj 1223165341640099735851 prosti faktor od $6^{175}-1$, ali se na kraju ispostavilo da je umnožak dvaju 11-znamenkastih prostih brojeva.

Dodatak A

Python kodovi

U ovom dodatku se nalaze implementacije nekih od algoritama koje smo vidjeli u prethodnim poglavljima.

A.1 Pollardova ρ metoda za faktorizaciju

```
import math

#funkcija pollard_rho prima broj n kojeg zelimo faktorizirati,
#funkciju f pomocu koje generiramo niz (x_i) te pocetnu vrijednost x_0
#vraca neki prosti faktor od n
def pollard_rho(n, f, x_0):
    sequence_x_i = [x_0]   #lista u koju spremamo vrijednosti niza (x_i)
    x = x_0
    d = 1                  #u d spremamo vrijednosti gcd

    while d == 1:
        x = f(x) % n       #racunamo elemente niza

        #racunamo gcd razlike nove vrijednosti x i svih prethodno
        #izracunatih vrijednosti i n
        for x_i in sequence_x_i:
            d = math.gcd(abs(x_i-x), n)
            if d > 1:      #ako je gcd > 1, onda smo gotovi
                return d

        sequence_x_i.append(x)
```

Floydova varijanta

```
import math

#funkcija pollard_rho_floyd prima broj n kojeg zelimo faktorizirati,
#funkciju f pomocu koje generiramo niz (x_i) te pocetnu vrijednost x_0
#vraca neki prosti faktor od n
def pollard_rho_floyd(n, f, x_0):
    x = x_0                 #u x spremamo vrijednosti niza (x_i)
    y = x_0                 #u y spremamo vrijednosti niza (x_{2i})
    d = 1                   #u d spremamo vrijednosti od gcd(y-x,n)

    while d == 1:
        x = f(x) % n        #racunamo niz (x_i) mod n
        y = f(f(y)) % n     #racunamo niz (x_{2i}) mod n
        d = math.gcd(abs(y-x), n)

    return d
```

Brentova varijanta

```
import math

#funkcija pollard_rho_brent prima broj n kojeg zelimo faktorizirati,
#funkciju f pomocu koje generiramo niz (x_i) te pocetnu vrijednost x_0
#vraca neki prosti faktor od n
def pollard_rho_brent(n, f, x_0):
    x = x_0                 #u x spremamo vrijednosti x_{2^k-1}
    y = f(x_0) % n          #spremamo vrijednosti niza (x_i) mod n
    t = 1                   #u t spremamo vrijednosti 2^{k-1}
    d = math.gcd(y-x, n)
    while d == 1:
        x = y

        for i in range(t):  #racunamo elemente niza bez gcd
            y = f(y) % n

        for i in range(t):  #racunamo elemente niza sa gcd
            y = f(y) % n
            d = math.gcd(abs(y-x), n)
            if d > 1:       #ako smo pronasli d > 1, gotovi smo
                return d

        t = 2*t
```

A.2 Pollardova $p-1$ metoda za faktorizaciju

Prva i druga faza (Varijacija velikog prostog broja)

```
import math
import sympy as sp

#pomocna funkcija koja provjerava je li b potencija prostog broja
#ako je, vraca taj prosti broj, inace vraca False
def prime_power(b):
    #ako je b prost, onda vrati b
    if sp.isprime(b):
        return b

    #inace, provjeravamo proste brojeve <= sqrt(b)
    for i in range(1, math.sqrt(b) + 1):
        if sp.isprime(i):
            if b % i == 0:
                while b % i == 0:
                    b = b / i
                if b == 1:
                    return i
                else:
                    return False


#funkcija pollard_p_1 prima broj n kojeg treba faktorizirati, granicu B1
#te dva opcionalna argumenta: bazu a (defaultna vrijednost joj je 2) i
#granicu B2 koja nam sluzi za drugu fazu (defaultno 0)
#vraca prosti faktor od n
def pollard_p_1(n, B1, a = 2, B2 = 0):
    if n % a == 0:        #provjeravamo je li n djeljiv s a,
        return a          #ako je vratimo a

    d = 1                 #u d spremamo vrijednosti gcd
    m = 1                 #u m spremamo vrijednosti lcm

    #provjeravamo za sve brojeve manje od granice B
    for b in range(2, B+1):
        m = prime_power(b) * m if prime_power(b) else m
        d = math.gcd((pow(a,m,n)-1), n)
        if d > 1:
            return d

    #drugu fazu provodimo jedino ako je granica B2 veca od granice B1
    if B2 > B1:
        c = pow(a,m,n)  #posljednja vrijednost iz prve faze
```

```
44          #lista prostih brojeva do granice B2
45          primes_to_B2 =
46              [prime for prime in range(B2+1) if sp.isprime(prime)]
47
48          #za svaki prosti broj racunamo c^p te gcd
49          for prime in primes_to_B2:
50              d = math.gcd((pow(c, prime, n)-1), n)
51              if d > 1:
52                  return d
53
54      return d
```

Bibliografija

[1] S. Agrawal, *Pollard's $p-1$ factorization*, https://web.archive.org/web/20160812075833/https://math.berkeley.edu/~sagrawal/su14_math55/notes_pollard.pdf.

[2] Brilliant, *Primitive roots*, https://brilliant.org/wiki/primitive-roots.

[3] R. Crandall i C. Pomerance, *Prime Numbers*, 2., Springer, 2005.

[4] A. Das, *Computational Number Theory*, Chapman & Hall/CRC, 2013.

[5] A. Dujella, *Teorija brojeva*, Školska knjiga, 2019.

[6] A. Dujella i M. Maretić, *Kriptografija*, Element, 2007.

[7] Programming Praxis, *Extending Pollard's $P-1$ Factorization Algorithm*, https://programmingpraxis.com/2010/03/19/extending-pollards-p-1-factorization-algorithm/.

[8] _____, *Pollard's $P-1$ Factorization Algorithm, Revisited*, https://programmingpraxis.com/2011/09/16/pollards-p-1-factorization-algorithm-revisited/.

[9] _____, *Pollard's $P-1$ Factorization Algorithm*, https://programmingpraxis.com/2009/07/21/pollards-p-1-factorization-algorithm/.

[10] H. Riesel, *Prime Numbers and Computer Methods for Factorization*, 2., Springer Science & Business Media, 1994.

[11] M. Vuković, *Složenost algoritama*, 2020, https://moodle.srce.hr/2020-2021/pluginfile.php/5029287/mod_resource/content/1/00-SA-skripta-2020.pdf.

[12] S. S. Wagstaf, Jr., *The joy of factoring*, AMS, 2013.

[13] Wikipedia, *Birthday problem*, https://en.wikipedia.org/wiki/Birthday_problem.

[14] ———, *Discrete logarithm*, https://en.wikipedia.org/wiki/Discrete_logarithm#Cryptography.

[15] ———, *Factorization*, https://en.wikipedia.org/wiki/Factorization.

[16] ———, *Integer factorization*, https://en.wikipedia.org/wiki/Integer_factorization.

[17] ———, *Pell number*, https://en.wikipedia.org/wiki/Pell_number.

[18] ———, *Pollard's $p-1$ algorithm*, https://en.wikipedia.org/wiki/Pollard%27s_p_%E2%88%92_1_algorithm#Example.

[19] ———, *Pollard's rho algorithm for logarithms*, https://en.wikipedia.org/wiki/Pollard's_rho_algorithm_for_logarithms.

[20] ———, *Pollard's rho algorithm*, https://en.wikipedia.org/wiki/Pollard%27s_rho_algorithm#Algorithm.

[21] ———, *Primitive root modulo n*, https://en.wikipedia.org/wiki/Primitive_root_modulo_n.

[22] ———, *Random walk*, https://en.wikipedia.org/wiki/Random_walk.

Sažetak

U teoriji brojeva, cjelobrojna faktorizacija je rastav složenog cijelog broja u produkt manjih cijelih brojeva. Ako su ti brojevi dodatno ograničeni na proste faktore, onda ovaj proces nazivamo rastav broja na proste faktore. Faktorizacija cijelih brojeva već je od davnina predmet proučavanja brojnih matematičara. Pronalazak prostog faktora nekog složenog broja smatra se teškim problemom i često se primjenjuje u kriptografiji. John M. Pollard je osmislio dvije bitne metode za faktorizaciju velikih prirodnih brojeva: ρ (rho) metodu i $p-1$ metodu.

Pollardova ρ metoda je efikasna za velike brojeve koji imaju barem jedan mali prosti faktor. Ideja metode je prvo konstruirati niz cijelih brojeva (x_i) modulo p, gdje je p prosti faktor koji tražimo. Zatim treba pronaći period tog niza te naposljetku odrediti prosti faktor p. Ovakav algoritam za faktorizaciju je neefikasan ako je period niza velik. Matematičari poput Floyda i Brenta su, koristeći svoje algoritme za pronalaženje ciklusa u nizu, dodatno poboljšali Pollardovu ρ metodu. Floydova varijanta se temelji na paralelnom računanju nizova (x_i) i (x_{2i}) te računanju $\gcd(x_{2k} - x_k, n)$. Kod Brentove varijante računamo niz (x_i) te $\gcd(x_i - x_{2^k-1}, n)$ za $k \in \mathbb{N}$ i $3 \cdot 2^{k-1} \leq i \leq 2^{k+1} - 1$. U obje varijante smo gotovi ako smo dobili da je $\gcd > 1$, no može se dogoditi da nam algoritmi ne daju prosti faktor kao izlaz. Metode se dodatno mogu ubrzati periodičnim računanjem najvećeg zajedničkog djelitelja.

Analognu metodi za faktorizaciju, Pollard je izumio i ρ metodu za izračun diskretnih logaritama. Problem diskretnog logaritma za cikličku grupu G se bavi pronalaskom broja k takvog da vrijedi $h = g^k$ za neki element $h \in G$, gdje je g neki generator grupe. Ovaj problem se također smatra teškim te se primjenjuje u kriptografiji. Objasnili smo Pollardovu ρ metodu za problem diskretnog logaritma za grupu $\mathbb{Z}_p^*$, gdje je p neparan prosti broj.

Pollardova $p-1$ metoda se temelji na Malom Fermatovom teoremu. Uspješna je za velike brojeve za koje vrijedi da se faktorizacija broja $p-1$ sastoji samo od malih prostih faktora, gdje je p prosti faktor broja kojeg želimo faktorizirati. Ideja je pronaći neki višekratnik m od $p-1$ te pomoću Euklidovog algoritma izračunati faktor p. Dakle, ako dobijemo da je $\gcd(a^m - 1, n)$ netrivijalan, pronašli smo faktor od n. I ova metoda nam može ne dati prosti faktor kao izlaz. Ako za izlaz dobijemo da je najveći zajednički djelitelj jednak 1, možemo primijeniti drugu fazu algoritma, tzv. varijaciju velikog prostog broja.

Summary

In number theory, integer factorization is the decomposition of a composite number into a product of smaller integers. If these factors are further limited to prime numbers, the process is called prime factorization. Integer factorization has been a subject of study by many mathematicians for a long time. Finding a prime factor of a composite number is considered a difficult problem and is often used in cryptography. John M. Pollard has invented two essential methods for factorizing large positive integers: ρ (rho) method and $p-1$ method.

Pollard's ρ mehod is effective for large numbers that have at least one small prime factor. The idea is to first construct a sequence of integers (x_i) modulo p, where p is the prime factor we are trying to find. Then we need to find the period of this sequence and finally determine the prime factor p. This factorization algorithm is inefficient if the period of the sequence is large. Mathematicians like Floyd and Brent further improved Pollard's ρ method by using their cycle finding algorithms. Floyd's variant is based on the parallel calculation of the sequences (x_i) and (x_{2i}) and the calculation of $\gcd(x_{2k} - x_k, n)$. For the Brent variant, we compute the sequence (x_i) and $\gcd(x_i - x_{2^k-1}, n)$ for $k \in \mathbb{N}$ and $3 \cdot 2^{k-1} \leq i \leq 2^{k+1} - 1$. In both variants we are done if we get that $\gcd > 1$, but it may happen that the algorithms do not give us a prime factor as the output. Methods can be further accelerated by periodically calculating the greatest common divisor.

Analogous to the factorization method, Pollard invented the ρ method for calculating discrete logarithms. The discrete logarithm problem for a cyclic group G concerns finding the number k such that $h = g^k$ is valid for some element $h \in G$, where g is a generator of the group. This problem is also considered difficult and is applied in cryptography. We have explained Pollard's ρ method for the discrete logarithm problem for the group $\mathbb{Z}_p^*$, where p is an odd prime number.

Pollard's $p-1$ method is based on Little Fermat's theorem. It is successful for large numbers for which it is true that the factorization of the number $p-1$ consists only of small prime factors, where p is the prime factor of the number we want to factorize. The idea is to find a multiple m of $p-1$ and use the Euclidean algorithm to calculate the factor p. So if we get that $\gcd(a^m - 1, n)$ is non-trivial, we have found a factor of n. This method may also not give us a prime factor as a output. If for the output we get that the greatest common

divisor is equal to 1, we can apply the second phase of the algorithm, the so-called large prime variation.

Životopis

Martina Gaćina rođena je 19. veljače 1998. godine u Zagrebu. U Šibeniku završava osnovnu školu Tina Ujevića te nakon toga prirodoslovno-matematičku gimnaziju u Gimnaziji Antuna Vrančića. Godine 2016. upisuje preddiplomski studij Matematika; smjer: istraživački na Prirodoslovno-matematičkom fakultetu u Zagrebu. Nakon završetka preddiplomskog studija, 2019. godine upisuje diplomski studij Računarstvo i matematika na istom fakultetu.