Problem PRIMES pripada klasi PTIME

Grgurica, Josip

Master's thesis / Diplomski rad

2014

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:217:343801

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-28**

Repository / Repozitorij:

Repository of the Faculty of Science - University of Zagreb

zir.nsk.hr

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Josip Grgurica

PROBLEM PRIMES PRIPADA KLASI PTIME

Diplomski rad

Voditelji rada:
prof. dr. sc.
Boris Širola
prof. dr. sc.
Mladen Vuković

Zagreb, 2014

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. _____

Sadržaj

1 Uvod **1**
1.1 Opis rada 1
1.2 Prsten polinoma 2
1.3 Teorija brojeva 7
1.4 Teorija složenosti 17
1.5 Algoritmi sa slučajnim elementima 22
1.6 Povijest problema PRIMES 24

2 Agrawal-Kayal-Saxenov algoritam **31**
2.1 AKS algoritam - dokaz korektnosti 31
2.2 Složenost AKS algoritima 43

Bibliografija **53**

Poglavlje 1

Uvod

1.1 Opis rada

Među mnogim granama matematike, teorija brojeva, koja se u grubo može definirati kao istraživanje prirodnih brojeva i njihovih svojstava, je jedna od najstarijih grana matematike. Jedan od razloga zašto je teorija brojeva očuvala svoju zanimljivost kroz sve ove godine je taj što se problemi unutar teorije brojeva mogu jednostavno opisati tako da ih i matematički početnici mogu razumijeti. No, to ne znači da se svi problemi unutar teorije brojeva mogu na jednostavan način riješiti. Promotrimo npr. Goldbachovu slutnju.

Godine 1742. pruski matematičar Christian Goldbach poslao je pismo Leonhardu Euleru u kojem je opisao svoju slutnju, a koja glasi da se svaki cijeli broj veći od 5 može napisati kao zbroj 3 prosta broj. Tijekom 18. stoljeća broj 1 je smatran prostim brojem, tako da današnja hipoteza glasi: "Svaki se paran cijeli broj veći od 2 može zapisati kao zbroj 2 prosta broja". Iako problem zvuči jednostavno njegovo rješenje je ostalo nepoznato do danas. Stoljećima su prosti brojevi velik izazov vrhunskim matematičarima. Iako svaki učenik osnovne škole može odrediti produkt dva broja, određivanje faktorizacije danog prirodnog broja nije jednostavno kao što se na prvu čini i matematičari taj problem istražuju preko 2000 godina. Kada proučavamo proste brojeve, prvo pitanje koje nam prirodno dolazi je sljedeće: "Za dani prirodni broj n, kako možemo odrediti da li je on prost?".

Kada bi naš cilj bio samo odrediti prostost nekog prirodnog broja n, ne mareći pritom na broj koraka koji će biti potrošen da bi izveli taj postupak, mogli bi jednostavno pokušati podijeliti n sa svim brojevima manjim od $\sqrt{n}$. No iako nam nije teško podijeliti dva broja, postupak je ipak eksponencijalne složenosti.

Prethodni opis određivanja prostosti danog prirodnog broja je primjer *deterministič kog* algoritma, tj. algoritma kojem je svaki korak jedinstveno određen. Nisu svi algoritmi deterministički, postoje i *algoritmi sa slučajnim elementima*. Takav algoritam bi bio onaj, kojem je vjerojatnost da greškom složen prirodni broj proglasi prostim, veća od 0.

Godine 2002. Manindra Agrawal, Neeraj Kayal i Nitin Saxena razvili su novi deterministički algoritam za provjeru prostosti. Ono što je njihov algoritam učinilo drugačijim i boljim od prije opisanih je to što je njegovo izvođenje mnogo brže. Dakle, za ulazni podatak duljine n bitova, broj koraka koje napravi njihov algoritam je ograničen nekim polinomom u n. Cilj ovog diplomskog rada je detaljno opisati Agrawal, Kayal, Saxenin algoritam i dati dokaz da se izvodi u polinomnom vremenu.

Diplomski rad je podijeljen u dva poglavlja koja imaju više zasebnih dijelova. Prvo poglavlje je uvodno poglavlje u kojem navodimo bitne definicije i rezultate iz teorije brojeva, teorije složenosti i algebre koji će nam biti potrebni u drugim dijelovima diplomskog rada. Prvo poglavlje se sastoji od pet točaka, a to su: *Prsten polinoma*, *Teorija brojeva*, *Teorija složenosti*, *Algoritmi sa slučajnim elementima* i *Povijest problema PRIMES*. U točki *Prsten polinoma* navodimo definicije i dajemo bitne rezultate o prstenu polinoma koji su nam potrebni da bi mogli dokazati korektnost algoritma kojeg su konstruirali Agrawal, Saxena i Kayal. U točkama *Teorija brojeva* i *Teorija složenosti* bavimo se osnovnim definicijama i teoremima iz te dvije grane koje su nam potrebne u nastavku diplomskog rada. U točki *Algoritmi sa slučajnim elementima* navodimo definicije i bitnije rezultate o algoritmima sa slučajnim elementima koji će nam poslužiti da bolje shvatimo algoritme kao što su Solovay-Strassenov algoritam, Miller-Rabinov algoritam i drugi. U točki *Povijest problem PRIMES* opisat ćemo neke algoritme za testiranje prostosti i dat ćemo dokaz da je problem PRIMES u presjeku klasa složenosti NP i co-NP. Drugo poglavlje sastoji se od dvije točke: *AKS algoritam - dokaz korektnosti* i *Složenost AKS algoritma*. U točki *AKS algoritam - dokaz korektnosti* kroz niz rezultata kao što je poznata lema Hendrika Lenstre, pokazujemo da je algoritam koji su dali Agrawal, Saxena i Kayal korektan. U točki *Složenost AKS algoritma* pokazujemo da je algoritam, za koji smo u prethodnoj točki pokazali da je korektan, polinomne složenosti.

1.2 Prsten polinoma

U ovoj pripremnoj točki ukratko ćemo navesti neke relevantne definicije i rezultate o prstenima, i posebno prstenima polinoma. No prije toga podsjetimo se na neke oznake i osnovne činjenice o grupama. Grupa, kao algebarska struktura, jedan je od osnovnih pojmova u matematici; za više detalja vidjeti npr. [16, Pogl. 1]. Grupe se pojavljuju u gotovo svim granama matematike; npr. u analizi, (linearnoj) algebri, teoriji brojeva itd.

Ukoliko imamo neku apstraktnu grupu G, vrlo često se pripadna binarna operacija koja tu grupu definira označava u multiplikativnoj formi, s točkom, što eventualno naglašavamo pisanjem $G = (G, \cdot)$. Tako za dva elementa $x, y \in G$ množenjem dobivamo novi element grupe $x \cdot y \in G$; ili često čak i ispuštamo točku kao znak množenja u grupi, pa pišemo

$xy \in G$. Posebno, za bilo koji element grupe $x \in G$ definiramo njegovu n-tu potenciju

$$\overbrace{x \cdot x \cdot x \cdots x}^{n \text{ puta}} = x^n, \qquad n \in \mathbb{N}.$$

Nadalje, posebno stavljamo $x^0 = e$, gdje je e neutralni element od G, te $x^{-n} = (x^{-1})^n$, gdje je x^{-1} inverzni element od x. Ponekad se operacija u grupi piše u tzv. aditivnoj formi, sa znakom zbrajanja $+$. To je posebno uobičajeno kada je grupa komutativna. I u tom slučaju neutral neke takve grupe A pišemo kao nulu 0, a inverzni element od $a \in A$ kao $-a$. Analogno kao i u multiplikativnoj formi imamo

$$\overbrace{a + a + a \cdots + a}^{n \text{ puta}} = na, \qquad n \in \mathbb{N};$$

te $0a = 0$ i $(-n)a = n(-a)$, za $n \in \mathbb{N}$.

Ako je G grupa, **red grupe** G definiramo kao

$$|G| := \text{card}(G);$$

tj. red grupe je kardinalni broj skupa G. Posebno, ako je red grupe konačan, kažemo da je G konačna grupa; inače je G beskonačna grupa. Podsjetimo se i na to da u nekoj grupi $G = (G, \cdot)$, s neutralom e, za neki $x \in G$ kažemo da je $n \in \mathbb{N}$ **red elementa** x ukoliko je to najmanji prirodan broj takav da vrijedi $x^n = e$. Vrlo često se posljednja činjenica u teoriji grupa označava s

$$\text{ord}_G(x) = n \qquad \text{ili} \qquad \text{o}_G(x) = n.$$

Za ono što sljedi podsjetimo se i na pojam kvocijentne grupe. Ukoliko je G grupa, kažemo da je neka njezina podgrupa N normalna ako je

$$xNx^{-1} = N, \qquad \forall x \in G.$$

Za takve G i N, na kvocijentnom skupu G/N dobro je definirana operacija

$$G/N \times G/N \to G/N, \qquad (xN, yN) \mapsto xyN,$$

i ona na G/N određuje strukturu grupe. Ta se grupa zove **kvocijentna grupa**, od G po N.

Za razliku od grupe, gdje imamo samo jednu "unutarnju operaciju", prsten je algebarska struktura s dvije takve operacije; za više detalja vidjeti npr. [16, Pogl. 2]. One se standardno označavaju s $+$ i $\cdot$ simbolima, te zovu zbrajanje i množenje. Preciznije rečeno; neki neprazan skup $R = (R, +, \cdot)$ je **prsten** ukoliko je $(R, +)$ komutativna grupa, $(R, \cdot)$ je polugrupa, te vrijedi distributivnost množenja prema zbrajanju. Element 0, neutral u grupi $(R, +)$, zove se **nula** prstena R. Ako postoji element $1 = 1_R \in R$ takav da je

$$1 \cdot x = x \cdot 1 = x, \qquad \forall x \in R,$$

kažemo da je 1 **jedinični element**, ili kraće jedinica, prstena R. I tada govorimo da je R **prsten s jedinicom**.

Element $w \in R$, gdje je R prsten s jedinicom, je **invertibilan** ako postoji element $w' \in R$ takav da je

$$ww' = w'w = 1.$$

Lako se provjeri da je

$$R^* := \text{skup svih invertibilnih elemenata u } R,$$

grupa s obzirom na operaciju množenja u prstenu R; ona se zove **grupa invertibilnih elemenata** u R. Podsjetimo se i da je prsten R **tijelo**, ili **prsten s dijeljenjem**, ako je svaki ne-nul element u R invertibilan; tj. $R^* = R \setminus \{0\}$ je grupa invertibilnih elemenata u R. Komutativno tijelo zove se **polje**. Nadalje, za element r nekog polja $\mathbb{F}$ kažemo da je on n**-ti korijen jedinice** ako je $r^n = 1$. Kažemo da je r **primitivni** n**-ti korijen jedinice** u $\mathbb{F}$ ako je to element reda n u $\mathbb{F}$; tj., ako je $\mathrm{ord}_{\mathbb{F}^*}(r) = n$. (Naprimjer; brojevi $\{1, \imath, -1, -\imath\}$ su svi četvrti korijeni jedinice u polju kompleksnih brojeva $\mathbb{C}$, ali samo $\{\imath, -\imath\}$ su primitivni četvrti korijeni jedinice.)

Sada se podsjetimo na pojmove ideala u prstenu i kvocijentnog prstena. Kako će svi naši prsteni, koje u daljnjem promatramo, biti komutativni, spomenute pojmove ćemo uvesti samo u komutativnoj situaciji. Neka je A komutativan prsten, s jedinicom. Podskup $I \subseteq A$ je **ideal** u A ako su ispunjena sljedeća dva uvjeta:

1. I je potprsten od A;
2. Za sve $a \in A$ i $x \in I$ je $ax \in I$.

Posebno primijetimo kako je za svaki element $x \in A$, skup

$$(x) := Ax = \{ax \mid a \in A\}$$

ideal u A; gdje govorimo da je to ideal **generiran elementom** x. U skladu s tim, kažemo da je neki ideal I u prstenu A **glavni** ukoliko je on oblika $I = (x)$, za neki element $x \in A$.

Analogno pojmu kvocijentne grupe, definiramo i pojam kvocijentnog prstena. Jedine razlike će biti u tome što ćemo prsten "cijepati" po idealima, i imat ćemo dvije unutarnje operacije. Sasvim precizno; recimo da imamo neki prsten $A = (A, +, \cdot)$ i neki ideal I u njemu. Onda je poebno $(I, +)$ normalna podgrupa od $(A, +)$, pa je dobro definiranaa kvocijentna grupa

$$A/I = (A, +)/(I, +).$$

Ukoliko na toj aditivnoj grupi A/I sada definiramo operaciju množenja

$$A/I \times A/I \to A/I, \qquad (x + I, y + I) \mapsto xyI,$$

onda A/I ima strukturu prstena; on se zove **kvocijentni prsten** od A po I.

Sada prelazimo na glavni dio ove točke, gdje ćemo govoriti o prstenima polinoma s koeficijentima iz nekog komutativnog prstena. Polinomi su osnovne funkcije u matemtici. Posebno, polinomi nad $\mathbb{R}$ ili $\mathbb{C}$, tj. s realnim ili kompleksnim koeficijentima, su objekti bez kojih je npr. nemoguće zamisliti matematičku analizu. Neka je sada A proizvoljan komutativan prsten s jedinicom. **Polinom** nad A je formalni izraz

$$f(X) = \sum_{i=0}^{n} a_i X^i = a_0 + a_1 X + \dots a_n X^n,$$

gdje je $n \in \mathbb{N}$, koeficijenti a_i, $0 \leq i \leq n$ su elementi iz A i X je simbol koji ne pripada A. Po konvenciji izraze $a_i X^i$ za koje je $a_i = 0$ ne zapisujemo. Stoga kada uspoređujemo, zbrajamo ili množimo dva polinoma $f(X)$ i $g(X)$ nad A možemo pretpostaviti da oba sadrže jednake potencije od X. Za polinome

$$f(X) = \sum_{i=0}^{n} a_i X^i \quad \text{i} \quad g(X) = \sum_{i=0}^{n} b_i X^i$$

nad A kažemo da su jednaki ako i samo ako vrijedi $a_i = b_i$ za sve $0 \leq i \leq n$. Zbroj polinoma $f(X)$ i $g(X)$ nad A definiramo kao

$$f(X) + g(X) = \sum_{i=0}^{n} (a_i + b_i) X^i. \tag{1.1}$$

Umnožak dva polinoma $f(X)$ i $g(X)$, dana kao gore definiramo s

$$f(X)g(X) = \sum_{k=0}^{n+m} c_k X^k \quad \text{gdje je} \quad c_k = \sum_{\substack{i+j=k \\ 0 \leq i \leq n, 0 \leq j \leq m}} a_i b_j. \tag{1.2}$$

Lako se pokaže da je skup svih polinoma nad prstenom A, zajedno s ovim operacijama zbrajanja i množenja, prsten. Taj se prsten zove **prsten polinoma** nad A, i označava s $A[X]$.

Nula u prstenu $A[X]$ je polinom čiji su svi koeficijenti jednaki 0; označavamo ga s 0, i zovemo nul-polinom. U nastavku ćemo promatrati polinome nad poljima. Neka je $f(X) = \sum_{i=0}^{n} a_i X^i$ polinom nad A različit od nul-polinoma (možemo pretpostaviti da $a_n \neq 0$). Tada a_n nazivamo *vodećim koeficijentom*, a_0 slobodnim koeficijentom, a n *stupnjem* u oznaci $\deg f(X)$. Polinome stupnja manjeg ili jednakog 0 nazivamo *konstantnim polinomima*. Ako A ima jedinicu 1 i ako je vodeći koeficijent od $f(X)$ jednak 1, tada kažemo da je $f(X)$ *normiran*. Neka je $\mathbb{F}$ polje. Pojam djeljivosti unutar prstena $\mathbb{F}[X]$ podrazumijeva sljedeće. Polinom $g \in \mathbb{F}[X]$ *dijeli* polinom $f \in \mathbb{F}[X]$ ako postoji polinom $h \in \mathbb{F}[X]$ takav da vrijedi

$f = gh$. Kao i kod prstena cijelih brojeva, unutar prstena polinoma nad poljem postoji pojam dijeljenja s ostatkom. Dokaz sljedećeg teorema u malo općenitijoj formi može se pronaći u [16], str. 85, teorem 8.7.

Teorem 1.2.1. *Neka je $g \neq 0$ polinom u $\mathbb{F}[X]$. Tada za svaki $f \in \mathbb{F}[X]$ postoje jedinstveni polinomi $q, r \in \mathbb{F}[X]$ takvi da vrijedi*

$$f = qg + r, \quad gdje \quad \deg(r) < \deg(g).$$

Proste elemente prstena $\mathbb{F}[X]$ nazivamo ireducibilnim polinomima. Sada ćemo dati formalnu definiciju.

Definicija 1.2.2. *Za polinom $p \in \mathbb{F}[X]$ kažemo da je **ireducibilan nad** $\mathbb{F}$ ako je p pozitivnog stupnja i $p = bc$, gdje $b, c \in \mathbb{F}[X]$, povlači da je b ili c konstantni polinom.*

Drugim riječima, polinom pozitivnog stupnja je ireducibilan nad $\mathbb{F}$ ako dopušta samo trivijalnu faktorizaciju. Polinome u $\mathbb{F}[X]$ koji nisu ireducibilni nazivamo *reducibilnima nad* $\mathbb{F}$. Ireducibilni polinomi su od velike važnosti za strukturu prstena $\mathbb{F}[X]$ pošto se svaki polinom iz $\mathbb{F}[X]$ može zapisati kao produkt ireducibilnih polinoma. Navodimo sada teorem o jedinstvenoj faktorizacija polinoma u prstenu $\mathbb{F}[X]$. Dokaz teorema može se pronaći u [10], str. 23, teorem 1.59.

Teorem 1.2.3. *Svaki polinom pozitivnog stupnja $f \in \mathbb{F}[X]$ možemo zapisati na sljedeći način:*

$$f = ap_1^{e_1} \dots p_k^{e_k},$$

gdje je $a \in \mathbb{F}$, $p_1, \dots, p_k$ su različiti normirani ireducibilni polinomi u $\mathbb{F}[X]$ i $e_1, \dots, e_k$ su pozitivni prirodni brojevi. Štoviše, ovakva faktorizacija je jedinstvena do na poredak faktora.

Prethodna faktorizacija se naziva još i kanonska faktorizacija polinoma f u $\mathbb{F}[X]$. Glavno pitanje koje nas zanima u vezi polinoma iz $\mathbb{F}[X]$ je to je li dani polinom ireducibilan nad poljem $\mathbb{F}$. Dokaz sljedećeg torema može se pronaći u [10], str. 25.

Teorem 1.2.4. *Za $f \in \mathbb{F}[X]$, kvocijentni prsten $\mathbb{F}[X]/(f)$ je polje ako i samo ako je f ireducibilan nad $\mathbb{F}$.*

Sa $\mathbb{F}_p$ ćemo označavati konačno polje sa p elemenata, gdje je p prost broj. Primijetimo da je zapravo, do na izomorfizam, to polje jednako kvocijentnom prstenu ostataka modulo p. Tojest, $\mathbb{F}_p = \mathbb{Z}_p$; gdje je općenito za prirodan broj n prsten ostataka modulo n,

$$\mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z}.$$

Sada nas zanima koliko je elemenata u polju $\mathbb{F}_p[X]/(h)$, gdje je h ireducibilan polinom nad $\mathbb{F}_p$. Sljedeći je teorem izravna posljedica prethodnog teorema.

Teorem 1.2.5. *Ako je p prost broj i $h(X)$ ireducibilan polinom stupnja d u $\mathbb{F}_p$ onda je $\mathbb{F}_p[X]/(h)$ konačno polje reda p^d.*

Sljedeća nam lema govori koliki je broj korijena polinoma u nekom polju. Ona se lako dokazuje matematičkom indukcijom po stupnju polinoma, uz korištenje sljedeće jednostavne činjenice: Ako je $\mathbb{F}$ polje i $\alpha \in \mathbb{F}$ korijen polinoma $f \in \mathbb{F}[X]$, onda linearan polinom $X - \alpha$ dijeli $f(X)$ u $\mathbb{F}[X]$.

Lema 1.2.6. *Neka je $\mathbb{F}$ polje, neka je $f(X)$ polinom u $\mathbb{F}[X]$ i neka je $d \geq 1$ stupanj od $f(X)$. Tada postoji najviše d elemenata $\alpha \in \mathbb{F}$ takvih da $f(\alpha) = 0$.*

Sljedeći pojam je važan u proučavanju tzv. ciklotomskih polja.

Definicija 1.2.7. *Za $n \in \mathbb{N}$ definiramo **n-ti ciklotomski polinom** $\Phi_n(X)$ nad poljem $\mathbb{F}$ kao*

$$\Phi_n(X) := \prod_{\omega}(X - \omega),$$

gdje produkt uzimamo po svim elementima ω koji su n-ti primitivni korijeni jedinice u polju $\mathbb{F}$.

Sljedeća lema omogućava rekurzivno računanje ciklotomskih polinoma, počevši od prvog ciklotomskog polinoma $\Phi_1(X) = X - 1$; za njezin dokaz vidjeti npr. [10], str. 60, Teorem 2.45.

Lema 1.2.8. *Za sve $n \in \mathbb{N}$ vrijedi jednakost*

$$\prod_{d|n} \Phi_d(X) = X^n - 1.$$

Isto tako, iz prethodne leme vidimo da ako d dijeli n, onda i polinom $\Phi_d(X)$ dijeli $X^n - 1$. Navodimo ovdje i ovu lemu, čiji se dokaz može vidjeti u [10], str. 61, Teorem 2.47.

Lema 1.2.9. *Neka je $n \in \mathbb{N}$ i neka je p prost broj takav da vrijedi $(p, n) = 1$. Tada $\Phi_n(X)$ možemo rastaviti na $\frac{\varphi(n)}{o_n(p)}$ različitih normiranih ireducibilnih polinoma iz $\mathbb{F}_p[X]$.*

1.3 Teorija brojeva

U ovoj uvodnoj točki cilj nam je uvesti neke osnovne pojmove i rezultate iz teorije brojeva koji se koriste ili na koje ćemo se pozivati u ostalim dijelovima diplomskog rada. Teorija brojeva je grana matematike koja se ponajprije bavi proučavanjem svojstava skupa prirodnih brojeva. Prosti brojevi i faktorizacija broja na proste faktore od velike su važnosti u teoriji brojeva. Djeljivost, funkcije kao što je Eulerova funkcija i teoremi kao što je

mali Fermatov teorem, pojmovi kao što su najveća zajednička mjera i najmanji zajednički višekratnik od fundamentalne važnosti su za teoriju brojeva. Najveću zajedničku mjeru prirodnih brojeva $n_1, \dots n_k$ označavat ćemo sa $(n_1, \dots, n_k)$.

Teorem 1.3.1. ***(Mali Fermatov teorem)*** *Ako je n prost broj onda za svaki prirodni broj b koji nije djeljiv sa n vrijedi*

$$b^{n-1} \equiv 1 \pmod{n}. \tag{1.3}$$

Važno je naglasiti da obrat ovog teorema ne vrijedi. Naime, n može biti složen, a da ipak za neki b vrijedi kongruencija (1.3). Štoviše, kako pokazuje sljedeći instruktivan i jednostavan konkretan primjer, posljednje je moguće i za svaki b koji je relativno prost s n.

Primjer 1.3.2. Neka je $n = 561$; to je složen broj čija je faktorizacija na proste faktore $561 = 3 \cdot 11 \cdot 17$. Pokažimo sljedeću tvrdnju:

- *Za* bilo koji *broj* $b > 1$ *takav da je* $(b, 561) = 1$ *imamo*

$$b^{560} \equiv 1 \pmod{561}$$

Dovoljno je pokazati da je $b^{560} \equiv 1 \pmod{p_i}$, kada je $p_1 = 3$, $p_2 = 11$ ili $p_3 = 17$. (Jer ako je posljednja kongruencija ispunjena, tj. p_i dijeli $b^{560} - 1$, onda produkt $p_1\, p_2\, p_3 = 561$ također dijeli $b^{560} - 1$; što znači da imamo tvrdnju.) Pokažimo to samo za npr. $p_3 = 17$, što je zapravo najmanje trivijalan slučaj. Tu je, naravno, dovoljno gledati samo one b-ove takve da je $1 < b < 17$ i $(b, 561) = 1$; tj.

$$b \in \{2, 4, 5, 7, 8, 10, 13, 14, 16\}.$$

Sada, za $b = 2$ imamo

$$2^4 \equiv -1 \pmod{17} \quad \Rightarrow \quad 2^8 \equiv 1 \pmod{17}.$$

Kako je $560 = 8 \cdot 70$, sljedi da je

$$2^{560} \equiv (2^8)^{70} \equiv 1^{70} \equiv 1 \pmod{17}.$$

Za $b = 4$ imamo

$$4^2 \equiv -1 \pmod{17} \quad \Rightarrow \quad 4^{560} \equiv (4^2)^{280} \equiv (-1)^{280} \equiv 1 \pmod{17}.$$

Za $b = 5$ imamo

$$5^2 \equiv 8 \pmod{17} \quad \Rightarrow \quad 5^4 \equiv 64 \equiv -4 \pmod{17},$$

i onda

$$5^8 \equiv (-4)^2 \equiv -1 \pmod{17} \quad \Rightarrow \quad 5^{16} \equiv 1 \pmod{17}.$$

Kako je $560 = 16 \cdot 35$, slijedi da je

$$5^{560} \equiv (5^{16})^{35} \equiv 1^{35} \equiv 1 \pmod{17}.$$

Za $b = 7$ imamo

$$7^2 \equiv -2 \pmod{17} \quad \Rightarrow \quad 7^8 \equiv (-2)^4 \equiv -1 \pmod{17} \quad \Rightarrow \quad 7^{16} \equiv 1 \pmod{17}.$$

Kao i za $b = 5$, zaključujemo da je

$$7^{560} \equiv (7^{16})^{35} \equiv 1^{35} \equiv 1 \pmod{17}.$$

Za $b = 8$, koristeći gornji račun za $b = 5$, imamo

$$8^2 \equiv -4 \pmod{17} \quad \Rightarrow \quad 8^8 \equiv 1 \pmod{17} \quad \Rightarrow \quad 8^{560} \equiv 1 \pmod{17}.$$

Konačno; za $b = 10$ imamo

$$10^{560} \equiv (17 - 7)^{560} \equiv 7^{560} \equiv 1 \pmod{17}.$$

Analogno tretiramo slučajeve $b = 13$, 14 i 16.

Definicija 1.3.3. *Ako je n neparan složen broj, te b cijeli broj takav da je $(b, n) = 1$ i $b^{n-1} \equiv 1 \pmod{n}$, onda kažemo da je n **pseudoprost u bazi** b.*
*Za skup prirodnih brojeva $\{b_1, \ldots, b_k\}$ kažemo da je **skup baza** za prirodni broj n ako vrijedi da je n pseudoprost u bazi b_i za $1 \leq i \leq k$.*

Kako bismo bolje razumjeli koliko se često može nalaziti prirodne brojeve koji su pseudoprosti u nekoj bazi, bit će potrebno razmotriti neka algebarska svojstva toga pojma. U tu svrhu, ali i da bismo vidjeli koje se tu vrste argumenata koriste, dokazujemo tek jedan, niže navedeni, teorem ([6], Teorem 12.21). Ali prije samog iskaza teorema i njegova dokaza, podsjetimo se ukratko na neke činjenice o prstenima ostataka i pripadnim grupama invertibilnih elemenata.

Elemente kvocijentnog prstena $\mathbb{Z}/n\mathbb{Z}$, za $n \in \mathbb{N}$, zovemo *ostatci modulo n* i označavamo ih s $\overline{x}$, za $x \in \mathbb{N}$; tj. imamo

$$\overline{x} := x + n\mathbb{Z}, \qquad \text{za } x \in \{0, 1, \ldots, n-1\}.$$

Dakle kao skup je

$$\mathbb{Z}/n\mathbb{Z} = \{\overline{0}, \overline{1}, \ldots, \overline{n-1}\};$$

a njegove elemente standardno zbrajamo i množimo modulo n. Nadalje, dobro je poznato da je grupa invertibilnih elemenata u $\mathbb{Z}/n\mathbb{Z}$ jednaka

$$(\mathbb{Z}/n\mathbb{Z})^* = \{\overline{k} \mid k \in \{1, \ldots, n-1\} \text{ takav da } (n,k) = 1\}.$$

Nadalje, broj elemenata u toj grupi jednak je $\varphi(n)$, gdje je φ dobro poznata *Eulerova funkcija* (v. npr [3], Poglavlje 3, za više detalja o funkciji φ).

Vezano uz definiciju koja slijedi podsjetimo se da za element g neke grupe G, s neutralnim elementom e, kažemo da je (konačnog) *reda* $d \in \mathbb{N}$ ako je

$$g^d = e \qquad \& \qquad g^i \neq e, \quad \forall 1 \leq i < d.$$

Drugim riječima, red elementa $g \in G$ jednak je broju elemenata cikličke podgrupe $\langle g \rangle$ od G, generirane tim elementom.

Definicija 1.3.4. *Neka su a i r relativno prosti prirodni brojevi. Najmanji prirodan broj d sa svojstvom da je $a^d \equiv 1 \pmod{r}$ zove se* ***red*** *od a modulo r, i označava s $\mathrm{o}_r(a)$.*

Napomena 1.3.5. Zapravo, za relativno proste prirodne brojeve a i r je broj $\mathrm{o}_r(a)$ jednak upravo redu elementa $\overline{a}$ u multiplikativnoj grupi $(\mathbb{Z}/r\mathbb{Z})^*$. Naime, kongruencija $a^d \equiv 1 \pmod{r}$ znači da r dijeli $a^d - 1$; a to pak znači da d-ta potencija $\overline{a}^d$, elementa $\overline{a}$ u grupi $(\mathbb{Z}/r\mathbb{Z})^*$, jest jednaka neutralnom elementu $\overline{1}$ te grupe. Kako je d izabran kao najmanji takav broj, rečeno slijedi.

Sada smo spremni za najavljeni teorem.

Teorem 1.3.6. *Neka je n neparan broj. Tada vrijede sljedeće tvrdnje:*

1. *Broj n je pseudoprost u bazi b ako i samo ako red elementa $\overline{b}$, u multiplikativnoj grupi $(\mathbb{Z}/n\mathbb{Z})^*$, dijeli $n-1$.*
2. *Ako je n pseudoprost u bazama b_1 i b_2, onda je on pseudoprost i u bazama $b_1 b_2$, te b_1^{-1} i $b_1 b_2^{-1}$ modulo n.*
3. *Ako n ne zadovoljava (1.1) za neku bazu b, za koju je $(n,b) = 1$, onda $c^{n-1} \not\equiv 1 \pmod{n}$ za najmanje pola svih mogućih baza c (tj. brojeva $1 \leq c < n$ koji su relativno prosti s n).*

Dokaz. 1. Neka je d red elementa $\overline{b}$ u grupi $(\mathbb{Z}/n\mathbb{Z})^*$. To znači da je $\overline{b}^d = \overline{1}$, i da je d najmanji prirodan broj s tim svojstvom. S druge strane, to da je n pseudoprost u bazi b znači, po definiciji, da imamo $b^{n-1} \equiv 1 \pmod{n}$. A ta kongruencija znači da je $\overline{b}^{n-1} = \overline{1}$, u grupi $(\mathbb{Z}/n\mathbb{Z})^*$.

Sada; ako d dijeli $n-1$, onda iz $\overline{b}^d = \overline{1}$ očito slijedi $\overline{b}^{n-1} = \overline{1}$; tj. imamo implikaciju s desna na lijevo. Obratno; ako imamo $\overline{b}^{n-1} = \overline{b}^d = \overline{1}$, onda napišemo, po teoremu o djeljenju s ostatkom u $\mathbb{Z}$,

$$n-1 = q\,d + r, \qquad \text{za neke } q \text{ i } r \text{ takve da je } 0 \leq r < d.$$

Pretpostavimo da je $r > 0$. Onda bi bilo

$$\overline{1} = \overline{b}^{n-1} = (\overline{b}^d)^q\,\overline{b}^r = \overline{1}^q\,\overline{b}^r = \overline{b}^r;$$

tj. red od $\overline{b}$ je prirodan broj strogo manji od d. No to je nemoguće. Zaključujemo da je nužno $r = 0$; tj. da d dijeli $n-1$.

2. Iz kongruencija $b_1^{n-1} \equiv b_2^{n-1} \equiv 1 \pmod{n}$ slijedi da je i

$$(b_1 b_2)^{n-1} \equiv b_1^{n-1} b_2^{n-1} \equiv 1 \cdot 1 \equiv 1 \pmod{n}.$$

Znači; ako je n pseudoprost u bazama b_1 i b_2, onda je on također pseudoprost i u bazi $b_1 b_2$.

Neka je sada $b_1^{n-1} \equiv 1 \pmod{n}$; tj. n je pseudoprost u bazi b_1. Kako smo već rekli, to znači da je $\overline{b}^{n-1} = \overline{1}$, u grupi $(\mathbb{Z}/n\mathbb{Z})^*$. Posebno, $\overline{b_1}$ je invertibilan element u toj grupi. Neka je $\overline{\beta_1} = (\overline{b_1})^{-1}$ njegov inverz, za neki (jedinstven!) $\beta_1 \in \{1, \ldots, n-1\}$. Sada imamo

$$\overline{1} = \overline{\beta_1}\,\overline{b_1} \quad \Rightarrow \quad \overline{1} = (\overline{1})^{n-1} = (\overline{\beta_1})^{n-1}(\overline{b_1})^{n-1} = (\overline{\beta_1})^{n-1}\overline{1} = (\overline{\beta_1})^{n-1};$$

tj. $(\overline{\beta_1})^{n-1} = \overline{1}$. Ali posljednja jednakost znači da je $\beta_1^{n-1} \equiv 1 \pmod{n}$, tj. n je pseudoprost u bazi β_1, po definiciji. Preostaje vidjeti da je zapravo β_1 u iskazu teorema označen s b_1^{-1}.

Tvrdnja da je n i pseudoprost u bazi $b_1 b_2^{-1}$, kada se b_2^{-1} uzima modulo n, je sada jasna.

3. Neka je $\mathcal{B} = \{b_1, \ldots, b_s\}$ skup svih, međusobno različitih, baza za koje je n pseudoprost. Fiksirajmo bazu b, kao u iskazu tvrdnje 3., za koju n nije pseudoprost; tj. vrijedi $b^{n-1} \not\equiv 1 \pmod{n}$. Ali onda, po dokazanom u 2., jasno je da se skup $b\mathcal{B} = \{bb_1, \ldots, bb_s\}$ sastoji od (međusobno različitih) baza za koje n nije pseudoprost. Dakle, postoji barem s baza za koje n nije pseudoprost, što je i trebalo pokazati.

□

Pretpostavimo sada da imamo neki ("veliki") prirodan broj n za koji želimo znati da li je on prost broj. Ideja koja koristi pseudoproste brojeve je sljedeća. Uzimamo nasumce "manje" prirodne brojeve b, koji su relativno prosti s n, i gledamo da li kongruencija (1.1) vrijedi ili ne. Ukoliko nađemo neki b za koji (1.1) ne vrijedi, n je očito složen broj; po Malom Fermatovom teoremu. No ukoliko (1.1) vrijedi za "mnogo" b-ova, možemo se nadati da bi n mogao biti prost. No tu, kako smo već i vidjeli u Primjeru 1.2.2, postoji mogućnost da smo ipak "na krivom putu". S tim u vezi imamo sljedeću definiciju.

Definicija 1.3.7. *Složen broj za koji vrijedi relacija (1.3) za svaki cijeli broj b takav da je* $(b, n) = 1$ *zove se* ***Carmichaeleov broj****.*

Prva pitanja koja se sada ovdje postavljaju su ova: *Ima li uopće Carmichaelovih brojeva, i ako da, ima li ih konačno mnogo ili beskonačno mnogo? Nadalje, što možemo reći o "strukturi", ili nekim bitnijim svojstvima, Carmichaelovih brojeva?* Odgovor na prvi dio prvog pitanja već znamo; on je potvrdan, jer kako smo pokazali u spomenutom Primjeru 1.2.2, broj 561 je Carmichaelov. Štoviše, može se pokazati da je to *najmanji* Carmichaelov broj.

Sljedeća propozicija govori o strukturi Carmichaelovih brojeva; detaljan dokaz može se naći npr. u [8], Odjeljak V.1.

Propozicija 1.3.8. *(Svojstva Carmichaeleovih brojeva) Neka je n neparan složen broj. Tada vrijede sljedeće tvrdnje:*

1. *Ako je n Carmichaelov broj, onda je n kvadratno slobodan; tj. nije djeljiv s* p^2*, za nikoji prost broj p.*
2. *Ako je n kvadratno slobodan, onda je on Carmichaelov broj ako i samo ako* $p-1|n-1$ *za svaki prost djelitelj p od n.*
3. *Carmichaelov broj mora imati barem tri različita prosta djelitelja; tj., ne može biti oblika* $n = pq$*, za neka dva međusobno različita prosta broja p i q.*

Napomenimo isto tako da se u dokazu prve dvije tvrdnje propozicije, gdje ima nešto posla, koristi činjenica iz sljedeće leme, o grupama invertibilnih elemenata za prstene ostataka; dokaz leme može se naći npr. u [7], Lema 12.24. Onda, imajući prve dvije tvrdnje propozicije, treća se tvrdnja dobije dosta lako.

Lema 1.3.9. *Ako je p prost broj, onda je grupa* $(\mathbb{Z}/p^2\mathbb{Z})^*$ *ciklička.*

Posljednji rezultat o Carmichaelovim brojevima, koji ovdje navodimo, također bez dokaza, je sljedeći važan i netrivijalan teorem koji su 1993. godine dokazali Alford, Granville i Pomerance u radu [3].

Teorem 1.3.10. *Postoji beskonačno mnogo Carmichaelovih brojeva. Preciznije rečeno, za pozitivan relan broj X definirajmo skup*

$$C(X) := \{n \mid n \leq X \ \ \& \ \ n \text{ je Carmichaelov broj}\}.$$

Onda ako je X "jako velik", za kardinalni broj $\#C(X)$ *imamo ocjenu*

$$\#C(X) > X^{2/7};$$

tj., posljednja nejednakost vrijedi asimptotski kada $X \to \infty$*.*

Da bi mogli definirat tzv. Legendreov simbol trebat će nam pojam kvadratnih ostataka. Stoga sada navodimo definiciju.

Definicija 1.3.11. *Neka je p neparan prost broj. Za broj b, koji je relativno prost s p, kažemo da je* ***kvadratni ostatak modulo p*** *ako kongruencija $x^2 \equiv b \pmod{p}$ ima rješenja. U protivnom, kažemo da je b kvadratni neostatak.*

Primjer 1.3.12. Sada ćemo odrediti kvadratne ostatke i neostatke brojeva 7, 11 i 13.

- Neka je $p = 7$. Prirodni brojevi b takvi da $1 \leq b \leq 6$ su relativno prosti s p pošto je p prost broj. Promotrimo sljedeći niz kongruencija,

$$\begin{aligned} 1^2 &\equiv 1 \pmod 7 & 2^2 &\equiv 4 \pmod 7 \\ 3^2 &\equiv 2 \pmod 7 & 4^2 &\equiv 2 \pmod 7 \\ 5^2 &\equiv 4 \pmod 7 & 6^2 &\equiv 1 \pmod 7 \end{aligned}$$

  Iz prethodnog vidimo da $x^2 \equiv b \pmod 7$ ima rješenje jedino ako je b jedan od sljedećih brojeva $1, 2, 4$. Dakle, 1, 2, 4 su kvadratni ostatci, a 3, 5, 6 kvadratni neostatci modulo 7.

- Neka je $p = 11$. Prirodni brojevi b takvi da $1 \leq b \leq 10$ su relativno prosti s p pošto je p prost broj. Promotrimo sljedeći niz kongruencija,

$$\begin{aligned} 1^2 &\equiv 1 \pmod{11} & 2^2 &\equiv 4 \pmod{11} \\ 3^2 &\equiv 9 \pmod{11} & 4^2 &\equiv 5 \pmod{11} \\ 5^2 &\equiv 3 \pmod{11} & 6^2 &\equiv 3 \pmod{11} \\ 7^2 &\equiv 5 \pmod{11} & 8^2 &\equiv 9 \pmod{11} \\ 9^2 &\equiv 4 \pmod{11} & 10^2 &\equiv 1 \pmod{11} \end{aligned}$$

  Iz prethodnog vidimo da $x^2 \equiv b \pmod{11}$ ima rješenje jedino ako je b jedan od sljedećih brojeva 1, 3, 4, 5, 9. Dakle, 1, 3, 4, 5, 9 su kvadratni ostatci, a 2, 6, 7, 8, 10 kvadratni neostatci modulo 11.

- Neka je $p = 13$. Prirodni brojevi b takvi da $1 \leq b \leq 12$ su relativno prosti s p pošto je p prost broj. Promotrimo sljedeći niz kongruencija,

$$\begin{aligned} 1^2 &\equiv 1 \pmod{13} & 2^2 &\equiv 4 \pmod{13} \\ 3^2 &\equiv 9 \pmod{13} & 4^2 &\equiv 3 \pmod{13} \\ 5^2 &\equiv 12 \pmod{13} & 6^2 &\equiv 10 \pmod{13} \\ 7^2 &\equiv 10 \pmod{13} & 8^2 &\equiv 12 \pmod{13} \\ 9^2 &\equiv 3 \pmod{13} & 10^2 &\equiv 9 \pmod{13} \\ 11^2 &\equiv 4 \pmod{13} & 12^2 &\equiv 1 \pmod{13} \end{aligned}$$

Iz prethodnog vidimo da $x^2 \equiv b \pmod{13}$ ima rješenje jedino ako je b jedan od sljedećih brojeva 1, 3, 4, 9, 10, 12. Dakle, 1, 3, 4, 9, 10, 12 su kvadratni ostatci, a 2, 5, 6, 7, 8, 11 kvadratni neostatci modulo 13.

Sada ćemo iskazati teorem koji nam govori kakav je odnos kvadratnih ostataka i neostataka neparnog prostog broja p. No, prije moramo definirati pojam reduciranog sustava ostataka modulo neki broj.

Definicija 1.3.13. ***Reducirani sustav ostataka modulo*** *m je skup cijelih brojeva r_i sa svojstvom da je $(r_i, m) = 1$, $r_i \neq r_j$ za $i \neq j$, te da za svaki cijeli broj x takav da je $(x, m) = 1$ postoji r_i takav da je $x \equiv r_i \pmod{m}$.*

Teorem 1.3.14. *Neka je p neparan prost broj. Reducirani sustav ostataka modulo p sastoji se od $\frac{p-1}{2}$ kvadratnih ostataka i $\frac{p-1}{2}$ kvadratnih neostataka.*

Dokaz ovog teorema može se pronaći u [4], str. 29, Teorem 3.1.
Sada definiramo Legendreov simbol.

Definicija 1.3.15. *Za neparan prost broj p i cijeli broj b* ***Legendreov simbol*** *definiramo kao:*

$$\left(\frac{b}{p}\right) = \begin{cases} 0, & \text{ako } p \text{ i } b \text{ nisu relativno prosti,} \\ 1, & \text{ako je } b \text{ kvadratni ostatak modulo } p, \\ -1, & \text{ako je } b \text{ kvadratni neostatak modulo } p \end{cases}$$

Ovdje navodimo neka svojstva Legendreovih simbola. Dokazi tih svojstava dani su primjerice u [4], str. 30-32.

Propozicija 1.3.16. *Neka su $a, b \in \mathbb{N}$ i neka je p neparan prost broj. Tada vrijede sljedeća svojstva:*

1. $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$ *(multiplikativnost)*
2. $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$ *ako vrijedi* $a \equiv b \pmod{p}$
3. *Ako je b prost broj različit od p, tada vrijedi*

$$\left(\frac{b}{p}\right) = (-1)^{(b-1)(p-1)/4}\left(\frac{p}{b}\right);$$

   ovo svojstvo se još naziva i Gaussov kvadratni zakon reciprociteta.

4. $\left(\frac{1}{p}\right) = 1$, $\left(\frac{2}{p}\right) = (-1)^{(p^2-1)/8}$
5. $\left(\frac{a^2}{p}\right) = \begin{cases} 1, & \text{ako } p \text{ ne dijeli } a \\ 0, & \text{ako } p \text{ dijeli } a \end{cases}$

Jacobijev simbol proširuje pojam Legendreovog simbola na "nazivnike" koji nisu prosti brojevi. Sada navodimo definiciju.

Definicija 1.3.17. ***Jacobijev simbol****, u oznaci* $\left(\frac{b}{n}\right)$*, definiramo kao umnožak Legendreovih simbola koji odgovaraju prostim faktorima neparnog broja n.*

$$\left(\frac{b}{n}\right) = \left(\frac{b}{p_1}\right)^{a_1} \dots \left(\frac{b}{p_k}\right)^{a_k}$$

gdje je $n = p_1^{a_1} \dots p_k^{a_k}$.

Primijetimo da iz definicije lako možemo provjeriti da vrijede sljedeće tvrdnje.

Lema 1.3.18. *Za Jacobijev simbol vrijedi:*

1. *Ako je n prost broj, onda su Legendreov i Jacobijev simbol jednaki.*
2. *Ako je* $(b, n) > 1$*, onda je* $\left(\frac{b}{n}\right) = 0$*; inače je* $\left(\frac{b}{n}\right) \in \{-1, 1\}$.
3. *Ako je b kvadratni ostatak modulo n, onda je b kvadratni ostatak modulo* p_i *za svaki i. Zato je* $\left(\frac{b}{p_i}\right) = 1$ *za svaki i, pa je i* $\left(\frac{b}{n}\right) = 1$.

No primijetimo da $\left(\frac{b}{n}\right) = 1$ ne povlači da je b kvadratni ostatak modulo n. Na primjer, $\left(\frac{3}{25}\right) = \left(\frac{3}{5}\right)^2 = (-1)^2 = 1$, ali kongruencija $x^2 \equiv 3 \pmod{25}$ nema rješenja jer su kvadratni ostatci modulo 25 elementi skupa $\{0, 1, 4, 6, 9, 11, 14, 16, 19, 21, 24\}$. Da bi b bio kvadratni ostatak modulo n nužno je i dovoljno da su svi $\left(\frac{b}{p_i}\right)$ jednaki 1.

Ovdje navodimo neka svojstva Jacobijevih simbola. Dokazi tih svojstava dani su primjerice u [4], str. 35-36.

Propozicija 1.3.19. *Neka su* $m, n \in \mathbb{N}$ *neparni prirodni brojevi. Tada vrijedi sljedeće:*

1. *Ako je* $(a, n) = 1$*, onda je* $\left(\frac{a^2}{n}\right) = \left(\frac{a}{n^2}\right) = 1$
2. $\left(\frac{-1}{n}\right) = (-1)^{\frac{n-1}{2}}$, $\left(\frac{2}{n}\right) = (-1)^{\frac{n^2-1}{8}}$
3. *Ako vrijedi* $(m, n) = 1$ *tada* $\left(\frac{m}{n}\right)\left(\frac{n}{m}\right) = (-1)^{\frac{m-1}{2}\frac{n-1}{2}}$

Primijetimo da u postupku računanja Legendreovih simbola uz pomoć svojstva 2 iz propozicije 1.3.16 možemo dobiti da je "nazivnik" veći od "brojnika", pa se "brojnik" više ne može reducirati uz pomoć svostva 2. Da bi mogli nastaviti računati Legendreov simbol možemo zamijeniti "brojnik" s nazivnikom (što nam dozvoljava svojstvo 3 propozicije 1.3.16), no prije toga moramo "brojnik" rastaviti na proste faktore. Budući da nije poznat polinomni algoritam za rastavljanje broja na proste faktore, računanje Jacobijevih simbola je puno efikasnije.

Definicija 1.3.20. *Ako je n neparan složen broj, te b cijeli broj takav da je $(b,n) = 1$ i vrijedi*

$$\left(\frac{b}{n}\right) \equiv b^{(n-1)/2} \pmod{n} \tag{1.4}$$

*onda n zovemo **Eulerov pseudoprost broj u bazi** b.*

Neka je n Eulerov pseoudoprost broj u bazi b. Iz $(b,n) = 1$ slijedi da je $\left(\frac{b}{n}\right) = 1$ ili $\left(\frac{b}{n}\right) = -1$. Stoga vrijedi $b^{(n-1)/2} \equiv 1 \pmod{n}$ ili $b^{(n-1)/2} \equiv -1 \pmod{n}$. Pošto za cijele brojeve a, b, c i d takve da $a \equiv b \pmod{m}$ i $c \equiv d \pmod{m}$ vrijedi $ac \equiv bd \pmod{m}$ dobivamo da vrijedi $b^{n-1} \equiv 1 \pmod{n}$. Dakle ako je n Eulerov pseudoprost broj u bazi b tada je on i pseudoprost u bazi b. No obrat ne vrijedi, što pokazujemo u sljedećem primjeru.

Primjer 1.3.21. *Broj* 91 *je pseudoprost u bazi* 3 *jer vrijedi* $(91,3) = 1$ *i jer iz*

$$3^{90} \equiv 27^{30} \equiv (-1)^{30} \equiv 1 \pmod{7}$$
$$3^{90} \equiv 27^{30} \equiv 1^{30} \equiv 1 \pmod{13}$$

proizlazi $3^{90} \equiv 1 \pmod{91}$. *Međutim vrijedi* $3^{45} \equiv 729^7 \times 27 \equiv 27 \pmod{91}$ *(*729 *daje ostatak* 1 *pri djeljenju s* 91*) pa* 91 *nije Eulerov pseudoprost broj u bazi* 3.

Teorem koji ćemo sada navest koristit ćemo u Solovay-Strassenovom algoritmu sa slučajnim elementima. Dokaz se može pronaći primjerice u [6], teorem 4.2.

Teorem 1.3.22. *Označimo sa P_n skup cijelih brojeva b koji su strogo manji od n i za koje vrijedi da je n Eulerov pseudoprost broj u bazi b. Tada za sve neparne složene brojeve n vrijedi*

$$|P_n| \leq \frac{\varphi(n)}{2},$$

gdje je $\varphi(n)$ Eulerova funkcija.

Neka je n neparan prirodni broj te vrijedi $(b,n) = 1$ i $b^{n-1} \equiv 1 \pmod{n}$. Budući da je $n-1$ paran, možemo pokušati "vaditi" drugi korijen iz ove kongruencije, tj. dizati b na potencije $(n-1)/2, (n-1)/4, \ldots, (n-1)/2^s$ (gdje je $t = (n-1)/2^s$ neparan broj). Pretpostavimo da u i-tom koraku prvi put na desnoj strani dobijemo nešto različito od 1, recimo $b^{(n-1)/2^i} \equiv a \pmod{n}$. Znamo da su -1 i 1 jedina rješenja kongruencije $x^2 \equiv 1 \pmod{p}$ kada je p prost broj. Stoga ako je n prost mora vrijediti $a = -1$ jer je $b^{(n-1)/2i-1} \equiv 1 \pmod{n}$. Ova činjenica motivira sljedeću definiciju.

Definicija 1.3.23. *Neka je n neparan složen broj, te neka je $n - 1 = 2^s t$, gdje je t neparan. Ako za cijeli broj b vrijedi*

$$b^t \equiv 1 \pmod{n}$$

ili postoji $r < s$ takav da je

$$b^{2^r t} \equiv -1 \pmod{n}$$

onda kažemo da je n ***jak pseudoprost broj u bazi*** *b.*

Sada navodimo teorem koji nam je potreban da bi lakše shvatili korake prilikom navođenja Miller-Rabinovog algoritma. Dokaz teorema može se pronaći primjerice u [5], str. 216, Teorem 5.15.

Teorem 1.3.24. *Neka je n neparan složen broj. Tada je n jak pseudoprost broj u bazi b za najviše* $(n-1)/4$ *baza b, gdje je* $0 < b < n$.

1.4 Teorija složenosti

U ovom dijelu uvodimo osnovne pojmove iz teorije složenosti koje ćemo kasnije koristiti. Većina definicija i teorema koje koristimo možete pronaći u [17] i [14]. Teorija složenosti je grana teorijskog računarstva u matematici koja se bavi klasificiranjem problema prema njihovoj težini i traženjem odnosa među klasama složenosti.

Složenost algoritma je cijena njegova izvođenja za rješavanje problema iz neke klase. Obično se mjeri u vremenu njegova izvođenja. No, pitanje je kako odrediti vrijeme izvođenja zadanog algoritma. Zbog sve većeg razvoja računala, postojanja različitih programskih jezika i njihovih prevodioca bilo bi teško, gotovo nemoguće odrediti idealnu platformu za testiranje algoritama. Zbog toga se analiza složenosti algoritama bazira na teorijskom računalu. Najčešće se koriste Turingovi strojevi. Turingov stroj je jednostavni apstraktni uređaj za manipulaciju simbolima koji se koristi da bi simulirao logiku bilo kojeg računalnog algoritma. Simboli se nalaze na beskonačnoj traci koja simulira memoriju računala, a simboli su iz nekog unaprijed zadanog alfabeta. Traka je podijeljena na registre i u svakom registru se može nalaziti samo jedan simbol. Turingov stroj također ima glavu za čitanje i konačan skup stanja, uz uvjet da glava za čitanje može promatrati samo jedan registar. Glava čitača može se pomaknuti za jedan registar u lijevo ili desno, može pročitati simboli iz registra i upisati simbol u registar. Pojam Turingovog stroja definirao je Alan Turing 1936. godine.

Da bi mogli opisivati probleme i određivati da li pripadaju nekoj klasi treba nam pojam jezika. Sada navodimo osnovne pojmove vezane uz jezike.

Definicija 1.4.1. *Nega je* Γ *neki alfabet (tj. proizvoljan neprazan skup). Elemente skupa* Γ *nazivamo* ***simboli****. Svaki konačan niz elementa iz* Γ *nazivamo* ***riječ****. Sa* Γ^* *označavamo skup svih riječi skupa* Γ*. Pretpostavljamo da za svaki alfabet* Γ *skup* Γ^* *sadrži praznu*

riječ, tj. riječ koja ne sadrži niti jedan element od Γ. *Praznu riječ označavamo sa* ϵ. *Ako je alfabet* Γ *konačan ili prebrojiv skup, tada je skup svih riječi* Γ^* *prebrojiv skup. Svaki podskup skupa* Γ^* *nazivamo* ***jezik***. *Duljina neke riječi* α *je broj simbola koje ta riječ sadrži. Duljinu riječi označavamo sa* $|\alpha|$.

Sada navodimo formalne definicije deterministčkog i nedeterminističkog Turingovog stroja.

Definicija 1.4.2. ***Deterministički Turingov stroj*** *definiramo kao uređenu sedmorku* $(Q, \Sigma, \Gamma, \delta, q_0, q_{DA}, q_{NE})$, *gdje je redom:*

- Q *je konačan skup čije elemente nazivamo* ***stanja****;*
- Σ *je konačan skup, čije elemente nazivamo* ***ulazni simboli***. *Pretpostavljamo da* Σ *ne sadrži "prazan simbol" koji označavamo sa* ϵ*;*
- Γ *je konačan skup kojeg nazivamo* ***alfabet Turingovog stroja***. *Pretpostavljamo da je prazan simbol* ϵ *element od* Γ, *te* $\Sigma \subseteq \Gamma$*;*
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, D, S\}$ *je proizvoljna funkcija koju nazivamo* ***funkcija prijelaza*** *(L znači pomak lijevo na traci, D pomak desno na traci, a S znači da glava za čitanje ostaje na istom mjestu);*
- q_0 *je element iz* Q *i nazivamo ga* ***početno stanje****;*
- q_{DA} *je element skupa* Q *i nazivamo ga* ***stanje prihvaćanja****;*
- q_{NE} *je element skupa* Q *i nazivamo ga* ***stanje odbijanja***.

Definicija 1.4.3. ***Nedeterministički Turingov stroj*** *definiramo kao uređenu sedmorku* $(Q, \Sigma, \Gamma, \delta, q_0, q_{DA}, q_{NE})$, *gdje simboli* Q, Σ, Γ, q_0, q_{DA} *i* q_{NE} *imaju isto značenje kao i kod deterministčkih Turingovih strojeva. No, sada je promijenjena kodomena funkcije prijelaza, tj. imamo* $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, D, S\})$.

Nedeterminizam se najbolje očituje u tome što stroj s istim ulazom i u istom stanju može postupati na bitno različite načine. Izračunavanje nedeterminističkog Turingovog stroja je stablo čije grane odgovaraju različitim mogućnostima postupanja. Ako neka grana izračunavanja vodi prihvaćajućem stanju kažemo da stroj prihvaća taj ulaz.

Pošto sada znamo što je riječ i što je Turingov stroj možemo objasniti što je to konfiguracija Turingovog stroja. Pretpostavimo da za stanje q Turingovog stroja M vrijedi da se na njegovoj traci nalaze dvije riječi u i v, tada konfiguraciju Turingovog stroja M zapisujemo kao uqv ako vrijedi da je sadržaj trake uv i pozicija glave Turingovog stroja je na prvom

simbolu riječi v.

Da bi jezike (probleme) mogli svrstavati u određene klase potrebani su nam pojmovi kao što su Turing-odlučivost, funkcije složenosti i vremenska složenost. Ove pojmove ćemo u nastavku uvesti kroz par definicija. Najprije definirajmo pojam Turing prepoznatljivosti.

Definicija 1.4.4. *Kažemo da Turingov stroj T s alfabetom Γ **prepoznaje neku riječ** $w \in \Gamma^*$ ako taj Turingov stroj staje u konačno mnogo koraka u završnom stanju q_{DA} kada je na početku rada stroja na traci zapisana samo riječ w. Za proizvoljan Turingov stroj T sa $L(T)$ označavamo skup svih riječi koje T prepoznaje.*
*Kažemo da neki Turingov stroj T **prepoznaje jezik** L ako vrijedi $L = L(T)$. Za neki jezik L kažemo da je **Turing-prepoznatljiv**, ili samo kratko **prepoznatljiv**, ako postoji Turingov stroj koji ga prepoznaje.*

Primijetimo da je svaki konačan jezik Turing-prepoznatljiv. Sada dajemo definiciju Turing-odlučivosti.

Definicija 1.4.5. *Za neki jezik L kažemo da je **Turing-odlučiv**, ili kratko **odlučiv**, ako postoji Turingov stroj T koji ga prepoznaje, te za svaku riječ $w \in \Gamma \setminus L$ stroj staje u završno stanju q_{NE}.*

Da bi bili u stanju uspoređivati i analizirati različite algoritme trebamo izračunati njihovu složenost koja se ovisno o duljini ulaznih podataka može prikazati kao funkcija sa skupa $\mathbb{N}$ u $\mathbb{R}$. Takve funkcije nazivamo **funkcije složenosti**.
U slučaju da imamo dva različita algoritma koji rješavaju isti problem trebamo znati uspoređivati njihove funkcije složenosti. Da bi znali uspoređivati različite funkcije složenosti potrebna nam je sljedeća definicija.

Definicija 1.4.6. *Neka su $f, g : \mathbb{N} \to \mathbb{R}$ dvije realne funkcije nad skupom prirodnih brojeva. Tada pišemo:*

> *$f(n) = O(g(n))$ ako postoji konstanta $c \in \mathbb{R}$, $c > 0$ takva da za sve dovoljno velike n vrijedi $|f(n)| \leq c|g(n)|$.*

Definirajmo sada vremensku složenost deterministčkog i nedeterminističkog Turingovog stroja.

Definicija 1.4.7. *(Vremenska složenost)*

> ***Vremenska složenost deterministčkog** stroja T je funkcija $time_T : \mathbb{N} \to \mathbb{N} \cup \{\infty\}$, gdje je $time_T(n)$ maksimalan broj koraka koje stroj T napravi za svaki ulazni podatak duljine n.*

Vremenska složenost nedeterminističkog stroja T je funkcija $time_T : \mathbb{N} \to \mathbb{N} \cup \{\infty\}$, *gdje je* $time_T(n)$ *maksimalan broj koraka, uzimajući u obzir sve grane, koje stroj T napravi za svaki ulazni podatak duljine n.*

Ako je $time_T$ *vremenska složenost stroja T (deterministčkog ili nedeterminističkog) tada kažemo da stroj t radi u vremenu* $time_T$, *tj. da je T* $time_T$***-vremenski složen Turingov stroj***.

Definirajmo klase složenosti *PTIME* i *NPTIME*.

Definicija 1.4.8. *Neka je* $f : \mathbb{N} \to \mathbb{R}^+$ *proizvoljna funkcija. Klasa vremenske složenosti* ***DTIME(****$f(n)$****)*** *je skup svih jezika koji su odlučivi s nekim* $O(f(n))$*-vremenskim složenim deterministčkim Turingovim strojem.*

Klasa vremenske složenosti ***NTIME(****$f(n)$****)*** *je skup svih jezika koji su odlučivi s nekim* $O(f(n))$*- vremenskim složenim nedeterminističkim Turingovim strojem.*

Sada ćemo definirati polinomne i eksponencijalne Turingove strojeve.

Definicija 1.4.9. *Neka je T Turingov stroj (deterministički ili nedeterministički). Kažemo da je stroj T:*

- ***(vremenski) polinoman*** *ako postoji neki polinom f tako da vrijedi* $time_T(n) = O(f(n))$.
- ***(vremenski) eksponencijalan*** *ako postoji eksponencijalna funkcija g tako da vrijedi* $time_T(n) = O(g(n))$.

Sada smo uveli sve pojmove koji su nam potrebni da bi definirani klase složenost P i NP.

Definicija 1.4.10. *S* ***PTIME*** *ili samo kratko* ***P****, označavamo klasu svih jezika koji su odlučivi na nekom deterministčkom Turingovom stroju vremenske složenosti* $O(n^k)$*, za neki* $k \in \mathbb{N}$. *Dakle, vrijedi*

$$P = \bigcup_{k \in \mathbb{N}} DTIME(n^k)$$

Definicija 1.4.11. *Sa* ***NPTIME****, ili samo kratko s* ***NP****, označavamo klasu svih jezika koji su odlučivi na nekom nedeterminističkom Turingovom stroju vremenske složenosti* $O(n^k)$*, za neki* $k \in \mathbb{N}$. *Dakle, vrijedi*

$$NP = \bigcup_{k \in \mathbb{N}} NTIME(n^k)$$

Sljedeći teorem je jako bitan jer jako dobro opisuje klasu NP. U suštini govori da je za svaki problem iz klase NP lako provjeriti je li nešto njegovo riješenje. Koristit ćemo ga kada budemo dokazivali Prattov teorem.

Teorem 1.4.12. ***(O certifikatu)*** *Za svaki jezik L vrijedi:*

$$L \in NP \Leftrightarrow (\exists R \in P)\,(\exists k \in \mathbb{N})$$
$$L = \{x : (\exists c)(|c| = O(|x|^k) \wedge R(x, c))\}$$

Riječ c nazivamo ***certifikat*** *za riječ x, a jezik koji sadrži sve certifikate nazivamo certifikat za jezik L.*

Za jezik $L \subseteq \Gamma^*$ označimo sa $\overline{L}$ njegov komplement $\Gamma^* \setminus L$. Za svaku klasu složenosti C možemo definirati klasu co-C kao co-$C = \{\overline{L} \mid L \in C\}$.

Sada navodimo definiciju jezika kojim ćemo se najviše baviti u ovom radu.

Definicija 1.4.13. *Jezike* ***PRIMES*** *i* $\overline{\textbf{\textit{PRIMES}}}$ *definiramo na sljedeći način*

$$PRIMES = \{p \in \mathbb{N} \mid p \text{ je prost}\}$$

$$\overline{PRIMES} = \{a \in \mathbb{N} \mid a \text{ je složen}\}$$

Složenost algoritama

U sljedećoj propoziciji navest ćemo složenost nekih algoritama iz teorije brojeva koje ćemo koristiti u drugim dijelovima ovog rada. Vrijeme izvođenja algoritma promatramo kao broj operacija koje algoritam izvede za ulazni podatak koji je prirodan broj koji je zapisan u binarnom brojevnom sustavu.

Propozicija 1.4.14. *Neka su a, b i n prirodni brojevi takvi da vrijedi* $1 \leq a \leq b \leq n$. *Tada vrijedi:*

1. *Produkt ab možemo izračunati u vremenu* $O(\log_2^2 n)$.
2. *Prirodni broj* $\lfloor b/a \rfloor$ *možemo izračunati u vremenu* $O(\log_2^2 n)$.
3. *Prirodni broj* $b \pmod a$ *možemo izračunati u vremenu* $O(\log_2^2 n)$.
4. *Za prirodni broj n i prirodne brojeve* $q_1, \ldots, q_k$ *takve da vrijedi* $k \leq \log_2 n$, *u vremenu* $O(\log_2^4 n)$ *možemo provjeriti da li postoje* $a_1, \ldots, a_k \in \mathbb{N}$ *za koje vrijedi* $n = \sum_{i=1}^{k} q_i^{a_i}$.
5. *Neka su* $a \geq 2$, $b \geq 1$ *i* $r \geq 2$ *prirodni brojevi. Vrijednost od* $a^b \pmod r$ *možemo izračunati u vremenu* $O(\log_2^2 n + \log_2 b \log_2^2 r)$, *gdje je* $n = \max(a, r)$.

6. *Neka su $a \geq 2$ i $b \geq 1$ prirodni brojevi. Potenciju $n = a^b$ možemo izračunati u vremenu $O(\log^2 n \log b)$.*

7. *Neka su a i n prirodni brojevi. Za računanje Jacobijevog simbola $\left(\frac{a}{n}\right)$ treba nam $O(\log_2^2 n)$ operacija.*

Dokaz. Sada ćemo za ilustraciju dokazati tvrdnju 5, a dokazi za ostale algoritme mogu se pronaći u [15], str. 21-25. Sljedeći algoritam računa $a^b \pmod r$:

Ulaz: Pozitivni prirodni brojevi a, b i r
Izlaz: $a^b \pmod r$
$x := a \pmod r$; $y := b$; $z := 1$;
while $y \neq 0$ **do**
  if *y je paran* **then**
    $y := y/2$; $x := x^2 \pmod r$;
  else
    $y := y - 1$; $z := zx \pmod r$;

U bilo kojem koraku algoritma vrijednost varijabli x i z su cijeli brojevi u rasponu od $[0, r-1]$. Iz tvrdnje 3 ove propozicije za inicijalizaciju varijable x potrebno nam je $O(\log^2 n)$ koraka, gdje je $n = \max(a, r)$. Broj iteracija koje čini while petlja je $O(\log b)$, a iz tvdrnji 2, 3 i 4 ove propozicije slijedi da u svakoj iteraciji se napravi najviše $O(\log^2 r)$ koraka. Dakle ukupno je potrebno $O(\log^2 n + \log b \log^2 r)$ koraka. □

1.5 Algoritmi sa slučajnim elementima

U ovom diplomskom radu kao standardni model izračunavanja uzeli smo deterministički (nedeterministički) Turingov stroj, no u praksi to ipak nije najidealnije rješenje. Pogledamo li primjerice simulacije procesa u kojima je potrebna doza slučajnosti kao što su vremenske prognoze, nuklearne fisije ili burze, model determinističkog (nedeterminističkog) Turingovog stroja nam neće biti dovoljan. Svi procesi u kojima je potrebna doza slučajnosti koriste razne generatore slučajnih brojeva, pa bi za takve procese mnogo bolji model izračunavanja bio Turingov stroj u kojem postoji neki oblik generatora slučajnih brojeva.

Algoritam sa slučajnim elementima je algoritam koji je dizajniran tako da u svom izvršavanju koristi izlaz (rezultat) nekog slučajnog procesa.

Turingov stroj sa slučajnim elementima

Turingov stroj sa slučajnim elementima definiramo na isti način kao i nedeterministički Turingov stroj samo što je definicija njegova izračunavanja dosta drugačija. Definicija

izračunavanja nedeterminističkog Turingovog stroja odnosi se na pitanje da li postoji izračunavanje stroja koje počevši s konkretnim ulazom dolazi do određene konfiguracije. Izračunavanje Turingovog stroja sa slučajnim elementima odnosi se na vjerojatnost da dođemo do određene konfiguracije s tim da je u svakom koraku sljedeći korak izabran uniformno među svim mogućim koracima koji su nam na raspolaganju. Stoga, ako tranzicijska funkcija stroja preslika trenutni par simbola u više mogućih trojki, tada je u izračunavanju sa slučajnim elementima jedna od mogućih trojki izabrana na slučajan način i sljedeća konfiguracija je određena prema tome.

Sada ćemo objasniti razliku između nedeterminističkog modela Turingovog stroja i modela Turingovog stroja sa slučajnim elementima. Kod nedeterminističkog Turingovog stroja uzimamo u obzir postojanje točnog niza izbora (koji vode željenom ishodu), i odbacujemo pitanje kako su ti izbori načinjeni. Kod Turingovog stroja sa slučajnim elementima u svakom koraku radimo uniformno slučajan izbor unutar skupa unaprijed određenih mogućnosti i uzimamo u obzir vjerojatnost dosezanja željenog rezultata. Prije nego damo definiciju Turingovog stroja sa slučajnim elementima spomenimo u kakvom je odnosu sa algoritmima sa slučajnim elementima. Algoritme sa slučajnim elementima modeliramo uz pomoć Turingovih strojeva sa slučajnim elementima da bi mogli odrediti njihovu složenost i pripadnost određenim klasama složenosti.

Definicija 1.5.1. ***Turingov stroj sa slučajnim elementima*** *je vrsta nedeterminističkog Turingovog stroja u kojem se svaki nedeterministički korak naziva* ***korak bacanja novčića*** *i ima dva legalna sljedeća prijelaza. Svakoj grani b stroja M pri njegovom izračunavanju na ulazu x dodjeljujemo vjerojatnost na sljedeći način.*
Vjerojatnost na grani b definiramo kao:

$$P(b) = 2^{-k},$$

pri čemu je k broj ***bacanja novčića*** *koje stroj napravi na grani b.*
Vjerojatnost da M prihvaća neki ulaz x definiramo kao

$$P(M \text{ prihvaća } x) = \sum_{b \text{ je prihvaćajuća grana}} P(b).$$

Dakle, vjerojatnost da M prihvaća x je vjerojatnost da dosegnemo prihvaćajuću konfiguraciju ako pokrenemo M na ulazu x i u svakom koraku dopuštamo stroju da na slučajan način bira sljedeću konfiguraciju. Vrijedi

$$P(M \text{ odbija } x) = 1 - P(M \text{ prihvaća } x).$$

Kada Turingov stroj sa slučajnim elementima prepoznaje neki jezik, on mora prihvatiti sve riječi koje su unutar tog jezika i odbiti one koje nisu, no sad dopuštamo stroju da ima malu

vjerojatnost greške. Za $0 \leq \epsilon \leq \frac{1}{2}$ kažemo da polinomni Turingov stroj M sa slučajnim elementima **prihvaća jezik A s vjerojatnošću greške** ϵ ako

1. $w \in A$ povlači $P(M \text{ prihvaća } w) \geq 1 - \epsilon$, i

2. $w \notin A$ povlači $P(M \text{ odbacuje } w) \geq 1 - \epsilon$.

Drugim riječima, vjerojatnost da ćemo dobiti pogrešan odgovor kada simuliramo M je najviše ϵ. Uzet ćemo u obzir granice vjerojatnosti koje ovise o duljini ulaznog podatka n. Naprimjer, vjerojatnost greške $\epsilon = 2^{-n}$ nam pokazuje da imamo eksponencijalno malu vjerojatnost greške.

Nas u ovom dijelu zanimaju algoritmi sa slučajnim elementima koje rade u efikasnom vremenu. Vremensku složenost Turingovog stroja sa slučajnim elementima mjerimo na jednak način kao i kod nedeterminističkog Turingovog stroja, promatrajući onu granu izračunavanja koja daje najlošije rezultate za svaki ulazni podatak.

Sada ćemo definirati klasu složenosti *BPP* ili Bounded-error Probabilistic Polynomial-time.

Definicija 1.5.2. ***BPP*** *je klasa svih jezika prepoznatljivih polinomnim Turingovim strojem sa slučajnim elementima s vjerojatnošću greške* $\frac{1}{3}$.

Sljedeća lema nam tvrdi kako je izbor konstante $\frac{1}{3}$ nevažan pošto bilo koja konstanta između 0 i $\frac{1}{2}$ daje istu klasu što nam omogućava da na jednostavan način grešku načinimo eksponencijalno malom. Primijetimo da će algoritam sa slučajnim elementima čija je vjerojatnost greške 2^{-100} prije dati pogrešan odgovor zbog kvara na hardveru računala na kojem se izvršava nego što bi teorijski mogao dati pogrešan odgovor.

Lema 1.5.3. *Neka je* ϵ *proizvoljni broj iz* $\langle 0, \frac{1}{2} \rangle$. *Neka je poly(n) proizvoljni polinom. Tada za polinomni Turingov stroj sa slučajnim elementima* M_1 *koji radi s vjerojatnošću greške* ϵ *postoji ekvivalentni polinomni Turingov stroj* M_2 *koji radi s vjerojatnošću greške* $2^{-poly(n)}$.

Potpuni dokaz prethodne leme i više detalja o algoritmima sa slučajnim elementima može se primjerice pronaći u [11].

1.6 Povijest problema PRIMES

Problem ispitivanja je li zadani prirodni broj prost te rastavljanje složenih brojeva na proste faktore, smatraju se najvažnijim problemima u aritmetici.

Po definiciji neki prirodan broj je prost ako je djeljiv jedino sa 1 i samim sobom. Iz definicije prostog broja lako možemo konstruirati algoritam u kojem n dijelimo sa svim brojevima m takvim da vrijedi $2 \leq m \leq \sqrt{n}$. Ako za neki broj m uspijemo podijeliti n tada je n složen, inače je prost. No ovaj algoritam nije efikasan jer nije kao što bi očekivali polinomne složenosti već eksponencijalne što ćemo sada pokazati.

Propozicija 1.6.1. *Složenost algoritma koji ispituje djeljivost zadanog prirodnog broja n sa svakim prirodnim brojem manjim od $\sqrt{n}$ je eksponencijalna.*

Dokaz. Složenost algoritma ne promatramo po vrijednosti njegova ulaza već po duljini u binarnoj bazi. U svakom od $\sqrt{n}$ koraka algoritma moramo podijeliti dva broja za što nam po propoziciji 1.4.14 treba $O(\log_2^2 n)$ operacija. Dakle, složenost algoritma je $O(\sqrt{n}\log_2^2 n)$. No duljina broja n u bazi 2 je $d = \log_2 n$, što nam daje da je složenost algoritma $O(2^{\frac{d}{2}}d^2)$. Dakle algoritam je eksponencijalan u odnosu na duljinu ulaznog podatka. □

Prattov teorem

Prattov certifikat je certifikat prostosti koji se temelji na obratu malog Fermatovog teorema koji je još poznat kao Lehmerov teorem [9], str. 330, teorem 2.

Teorem 1.6.2. ***(Lehmerov teorem)*** *Ako vrijedi $r^x \equiv 1 \pmod{n}$ za $x = n - 1$, te vrijedi $r^{\frac{x}{q}} \not\equiv 1 \pmod{n}$ za sve proste dijelitelje q od n tada je n prost.*

Broj r iz Lehmerovog teorema nazivamo svjedokom prostosti za n.

Pratt je 1975. godine pokazao da se uz pomoć Lehmerovog teorema može konstruirati nedeterministički algoritam koji rekurzivno generira certifikate za proste faktore od $n - 1$ i daje nam odgovor da li je n prost broj. Kao posljedica ovog rezultata Pratt je postao prvi čovjek koji je pokazao da problem ispitivanja prostosti leži u klasi složenosti NP.

Teorem 1.6.3. ***(Prattov teorem)*** $PRIMES \in NP \cap coNP$.

Dokaz. Prvo ćemo pokazati da je PRIMES $\in$ coNP, odnosno $\overline{PRIMES} \in$ NP. Da bi to dokazali potrebno je za svaki element iz jezika $\overline{PRIMES}$ pronaći certifikat. Znamo da je jezik $\overline{PRIMES}$ jezik koji sadrži sve prirodne brojeve koji nisu prosti, tj sve brojeve koji su složeni. Za svaki složeni broj postoji njegov djelitelj koji je različit od 1 ili njega samog. Taj broj uzmemo za certifikat. U polinomnom vremenu možemo provjeriti da certifikat dijeli zadani broj (po lemi 1.4.14 za dijeljenje dva broja treba nam $O(\log_2^2 n)$ operacija).

Da bi pokazali da je PRIMES $\in$ NP, pokazat ćemo da za svaki prost broj postoji certifikat $C(p)$ polinomne duljine i koji možemo provjeriti u polinomnom vremenu.
Prepostavimo da je p prost broj. Certifikat ćemo konstruirati tako da sadrži r iz teorema

1.6.2. Iz propozicije 1.4.14 slijedi da relaciju $r^{p-1} \equiv 1 \pmod{p}$ možemo provjeriti u polinomnom vremenu. No sam r nam nije dovoljan za certifikat jer vrijedi primjerice

$$20^{21-1} \equiv 1 \pmod{21}.$$

(Primijetimo da je $20 \equiv -1 \pmod{21}$, i onda $20^{20} \equiv (-1)^{20} \equiv \pmod{21}$; ali 21 nije prost broj). Prisjetimo li se teorema 1.4.12 vidimo da nam nedostaju i svi prosti dijelitelji od $p-1$. Dakle, naš certifikat bi imao sljedeći oblik

$$C(p) = (r; q_1, \ldots, q_k),$$

gdje su q_i prosti dijelitelji od $p-1$.
No i ovakva definicija certifikata nam nije dovoljna. Uzmimo za primjer broj 91 i njegov sljedeći certifikat za prostost:

$$C(91) = (10; 2, 45).$$

Uistinu vrijedi sljedeće, $10^{90} \equiv 1 \pmod{91}$, jer dijeljenjem 90 sa 2 i 45 dobivamo 1 i naposljetku $10^{\frac{90}{2}} \not\equiv 1 \pmod{91}$ i $10^{\frac{90}{45}} \not\equiv 1 \pmod{91}$. No vrijedi $91 = 7 \times 13$, tj. 91 je složen.
Problem je u tome što nas $C(p) = (r; q_1, \ldots, q_k)$ ne uvjerava da su q_i-ovi prosti brojevi (45 je složen), stoga u $C(p)$ moramo dodati i certifikate prostosti za svaki q_i. Rekurzija staje kad je $p = 2$ i kad nema prostih dijelitelja od $p-1$. Dakle certifikat da je p prost broj je sljedeći:

$$C(p) = (r; q_1, C(q_1), \ldots, q_k, C(q_k)).$$

Npr. evo jedan certifikat da je 67 prost broj,

$$C(67) = (2; 2, 3, (2; 2), 11, (8; 2, 5, (3; 2))),$$

gdje je broj prije znaka ";" r (u ovom slučaju 2), nakon kojeg zapisujemo proste djelitelje od $p-1$ (tj. 66, a prosti djeljitelji su 2, 3 i 11) te rekurzivno dodajemo i njhove certifikate koji se nalaze unutar oblih zagrada. Certifikat za 2 nam ne treba jer znamo da je prost broj. Par $(2; 2)$ je certifikat za 3, a $(8; 2, 5, (3; 2))$ je certifikat za 11. Ako je p prost broj, tada iz teorema 1.4.12 i indukcije odgovarajući $C(p)$ postoji, a ako p nije prost tada teorem 1.4.12 povlači da certifikat ne postoji.

Preostaje dokazati da je duljina certifikata dovoljno mala i da se dovoljno brzo može provjeriti njegova točnost. Za proizvoljni broj n njegova duljina u binarnoj bazi iznosi $\lfloor \log_2 n \rfloor + 1$. Prvo ćemo indukcijom po p pokazati da duljina njegovog certifikata $C(p)$ najviše iznosi $4 \log_2^2 p$. Tvrdnja vrijedi za $p = 2$ i $p = 3$. Za broj 2 znamo da je prost broj stoga je duljina njegova certifikata 0 što je manje od $4 \log_2^2 2 = 4$. Duljina certifikata $C(3) = (2; 2)$ iznosi 7 (2 puta po 2 bita za broj 2, jedan bit za separator i još 2 bita za zagrade) što je manje od $4\lfloor \log_2^2 3 \rfloor = 8 < 4 \log_2^2 3$. Za proizvoljni p, $p-1$ će imati najviše

$k \leq \lfloor \log_2 p \rfloor$ prostih dijelitelja $q_1 = 2, \ldots, q_k$. Certifikat za $C(p)$ će sadržavati 2 zagrade, $2k - 1 < 2\lfloor \log_2 p \rfloor - 1$ separatora (po jedan prije svakog q_i-a i $C(q_i)$-a osim za 2), broj r koji zauzima najviše $\lfloor \log_2 p \rfloor + 1$ bitova, q_i-eve za $1 \leq i \leq k$ što iznosi najviše $2\lfloor \log_2 p \rfloor$ bitova jer iz svojstava logaritamske funkcije i funkcije pod ($\lfloor \cdot \rfloor$) vrijedi

$$\begin{aligned} \sum_{i=1}^{k} (\lfloor \log_2 q_i \rfloor + 1) = \sum_{i=1}^{k} \lfloor \log_2 q_i \rfloor + k &\leq \left\lfloor \sum_{i=1}^{k} \log_2 q_i \right\rfloor + \lfloor \log_2 p \rfloor \\ &= \left\lfloor \log_2 \prod_{i=1}^{k} q_i \right\rfloor + \lfloor \log_2 p \rfloor \\ &\leq \lfloor \log_2 (p-1) \rfloor + \lfloor \log_2 p \rfloor \\ &\leq 2\lfloor \log_2 p \rfloor \end{aligned} \tag{1.5}$$

Sada uz pretpostavku indukcije, tj $|C(q_i)| \leq 4 \log_2^2 q_i$ imamo

$$|C(p)| \leq 5\lfloor \log_2 p \rfloor + 2 + 4 \sum_{i=2}^{k} \log_2^2 q_i.$$

Za pozitivne brojeve $a_1, \ldots, a_n$ očito vrijedi $a_1^2 + \cdots + a_n^2 \leq (a_1 + \cdots + a_n)^2$. Zbog prethodne relacije i svojstava logaritamske funkcije vrijedi sljedeće

$$\sum_{i=2}^{k} \log_2^2 q_i \leq \left(\sum_{i=2}^{k} \log_2 q_i \right)^2 = \left(\log_2 \prod_{i=2}^{k} q_i \right)^2 \leq \left(\log_2 \frac{p-1}{2} \right)^2 = (\log_2 (p-1) - 1)^2 < (\log p - 1)^2 .$$

Sada vrijedi

$$|C(p)| < 5 \log_2 p + 2 + 4(\log_2 p - 1)^2 = 4 \log_2^2 p - 3 \log_2 p + 6,$$

što je za $p \geq 5$ manje od $4 \log_2^2 p$ jer vrijedi $6 - 3 \log_2 5 \approx -1$, a $\log p$ je rastuća funkcija. Dakle dokazali smo da je certifikat polinomne duljine.

Pošto smo dokazali da je duljina certifikata za proizvoljni p najviše $4 \log^2 p$ slijedi da je broj certifikata koje rekurzivno trebamo priložiti da bi provjerili $C(p)$ polinomno ograničen. Za provjeru svakog od tih certifikata po teoremu 1.6.2 treba provjeriti da li vrijedi:

- $r^{p-1} \equiv 1 \pmod{p}$
- $r^{\frac{p-1}{q_i}} \not\equiv 1 \pmod{p}$

- $p-1$ može reducirati na 1 uzastopnim dijeljenjem s brojevima q_i.

Iz propozicije 1.4.14 slijedi da te provjere možemo obaviti u polinomnom vremenu. Dakle certifikat $C(p)$ možemo provjeriti u polinomnom vremenu. □

Fermatov algoritam sa slučajnim elementima

Teoremi koji u potpunosti karakteriziraju proste brojeve nisu jednostavni za provjeru, zato se u praksi često koriste testovi prostosti za koje vrijedi da ako broj ne zadovolji kriterij onda je sigurno složen, a ako ga zadovolji onda je "vjerojatno prost", što znači da je mala vjerojatnost da je složen.
Efikasniji testovi od onog iz leme 1.6.1 zasnivaju se na nekim važnim svojstvima prostih brojeva. Prvo takvo svojstvo je mali Fermatov teorem 1.3.1.
Uz pomoć teorema 1.3.6 lako možemo konstruirati algoritam sa slučajnim elementima. Ako pretpostavimo da postoji baza b za koju ne vrijedi (1.3), te ako uzmemo k slučajno odabranih baza i n zadovolji test (1.3) za sve njih, onda je po Teoremu 1.3.6 vjerojatnost da je n složen manja ili jednaka od $1/2^k$. Nažalost, nedostatak ovog testa je u tome što postoje složeni brojevi koji su pseudoprosti u svim bazama b.

Solovay–Strassenov algoritam sa slučajnim elementima

Solovay-Strassenov test je zasnovan na Eulerovim pseoudoprostim brojevima. Da bi u potpunosti razumjeli algoritam potrebno nam je dobro poznavanje određenih teorema i definicija iz teorije brojeva, no sama upotreba algoritma je jednostavna. Vrlo zanimljiva značajka Solovay-Strassenovog testa je ta da nam daje vrlo brzo odgovor, pri čemo je velika vjerojatnost da je to stvarno točan odgovor.

Teorem 1.3.22 predstavlja osnovu za Solovay-Strassenov test prostosti. U osnovi, teorem kaže da je barem polovica brojeva manjih od n koji su relativno prosti sa n svjedok njegove složenosti (svjedok u smislu da ako ga slučajnim izborom odaberemo u algoritmu, dokazat ćemo složenost od n-a i algoritam staje).

Sada navodimo korake Solovay-Strassenovog alogritma:

1. Neka je n neparan broj za kojeg želimo ustanoviti je li prost ili složen. Neka je k prirodan broj koji određuje točnost testa. Odaberimo na slučajan način k brojeva b takvih da je $0 < b < n$.

2. Za svaki od tih b-ova izračunamo obje strane izraza 1.4. Za računanje $b^{(n-1)/2} \pmod n$ iz propozicije 1.4.14 slijedi da nam treba $O(\log_2^2 n)$ operacija. Za računanje Jacobijevog simbola $\left(\frac{b}{n}\right)$ po propoziciji 1.4.14 nam treba $O(\log_2^2 n)$ operacija.

3. Ako brojevi $b^{(n-1)/2}$ i $\left(\frac{b}{n}\right)$ nisu kongruentni modulo n, onda znamo da je n složen i test staje. Inače uzimamo idući b.

4. Ako n prođe test za k različitih b-ova, onda je vjerojatnost da je n složen manja ili jednaka $\frac{1}{2^k}$. Stoga, ako je k dovoljno velik, n proglašavamo "vjerojatno prostim".

Miller–Rabinov algoritam sa slučajnim elementima

Miller-Rabinov test prostosti je test koji je sličan Fermatovom algoritmu sa slučajnim elementima i Solovay-Strassenovom algoritmu. U originalnoj verziji Miller je 1976. konstruirao deterministički algoritam, ali je ovisio o proširenoj Riemanovoj hipotezi. Rabin ga je modificirao u algoritam sa slučajnim elementima koji ne ovisi o proširenoj Riemanovoj hipotezi. (vidi [12]).

Miller-Rabinov test se temelji na jakim pseudoprostim brojevima, a teorem 1.3.24 nam daje osnovu za Miller-Rabinov test prostosti.

Sada navodimo korake Miller-Rabinovog algoritma sa slučajnim elementima:

1. Neka je n neparan broj za kojeg želimo ustanoviti je li prost ili složen. Neka je $n - 1 = 2^s t$, gdje je t neparan.

2. Na slučajan način odaberemo b, $0 < b < n$. Izračunamo $b^t \pmod{n}$.

3. Ako dobijemo 1 ili -1, zaključujemo da je n prošao test, te biramo sljedeći b.

4. U protivnom, uzastopno kvadriramo b^t modulo n sve dok ne dobijemo rezultat -1.

5. Ako dobijemo -1, ond je n prošao test, te biramo sljedeći b.

6. Ako nikad ne dobijemo -1, tj. ako dobijemo da je $b^{2^{r+1}t} \equiv 1 \pmod{n}$ onda sigurno znamo da je n složen.

7. Ako n prođe test za k b-ova, onda je vjerojatnost da je n složen manja ili jednaka $\frac{1}{4^k}$.

Uz pomoć Miller-Rabinovog algoritma lako možemo dokazati sljedeći teorem.

Teorem 1.6.4. *Jezik **PRIMES** pripada klasi složenosti **BPP**.*

Dokaz prethodnog teorema može se pronaći primjerice u [14], str. 375, teorem 10.9.

Adleman-Pomerance-Rumelyev test prostosti

Algoritmi sa slučajnim elementima koji određuju da li je prirodni broj prost bili su poznati godinama sve dok 1983.g Adleman, Pomerance i Rumely nisu konstruirali deterministički "skoro polinomni" algoritam koji daje dokaz prostosti ili složenosti. Složenost algoritma je $(\log n)^{O(\log\log\log n)}$. Eksponent u izrazu raste jako sporo što opravdava ranije spomenuti izraz "skoro polinomni". Svi prijašnji deterministički algoritmi za testiranje prostosti bili su eksponencijalni tako da je ovaj algoritam vrlo značajan. U svojim koracima algoritam koristi ciklotomska polja (vidi [1]).

Atkin-Goldwasser-Kilian-Morainov certifikat prostosti

Godine 1986. Goldwasser i Kilian opisali su novi certifikat koji se temelji na teoriji eliptičkih krivulja. Atkin i Morain su taj certifikat uzeli kao osnovu za Atkin-Goldwasser-Kilian-Morain certifikat prostosti. Kao što se Prattov certifikat temelji na Lehmerovom teoremu tako se i Atkin-Goldwasser-Kilian-Morain certifikat temelji na sljedećem teoremu Goldwassera i Kiliana:

Teorem 1.6.5. *Pretpostavimo da imamo:*

- *pozitivni cijeli broj n koji nije djeljiv sa* 2 *ili* 3
- M_x, M_y, A, B *iz* $\mathbb{Z}_n$ *za koje vrijedi* $M_y^2 = M_x^3 + AM_x + B$ *te da je* $4A^2 + 27B^2$ *relativno prosto sa n*
- *prost broj* $q > n^{\frac{1}{2}} + 1 + 2n^{\frac{1}{4}}$

Tada je $M = (M_x, M_y)$ *netrivijalna točka na eliptičkoj krivulji* $y^2 = x^3 + Ax + B$. *Dodajmo kM samome sebi k puta koristeći standardno zbrajanje točaka eliptičkih krivulja. Tada, ako je qM jedinična, n je prost.*

Da bi dobili certifikat za n, brojeve iz teorema zakodiramo. Certifikat se može provjeriti u vremenu $O(\log^2 n)$, a algoritam sa slučajnim elementima koji generira certifikate je polinoman za sve osim za mali broj prostih brojeva.

Aldelman i Huang su 1993.g koristeći modificirani Goldwasserov i Kilianov teorem konsturirali algoritam sa slučajnim elementima koji u polinomnom vremenu odlučuje o prostosti zadanog broja za sve ulaze.
Za više informacija pogledati [5], str. 222-223.

Poglavlje 2

Agrawal-Kayal-Saxenov algoritam

U ovom poglavlju ćemo predstaviti algoritam polinomne vremenske složenosti za ispitivanje prostosti ulaznog broja koji su konstruirali Manindra Agrawal, Neeraj Kayal i Nitin Saxena. Pokazat ćemo da je korektan i da je polinomne složenosti.

2.1 AKS algoritam - dokaz korektnosti

U ovoj točki opisujemo korake algoritma i predstavljamo dokaz da je algoritam korektan. Korektnost će označavati činjenicu da algoritam vraća kao izlaz "broj je prost" ako i samo ako je ulazni broj prost. Taj rezultat dokazujemo kroz niz lema.

Motivacija

Najprije uvodimo, uz detaljno pojašnjenje, jednu notaciju koja će se koristiti u daljnjem. U tu svrhu podsjetimo se da za proizvoljan $n \in \mathbb{N}$ s $\mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z}$ označavamo prsten ostatka modulo n; to je kvocijentni prsten od $\mathbb{Z}$ po idealu $n\mathbb{Z}$. Elemente tog prstena pišemo kao $\overline{x} = x + n\mathbb{Z}$, za $x \in \mathbb{Z}$. Malo drugačije rečeno, imamo epimorfizam prstena s jedinicom

$$\bar{}\, : \mathbb{Z} \to \mathbb{Z}_n, \qquad \mathbb{Z} \ni x \mapsto \overline{x} \in \mathbb{Z}_n.$$

Sada to preslikavanje "podignemo" do preslikavanja među prstenima polinoma, u istoj oznaci,

$$\bar{}\, : \mathbb{Z}[X] \to \mathbb{Z}_n[X], \tag{2.1}$$

dano kao

$$f(X) = c_0 + c_1X + \cdots + c_kX^k \longmapsto \overline{f}(X) := \overline{c_0} + \overline{c_1}X + \cdots + \overline{c_k}X^k.$$

Lako je pokazati da vrijedi sljedeća lema.

Lema 2.1.1. *Preslikavanje* $\bar{}\,: \mathbb{Z}[X] \to \mathbb{Z}_n[X]$ *je također epimorfizam prstena s jedinicom.*

Sada smo u mogućnosti uvesti najavljenu notaciju.

Definicija 2.1.2. *Za dva polinoma* $f, g \in \mathbb{Z}[X]$ *reći ćemo da je* $f = g$ *unutar prstena polinoma* $\mathbb{Z}_n[X]$*, i pisati*

$$f \equiv g \pmod{n},$$

ukoliko je $\overline{f - g}(X) = \overline{f}(X) - \overline{g}(X)$ *nul-polinom u* $\mathbb{Z}_n[X]$.

Napomena 2.1.3. *Primijetimo kako gornja definicija u ekvivalentnoj formi glasi ovako:*

- *Vrijedi* $f \equiv g \pmod{n}$ *ako i samo ako je svaki koeficijent u polinomu* $f(X) - g(X) \in \mathbb{Z}[X]$ *djeljiv s n.*

Znači; za dva polinoma

$$f(X) = a_0 + a_1 X + \cdots + a_k X^k, \qquad a_i \in \mathbb{Z},$$
$$g(X) = b_0 + b_1 X + \cdots + b_k X^k, \qquad b_i \in \mathbb{Z},$$

iz $\mathbb{Z}[X]$ *je* $f \equiv g \pmod{n}$ *ako i samo ako*

$$n | a_i - b_i \quad za\ i = 0, 1, \ldots, k.$$

Ideju na kojoj se temelji AKS algoritam možemo uočiti na Pascalovom trokutu.

$$\begin{array}{ccccccccccc}
 & & & & & 1 & & & & & \\
 & & & & 1 & & 1 & & & & \\
 & & & \boxed{1} & & \boxed{2} & & \boxed{1} & & & \\
 & & \boxed{1} & & \boxed{3} & & \boxed{3} & & \boxed{1} & & \\
 & & 1 & & 4 & & 6 & & 4 & & 1 \\
 & \boxed{1} & & \boxed{5} & & \boxed{10} & & \boxed{10} & & \boxed{5} & & \boxed{1} \\
 & 1 & & 6 & & 15 & & 20 & & 15 & & 6 & & 1 \\
\boxed{1} & & \boxed{7} & & \boxed{21} & & \boxed{35} & & \boxed{35} & & \boxed{21} & & \boxed{7} & & \boxed{1} \\
1 & & 8 & & 28 & & 56 & & 70 & & 56 & & 28 & & 8 & & 1
\end{array}$$

Slika 2.1: Pascalov trokut

Uokvireni brojevi na prethodnoj slici pripadaju redovima s prostim rednim brojem počevši od nule, odnosno odgovaraju binomnom koeficijentu $\binom{p}{i}, 0 \leq i \leq p$. Primijetimo da su brojevi u sredini uokvirenih redova višekratnici tog prostog broja, dok za ostale redove to ne vrijedi. Iskazat ćemo to svojstvo sljedećom karakterizacijom prostih brojeva koja je dala temelj za algoritam koji su konstruirali Agrawal, Kayal i Saxena.

Lema 2.1.4. *Neka je $a \in \mathbb{Z}$, $n \in N$, $n \geq 2$ i $(a, n) = 1$. Tada je n prost ako i samo ako je*

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

Dokaz. Prvo primijetimo kako, po binomnom teoremu, imamo

$$(X + a)^n = X^n + \sum_{i=1}^{n-1} \binom{n}{i} a^{n-i} X^i + a^n,$$

i onda

$$f(X) := (X + a)^n - (X^n + a) = \sum_{i=1}^{n-1} \binom{n}{i} a^{n-i} X^i + (a^n - a).$$

Posebno, koeficijenti c_i, uz X^i u polinomu $f(X)$ jednaki su

$$c_i = \begin{cases} \binom{n}{i} a^{n-i} & \text{za } 1 \leq i \leq n-1, \\ a^n - a & \text{za } i = 0. \end{cases}$$

($\Rightarrow$) Sada; za dokaz implikacije leme s lijeva na desno, pretpostavimo da je n prost. Onda, jer je svaki binomni koeficijent $\binom{n}{i}$ očito djeljiv s n, imamo:

$$c_i \text{ je djeljiv s } n, \qquad \forall 1 \leq i \leq n-1.$$

Nadalje, po Malom Fermatovom teoremu:

$$c_0 \text{ je djeljiv s } n.$$

No dokazano, imajući na umu Napomenu 2.1.3 točno znači da je $f \equiv 0 \pmod{n}$, tj. da imamo $(X + a)^n \equiv X^n + a \pmod{n}$.

($\Leftarrow$) Za ovu implikaciju leme, pretpostavimo da je n složen broj. Neka je onda q neki prost djelitelj od n i $k \in \mathbb{N}$ *najveći mogući* takav da $q^k | n$. (Znači $q^{k+1} \nmid n$.) Pogledajmo koeficijent

$$c_q = \binom{n}{q} a^{n-q}.$$

Tvrdimo da taj c_q nije djeljiv s q^k. (To će onda dati da $f(X) \not\equiv 0 \pmod{n}$; i dokaz je gotov!) Jasno; jer je $(a, n) = 1$, onda je i $(a, q^k) = 1$. To znači da posebno q^k ne dijeli a^{n-q}. Dalje; ako raspišemo binomni koeficijent

$$\binom{n}{q} = \frac{n(n-1)\cdots(n-q+1)}{q!},$$

jasno je da nikoji faktor $n-j$, za $j=1,\ldots,q-1$, nije djeljiv s q. Zaključak: q^{k-1} dijeli $\binom{n}{q}$, ali q^k ne dijeli $\binom{n}{q}$!

Konačno imamo da svakako potencija q^k ne dijeli koeficijent c_q, a onda niti n ne dijeli taj c_q. Dakle, kako je već i rečeno, slijedi da $f(X) \not\equiv 0 \pmod{n}$.

□

Prije nego li nastavimo s dokazom glavnog rezultata, uvodimo još jednu važnu notaciju; gdje ćemo i sada sve potrebno detaljno objasniti. Neka je opet $n \in \mathbb{N}$, $n \geq 2$, i $\mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z}$ prsten ostataka modulo n. Prije smo gledali epimorfizam (2.1). Dalje; neka je sada $h = h(X) \in \mathbb{Z}_n[X]$ neki polinom. Gledajmo glavni ideal $(h(X))$, i kvocijentni prsten

$$\mathcal{A} := \mathbb{Z}_n[X]/(h(X)).$$

Neka je $\varepsilon : \mathbb{Z}_n[X] \to \mathcal{A}$ kanonski epimorfizam

$$\varphi \mapsto \varphi + (h(X)).$$

Komponiranjem dobivamo epimorfizam

$$\omega : \mathbb{Z}[X] \to \mathcal{A},$$
$$\omega(f(X)) := \varepsilon(\overline{f}(X)).$$

Definicija 2.1.5. *Za dva polinoma $f, g \in \mathbb{Z}[X]$ reći ćemo da je $f = g$ unutar prstena $\mathcal{A} = \mathbb{Z}_n[X]/(h(X))$, i pisati*

$$f \equiv g \pmod{h(X), n},$$

ukoliko je $\omega(f(X)) - \omega(g(X)) = \omega(f(X) - g(X)) = 0_{\mathcal{A}}$, nula u prstenu $\mathcal{A}$.

Analogno kao što smo u Napomeni 2.1.3 dali ekvivalentnu definiciju za kongruenciju $f \equiv g \pmod{n}$, tako ćemo sada dati pojašnjenje za prethodnu definiciju. Naime, evidentno je da kongruenciju iz gornje Definicije 2.1.5 u ekvivalentnoj formi možemo napisati kao u ovoj lemi.

Lema 2.1.6. *Imamo $f \equiv g \pmod{h(X), n}$ ako i samo ako vrijede sljedeća dva uvjeta:*

1. *Imamo $f \equiv g \pmod{n}$; tj svaki koeficijent u polinomu $f(X) - g(X) \in \mathbb{Z}[X]$ djeljiv je s n;*

2. *Polinom $\overline{f}(X) - \overline{g}(X)$ djeljiv je u $\mathbb{Z}_n[X]$ polinomom $h(X)$.*

Sljedeća je lema zapravo sasvim očekivan rezultat, kada se sjetimo da analogno vrijedi za "obične" kongruencije u $\mathbb{Z}$. Naime, ako su $a_1, a_2, b_1, b_2 \in \mathbb{Z}$ i $n \in \mathbb{N}$, te ako imamo $a_i \equiv b_i \pmod{n}$ za $i = 1, 2$, onda je

$$a_1 + a_2 \equiv b_1 + b_2 \pmod{n},$$
$$a_1 a_2 \equiv b_1 b_2 \pmod{n}.$$

Lema 2.1.7. *Neka su dani polinomi $f_1, f_2, g_1, g_2 \in \mathbb{Z}[X]$, te neka za neke $n \in \mathbb{N}$ i $h(X) \in \mathbb{Z}_n[X]$ imamo*

$$f_i \equiv g_i \pmod{h(X), n} \qquad za\ i = 1, 2.$$

Tada je

$$f_1 + f_2 \equiv g_1 + g_2 \pmod{h(X), n},$$
$$f_1 f_2 \equiv g_1 g_2 \pmod{h(X), n}.$$

Dokaz. Pokažimo samo tvrdnju za množenje; verzija za zbrajanje je još jednostavnija. U tu svrhu stavimo $f := f_1 f_2$ i $g := g_1 g_2$. Uzimajući u obzir Lemu 2.1.6 mi prvo moramo primijetiti da je svaki koeficijent u polinomu $f - g \in \mathbb{Z}[X]$ djeljiv s n. No to odmah slijedi iz jednakosti

$$f - g = (f_1 - g_1) f_2 + g_1 (f_2 - g_2) \tag{2.2}$$

i činjenice koja govori da je svaki koeficijent u polinomu $f_i - g_i \in \mathbb{Z}[X]$ djeljiv s n, za $i = 1, 2$. Jasno; za posljednju činjenicu trebamo samo pretpostavku ove leme i ponovo Lemu 2.1.6.

Da bismo dovršili dokaz potrebno je još vidjeti da je polinom $\overline{f} - \overline{g} \in \mathbb{Z}_n[X]$ djeljiv polinomom $h(X)$; tu i opet koristimo Lemu 2.1.6. Ali iz (2.2) i Leme 2.1.1 zaključujemo da je

$$\overline{f} - \overline{g} = (\overline{f_1} - \overline{g_1})\overline{f_2} + \overline{g_1}(\overline{f_2} - \overline{g_2}).$$

Kako po pretpostavci leme, uz uvjet 2. Leme 2.1.6 imamo da h dijeli $\overline{f_i} - \overline{g_i}$ za $i = 1, 2$, odmah slijedi i da h dijeli $\overline{f} - \overline{g}$. Još jednom po Lemi 2.1.6; dokaz je gotov. □

Karakterizacija iz leme 2.1.4 nam daje jednostavnu metodu za provjeru prostosti nekog broja. Uzmimo cijeli broj a takav da vrijedi $(a, n) = 1$ i izračunajmo:

$$f(X) := (X + a)^n - (X^n + a)$$

unutar prstena $\mathbb{Z}_n[X]$. Ako je $f(X) = 0$, tada je n prost, a inače je složen. Ovakav test je korektan, no njegova efikasnost tj. složenost nije zadovoljavajuća. Test zahtjeva računanje n koeficijenata, a pošto nam složenost algoritma ovisi o duljini ulaznog podatka u binarnom brojevnom sustavu njegova složenost je eksponencijalna. Jednostavan način da smanjimo broj koeficijenata unutar ovakvog testa bio bi da $f(X)$ računamo modulo n i modulo neki

polinom malog stupnja, recimo $X^r - 1$. Iako svaki prost broj n zadovoljava jednakost $(X + a)^n - (X^n + a) = 0$ unutar prstena $\mathbb{Z}_n[X]/(X^r - 1)$, za sve vrijednosti a i r, poneki složen broj n može zadovoljavati istu jednakost za neke vrijednosti a i r. No ako pažljivo odaberemo r i ako prethodni identitet vrijedi za nekoliko različitih vrijednosti a, možemo pokazati da je n potencija prostog broja. Agrawal, Kayal i Saxena su uspjeli vrijednost od r i broj različith a ograničiti sa $\log_2 n$, što im je dalo deterministički polinomni algoritam za određivanje prostosti. Naglasimo kako ćemo mi u daljnjem uvijek za logaritam u bazi 2 pisati samo log; tj.,

$$\log x := \log_2 x, \qquad x \in (0, +\infty).$$

AKS algoritam

Sada navodimo Agrawal, Kayal, Saxenov algoritam. Nazvati ćemo ga *AKS algoritam*.

Ulaz: prirodni broj $n > 1$
Izlaz: odluka je li n prost ili složen

1 **if** $n = a^b$ *za* $a \in \mathbb{N}$ *i* $b > 1$ **then** vrati SLOŽEN;
2 Pronađi najmanji r takav da je $o_r(n) > \log^2 n$
3 **if** $1 < (a, n) < n$ *za neki* $a \leq r$ **then** vrati SLOŽEN;
4 **if** $n \leq r$ **then** vrati PROST;
5 **for** $a = 1$ *do* $\lfloor \sqrt{\varphi(r)} \log n \rfloor$ **do**
  if $((X + a)^n \not\equiv X^n + a \pmod{X^r - 1, n})$ **then** vrati SLOŽEN ;
6 vrati PROST

Teorem 2.1.8. *AKS algoritam, čiji je ulazni podatak prirodni broj n, na izlazu daje poruku "PROST" ako i samo ako je broj n prost.*

Prethodni teorem ćemo dokazati kroz niz lema. Stoga započnimo.

Lema 2.1.9. *Ako je n prost broj, tada AKS algoritam na završetku rada daje poruku PROST.*

Dokaz. Pretpostavimo da je n prost broj. Tada zasigurno n nije oblika a^b gdje su $a, b > 1$, pa prvi korak algoritma neće vratiti SLOŽEN. Pošto je n prost, znamo da za svaki $x \in \mathbb{N}, x < n$ vrijedi $(x, n) = 1$. Dakle, korak 3 neće vratiti SLOŽEN. Već smo prije u lemi 2.1.4 vidjeli da ako je n prost, vrijedi $(X + a)^n \equiv X^n + a \pmod{n}$, pa ni korak 5 neće vratiti SLOŽEN. Dakle, algoritam će vratiti PROST. □

Pretpostavimo da algoritam vraća PROST. Ako algoritam vrati prost u koraku 4 tada znamo da za svaki $m < n$ vrijedi $(m, n) = 1$ (što je provjerno u koraku 3), dakle n je prost. Ostaje nam još slučaj kada algoritam vraća PROST u koraku 6. Ovaj slučaj je daleko

složeniji od prethodnog.
Započet ćemo s lemom koja će ograničiti veličinu broja r kojeg smo pronašli u koraku 2. Granica koju dobijemo biti će nam važna da kasnije odredimo složenost algoritma.

Lema 2.1.10. *Neka je n proizvoljan prirodan broj. Postoji $r \in \mathbb{N}$ sa sljedećim svojstvima:*

1. $r \leq \max\{3, \lceil \log^5 n \rceil\}$
2. $\mathrm{o}_r(n) > \log^2 n$
3. $(r, n) = 1$

Neka je $r \in \mathbb{N}$ neki proizvoljan, ali fiksirani prirodan broj, koji zadovoljava uvjete navedene u prethodnoj lemi. Pošto je $\mathrm{o}_r(n) > 1$, mora postojati prost dijelitelj p od n takav da vrijedi $\mathrm{o}_r(p) > 1$. Korak 3 nije vratio SLOŽEN pa znamo da vrijedi $(n, r) = (p, r) = 1$. Stoga vrijedi sa su $n, p \in \mathbb{Z}_r^*$. Znamo još da vrijedi $p > r$ jer bi inače koraci 3 i 4 vratili odluku o prostosti broja n. Stoga ćemo kroz ostatak diplomskog rada pretpostaviti sljedeće:

- Prirodni broj n ima prost dijelitelj p
- r je manji od $\lceil \log^5 n \rceil$ i $\mathrm{o}_r(n) > \log^2 n$
- $(r, n) = 1$
- Označimo $l := \lfloor \sqrt{\varphi(r)} \log n \rfloor$

Sada ćemo se fokusirati na korak 5 algoritma. Za vrijeme koraka 5, algoritam provjerava da li vrijedi $(X + a)^n \equiv (X^n + a) \pmod{X^r - 1, n}$ za sve $1 \leq a \leq l$. Zbog pretpostavke da algoritam ne vraća SLOŽEN vrijede sljedeće kongruencije:

$$(X + a)^n \equiv X^n + a \pmod{X^r - 1, n} \qquad \text{za } 0 \leq a \leq l.$$

Pošto $p|n$, onda je i $(X+a)^n = X^n+a$ unutar prstena $\mathbb{Z}_p[X]/(X^r-1)$; tj. vrijede i kongruencije

$$(X + a)^n \equiv X^n + a \pmod{X^r - 1, p} \qquad \text{za } 0 \leq a \leq l. \tag{2.3}$$

Također, zbog Leme 2.1.4, imamo i kongruencije

$$(X + a)^p \equiv X^p + a \pmod{X^r - 1, p} \qquad \text{za } 0 \leq a \leq l. \tag{2.4}$$

Iz kongruencija (2.3) i (2.4) odmah slijedi da vrijede i kongruencije

$$(X + a)^{n/p} \equiv X^{n/p} + a \pmod{X^r - 1, p} \qquad \text{za } 0 \leq a \leq l. \tag{2.5}$$

To posebno pokazuje da brojevi n i n/p oba zadovoljavaju iste kongruencije kao i sam prost broj p. Analizu ćemo započeti tako što ćemo prethodnom svojstvu dati ime.

Definicija 2.1.11. *Neka je $f \in \mathbb{Z}[X]$ i $m \in \mathbb{N}$. Kažemo da je m **introspektivan** za f ako vrijedi:*

$$f(X)^m \equiv f(X^m) \pmod{X^r - 1, p}.$$

Instruktivno je i korisno pogledati ovdje sljedeće primjere za introspektivnost.

Primjer 2.1.12.

- Iz kongruencija (2.4) i (2.5) jasno je da su brojevi n/p i p oba introspektivna za polinome $X + a$, za $0 \leq a \leq l$.
- Neka je $m = 2$, $r = 2$ i $p = 3$. Pokažimo da je m introspektivan za polinom $f(X) = X^2+3$. Neka je $f_1(X) = f(X^2) = X^4+3$ i neka je $f_2(X) = f(X)^2 = X^4+6X^2+9$. Neka je

  $$g(X) := f_1(X) - f_2(X) = X^4 + 6X^2 + 9 - (X^4 + 3) = 6X^2 + 6.$$

  Pošto su koefcijenti od $g(X)$ djeljivi s 3 vrijedi $f(X^2) \equiv f(X)^2 \pmod 3$. Pošto su $\overline{f_1}(X) = X^4$ i $\overline{f_2}(X) = X^4$ u $Z_3[X]$, slijedi da $X^2 - 1$ dijeli $\overline{f_1}(X) - \overline{f_2}(X) = 0$. Stoga po lemi 2.1.6 slijedi da je za $r = 2$ i $p = 3$ broj $m = 2$ introspektivan za polinom $X^2 + 3$.

Sada ćemo pokazati da su introspektivni brojevi zatvoreni s obzirom na množenje i da je skup polinoma za koje je dani broj introspektivan također zatvoren s obzirom na množenje.

Lema 2.1.13. *Neka su $f, g \in \mathbb{Z}[X]$, $s, t \in \mathbb{N}$. Vrijedi sljedeće:*

1. *Ako su s i t introspektivni za f, tada je i $s \cdot t$ introspektivan za f.*
2. *Ako je s introspektivan za f i g, tada je s introspektivan za fg.*

Dokaz.

1. Budući je s introspektivan za $f(X)$, vrijedi

   $$[f(X)]^{s \cdot t} \equiv [f(X^s)]^t \pmod{X^r - 1, p};$$

   ovdje koristimo i Lemu 2.1.7. Jer je t introspektivan za $f(X)$, zamjenom X sa X^s u definiciji za introspektivnost za t dobivamo,

   $$\begin{aligned} [f(X^s)]^t &\equiv [f(X^{s \cdot t})] \pmod{X^{s \cdot r} - 1, p} \\ &\equiv [f(X^{s \cdot t})] \pmod{X^r - 1, p} \end{aligned}$$

Prethodna jednakost vrijedi jer $X^r - 1$ dijeli $X^{s \cdot r} - 1$. Kombiniranjem prethodne dvije jednakosti dobivamo

$$[f(X)]^{s \cdot t} \equiv [f(X^{s \cdot t})] \pmod{X^r - 1, p};$$

ovdje ponovo koristimo Lemu 2.1.7.

2. Vrijedi

$$\begin{aligned}[f(X) \cdot g(X)]^s &\equiv [f(X)]^s \cdot [g(X)]^s \pmod{X^r - 1, p} \\ &\equiv f(X^s) \cdot g(X^s) \pmod{X^r - 1, p}\end{aligned}$$

□

Imajući u vidu prethodnu lemu slijedi da je svaki element skupa

$$I = \left\{ \binom{n}{p}^i \cdot p^j \mid i, j \geq 0 \right\}$$

introspektivan za svaki polinom iz skupa

$$P = \{ \prod_{a=0}^{l} (X + a)^{e_a} \mid e_a \geq 0 \}.$$

Sada ćemo, pomoću gornja dva skupa, definirati dvije grupe koje će imati glavne uloge u onome što slijedi.

Grupu I_r definiramo kao skup svih ostataka brojeva u I modulo r. Svi elementi iz I su relativno prosti s r pa je I_r podgrupa od $\mathbb{Z}_r^*$. Pošto znamo da grupu I_r generiraju n i p modulo r, i pošto vrijedi $\mathrm{o}_r(n) > \log^2 n$, slijedi da je $|I_r| > \log^2 n$. Red grupe I_r označavamo s $|I_r| = t$.

Da bi definirali drugu grupu potrebne su nam neke činjenice o ciklotomskim polinomima, nad konačnim poljima, koje smo naveli u točki *Prsten polinoma* u prvom poglavlju. Neka je $\Phi_r(X)$ r-ti ciklotomski polinom nad poljem $\mathbb{F}_p = \mathbb{Z}_p$. Iz lema 1.2.8 i 1.2.9 znamo da $\Phi_r(X)$ dijeli polinom $X^r - 1$ i faktorizira ga na ireducibilne faktore stupnja $\mathrm{o}_r(p)$. Neka je $h(X)$ jedan takav ireducibilan faktor. Pošto vrijedi $\mathrm{o}_r(p) > 1$, stupanj od $h(X)$ je veći od jedan. Sada ćemo definirati drugu bitnu grupu. Ta se grupa, koju ćemo označavati u daljnjem s G, sastoji od polinoma iz P modulo $h(X)$ i p. Drugim riječima, uz prije uveden epimorfizam $x \mapsto \overline{x}$ iz $\mathbb{Z}$ na $\mathbb{Z}_p$, imamo da je

$$G = \{ \overline{f} + (h(X)) \mid f \in P \}.$$

Ova grupa je očito generirana s polinomima $X, X+1, \ldots, X+l$ i čini podgrupu multipliktivne grupe polja

$$\mathbb{F} = \mathbb{Z}_p[X]/(h(X));$$

primijetimo da je, po Teoremu 1.2.5, $\mathbb{F}$ konačno polje čije je $\mathbb{Z}_p$ potpolje. U sljedeće dvije leme ćemo dati gornju i donju granicu za $|G|$. Prvu lemu je dokazao Hendrik Lenstra Jr. Lema je korisna pošto ne koristi komplicirane teoreme iz analitičke teorije brojeva, a uz to smanjila je kompleksnost samog algoritma.

Lema 2.1.14. *Za red grupe G imamo ocjenu*

$$|G| \geq \binom{t+l}{t-1}.$$

Dokaz. Primijetimo da je X primitivni r-ti korijen jedinice u $\mathbb{F}$, pošto je $h(X)$ faktor ciklotomskog polinoma.

Sada ćemo pokazati da se svaka dva polinoma stupnja manjeg od t iz P preslikavaju u različite elemente iz G. U tu svrhu, neka su $f(X)$ i $g(X)$ takva dva polinoma iz P. Pretpostavimo da vrijedi $f(X) = g(X)$ u polju $\mathbb{F}$. Neka je $m \in I$. Također vrijedi $[f(X)]^m = [g(X)]^m$ u polju $\mathbb{F}$. Pošto je m introspektivan za oba polinoma f i g, te $h(X)$ dijeli $X^r - 1$, dobivamo

$$f(X^m) = g(X^m)$$

u $\mathbb{F}$. Slijedi da je X^m korijen polinoma $Q(Y) = f(Y) - g(Y)$ za svaki $m \in I_r$. Znamo da u $\mathbb{F}$ imamo $\varphi(r)$ različitih r-tih primitivnih korijena jedinice i ako znamo jedan, u ovom slučaju X, ostali su nam dani sa X^k, gdje je k takav da vrijedi $1 \leq k < r$ i $(k, r) = 1$. Pošto vrijedi $(m, r) = 1$ (I_r je podgrupa od $\mathbb{Z}_r^*$), svaki takav X^m je r-ti primitivni korijen jedinice. Stoga postoji $|I_r| = t$ različitih korijena od $Q(Y)$ u $\mathbb{F}$. No prisjetimo se da smo f i g izabrali tako da im je stupanj manji od t, stoga je i stupanj od $Q(Y)$ manji od t. Time smo dobili kontradikciju sa lemom 1.2.6, pa vrijedi $f(X) \neq g(X)$ u $\mathbb{F}$.

Primjetimo da je $i \neq j$ u $\mathbb{Z}_p$, za $1 \leq i \neq j \leq l$ jer je $l = \lfloor \sqrt{\varphi(r)} \log n \rfloor < \sqrt{r} \log n < r$ i $p > r$. Slijedi da su elementi $X, X+1, \ldots, X+l$ svi međusobno različiti u $\mathbb{F}$. Također, jer je stupanj od h veći od jedan, vrijedi $X + a \neq 0$ u $\mathbb{F}$, za svaki $0 \leq a \leq l$. Dakle, postoji najmanje $l + 1$ različitih polinoma stupnja jedan u G.

Iz prethodnih opažanja slijedi da imamo bar $\binom{t+l}{t-1}$ različitih polinoma stupnja manjeg od t u grupi G. □

Kada n nije potencija od p, veličinu grupe G možemo ograničiti i s gornje strane.

Lema 2.1.15. *Ako n nije potencija od p, tada vrijedi* $|G| \leq n^{\sqrt{t}}$.

Dokaz. Promotrimo sljedeći podskup od I:

$$\hat{I} = \left\{ \left(\frac{n}{p}\right)^i \cdot p^j \,\middle|\, 0 \leq i, j \leq \lfloor \sqrt{t} \rfloor \right\}.$$

Sad ćemo pokazati da ako n nije potencija od p, tada skup $\hat{I}$ ima $(\lfloor \sqrt{t} \rfloor + 1)^2 > t$ različitih elemenata. Neka su $0 \leq i, j \leq \lfloor \sqrt{t} \rfloor$, $0 \leq i', j' \leq \lfloor \sqrt{t} \rfloor$. Pretpostavimo da vrijedi $\left(\frac{n}{p}\right)^i \cdot p^j = \left(\frac{n}{p}\right)^{i'} \cdot p^{j'}$. Dokazat ćemo da je $i = i'$ i $j = j'$. Budući n nije potencija od p, $\frac{n}{p} = q$, za neki $q \in \mathbb{N}$ koji također nije potencija od p. Sada vrijedi sljedeće

$$q^i \cdot p^j = q^{i'} \cdot p^{j'}.$$

Bez smanjenja općenitosti možemo pretpostaviti $i < i'$ pa dobivamo sljedeće

$$p^j = q^{i'-i} \cdot p^{j'}.$$

No pošto je p prost broj, q mora biti potencija od p. Time je dobivena kontradikcija pa vrijedi $i = i'$ i $j = j'$, dakle $|\hat{I}| = (\lfloor \sqrt{t} \rfloor + 1)^2$.

Kako je $|I_r| = t$, slijedi da barem dva broja iz $\hat{I}$ moraju biti jednaka modulo r. Neka su m_1 i m_2 takvi brojevi te neka je $m_1 > m_2$. Tada je $m_1 = m_2 + k \cdot r$, za neki $k \in \mathbb{N}$. Vrijedi

$$\begin{aligned} X^{m_1} &\equiv X^{m_2 + k \cdot r} \pmod{X^r - 1} \\ &\equiv X^{m_2} \cdot (X^r)^k \pmod{X^r - 1} \\ &\equiv X^{m_2} \pmod{X^r - 1}. \end{aligned}$$

Neka je $f(X) \in P$. Tada, jer je svaki broj iz $\hat{I}$ introspektivan za svaki polinom iz P, vrijedi

$$\begin{aligned} [f(X)]^{m_1} &\equiv f(X^{m_1}) \pmod{X^r - 1, p} \\ &\equiv f(X^{m_2}) \pmod{X^r - 1, p} \\ &\equiv [f(X)]^{m_2} \pmod{X^r - 1, p}. \end{aligned}$$

Dakle, vrijedi

$$[f(X)]^{m_1} = [f(X)]^{m_2}$$

u polju $\mathbb{F}$. Stoga, $f(X) \in G$ je korijen polinoma $Q'(Y) = Y^{m_1} - Y^{m_2}$ u polju $\mathbb{F}$. (Primijetimo kako smo ovdje, da bismo bili sasvim precizni, trebali pisati da je $\overline{f}(X) + (h(X)) \in G$ korijen od $Q'(Y)$.) Kako je $f(X)$ proizvoljan element iz G, polinom $Q'(Y)$ ima barem $|G|$ različitih korijena u $\mathbb{F}$. Stupanj od $Q'(Y)$ je $m_1 \leq (\frac{n}{p} \cdot p)^{\lfloor \sqrt{t} \rfloor} \leq n^{\sqrt{t}}$. Slijedi da je $|G| \leq n^{\sqrt{t}}$. □

Sada smo spremni dovršiti dokaz korektnosti algoritma.

Lema 2.1.16. *Ako AKS algoritam, koji na ulazu ima prirodan broj n, završi s porukom PROST, tada je n prost broj.*

Dokaz. Pretpostavimo da algoritam vraća PROST. Znamo da za $t = |I_r|$ i $l = \lfloor \sqrt{\varphi(r)} \log n \rfloor$ vrijedi

$$|G| \geq \binom{t+l}{t-1}.$$

Prije smo pokazali da je $t > \log^2 n$, tj. $\sqrt{t} > \log n$ pa vrijedi $t > \sqrt{t} \log n$. Slijedi da je $t \geq \lfloor \sqrt{t} \log n \rfloor + 1$, odnosno $t + l \geq l + 1 + \lfloor \sqrt{t} \log n \rfloor$. Zbog prethodne nejednakosti i simetrije binomnih koeficijenata vrijedi

$$\binom{t+l}{t-1} = \binom{t+l}{l+1} \geq \binom{l+1+\lfloor \sqrt{t} \log n \rfloor}{l+1} = \binom{l+1+\lfloor \sqrt{t} \log n \rfloor}{\lfloor \sqrt{t} \log n \rfloor}$$

Pošto je t red podgrupe $I_r \leq \mathbb{Z}_r^*$, a za red posljednje grupe znamo da je $|\mathbb{Z}_r^*| = \varphi(r)$, zaključujemo da vrijedi $t \leq \varphi(r)$. Sada slijedi $l = \lfloor \sqrt{\varphi(r)} \log n \rfloor \geq \lfloor \sqrt{t} \log n \rfloor$. Iz toga slijedi

$$\binom{l+1+\lfloor \sqrt{t} \log n \rfloor}{\lfloor \sqrt{t} \log n \rfloor} \geq \binom{2\lfloor \sqrt{t} \log n \rfloor + 1}{\lfloor \sqrt{t} \log n \rfloor}$$

Zbog $\lfloor \sqrt{t} \log n \rfloor > \lfloor \log^2 n \rfloor \geq 1$ vrijedi

$$\binom{2\lfloor \sqrt{t} \log n \rfloor + 1}{\lfloor \sqrt{t} \log n \rfloor} > 2^{\lfloor \sqrt{t} \log n \rfloor + 1}$$

Dokažimo prethodnu nejednakost indukcijom po $x = \lfloor \sqrt{t} \log n \rfloor$. Za $x = 2$ vrijedi

$$\binom{5}{2} = 10 > 2^3 = 8.$$

Pretpostavimo da nejednakost vrijedi za $x = k$ za neki $k \in \mathbb{N}$, tj. da vrijedi

$$\binom{2k+1}{k} > 2^{k+1}.$$

Tada za $x = k + 1$ vrijedi

$$\begin{aligned}\binom{2(k+1)+1}{k+1} = \binom{2k+3}{k+1} &= \frac{2k+3}{k+1}\binom{2k+2}{k} \\ &= \frac{2k+3}{k+1}\binom{2k+2}{k+2} \\ &= \frac{2k+3}{k+1}\cdot\frac{2k+2}{k+2}\binom{2k+1}{k+1} \\ &= \frac{2k+3}{k+1}\cdot\frac{2k+2}{k+2}\binom{2k+1}{k} \\ &= 2\cdot\frac{2k+3}{k+2}\binom{2k+1}{k}\end{aligned}$$

Zbog $\frac{2k+3}{k+2} > 1$ i pretpostavke indukcije vrijedi

$$\binom{2(k+1)+1}{k+1} > 2\cdot 2^{k+1} = 2^{k+2}.$$

Time je nejednakost dokazana. Sada zbog $2^{\lfloor \sqrt{t}\log n\rfloor+1} \geq 2^{\sqrt{t}\log n} = n^{\sqrt{t}}$ vrijedi

$$|G| > n^{\sqrt{t}}.$$

Prema lemi 2.1.15, ako n nije potencija od p, tada je $|G| \leq n^{\sqrt{t}}$. Stoga je $n = p^k$ za neki $k > 0$. Ako je $k > 1$, tada će algoritam u koraku 1 vratiti SLOŽEN pa mora vrijediti $k = 1$. Tada je $n = p$, tj. n je prost broj.

□

2.2 Složenost AKS algoritima

U ovoj točki dokazat ćemo da je AKS algoritam polinomne vremenske složenosti. U tu svrhu navest ćemo nekoliko pomoćnih algoritama, te ćemo dokazati njihovu korektnost i raspraviti vremensku složenost. Analizu vremenske složenosti AKS algoritma provest ćemo tako da detaljno analiziramo svaki korak algoritma. Ovdje ponovno navodimo AKS algoritam s numeriranim koracima.

Ulaz: prirodni broj $n > 1$
Izlaz: odluka je li n prost ili složen

1 **if** $n = a^b$ *za* $a \in \mathbb{N}$ *i* $b > 1$ **then** vrati SLOŽEN;
2 Pronađi najmanji r takav da je $o_r(n) > \log^2 n$
3 **if** $1 < (a, n) < n$ *za neki* $a \leq r$ **then** vrati SLOŽEN;
4 **if** $n \leq r$ **then** vrati PROST;
5 **for** $a = 1$ *do* $\lfloor \sqrt{\varphi(r)} \log n \rfloor$ **do**
  if $((X + a)^n \not\equiv X^n + a \pmod{X^r - 1, n})$ **then** vrati SLOŽEN ;
6 vrati PROST

Prisjetimo se da je obično zbrajanje prirodnih brojeva, kao i modularno zbrajanje, množenje i dijeljenje polinomne vremenske složenosti (vidi lemu 1.4.14).

Pošto se **prvi** korak AKS algoritma sastoji od određivanja da li je zadani prirodan broj $n \geq 2$ savršena potencija, tj. određivanja da li postoje prirodni brojevi $a \geq 2$ i $b \geq 2$ takvi da vrijedi $n = a^b$, prvo ćemo razmatrati problem prepoznavanja savršene potencije. Za zadani broj n, očito broj b ne može biti veći od $\log n$. To znači da se problem svodi na pitanje da li za $2 \leq b \leq \log n$ postoji $a \geq 2$ takav da $n = a^b$. Lako je vidjeti da vrijedi: prirodan broj n je savršena potencija s eksponentom b ako i samo ako vrijedi $\lfloor n^{\frac{1}{b}} \rfloor^b = n$. Iz tog razloga promatramo algoritam koji za fiksirani $b \geq 2$ računa $\lfloor n^{\frac{1}{b}} \rfloor$. Taj algoritam ćemo označavati sa *root*. Algoritam *root*, kao i svi detalji potrebni za dokazivanje njegove složenosti mogu se pronaći u [15] na stranama 25–29. Sada navodimo pseudokod algoritma *root*, te nakon toga nizom lema dokazujemo da je taj algoritam polinomne vremenske složenosti.

Algoritam *root*

Ulaz: $n, b \in \mathbb{N}$
Izlaz: $\lfloor n^{\frac{1}{b}} \rfloor$

x_0 := bilo koji prirodni broj takav da vrijedi $x_0^k \geq n$
$i := 0$
while $x_i^k > n$ **do**
  $x_{i+1} := \lfloor ((b-1)x_i + n/x_i^{b-1})/b \rfloor$
  $i := i + 1$
vrati x_i

Lema 2.2.1. *Algoritam root je korektan, tj. za sve prirodne brojeve n, b koji su ulazni podaci algoritma, na izlazu algoritam root daje broj $\lfloor n^{\frac{1}{b}} \rfloor$.*

Dokaz. Pokažimo prvo da za sve $i \geq 0$ za koje postoje brojevi x_i (čija definicija je navedena u algoritmu) vrijedi $x_i \geq \lfloor n^{\frac{1}{b}} \rfloor$. Pošto je x_0 prirodni broj i vrijedi $x_0^b \geq n$ jasno je da je

$x_0 \geq \lfloor n^{\frac{1}{b}} \rfloor$. Promotrimo sljedeću funkciju

$$f(x) = (b-1)x^b - bn^{\frac{1}{b}}x^{b-1} + n.$$

Primijetimo da vrijedi

$$f'(x) = b(b-1)x^{b-2}(x - n^{\frac{1}{b}}) > 0,$$

za sve $x > n^{\frac{1}{b}}$.

Stoga za sve x za koje vrijedi $x^b > n$ vrijedi $f(x) > f(n^{\frac{1}{b}}) = 0$. Prethodna nejednakost je ekvivalentna sa sljedećom:

$$\frac{(b-1)x + \frac{n}{x^{b-1}}}{b} > n^{\frac{1}{b}}.$$

Dakle, ako je $x_i^b > n$, tada x_{i+1} postoji i vrijedi

$$x_{i+1} = \left\lfloor \frac{(b-1)x_i + \frac{n}{x_i^{b-1}}}{b} \right\rfloor \geq \lfloor n^{\frac{1}{b}} \rfloor \qquad (*)$$

Time je prva pomoćna tvrdnja dokazana.

Iz nejednakosti $(*)$ slijedi:

$$x_{i+1} \leq \frac{(b-1)x_i + \frac{n}{x_i^{b-1}}}{b}.$$

Pošto je $x_i^b > n$, tada vrijedi $x_{i+1} < x_i$. Time smo dokazali da je niz (x_i) padajući.

Sada, iz činjenice da je niz (x_i) padajući, slijedi da *while petlja* algoritma *root* staje nakon konačnog broja koraka za svaki ulaz. Izlazni rezultat algoritma je prvi broj x_i za koji vrijedi $x_i^b \leq n$. Pošto je $x_i \leq n^{\frac{1}{b}}$, vrijedi $x_i \leq \lfloor n^{\frac{1}{b}} \rfloor$. Ranije smo pokazali da je $x_i \geq \lfloor n^{\frac{1}{b}} \rfloor$, stoga algoritam $root(n, b)$ na izlazu daje broj $x_i = \lfloor n^{\frac{1}{b}} \rfloor$. □

Primijetimo da algoritam *root* računa $\lfloor n^{\frac{1}{b}} \rfloor$ za bilo koji početni x_0 takav da $x_0^b \geq n$. Da bi dobili dobru ogradu na broj koraka koje napravi *while petlja* algoritma *root*, trebamo pažljivo izabrati x_0 (vidi [15], Lemma 4.25).

Lema 2.2.2. *Neka je m prirodni broj takav da vrijedi $2^{(m-1)b} < n \leq 2^{mb}$ i neka je $x_0 = 2^m$. Tada vrijedi da je $x_0^b \geq n$ i $\lfloor n^{\frac{1}{b}} \rfloor \leq x_0 < 2n^{\frac{1}{b}}$.*

Dokaz. Pošto vrijedi $x_0^b = 2^{mb} \geq n$, tada

$$\lfloor n^{\frac{1}{b}} \rfloor \leq n^{\frac{1}{b}} \leq 2^m = x_0 = 2 \cdot 2^{m-1} < 2n^{\frac{1}{b}}.$$

□

Lema 2.2.3. *Neka je $i \geq 0$ prirodan broj takav da $n^{\frac{1}{b}} < x_i < 2n^{\frac{1}{b}}$. Tada postoji prirodan broj x_{i+1} (koji je definiran u algoritmu root) za kojeg vrijedi sljedeće:*

$$x_{i+1} - n^{\frac{1}{b}} \leq \frac{b-1}{b+1}(x_i - n^{\frac{1}{b}}).$$

Dokaz. Znamo da vrijedi sljedeće:

$$x_{i+1} - n^{\frac{1}{b}} \leq \frac{(b-1)x_i + \frac{n}{x_i^{b-1}}}{b} - n^{\frac{1}{b}}.$$

Desna strana prethodne nejednakosti je manja ili jednaka $\frac{b-1}{b+1}(x_i - n^{\frac{1}{b}})$ ako i samo ako

$$(b-1)x_i^b - 2bn^{\frac{1}{b}}x_i^{b-1} + (b+1)n \leq 0.$$

Promotrimo sljedeću funkciju:

$$g(x) = (b-1)x_i^b - 2bn^{\frac{1}{b}}x_i^{b-1} + (b+1)n.$$

Primijetimo da vrijedi:

$$g'(x) = b(b-1)x^{b-2}(x - 2n^{\frac{1}{b}}) < 0,$$

za sve x takve da $n^{\frac{1}{b}} < x < 2n^{\frac{1}{b}}$. Stoga, za sve x takve da $n^{\frac{1}{b}} < x < 2n^{\frac{1}{b}}$, vrijedi $g(x) < g(n^{\frac{1}{b}}) = 0$. □

Pokažimo da *while petlja* algoritma *root* radi $O(\log n)$ koraka (vidi [15], Lemma 4.27).

Lema 2.2.4. *Neka je b prirodan broj takav da vrijedi $2 \leq b \leq \log n$ i neka je x_0 kao u lemi 2.2.2. Tada while petlja algoritma root s ulazom n i b radi $O(\log n)$ koraka.*

Dokaz. Označimo sa l broj koraka koje načini *while petlja.* Možemo pretpostaviti da je $l \geq 2$. Iz algoritma *root*, činjenice da je niz (x_i) padajući, te leme 2.2.2, slijedi:

$$x_l \leq n^{\frac{1}{b}} < x_{l-1} < x_{l-2} < \cdots < x_0 < 2n^{\frac{1}{b}}.$$

Primijenimo li prethodno razmatranje, tj. činjenicu da za x_i takav da je $n^{\frac{1}{b}} < x_i < 2n^{\frac{1}{b}}$ postoji x_{i+1} za kojeg vrijedi:

$$x_{i+1} - n^{\frac{1}{b}} \leq \frac{b-1}{b+1}(x_i - n^{\frac{1}{b}}),$$

dobivamo sljedeće:

$$x_{l-2} - n^{\frac{1}{b}} \leq \left(\frac{b-1}{b+1}\right)^{l-2}(x_0 - n^{\frac{1}{b}}) \leq \left(\frac{b-1}{b+1}\right)^{l-2} n^{\frac{1}{b}}.$$

Pošto je $n^{\frac{1}{b}} < x_{l-1} < x_{l-2}$ vrijedi $x_{l-2} - n^{\frac{1}{b}} > 1$. Dakle, vrijedi:

$$1 < \left(\frac{b-1}{b+1}\right)^{l-2} n^{\frac{1}{b}}.$$

Logaritmiramo li prethodnu nejednakost dobivamo sljedeće:

$$0 < (l-2)\log\left(\frac{b-1}{b+1}\right) + \log n^{\frac{1}{b}}.$$

Iz toga slijedi da vrijedi:

$$l - 2 < \frac{\log n^{\frac{1}{b}}}{\log\left(\frac{b+1}{b-1}\right)}.$$

Iz svega ovoga slijedi da je broj koraka *while petlje* ograničen sa:

$$O\left(\frac{\log n}{b\log\left(\frac{b+1}{b-1}\right)}\right).$$

Ako b ograničimo odozgo s konstantom, tada dobivamo granicu $O(\log n)$. Primijetimo da sljedeće:

$$\left(\frac{b+1}{b-1}\right)^b = \left(1 + \frac{2}{b-1}\right)^{b-1}\left(1 + \frac{2}{b-1}\right)$$

konvergira u e^2 kada $b \to \infty$. Dakle, ako je b veći od neke konstante b_0, tada vrijedi: $\left(\frac{b+1}{b-1}\right)^b \geq e$, tj. $b\log\left(\frac{b+1}{b-1}\right) \geq \log e$. Stoga za sve b takve da $b_0 \leq b \leq \log n$, broj koraka koje načini *while petlja* alogritma *root* je $O(\log n)$. □

Sada smo spremi dokazati da je algoritam *root* polinomne vremenske složenosti (vidi [15], Lemma 4.28).

Lema 2.2.5. *Neka su n i b prirodni projevi takvi da vrijedi* $2 \leq b \leq \log n$. *Algoritam root s ulaznim podacima n i b, te je* x_0 *kao u lemi 2.2.2, računa* $\lfloor n^{\frac{1}{b}} \rfloor$ *u vremenu* $O(\log^3 n \log\log n)$.

Dokaz. Iz činjenice da je niz (x_i) padajući i leme 2.2.2 slijedi da vrijedi $x_i^b \leq x_0^b < 2^b n \leq n^2$, za sve i za koje x_i postoji. Za dani x_i, potencije x_i^{b-1} i x_i^b možemo izračunati u vremenu $O(\log^2 n \log\log n)$. Potenciju x_{i+1} možemo izračunati na sljedeći način:

$$x_{i+1} = \left\lfloor \frac{(b-1)x_i^b + n}{bx_i^{b-1}} \right\rfloor,$$

za što nam treba $O(\log^2 n)$ koraka. Iz svega prethodnog i leme 2.2.4 slijedi da algoritam *root* računa $\lfloor n^{\frac{1}{b}} \rfloor$ u vremenu $O(\log^3 n \log\log n)$. □

Sljedeći teorem nam govori da za zadani prirodni broj n možemo u polinomnom vremenu odrediti je li savršena potencija (vidi [15], Lemma 4.29).

Teorem 2.2.6. *Neka je $n \geq 2$ prirodan broj. Tada postoji algoritam koji u vremenu $O(\log^4 n \log\log n)$ određuje da li postoje $a \geq 2$ i $b \geq 2$ takvi da vrijedi $n = a^b$.*

Dokaz. Ako je $n = a^b$ za prirodne brojeve $a \geq 2$ i $b \geq 2$, tada b mora biti manji ili jednak $\log n$. Iz leme 2.2.5 slijedi da za svaki b, takav da $2 \leq b \leq \log n$, broj $\lfloor n^{\frac{1}{b}} \rfloor$ možemo izračunati u vremenu $O(\log^3 n \log\log n)$. Ako je $\lfloor n^{\frac{1}{b}} \rfloor^b = n$ za bar jedan b, tada je broj n oblika a^b. Dakle, da bi saznali da li je n savršena potencija treba nam ukupno $O(\log^4 n \log\log n)$ koraka. □

Sada dolazimo do vrlo važnog dijela za dokaz složenosti AKS algoritma. U **drugom** koraku AKS algoritma tražimo broj r takav da je $o_r(n) > \log^2 n$ (n je broj na ulazu). Jako je bitno da nađemo broj r dovoljno brzo, tj. u polinomnom vremenu u ovisnosti o n. U prethodnom poglavlju vidjeli smo da je broj r ograničen sa $\log^5 n$ (vidi lemu 2.1.10). Dakle, i taj drugi korak AKS algoritma možemo izvesti u polinomnom vremenu.

U **trećem** koraku AKS algoritma moramo izračunati najveći zajednički djelitelj dva broja. Najveći zajednički djeljitelj brojeva a i b možemo odrediti pomoću poznatog Euklidovog algoritma. Očito vrijedi:

$$(a, b) = (qb + r, b) = (r, b) = (b, r). \tag{2.6}$$

Druga jednakost slijedi iz toga što je qb djelivo s b. Zapažanje takve rekurzivne veze nam daje osnovu za Euklidov algoritam. Očito je $(r, 0) = r$ pa znamo i uvjet za zaustavljanje algoritama. Sada navodimo Euklidov algoritam u formi pseudokoda.

Euklidov algoritam

Ulaz: $a, b \in \mathbb{Z}$
Izlaz: najveća zajednička mjera od a i b

if $|b| > |a|$ **then** zamijeni a i b;
while $b > 0$ **do**
  $a := b$
  $b := a \pmod b$
vrati a

Teorem 2.2.7. *Vremenska složenost Euklidovog algoritma za traženje najvećeg zajedničkog djeljitelja (a, b), pri čemu je $a \geq b$, je $O(\log^3 a)$.*

Dokaz. Kako bi odredili vremensku složenost, najvažnije je odrediti koliko puta moramo ponoviti rekurzivni korak u (2.3) da bi došli do rješenja. Razmotrimo općeniti i–ti korak. Tada imamo:

$$(r_{i-1}, r_i) = (r_i, r_{i+1}) \quad \text{gdje je} \quad r_{i-1} = kr_i + r_{i+1},$$

odnosno r_{i+1} je ostatak dijeljenja r_{i-1} s r_i. To znači da mora vrijediti: $r_{i-1} \geq r_i + r_{i+1} > 2r_{i+1}$. Pomnožimo posljednju nejednakost s r_i, te je malo posložimo. Tada dobivamo sljedeće:

$$r_{i+1}r_i < \frac{r_i r_{i-1}}{2}.$$

Nastavljajući isti postuapk do r_0 i r_1 dobivamo:

$$r_{i+1}r_i < \frac{r_i r_{i-1}}{2} < \frac{r_1 r_0}{2^i}.$$

Ako je $r_{i+1}r_i < 1$ onda će Euklidov algoritam stati jer je sigurno jedan od tih brojeva 0. Iz ovog posljednjeg malim računom lako dobijemo za koje i vrijedi posljednja nejednakost:

$$r_{i+1}r_i < \frac{r_1 r_0}{2^i} < 1 \qquad \Rightarrow \qquad r_1 r_0 < 2^i \qquad \Rightarrow \qquad i > \log_2 r_1 r_0.$$

Dakle, nakon $O(\log r_1 r_0)$ tj. $O(\log ab) = O(\log a)$ koraka Euklidov algoritam će sigurno stati.

U svakom koraku algoritma moramo obaviti dijeljenje, koje je vremenske složenosti $O(\log a \log b) = O(\log^2 a)$. To znači da je vremenska složenost Euklidovog algoritma $O(\log^3 a)$. □

U **četvrtom** koraku AKS algoritma moramo usporediti dva prirodna broja. To možemo obaviti u vrememnu $O(\log n)$.

U **petom** koraku AKS algoritma najzahtjevnije je izračunati koeficijente polinoma $(X-a)^n \pmod{X^r - 1}$ iz $\mathbb{Z}_n[X]$. Sljedeći algoritam računa koeficijente od $(X-a)^n \pmod{X^r - 1}$ iz $\mathbb{Z}_n[X]$. Pseudokod ovog algoritma je dan u [15], str. 23.

Algoritam *poly*

Ulaz: $n, r, a \in \mathbb{Z}$ takvi da $2 \leq r < n$ i $1 \leq a < n$
Izlaz: Koeficijenti polinoma $(X-a)^n \pmod{X^r - 1}$ iz $\mathbb{Z}_n[X]$

$f(X) := 1$; $g(X) := X - a$; $y := n$
while $y \neq 0$ **do**
  if *y je paran* **then**
    $y := y/2$
    $h(X) := g(X)g(X)$
    $g(X) := h(X) \pmod{X^r - 1}$
  else
    $y := y - 1$
    $h(X) := f(X)g(X)$
    $f(X) := h(X) \pmod{X^r - 1}$
vrati $f(X)$

Lema 2.2.8. *Neka su n, r i a prirodni brojevi takvi da* $2 \leq r < n$ *i* $1 \leq a < n$. *Algoritam poly računa koeficijente polinoma* $(X-a)^n \pmod{X^r-1}$ *iz* $\mathbb{Z}_n[X]$ *u vremenu* $O(r^2 \log^3 n)$.

Dokaz. Pošto algoritam radi s koeficijentima iz $\mathbb{Z}_n[X]$ pretpostavimo da se u svim operacijama u kojima korisimo polinome $g(X)$, $h(X)$ i $f(X)$ koeficijenti reduciraju modulo n. Primijetimo da će polinom koji algoritam vraća biti oblika:

$$(X-a)^n \pmod{X^r-1} \in \mathbb{Z}_n[X].$$

Zatim, primijetimo da su u bilo kojem koraku algoritma stupnjevi polinoma $f(X)$ i $g(X)$ manji ili jednaki $r-1$. Stoga je stupanj od $h(X)$ manji li jednak $2r-2$. Vidimo da nam za računanje $h(X) = g(X)g(X)$ treba $O(r^2)$ množenja modulo n. Svaki put množimo prirodne brojeve iz skupa $\{1, \ldots, n-1\}$, pa $h(X)$ možemo izračunati u vremenu $O(r^2 \log^2 n)$. Jednaka ograda vrijedi i za računanje $h(X) = f(X)g(X)$.

Polinom $h(X)$ možemo zapisati kao $h(X) = \sum_{i=0}^{2r-2} h_i X^i$, gdje su koeficijenti h_i iz skupa $\{1, \ldots, n-1\}$. Tada vrijedi sljedeće:

$$h(X) \pmod{X^r-1} = \sum_{i=0}^{r-2} [h_i + h_{r+i} \pmod{n}] X^i + h_{r-1} X^{r-1}.$$

Dakle, reduciranje $h(X)$ modulo X^r-1 možemo napraviti u vremenu $O(r \log n)$. Pošto *while petlja* radi $\log n$ koraka, algoritam izračunava $(X-a)^n \pmod{X^r-1}$ u vremenu $O(r^2 \log^3 n)$. □

Sada smo spremni konačno dokazati teorem o ukupnoj vremenskoj složenost AKS algoritma.

Teorem 2.2.9. *Vremenska složenost AKS algoritma je* $O(\log^{16.5} n)$.

Dokaz. Razmotrimo složenost AKS algoritma po koracima. Neka je na ulazu AKS algoritma broj n.

1. Iz teorema 2.2.6 slijedi da u vremenu $O(\log^4 n \log\log n)$ možemo obaviti prvi korak AKS algoritma.

2. U drugom koraku tražimo r takav da je $o_r(n) > \log^2 n$. To možemo učiniti tako da promatramo uzastopne vrijednosti od 1 do r i provjeravamo je li $n^k \not\equiv 1 \pmod{r}$ za sve $k \leq \log^2 n$. Iz leme 2.1.10 znamo da ćemo traženi r naći u $O(\log^5 n)$ koraka. U svakom koraku moramo napraviti $\log^2 n$ potenciranja, što je složenosti $O(\log^2 n \log^2 n \log\log^2 n) = O(\log^4 n \log\log^2 n)$. Dakle, ukupna složenost drugog koraka AKS algoritma je $O(\log^9 n \log\log^2 n)$.

3. Treći korak zahtijeva izračunavanje najvećeg zajedničkog djelitelja (a, n) za sve $a \leq r$. Iz leme 2.2.7 znamo da je složenost Euklidovog algoritma jednaka $O(\log^3 n)$. Pošto znamo da je r ograničen sa $\log^5 n$ ukupna složenost trećeg koraka AKS algoritma jednaka je $O(\log^8 n)$.

4. U četvrtom koraku AKS algoritma potrebno je učiniti samo jedno uspoređivanje dva prirodna broja. To znači da je taj korak AKS algoritma vremenske složenosti $O(\log n)$.

5. U petom koraku AKS algoritma trebamo provjeriti $\lfloor \sqrt{\varphi(r)} \log n \rfloor$ jednadžbi. U lemi 2.2.8 smo vidjeli da $(X - a)^n \pmod{X^r - 1}$ iz $\mathbb{Z}_n[X]$ možemo izračunati u vremenu $O(r^2 \log^3 n)$. Stoga je ukupna složenost petog koraka jednaka:

$$O(r^2 \log^3 n \sqrt{\varphi(r)} \log n) = O(r^{\frac{5}{2}} \log^4 n) = O(\log^{\frac{33}{2}} n). \tag{2.7}$$

Kao što vidimo, peti korak AKS algoritma dominira nad ostalima po složenosti, stoga ukupna složenost AKS algoritma iznosi $O(\log^{16.5} n)$. □

Algoritam AKS u petom koraku mora napraviti $\lfloor \sqrt{\varphi(r)} \log n \rfloor$ koraka da bi veličina grupe G iz točke *2.1 AKS algoritam - dokaz koraktnosti* bila dovoljno velika. Broj koraka bi se mogao smanjiti ako bi mogli pokazati da se uz pomoć manjeg skupa polinoma $(X + a)$ može dobiti grupa odgovarajuće veličine.

Promotrimo sljedeću slutnju koju su dali R. Bhattacharjee i P. Pandey. Kayal i Saxena su je provjerili za $r \leq 100$ i $n \leq 10^{10}$.

Slutnja 2.2.10. *Ako je r prost broj koji ne dijeli r i ako vrijedi*

$$(X - 1)^n \equiv X^n - 1 \pmod{X^r - 1, n},$$

tada vrijedi da je n prost ili $n^2 \equiv 1 \pmod{r}$.

Kad bi prethodna slutnja bila točna, tada bi se AKS algoritam mogao modificirati tako da tražimo prvi r koji ne dijeli $n^2 - 1$. Za takav r bi vrijedlio $2 \leq r \leq 4 \log n$. Stoga bi vrijedilo da je složenost AKS algoritma $O(r^{\frac{5}{2}} \log^4 n) = O(\log^{6.5} n)$.

Iako je AKS deterministčki algoritam polinomne vremenske složenosti, u praksi se ipak koriste više algoritmi sa slučajnim elementima.

Bibliografija

[1] L. M. Adleman, C. Pomerance and R. S. Rumley, *On distinguishing prime numbers from composite numbers*, Annals of mathematics **117** (1983), 173-206.

[2] M. Agrawal, N. Kayal i N. Saxena, *PRIMES is in P*, Annals of Math. (2) **160** (2004), 781–793.

[3] W. R. Alford, A. Granville and C. Pomerance; *There are infinitely many Carmichael numbers*, Annals of Math. (2) **139**, no. 3 (1994), 703-722.

[4] A. Dujella, *Uvod u teoriju brojeva (skripta)*, PMF - Matematički odjel, Sveučilište u Zagrebu, 2006 "`http://web.math.pmf.unizg.hr/~duje/utb/utblink.pdf`"

[5] A. Dujella, M. Maretić, *Kriptografija*, Element, Zagreb, 2007.

[6] A. Dujella, *Predavanja iz kolegija Kriptografija*, PMF - Matematički odjel, Sveučilište u Zagrebu, "`http://web.math.pmf.unizg.hr/~duje/kript/solstras.html`"

[7] G. Everest, T. Ward, *An Introduction to Number Theory*, Springer, 2005.

[8] N. Koblitz; *A Course in Number Theory and Cryptography*, Grad. Texts in Math 114, Springer, 1994.

[9] D. H. Lehmer, *Tests for primality by the converse of fermat's theorem*, Bull. Amer. Math. Soc. , **33**, (1927) 327-340.

[10] R. Lidl, H. Niederreiter, *Introduction to finite fields and their applications*, 2^{nd} Revised Edition, Cambridge University Press, 1994.

[11] S. Manhart, *Algoritmi sa slučajnim elementima*. Diplomski rad, PMF-Matematički odjel, Sveučilište u Zagrebu, 2008.

[12] G. L. Miller, *Riemann's hypothesis and tests for primality*, Journal of Computer and System Sciences **13** (1976), 300-317.

[13] C. H. Papadimitrou, *Computational Complexity*, Addison - Wesley Publishing Company, Boston, 1994.

[14] M. Sipser, *Introduction to the Theory of Computation, Second Edition*, Thomson Course Technology, Boston, 2006.

[15] M. Smid, *Primality testing in polynomial time*, preprint, School of Computer Science, Carleton University, Ottawa, 2003. "http://people.scs.carleton.ca/~michiel/primes.pdf"

[16] B. Širola, *Algebarske strukture (skripta)*, PMF - Matematički odsjek, Zagreb, 2010 "http://web.math.pmf.unizg.hr/nastava/alg/predavanja/ASpred.pdf"

[17] M. Vuković, *Složenost algoritama (skripta)*, PMF - Matematički odsjek, Zagreb, 2013. "http://web.math.pmf.unizg.hr/~vukovic/Diplomski-kolegiji/SA/SA-skripta-2013-verzija-4.pdf"

Sažetak

U računalnoj znanosti se dugo vremena čekalo na deterministički algoritam koji u polinomnom vremenu može odrediti je li neki prirodan broj prost. Agrawal-Kayal-Saxenin algoritam je prvi algoritam za koji se dokazalo da određuje prostost nekog broja u polinomnom vremenu, tj. navedena trojica matematičara su pokazali da jezik *PRIMES* pripada klasi složenosti *P*. Cilj ovog diplomskog rada bio je pokazati da jezik *PRIMES* pripada klasi *P*, tj. predstaviti Agrawal-Kayal-Saxenov algoritam, dokazati njegovu korektnost i odrediti njegovu složenost. U prvom, uvodnom poglavlju diplomskog rada dali smo bitne definicije i rezultate iz teorije brojeva, teorije složenosti i algebre. Te definicije i rezultati su nam bili potrebni da bi u drugom, glavnom dijelu diplomskog rada mogli dokazati korektnost Agrawal-Kayal-Saxeninog algoritam i pokazati da pripada klasi složenosti *P*. Također smo u prvom poglavlju opisali algoritme sa slučajnim elementima te smo prošli kroz povijest problema *PRIMES*.
Glavni dio diplomskog rada smo podijelili u dva dijela. U prvom dijelu, *AKS algoritam - dokaz korektnosti*, dali smo motivaciju za algoritam koja se temeljila na Pascalovu trokutu, te smo predstavili Agrawal-Kayal-Saxenin algoritam i kroz niz rezultata pokazali smo da je korektan. U drugom dijelu smo uz pomoć raznih algoritama, koji su nam olakšali dokazivanje, pokazali da je Agrawal-Kayal-Saxenin algoritam polinomne složenosti.

Summary

Deterministic algorithm for primality test was long time expected thing in computer science. Agrawal-Kayal- Saxena algorithm is the first algorithm which in polynomial time determines whether a given number is prime, namely the threesome has shown that language *PRIMES* is in complexity class *P*. The main goal of this work is to show that language *PRIMES* is in complexity class *P*, namely introduce Agrawal-Kayal-Saxena primality test, prove its correctness and determine its complexity. In the first, introductory chapter of this work we have gave essential definitions and results from number theory, complexity theory and algebra. We need this definitions and results for our second, main chapter, so that we can prove correctness of Agrawal-Kayal-Saxena primality test and show that it is in complexity class *P*. In the introductory chapter we have also described randomized algorithms and we have gave walk through the histroy of PRIMES problem.
The main part of this work we have divided into two parts. We have mentioned Pascal triangle as a motivation for the AKS algorithm. We have also introduced Agrawal-Kayal-Saxena primality test and with help of various results we have proved its correctness. In the second part of main chapter, with help of various algorithms we have proved that Agrawal-Kayal-Saxena primality test has polynomial complexity.

Životopis

Josip Grgurica rođen je 11. kolovoza 1989. godine u Šibeniku. U Šibeniku pohađa Osnovnu školu Petra Krešimira IV., a zatim i Opću gimnaziju Antuna Vrančića. Kroz osnovnu i srednju školu trenira nogomet u HNK Šibenik. Nakon završetka srednje škole, 2008. godine upisuje preddiplomski studij matematike na Matematičkom odsjeku Prirodoslovno-matematičkog fakulteta u Zagrebu. Kroz tri godine na preddiplomskom studiju razvija interes prema računarstvu i 2011. godine upisuje diplomski studij Računarstvo i matematika. Tokom diplomskog studija najzanimljiviji dio računarstva mu je složenost algoritama pa za diplomski rad bira temu *Problem PRIMES pripada klasi PTIME*. U ljeto 2013. godine seli se u Split te se zapošljava u programerskoj tvrtci Extensionengine sa sjedištem u Bostonu u kojoj trenutno radi na projektu HBX za Harvard Business School.