ODLOČITVENI PROBLEM PRAŠTEVILO

TADEJ NOVAK

Math. Subj. Class. (2000): 11-04, 11A41, 11A51

povzetek

Odločitveni problem PRAŠTEVILO je problem s področja teoretičnega računalništva, ki ga od lanskega leta dalje uvrščamo v množico v polinomskem času deterministično rešljivih problemov (P-problemi). V članku je opisan algoritem ter dokaz njegove pravilnosti in časovne zahtevnosti. Narejen je tudi pregled algoritmov, ki se uporabljajo v praksi.

DECISION PROBLEM PRIMES

abstract

Decision problem PRIMES is problem from theoretical computer field of science. From August 2002 it is classified in the set of problems, which can be solved in deterministic way in polynomial time (P-problems). Algorithm and the proof of its correctness and time complexity is presented in the article. We also summarize algorithms which are used in practice.

1 Uvod

citat

Kako za dano število ugotoviti, ali je praštevilo (t.i. odločitveni problem PRAŠTEVILO), je bil problem, ki je okupiral matematike že v antičnih časih. Karl Friedrich Gauss (1777-1855), je v svoji knjigi Disquistiones Arithmeticae (1801) zapisal tudi: "Menim, da čast znanosti narekuje, da z vsemi sredstvi iščemo rešitev tako elegantnega in tako razvpitega problema." Od leta 1960 dalje se je s prihodom računalnikov spremenil poudarek od iskanja matematične formule do iskanja učinkovitega algoritma.

zgodovina reševanja problema PRAŠTEVILO

Do leta 2002 sta bili za problem PRAŠTEVILO najbolj poznani dve vrsti algoritmov polinomske časovne zahtevnosti, to je deterministični, katerih pravilnosti nismo znali dokazati, ter verjetnostni algoritmi. Omenimo najpomembnejše prelomnice pri iskanju čimboljšega algoritma za problem PRAŠTEVILO. Najstarejši poznan algoritem je Eratostenovo rešeto iz leta 240 p.n.š., toda njegova časovna zahtevnost je $O(n)$, torej eksponentna glede na dolžino zapisa obravnavanega števila n. Večji korak naprej je napravil Fermat v 17. stoletju, ko je dokazal, da za vsako naravno število a velja $a^{p-1} \equiv 1 \pmod{p}$ za vsako praštevilo p, ki ne deli a (Fermatov mali izrek). Izrek je uporabljen pri dokazovanju pravilnosti mnogih algoritmov za problem PRAŠTEVILO. Leta 1976 je Miller predstavil deterministično algoritem polinomske časovne zahtevnosti, katerega pravilnost pa ni v celoti dokazana, saj temelji na predpostavki, da velja razširjena Riemannova hipoteza[1]. Nato sta Solovay in Strassen leta 1977 predstavila verjetnosten algoritem časovne zahtevnosti $O(\log^3 n)$. Rabin je leta 1980 modificiral Millerjev test in sestavil verjetnostni algoritem, katerega pravilnost delovanja je lahko dokazal, saj se mu ni bilo potrebno sklicati na Riemannovo hipotezo. Ta algoritem je še danes med najučinkovitejšimi. Največji preskok na poti do determinističnega algoritma polinomske časovne zahtevnosti pa so napravili Adleman, Pomerance in Rumely leta 1983 z algoritmom časovne zahtevnosti $O(\log n^{O(\log\log\log n)})$. Goldwasser in Kilian

[1] http://www.claymath.org/prizeproblems/riemann.htm

sta leta 1986 napisala verjetnosten algoritem z uporabo eliptičnih krivulj, katerega pričakovana časovna zahtevnost je polinomska za skoraj vse možne vhodne podatke, oz. za vse pod neko hipotezo. Leta 2002 pa so Agrawal, Kayal in Saxena napisali deterministično algoritem časovne zahtevnosti največ $O\left(\log^{12} n\right)$ - v praksi $O\left(\log^{6} n\right)$. Predstavili so tudi deterministično algoritem časovne zahtevnosti $O\left(\log^{3} n\right)$, vendar pa njegove pravilnosti še niso uspeli v celoti dokazati.

Opise Eratostenovega rešeta, Solovay-Strassenovega testa ter Miller-Rabinovega najdemo v učbeniku za kriptografijo Stinson [16] ali pa v Bressoud [7], idejo verjetnostnih algoritmov pa bomo spoznali v razdelku 4.

o oznaki P

Oznaka P predstavlja množico problemov, za katere obstaja deterministično algoritem $\mathcal{A}$ polinomske časovne zahtevnosti glede na dolžino zapisa vhodnih podatkov. Torej obstaja polinom $p(x) \in \mathbb{Z}[x]$, da je $T_{\mathcal{A}}(x) \leq p(n)$ za vsak vhodni podatek x dolžine n, kjer je $T_{\mathcal{A}}(x)$ oznaka za število korakov algoritma. Deterministične algoritme lahko bolj natančno opredelimo kot Turingove stroje, o katerih je za Obzornik pisal že Tomaž Pisanski [15], časovno zahtevnost pa kot število izpolnjenih ukazov stroja. Več o algoritmih in časovnih zahtevnostih si lahko preberemo tudi v Horowitz in Sahni [11].

vpeljava oznak $\mathbb{P}$, $P(n)$, F_{p^d}, $h(x)$, $\widetilde{O}(t(n))$, $\log$, $o_r(n)$

Vpeljimo še nekaj oznak, ki jih bomo pogosto uporabljali v nadaljevanju. S $\mathbb{P}$ označimo množico praštevil, s $P(n)$ največji praštevilski delitelj števila n, GFp^d pa naj bo končni obseg moči p^d. Naj bo $\widetilde{O}(t(n))$ oznaka za $O(t(n)\,\texttt{poli}(\log t(n)))$, kjer je t neka funkcija in `poli` nek polinom. Oznako srečamo na primer, ko s pomočjo rekurzivnega izračuna izboljšamo časovno zahtevnost za en red s tem da pridobimo logaritemski faktor $\log n$, ki je asimptotično zanemarljiv v primerjavi z n. To pa je tudi razlog, zakaj namesto $\widetilde{O}$ ponavadi pišemo kar O. Vseskozi bomo z log označevali logaritem z osnovo 2, z $o_r(n)$ pa red elementa n v grupi $\mathbb{Z}_r^*$.

vsebina članka

V osrednjem delu članka bomo predstavili deterministični algoritem `PrastQ` polinomske časovne zahtevnosti, ki reši problem `PRAŠTEVILO`. Na kratko rečemo, da je `PRAŠTEVILO` v P. Na koncu bomo opisali še algoritme za reševanje problema `PRAŠTEVILO`, ki se uporabljajo v praksi. Predstavljeni bodo tudi nekateri odprti problemi.

2 `PRAŠTEVILO` je v P

Trditev iz naslova razdelka bomo dokazali v treh podrazdelkih. V 2.1 bomo predstavili deterministični algoritem za problem `PRAŠTEVILO`, povzet po Agrawal in ostali [1], v 2.2 bomo dokazali, da je polinomske časovne zahtevnosti, v 2.3 pa še pravilnost delovanja.

2.1 Ključna ideja in algoritem

karakterizacija praštevil

Poglejmo si karakterizacijo praštevil, ki je ključnega pomena pri konstrukciji algoritma.

Trditev 1. *Naj bosta a in n tuji si naravni števili. Potem je n praštevilo natanko tedaj, ko je*

$$(x-a)^n \equiv (x^n - a) \pmod{n} \qquad \textit{(kot polinom)} \tag{1}$$

Dokaz. Definirajmo $w(x) = (x-a)^n - (x^n - a)$, kar je z upoštevanjem binomske formule enako $w(x) = \sum_{0<i<n}(-1)^i \binom{n}{i} a^{n-i}x^i + (-1)^n a^n + a$. Trditev dokažemo v vsako smer posebej.

($\Rightarrow$) Naj bo $n \in \mathbb{P}$. Potem je $\binom{n}{i} \equiv 0 \pmod{n}$ za vsak i, $0 < i < n$. Če upoštevamo še Fermatov izrek, dobimo $w(x) \equiv 0 \pmod{n}$ za vsak $n \in \mathbb{P}$.

($\Leftarrow$) Naj bo n sedaj sestavljeno število. Potem obstaja $q \in \mathbb{P}$, $1 < q < n$, in obstaja $k \in \mathbb{N}$, tako da $q^k | n$ in $q^{k+1} \nmid n$. Ker je $\binom{n}{q} = \frac{n(n-1)\cdots(n-q+1)}{q(q-1)\cdots 1}$, velja $q^{k-1} | \binom{n}{q}$ ter $q^k \nmid \binom{n}{q}$ in zato n ne deli $\binom{n}{q}$. Ker je n tuj a, je $\mathrm{D}(n, a^{n-q}) = 1$ in zato koeficient $(-1)^q \binom{n}{q} a^{n-q}$ v polinomu $w(x)$ pred x^q ni deljiv z n. ■

časovna prezahtevnost izračuna $(x-a)^n \bmod n$ — — — izboljšava in razlaga operacije $\bmod x^r - 1, n$

Če preverimo kongruenco (1) za nek $a < n$, pri katerem je $\mathrm{D}(a, n) = 1$, npr. $a = 1$, potem se lahko na podlagi trditve 1 odločimo, ali je n praštevilo. Za izračun $(x-a)^n$ uporabimo algoritem 'kvadriraj in množi', vendar pa je časovna zahtevnost takega izračuna eksponentna glede na dolžino zapisa števila n, saj je kvadriranje polinoma $(x-a)^{\lfloor n/2 \rfloor}$, ki je stopnje $\lfloor n/2 \rfloor$, časovne zahtevnosti vsaj $O(n)$. Algoritem izboljšamo tako, da izberemo "primeren" $r \in \mathbb{N}$ in za $a = 1$ do $2\sqrt{r}\log n$ preverjamo

$$(x-a)^n \equiv (x^n - a) \pmod{x^r - 1, n} \tag{2}$$

To so kongruence v faktorskem kolobarju $\mathbb{Z}[x]/(n, x^r - 1)$, to je faktorskem kolobarju kolobarja $\mathbb{Z}[x]$ po idealu $(n, x^r - 1)$. Če je n praštevilo, je kongruenca (2) posledica trditve 1. Težava pa je v tem, da kongruenca (2) lahko velja tudi pri nekaterih sestavljenih številih.

r je primeren če...

Dejali bomo, da je eksponent r "primeren", če je (i) $r \in \mathbb{P}$, (ii) $r = O(\log^6 n)$ in (iii) $r - 1$ vsebuje praštevilski delitelj q velikosti vsaj $r^{\frac{1}{2}+\delta}$, $\delta > 0$, ter lastnostjo $q | o_r(n)$. Tako dobljen algoritem za testiranje praštevilskosti bomo poimenovali `PrastQ` in je predstavljen v okvirju Slika 2.1. Pokazali bomo, da če pri takem r velja kongruenca (2) za vsak a, $1 \le a \le 2\sqrt{n}\log n$, potem je n praštevilo. V algoritmu pogoj za velikost q preverimo s $q \ge 4\sqrt{r}\log n$, iz $n^{\frac{r-1}{q}} \not\equiv 1 \pmod{r}$ pa sledi $q | o_r(n)$, saj je $n^{r-1} \equiv 1 \pmod{r}$.

2.2 Časovna zahtevnost

časovne zahtevnosti (pod)programov

Program `PrastQ` si lahko ogledamo tudi na internetu[2]. Napisan je v Mathematici in uporablja funkcije `AnabQ, LargePrime, PoliPowerMod` in `CongruenceQ`.

`AnabQ[n]` preveri, ali je število n oblike a^b. Naj bo m tak, da je $n = m^m$, torej je $m = O(\log n)$. Algoritem preizkusi vse celoštevilske $a \in [2, m]$ in preveri, če $b = \log_a n \in \mathbb{Z}$, nato pa še vse celoštevilske $b \in [2, m]$ in preveri, če $a = \sqrt[b]{n} \in \mathbb{Z}$. Skupaj je to $2m$ korakov in časovna zahtevnost algoritma je $O(\log^3 n)$ bitnih operacij.

`LargePrime[n]` s preizkusom vseh možnih deliteljev manjših od $\sqrt{n}$ ugotovi največji praštevilski delitelj števila n in je časovne zahtevnosti $O(\sqrt{n})$

[2] `http://www.fmf.uni-lj.si/~novakta/research/PrastQ.htm`

Slika 1: Algoritem za testiranje prašteviloskosti

Procedure : PrastQ

Input: celo število $n > 1$
Output: SESTAVLJENO / PRAŠTEVILO

```
1.  if n je oblike a^b, b > 1 output SESTAVLJENO;
2.  r = 2;
3.  while r < n {
4.      if D(n, r) ≠ 1 output SESTAVLJENO;
5.      if r je praštevilo
6.          naj bo q največji praštevilski delitelj od r − 1;
7.          if q ≥ 4√r log n and n^((r−1)/q) ≢ 1 (mod r)
8.              break; (* while *)
9.      r ← r + 1;
10. }
11. for a = 1 to 2√r log n
12.     if (x − a)^n ≢ (x^n − a) (mod x^r − 1, n) output SESTAVLJENO;
13. output PRAŠTEVILO;
```

`PoliPowerMod[u,r,v,n]` izračuna $u(x)^r \bmod (v(x), n)$ s pomočjo algoritma 'kvadriraj in množi' (glej Stinson [16, str. 127]), kjer na vsakem koraku rezultat tudi reduciramo po modulu $x^r - 1$ in po modulu n. To storimo tudi na začetku izvajanja. Potrebnih je torej $O(\log r)$ korakov. Če uporabimo Fourierovo transformacijo za množenje polinomov (glej Bini in Pan [6, str. 8–13]) in še hitro metodo za deljenje polinomov (glej Bini in Pan [6, str. 18–23]), je časovna zahtevnost algoritma $\widetilde{O}((\log n)(\deg u + \log r \deg v))$.

`CongruenceQ[a,n,r]` nam s pomočjo `PoliPowerMod` preveri, ali je $(x-a)^n \equiv (x^n - a) \pmod{x^r - 1, n}$. S tem potroši $\widetilde{O}(r \log^2 n)$ bitnih operacij.

Za algoritem `PrastQ[n]` bomo s trditvijo 4 pokazali, da najde primeren r v največ $O(\log^6 n)$ korakih (algoritem se lahko ustavi še predno najde primeren r). Časovna zahtevnost enega koraka v `while` zanki je $\widetilde{O}(\log^3 n)$, v `for` zanki pa $\widetilde{O}(\log^8 n)$. Skupaj algoritem `PrastQ[n]` potroši največ $\widetilde{O}(\log^{12} n)$ bitnih operacij, kar pomeni, da je `PRAŠTEVILO` v P.

lemi o gostoti praštevil

Lema 2. *(Fouvry [8], Baker in Harman [4]) Obstajata konstanti $c > 0$ in n_0, tako da za vsak $n \geq n_0$ velja*

$$|\{p | p \text{ je praštevilo }, p \leq n \text{ in } P(p-1) > n^{2/3}\}| \geq c\frac{n}{\log n}$$

Lema 3. *(Apostol [2]) Naj bo $\pi(n)$ število praštevil $\leq n$. Potem za vsak $n \geq 1$ velja*

$$\frac{n}{6 \log n} \leq \pi(n) \leq \frac{8n}{\log n}$$

trditev, ki nam da polinomsko časovno zahtevnost za `PrastQ`

Trditev 4. *Obstajajo konstante c_1, c_2 in n_1 pri katerih za vsak $n \geq n_1$ obstaja praštevilo r na intervalu $[c_1 \log^6 n, c_2 \log^6 n]$ za katerega velja, da ima $r-1$ praštevilski delitelj q, ki zadošča pogojema $q \geq 4\sqrt{r} \log n$ in $n^{\frac{r-1}{q}} \not\equiv 1 \pmod{r}$.*

Dokaz. Vzemimo konstanti c in n_0 iz leme 2. Definirajmo c_1 in c_2 tako, da velja (i) $c_2 > c_1 > 0$, (ii) $\frac{c\,c_2}{7} - \frac{8c_1}{6} > 0$ ter (iii) $c_1^4 > 4^6 c_2^3$. Bralec si lahko nariše ustrezne pare (c_1, c_2) kot neprazno območje v $\mathbb{R}^2$. Praštevila p, za katera velja $P(p-1) > p^{2/3}$ poimenujmo 'posebna praštevila'.

S pomočjo leme 2 in leme 3 ocenimo, koliko je posebnih praštevil na intervalu $[c_1 \log^6 n, c_2 \log^6 n]$, kjer je $n \geq n_0$ in $\log n \geq c_2$:

$$\begin{aligned}
&|\{p \in \mathbb{P} \mid p \in [c_1 \log^6 n, c_2 \log^6 n], P(p-1) > p^{2/3}\}| \geq \\
&\quad \geq \quad |\{p \in \mathbb{P} \mid p \leq c_2 \log^6 n, P(p-1) > p^{2/3}\}| - |\{p \in \mathbb{P} \mid p \leq c_1 \log^6 n\}| \geq \\
&\quad \geq \quad c\frac{c_2 \log^6 n}{\log(c_2 \log^6 n)} - \frac{8(c_1 \log^6 n)}{\log(c_1 \log^6 n)} \\
&\quad \geq \quad \frac{\log^6 n}{\log\log n}\left(\frac{c\,c_2}{7} - \frac{8c_1}{6}\right) = c_3 \frac{\log^6 n}{\log\log n}
\end{aligned}$$

kjer smo definirali $c_3 = \frac{c\,c_2}{7} - \frac{8c_1}{6}$ in po definiciji za c_1 in c_2 je $c_3 > 0$.

Naj bo $x = c_2 \log^6 n$ in

$$N = (n-1)(n^2-1)(n^3-1)\cdots(n^{\lfloor x^{1/3} \rfloor} - 1)$$

Pokažimo, da obstaja posebno praštevilo na intervalu $[c_1 \log^6 n, c_2 \log^6 n]$, ki ne deli N. Najprej ocenimo število praštevilskih deliteljev števila N:

$$\begin{aligned}
&|\{p \in \mathbb{P} \mid p \text{ deli } N\}| \leq \\
&\quad \leq \quad \log N = \log(n-1) + \log(n^2-1) + \cdots + \log(n^{x^{1/3}} - 1) \leq \\
&\quad \leq \quad \log(n) + 2\log n + \cdots + x^{1/3}\log n = \frac{x^{1/3}(x^{1/3}+1)}{2}\log n
\end{aligned}$$

Za dovolj velik n in s tem tudi x je to manj kot $x^{2/3}\log n$, oziroma $c_2^{2/3}\log^5 n$, kar je za dovolj velik n manj kot $c_3 \frac{\log^6 n}{\log\log n}$. Torej obstaja posebno praštevilo na intervalu $[c_1 \log^6 n, c_2 \log^6 n]$, ki ne deli N. Izberimo enega in ga označimo z r. Pokažimo, da r ustreza pogojem iz trditve. Ocenimo velikost $q = P(r-1)$:

$$q \geq r^{2/3} \geq c_1^{2/3}\log^4 n = \frac{c_1^{2/3}}{c_2^{1/2}}(c_2 \log^6 n)^{1/2}\log n \geq 4\sqrt{r}\log n$$

Če hočemo pokazati $n^{\frac{r-1}{q}} \not\equiv 1 \pmod r$, je dovolj, da pokažemo $q = o_r(n)$ in $\frac{r-1}{q} < o_r(n)$. Ker r ne deli N, potem r ne deli $(n^i - 1)$ za noben i, $1 \leq i \leq \lfloor x^{1/3} \rfloor$, kar pomeni, da $n^i \not\equiv 1 \pmod r$ za noben i, $1 \leq i \leq \lfloor x^{1/3} \rfloor$, torej je $o_r(n) > x^{1/3}$.

Ker je $\frac{r-1}{q} \leq \frac{r}{r^{2/3}} = r^{1/3} \leq (c_2 \log^6 n)^{1/3} = x^{1/3} < o_r(n)$ in ker red elementa, deli moč grupe, torej $o_r(n)$ deli $(r-1)$, je $q = o_r(n)$. ■

2.3 Dokaz pravilnosti algoritma

Če je n praštevilo, se algoritem ne more ustaviti v vrsticah 1, 4 ali 12, torej nam vrne `PRAŠTEVILO`. Za dokaz pravilnosti v drugo smer pa bomo potrebovali nekaj splošnih dejstev iz algebre - o grupah in končnih obsegih. Nato bomo v lemi 7 definirali grupo G in ocenili njeno velikost. Definirali

bomo tudi množico $I_{g(x)}$ in v lemah 8 in 9 pokazali njene lastnosti, nato pa bomo v trditvi 10 s protislovjem pokazali, da če je n sestavljeno število, potem algoritem vrne `SESTAVLJENO`.

Lema 5. *Naj bo G grupa z enoto e in $a \in G$. Potem je*

$$a^n = e \iff \mathrm{red}(a) | n$$

■

polinomi, grupe in končni obsegi

Lema 6. *Naj bosta p in r praštevili in $p \neq r$. Potem velja:*

1. *Multiplikativna grupa obsega $F = \mathrm{GF}(p^d)$, $d > 0$, označena z F^*, je ciklična.*
2. *Če je $f(x)$ polinom s celoštevilskimi koeficienti, velja*

$$f(x)^p \equiv f(x^p) \pmod{p}$$

3. *Če je $h(x)$ katerikoli faktor od $x^r - 1$ in $m \equiv m_r \pmod{r}$, potem velja*

$$x^m \equiv x^{m_r} \pmod{h(x)}$$

4. *Polinom $\frac{x^r-1}{x-1}$ se faktorizira v nerazcepne polinome stopnje $o_r(p)$.*

Dokaz.
(1) Katerikoli učbenik iz algebre o končnih obsegih, npr. Vidav [17].

(2) Naj bo $f(x) = a_0 + a_1 x + \cdots + a_d x^d$. Koeficient pred x^i v $f(x)^p$ enak

$$\sum_{\substack{i_0+i_1+\cdots+i_d=p \\ i_1+2i_2+\cdots+di_d=i}} \frac{p!}{i_0! i_1! \cdots i_d!} a_0^{i_0} a_1^{i_1} \cdots a_d^{i_d}$$

Ta koeficient je deljiv s p natanko tedaj, ko so vsi $i_k < p$ v vsoti. Upoštevamo še Fermatov izrek in dobimo $f(x)^p \equiv \sum_{k=0}^{d} a_k x^{kp} \pmod{p}$. Dokaz je končan, saj je tudi $f(x^p) = \sum_{k=0}^{d} a_k x^{kp}$.

(3) Naj bo $m = kr + m_r$. Če kongruenco $x^r \equiv 1 \pmod{x^r - 1}$ na obeh straneh potenciramo na k in nato še obe strani pomnožimo z x^{m_r}, dobimo $x^m \equiv x^{m_r} \pmod{x^r - 1}$. Ker $h(x)$ deli $x^r - 1$, je $x^m \equiv x^{m_r} \pmod{h(x)}$.

(4) Naj bo $d = o_r(p)$, $Q(x) = \frac{x^r-1}{x-1}$, $h(x)$ nek nerazcepen faktor od $Q(x)$ in $k = \deg h(x)$. Pokažimo, da $k|d$ in $d \leq k$, torej da velja $k = d$.
Naj bo $F = \mathrm{GF}(p^k)$ definiran s polinomom $h(x)$ in naj bo $g(x)$ generator F^*. Po 2. točki leme je

$$g(x)^p \equiv g(x^p) \pmod{p}$$

oziroma, če to točko uporabimo d-krat, dobimo

$$g(x)^{p^d} \equiv g(x^{p^d}) \pmod{p}$$

Ker je $p^d \equiv 1 \pmod{r}$, je po 3. točki leme

$$g(x^{p^d}) \equiv g(x) \pmod{h(x)}$$

Predzadnja kongruenca velja tudi po modulu $h(x)$, saj če sta dva elementa enaka v kolobarju $\mathbb{Z}_p[x]$, potem predstavljata isti element v faktorskem kolobarju $\mathbb{Z}_p[x]/h(x)$, ki pa je izomorfen $\mathbb{Z}[x]/(h(x), p)$. Podobno velja za zadnjo kongruenco in zato je

$$g(x)^{p^d} \equiv g(x) \pmod{h(x), p}$$

Torej $h(x)$ deli $g(x)(g(x)^{p^d-1} - 1)$ v obsegu F. Ker je $h(x)$ nerazcepen v F, deli vsaj enega od faktorjev. Če bi $h(x)$ delil $g(x)$, bi bil $g(x)$ ničeln element v F, torej je

$$g(x)^{p^d-1} \equiv 1 \pmod{h(x)}$$

Ker je $g(x)$ generator grupe F^*, je njegov red enak moči te grupe, $\mathrm{red}(g(x)) = p^k - 1$. Po lemi 5 velja $p^k - 1 | p^d - 1$, to pa je res natanko tedaj, ko $k|d$.
Pokažimo še, da velja $d \leq k$. Ker $h(x)$ deli $x^r - 1$ v $\mathbb{Z}_p[x]$, je

$$x^r \equiv 1 \pmod{h(x), p}$$

Iz leme 5 sledi, da $\mathrm{red}(x)$ deli r in ker je $r \in \mathbb{P}$ in ker x ni enica v F, je $\mathrm{red}(x) = r$. Ker red vedno deli moč grupe, velja $r | p^k - 1$, torej $p^k \equiv 1 \pmod{r}$. Po definiciji d je $d \leq k$. ■

nadaljevanje dokaza pravilnosti

Naj bo n sedaj sestavljeno število in predpostavimo, da se algoritem ne ustavi v vrsticah 1 ali 4. Naj bo r praštevilo, pri katerem se ustavi zanka `while`, in $q = P(r-1)$. Potem obstaja praštevilski faktor p števila n, tak da q deli $o_r(p)$.

Dokaz. Naj bo $n = \prod_i p_i^{k_i}$. Potem $o_r(n)$ deli $\mathrm{v}_i(o_r(p_i))$, kjer je v oznaka za najmanjši skupni večkratnik (če bi bil $n = \prod_i p_i$, potem bi bil $o_r(n) = \mathrm{v}_i(o_r(p_i))$). Ker q deli $o_r(n)$ (eden od pogojev za primernost r), potem q deli tudi $\mathrm{v}_i(o_r(p_i))$ in ker je q praštevilo, obstaja i, tak da $q | o_r(p_i)$. ■

Ker iz $(x-a)^n \equiv (x^n - a) \pmod{x^r - 1, n}$ sledi

$$(x-a)^n \equiv (x^n - a) \pmod{h(x), p}$$

lahko na kongruence iz algoritma gledamo kot na kongruence v obsegu $F = \mathbb{Z}_p[x]/h(x)$, kjer je $h(x)$ nerazcepen faktor od $x^r - 1$ stopnje $d = o_r(p)$, glej lemo 6.4.

lema ključnega pomena (moč grupe G)

Lema 7. *Naj bo $l = 2\sqrt{r}\log n$. V obsegu $F = \mathbb{Z}_p[x]/h(x)$ je multiplikativna podgrupa*

$$G = \left\{ \prod_{1 \leq a \leq l} (x-a)^{\alpha_a} | \alpha_a \geq 0 \right\}$$

ciklična in moči $> (\frac{d}{l})^l \geq n^{2\sqrt{r}}$.

Dokaz. Ker je F^* ciklična, je tudi $G \leq F^*$ ciklična. Ocenimo še velikost grupe G s pomočjo množice

$$S = \left\{ \prod_{1 \leq a \leq l} (x-a)^{\alpha_a} \mid \alpha_a \geq 0, \sum_{1 \leq a \leq l} \alpha_a \leq d-1 \right\}$$

Ker se algoritem ne ustavi v vrsticah 1 ali 4, nam najde r, za katerega velja $r > q \geq 4\sqrt{r}\log n > l$ in $\mathrm{D}(n,b) = 1$ za vsak $b \leq r$. Če bi bila katera $a_1, a_2 \in \mathbb{N}$, $1 \leq a_1 < a_2 \leq l$, kongruentna po modulu p, potem bi p delil $a_2 - a_1$, kar pa je v protislovju z $\mathrm{D}(n, a_2 - a_1) = 1$, saj je $0 < a_2 - a_1 < a_2 \leq l < r$.

Ker so vsi polinomi v S stopnje $\leq d-1 < \deg h(x)$ in ker so vsi a-ji različni po modulu p, nam elementi v S predstavljajo paroma različne elemente iz F. Vsak polinom v S je produkt $d-1$ faktorjev, pri čemer imamo $l+1$ možnih faktorjev na izbiro, to so 1, $x-1$, $\ldots$, $x-l$. Iz formule za kombinacije s ponavljanjem dobimo

$$\begin{aligned} |G| &\geq |S| = \binom{(d-1)+(l+1)-1}{d-1} = \binom{d+l-1}{l} = \\ &= \frac{(d+l-1)(d+l-2)\cdots d}{l!} > \frac{d^l}{l^l} = \left(\frac{d}{l}\right)^l \end{aligned}$$

Ker q deli $o_r(p)$, je $d = o_r(p) \geq q \geq 4\sqrt{r}\log n = 2l$ in zato

$$|G| \geq \left(\frac{2l}{l}\right)^l = 2^l = 2^{2\sqrt{r}\log n} = n^{2\sqrt{r}}$$

■

Naj bo $g(x)$ generator grupe G. Označimo z o_g red elementa $g(x)$ v $\mathbb{Z}_p[x]/h(x)$. Enak je moči grupe G in zato velik vsaj $n^{2\sqrt{r}}$. Definirajmo *def.* $I_{g(x)}$

$$I_{g(x)} = \{m \in \mathbb{N} \mid g(x)^m \equiv g(x^m) \pmod{x^r - 1, p}\}$$

in pokažimo nekaj njenih osnovnih lastnosti.

Lema 8. *Množica $I_{g(x)}$ je zaprta za množenje.*

Dokaz. Naj bosta $m_1, m_2 \in I_{g(x)}$. To pomeni

$$g(x)^{m_1} \equiv g(x^{m_1}) \pmod{x^r - 1, p} \tag{3}$$

$$g(x)^{m_2} \equiv g(x^{m_2}) \pmod{x^r - 1, p} \tag{4}$$

Če v kongruenci (4) pišemo x^{m_1} namesto x in upoštevamo, da $x^r - 1$ deli $x^{m_1 r} - 1$, dobimo $g(x^{m_1})^{m_2} \equiv g(x^{m_1 m_2}) \pmod{x^r - 1, p}$, kar nam da skupaj s kongruenco (3)

$$g(x)^{m_1 m_2} \equiv g(x^{m_1 m_2}) \pmod{x^r - 1, p}$$

to pa pomeni, da $m_1 m_2 \in I_{g(x)}$. ■

Lema 9. *Če sta $m_1, m_2 \in I_{g(x)}$, potem iz $m_1 \equiv m_2 \pmod{r}$ sledi $m_1 \equiv m_2 \pmod{o_g}$.*

Dokaz. Ker je $m_2 \in I_{g(x)}$, je $g(x)^{m_2} \equiv g(x^{m_2}) \pmod{h(x), p}$, saj $h(x)$ deli $x^r - 1$. Ker je $m_1 \equiv m_2 \pmod{r}$, je $m_2 = m_1 + kr$ in če upoštevamo še lemo 6.3, dobimo $g(x)^{m_1} g(x)^{kr} \equiv g(x^{m_1}) \pmod{h(x), p}$ in ker $m_1 \in I_{g(x)}$, dobimo, da $h(x)$ deli $g(x)^{m_1}(g(x)^{kr} - 1)$ v $\mathbb{Z}_p[x]$. Ker je $h(x)$ nerazcepen nad $\mathbb{Z}_p$, velja $h(x)$ deli $g(x)$ ali $h(x)$ deli $g(x)^{kr} - 1$. Če bi bil $g(x)$ deljiv s $h(x)$, bi bil ničeln element v F. Torej je $g(x)^{kr} \equiv 1 \pmod{h(x), p}$ in po lemi 5 o_g deli kr in s tem $m_2 - m_1$. ■

konec dokaza pravilnosti

Trditev 10. *Če je n sestavljeno število, algoritem vrne* `SESTAVLJENO`.

Dokaz. Recimo, da je n sestavljeno število in da algoritem vrne `PRAŠTEVILO`. Pokažimo najprej, da je $\{1, p, n\} \subset I_{g(x)}$. Po definiciji $I_{g(x)}$ je očitno, da $1 \in I_{g(x)}$, iz druge točke leme 6 pa sledi $p \in I_{g(x)}$. Če upoštevamo, da je $g(x) \in G$, torej da je oblike

$$g(x) = \prod_{1 \le a \le l} (x - a)^{\gamma_a}$$

in še, da so pravilne vse kongruence v zanki `for`, saj algoritem vrne `PRAŠTEVILO`, dobimo

$$g(x)^n \equiv g(x^n) \pmod{x^r - 1, p}$$

Sedaj definirajmo

$$E = \{n^i p^j | 0 \le (i, j) \le \lfloor \sqrt{r} \rfloor\}$$

Ker je $\{1, p, n\} \subset I_{g(x)}$ in ker je $I_{g(x)}$ zaprta za množenje, je $E \subset I_{g(x)}$. Ker je $|E| = (1 + \sqrt{r})^2 > r$, obstajata po Dirichletovem principu $n^{i_1} p^{j_1}$ in $n^{i_2} p^{j_2}$ iz E, $(i_1, j_1) \neq (i_2, j_2)$ taka, da velja $n^{i_1} p^{j_1} \equiv n^{i_2} p^{j_2} \pmod{o_g}$. Ker je $o_g > n^{2\sqrt{r}}$ in ker sta $n^{i_1} p^{j_1}$, $n^{i_2} p^{j_2} \le n^{2\sqrt{r}}$, je $n^{i_1} p^{j_1} = n^{i_2} p^{j_2}$. Torej je $n^{i_1 - i_2} = p^{j_2 - j_1}$ in algoritem bi se moral ustaviti v vrstici 1. To pa je v protislovju s predpostavko, da algoritem vrne `PRAŠTEVILO`. ■

3 Pospešitve in odprti problemi

Predstavili bomo dve hipotezi s področja, ki obravnava praštevila. Če velja hipoteza 11, potem je časovna zahtevnost algoritma `PrastQ` največ $\widetilde{O}(\log^6 n)$, s pomočjo hipoteze 13 pa dobimo deterministični algoritem časovne zahtevnosti $\widetilde{O}(\log^3 n)$.

Če sta r in $\frac{r-1}{2}$ praštevili, imenujemo $\frac{r-1}{2}$ **Sophie Germain** praštevilo in r **co-Sophie Germain** praštevilo.

hipoteza 1 časovna zahtevnost je v praksi $\widetilde{O}(\log^6 n)$

Hipoteza 11. *(Hardy in Littlewood [10], preverjena za $r \le 10^{10}$) Število co-Sophie Germain praštevil manjših od n je asimptotično enako $D\frac{n}{\log^2 n}$, kjer je $D \in \mathbb{R}$.*

Konstanti D rečemo **konstanta praštevilskih dvojčkov**. Če hipoteza velja, potem nam naslednja lema zagotavlja, da je časovna zahtevnost algoritma `PrastQ` največ $\widetilde{O}(\log^6 n)$.

Lema 12. *Če velja hipoteza 11, potem obstajajo take pozitivne konstante n_0 in c_1 in c_2, da obstaja 'primeren' r na intervalu $[c_1 \log^2 n, c_2 \log^2 n]$ za vsak $n \geq n_0$.*

Dokaz. Zaradi preprostejšega izračuna v dokazu vzemimo $c_1 = 100$ in $c_2 = 101$, čeprav bi bila ustrezna že $c_1 \approx c_2 \approx 64$. Definirajmo množico $T = [c_1 \log^2 n, c_2 \log^2 n] \cap \mathbb{P}_{co-Sop-Ger}$ in s pomočjo hipoteze 11 asimptotično ocenimo njeno moč:

$$|T| \approx \frac{D(c_2 \log^2 n)}{(\log(c_2 \log^2 n)^2} - \frac{D(c_1 \log^2 n)}{(\log(c_1 \log^2 n)^2} \approx \frac{D(c_2 - c_1)\log^2 n}{4(\log\log n)^2}$$

Pokažimo, da za praštevila $r \in T$ velja $P(r) \geq 4\sqrt{r}\log n$:

$$\begin{aligned} P(r) &= \frac{r-1}{2} \geq \frac{c_1 \log^2 n - 1}{2} \geq \frac{c_1 - 1}{2}(\log n)(\log n) \geq \\ &\geq \frac{c_1 - 1}{2}\sqrt{\frac{r}{c_2}}\log n \geq \frac{c_1 - 1}{2\sqrt{c_2}}\sqrt{r}\log n \geq 4\sqrt{r}\log n \end{aligned}$$

Na intervalu $[c_1 \log^2 n, c_2 \log^2 n]$ je asimptotično $O\left(\frac{\log^2 n}{(\log\log n)^2}\right)$ co-Sophie Germain praštevil. Neprimerna so tista, pri katerih je $n^{\frac{r-1}{q}} \equiv 1 \pmod r$, torej $n^2 \equiv 1 \pmod r$, kar pomeni $n^2 - 1 = kr$. Neprimernih je torej največ $\log(n^2 - 1) \approx 2\log n$, saj ima $n^2 - 1$ največ $\log(n^2 - 1)$ praštevilskih deliteljev (možnosti za r). Seveda pa je $O(\log n)$ asimptotično manj kot $O\left(\frac{\log^2 n}{(\log\log n)^2}\right)$, torej za dovolj velik n obstaja 'primeren' $r \in T$. ■

drugačen algoritem
- preprostejši
- hitrejši
($\widetilde{O}(\log^3 n)$)

Sedaj bomo predstavili še hitrejši algoritem, katerega pravilnost še ni dokazana. Da smo za `PrastQ` dobili grupo G moči vsaj $n^{2\sqrt{r}}$, smo morali vzeti $l = 2\sqrt{r}\log n$. Konstanto l lahko ustrezno zmanjšamo in še vedno dobimo grupo zahtevane moči.

Hipoteza 13. *(Bhattacharjee in Pandey [5][3], preverjena v Kayal in Saxena [12][4] za $r \leq 100$ in $n \leq 10^{10}$) Če r ne deli n in če velja*

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

potem je n praštevilo ali pa $n^2 \equiv 1 \pmod r$

Če hipoteza 13 drži, lahko sestavimo hiter algoritem za testiranje praštevilskosti, glej okvir Slika 3. Poiščemo r, ki ne deli niti n niti $n^2 - 1$. Po lemi 14 obstaja na intervalu $[2, 6\ln n]$. Nato za ta r preverimo kongruenco iz hipoteze 13 in časovna zahtevnost algoritma je tako omejena z $\widetilde{O}(\log^3 n)$.

Lema 14. *Na intervalu $[2, 6\ln n]$ obstaja r, ki ne deli niti n niti $n^2 - 1$.*

Dokaz. Po učbeniku Apostol [2] za analitično teorijo števil povzamemo $\prod_{\substack{p \in \mathbb{P} \\ p \leq x}} p \geq e^{\frac{x}{2}}$ za vsak $x \geq 5$. Če bi vsa praštevila $p \leq 6\ln n$ delila $n^2 - 1$ ali n, torej njun produkt $n^3 - n$, potem bi bil $\prod_{\substack{p \in \mathbb{P} \\ p \leq 6\ln n}} p \leq n^3 - n$. Iz zgornje ocene pa vemo, da je $\prod_{\substack{p \in \mathbb{P} \\ p \leq 6\ln n}} p \geq e^{\frac{6\ln n}{2}} = n^3$. ■

[3] `http://www.cse.iitk.ac.in/research/btp2001/primality.htm`
[4] `http://www.cse.iitk.ac.in/research/btp2002/primality.htm`

Slika 2: Algoritem za testiranje praštevilskosti, pravilnost temelji na hipotezi 13

Procedure : `PrastHipoQ`

Input: `celo število` $n > 1$
Output: `SESTAVLJENO / PRAŠTEVILO`

```
1.  r = 2;
2.  while r < n {
3.      if n ≡ 0 (mod r) output SESTAVLJENO;
4.      if n^2 ≢ 1 (mod r) break; (* while *)
5.      r ← r + 1;
6.  }
7.  if (x − 1)^n ≡ (x^n − 1) (mod x^r − 1, n)
8.      output PRAŠTEVILO;
9.  else
10.     output SESTAVLJENO;
```

4 Iskanje praštevil v praksi

algoritem za iskanje

V praksi mnogokrat potrebujemo naključno izbrano veliko praštevilo. Pri implementaciji RSA kriptosistema, glej Stinson [16, str. 124], potrebujemo za 1024 bitni modul naključno izbrano 512 bitno praštevilo. Dobimo ga tako, da izberemo naključno 512 bitno liho število in preverimo, ali je praštevilo. Glede na lemo 3 najdemo praštevilo v $O(\log(n))$ korakih. Če bi v tem programu uporabljali algoritem `PrastQ`, bi ob predpostavki, da velja hipoteza 11, potrošil največ 512^6 bitnih operacij (od vseh korakov pri iskanju praštevila je najdaljši tisti, ko najdemo praštevilo). Ker je $512^6 = 1.8 \cdot 10^{16}$, bi računalnik z $1GHz = 10^9 Hz$ procesorjem za poiskati eno 512 bitno praštevilo potreboval velikostnega reda 10^7 sekund, oziroma 4 mesece, kar pa je nesprejemljivo.

verjetnostni algoritmi za iskanje so hitrejši

Zato se v praksi še vedno uporabljajo verjetnostni algoritmi za problem `PRAŠTEVILO`. Verjetnostni algoritmi potrebujejo funkcijo za generiranje naključnih števil. Posebna zvrst teh algoritmov so Monte-Carlo algoritmi, pri katerih je posebnost to, da vrnejo enega od dveh odgovorov, npr. `TRUE` ali `FALSE`, in eden od odgovorov je vedno pravilen, drugi pa je lahko tudi napačen.

Najbolj znana sta Solovay-Strassenov test ter Miller-Rabinov test, glej Stinson [16, str. 129–138]. Čeprav je Miller-Rabinov test hitrejši, si bomo raje ogledali Solovay-Strassenov, glej okvir Slika 4, saj je na videz bolj preprost in bomo z njim lažje prikazali, kako delujejo Monte-Carlo algoritmi.

opis časovne zahtevnosti

Časovna zahtevnost testa je $O(\log^2 n)$, definicije in postopka za izračun vrednosti Jacobijevega simbola $\left(\frac{a}{n}\right)$ ne bomo opisali, saj presega okvir članka. Pri delovanju testa je pomembno to, da če vrne `SESTAVLJENO`, potem je n res sestavljeno število, če pa vrne `PRAŠTEVILO`, pa odgovor ni nujno pravilen. Verjetnost napake je v našem primeru manjša od 1/2

$$P(\text{algoritem vrne } \texttt{PRAŠTEVILO} \mid n \text{ ni praštevilo}) < 1/2$$

Slika 3: Solovay-Strassenov verjetnostni test za testiranje praštevilskosti

Solovay-Strassenov test

Procedure : `PrastVerjTestQ`

Input: `celo število` $n > 1$

Output: `SESTAVLJENO / PRAŠTEVILO`

```
1.  naključno izberi celo število a, 1 ≤ a ≤ n − 1;
2.  if (a/n) ≡ a^((n−1)/2) (mod n)
3.      output PRAŠTEVILO;
4.  else
5.      output SESTAVLJENO;
```

Če test ponovimo m-krat potem z nekaj truda (glej Stinson [16, str. 135]) za velike n izpeljemo

$$P(n \text{ ni praštevilo} \mid \text{algoritem } m\text{-krat vrne } \texttt{PRAŠTEVILO}) \leq \frac{\ln n - 2}{\ln n - 2 + 2^{m+1}}$$

in ne $1 - (1/2)^m$, kakor bi intuitivno pričakovali. Če test ponovimo $\log n$-krat in vedno dobimo odgovor `PRAŠTEVILO`, je torej verjetnost, da n ni praštevilo velikostnega reda $\frac{1}{n}$, kar je dovolj zanesljivo za uporabo v praksi. Časovna zahtevnost Solovay-Strassenovega algoritma za preverjanje, ali je n praštevilo, je torej $O(\log^3 n)$. Prav tolikšna je tudi za Miller-Rabinov algoritem. Podobno kot prej lahko dobimo, da na $1GHz$ računalniku stestiramo 512-bitno praštevilo v približno 0,13 sekunde.

Praštevila v Mathematici

V programskem paketu `Mathematica 4` sta za testiranje praštevilskosti vgrajeni funkciji `PrimeQ` in `ProvablePrimeQ`, pri slednji je potrebno predhodno naložiti paket `NumberTheory‘PrimeQ‘`.

delovanje PrimeQ

Za testiranje praštevilskosti funkcija `PrimeQ` najprej preveri, če ima število n kak majhen praštevilski delitelj, nato uporabi Miller-Rabinov test (krepki psevdopraštevilski test) pri bazi 2 in bazi 3, na koncu pa še Lucasov test[5].

delovanje ProvablePrimeQ

Funkcija `ProvablePrimeQ` uporablja verjetnostni algoritem, ki vedno vrne pravilen rezultat. Pričakovana vrednost za časovno zahtevnost algoritma je polinomska za skoraj vsa naravna števila, oz. za vsa pod neko hipotezo. Algoritem uporablja eliptične krivulje in sta ga prva predstavila Goldwasser in Kilian [9], Atkin pa je s pomočjo uporabe idej množenja kompleksnih števil (del teorije števil, ki združuje kompleksno analizo, Galoisovo teorijo in modularne forme) pokazal, kako bi se ga dalo uporabiti v praksi, glej [3]. To je leta 1989 storil Morain [13] in to je tudi prvi algoritem, ki je certificiral več kot 1000 mestna praštevila (titanic primes) brez upoštevanja njihovih posebnih lastnosti (npr. praštevila oblike $10^k + 1$). Za certificiranje praštevila $10^{100} + 267$ potrebuje Mathematica na 300 MHz računalniku 35 sekund.

Čeprav je problem certificiranja praštevil že globoko raziskan, bomo še vedno čakali na bolj eleganten algoritem. Mogoče še dolgo časa, saj so za varnost RSA kriptosistema bolj pomembni rezultati in algoritmi za problem faktorizacije, ki je še vedno odprt.

Literatura

[1] M. Agrawal, N. Kayal, N. Saxena, *PRIMES is in P*, Preprint,

[5] `http://mathpages.com/home/kmath473.htm`

http://www.cse.iitk.ac.in/primality.pdf, 6. avg. 2002.

[2] T. M. Apostol, *Introduction to Analytic Number Theory*, Springer-Verlag, 1996.

[3] A. O. L. Atkin, F. Morain, *Elliptic Curves and Primality Proving*, Mathematics of Computation, 1993, 29-68.

[4] R. C. Baker, G. Harman, *The Brun-Titchmarsh Theorem on average*, Proceedings of a conference in Honor of Heini Halberstam, Volume 1, 1996, 39–103.

[5] R. Bhattacharjee, P. Pandey, *Primality testing. Technical report*, IIT Kanpur, 2001.

[6] D. Bini, V. Pan, *Polynomial and Matrix Computation: Fundamental Algorithms, Volume 1*, Birkhäuser, BostonBaselBerlin, 1994.

[7] D.Bressoud, *Factorization and Primality Testing*, SpringerVerlag, 1989.

[8] E. Fouvry, *Theoreme de Brun-Titchmarsh application au theoreme de Fermat*, Invent. Math., **79** (1985), 383–407.

[9] S. Goldwasser, J. Kilian, *Almost All Primes Can Be Quickly Certified*, Proceedings of 18th STOC, 1986, 316-329.

[10] G. H. Hardy and J. E. Littlewood, *Some problems of 'Partitio Numerorum' III: On the expression of a number as a sum of primes*, Acta Mathematica, **44** (1922), 1–70.

[11] E. Horowitz, S. Sahni, *Fundamentals of Computer Algorithms*, Computer Science Press, Rockville, 1978.

[12] N. Kayal, N. Saxena, *Towards a deterministic polynomial-time test. Technical report*, IIT Kanpur, 2002.

[13] F. Morain, *Implementation of the AtkinGoldwasserKilian Primality Testing Algorithm*, INRIA Research Report, #911, Oktober 1988.

[14] R. Petek, *RSA kriptosistem, diplomsko delo*, Ljubljana, 2000.

[15] T. Pisanski, *Izračunljivost in rešljivost*, Obzornik za mat. in fiz., **25** (1978), 41–54.

[16] D. R. Stinson, *Cryptography: Theory and Practice*, CRC Press, 1995.

[17] I. Vidav, *Algebra, 2. natis*, DMFA SRS, IMFM in FNT, Ljubljana, 1980.