Technická univerzita Košice
Fakulta elektrotechniky a informatiky
Katedra elektroniky a multimediálnych telekomunikácií

Zadanie z predmetu

Aplikovaná kryptografia

Vypracovali: Roman Garaj
Štefan Asztalos
Marcel Bandžuch
Viktor Fábry

4.3 (True) Primality tests

Prvotný test v tomto odseku je postup pri ktorom kladné celé čísla môžu byť overované ako prvočíslo, a sú často používané ako prvotná previerka algoritmov. Tento pôvodný test je obvykle viac výpočtovo intenzívny než pravdepodobnostný prvotný test. V dôsledku toho, skôr než použijeme jedno z týchto kritérií ku kandidátovi prvočísla n, kandidát by mal byť podriadený pravdepodobnostnému prvotnému testu takému ako Miller-Rabinov algoritmus.

4.34 Definícia: Celé číslo n ktoré je určené prvočíslom na základ prvotného preskúšania algoritmu sa nazýva preukázateľné prvočíslo.

4.3.1 Testovanie Mersennových čísel

Účinný algoritmus je známy pre prvotnú analýzu špeciálnych tried čísel, takých ako Mersenneho čísla a Fermatove čísla. Mersennove prvočísla n sú vhodné pretože aritmetika v poli Z_n pre také n môže byť realizovaná veľmi účinne. Lucas-Lehmerov test pre Mersennove čísla je algoritmus.

4.35 Definícia: Nech $s \geq 2$ je celé číslo. Mersennove číslo je celé číslo tvaru $2^s - 1$. Ak $2^s - 1$ je prvočíslo, potom sa nazýva Mersennove prvočíslo.

Ďalšie nutné a postačujúce podmienky pre Mersenneove číslo ktoré je prvočíslom.

4.36 Fakt: Nech $s \geq 3$. Mersennove číslo $n = 2^s - 1$ je prvočíslo ak a len ak nasledovné dve podmienky sú splnené:
(1) s je prvočíslo; a
(2) postup celých čísel definovaných $u_0 = 4$ a $u_{k+1} = \left(u_k^2 - 2\right) \bmod n$ pre $k \geq 0$ vyhovuje $u_{s-2} = 0$.

Fakt 4.36 vedie k nasledovnému deterministickému mnohočlenu algoritmu určujúcemu (s istotou) či Mersennove číslo je prvočíslo.

4.37 algoritmus Lucas-Lehmer prvotný test pre Mersennove čísla
VSTUP: Mersenne číslo $n = 2^s - 1$ s $s \geq 3$.
VÝSTUP: odpoveď "prvočíslo" alebo "zložené" k otázke: "je n prvočíslo?"
1. Použitie vzoru na kontrolu ak s má nejaké činiteľe medzi 2 a $\lfloor \sqrt{s} \rfloor$. Ak to urobí, potom je výsledok „zložené".
2. Položiť $u = 4$.
3. Pre k od 1 do $s - 2$ urobte nasledovné: $u \leftarrow \left(u^2 - 2\right) \bmod n$.
4. Ak $u = 0$ odpoveď „prvočíslo“. Inak, odpoveď „zložené“.

Tabuľa 4.2: Známe Mersennove prvočísla.

index j	M_j	desiatková číslica	index j	M_j	desiatková číslica
1	2	1	18	3217	969
2	3	1	19	4253	1281
3	5	2	20	4423	1332
4	7	3	21	9689	2917
5	13	4	22	9941	2993
6	17	6	23	11213	3376
7	19	6	24	19937	6002
8	31	10	25	21701	6533
9	61	19	26	23209	6987
10	89	27	27	44497	13395
11	107	33	28	86243	25962
12	127	39	29	110503	33265
13	521	157	30	132049	39751
14	607	183	31	216091	65050
15	1279	386	32?	756839	227832
16	2203	664	33?	859433	258716
17	2281	687			

Tabuľa ponúka 33 známych exponent $M_j, 1 \le j \le 33$, ku ktorému $2^{M_j} - 1$ je Mersennovo prvočíslo, a tiež niekoľko desiatkových číslic v $2^{M_j} - 1$. Otáznik po j = 32 a j = 33 ukazuje, že nie je známe či tam je každý iný exponent s medzi M_{31} a týmito číslami pre ktorý $2^s - 1$ je prvočíslo.

4.3.2 Primality test s použitím faktorizácie n – 1

Táto sekcia uvádza výsledky ktoré sa môžu používať k preskúšaniu že celé číslo n je prvočíslo, za predpokladu , že faktorizácia alebo čiastková faktorizácia $n-1$ je známa. To sa môže zdať ako nadbytočné uvažovanie techniky ktorú požaduje faktorizácia $n-1$ ako podriadený problém, ak celé číslo tohoto čísla môže byť faktor, prvotné n samo o sebe môže byť určené pomocou faktora n. Ale, faktorizácia $n-1$ môže byť ľahšie počítateľná ak n má špeciálny tvar, taký ako Fermatovo číslo $n = 2^{2^k} + 1$. Iná situácia, kde faktorizácia $n-1$ môže byť jednoducho počítateľná, je keď kandidát n je "konštruovaný" pomocou špeciálnych metód.

4.38 Fakt: Nech $n \ge 3$ je celé číslo. Potom n je prvočíslo ak a len ak existuje celé číslo zodpovedajúce podmienkam:
(1) $a^{n-1} \equiv 1 (\mathrm{mod}\, n)$;a
(2) $a^{n-1/q} \neq 1 (\mathrm{mod}\, n)$ pre každé prvočíslo deliteľa q z $n-1$.
Tento záver nasleduje z faktu že Z_n^* má časť rozkladu $n-1$ ak a len ak n je prvočíslo; prvok spĺňajúci podmienky (1) a (2) majú rozklad $n-1$.

4.39 poznámka (pôvodný test založený na fakt 4.38) ak n je prvočíslo, číslo častí spĺňajúce podmienky $n-1$ je určitý $\phi(n-1)$. Z toho dôvodu, dôkaz kandidát n prvočíslo, jeden jednoducho zvolí celé číslo $a \in Z_n$ náhodou a používa fakt 4.38 na kontrolu ak má splnenú podmienku $n-1$.Ak toto existuje , potom n je určite prvočíslo. Inak, iné $a \in Z_n$ je vybrané a test je opakovaný. Ak n je skutočne prvočíslo, očakávané číslo iterácie skôr než časť podmienky $n-1$ je vybraný je $O(\ln \ln n)$; toto nasledovať od $(n-1)/\phi(n-1) < 6 \ln \ln n$ 1) pre $n > 5$. Teda, ak také nie je možné nájsť po "rozumnom" množstve (napr.:12 ln ln n) iterácii, potom *n* je pravdepodobne kompozitný a musí byť zase podriadený pravdepodobnostnému prvočíselnému testu ako je napr.: Miller-Rabinov test (Algorithm 4.24). Tento postup je, vskutku, jeden pravdepodobnostne zložený test.

Tento výsledok dáva metóda pre overenie prvočíselnosti, ktorá požaduje znalosť len jedného čiastkového rozloženia na činitele *n - 1*.

4.40 Fakt *(Pocklingtonova teoréma)* Nech *n ≥ 3* je celé číslo, a nech *n = RF +1* (t.j . F delenie n - 1) kde prvočíselné rozloženie *F* na činitele je $F = \prod_{j=1}^{t} q_j^{e_j}$.

Keď tam existuje celé číslo *a* zodpovedajúce podmienkam:

(1)$a^{n-1} \equiv 1 \pmod n$;
(2)gcd $(a^{(n-1)/q_j} - 1, n) = 1$ pre každé *j* ; $1 \le j \le t$,

Potom každý deliteľ prvočísla *p* pre *n* je kongruentný k 1 modulo *F*. Z toho vyplýva že ak $F > \sqrt{n} - 1$,potom n je prvočíslo.
Ak *n* je skutočne prvočíslo, potom tento výsledok zriadi, že väčšina celých čísel *a* spĺňajúce podmienky (i) a (ii) faktu 4.40, za predpokladu $F > \sqrt{n} - 1$ sú dosť veľké.

4.41 Fakt Nech *n = RF +1* je nepárne prvočíslo s $F > \sqrt{n} - 1$ a s *gcd(R,F) = 1*.Nech rozdielne prvočíselné činitele *F* sú $q_1, q_2; : : : ; q_t$. Potom pravdepodobnosť, že náhodne vybraná báza *a* $1 \le a \le n-1$,vybaví obide podmienky (i) $a^{n-1} \equiv 1 \pmod n$; a aj (ii) gcd $(a^{(n-1)/q_j} - 1, n) = 1$ pre každé *j* ; $1 \le j \le t$, je $\prod_{j=1}^{t}(1 - 1/q_j) \ge 1 - \sum_{j=1}^{t} 1/q_j$

Teda, ak rozložíme na *n - 1* činiteľov deliteľa $F > \sqrt{n} - 1$,potom budeme ho poznať pre test prvočíselnosti *n*, prvé môže jednoducho zvoliť ako náhodné celé číslo *a* z intervalu [2; n - 2] až kým sa nevyskytne jeden, ktorý spĺňa podmienky (1) a (2) faktu 4.40, čo bude znamenať, že *n* je prvočíslo. Ak také nie je možné nájsť po "rozumnom" množstve iterácii, potom n je pravdepodobne kompozitný a môže byť podriadený k pravdepodobnostnému prvočíselnému testu.
Táto metóda je, vskutku, pravdepodobnostný zložený test. Nasledujúci výsledok dáva metóda pre preverenie prvočíselnosti ktorá požaduje len rozloženie na (n-1) činiteľov deliteľa *F*, ten je väčší ako $\sqrt[3]{n}$.

4.41 Fakt Nech n≥ 3 je nepárne celé číslo. Nech *n = 2RF + 1*, a predpokladáme, že tam existuje celé číslo *a* zodpovedajúci obom podmienkam: (i) $a^{n-1} \equiv 1 \pmod n$; a aj (ii) gcd $(a^{(n-1)/q} - 1, n) = 1$ pre každý deliteľ prvočísla *q*. Nech *x* ≥ 0 a *y* je definovaný pomocou $2R = xF + y$ a $0 \le y < F$.

Ak $F = \sqrt[3]{n}$ a $y^2 - 4x$ nie je 0 ani ideálna mocnina, vtedy je *n* prvočíslo.

4.3.3 Jacobiho sčítací test

Jacobiho sčítací test je iný správny test prvočíselnosti. Základná myšlienka je skúšať skupinu zhôd ktoré sú analogické s Fermatovou vetou v istom cyclotomic znení. Doba výpočtu Jacobiho sčítacieho testu pre určenie prvočíselnosti pre celé číslo *n* je $O\left((\ln n)^{c \ln\ln\ln n}\right)$ bitová operácia pre istú konštantu *c*. Verzia Jacobiho sčítacieho prvočíselného testu používaného v praxi je náhodný algoritmus ktorý je ohraničený $O\left(k(\ln n)^{c \ln\ln\ln n}\right)$ krokmi s pravdepodobnosťou najmenej $1-\left(\frac{1}{2}\right)^k$ pre všetky $k \geq 1$ a stále dáva správnu riešenie. Jednou nevýhodou algoritmu je to, že netvorí "certifikát" ktorý by umožňoval kontrolovať odpoveď vo veľmi krátkom čase než sám algoritmus prebieha.
Jacobiho sčítací test je, skutočne, praktický v tom zmysle, že s prvočíslami ktoré majú dĺžku aj viacero sto desiatkových číslic môžeme spracovať v počítači za niekoľko minút. Ale, tento test nie je možné tak jednoducho naprogramovať ako pravdepodobnostný Miller-Rabinov test a výsledný kľuč nie je kompaktný.

4.3.4 Test použitívania eliptických oblúkov

Algoritmy eliptického oblúku skúšania prvočíselnosti sú založené na eliptickom oblúku analogického Pocklingtonovým teorémom (fakt 4.40). Táto verzia algoritmu používaná v praxi sa zvyčajne odvoláva na Atkinsov test alebo na algoritmus eliptického oblúku skúšania prvočíselnosti (ECPP). Pri heuristických argumentoch, predpokladaná doba výpočtu pre tento algoritmus pre skúšanie prvočíselnosti celého čísla *n* je možné ukázať na $O\left((\ln n)^{6+\varepsilon}\right)$ bitových operáciách pre každé $\varepsilon > 0$. Atkinsov test má výhodu nad Jacobiho sčítacím testom (4.3.3) v tom, že tvorí krátke certifikáty prvočíselnosti ktoré môžeme používať k výkonnému preverovaniu prvočíselnosti. Atkinsov test sa používa pri dokazovaní prvočíselnosti čísel dlhých viac než 1000 desiatkových číslic.

Zdrojovy text k algoritmu Lucas-Lehmer pre Mersennove čísla v prog. Jazyku C++:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>


int main(void)
{
        unsigned int  s,i,j;
        unsigned long n,nn,u;
        double  tmp;

        printf("\n\nZadaj n vacsie ako 7: ");
        scanf("%u",&n);


   s=0;
        nn=n;
        nn=nn+1;
        while(nn!=0 && nn!=1 ){s++;nn=nn/2;}
        printf("Exponent s=%d",s);

        if((unsigned long)(pow(2,s)-1)!=n)
                {printf("\nZadane cislo nesplnuje vstupnu podmienku (n=2^s-1)!\n");
                printf("\n Stlac lubovolny klaves pre ukoncenie...");
                getch();
                return 0;}

        tmp = sqrt((double)s);
        i = (unsigned int)tmp;
        if((tmp - (double)i) != 0)
                ++i;

        printf("\nCislo %u ",n);

        for(j = 2; j <= i; j++)
        {
                if((s % j) == 0)
                {
                        printf("nie je prvocislo\n\nStlac lubovolny klaves pre ukoncenie...\n\n");
                        getch();
                        return 0;
                }
        }

        u = 4;
        for(j = 1; j < s - 1; j++)
        {
                u = (((u * u) - 2) % n);
        }
        if(u == 0)
                printf("je prvocislo\n\nStlac lubovolny klaves pre ukoncenie...\n\n");
        else
                printf("nie je prvocislo\n\nStlac lubovolny klaves pre ukoncenie...\n\n");
                getch();
        return 0;
```