4. Універсальна SH-модель обчислювача складається з двох SH-моделей – функціональної та керуючої. Вона дає змогу оптимізувати обчислювальні системи за характеристиками складності.

1. Черкаський М.В. SH-модель алгоритму // Вісник НУ "Львівська політехніка" – Львів , 2001, №433, с.127-134. 2. Черкаський М.В., Мурад Хусейн Халіл. Універсальна SH-модель // Вісник НУ "Львівська політехніка" – Львів , 2004, №523, с. 150-154. 3. Cherkaskyy M. Theoretical Fundamentals Software/Hardware Algorithms. //Modern Problems of Radio Engineering, Telecommunications and Computer Science. Proceedings of the International Conference TCSET"2004. February 24-28, 2004, Lviv – Slavsko, Ukraine. Lviv, Publishing House of Lviv Polytechnic, 2004, p. 9–13. 4. Черкаський Микола, Мурад Хусейн Халіл. Комп'ютерні алгоритмічні системи. // Вісник НУ "Львівська політехніка" – Львів , 2004, №508, с.274-280. 5. Трахтенброт Б.А. Алгоритмы и вычислительные автоматы. – М. 1974. 6. George F.Luger Artificial Intelligence. Fourth Edition, Addison Wesley, 2003.

УДК 681.322

Р. Оберишин, Р.Б. Попович
Національний університет "Львівська політехніка"
кафедра електронних обчислювальних машин

ПРО ТЕСТУВАННЯ ВЕЛИКИХ НАТУРАЛЬНИХ ЧИСЕЛ НА ПРОСТОТУ

Виконано порівняння відомих сучасних тестів простоти великих натуральних чисел. Проаналізовано детермінований поліноміальний тест простоти, запропонований Агравалом, Кайалом та Саксеною.

A comparison of different known today big integers primality tests is done. Deterministic polynomial-time primality test offered by Agrawal, Kayal and Saxena is analysed.

Вступ. Криптографія – це захист інформації за допомогою її перетворення, що унеможливлює прочитання цієї інформації сторонньою особою. Причиною бурхливого розвитку криптографії є широке використання комп'ютерних мереж, зокрема глобальної мережі Internet, по яких передаються великі обсяги інформації державного, військового, комерційного й приватного характеру, що не допускає можливості доступу до неї сторонніх осіб. Широковживаним є алгоритм RSA шифрування з відкритим ключем. Багато провідних світових IT-компаній вклали в його розвиток значні кошти, він став основою функціонування Internet-платежів. Алгоритм RSA використовується також в системі електронних платежів НБУ – загальнодержавній платіжній системі, що забезпечує здійснення розрахунків між банківськими установами, органами державного казначейства на території України із застосуванням електронних засобів приймання, опрацювання, передавання та захисту інформації.

Генерація ключів для криптосистем з відкритим ключем, схеми електронного підпису передбачають генерацію великих випадкових простих чисел. У такому разі прості числа тримають в таємниці від зловмисника. Щоб позбавити зловмисника шансів зламати криптосистему, ці прості числа треба вибирати у випадковий спосіб і вони мають бути великими.

Огляд відомих тестів простоти. Є два типи тестів простоти [1-3]: ймовірнісні та детерміновані. Імовірнісні тести мають меншу асимптотичну скаладність, а детерміновані – більшу складність [2].

Імовірнісні алгоритми дають відповідь про простоту з деякою можливою ймовірністю помилки. Якщо ми збираємось використати прості числа для "індустріальних" застосувань (наприклад, для шифрування RSA), нам часто не потрібно доводити їхню простоту. Може бути достатньо знати,

що ймовірність того, що ці числа можуть бути складеними, менша від $(1/4)^{200}$. Ця ймовірність помилки незначна. Проте математики багато років все ж намагаються позбутися її зовсім.

Основою тестів простоти є мала теорема Ферма [2]: якщо n – просте ціле число, то для будь-якого цілого числа a, яке не має спільних дільників з n,

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

Теорема Ферма дає потужний тест складеності. Маючи $n>1$, вибираємо $a>1$ та обчислюємо a^{n-1} за модулем n (є дуже простий спосіб швидко зробити це послідовними піднесеннями до квадрата, так зване "бінарне піднесення до степеня"). Якщо результат не дорівнює одиниці за модулем n, то n складене. Якщо результат – одиниця за модулем n, то n може бути простим. У такому разі n називають слабким імовірнісним простим за основою a.

Може бути порівняно мало слабких імовірнісних простих чисел, які є складеними, але їх є нескінченна кількість для кожної основи $a>1$, тому нам потрібний сильніший тест. Одним із способів зробити тест точнішим є використання багатьох основ (перевірити за основою 2, тоді за 3, тоді за 5 і т.д.). Але тут ми стикаємось з "аномалією", яка називається числа Карміхаеля. Складене число n є числом Карміхаеля, якщо $a^{n-1}=1 \pmod{n}$ для кожного цілого a взаємно простого з n.

Повторюваний слабкий імовірнісний тест простоти для числа Карміхаеля не покаже, що воно складене, поки ми не натрапимо на один з його дільників. Хоч числа Карміхаеля зустрічаються "рідко", в 1994 було показано, що їх є нескінченна кількість [2].

Кращий спосіб зробити тест Ферма точнішим є використати факт, що коли непарне число n просте, то існують рівно два квадратні корені з числа 1 за модулем n: 1 та -1. Отже, квадратний корінь з a^{n-1}, $a^{(n-1)/2}$ (оскільки n непарне), є 1 або -1 (Ми могли б знайти, чому дорівнює цей корінь за допомогою символу Якобі, але ми хочемо збудувати сильніший тест). Якщо $(n-1)/2$ парне, можна знову добувати квадратний корінь. У результаті отримуємо такий алгоритм.

Запишемо $n-1=2^s d$ де d непарне та s невід'ємне: n є сильним імовірнісним простим за основою a, якщо $a^d=1 \pmod{n}$ або $(a^d)^{2^r}=-1 \pmod{n}$ для деякого невід'ємного r, меншого від s.

Знову всі числа $n>1$, які не проходять цей тест, є складеними; числа, що проходять його, можуть бути простими. Доведено [2], що сильний імовірнісний тест простоти є неправильним не більше ніж у 1/4 випадків (3 з 4 чисел, які проходять його, будуть простими).

Є два типи детерміновані алгоритмів: алгоритми, які спираються на недоведені припущення, наприклад, розширену гіпотезу Рімана, та алгоритми, які не спираються ні на які недоведені припущення, наприклад, циклотомічний метод. Якщо вважати, що відповідне припущення неправильне, перші названі детерміновані алгоритми можна розглядати як імовірнісні.

У 1983 Адлеман, Померанце та Румелі запропонували детермінований тест простоти (ще його називають циклотомічним методом [6]). Вони пробували багаточлени n^m-1 для показників m таких, як 5040, тоді кожне просте q таке що $q-1$ ділить 5040 (яке не ділить n) повинно ділити $n^{5040}-1$ (за малою теоремою Ферма). Алгоритм має таку складність $(\log n)^{O(\log \log \log n)}$. Це так звана квазіполіноміальна складність.

Коен, Ленстра, Босма, ван дер Хулст та Міхайлеску спростили алгоритм як теоретично, так і алгоритмічно. (Вони вдосконалили його заміною загального закону взаємності для показника степеня за модулем на багато простіші для обчислень суми Якобі). Цей алгоритм ефективний на практиці, числа з 2000 десяткових розрядів можуть бути протестовані за приблизно 10^{14} тактів.

Аткін розробив алгоритм доведення простоти на еліптичних кривих (ECPP алгоритм) [4,6]. Він є ефективним на практиці. Використання ECPP означає перехід від використання кілець до деяких спеціальних груп. ECPP був використаний для доведення простоти чисел з більш ніж 1000 десяткових розрядів.

Зауважимо, що застосовуються також детальніші класифікації тестів простоти [6]. У такому разі для кожного тесту вказують:

1) у чому полягає метод: короткий виклад твердження, яке представляє метод. Наприклад: якщо n не є слабким імовірнісним простим за основою a, тобто $a^n \neq a \bmod n$, то n складене. Числом, що досліджується, завжди є n. Допоміжні дані (такі, як a у прикладі) називають сертифікатом;

2) результат сертифікації: що метод твердить про число, що досліджується (доводить простоту, доводить складеність, за припущенням сертифікує простоту);

3) сертифікат існує, щоб встановити: які числа може опрацьовувати (сертифікувати) метод (кожне просте, за припущенням кожне просте, кожне складене, майже кожне складене, майже кожне просте);

4) час перевірки сертифіката: наскільки швидко можна перевірити, чи допоміжні дані є сертифікатом для числа n. Наприклад, можливі такі оцінки часу перевірки $(\log n)^{1+o(1)}$, $(\log n)^{2+o(1)}$, $(\log n)^{O(\log \log \log n)}$.

5) час знаходження сертифіката: наскільки швидко можна знайти сертифікат. Термін "випадковий" у такому разі означає, що алгоритм пошуку сертифіката використовує випадковість.

Постановка задачі. Відомі тести простоти мають недоліки та переваги. Дослідження тестів є актуальною задачею, оскільки виявлені недоліки та переваги сприяють ефективнішому їхньому використанню та створенню нових удосконалених тестів.

Детермінований поліноміальний алгоритм. В 2002 році Аграваль, Каял і Саксена запропонували поліноміальний детермінований алгоритм AKS перевірки цілих чисел на простоту [4], який не спирається ні на які недоведені припущення. Його асимптотична складність $O((\log n)^{12} f(\log \log n))$, де n – ціле число, що перевіряється, і f – деякий багаточлен. log означає логарифм за основою два. Отже, позитивно вирішене досі відкрите питання про належність задачі тестування простоти до класу P. Високий показник степеня 12 – це верхня межа, яку вдалося отримати авторам. Передбачається, що реальна складність алгоритму може бути нижчою. Евристично, алгоритм працює набагато краще: на підставі добре відомої гіпотези про частоту простих чисел Софі Жермен, значення показника 12 можна замінити на 6.

AKS базується на такій простій версії малої теореми Ферма [4]:

припустимо, що a і p є взаємно прості цілі числа ($p > 1$). p просте тоді і тільки тоді, коли

$$(x-a)^p = (x^p-a) \pmod{p}.$$

У такій формі цей факт доволі важко використати, оскільки потрібно перевірити надто багато коефіцієнтів. Тому використовується простіша умова:

$$(x-a)^p = (x^p-a) \pmod{x^r-1, p}$$

для відповідно вибраного r.

Алгоритм AKS доводить простоту за допомогою комбінаторики: якщо можна записати багато елементів певної підгрупи циклотомічного розширення кільця Z_n, то n є степенем простого числа.

Псевдокод алгоритму AKS виглядає так:

```
Вхід: натуральне n > 1
if (n має вигляд a^b, де b > 1)
then вихід СКЛАДЕНЕ
    r := 2
while (r < n)
if НСД (n,r)≠ 1 then вихід СКЛАДЕНЕ
if (r просте більше 2) then {
нехай q найбільший дільник r-1
if (q > 4sqrt(r)log n) and n^((r-1)/q) ≠ 1 (mod r)
    then break
}
r := r+1
for a = 1 to 2sqrt(r)log n
    {
    if ((x-a)^n ≠(x^n-a)(mod x^r-1,n))
    then вихід СКЛАДЕНЕ
}
then вихід ПРОСТЕ
```

Наведений алгоритм повертає ПРОСТЕ тоді і тільки тоді, коли n – просте число.

Алгоритм має два цикли. Перший цикл (while) намагається знайти таке просте r, що $r-1$ має великий простий дільник $q \geq 4 \text{ sqrt}(r) \log n$ та q ділить $o_r(n)$, де $o_r(n)$ є порядком n за модулем r

Другий цикл (for) алгоритму використовується для перевірки: чи виконується наведена раніше рівність за модулем x^r-1 та p . Цей крок є "вузьким місцем" у цьому алгоритмі.

На цьому кроці алгоритм AKS обчислює $(x-a)^n$ у кільці $Z_n[x]/(x^r-1)$ для всіх a від 1 до $s=2\text{sqrt}(r) \log n$. Стандартний підхід до виконання цього завдання передбачає послідовні піднесення до квадрата.

Інший можливий шлях: перехід від елементів кільця Z_n $[x]/(x^r-1)$ до $Z[x]/(x^r-1)$, а потім до $Z/(2^{kr}-1)$, де $k= [\lg rn^2]$. Множення виконуємо в останньому кільці з використанням швидких алгоритмів множення цілих чисел. Тоді відтворюємо результат у початковому кільці.

Також можна пробувати розкладати багаточлен x^r-1 на взаємно прості множники й виконувати окремі обчислення за модулем кожного із множників на підставі китайської теореми про залишки.

Хоч наведений вище алгоритм і показує поліноміальність задачі перевірки простоти чисел, реальна складність цього алгоритму настільки висока, що наразі він має тільки теоретичне значення. Це пов'язано, з одного боку, з тим, що асимптотична оцінка починає ефективно працювати тільки для достатньо великих значень n. Тому перший цикл алгоритму знайде шукане число r також тільки для достатньо великих значень n, а для менших простих чисел дасть відповідь $r = n$, що фактично означатиме перевірку простоти повним перебиранням усіх можливих дільників числа n. З іншого боку, відомий циклотомічний неполіноміальний детермінований алгоритм, запропонований Адлеманом, Померанцем та Румелі, і поліпшений Коеном та Ленстрою. Він має складність $(\log n)^{O(\log \log \log n)}$, що при всіх практично значущих значеннях n дає кращу оцінку, ніж у вказаного поліноміального алгоритму.

Варто також зауважити, що на практиці часто зручніше використовувати значно швидші ймовірнісні алгоритми, на зразок розглянутих у другому підрозділі тестів.

Сьогодні запропоновано низку вдосконалень вказаного алгоритму [4-8]. Частина із них пов'язана із зменшенням чисел r та s, які фігурують в алгоритмі AKS.

Дійсно, "вузьке місце" алгоритму AKS – його другий цикл – вимагає виконання приблизно $s \log n$ піднесень до квадрата натуральних чисел, кожне з яких має приблизно $2r \log n$ бітів. Обчислення можна пришвидшити за допомогою "низькорівневих" вдосконалень у піднесенні цілих чисел до квадрата та за допомогою "високорівневих" уточнень величин r та s. Прогрес у "високорівневих" вдосконаленнях сьогодні оцінюється гіпотетичною асимптотикою добутку $r\ s$, поділеною на величину $(\log n)^4$.

Для оригінальному алгоритму ця оцінка виглядала так: $r\ s/(\log n)^4 \in 1024 + o(1)$. Після низки вдосконалень отримано оцінку $r\ s/(\log n)^4 \in 0{,}0005026686484... + o(1)$.

Отже, швидкодія алгоритму покращена на множник 2037127,2....+ $o(1)$. Звичайно, у такому разі "швидше в два мільйони разів" ще не означає "швидше".

Авторами проаналізовано запропоновані в літературі вдосконалення, які дали змогу отримати вказану оцінку. Усі вони включені в модифікований алгоритм AKS, псевдокод якого наведено нижче:

```
Вхід: натуральне число n > 1
if (n має вигляд a^b ,де b > 1) then вихід СКЛАДЕНЕ
r := 3
while (r < n)
{
if (n примітивний корінь за модулем r) break
r:=r+1;
}
for d:=0 to φ(r)-1 do
for i:=0 to d do
for j:=0 to φ(r)-d-1 do
```

{
s:=0;
while not $(2s,i)(d,i)(2s-i,j)(\varphi(r)-d-1,j)\geq h^{[\sqrt{\varphi(r)}/3]}$

do s:=s+1;
}
for b:=2 to s+1 do
{
if НСД(n,r) $\neq 1$ then вихід СКЛАДЕНЕ
if $b^{n-1} \neq 1 (\text{mod } n)$ then вихід СКЛАДЕНЕ
}
for b:=2 to s+1 do
for c:=2 to s+1 do
{
if НСД(n,b-c) $\neq 1$ then вихід СКЛАДЕНЕ
if НСД($n,bc-1$) $\neq 1$ then вихід СКЛАДЕНЕ
}
for b:=2 to s+1
if $(x-b)^n \neq (x^n-b) \ (\text{mod } x^r-1,n)$) then вихід СКЛАДЕНЕ
вихід: ПРОСТЕ;

Зауважимо, що крім описаних раніше вдосконалень алгоритму, запропоновано також алгоритми, які заміняють циклотомічні розширення кільця Z_n на випадкові розширення Куммера цього самого кільця [7,8].

Висновки. Виконано порівняння різних відомих сьогодні тестів великих натуральних чисел на простоту. Найкращим сьогодні детермінованим алгоритмом вважають циклотомічний метод, який дає змогу тестувати числа з декількома тисячами десяткових розрядів.

Проаналізовано запропонований Агравалом, Кайалом і Саксеною поліноміальний детермінований алгоритм. Цей алгоритм потребує детальніших досліджень. Незважаючи на те, що цей тест є детермінованим і поліноміальним, його використання сьогодні ускладнено невисокою швидкістю роботи. Нині вимагається побудувати "розумний" алгоритм, який працюватиме швидше ніж розглянутий вище. Після подальших вдосконалень цей алгоритм, можливо, конкуруватиме з циклотомічним методом та методом на основі еліптичних кривих.

1. Вербіцький О. В. Вступ до криптології. Львів; 1998. 2. Ємець В.Ф., Мельник А.О., Попович Р.Б. Сучасна криптографія. Основні поняття. Львів, 2003. 3. Шнайер Б. Прикладная криптография. Протоколи, алгоритми, исходные тексти на язике Си. М.; 2003. 4. M. Agrawal, N. Kayal and N. Saxena, PRIMES is in P. http://www.cse.iitk.ac.in/news/primality.pdf. 5. D. J.Bernstein, Proving primality after Agrawal, Kayal and Saxena. http://cr.yp.to/papers.html#aks. 6. D.J. Bernstein, Distinguishing prime numbers from composite numbers: the state of the art in 2004. http: // cr.yp.to/papers.html#prime2004. 7. P.Berrizbeitia, Sharpening Primes is in P for a large family of numbers. http://archiv.org/abs/math/NT/0211334. 8. Q.Cheng, On the bounded sum-of-digits discrete logarithm problem in finite field. http://www.cs.ou.edu/~qcheng/pub.html.