**УДК 681.322**

**Р.Б. Попович**
Національний університет “Львівська політехніка”,
кафедра електронних обчислювальних машин

# ДЕЯКІ ЗАУВАЖЕННЯ ВІДНОСНО РЕАЛІЗАЦІЇ AKS ТЕСТУ ПРОСТОТИ



**Проаналізовано поліноміальний детермінований AKS тест простоти та його модифікації з погляду практичної реалізації. Наведено експериментальні дані.**

**They have analized a polynomial deterministic AKS primality test and its modifications from point of view of practical implementation. Experimental data are given.**

## Вступ

Прості числа мають фундаментальне значення в математиці загалом і в теорії криптографії зокрема. Криптографія – це захист інформації шляхом її перетворення, що виключає прочитання цієї інформації сторонньою особою. Причиною бурхливого розвитку криптографії є широке використання комп'ютерних мереж, зокрема глобальної мережі Internet, якими передаються великі обсяги інформації державного, військового, комерційного й приватного характеру, що не допускає можливості доступу до неї сторонніх осіб.

Ефективні тести простоти використовують також на практиці: низка широковживаних криптографічних протоколів потребує великих простих чисел. Генерація ключів для криптосистем з відкритим ключем, схеми електронного підпису передбачають генерацію великих випадкових простих чисел. У цьому разі прості числа тримаються в таємниці від зловмисника. Щоб позбавити зловмисника шансів зламати криптосистему, ці прості числа треба вибирати випадково і вони мають бути великими.

$\varphi(r)$ позначатиме функцію Ейлера, яка дає число цілих, менших від $r$ та взаємно простих з $r$; $|S|$ – кількість елементів множини $S$; $o_r(n)$ – мультиплікативний порядок цілого $n$ за модулем $r$. Скорочення нсд надалі означатиме найбільший спільний дільник двох цілих чисел.

## Огляд відомих тестів простоти

Виділяємо два типи тестів простоти [1–4]: алгоритми доведення простоти числа (утворюють сертифікат простоти) та алгоритми доведення складеності числа (утворюють сертифікат складеності). Кожен із цих двох типів тестів може завершувати свою роботу за наперед передбачуваний час (детермінований алгоритм) або час до завершення не можна передбачити (імовірнісний алгоритм).

Якщо алгоритм доведення складеності числа не довів складеності числа $n$ за певне число спроб, то таке число називаємо ймовірнісним простим. Тобто це число є простим з деякою можливою ймовірністю помилки. Якщо ми збираємось використати прості числа для „прикладних” застосувань, нам часто не потрібно доводити їх простоту. Може бути достатньо знати, що ймовірність того, що ці числа можуть бути складеними, є достатньо малою. Власне це ми й маємо в описаній ситуації.

Ефективні ймовірнісні алгоритми доведення складеності відомі давно. Водночас алгоритми доведення простоти залишились з точки зору швидкості виконання далеко позаду [4]. Треба зауважити, що сьогодні на практиці у багатьох випадках використовують порівняно швидші ймовірнісні алгоритми доведення складеності.

Основою тестів простоти є мала теорема Ферма [1, 2]: якщо $n$ – просте ціле число, то для будь-якого цілого числа $a$, яке не має спільних дільників з $n$,

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

На жаль, виконання наведеної умови навіть для всіх цілих $a$ взаємно простих з $n$ ще не ґарантує простоту числа $n$. Складене число, яке задовольняє наведену умову для всіх взаємно простих з ним, називаємо числом Карміхаеля. Таким, наприклад, є 561=3·11·17. Хоч числа Карміхаеля зустрічаються "рідко", в 1994 р. було показано, що множина таких чисел нескінченна [2].

Найшвидший детермінований алгоритм доведення простоти запропонований 1983 р. Адлеманом, Померанце та Рамлі та вдосконалений низкою авторів (Коен, Х. Ленстра, Міхайлеску) [4]. Основу цього алгоритму становить перевірка умов, аналогічних малій теоремі Ферма, в полях алгебраїчних чисел. Алгоритм має таку асимптотичну складність $(\log n)^{O(\log\log\log n)}$. Це так звана квазіполіноміальна складність. Алгоритм ефективний на практиці, з його використанням протестовано числа з кількома тисячами десяткових розрядів.

Найкращим сьогодні практичним, але не детермінованим алгоритмом доведення простоти довільних чисел вважається метод ECPP, який використовує обчислення на еліптичних кривих над кільцями залишків $Z_n$. Він був запропонований в роботі Гольдвассер та Кіліана й вдосконалений Аткіном, Шаллітом [4]. За деяких правдоподібних припущень про розподіл простих чисел цей алгоритм для будь-якого простого числа $n$ доводить його простоту в середньому за $O(\log^4 n)$. В основу цього алгоритму покладено твердження, в деякому сенсі подібне до малої теореми Ферма. Метод ECPP було використано для доведення простоти чисел з 5000 десяткових розрядів.

## Постановка задачі

Відомі тести простоти мають з погляду практичної реалізації свої недоліки та переваги. Дослідження цих тестів є актуальним завданням, оскільки виявлені недоліки та переваги сприяють ефективнішому їх використанню та створенню нових удосконалених тестів.

## Детермінований поліноміальний алгоритм

У серпні 2002 року Агравал, Каял і Саксена запропонували алгоритм AKS перевірки цілих чисел на простоту [5]. Ідея алгоритму полягає в доведенні простоти за допомогою комбінаторики: якщо можна записати багато елементів простого циклотомічного розширення кільця $Z_n$ цілих чисел за модулем $n$, то $n$ степінь простого числа. Отже, позитивно вирішене досі відкрите питання про приналежність задачі тестування простоти до класу P алгоритмів поліноміальної складності.

Ключове значення AKS полягає в тому, що це перший опублікований алгоритм, який одночасно є поліноміальним, детермінованим та не спирається на жодні недоведені припущення. Тобто, максимальний час виконання алгоритму можна подати як поліном від числа розрядів тестованого числа; він ґарантує вирішення задачі: є число простим чи складеним (а не отримання ймовірнісного результату); і його коректність не залежить від коректності якоїсь допоміжної недоведеної гіпотези (наприклад, гіпотези Рімана).

Сьогодні асимптотичну складність алгоритму оцінюють як $O((\log n)^6 f(\log\log n))$, де n – ціле число, що перевіряється, і $f$ – деякий багаточлен. log означає логарифм за основою два.

Запропоновано низку вдосконалень вказаного алгоритму [6–11]. Частина із них пов'язана із зменшенням чисел $r$ та $s$, які фігурують в алгоритмі AKS.

D. Bernstein сформулював у [6] теорему, яка враховує всі ці вдосконалення.

***Теорема.*** Нехай $n$ та $r$ – додатні цілі числа. Нехай $d$, $i$ та $j$ – невід'ємні цілі числа. Нехай $S$ – скінченна множина цілих чисел з 0,1,-1 $\notin$S. Припустимо, що $n$ – примітивний корінь за модулем $r \geq 3$;

що нсд($n$, $b$-$b'$)=1 для всіх різних $b$, $b' \in S$

що нсд($n$, $bb'$-1)=1 для всіх $b$, $b' \in S$

що $b^{n-1}=1 \bmod n$ для всіх $b \in S$;

що $\binom{2|S|}{i}\binom{d}{i}\binom{2|S|-i}{j}\binom{\varphi(r)-1-d}{j} \geq n^{\lceil\sqrt{\varphi(r)/3}\rceil}$

та що $(x-b)^n=x^n-b \pmod{n,x^r-1}$ для всіх $b \in S$. Тоді $n$ є степінем простого числа.

Маючи додатне ціле $n$, застосовуємо теорему, як описано нижче.

Перевірити, чи $n$ є степенем цілого числа. Якщо так, то $n$ – складене.

Знайти таке найменше $r \geq 3$, що $n$ – примітивний корінь за модулем $r$. Вибрати

ціле $d$ між 0 та $\varphi(r)$-1;

ціле $i$ між 0 та $d$;

ціле $j$ між 0 та $\varphi(r)$-1-$d$.

Вибрати таке ціле $s$, що $\binom{2s}{i}\binom{d}{i}\binom{2s-i}{j}\binom{\varphi(r)-1-d}{j} \geq n^{\lceil\sqrt{\varphi(r)/3}\rceil}$

Покласти S={2,3,…,s}

Перевірити, чи нсд($n$, $b$-$b'$)=1 для всіх різних $b$, $b' \in S$. Якщо ні, то $n$ складене.

Перевірити, чи нсд($n$, $bb'$-1)=1 для всіх $b$, $b' \in S$. Якщо ні, то $n$ складене.

Перевірити, чи $b^{n-1}=1 \bmod n$ для всіх $b \in S$. Якщо ні, то $n$ є складеним.

Перевірити, чи $(x-b)^n = x^n - b \pmod{n, x^r-1}$ для всіх $b \in S$. Якщо ні, то $n$ є складеним.

Якщо на жодному з кроків не отримано відповідь, що $n$ складене число, то $n$ є простим числом.

Проведено низку експериментів, пов'язаних з реалізацією алгоритму AKS. Їх виконували з використанням середовища програмування MS Visual Studio 6.0 та бібліотеки Miracle для роботи з великими цілими числами. Згадана бібліотека дає змогу виконувати операції додавання, множення, піднесення до степеня, знаходження найбільшого спільного дільника тощо для цілих чисел, які мають тисячі десяткових розрядів.

У результаті:

1) практично в усіх експериментах $r$ виявлялось простим числом, а не складеним;

2) практично в усіх експериментах $o_r(n) = r$ -1, тобто $n$ – примітивний корінь за модулем $r$;

3) в усіх експериментах $r$ виявлялось порядку $(\log n)^2$.

Переважно тест простоти розглядають неявно, вважаючи, що під час його програмної реалізації можна користуватися як завгодно великим обсягом оперативної (швидкодіючої) пам'яті і оцінюють лише обсяг потрібних обчислень. Насправді це не так.

І коли дійсно треба реалізувати тест, обсяг пам'яті є не менш серйозною проблемою, ніж час виконання.

Наприклад, нехай число має 500 десяткових розрядів (тобто 1660 біт), що є в діапазоні практичних інтересів.

Утворено випадкове ціле число із 500 десятковими розрядами та з використанням найпростіших перевірок (пробні ділення на малі прості числа й проби Ферма) знайдено найближче імовірно просте число:

$n$=47940917743537973692644475812229992529786814872499245562292570755330643813134833171
00458499922259011023504448867777071492685879122518530652291843944762804120812191729618
87681039791349635530458534138304162846593055485126172037102380279251953595226831353710
55658904063286578635833582214067827360654259330234894372332159730250159696606450783603
91073101715445822433684928993232433968828438559337786465588280324886300918982672383912
96818317568544802394638532517956101892005378546853479529063224213885215 03

Шляхом підбору знайдено, що $n$ є примітивним коренем за модулем простого числа $r$=2755759; $\varphi(r)=o_r(n)$=27555758.

Зауважимо, що безпосереднє застосування наведеної раніше нерівності для великих $n$ є проблематичним. Тому ми брали логарифми лівої та правої частин.

Візьмемо в нерівності згідно з наведеною теоремою $s$=0.0497$\varphi(r)$=136961; $i$=$j$=0.047$\varphi(r)$=129520; $d$=0.5$\varphi(r)$=1377879. У результаті остання нерівність виконується: логарифм за основою 2 лівої частини нерівності дорівнює 1581626.22, а логарифм за основою 2 правої частини дорівнює 1581407.72.

Багаточлени, які виникають при перевірці 136961 рівностей в алгоритмі, а саме при обчисленні $(x-b)^n \pmod{n, x^r-1}$, для всіх $b \in S$ повинні мати степінь $r$ -1, тобто записуватися $r$ коефіцієнтами, кожен з яких є залишком за модулем $n$. Тоді пам'ять, необхідна для запису одного

багаточлена, не менша, ніж $r \cdot \log n = 2755759 \cdot 1660$ біт=545 Мбайт. Навіть беручи до уваги різні підходи до зменшення $r$, ще неможливо тримати багаточлен степеня $r$ в кеші процесора.

З цього погляду детермінована версія алгоритму, запропонована Х. Ленстрою та Померанце [8], має лише теоретичне значення ускладнення обчислень нічим не виправдано: адже практично значення $r$ завжди порядку $(\log n)^2$.

Відносно ймовірнісної версії, запропонованої Бернштейном [7]: теоретично вона найшвидша, але практично не реалізована, бо в ній для перевірки треба працювати з поліномами від двох змінних. Зауважимо, що ця версія заміняє циклотомічні розширення кільця $Z_n$ на випадкові розширення Куммера цього ж кільця [7, 10]. Для неї потрібний ще більший обсяг пам'яті порівняно з попереднім випадком.

Для порівняння ECPP (доведення простоти на еліптичних кривих) не вимагає багато пам'яті. Щоб зробити алгоритм AKS реалізовним на персональному комп'ютері, дуже бажаним є економне з погляду пам'яті піднесення $x$-$b$ до степеня. Під час реалізації тесту простоти, можливо, краще дотримуватися ECPP і запустити ітерацію AKS, коли проміжне просте стає „зручним".

Отже, з погляду практичної реалізації початковий варіант тесту AKS з вдосконаленнями (і, можливо поєднано з методом ECPP) є найвідповіднішим.

**Висновки**

Проведено низку експериментів, пов'язаних з реалізацією AKS тесту простоти. Їх виконували з використанням середовища програмування MS Visual Studio 6.0 та бібліотеки Miracle для роботи з великими цілими числами.

У результаті:

1) практично в усіх експериментах $r$ виявлялось простим числом, а не складеним;

2) практично в усіх експериментах $o_r(n) = r - 1$, тобто $n$ – примітивний корінь за модулем $r$;

3) в усіх експериментах $r$ виявлялось порядку $(\log n)^2$.

Проблемою при реалізації цього алгоритму є потреба у великому обсязі пам'яті.

З цього погляду версії алгоритму, запропоновані Х. Ленстрою та Померанце, Бернштейном мають сьогодні лише теоретичне значення: практично вони нереалізовні.

З погляду практичної реалізації початковий варіант тесту з вдосконаленнями виглядає найвідповіднішим.

Хоча алгоритм AKS є детермінованим та поліноміальним, його реальна складність настільки висока, що він має тільки теоретичне значення. Це пов'язано з тим, що асимптотична оцінка починає ефективно працювати тільки для достатньо великих значень числа $n$.

Після подальших вдосконалень цей алгоритм зможе конкурувати з 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: БаК, 2003. 3. Шнайер Б. Прикладная криптография. Протоколи, алгоритми, исхолние тексти на язике Си. – М.: Триумф, 2003. 4. Bernstein D.J. Distinguishing prime numbers from composite numbers: the state of the art in 2004. http://cr.yp.to/papers.html#prime 2004. 5. Agrawal M., Kayal N. and Saxena N. PRIMES is in P. Annals of Mathematics" – 2004. 160, No. 2. – P. 781–793. 6. Bernstein D.J. Proving primality after Agrawal, Kayal and Saxena. http://cr.yp.to/papers.html#aks. 7. Granville A. It is easy to determine whether a given integer is prime. Bulletin of the American Mathematical Society. – 2005. – Vol. 47, No.1. – P. 3–38. 8. Lenstra Jr., Pomerance Jr. and Carl. Primality Testing with Gaussian Periods, http://www.math.dartmouth.edu/~carlp/PDF/complexity12.pdf, preliminary version July 20, 2005. 9. Voloch J.F. On some subgroups of the multiplicative group of finite rings, 2003. http://www.ma.utexas.edu/users/voloch/preprint.html. 10. Berrizbeitia P. Sharpening Primes is in P for a large family of numbers. http://archiv.org/abs/math/NT/0211334, August 23, 2005. 11. Q.Cheng, Primality proving via one round in ECPP and one iteration in AKS. http://www.cs.ou.edu/~qcheng/pub.html.*