УДК 511.333

Рябухо О.М., Турка Т.В.

[1] *кандидат фізико-математичних наук, завідувач кафедри алгебри, ДВНЗ «ДДПУ»*

[2] *кандидат фізико-математичних наук, асистент кафедри алгебри, ДВНЗ «ДДПУ»*

e-mail: renelena@mail.ru, tvturka@mail.ru

ДОСЛІДЖЕННЯ ІМОВІРНІСНИХ АЛГОРИТМІВ ТЕСТУВАННЯ ПРОСТОТИ ЧИСЕЛ

Описані декілька класичних імовірнісних алгоритмів для визначення простоти числа. На прикладі тестів Ферма та Соловея-Штрассена досліджено, наскільки ефективними є такі тести. Розглянуте питання забезпечення необхідної точності визначення простоти при використанні таких тестів.

Ключові слова: *імовірнісний алгоритм, прості числа, псевдопрості числа, слабо псевдопрості числа, ефективність тесту.*

Вступ

Із появою у 1976 році ідеології відкритого ключа та з розвитком можливостей обчислювальної техніки в криптографії почали активно використовувати фундаментальні результати теорії чисел і сучасної алгебри. Так однією з головних обчислювальних задач сучасної криптографії є задача пошуку великих простих чисел.

Більшість сучасних асиметричних криптосистем так чи інакше використовують великі прості числа. Саме вони добре підходять для побудови односторонніх функцій. Нагадаємо, що односторонніми називають такі функції, які легко обчислюються для довільного вхідного значення, але аргумент при заданому значенні функції знайти важко. Такі задачі, як обчислення дискретного логарифму, або розклад на множники, можуть бути використані при побудові односторонніх функцій і традиційно вважаються складними.

Наведемо перелік криптосистем та алгоритмів, в яких використовуються прості числа:

- обмін ключами за Діффі-Хелманом;
- криптосистема RSA (шифрування та підпис);
- алгоритм цифрового підпису DSA;
- схема Ель-Гамаля;
- криптосистема Рабіна;
- криптографія на еліптичних кривих, зокрема ГОСТ Р 34.10-2001.

© Рябухо О.М., Турка Т.В., 2013

В них на перший план виходить порядок чисел, з якими доводиться працювати. Такі числа повинні бути достатньо великими, щоб забезпечувати криптоаналітичну стійкість використовуваного алгоритму. В той же час, їх потрібно генерувати порівняно швидко. Це обумовлює важливість побудови ефективних алгоритмів для перевірки великого випадкового числа на простоту (випадковість обраного числа теж є важливою для забезпечення стійкості). Всі такі алгоритми можна поділити на дві групи: детерміновані та імовірнісні. Перші встановлюють простоту числа математично строго, і, як правило, потребують багато часу для перевірки, в той час як другі просто мінімізують імовірність похибки у визначенні простоти після кожного послідовного свого запуску.

Постановка задачі.

Сформулюємо основні напрямки, у яких розвивається проблема пошуку великих простих чисел.

1. Для заданого великого натурального числа n з'ясувати, чи є воно простим. Можна послідовно перебирати прості числа менші за задане число (точніше менші за $\sqrt{n}$), щоб знайти всі його дільники. Тоді набуває актуальності не стільки час роботи алгоритму, як його асимптотична поведінка при зростанні кількості цифр числа n. Довгий час задача перевірки числа на простоту була яскравим прикладом задачі, яка ефективно розв'язується імовірнісними алгоритмами, а не детермінованими. Нарешті в 2002 році індійські математики Агравал, Кайал і Саксена [2] запропонували детермінований поліноміальний алгоритм.

2. Знайти велике просте число. Найбільші прості числа, знайдені на сьогодні, мають спеціальний вигляд. Для чисел спеціального вигляду розроблені алгоритми перевірки на простоту, які не застосовні до звичайних чисел. Такі числа не можна вважати випадковими, і вони майже не застосовуються в асиметричних криптосистемах. Наприклад, числа Мерсенна, це числа виду $M_p = 2^p - 1$. На початку лютого 2013 року математик Кертис Купер, участник проекту обчислень GIMPS (Great Internet Mersenne Prime Search), знайшов 48-е просте число Мерсенна. Десятковий запис такого числа має 17 425 170 знаків.

3. Знайти велике випадкове просте число. В цьому випадку ми тестуємо різні випадкові числа заданої складності (під складністю маємо на увазі кількість знаків в бінарному записі числа), аж поки не натрапимо на просте. Якщо наш алгоритм дає негативну відповідь, або просто занадто довго перевіряє вибране число — ми міняємо його на якесь інше — з надією на те, що

для нього алгоритм працюватиме ефективніше.

4. Знайти просте число, яке буде дільником заданого натурального числа, або, більш загально, розкласти задане натуральне число на множники. Ця задача найчастіше постає в криптоаналізі, і в деякому сенсі є узагальненням першої — адже якщо нам не вдалося знайти жодного власного дільника числа p — це і означає, що воно просте. Насправді, розкласти число на множники набагато складніше, ніж перевірити його на простоту — існування поліноміального алгоритму для розкладу числа на множники не доведене.

Зосередимося на третій проблемі, яка, по суті, є послідовним застосуванням першої. Формалізуємо задачу.

Дано натуральне число p. Встановити, чи буде воно простим, за допомогою деякого алгоритму. Якщо алгоритм безсилий, то міняємо число на інше, бо для нас важливіше знайти хоча б одне таке просте число. Найважливішим критерієм якості нашого алгоритму буде час його виконання. Цей час можна суттєво зменшити, якщо використовувати імовірнісні методи перевірки. Імовірнісні тести працюють набагато швидше детермінованих, але мають певний недолік. Після позитивного проходження числом тесту, залишається імовірність того, що воно насправді складене.

Вимоги до імовірнісного алгоритму перевірки простоти.

1. З самого початку ми вважаємо, що число просте, а за допомогою алгоритму спробуємо встановити, чи є воно складеним (встановити непростоту числа значно легше, ніж простоту).

2. Алгоритм встановлює, чи число просте, але може помилитись — тобто він на складене число може сказати, що воно просте.

3. Алгоритм залежить від параметра, який вибирається щоразу випадковим. Таким чином, ми можемо застосовувати алгоритм багато разів з різними значеннями параметра, і з кожним наступним застосуванням алгоритму, імовірність того, що досліджуване число є складеним, повинна зменшуватись.

4. Коли параметр пробігає всю наперед визначену параметричну множину, то наш імовірнісний алгоритм перетворюється на детермінований — тобто ми можемо стверджувати, що досліджуване число насправді просте.

Алгоритм часткового ділення.

Можна розглядати алгоритм часткового ділення як імовірнісний тест. Нехай нам треба перевірити простоту числа n. Ми перевіряємо, чи не ділиться воно на жодне з чисел $2, 3, ..., a$. В цьому випадку a виступає параметром, а параметричною множиною буде $2, 3, ..., [\sqrt{n}]$. Коли a пробігає всю множину параметрів, то тест стає детермінованим. Зрозуміло, навряд чи можна го-

ворити про ефективність цього тесту, якщо ми плануємо перевіряти за його допомогою дуже великі числа. Але перед перевіркою даного числа на простоту за допомогою малої теореми Ферма та інших методів доцільно провести базове дослідження на наявність малих простих дільників. Це дає можливість суттєво скоротити час пошуку простого числа, у випадку, коли нас цікавить знаходження хоча б одного такого числа (ця проблема постає в короткочасних процедурах шифрування, коли час кодування-декодування для нас відіграє більшу роль, ніж час, який потрібен для криптоаналізу).

Алгоритм базового методу може бути представлений наступними кроками:

1. Задаємо l — нижню межу діапазону, в якому повинне знаходитись просте число.

2. Задаємо u — верхню границю діапазону, в якому повинне знаходитись просте число.

3. Перевіряємо коректність задання діапазону $2 < l \leqslant u$.

4. Підраховуємо максимальну кількість спроб $r \leftarrow 100([\log_2 u] + 1)$.

5. $r \leftarrow r - 1$.

6. Якщо $r < 0$, то ліміт спроб вичерпано (подія має дуже малу ймовірність, і на практиці майже ніколи не зустрічається. В тому разі, коли це все ж сталося, треба просто перезапустити тест).

7. Вибираємо в заданому інтервалі $1 \leqslant n \leqslant u$ випадкове число n.

8. Якщо n менше за 2000, то тестування виконується методом пробного ділення на всі відомі прості числа, менші за n (такі числа називаються табульованими).

9. Якщо n більше за 2000, то тестування виконується методом пробного ділення на всі прості числа, менші за 2000. Якщо дільник існує, то переходимо до кроку 5.

10. На цьому кроці можна приступати до перевірки числа n на простоту іншим методом (який не пов'язаний з методом часткового ділення).

11. Якщо тест негативний (n виявилось складеним), переходимо до кроку 5.

12. Якщо тест позитивний, то ми згенерували просте число, що нам і було потрібно.

Ключовим в цьому тесті являється пункт 9. Він дозволяє відкинути 85% чисел перед запуском обраного нами алгоритму для дослідження простоти. Слід відмітити, що ефективність часткового ділення ніяк не залежить від основного алгоритму (який запускається на кроці 10). Очевидна доцільність застосування часткового ділення.

Псевдопрості числа.

Псевдопростим числом називають складене натуральне число, яке має деякі властивості простих чисел. Існування псевдопростих чисел перешкоджає роботі алгоритмів, які використовують ті чи інші властивості простих чисел.

Згідно малої теореми Ферма для довільного простого числа p та для довільного натурального числа n, взаємнопростого з p, має місце конгруенція:

$$a^{p-1} \equiv 1(\mathrm{mod}\ p). \tag{1}$$

На основі цієї теореми можна побудувати досить потужний тест на простоту.

Тест Ферма. Для $n > 1$ вибираємо $a > 1$, і обчислюємо $a^{n-1}(\mathrm{mod}\ n)$, якщо результат не 1, то n складене, якщо 1, то n — псевдопросте за основою a або псевдопросте число Ферма. Деякі складені числа є псевдопростими за будь-якою основою. Це так звані абсолютно псевдопрості числа, їх ще називають числами Кармайкла. Ситуація, коли досліджуване число виявиться абсолютно псевдопростим, дуже малоймовірна, але формально ми не можемо її опускати.

Ефективність тесту Ферма для слабо псевдопростих чисел.

Дослідимо, наскільки ефективним є тест Ферма для чисел, які не є числами Кармайкла. Такі числа будемо називати слабо псевдопростим.

Позначимо за W множину всіх значень параметра a, для яких n проходить тест на простоту за Ферма, тобто:

$$W = \{a \in N : a^{n-1} \equiv 1(\mathrm{mod}\ n)\}. \tag{2}$$

Оскільки, за нашим припущенням, n не являється абсолютно псевдопростим, то $|W| < |Z_n^*|$ — тобто W не може збігатись зі всією мультиплікативною групою Z_n^* лишків, взаємнопростих з n. Легко зрозуміти, що W-підгрупа групи Z_n^*, тому

$$\frac{|Z_n^*|}{|W|} \geqslant 2. \tag{3}$$

Це означає, що серед елементів параметричної множини нашого імовірнісного тесту максимум половина призводить до продовження прогонки (всі лишки з $Z_n^* \setminus W$ зразу покажуть, що n не є простим). Звідси негайно слідує, що після k прогонок тесту Ферма для слабо псевдопростого числа імовірність похибки в визначенні простоти числа n становитиме не більше 2^{-k} (звичайно ж, за умови випадкового вибору лишків з параметричної множини).

Таким чином, тест Ферма є ефективним для чисел, які не є абсолютно псевдопростими — ми можемо із як завгодно великою точністю встановлювати простоту таких чисел. Проблема лише в тому, що слід спершу відсіяти

числа Кармайкла, а це майже нереально для тих порядків чисел, які зараз застосовуються в криптографічних протоколах.

Тест Соловея-Штрассена

Розглянемо тест, який будується на узагальненні малої теореми Ферма. Використовуємо критерій Ейлера.

Твердження 1. *Для довільного непарного n наступні умови еквівалентні:*

1. *n — просте;*
2. *для довільного $a \in Z$ виконується конгруенція:*

$$a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) (\mathrm{mod} n). \tag{4}$$

Цей критерій дає нам змогу використати наступний імовірнісний тест.

1. Вибираємо U — коефіцієнт точності (чим більший цей коефіцієнт, тим вища точність встановлення простоти n).
2. $k \leftarrow 1$.
3. Вибираємо з впорядкованого по величині масиву простих чисел k — не просте число.
4. Перевіряємо, чи виконується конгруенція (4).
5. Якщо конгруенція не виконується, то відповідь «n — складене».
6. Якщо конгруенція виконується, то $k \leftarrow k + 1$.
7. Якщо $k < U$, то переходимо до третього кроку.
8. Якщо $k > U$, то відповідь: «n — псевдопросте з імовірністю $1 - 2^{-U}$».

Легко бачити, що цей тест задовольняє вимоги ефективного імовірнісного тесту на простоту: множиною параметрів виступають прості числа, менші за $\sqrt{n}$, і після кожного прогону тесту імовірність того, що тест неправильно визначив простоту числа, зменшується вдвічі. При цьому тест Соловея-Штрассена позбавлений головного недоліку тесту Ферма — для всіх натуральних n багатократна прогонка тесту зменшує імовірність похибки до нуля.

Слід також відмітити головний недолік тесту Соловея-Штрассена — це час його виконання. Якщо піднесення до степеня за модулем можна здійснити за порівняно малий час, то про обчислення символу Лежандра цього

сказати не можна. Проблема ускладнюється необхідністю проводити тест декілька разів — тим більше, чим меншу імовірність похибки ми вважаємо за прийнятну.

Спробуємо формалізувати задачу. Нехай для нас прийнятна імовірність похибки ϵ. Для її забезпечення нам достатньо k прогонок тесту, де $2^{-k} < \epsilon$. Ми можемо суттєво зменшити час, необхідний для забезпечення достатньої точності визначення простоти, якщо покращимо оцінку

$$\frac{|Z_n^*|}{|W|} \geqslant 2 \tag{5}$$

замінивши двійку на якесь більше число.

Бачимо, що надто велика кількість прогонок тесту Соловея-Штрассена, необхідна для досягнення прийнятної точності оцінювання простоти великих натуральних чисел, робить його практично не застосовним. Проте, тест допускає модифікації, які дозволяють оптимізувати кількість його послідовних застосувань для досягнення заданої точності. Подальша оптимізація тесту Соловея-Штрассена реалізована в імовірнісних тестах Леманна, Міллера-Рабіна. Ці тести детально розглянуті в [4].

Оцінки швидкості алгоритмів

Наведемо емпіричні оцінки швидкості алгоритмів (кількість операцій, необхідних для проведення одного циклу тесту), описаних в цій роботі. Такі оцінки найчастіше подаються у вигляді деякої функції від числа, що перевіряється, і містять в собі деякі константи, чисельне значення яких для нас неважливе — адже, має значення лише асимптотика цієї функції при збільшенні порядку досліджуваного числа. Доведення цих оцінок не приводиться, оскільки воно опирається на деякі нетривіальні факти теорії алгоритмів. Детально методи отримання оцінок такого роду описані в книзі [6].

1. Для тесту на основі часткового ділення кількість операцій для повної перевірки числа n на простоту не перевищує $C\sqrt{n}$ для деякої константи $C > 0$. Ця оцінка слідує з того, що для кожного часткового ділення потрібна лише скінченна кількість операцій (для нас потрібно, щоб вона не залежала від n). На практиці, вже для чисел порядку 1030 отримуємо таку кількість операцій, яку не здатний виконати за прийнятний час найпотужніший комп'ютер. Для імовірнісної модифікації тесту часткового ділення час залежить від потужності параметричної множини, яка забезпечує задану точність перевірки простоти. В розглянутому нами прикладі ця множина має потужність $O(\log n)$.

2. Для тестів Ферма та Соловея-Штрассена кількість операцій оцінюється числом $O(\log n)^3$, тобто є поліноміальною функцією від кількості знаків числа n. Необхідність запускати ці тести велику кількість разів збільшує цю оцінку до $O(\log n)^4$ — відповідно до потужності параметричної множини.

В той же час, найшвидші детерміновані тести показують значно гірші результати. Поліноміальний алгоритм, запропонований Агравалом, Кайялом та Саксеною, дає результат $O((\log n)^{\frac{15}{2}})$.

Висновки

Дослідження алгоритмічних проблем теорії чисел є актуальним з часу винайдення перших криптосистем з відкритим ключем. Причому важливим є як удосконалення самих криптографічних алгоритмів та методів генерації простих чисел, так і дослідження стійкості цих алгоритмів з точки зору криптоаналізу. Використання імовірнісних тестів дає нам суттєвий виграш в швидкості. Значення похибки при встановленні простоти числа, яке отримується вже після невеликої кількості прогонок тестів, може задовольнити навіть вибагливого криптографа.

Література

[1] *Alford W.R.* There are Infinitely Many Carmichael Numbers / W.R. Alford, A. Granville, C. Pomerance // Annals of Mathematics. — 1994. — № 139. — P. 703 – 722.

[2] *Agrawal M.* Primes is in P / M. Agrawal, N. Kayal, N. Saxena // Annals of Mathematics. — 2004. — № 160. — P. 781 – 793.

[3] *Василенко О.Н.* Современные способы проверки простоты чисел / О.Н. Василенко // Кибернетический сборник. — 1988. — № 25. — С. 162 – 187.

[4] *Василенко О.Н.* Теоретико-числовые алгоритмы в криптографии / О.Н. Василенко. — М.: МЦНМО, 2006. — 336 с.

[5] *Коблиц Н.* Курс теории чисел в криптографии / Н. Коблиц. — М.: Научное изд-во: ТВП, 2001. — 254 с.

[6] *Черемушкин А.В.* Лекции по арифметическим алгоритмам в криптографии / А.В. Черемушкин. — М.: МЦНМО, 2002. — 104 с.