НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Звіт з виконання комп'ютерного практикуму

ДОСЛІДЖЕННЯ МЕТОДІВ РЕАЛІЗАЦІЇ ПСЕВДОВИПАДКОВИХ ПОСЛІДОВНОСТЕЙ

Виконали студенти
групи ФІ-32мн
Величко Олена,
Мельник Ілля,
Міснік Аліна

Перевірили:
Селюх П.В.

Київ — 2023

Мета роботи: Дослідження методів реалізації псевдовипадкових послідовностей, ефективність алгоритмів тестування на простоту. Розробити власну концепцію бібліотеки генерації псевдовипадкової послідовності, тестування простоти чисел та генерації простих чисел

Постановка задачі: Розглянути та порівняти між собою алгоритми тестування на простоту: імовірнісні, гіпотетичні, детерміновані. Порівняти теоретичну та практичну похибки ймовірнісних алгоритмів тестування (Ферма, Соловея-Штрассена, Міллера-Рабіна). Розглянути алгоритми генерації випадкових чисел – Маурера та Чебишева.

1 ХІД РОБОТИ

1.1 Теоретичний опис

Як ми знаємо, асиметрична криптографія потребує особливої уваги до генерації ключів. Наприклад, у криптосистемі RSA дуже важливо правильно згенерувати два секретних параметри p та q, добуток яких стане майбутнім модулем. При генерації, ми повинні гарантувати рівномірність, незалежність та однорідність знаків (в основному це застосовується до бітових чи байтових послідовностей). З курсу асиметричної криптографії ми вже змогли зрозуміти, що найкраще для такої роботи підійдуть спеціалізовані алгоритми генерації псевдовипадкової послідовності (в нашому випадку Блюм-Блюм-Шуба). Вбудовані генератори мов програмування на жаль не можуть задовольнити нас рівномірністю знаків, алгоритми на принципі ЛРЗ мають обмежений період та великий час генерації. Після того, як ми згенерували послідовність потрібної нам довжини, постає питання про те, як перевірити чи буде воно простим?

В наш час існує ряд ефективних методів тестування великих чисел на простоту – імовірнісних, детермінованих, а також гіпотетичних (що спираються на деяку математичну гіпотезу). Звичайно, не усі вони є оптимальні за часом і складністю обрахунку, тож розберемося, які алгоритми кращі.

1.2 Імовірнісні алгоритми

На практиці в більшості випадків використовуються імовірнісні тести. Це пов'язано з тим, що вони не потребують точної перевірки на взаємну простоту з усіма необхідними простими числами до нього (принаймні менших за квадратний корінь числа), а використовують ймовірність того, що це число просте. Числа, що відповідно проходять ймовірнісні тести

називають псевдопростими. Це пов'язано з тим, що це число має деяку властивість простого числа, але не обов'язково є ним. Та це не є проблемою, оскільки за правильного підбору кількості запусків тесту (раундів) можна досягти точності у перевірці на простоту, оскільки тоді ймовірність помилки буде спадати. Тож поглянемо на часові оцінки відомих тестів:

1) Алгоритм Міллера-Рабіна — $O(k \cdot log^3 n)$
2) Алгоритм Соловея-Штрассена — $O(k \cdot log^3 n)$
3) Алгоритм Ферма — $O(k \cdot log^2 n \times loglogn \times logloglogn)$

1.3 Детерміновані алгоритми

Детерміновані алгоритми це ті, що з урахуванням конкретних вхідних даних, завжди будуть видавати ті ж самі вихідні дані, а основна машина завжди проходитиме через однакову послідовність станів. Тобто ми зможемо остаточно отримати відповідь для нашого числа, запустивши алгоритм і при цьому необхідно пройти визначену кількість ітерацій. І це є головним мінусом даного алгоритму. Одним із відомих нині детермінованих алгоритмів тестування чисел на простоту є AKS-тест або тест простоти Агравал-Кайал-Саксена або тест простоти трьох індусів). Оскільки даний алгоритм для нас був маловідомий, приведемо його формулювання:

Якщо
$$(x-a)^n \equiv (x^n - a)(mod n, x^r - 1)$$
для всіх $a < \sqrt{\phi(r)} logn$
то n — просте

Час роботи такого алгоритму на жаль не є оптимальним, як у імовірнісних алгоритмів і дорівнює : $O(log^{12} n \times loglogn)^c$. Для наведеної вище модифікації дорівнює: $O(log^6 n)$

Іншим цікавим алгоритмом детермінованого тестування є тест Міллера. Він базується на тому, що непарне складене число n або є степенем деякого простого числа, або існує просте число, яке належить

інтервалу від 2 до деякої функції $f(n)$, залежної від n. Час роботи такого алгоритму оцінюється у $O(n^{\frac{1}{7}})$.

1.4 Гіпотетичні алгоритми

Іноді алгоритми можна покращити. Для цього використовують гіпотези, які відповідно можуть спростити деякі кроки. Єдиною проблемою є те, що гіпотези це всього-на-всього припущення, яке не доведене. Воно може працювати швидко в нашому випадку, але правильність цих обчислень ставиться під сумнів. Приведемо декілька модифікованих алгоритмів з гіпотезами:

1) Тест простоти Агравал-Кайал-Саксена, як не дивно має гіпотетичні версії. Наприклад, згідно з гіпотезою Артіна, що існування та кількісна оцінка простих чисел, за модулем яких задане ціле число є первісним коренем, час алгоритму можна покращити до $O(log^6 n)$. На деякому проміжку чисел, доведено правильність даної гіпотези, але питання чи працює це для нескінченності залишається відкритим. Аналогічну оцінку дає і гіпотеза Софі Жермен про сильні прості числа.

2) Тест Міллера також можна покращити, використавши гіпотезу Рімана, про розподіл нулів дзета-функцій Рімана до $O(log^4 n)$. Проблема полягає лиш в тому, що дана гіпотеза досі недоведена.

1.5 Порівняння імовірнісних алгоритмів

Отже, як ми зрозуміли, наразі найкраще використовувати імовірнісні алгоритми тестування простоти, оскільки вони мають найкращі часові оцінки. Постає лише питання, який саме алгоритм краще обрати. Для цього звіримо помилки трьох відомих алгоритмів: Ферма, Соловея-Штрассена та Міллера-Рабіна. Для цього ми реалізували три алгоритми, як генератор псевдовипадкової послідовності було обрано генератор Блюм-Блюм-Шуба.

Для покращення результатів, використовується метод пробних ділень.

1) **Тест Ферма** ґрунтується на перевірці чи є відповідне число псевдопростим за деякою основою. За проходження тесту, таке число називають псевдопростим Ферма. Теоретична ймовірність правильного результату $p = 1 - \frac{1}{2^k}$, де k – кількість ітерацій.

2) **Тест Соловея-Штрассена** ґрунтується на перевірці чи відповідне число є псевдопростим Ойлера за деякою основою. Теоретична ймовірність правильного результату $p = 1 - \frac{1}{2^k}$, де k – кількість ітерацій.

3) **Тест Міллера-Рабіна** ґрунтується на перевірці чи відповідне число є сильно псевдопростим за деякою основою. Теоретична ймовірність правильного результату $p = 1 - \frac{1}{4^k}$, де k – кількість ітерацій.

	Теоретична	**Практична**
Ферма	$\frac{1}{2^k}$	0
Соловей-Штрассена	$\frac{1}{2^k}$	0
Міллера-Рабіна	$\frac{1}{4^k}$	0

Таблиця 1.1 – Теоретичні і практичні помилки тестів

	768 бітів			**1024 бітів**		
Ферма	274	449	2121	518	756	3209
Соловей-Штрассена	237	424	2100	452	701	3152
Міллера-Рабіна	244	406	2143	463	709	3212
	10 ітер.	100 ітер.	1000 ітер.	10 ітер.	100 ітер.	1000 ітер.

Таблиця 1.2 – Усереднений час роботи алгоритмів у мкс

1.6 Алгоритми генерації простих чисел

1.6.1 Алгоритм Маурера

Метод Маурера (1989 р.) є історично першим методом генерації доказово простого числа та заснований на теоремі Поклінґтона.

Теорема 1.1 (Теорема Поклінґтона)**.** *Нехай для цілого числа* $n \geqslant 3$ *такого, що* $n-1 = R \cdot F$*, відомий розклад числа* F *на множники* $F = \prod_{i=1}^{t} q_i^{e_i}$.

Якщо $\exists a$ *таке, що:*

1) $a^{n-1} \equiv 1 \mod n$,

2) $\gcd(a^{\frac{n-1}{q^i}} - 1, n) = 1$ *для кожного* i, $1 \leqslant i \leqslant t$, *тоді кожен простий дільник* p *числа* n *конгруентний* 1 *за модулем* F. *З цього випливає, що якщо* $F > \sqrt{n} - 1$, *то* n – *просте.*

Опишемо алгоритм Маурера, побудований на цій теоремі.

Вхід алгоритму: кількість бітів t.

Вихід алгоритму: просте число p, $|p| = t$ біт.

1) обирається будь-яке просте число p_s довжини t_s бітів.

2) $t_0 = t$, t_{i+1} обирається випадково з діапазону $[\frac{t_i}{2}] + 1 \leqslant t_{i+1} \leqslant t_i - 20$, якщо $t_i > 40$, інакше $[\frac{t_i}{2} + 1]$.

3) обчислюється $p_i = p_{i+1} \cdot R + 1$, обирається випадкове парне число R, яке забезпечить потрібну довжину p_i.

4) для $n = p_i$ та випадкового a перевіряється теорема Поклінґтона, якщо умови виконуються, то p_i – просте. Кроки 1)-4) повторюються, доки не буде побудовано ланцюг простих чисел $p_s < p_{s-1} < ... < p_0 = p$.

Оскільки алгоритм не займається пошуком основ a, що задовольняють умовам теореми Поклінгтона, а лише перевіряє ці умови для однієї випадкової основи, то якісь прості числа можуть бути ним не розпізнані, і перебір буде продовжено, поки не буде побудовано число, простоту якого вдасться довести.

1.6.2 Послідовність Чебишева

Для генерації простих чисел може також використовуватися метод, який базується на теоремі Чебишева. Ця теорема визначає скільки простих чисел знаходиться в певному діапазоні. Наведемо її формулювання.

Теорема 1.2 (Теорема Чебишева про розподіл простих чисел)**.** *Нехай* $\pi(x)$ — *кількість простих чисел, які не перевищують* x. *Тоді для будь-якого*

числа $x > 1$ існують такі константи $a, b > 0$, що виконується нерівність:

$$\frac{ax}{\ln x} \leqslant \pi(x) \leqslant \frac{bx}{\ln x}$$

Теорема Чебишева показує, що частка позитивних цілих чисел, які менші деякого цілого m і є простими, близька до $\frac{1}{\ln m}$. Таким чином, якщо ми виберемо випадково велике ціле позитивне непарне число x і будемо послідовно перевіряти на простоту числа $x, x+1, x+2, ...$, то в середньому ми вперше зустрінемо просте число на кроці з номером $\ln x$.

Алгоритм генерації простих чисел використовує цю теорему для побудови послідовності чисел, що потребують тестування на простоту.

Основна ідея полягає в тому, що можна вибрати такі значення параметрів у формулі послідовності, щоб забезпечити, що кожне число в послідовності буде мати простий множник у визначеному діапазоні.

Часто в контексті алгоритму знаходження простих чисел використовують формулу послідовності $n^2 - n + a$, де n — це порядковий номер числа в послідовності, а a — сталий параметр, значення якого можна вибрати так, щоб кожне число з послідовності було простим для великої кількості n. Ця формула ефективна, оскільки задовільняє умові рівномірності розподілу простих чисел у послідовності та забезпечує ефективне використання тестів на простоту.

В загальному випадку алгоритм вигдядає так:

1) Визначаємо границі діапазону, в якому будемо шукати прості числа, а також необхідну кількість простих чисел.

2) Використовуючи теорему Чебишева, будуємо послідовність чисел, яку будемо перевіряти на простоту. Числа знаходимо, наприклад, за формулою $n^2 - n + a$.

3) Перевіряємо кожне число з побудованої послідовності на простоту. Це можна зробити використовуючи різні алгоритми тестування на простоту, такі як тест Міллера-Рабіна або тест Соловея-Штрассена.

4) При знаходженні необхідної кількості простих чисел зупиняємо

генерацію.

Варто зауважити, що теорема Чебишева не вказує конкретну формулу для побудови послідовності, але надає теоретичну підтримку для вибору параметрів так, щоб забезпечити наявність простих чисел у визначеному діапазоні. Тобто наведена формула не є обов'язково єдиною чи найкращою і залежить від конкретної реалізації алгоритму та його властивостей.

ВИСНОВКИ

У даній роботі ми провели аналіз реалізації псевдовипадкових послідовностей, ефективність алгоритмів тестування на простоту та генерацію простих чисел.

Проаналізувавши усі види алгоритмів тестування, можемо зробити висновок, що найкраще на сьогодні варто використовувати імовірнісні алгоритми. Це пояснюється тим, що імовірнісні алгоритми досить швидкі і оптимізовані, особливо це актуально для малих систем, як токени, картки і смартфони. На відміну від детермінованих алгоритмів, ми можемо зафіксувати бажану нам ймовірність успіху і не перебирати основи за усіма необхідними числами з умови. Також, що не мало важливо, імовірнісні алгоритми мають більш передбачувані оцінки помилки, на відміну від гіпотетичних, оскільки ми точно не можемо бути впевнені у правильності гіпотези.

Аналізуючи імовірнісні алгоритми між собою, варто зазначити, що в нашому випадку вони усі показали безпомилковий результат. Можливо, на це вплинуло попереднє пробне ділення, а також особливості генерації з допомогою алгоритму Блюм-Блюм-Шуба. Аналізуючи їх швидкість, можна сказати, що алгоритми Ферма та Міллера-Рабіна найгірше працюють на великих ітераціях. А от алгоритм Соловея-Штрассена навпаки, починає вигравати за швидкістю. При цьому варто помітити, що даний алгоритм може бути швидшим за свого конкурента Міллера-Рабіна за малих ітерацій, але на великих числах. При цьому, Міллера-Рабіна найкраще себе показав у роботі з невеликою кількістю операцій і меншою кількістю бітів. Що робить його найкращим для використання у роботі з ними. Основною перевагою також є те, що ймовірність його похибки є набагато меншою, а отже гарантує цілісність і коректність роботи.

Метод Маурера генерує доказово просте число, використовуючи перевірку теореми Поклінґтона. Його перевага полягає в тому, що число,

отримане в результаті роботи алгоритму – буде стовідсотково простим, але його основним недоліком є те, що він може відкинути просте число, якщо випадкова основа a буде невдалою.

Чебишев знайшов оцінку для визначення кількості простих чисел у заданому діапазоні і довів їх існування в інтервалі (n, Cn), де $n > 1, C > 1$. За допомогою його теореми з'явилася можливість будувати послідовності так, щоб кожне число з послідовності було простим для великої кількості n. Це збільшило ефективність алгоритмів знаходження простих чисел і дало змогу правильного вибору параметрів для забезпечення наявності цих чисел у визначеному діапазоні.

Аналізуючи тести, нам вдалося відтворити бібліотеку з генерацією ПВЧ та тестами простоти. У якості алгоритму ПВЧ використовується відомий алгоритм BBS (Блюм-Блюма-Шуба), що є криптографічно стійким і надійним. В якості тесту на простоту, ми радимо використовувати алгоритм Міллера-Рабіна, оскільки ймовірність помилки лише для трьох його ітерацій становить $p = 0,0156$, що є доволі сильним результатом.