МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра кібербезпеки

КОНДРАТЮК Владислав Миколайович

Алгоритм перевірки числа на простоту на основі BPSW тесту / Number simplicity checking algorithm based on BPSW test

спеціальність: 125 – Кібербезпека

освітньо-професійна програма –Кібербезпека

Кваліфікаційна робота

Виконав студент групи КБм -21
В.М. Кондратюк

Науковий керівник
к.т.н., доцент Н.Г.Яцків

Кваліфікаційну роботу допущено
до захисту:

«____» _____ ___2022 р.

Завідувач кафедри

_____ В.В.Яцків

ТЕРНОПІЛЬ - 2022

Факультет комп'ютерних інформаційних технологій
Кафедра кібербезпеки
Освітній ступінь «магістр»
спеціальність: 125 - Кібербезпека
освітньо-професійна програма –Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ В.В.Яцків
____" _____ 2021 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Кондратюку Владиславу Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи:
Алгоритм перевірки числа на простоту на основі BPSW тесту / Number simplicity checking algorithm based on BPSW test
керівник роботи к.т.н., доцент Н.Г. Яцків

затверджені наказом по університету від 31 грудня 2021 року № 606

2. Строк подання студентом закінченої випускної кваліфікаційної роботи 16 листопада 2022 року.

3. Вихідні дані до кваліфікаційної роботи: завдання на випускну кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:
– провести аналіз алгоритмів пошуку простих чисел;
– дослідити властивості псевдопростих чисел;
– дослідити математичні властивості символів Якобі;
– провести дослідження алгоритмів Соловея-Штрасена, Міллера-Рабіна та послідовностей Люка;
– дослідити етапи методу Фробеніуса;
– дослідити математичні основи BPSW тесту простоти;
- виконати реалізацію запропонованих рішень.

5. Перелік графічного матеріалу у роботі.
Алгоритм визначення простоти числа на основі BPSW тесту.
Екранні форми розробленого програмного засобу.
Експериментальні дані тестування алгоритму.

6. Консультанти розділів кваліфікаційної роботи

	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 11 жовтня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строки виконання етапів кваліфікаційної роботи	Примітка
1	Аналіз предметної області	12.2021 р. – 03.2022 р.	
2	Математичні основи BPSW тесту простоти	03.2022 р. – 05.2022 р.	
3	Реалізація запропонованих рішень	05.2022 р. – 11.2022 р.	

Студент _____ Кондратюк В.М.
(підпис)

Керівник роботи _____ к.т.н., доцент Н.Г. Яцків
(підпис)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Алгоритм перевірки числа на простоту на основі BPSW тесту» на здобуття освітнього ступеня «Магістр» зі спеціальності 125 «Кібербезпека» освітньо-професійної програми «Кібербезпека» написана обсягом 89 сторінки і містить 16 ілюстрації, 11 таблиць, 2 додатки та 49 джерело за переліком посилань.

Метою кваліфікаційної роботи є розробка алгоритму та програмного засобу визначення простоти числа на основі BPSW тесту простоти.

Методи досліджень базуються на теорії чисел, комбінаториці та чисельних методах.

Результати дослідження: досліджено математичні основи існуючих тестів простоти, властивості простих та псевдопростих чисел. Проведено дослідження властивостей символів Якобі та алгоритму їх обчислення в тестах простоти. Проаналізовано математичні основи BPSW тесту простоти та його елементів.

Удосконалено алгоритм перевірки простоти на основі алгоритму BPSW тесту та реалізовано програмний засіб для тестування чисел на простоту мовою C++.

Результати роботи можуть успішно застосовуватися в криптографічних перетвореннях та протоколах для систем захисту інформації.

Ключові слова: BPSW, ПРОСТІ ЧИСЛА, АЛГОРИТМ ЛУКАСА, АЛГОРИТМ СЕЛФРІДЖА, ТЕСТ СОЛОВЕЯ-ШТРАССЕНА, ТЕСТ МІЛЛЕРА-РАБІНА, МЕТОД ФРОБЕНІУСА, ПОСЛІДОВНОСТІ ЛЮКА.

ABSTRACT

The qualifying work on the topic "Algorithm for checking the number for simplicity based on the BPSW test" for obtaining the Master's degree in the specialty 125 "Cybersecurity" of the educational and professional program "Cybersecurity" is written in the volume of 89 pages and contains 16 illustrations, 11 tables, 2 appendices and 49 sources according to the list of references.

The purpose of the qualification work is to develop an algorithm and a software tool for determining the simplicity of a number based on the BPSW simplicity test.

Research methods are based on number theory, combinatorics and numerical methods.

Research results: mathematical foundations of existing simplicity tests, properties of prime and pseudoprime numbers were investigated. The properties of Jacobi symbols and the algorithm for their calculation in simplicity tests were studied. The mathematical foundations of the BPSW simplicity test and its elements are analyzed.

The simplicity checking algorithm based on the BPSW test algorithm has been improved and a software tool for testing numbers for simplicity in C++ language has been implemented.

The results of the work can be successfully applied in cryptographic transformations and protocols for information protection systems.

Keywords: BPSW, SIMPLE NUMBERS, LUCAS ALGORITHM, SELFRIDGE ALGORITHM, SOLOWAY-STRASSEN TEST, MILLER-RABIN TEST, FROBENIUS METHOD, LUKE SEQUENCES.

ЗМІСТ

ВСТУП

Актуальність роботи. Роль простих чисел в криптографії важко переоцінити. Їх застосовують майже у всіх крипто перетвореннях. Із ростом вимог до криптостійкості шифрів та криптографічних протоколів зростає необхідність використання наборів простих чисел великої розрядності. Часто перевірка чисел на простоту становить основу обчислювальної складності алгоритмів. Тому важливим завданням залишається розробка ефективних методів перевірки простоти багаторозрядних чисел.

Мета і завдання дослідження. Мета роботи полягає в розробці удосконаленого тесту простоти на основі BPSW тесту:

Для досягнення даної мети ставились наступні завдання:

- дослідити властивості простих та псевдопростих чисел;
- дослідити процес обчислення символів Якобі;
- проаналізувати математичні основи ймовірнісних тестів простоти;
- дослідити кроки алгоритму BPSW;
- розробити вдосконалений тест простоти на основі BPSW алгоритму;
- розробити та провести тестування програмного засобу на основі запропонованого алгоритму .

Об’єкт дослідження - процеси програмного опрацювання чисел для криптографічних перетворень.

Предмет досліджень – ймовірнісні тести простоти.

Методи досліджень базуються на теорії чисел, комбінаториці та чисельних методах.

Наукова новизна одержаних результатів визначається наступним чином:

- Удосконалено тест простоти на основі алгоритму BPSW, що дало можливість знизити обчислювальну складність алгоритму та побудувати програмний засіб для перевірки чисел на простоту.

Практична цінність одержаних результатів полягає в тому, що:

– обґрунтовано підхід та розроблено алгоритм перевірки числа на простоту за допомогою вдосконаленого BPSW тесту ;

– реалізовано програмне забезпечення для тестування числа на простоту на мові C++.

Публікації та апробація до магістерської роботи.

1. Ігнатєв І.В., Кондратюк В.М., Алгоритм перевірки числа на простоту. Збірник матеріалів проблемно-наукової міжгалузевої конференції «Автоматизація та комп'ютерно – інтегровані технології» (АКІТ -2022), Тернопіль, 2022. 70-73 с.

2. Посвятовська О.Б., Стефурак Н.А., Кондратюк В.М., Дослідження властивостей простих чисел для BPSW. Збірник матеріалів науково-практичної конференції молодих вчених, аспірантів та студентів» (КБКІТ-2020), Тернопіль, 2022, С. 73-79.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Прості числа в криптографії

Проведемо аналіз властивостей простих чисел та використання їх у кодуванні. Проект Great Internet Mersenne Prime Search, перед яким стоїть завдання пошуку великої кількості простих чисел особливо рідкісного виду, нещодавно відкрив найбільше просте число, відоме на сьогоднішній день. У цьому числі 23249425 цифр — достатньо, щоб заповнити книгу з 9000 сторінок. Для порівняння: кількість атомів у всьому Всесвіті, що спостерігається, оцінюється в число не більше ніж сотнею десяткових знаків. Нове число, яке записується як $2^{77232917}-1$ (два в 77 232917 степені відняти один), було виявлено інтузіастом, який присвятив 14 років обчислювального часу цьому пошуку. На сьогоднішній день великі прості числа повсюдно використовуються в нашому повсякденному житті, наприклад, у кредитних картах та персональних комп'ютерах, тому постійно існує потреба у нових простих числах (що більше, тим краще) для генерації секретних кодів.

В сучасному світі наука криптографія має велике значення, оскільки теперішній час – це час інформаційних воєн. Адже, як сказав Натан Ротшильд: «Хто володіє інформацією – той володіє світом». На сьогоднішній день без шифрування інформації не обходиться жодна держава, організація чи підприємство, а засоби криптографічного захисту застосовуються повсюдно, у тому числі й для захисту комерційної інформації: електронна пошта, банківські операції, кредитні картки та мобільний телефонний зв'язок – все це захищено секретними кодами, що безпосередньо засновані на властивостях простих чисел. Тому актуально досліджувати особливості простих чисел та способи застосування їх у шифруванні, щоб не тільки розвивати технології, від яких ми залежимо, а й зберігати їхню безпеку.

Найпростіші числа є основою сучасної криптографії. Прості числа є одним із найцікавіших математичних явищ, яке привертає до себе увагу вчених і простих громадян протягом уже понад двох тисячоліть. Незважаючи на те, що зараз ми живемо у вік комп'ютерів та найсучасніших інформаційних програм, багато загадок простих чисел не вирішено досі [1-2].

Прості числа - це, ті натуральні числа, які діляться без залишку тільки на одиницю і саму себе. Прості числа називають «цеглою» у будівлі математики, «атомами» математики та «генетичним кодом» числа, тому що будь-яке складене число може бути представлене у вигляді єдиного можливого добутку простих чисел. Ця теорема була сформульована Евклідом і відома як "основна теорема арифметики". Отже, прості числа є основними елементами, з яких побудовані всі числа. Прості числа утворюють складені числа. По-перше, число 1 має лише один дільник: саме це число[3]. Тому його не відносять ні до складових чисел, ні до простих. По-друге, єдиним парним числом, що стосується простих чисел, є двійка. Будь-яке інше парне число сюди потрапити просто не може, оскільки крім себе та одиниці ділиться ще й на два.

Виділення простих чисел є складним завданням математики. Вчені протягом багатьох століть намагаються знайти формулу, яка б з безлічі натуральних чисел виділити прості. Перший, хто займався цим завданням, був великий математик давнини Ератосфен, який жив майже 2300 років тому [4].

Він запропонував такий спосіб: записав усі числа від одиниці до якогось числа, а потім викреслив одиницю, яка не є ні простим, ні складовим числом, потім викреслював через одне усі числа, що йдуть після 2 (числа, кратні двом, тобто $4,6,8$ і т.д.). Першим числом, що залишилося після 2 було 3. Далі викреслювалися через два всі числа, що йдуть після трьох (числа, кратні 3, тобто $6,9,12,$ і т.д.), зрештою залишалися невикресленими тільки прості числа: $2,3,5,7,11,13,...$ Оскільки в часи Ератосфена писали на воскових дощечках, а замість того, щоб числа викреслювати, дощечку в потрібному місці проколювали, то спосіб отримав назву "решета Ератосфена".

У наведеній таблиці 1.1, отриманої з допомогою решета Ератосфена, можна побачити прості числа з першої тисячі натуральних чисел.

З першого погляду видно, що прості числа непередбачувані. Наприклад, між 1 і 100 простих чисел більше, ніж між 101 і 200. Усього в першій тисячі 168 простих чисел. Можна припустити, що якщо продовжити нашу таблицю, то з кожною тисячею кількість простих чисел збільшуватиметься. Але це не так. Вже відомо, що, наприклад, серед тисяч чисел між 10100 і 10100 + 1000 знаходиться лише два простих числа. І ці числа складаються більш як зі ста цифр.

Таблиця 1.1 - Прості числа з першої тисячі натуральних чисел

2	3	5	7	11	13	17	19	23	29	31	37	41	43	47	53	59	61	67	71
73	79	83	89	97	101	103	107	109	113	127	131	137	139	149	151	157	163	167	173
179	181	191	193	197	199	211	223	227	229	233	239	241	251	257	263	269	271	277	281
283	293	307	311	313	317	331	337	347	349	353	359	367	373	379	383	389	397	401	409
419	421	431	433	439	443	449	457	461	463	467	479	487	491	499	503	509	521	523	541
547	557	563	569	571	577	587	593	599	601	607	613	617	619	631	641	643	647	653	659
661	673	677	683	691	701	709	719	727	733	739	743	751	757	761	769	773	787	797	809
811	821	823	827	829	839	853	857	859	863	877	881	883	887	907	911	919	929	937	941
947	953	967	971	977	983	991	997												

Теорема Евкліда: «Перших простих чисел існує більше від будь-якого зазначеного їх числа».

Звичайно, спосіб Ератосфена не зміг задовольнити вчених і вони намагалися знайти формулу простих чисел. Упродовж багатьох століть це зробити не вдавалося. У ряді простих чисел було знайдено багато закономірностей, але поставлене завдання залишалося без відповіді. Завданням пошуку максимально великих простих чисел свого часу займались Катальді, Декарт, Ферма, Мерсен, Лейбніц, Ейлер та багато інших математиків. У математиці широко відомий термін прості числа Мерсен. Прості числа Мерсен є простими числами спеціального виду: $M_p = 2^p - 1$ де p - інше просте число.

До 1750 було знайдено всього 8 простих чисел Мерсенна: M2, M3, M5, M_7, M_{13}, M_{17}, M_{19}, M_{31}. Те, що M_{31} - просте число, довів у 1750 Л. Ейлер. У 1876 році французький математик Едуард Люка встановив, що число $M_{127} = 170141183460469231731687303715884105727$ - просте. У 1883 р.

І.М.Первушин довів, що $M_{61} = 2305843009213693951$ є простим. Пізніше було встановлено, що числа M_{89} та M_{107} прості. Перші 12 простих чисел Мерсена були обчислені лише олівцем на папері, а для обчислення наступних вже використовувалися механічні настільні рахункові машини.

Масштабний проект з пошуку простих чисел GIMPS був запущений у 1997 році, і нині вважається найтривалішим безперервним процесом розподілених обчислень в історії людства: він продовжується вже майже 20 років. Зараз у пікові моменти у GIMPS бере участь 360.000 процесорів із сумарною продуктивністю 150 трлн операцій на секунду. За час роботи GIMPS учасники цього проекту виявили 14 простих чисел Мерсенна. Останнє з них 274 207 281-1 було виявлено 07 січня 2016 року. Всього зараз відомо 49 простих чисел Мерсена. У списку найбільших простих чисел, відомих на сьогоднішній день, десять перших місць займають числа Мерсенна.

Пошук великих простих чисел - досить складне завдання, тому що ще нікому не вдалося знайти точну формулу чи алгоритм, що дозволяє генерувати будь-які прості числа заданого розміру.

У 1975 р. Уітфілду Діффі і Мартіну Хеллману, які тоді працювали в Стенфордському університеті, спала на думку ідея асиметричного шифрування, або «шифрування з відкритим ключем». Ця система заснована на спеціальних математичних функціях, званих «односторонніми функціями з потаємним входом», які дозволяють зашифровувати текст, але роблять розшифрування практично неможливим завдання без знання коду, що використовується. Ідея полягає в тому, що кожен користувач має кілька ключів: відкритий і закритий. Якщо ми хочемо надіслати комусь повідомлення, ми зашифровуємо це повідомлення за допомогою відкритого ключа, тобто ключа, відомого всім. Але лише певна людина, яка має відповідний закритий ключ, може розшифрувати це повідомлення. Однією з переваг такого методу є те, що закритий ключ ніколи не передається і тому його не потрібно постійно змінювати з метою безпеки.

Простих чисел досить багато та їх розподіл аналітично не доведений. У таблиці 1.2 показана можливість (у відсотках) виявитися простим для k-значних чисел.

Таблиця 1.2 – Ймовірність простоти числа

Розрядність:	2	3	4	5	6	7	8	9	10	12	15	20
Імовірність (%):	20	14	10.8	8.67	7.23	6.20	5.43	4.83	4.34	3.62	2.90	2.17

Для порівняння, частка чисел, які не мають дільників менших 10^k (у відсотках) приведена у таблиці 1.3.

Таблиця 1.3 – Частка чисел, які не мають дільників менших 10^k

k:	1	2	3	4	5
Частка (%):	22.86	12.03	8.10	6.09	4.88

Таким чином, якщо брати числа близько кількох мільярдів, які не мають дільників менше 1000, то приблизно половина таких чисел буде простою. Серед 18-значних чисел, що не мають дільників, менше 1000 більше третини прості.

Важливим завданням залишається розробка ефективних методів перевірки простоти. Навіть найпростіший їх – метод послідовного розподілу попри всі прості числа непоганий. Для перевірки простоти числа n достатньо перевірити подільність на всі прості числа, менші за квадратний корінь з n. Оскільки простих чисел, менших 2^{32} всього 203 280 221 числа, то для перевірки простоти числа $< 2^{64}$ потрібно трохи більше 200 млн. поділів, тобто менше секунди часу.

У сучасній критографії мають практичне значення прості числа величиною до 2^{1024} (~308 десяткових цифр), максимально до ^{23000} (~900 десяткових цифр). Таким чином, із практичної точки зору нам треба вміти перевіряти простоту чисел завдовжки до 1000 десяткових знаків. Для цього потрібні ефективні методи.

Кількість простих чисел, що не перевищують x в теорії чисел, прийнято позначати $\pi(x)$. Наведемо деякі значення цієї функції у таблиці 1.4.

Таблиця 1.4 – Значення функції $\pi(x)$

$\pi(10^3)$	$\pi(10^6)$	$\pi(10^9)$	$\pi(2^{32})$	$\pi(10^{12})$	$\pi(10^{15})$	$\pi(10^{18})$	$\pi(2^{64})$
168	78 498	50 847 534	203 280 221	37 607 912 018	29 844 570 422 669	24 739 954 287 740 860	425 656 284 035 217 743

Розроблена множина найрізноманітніших алгоритмів таких перевірок та результати експериментів приведені в таблиці 1.4.

1.2 Псевдопрості числа

Більшість із них дають лише імовірнісну відповідь: число може виявитися гарантованим складеним, або «ймовірно простим». Тобто кожен із цих методів може прийняти деяке складове число за просте, але не навпаки. Такі числа називаються "псевдопростими". Найбільш простий метод заснований на малій теоремі Ферма.

Якщо p - просте і a не ділиться на p, то $a^{p-1} \equiv 1 \bmod p$. Зворотне твердження неправильне, тобто з того, що $ap^{-1} \equiv 1 \bmod p$ не випливає, що p – просте, наприклад $2^{11 \cdot 31-1} \equiv 1 \bmod 11 \cdot 31$. Тим не менш, якщо для деякого a виконано $a^{n-1} \equiv 1 \bmod n$ то число n майже напевно просте.

Складне число n називається псевдопростим на підставі a, якщо $a^{n-1} \equiv 1 \bmod n$.

Іншими словами, пседопрості числа - це ті, на яких запропонований спосіб перевірки помиляється.

Псевдопростих чисел досить багато. Наприклад, серед чисел менших 2^{32} псевдопростих за основою 2 виявляється 10 403 (на 200 млн. простих), серед чисел менших 2^{64} -118968378 (на $4 \cdot 10^{17}$ простих).

Перевірка з кількох різних підстав скорочує кількість помилок, але з радикально. Наприклад, серед 10403 чисел менших 2^{32} псевдопростих на підставі 2, псевдопростих і на підставі 3 виявляється 2318 штук.

1.3 Кратні множники псевдопростих чисел

Цікаво відзначити, що псевдопрості числа можуть мати кратні множники. Однак такі числа повинні мати дуже спеціальний вигляд.

Нехай $n = p^k q$ - псевдопросте на підставі a, де p - просте. Тоді $p^2, \ldots, p^k$ - псевдопрости на підставі a. Більше того, у цьому випадку $a^{p-1} \equiv 1 \bmod p^k$.

Прості числа p такі, що p^2 псевдопросто на підставі a зустрічаються, але рідко. Наприклад, відомо лише два таких числа при $a = 2$. Це 1093 та 3511. Інших таких чисел немає, принаймні при $p < 758 \cdot 109$. Є деякі підстави припускати, що таких чисел більше немає взагалі, але довести це не вдається.

Можна знайти такі пари (a, p), що $a^{p-1} \equiv 1 \bmod p^2$ для $a = 2, \ldots, 127$ і $p < 10^{10}$. Їх виявилося 212. Частину цих чисел приведемо в таблиці 1.5.

Таблиця 1.5 – Таблиця псевдопростих чисел

a	p	a	p	a	p	a	p
2	1093	25	53471161	62	1291	93	81551
2	3511	25	1645333507	63	36713	94	241
3	11	25	6692367337	63	401771	94	32143
3	1006003	26	71	64	1093	94	463033
4	1093	26	486999673	64	3511	95	2137
4	3511	26	6695256707	65	163	95	15061
5	20771	27	1006003	66	89351671	96	109

продовження таблиці 1.5

5	40487	30	160541	67	268573	96	5437
5	53471161	31	79	68	113	96	8329
5	1645333507	31	6451	68	2741	96	12925267
5	6692367337	31	2806861	69	223	97	2914393
6	66161	32	1093	69	631	98	28627
6	534851	32	3511	69	2503037	98	61001527
6	3152573	33	233	70	142963	100	487
7	491531	33	47441	71	331	100	56598313
8	1093	33	9639595369	75	347	101	1050139
8	3511	35	1613	75	31247	102	7559
…	…	…	…	…	…	…	…

Можна перевірити, що немає інших пар виду $(2,p)$ при $p < 758\cdot10^9$, виду $(3,p)$ при $p < 681\cdot10^9$ і виду $(5,p)$ при $p < 40\cdot10^9$.

Найменше число a, при якому немає зазначених пар (a,p) при $p < 5\cdot10^{10}$ – це 21. Звичайно, навряд чи варто очікувати, що таких пар немає ні для яких p, швидше за все, це лише питання обсягу обчислень . Тим не менш, немає і жодних підстав вважати, що такі пари існують для всіх чисел.

З іншого боку, кожному за простого p існує досить багато підходящих a, щоправда вони поступово розташовані великому відрізку $0\ldots p^2$.

1.4 Символи Якобі та Лежандра

При розгляді тестів простоти вже не обійтися без поняття квадратичного лишку. Тому розглянемо властивості залишків квадратів.

Будь-яке дійсне позитивне число є квадратом деякого іншого числа (і навіть двох): 25 є квадратом 5 і -5, число 7 є квадратом sqrt(7) і -sqrt(7). Негативні числа не є квадратами, нуль є квадратом лише себе.

Для відрахувань за простим модулем ситуація цілком аналогічна. Нехай p – просте число, наприклад візьмемо $p = 11$. Розглянемо квадрати всіх ненульових відрахувань по модулю p, як це показано таблиці 1.6.

Таблиця 1.6 – Квадратичні відрахування та невідрахування

x :	1	2	3	4	5	6	7	8	9	10
x^2 :	1	4	9	5	3	3	5	9	4	1

При цьому, звичайно, $(-x)^2 = x^2$. Видно, що (при p більшому 2) половина ненульових відрахувань є чиїмось квадратом, інша половина – ні. Вони називаються квадратичними відрахуваннями та невирахуваннями відповідно. Іншими словами, з квадратичних відрахувань можна витягти корінь, і невирахувань - не можна. При $p = 11$ відрахуваннями будуть {1,3,4,5,9} та невирахуваннями {2,6,7,8,10}.

Для простого p і цілого a визначимо символ Лежандра в такий спосіб:

$$\left(\frac{a}{p}\right) = \begin{cases} 0\ ,\ a \equiv 0 \bmod p \\ 1\ , a - \textit{квадратний залишок по модулю } p \\ -1\ , a - \textit{к, дратне невідрахування по модулю } p \end{cases}$$

Проведемо дослідження властивостей залишків квадратів.

$$\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right).$$

Поширимо символ Лежандра у разі складеного p. Отриману функцію називатимемо символом Якобі і позначатимемо точно також. Нехай n непарне і має наступний розклад на прості множники:

$$n = p_1^{k1} ... p_s^{k_s}$$

Тоді для будь-якого цілого числа a символом Якобі назвемо число:

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{k_1} ... \left(\frac{a}{p_s}\right)^{k_s}$$

Квадратичний закон взаємності можна представити в такому вигляді. Нехай p,q - непарні числа. Тоді:

$$\left(\frac{p}{q}\right) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}} \left(\frac{q}{p}\right).$$

Приватні випадки:

$$\left(\frac{-1}{p}\right) = \begin{cases} 1, & p \equiv 1 \bmod 4 \\ -1, & p \equiv 3 \bmod 4 \end{cases},$$

$$\left(\frac{2}{p}\right) = \begin{cases} 1, & p \equiv \pm 1 \bmod 8 \\ -1, & p \equiv \pm 3 \bmod 8 \end{cases},$$

$$\left(\frac{3}{p}\right) = \begin{cases} 1, & p \equiv \pm 1 \bmod 12 \\ -1, & p \equiv \pm 5 \bmod 12 \end{cases},$$

$$\left(\frac{5}{p}\right) = \left(\frac{p}{5}\right) = \begin{cases} 1, & p \equiv \pm 1 \bmod 5 \\ -1, & p \equiv \pm 2 \bmod 5 \end{cases},$$

$$\left(\frac{-7}{p}\right) = \begin{cases} 1, & p \equiv 1,2,4 \bmod 7 \\ -1, & p \equiv 3,5,6 \bmod 7 \end{cases}.$$

Символи якобі часто використовуються в тестах простоти та криптографічних перетвореннях для систем захисту інформації.

1.5 Тест Соловея-Штрассена

"Тест Соловея - Штрассена" є посиленням методу Ферма з використанням символу Якобі. Нехай p - Просте і a не ділиться на p. Позначимо $a^{(p-1)/2} \bmod p$ через x. Відповідно до теореми Ферма:

$$x^2 \equiv a^{p-1} \equiv 1 \bmod p,$$

тому $x \pm 1 \bmod p$. Насправді:

$$a^{\frac{p-1}{2}} \equiv \left(\frac{a}{p}\right) \bmod p.$$

Перевірка цього співвідношення для даного числа n називається тестом Соловея-Штрассена («СШ(а)»). На 203 мільйони простих, менших 2^{32} псевдопростих за основою 2 всього 10 403, з них тест Соловея-Штрассена проходять 5367 чисел.

1.6 Тест Міллера-Рабіна

Іншим покращенням методу Ферма є «метод Міллера-Рабіна». Нехай p - просте і p -1 ділиться на 2^k, $p-1 = q \cdot 2^k$. Позначимо a^q через y. Відповідно до теореми Ферма, $y^{2^k} \equiv 1 \bmod p$.

Нехай, наприклад, k =4. Тоді $y^{16} - 1 \equiv 0 \bmod p$ або: $(y^8+1)(y^4+1)(y^2+1)(y+1)(y-1) \equiv 0 \bmod p$.

Так як кільце відрахувань по простому модулю є полем і, отже, немає дільників нуля, одна з цих дужок повинна дорівнювати 0. Ця перевірка для даного числа *n* і називається тестом Міллера-Рабіна («MP(a)»).

Тест Міллера-Рабіна сильніший (включає більше перевірок) ніж тест Соловея-Штрассена. Складові числа, що проходять тест Міллера-Рабіна на основі *a* називаються сильно псевдопростими числами (strong pseudoprimes, SPSP(a)).

Розподіл простих чисел у певному діапазоні приведено в таблиці 1.7.

Повний список 104 складених чисел приведений на рисунку 1.1, менших 2^{32} і проходять тести «MP(2)» та «MP(3)».

Таблиця 1.7 – Порівняння результатів тестів простоти серед чисел менших 2^{32}.

простих:	203 280 221
псевдопростих на підставі 2:	10403
«СШ(2)»	5367
«MP(2)»	2314
псевдопростих на підставах 2 і 3:	2318
«СШ(2)» та «СШ(3)»	969
«MP(2)» та «MP(3)»	104
«MP(2)», «MP(3)» та «MP(5)»	6

Його можна використовувати для гарантованої перевірки простоти чисел розміром менше 2^{32}.

Насправді, для перевірки на простоту чисел менших 4759123141достатньо виконати тести MP(2), MP(7) і MP(61).

Пряма перевірка показує, що:

- MP(2) достатній для перевірки чисел < 2047.
- MP(2+3) достатній для перевірки чисел < 1373653 (1.3 млн.).
- MP(2+3+5) достатній для перевірки чисел < 25326001 (25 млн.).
- MP(2+3+5+7) достатній для перевірки чисел < 3215031751 (3 млрд.)

– МР(2+3+5+7+11) достатній для перевірки чисел $< 2\,152\,302\,898\,747\,(2 \cdot 10^{12})$.

– МР(2+3+5+7+11+13) достатній для перевірки чисел $< 3\,474\,749\,660\,383\,(3.4 \cdot 10^{12})$.

– МР(2+3+5+7+11+13+17) достатній для перевірки чисел $< 341\,550\,071\,728\,321\,(341 \cdot 10^{12})$.

– МР(2+3+5+7+11+13+17+23) достатній для перевірки чисел $< 3\,825\,123\,056\,546\,413\,051\,(3.8 \cdot 10^{18})$.

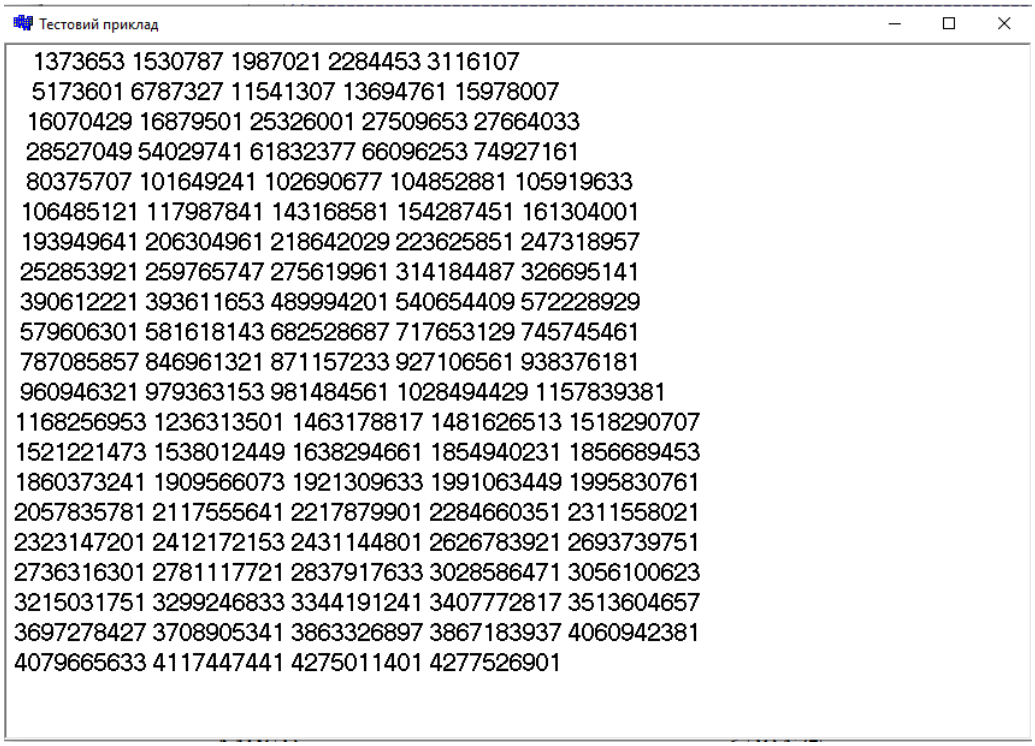

Рисунок 1.1 – Складені числа для тестування

У той же час, існують методи, які також є ймовірнісними, проте помилки бувають на інших групах чисел, ніж методи, засновані на теоремі Ферма. Один із них – «метод Люка» (він же – «Люк-Лемер»). Деякий варіант його спільного застосування із методом Ферма називається BPSW-методом.

1.7 Послідовності Люка

Нехай P, Q – два цілих числа. Послідовностями Люка $U_n = U_n(P,Q)$ і $V_n = V_n(P,Q)$ називаються послідовності, що задовольняють рекурентному співвідношенню:

$$U_n = PU_{n-1} - QU_{n-2},$$
$$V_n = PV_{n-1} - QV_{n-2}.$$

При $n \geq 2$ та початкових умов:

$$U_0 = 0, \quad U_1 = 1,$$
$$V_0 = 2, \quad V_1 = P.$$

При $P = 1$ і $Q = -1$ отримуємо:

$U_n = \{ 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots \}$ – числа Фібоначчі,

$V_n = \{ 2, 1, 3, 4, 7, 11, 18, 29, 47, 76, 123, \ldots \}$ – числа Люка,

При $P = 2$ і $Q = -1$: $U_n = \{ 0, 1, 2, 5, 12, 29, 70, 169, \ldots \}$ - числа Пелля, $V_n = \{ 2, 2, 6, 14, 34, 82, 198, 478, \ldots \}$ – числа Пелля-Люка. При $P = 3$ і $Q = 2$: $U_n = \{ 0, 1, 3, 7, 15, 31, 63, 127, \ldots \}$ – числа Мерсенна,

Число $D = P^2 - 4Q$ називається дискримінантом послідовності. У змістовних випадках воно має бути повним квадратом. Для чисел Фібоначчі дискримінант дорівнює 5.

Нехай:

$$\alpha, \beta = \frac{P \pm \sqrt{D}}{2}, \alpha + \beta = P, \alpha\beta = Q.$$

Тоді:

$$U_n = U_n(P,Q) = \frac{\alpha^n - \beta^n}{\alpha - \beta}, V_n = V_n(P,Q) = \alpha^n + \beta^n.$$

Послідовності Люка мають багато властивостей котрі варто досліджувати. Усі вони досить легко виводяться з цієї формули.

Алгоритми перевірки простоти числа пов'язані з числами Люка засновані на тому, що якщо p - просте і символ Якобі $jacobi(D, p) = -1$, то ірраціональна частина числа α^k дорівнює 0, якщо k ділиться на $p+1$. Якщо ж число n не просте, то ірраціональна частина числа α^{n+1} теж може виявитися дорівнює 0, але дуже рідко.

Основна перевага такого методу перевірки полягає в тому, що він помиляється зовсім на інших числах, ніж методи, що базуються на теоремі Ферма. Тому спільне застосування методів Міллера-Рабіна та Люка дає дуже надійний результат. На сьогоднішній день немає жодного прикладу, якби ці методи одночасно помилилися.

1.8 Метод Фробеніуса

Квадратичною ірраціональністю називатимемо число виду $z = a + b\,sqrt(c)$, де a,b,c – цілі числа, причому c вільно від квадратів.

Через z mod n позначатимемо число $(a \bmod n) + (b \bmod n)\,sqrt(c)$.

Складеним числом називатимемо число $z = a - b\,sqrt(c)$.

Поєднання мультиплікативно:

$$z_1 \cdot z_2 = z_1 \cdot z_2 .$$

Нормою $N(z)$ називатимемо ціле число:

$$N(z) = z \cdot z = a^2 - b^2 c.$$

Норма мультиптикативна: $N(z_1 z_2) = N(z_1) N(z_2)$ і $N(z \bmod n) = N(z) \bmod n$. Наприклад, нехай $z = 1+sqrt(3)$, тоді $z = 1-sqrt(3)$, $N(z) = N(z) = -2$, $z^{11} = 31648 + 18272\ sqrt(3)$, $N(z^{11}) = -2048 = (-2)^{11}$, $z^{11} \bmod 15 = 13 + 2\ sqrt(3)$.

Обчислення $z_a \bmod n$ приблизно вдвічі складніше, ніж операція для цілих чисел.

Якщо символ Якобі jacobi(*c,p*) дорівнює -1, то квадратичні ірраціональності за простим модулем *p* утворюють поле (поле Галуа) з p^2 елементів, що позначається $Z^p[sqrt(c)]$.

Нехай *p* - просте, $jacobi(c, p) = -1$ і *z* належить $Z^p[sqrt(c)]$. Тоді:

$$z^p \bmod p = z.$$

Для того щоб швидко та надійно перевірити число на простоту необхідно виконати ряд дій. Нехай *n* – непарне натуральне число, яке не є повним квадратом. Позначимо через *c* найменше серед чисел [-1,2,3,5,7,11,...] таке, що символ Якобі $jacobi(c/n) = -1$. Якщо $c \le 2$, то покладемо $z = 2 + sqrt(c)$, інакше $z = 1 + sqrt(c)$. Назвемо число *n* простим за Фробеніусом, якщо $z^n \equiv z \bmod n$.

Назвемо число псевдопростим за Фробеніусом (FPP) , якщо складне, але просто за Фробеніусом. Фактично, метод поєднує методи Ферма та Lucas-Lemer'a.

Невідомо жодного числа, псевдопростого Фробеніус, тобто. на якому метод помиляється. Доведено, що таких чисел немає менших $< 2^{60}$. Кратний простий множник FPP-числа не може бути меншим за 2^{32}. Простий

множник p FPP-числа n , у якого jacobi(c/p)=1, не може бути менше 2^{32} . Нехай p - простий множник FPP-числа n тоді $n/p > 2^{18}$.

Термін "тест Фробеніуса" (перевірки простоти чисел) іноді використовується в дещо іншому сенсі:

- Frobenius pseudoprime .
- Псевдопросте число Фробеніуса .

Теорему Фробеніуса можна використовувати для прострюювання ймовірнісного тесту простоти числа.

Нехай $n > 1$ - непарне, числа a, b, c такі, що a , b , c взаємно прості з n , $a - b^2 c \neq 0,1 \bmod n$ і jacobi (c / n) = -1. Тоді, якщо:

$$(a + b \text{ sqrt} (c))^n \equiv (a - b \text{ sqrt} (c)) \bmod p ,$$

то число n швидше за все, просте.

Як і інших аналогічних тестах підвищення надійності можна було б вимагати виконати цей тест кілька разів із різними a,b (міняти c немає сенсу). Але це зайве.

На сьогоднішній день не відомо жодного контрприкладу до цього тесту. Доведено, що їх немає за $n < 2^{60}$. Однак тест Фробеніуса в цьому місці книги розуміється в дещо іншому сенсі.

Враховуючи відсутність контрприкладів, замість того, щоб говорити про довільний (випадковий) вибір параметрів a , b , c , ми їх просто зафіксуємо.

Нехай n – непарне натуральне число, яке не є повним квадратом. Його індексом Фробеніуса $Ind_F (n)$ називатимемо найменше серед чисел [-1,2,3,4,5,6,...] таке, що символ Якобі $jacobi(c/n) = -1$.

З мультиплікативності символу Якобі слід, що й індекс $c = Ind_F (n)$ позитивний, він простий.

Неважко з'ясувати, коли індекс Фробеніуса набуває маленьких значень:

- Якщо $n \equiv 3 \bmod 4$, то $Ind_F (n) = -1$.
- Якщо $n \equiv 5 \bmod 8$, то $Ind_F (n) = 2$.

– Якщо $n \equiv 17 \bmod 24$, то $Ind_F(n) = 3$.

– Якщо $n \equiv 1 \bmod 24$, то $Ind_F(n) \geq 5$.

Нехай n – непарне натуральне число, яке є повним квадратом, позначимо $Ind_F(n)$ через c.

Якщо $c \leq 2$, то покладемо $z = 2 + \text{sqrt}(c)$, інакше $z = 1 + \text{sqrt}(c)$. Назвемо число n простим за Фробеніусом , якщо:

$$z^n \equiv z \mod n .$$

Нехай n =19. Тоді c = -1, $z = 2 + i$, z^n = -3565918 + 2521451 $i \equiv 2 - i \bmod n$.

Нехай n =33. Тоді c = -1, $z = 2 + i$, $z^n \equiv 2+22\ i \bmod n \neq z$.

Нехай n =17. Тоді c = 3, z = 1 + sqrt (3), z^n = 13160704+7598336 sqrt(3) ≡ 1-sqrt(3) mod $n = z$.

Звісно, що нас цікавлять випадки, коли цей метод помиляється.

Складне непарне натуральне число n назвемо *псевпростим за Фробеніус* (Frobenius pseudoprime, FPP), якщо воно просто за Фробеніус.

Зазначимо, що якщо n псевдопросто за Фробеніусом, то n псевдопросто на підставі $N(z)$, тобто тест Фробеніуса включає тест Ферма. Псевдопростих за Фробеніус чисел не існує. Іншими словами, тест Фробеніуса ніколи не помиляється.

Не треба намагатися знайти контрприклад прямим перебором. Доведено, що немає серед чисел менших 2^{60}. Набагато більше шансів знайти його як добуток простих.

Для числа n , псевдопростого за Фробеніус, зафіксуємо позначення:

– $c = Ind_F(n)$ - індекс Фробеніуса числа n , c з [-1,2,3,5,7,...],

– $z = a + b$ sqrt(c), тобто z =2+sqrt(c) або z =1+sqrt(c) залежно від c .

У визначенні простоти за Фробеніусом входить знаходження індексу Фробеніуса. Наскільки може бути великий? Та скільки завгодно. Однак, якщо розглядати числа обмеженої довжини, можна знайти і обмеження для цього індексу. Доведено , що й брати числа, менші 2^{64} , їх індекс Фробеніуса не

перевищує 277. Більше того, чисел з індексом більше 128 є 458 069 912 штук і всі вони не є псевдопростими.

Таким чином перевірено, що для псевдопитих за Фробеніусом чисел менших 2^{64} їх індекс Фробеніуса не перевищує 127.

Нехай n псевдопросто за Фробеніус, $c = Ind_F(n)$ - його індекс Фробеніуса. Простий дільник p числа n називатимемо Ф-негативним, якщо символ Якобі jacobi(c/p)=-1 і Ф-позитивним, якщо jacobi(c/p)=1.

Нагадаю, що згідно з визначенням, якщо jacobi(c/p) =1, то є квадратом по простому модулю p, тобто існує d таке, що $c \equiv d^2 mod$ p . Ф-позитивні дільники у псевдопростих за Фробеніусом чисел повинні задовольняти певну умову, яка виконується дуже рідко.

Нехай n псевдопросто Фробеніус і $n = p^s q$, де p, Φ - негативний множник і $s > 1$. Тоді:

$$z^p \equiv z \bmod p^2 .$$

Прості числа, що задовольняють цій умові зустрічаються рідко, не вдалося знайти жодного прикладу таких чисел. Прямим перебором можна одержати таке.

У псевдопростих за Фробеніус чисел при $c < 128$ не існує не існує кратних множників менших 2^{32}.

2 МАТЕМАТИЧНІ ОСНОВИ BPSW ТЕСТУ ПРОСТОТИ

2.1 Тест на простоту BPSW

Тест на простоту Бейлі-PSW (BPSW або BSW) фактично є тестом на складність, схожим на тест Ферма та тест простоти Міллера-Рабіна. Він названий на честь Роберта Бейлі, Карла Померанса, Джона Л. Селфріджа та Семюеля С. Вагстаффа-молодшого. Очевидно, вперше алгоритм був розроблений Бейлі (Baillie і Wagstaff, 1980), а Селфрідж (Pomerance, Selfridge та Wagstaff, 1980). Процедура полягає в наступному; зауважте, що деякі деталі можуть відрізнятися від однієї реалізації до іншої (Martin, 2004).

- Опрацювати всі $N < 3$ і всі парні N.
- Необхідно перевірити N для будь-яких малих простих дільників $p < 1000$.
- Виконати перевірку Міллера-Рабіна (сильне вірогідне просте число), основа 2, на N.
- Виконати (стандартний або сильний) тест Лукаса-Селфріджа на N, використовуючи послідовності Лукаса з параметрами, запропонованими Селфріджем.

На будь-якому етапі цієї процедури N можна виявити що число однозначно складене. Якщо ні, число пройшло (стандартний або сильний) тест BPSW і є простим або (стандартним або сильним) псевдопростим BPSW.

Станом на цей час, як стандартний, так і сильний тести працюють э одними з найбільш ефективних; невідомо жодного винятку (стандарт BPSW або сильний тест простоти). За допомогою таблиці сильних псевдопростих чисел з основою 2 Річарда Пінча автор підтвердив (травень 2005), що не існує псевдопростого числа BPSW (стандартного чи сильного) для $N < 10^{13}$. У січні 2007 року за допомогою коду автора та таблиці псевдопростих чисел Вільяма Голвея Мартін Фуллер визначив, що не існує псевдопростого числа BPSW (стандартного чи сильного) для $N < 10^{15}$. Нещодавно Джефф Гілкріст ,

використовуючи код автора та базу даних псевдопростих чисел, підготовлену Яном Фейтсмою, підтвердив (13 червня 2009), що псевдопросте число BPSW (стандартне або сильне) не існує нижче $N < 10^{17}$. Крім того, аналіз (24 жовтня 2009) Гілкрістом, розширення бази даних Feitsma до 2^{64} не виявив жодного псевдопростого числа BPSW (стандартного чи сильного) нижче 2^{64} (приблизно це дорівнює 1,8446744e19). Цю нижню межу 2^{64} для будь-якого псевдопростого числа BPSW згодом було окремо перевірено (11 липня 2011) Чарльзом Грейтхаусом; однак він також користувався базою даних Фейтсми, тому повністю незалежна перевірка ще не проводилася.

Необхідно зауважити, що Карл Померанс (1984) представив евристичний аргумент про те, що існує нескінченна кількість контрприкладів для стандартного тесту (і, ймовірно, також для сильного тесту, заснованого на подібних міркуваннях), і навіть що (для достатньо великого числа *x*, котре залежить від μ) кількість псевдопростих чисел BPSW менших за *x* перевищує $x^{(1-\mu)}$, де μ є як завгодно малим попередньо присвоєним додатнім числом. Тим не менш, досі не знайдено жодного псевдопростого числа BPSW. Отже, тест BPSW є ефективним проте не до кінця достовірним, що перевищує показники конкуруючих алгоритмів, таких як ймовірнісні тести Міллера-Рабіна.

Вважається, що деякі комерційні пакети математичних програм покладаються, частково або повністю, на тест BPSW для перевірки простоти, наприклад, Ribenboim [3], також Martin [4]. У деяких випадках ці пакунки використовують сильний тест BPSW та/або Лукаса або додають додаткові тести Міллера-Рабіна.

BPSW має наступну обчислювальну складність $O(\log n^3)$ бітових операцій, як і тест Ферма та алгоритм Міллера-Рабіна. Однак тест BPSW зазвичай вимагає приблизно в три-сім разів більше бітових операцій, ніж один тест Міллера-Рабіна. Сильна версія BPSW відрізняється лише заміною стандартного тесту Лукаса-Селфріджа на сильний тест Лукаса-Селфріджа. Сильний тест Лукаса-Селфріджа дає лише приблизно на 30 % більше псевдопростих чисел, ніж стандартна версія; наприклад, серед непарних

композитів $N<10^6$ є 219 стандартних псевдопростих чисел Лукаса-Селфріджа, 58 сильних псевдопростих чисел Лукаса-Селфріджа та 46 сильних псевдопростих чисел за основою 2 (ці загальні суми припускають відсутність скринінгу за допомогою непарних пробних дільників). Оскільки потужний тест Лукаса-Селфріджа вимагає приблизно на 10 % більше часу роботи, сильні тести виявляються більш ефективними.

Селфрідж вказав такі параметри для генерації послідовностей Лукаса: $P=1$ і $Q=(1-D)/4$, де D - перше ціле число в послідовності {5, -7, 9, -11, 13, - 15, ...}, для якого НСД(D,N)=1 і символ Якобі ($D|N$) = -1. Однак зауважте, що якщо N є ідеальним квадратом, такого D не існуватиме, і пошук D триватиме до ±sqrt(N), після чого НСД(D,N)=sqrt(N) буде вказувати, що N як складене. Отже, алгоритм також передбачає наявність попередньої перевірки на ідеальні квадрати (а також на парні цілі чи цілі числа менші 3).

Додаткові умови, які іноді накладаються, що N не ділить Q і що число D не має цілих квадратів, виявилися нерелевантними для цього застосування послідовностей Лукаса.

Залишається теоретично можливим, що для дуже великого N необхідний D буде настільки великим за величиною, що алгоритм стане занадто обчислювально складним. Враховуючи виключення ідеальних квадратичних значень N, на практиці такої поведінки не спостерігалося; справді, діапазон значень, що спостерігаються для D, досить малий - $|D|\max=47$ для $N<10^6$, $|D|\max=67$ для $10^{19}<N<10^{19}+10^{16}$. Бейлі та Вагстафф [4] представили аналітичні аргументи на підтримку цього спостереження [5]. Однак цю можливість не можна повністю виключити, і авторський код містить пастку переповнення для D.

Алгоритм BPSW має кілька різних реалізацій, що відрізняються один від одного лише деталями. Розглянемо алгоритм, що має вигляд:

1. Виконати тест Міллера-Рабіна на підставі 2.

2. Виконати сильний тест Лукаса-Селфріджа, використовуючи послідовності Лукаса з параметрами Селфріджа.

3. Повернути "просте" тільки в тому випадку, коли обидва тести повернули "просте".

Крім того, на початок алгоритму можна додати перевірку на тривіальні дільники, скажімо, до 1000. Це дозволить збільшити швидкість роботи на складових числах, щоправда, дещо уповільнивши алгоритм на простих.

Отже, алгоритм BPSW ґрунтується на наступних кроках:

1. (факт) тест Міллера-Рабіна і тест Лукаса-Селфріджа якщо і помиляються, то тільки в один бік: деякі складові цих алгоритмів пізнаються як прості. У зворотний бік ці алгоритми ніколи не помиляються.

2. (припущення) тест Міллера-Рабіна і тест Лукаса-Селфріджа якщо і помиляються, то ніколи не помиляються на одному числі одночасно.

Насправді, друге припущення начебто як і неправильне - евристичний доказ-спростування Померансу наведено нижче. Тим не менш, на практиці жодного псевдопростого досі не знайшли, тому умовно вважатимуться друге припущення вірним.

2.2 Сильний тест Лукаса-Селфріджа

Сильний тест Лукаса-Селфріджа складається з двох частин: алгоритму Селфріджа для обчислення деякого параметра і сильного алгоритму Лукаса, що виконується з цим параметром.

Серед послідовності 5, -7, 9, -11, 13 ... знайти перше число *D*, для якого $J(D,N) = -1$ і $\gcd(D,N) = 1$, де $J(x,y)$ - Символ Якобі.

Параметрами Селфріджа будуть $P = 1$ і $Q = (1-D)/4$.

Слід зазначити, що параметр Селфріджа немає для чисел, які є точними квадратами. Справді, якщо число є точним квадратом, то перебір *D*

досягне *sqrt(N)*, у якому виявиться, що $\gcd(D,N)>1$, тобто. виявиться, що число складене.

Крім того, параметри Селфріджа будуть обчислені неправильно для парних чисел та для одиниці; втім, перевірка цих випадків не складе труднощів.

Таким чином, перед початком алгоритму слід перевірити, що число N є непарним, великим 2 і не є повним квадратом, інакше (при невиконанні хоча б однієї умови) потрібно відразу вийти з алгоритму з результатом "число складене".

Нарешті, зауважимо, що й D для деякого числа N виявиться занадто великим, алгоритм з високою обчислювальною складністю. Хоча такі виключення не помічено на практиці (виявлялося цілком достатньо 4-байтного числа), проте ймовірність цієї події не варто виключати. Втім, наприклад, на відрізку $[1;10^6]\max(D)=47$, а на відрізку $[10^{19};10^{19}+10^6]\max(D)=67$. Крім того, Бейлі та Вагстаф у 1980 році аналітично довели це спостереження [6].

2.3 Сильний алгоритм Лукаса

Параметрами алгоритму Лукаса є числа D, P і Q такі, що $D=P^2-4\cdot Q$ D = P ^ 2 - 4 * Q і $P>0$.

Неважко помітити, що параметри, обчислені за алгоритмом Селфріджа, задовольняють цим умовам. Послідовності Лукаса - це послідовності U_k і V_k, що визначаються таким чином:

$$U_0=0;$$

$$U_1=1;$$

$$U_k=PU_{k-1}-QU_{k-2};$$

$$V_0=2;$$

$$V_1 = P;$$

$$V_k = PV_{k-1} - QV_{k-2}.$$

Далі припустимо, не хай $M = N - J(D,N)$. Якщо N просте, і $\gcd(N,Q) = 1$ gcd, маємо:

$$U_m = 0(\mathrm{mod}N).$$

Зокрема, коли параметри D, P, Q обчислені алгоритмом Селфріджа, маємо:

$$U_{N+1} = 0(\mathrm{mod}N).$$

Зворотнє перетворення, взагалі кажучи, виконувати неправильно. Тим не менш, псевдопростих чисел при виконанні даного алгоритму виявляється не дуже багато, на чому, власне, і ґрунтується алгоритм Лукаса.

Отже, алгоритм Лукаса полягає у обчисленні U_M та прирівнянні його до нуля.

Далі необхідно знайти якийсь спосіб прискорення обчислення U_K, інакше, зрозуміло, що ніякого практичного сенсу в цьому алгоритму не було б.

Отримаємо:

$$U_k = (a^k - b^k)/(a-b),$$

$$V_k = a^k + b^k,$$

де a і b – різні корені квадратного рівняння $x^2 - Px + Q = 0$.

Тепер такі рівності можна довести елементарно:

$$U_{2k} = U_k V_k (\mathrm{mod}N);$$

$$V_{2k} = V_k^2 - 2Q^k \,(\mathrm{mod} N).$$

Тепер, якщо уявити $M = E2^T$, де E - непарне число, легко отримати:

$$U_M = U_E V_E V_{2E} V_{4E} \ldots V_{2^{T-2}E} V_{2^{T-1}E} = 0 \,(\mathrm{mod}\, N),$$

і хоча один із множників дорівнює нулю по модулю N.

Відомо, що досить обчислити U_E і V_E, проте наступні множники $V_{2E}\, V_{4E} \ldots V_{2^{T-2}E}\, V_{2^{T-1}E}$ можна отримати з них .

Таким чином, залишилося оптимізувати обчислення U_E та V_E для непарного E.

Спочатку розглянемо такі формули додавання членів послідовностей Лукаса:

$$U_{i+j} = (U_i V_j + U_j V_i)/2 \,(\mathrm{mod}\, N),$$
$$V_{i+j} = (V_i V_j + DU_i U_j)/2 \,(\mathrm{mod}\, N).$$

Слід звернути увагу, що ділення виконується у полі $(\mathrm{mod}\ N)$. Формули ці доводяться дуже просто, і тут їхній доказ опущений. Тепер, володіючи формулами для складання і для подвоєння членів послідовностей Лукаса, зрозумілий спосіб прискорення обчислення U_E і V_E .

Дійсно, розглянемо двійковий запис числа E. Покладемо спочатку результат - U_E і V_E - рівними, відповідно, U_1 і V_1 . Пройдемося по всіх бітах числа E від молодших до старших, пропустивши тільки перший біт (початковий член послідовності). Для кожного i-го біта обчислюватимемо U_2^i і V_2^i із попередніх членів за допомогою формул подвоєння. Крім того, якщо поточний i-ий біт дорівнює одиниці, то до відповіді додаватимемо поточні U_2^i

і V_2^i за допомогою формул додавання. Після закінчення алгоритму, що виконується за $O(\log(E))$, отримаємо шукані U_E та V_E.

Якщо U_E або V_E дорівнювали нулю $(\bmod N)$, то число *N* просте (або псевдопросте). Якщо вони обидва відмінні від нуля, то обчислюємо $V_{2E}, V_{4E} \ldots V_{2^{T-2}E}, V_{2^{T-1}E}$. Якщо хоча б один з них можна порівняти з нулем по модулю *N*, то число *N* просте (або псевдопросте). Інакше число *N* є складеним.

2.4 Алгоритм Селфріджа

Тепер, коли було досліджено алгоритм Лукаса, можна докладніше зупинитися з його параметрах *D,P,Q*, однією з способів отримання яких є алгоритм Селфриджа.

Нагадаємо базові вимоги до параметрів:

$$P > 0,$$

$$D = P^2 - 4 \cdot Q ? 0.$$

Тепер продовжимо дослідження цих параметрів. *D* має бути точним квадратом $(\bmod N)$. Справді, інакше отримаємо:

$D = b^2$, звідси $J(D,N) = 1$, $P = b + 2$, , звідси .

Тобто. якщо *D* – точний квадрат, то алгоритм Лукаса стає практично звичайним імовірнісним тестом.

Один з кращих способів уникнути подібного – підбирати параметри так, щоб $J(D,N) = -1$.

Наприклад, можна вибрати перше число *D* із послідовності 5, -7, 9, -11, 13, ..., для якого $J(D,N) = -1$. Також нехай $P = 1$. Тоді $Q = (1 - D)/4$. Цей спосіб був запропонований Селфріджем.

Втім, є й інші способи вибору *D*. Можна вибирати його з послідовності 5, 9, 13, 17, 21, ... Також нехай *P* - найменше непарне, що приводить до $\mathrm{sqrt}(D)$. Тоді $Q = (1 - D)/4$.

Зрозуміло, що від вибору конкретного способу обчислення параметрів Лукаса залежить його результат - псевдопрості можуть відрізнятися при різних способах вибору параметра. Як показала практика, алгоритм, запропонований Селфріджем, виявився дуже вдалим: всі псевдопрості Лукаса-Селфріджа не є псевдопростими Міллера-Рабіна, принаймні жодного контрприкладу знайдено не було.

2.5 Евристичний доказ-спростування Померансу

Померанс 1984 року запропонував такий евристичний доказ. Його ствердження наступне: кількість BPSW-псевдопростих від 1 до X більша за X^{1-a} для будь-якого $a > 0$.

Нехай $k > 4$ – довільне, але фіксоване число. Нехай *T* - деяка велика кількість.

Нехай $P_k(T)$ - множина таких простих p в інтервалі $[T; T^k]$, для яких:

(1) $p = 3 (\mathrm{mod} 8), J(5, p) = -1$,

(2) число $(p-1)/2$ не є точним квадратом,

(3) число $(p-1)/2$ складено виключно з простих $q < T$,

(4) число $(p-1)/2$ складено виключно з таких простих *q*, що $q = 1 (\mathrm{mod} 4)$,

(5) число $(p+1)/4$ не є точним квадратом,

(6) число $(p+1)/4$ складено виключно з простих $d < T$,

(7) число $(p+1)/4$ складено виключно з таких простих *d*, що $q = 3 (\mathrm{mod} 4)$.

Зрозуміло, що приблизно 1/8 всіх простих у відрізку $[T; T^k]$ задовольняє умову (1). Також можна показати, що умови (2) та (5) зберігають деяку частину

чисел. Евристично, умови (3) і (6) дозволяють залишити деяку частину чисел з відрізка $(T; T^k)$. Нарешті, подія (4) має ймовірність $(c(\log T)^{-1/2})$, як і і подія (7). Таким чином, потужність множини $P_k(T)$ приблизно дорівнює при $T \to \infty$.

$$\frac{cT^k}{\log^2 T},$$

де c - деяка позитивна константа, яка залежить від вибору k.

Тепер можемо побудувати число n, яке не є точним квадратом, складене з l простих з $P_k(T)$, де l непарно і менше $T^2/\log(T^k)$. Кількість способів вибрати таке число n є приблизно:

$$\binom{[cT^k/\log^2 T]}{l} > e^{T^2(1-3/k)},$$

для великого T та фіксованого k. Крім того, кожне таке число n менше від e^{T^2}.

Позначимо через Q_1 добуток простих $q < T$, для яких $q = 1 (\mathrm{mod}\ 4)$ а через Q_3 - добуток простих $q < T$, для яких $q = 3 (\mathrm{mod}\ 4)$. Тоді $\gcd(Q_1 Q_3) = 1$ і $Q_1 Q_3 ? e^T$. Таким чином, кількість способів вибрати n з додатковими умовами:

$$n = 1 (\mathrm{mod}\ Q_1), n = -1 (\mathrm{mod}\ Q_3)$$

Отримаємо наступне твердження: $e^{T^2(1-3/k)}/e^{2T} > e^{T^2}(1-4/k)$ для великого T.

Але кожне таке n - це контрприклад до тесту BPSW. Дійсно, n буде числом Кармайкла (тобто числом, на якому тест Міллера-Рабіна буде помилятися за будь-якої підстави), тому воно автоматично буде псевдопростим на підставі 2. Оскільки $n = 3 (\mathrm{mod} 8)$ і кожне $p | n$ дорівнює $3 (\mathrm{mod} 8)$, очевидно,

що n також буде сильним псевдопростим на основі 2. Оскільки $J(5,n) = -1$, то кожне просте $p|n$ задовольняє $J(5,p) = -1$, так як $p+1|n+1$ для будь-якого простого $p|n$, звідси випливає, що n - псевдопростий Лукаса для будь-якого тесту Лукаса з дискримінантом 5.

Таким чином, ми показали, що для будь-якого фіксованого k і всіх великих T буде як мінімум $e^{T^{2(1-4/k)}}$ контрприкладів до тесту BPSW серед чисел, менших e^{T^2}. Тепер, якщо ми припустимо $x = e^{T^2}$ буде як мінімум $x^{1-4/k}$ контрприкладів, менших x. Оскільки k - випадкове число, то доказ означає, що кількість контрприкладів, менших x, є число, більше x^{1-a} для будь-якого $a > 0$.

2.6 Вдосконалення BPSW тесту простоти

Для перевірки чисел на простоту дуже ефективним є такий метод [7, 5, 1]. Нехай n – непарне натуральне число, в якому хочемо перевірити простоту. Це пропонується робити наступним чином.

Крок 1. Тест Міллера - Рабіна на підставі 2.

Крок 2. Тест Лукаса - Сельфідж. Зазначимо, що якщо у тесті із послідовності 5, −7, 9, . . . вибрано параметр D, то фактично при цьому перевіряється, що число n взаємно-просте з усіма числами, не перевищують $|D|$.

Числа, які проходять ці два тести, називатимемо BPSW псевдопростими.

Для натурального n і цілих (P, Q) позначимо через

$R(n) = R(n, P, Q)$ функцію (тут $D = P^2 - 4 * Q$):

$$R(n,P,Q) = \begin{cases} 0, J(D,n) = 0 \\ \gcd(2^{n-1} - 1, U_{n-1}(P,Q)), J(D,n) = 1 \\ \gcd(2^{n-1} - 1, U_{n+1}(P,Q)), J(D,n) = -1 \end{cases} \quad (1)$$

Якщо p – просте і взаємно-просте з $PQ(1-4Q)$, то $R(p)$ ділиться на p.

Значення функції R не дуже великі і можуть бути обчислені при не надто великих n ($\approx$ до 2^{18}).

Нехай n - BPSW-псевдопростий, $n = pq$, де p - просте. Тоді p – простий дільник $R(q)$.

Так як $2^{pq-1} = 2^{(p-1)q+(q-1)} \equiv 2^{q-1} \equiv 1 \mod p$, тобто $2^{q-1}-1$ ділиться на p.

Нехай тест Лукаса виконається з параметром $Q,D = 1\text{-}4Q$. По умові $J(D/q) = -1$. Нехай $J(D/q) = +1$. Тоді $J(D/q) = -1$. Так як:

$$U_{pq+1} = U_{(p-1)q+q+1} \equiv U_{q+1} \equiv 0 \bmod p,$$

так як U_{q+1} ділится на p. Таким чином, число p повинне ділити $2^{q-1}-1$ і U_{q+1}, тобто p- простий дільник $R(q)$.

Нехай $J(D/q) = -1$. Тоді $J(D/q) = +1$. Так як: $U_{pq+1} = U_{(p+1)q-q+1} \equiv U_{(p+1)-(q-1)} \equiv 0 \bmod p$.

Звідси слідує, що $U_{q-1} \equiv 0 \bmod p$, так як p простий дільник U_{q-1}, а значить, p знову буде дільником $R(q)$. Ця теорема дозволяє ефективно перевірити на псевдопростоту числа виду pq, де q – довільне натуральне, яке не перевищує деякої межі, а p – просте. Наприклад, нап $q < 100\,000$ і $|D| \leq 21$ таких чисел не виявлено. За допомогою бібліотеки GMP можна перевірити всі числа в межах до 218 і навіть дещо більше [8].

Крім того, немає жодного простого числа q, меншого за 220 такого, що $J(5, q) = -1$ і у R-функції якого $R(q) = R(q, P, Q)$ був би простий дільник, більший q.

Також можна перевірити і таке твердження. За $|D| < 128$ серед чисел $q < 218$, $J(D, q) = -1$ немає жодного, який мав би простий дільник числа $R(q)$ більший q.

Можна дещо уточнити. Нехай n - BPSW-псевдопростий, $n = pq$, де p -

просте при деякому $D = 1 - 4Q$. Тоді: якщо $J(D/p) = +1$, то p – простий дільник одного із чотирьох чисел.

$$\gcd(2^k-1,U_k),\ \gcd(2^k-1,V_k),\gcd(2^k+1,U_k),\ \gcd(2^k+1,V_k); \tag{2}$$

якщо $J(D/p) = -1$, то p – простий дільник одного із чотирьох чисел $\gcd(2^k-1,U_{k+1}),\ \gcd(2^k-1,V_{k+1}),\gcd(2^k+1,U_{k+1}),\ \gcd(2^k+1,V_{k+1}),$ де $k = (q-1)/2$.

Це слідує з того, що $2^{2k}-1=(2^k-1)(2^k+1)$ і $U_{2k}=U_kV_k$.

При конкретних обчисленнях ця теорема зручніша, ніж (7), оскільки має справу з числами вдвічі меншою за довжину [10].

Простий множник p з $J(D/p) = +1$.

Нехай n – BPSW-псевдопросте, $n = pq$, де p – просте та $J(D/p) = 1$.

Теорема (7) стверджує у разі, що p – простий дільник $2^{q-1}-1$ і U_{q+1}, та:

$$\begin{aligned} &a)\,2^{q-1}\equiv 1 \bmod p,\\ &b)\,U_{q+1}\equiv 0 \bmod p. \end{aligned} \tag{3}$$

Позначимо $(q-1)/2$ через k. Тоді (нагадаємо, $U_{2n}=U_nV_n$):

$$\begin{aligned} &a)\,(2^k-1)(2^k+1)\equiv 0 \bmod p,\\ &b)\,U_{k+1}V_{k+1}\equiv 0 \bmod p. \end{aligned} \tag{4}$$

Нехай γ :

$$\gamma=\frac{\alpha}{\beta}=\frac{P+\sqrt{Q}}{P-\sqrt{Q}}=\frac{P}{Q}\alpha-1.$$

Оскільки $J(D/p)=1$, то $\sqrt{Q}$ належить Z_p, отже і $\gamma \in Z_p$. Тому умови (4) можна записати так (нагадаю, $k = (q-1)/2$):

$$\begin{aligned} &a)\, 2^k \equiv \pm 1 \bmod p, \\ &b)\, \gamma^{k+1} \equiv \pm 1 \bmod p. \end{aligned} \tag{5}$$

Нехай n - BPSW-псевдопростий, $n = pq$, де p - просте і $J(D/p) = 1$. Тоді

а) число p задовольняє умову:

$$\gcd(ord(2, p), \rho(p, P, Q)) \le 2,$$

б) при фіксованому p число q задовольняє умові:

$$q \equiv q_0 \bmod LCM\ (\ ord(\ 2,p\ ),\rho,\rho(p,P,\ ),$$

де q_0 знаходиться з умов:

$$\begin{aligned} &a)\, q_0 \equiv 1 \bmod ord(2, p), \\ &b)\, q_0 \equiv -1 \bmod \rho(p, P, Q). \end{aligned}$$

Нагадаємо, що в цьому випадку і $ord(2, p)$, і $\rho(p,P,Q)$ є дільниками $p - 1$, а LCM($ord(2, p)$, $\rho(p,P,Q)$) буде дільником $2(p - 1)$ [11].

Доведення . Нехай $n = pq$. Оскільки $2^{pq-1} \equiv 1 \bmod p$ то $2^{q-1} \equiv 1 \bmod p$ або $q - 1 \equiv 0 \bmod \text{ord}(2, p)$ або:

$$q \equiv 1 \bmod ord(2,p). \tag{6}$$

Далі,

$$\mathrm{U}_{pq+1}(p,Q) \equiv 0 \bmod n \Rightarrow U_{pq+1}(P,Q) \equiv 0 \bmod p.$$

Маємо: $U_{p-1} \equiv 0 \bmod p$, тому:

$$\mathrm{U}_{pq+1} = U_{(p-1)q+q+1} \equiv U_{q+1} \equiv 0 \bmod p,$$

Тобто $q+1$ ділиться на $\rho(p,P,Q)$ або:

$$q \equiv -1 \bmod \rho(p,P,Q). \tag{7}$$

Виходить q має задовольняти умовам:

$$a)\, q \equiv 1 \bmod ord(2,p),$$
$$b)\, q \equiv -1 \bmod \rho(p,P,Q).$$

що доводить (б), причому і ord(2, p), і $\rho(p,P,Q)$ є дільниками $p-1$. Припустимо, що:

$$d = GCD(ord(2,p), \rho(p,P,Q)) > 2.$$

Тоді:

$$a)\, q \equiv 1 \bmod d,$$
$$b)\, q \equiv -1 \bmod d.$$

Числа, які відповідають умовам теореми, існують, але їх не так багато. Для їхнього пошуку можна скористатися наступним твердженням. Нехай n - BPSW-псевдопростий, p - простий дільник n і $J(D/p) = 1$. Тоді:

$$ord(2,p) \cdot \rho(p,P,Q) \le 2(p-1).$$

Оскільки ми маємо обмеження знизу на $\rho(n)$ (наслідок 4), можна написати таку нерівність [12].

Нехай n - BPSW-псевдопросте, $|Q| < 100$, p – простий дільник n та $J(D/p) = 1$. Тоді:

$$\frac{p-1}{ord(2,p)} > \ln(p)/4 + 1/2.$$

Нехай n - BPSW-псевдопростий, p - простий дільник n і $J(D/p) = 1$. Тоді:

а) якщо $p - 1$ ділиться на 2^k, $k > 1$, або $2^s = 1$, або $U_s = 0$, де $s = (p - 1)/2^{k-1}$;

б) якщо r – непарний простий дільник $p - 1$ кратності $k \geq 1$, то або $2^s = 1$, або $U_s = 0$, де $s = (p - 1)/r^k$.

Доведення випливає з того, що:

$$\gcd(ord(2,p),\rho p),\rho(Q)) \leq 2.$$

Числа p, менші 234 (17.17 млрд), які відповідають умовам теореми (10), знайдені при $|D| < 128$.

За простих D і $D = -15$ їх виявилося 170, їх 5 – при $Q = -1$. Це 61681, 363 101449, 4 278 255 361, 4 562 284 561 і 4 582 537 681.

Простий множник p, з $J(D/p) = -1$. При $|Q| < 32$ серед чисел $q < 2^{18}$, $J(1 - 4Q, q) = -1$ немає жодного, який мав би простий дільник числа $R(q)$ більший q.

Для простого p і цілих P, Q через $L(p) = L(p, P, Q)$ позначимо число:

$$L = LCM(ord(2, p), \rho p) P, Q)) .$$

Нехай n - BPSW-псевдопростий, $n = pq$, де p - просте і $J(D, p) = -1$. Тоді:

$$q \equiv 1 \bmod L(p, P, Q))$$

Нагадаємо, що в цьому випадку ord(2, p) – дільник $p - 1$, а $\rho(p,P,Q)$ – дільник $p+1$. Оскільки $J(D/p) = -1$, то $U_{p+1} \equiv 0 \bmod p$. Ми знаємо, що: $U_{pq+1} \equiv 0 \bmod pq, 2^{pq-1} \equiv 1 \bmod pq,$ або, враховуючи, що $U_{p+1} \equiv 0 \bmod p, U_{p-1} \equiv 0 \bmod p, 2^{q-1} \equiv 1 \bmod p,$ або

$q-1 \equiv 0 \bmod \rho(p,P,Q), q-1 \equiv 0 \bmod ord(2,p),$ або

$q \equiv 1 \bmod \rho(p,P,Q), q \equiv 1 \bmod ord(2,p).$

Ці дві умови можна поєднати в одну:

$$q \equiv 1 \bmod LCM(ord(2,p), \rho\, p)\, P, Q)).$$

Нехай n - BPSW-псевдопростий, $n = pq$, де p - просте і $J(D/p) = -1$. Позначимо $LCM(ord(2,p), \rho\, p)\, P, Q))$ через L. Тоді

а) якщо L ділиться на 5 та $J(p/5) = -1$, то $D = 5$;

б) якщо L ділиться на 5 та $J(p/5) = +1$, то $|D| > 5$;

в) якщо L ділиться на просте $s > 5$ та $J(p/s) = -1$, то $|D| \le s$;

г) якщо L ділиться на 3 та $J(p/3) = -1$, то $|D| \le 15$.

Сформулюємо трохи інакше.

Нехай n - BPSW-псевдопростий, $n = pq$, де p - просте і $J(D/p) = -1$. Позначимо $LCM(ord(2,p), \rho\, p)\, P, Q))$ через L. Тоді

а) якщо L ділиться на просте більше 3 і $J(p/s) = -1$, то $|D| \le s$;

б) якщо L ділиться на 3 та $J(p/3) = -1$, то $|D| \le 15$.

Нехай n - BPSW-псевдопростий, $n = pq$, де p - просте і $J(D/p) = -1$. Тоді

а) якщо ord(2, p) ділиться на просте більше 3 і $J(p/s) = -1$, то $|D| \le s$;

б) якщо ord(2, p) ділиться на 3 та $J(p/3) = -1$, то $|D| \le 15$.

Зберемо разом все, що відомо, у випадку $n = pq$ – добуток двох простих чисел. Оскільки $J(D/n) = -1$, то нехай $J(D/p) = -1$, $J(D/q) = +1$. Вважатимемо також, що $|Q| < 32$. Тоді:

(1) $J(D_0/n) = J(D_0/p)J(D_0/q) = 1$ за всіх $D_0 = 5, -7, \ldots$ менших за модулю $|D|$.

(2) p – простий дільник $R(q) = \gcd(2^{q-1} - 1, U_{q-1}(P,Q))$.

Це ж можна сформулювати так:

а) p – простий дільник $2^{q-1}-1$,

б) p – простий дільник $U_{q-1}(P,Q)$.

(3) q – простий дільник $R(p) = \gcd(2^{p-1}-1, U_{p+1}(P,Q))$.

Це ж можна сформулювати так:

а) q – простий дільник $2^{p-1}-1$,

б) q – простий дільник $U_{p+1}(P,Q)$.

(4) $p, q > 2^{18}$ (кордон може бути розширений).

(5) $\gcd(ord(2,q), \rho(q,P,Q)) \le 2$.

(6) Для q < 2^{34} є фіксований список (331 число).

(7) $p \equiv 1 \bmod ord(2,q)$, (тут $ord(2,q)$ буде дільником $q-1$).

(8) $p \equiv -1 \bmod \rho(q,P,Q)$, (тут $\rho(q,P,Q)$ буде дільником $q-1$).

(9) $p \equiv 1 \bmod ord(2,p)$, (тут ord(2, p) буде дільником $p-1$).

(10) $q \equiv 1 \bmod \rho(p,P,Q)$, (тут $\rho(p,P,Q)$) буде дільником $p+1$).

(11) $q > ord(2,p) \cdot \rho(p,P,Q)$.

(12) $(p-1)/ord(2,p) > \ln(p)/4 + 1/2$.

Нагадаємо (визначення 6), що через $L(p) = L(p, P, Q)$ позначено число LCM(ord(2,p), ρ(p, P, Q)).

Теорема 16. Нехай n - BPSW-псевдопростий, $n = p_1 \cdots p_k q$, де p_i – прості та $J(D, p_i) = -1$ при $i = 1, \ldots, k$ а q – довільне. Позначимо через S_i добуток всіх p_j, крім p_i:

$$S_i = p_1 \ldots p_k / p_i = n/(q\, p_i).$$

Тоді:

а) $qS_i \equiv 1 \bmod L(p_i, P, Q)$ $i = 1, \ldots k$;

б) нехай d_{ij} - найбільший спільний дільник $d_{ij} = \gcd(L(p_i), L(p_j))$ довільних i, j. Тоді $(p_i - p_j)S_{ij}$ ділиться на d_{ij}, де $S_{ij} = S_i/p_j = S_j/p_i$ – добуток всіх p_m, крім p_i та p_j.

Наслідок 5. Нехай n – BPSW-псевдопростий, $n = p_1p_2q$, де p_i – прості та $J(D, p_i) = -1$ при $i = 1, 2$ а q – довільне. Тоді:

а1) $qp_2 \equiv 1 \bmod L(p_1)$,

а2) $qp_1 \equiv 1 \bmod L(p_2)$,

б) $p_1 - p_2$ ділиться на $d_{12} = \gcd(L(p_1), L(p_2))$.

Нехай n – BPSW-псевдопростий, $n = p_1p_2q$, де p_i – прості та $J(D, p_i) = -1$ при $i = 1, 2$ а q – довільне. Тоді $p_1 - p_2$ ділиться на $\gcd(\mathrm{ord}(2, p_1), \mathrm{ord}(2, p_2))$.

3 РЕАЛІЗАЦІЯ ЗАПРОПОНОВАНИХ РІШЕНЬ

3.1 Характеристики алгоритму BPSW

У цьому розділі розглядаються лише числа n, менші 2^{64}, BPSWпсевдопрості з параметром D, за модулем меншим 128. Усі обчислення проводилися на процесорі AMD Athlon II X2240, 2.8 GHz.

Відповідно до теореми (5), параметр D або простий за модулем, або дорівнює −15.

Алгоритм перевірки такий.

Крок 1. Немає BPSW - псевдопростих чисел виду $n = pq$, де p - просте, і $q < 2^{19}$.

Перевірка за допомогою запропонованого математичного положення. Перевіряти можна на C++ за допомогою GMP. Час – близько доби.

Крок 2. Немає BPSW-псевдопростих чисел виду $n = pq$, де p - просте, $J(D / p) = +1$ і $p < 2^{34}$.

Числа p, менші за 2^{34} (17.17 млрд), що задовольняють умовам теореми (10) знайдено при |D| < 128 та наведені у файлі GCD_le_2bb.txt.

За простих D і D = −15 їх виявилося 170. Ці 170 чисел треба перевіряти окремо.

Крок 3. Немає BPSW-псевдопростих чисел виду $n = pq$, де p, q – прості, $J(D/p) = -1$, $J(D/q) = +1$, та $2^{19} < p < 2^{30}$.

Перевірка за допомогою запропонованих теоретичних рішень. Для кожного простого p із зазначеного інтервалу перебираємо всі D, для кожної пари (p, D) знаходимо $L = LCM(ord(2, p), \rho\, p)\, P, Q))$ і перебираємо всі q, що задовольняють порівнянню $q \equiv 1 \mod L(p, P, Q)$ та менші $2^{64}/p$.

При великих p величина L зазвичай виявляється великою і кількість допустимих або взагалі не буде, або їх буде мало. Тому перебір виявляється не надто великим. Час – близько двох діб. Після цих кроків можна дійти висновку,

що немає BPSW псевдопростbх чисел, менших 2^{64}, є добутком двох простих чисел.

Тести на простоту - вид алгоритмів, метою яких є визначення простоти заданого числа. Серед тестів розрізняють два підвиди: справжні та імовірнісні тести.

Справжні випробування дозволяють точно визначити простоту числа. Однак вони мають ряд суттєвих недоліків, що обмежують їхнє застосування. Серед них: умовність – тест працює лише для заздалегідь визначених чисел (тести для чисел Мерсенна, Ферма, Прота), мала швидкість обробки даних – поліноміальний час. Серед цих тестів у роботі були розглянуті та оптимізовані наступні тести:

- Тест Люка, що базується на однойменній теоремі, працює за поліномінальний час. У даному тесті визначається простота числа $2^p - 1$ за поліноміальний час від бітової довжини числа p. Саме завдяки цьому тесту числа Мерсенна майже завжди були самими великими знайдених простих чисел.

- Тест Адлемана - Померанса - Румелі - універсальний тест, що працює за $O(\log n^{c*\log\log n})$. Даний алгоритм може бути застосований на практиці у зв'язку з повільнозростаючою складністю.

- Тест Аггравала - Каяла - Саксени - універсальний тест, заснований на узагальненій теоремі Ферма, що працює за $O((\log n)^6)$. Головна особливість даного алгоритму полягає у тому, що він поліноміальний, універсальний, детермінований та безумовний на відміну від усіх попередніх алгоритмів [8].

- Тести за допомогою еліптичних кривих. Дані тести є одними з найбільш швидких із відомих методів перевірки числа на простоту. У гіршому випадку складність такого тесту становитиме $O((\log n)^{5+\Sigma})$. І для деяких випадків цей показник може бути покращено до $4+\sum$.

У свою чергу результатом імовірнісних тестів є лише ймовірність простоти числа. Головною перевагою таких тестів над дійсними є їх швидкість обробки даних (наприклад алгоритм Міллера - Рабіна зі

складністю $O((\log n)^3)$ [9]. Однак через ймовірність результатом відповіддю можуть стати так звані псевдопрості числа - числа, які пройшли тест на простоту, але не є. Для зменшення частоти появи подібних чисел використовуються раунди - повторні перевірки числа цим же тестом. Прикладом псевдопростих чисел є числа Кармайкла. У роботі [10] було розглянуто такі ймовірні тести:

- Тест Ферма, заснований на однойменній теоремі, що працює за $O(k \cdot \log n \cdot \log n))$.

- Тест Міллера — Рабіна, який використовує теорему Рабіна про свідків простоти, мінімальною швидкістю роботи $O(k \cdot (\log n)^3)$.

- Тест Соловея — Штрассена, заснований на символі Якобі, зі швидкістю роботи $O((\log n)^3)$.

Проаналізуємо універсальний оптимізаційний алгоритм, отриманий у процес аналізу специфіки кожного тесту для певних числових даних. Даний алгоритм аналогічний тесту BPSW, основу якого лежить використання кількох типів тестів. На рисунку 3.1 наведено графік залежності швидкості роботи тесту BPSW від розміру простого числа.

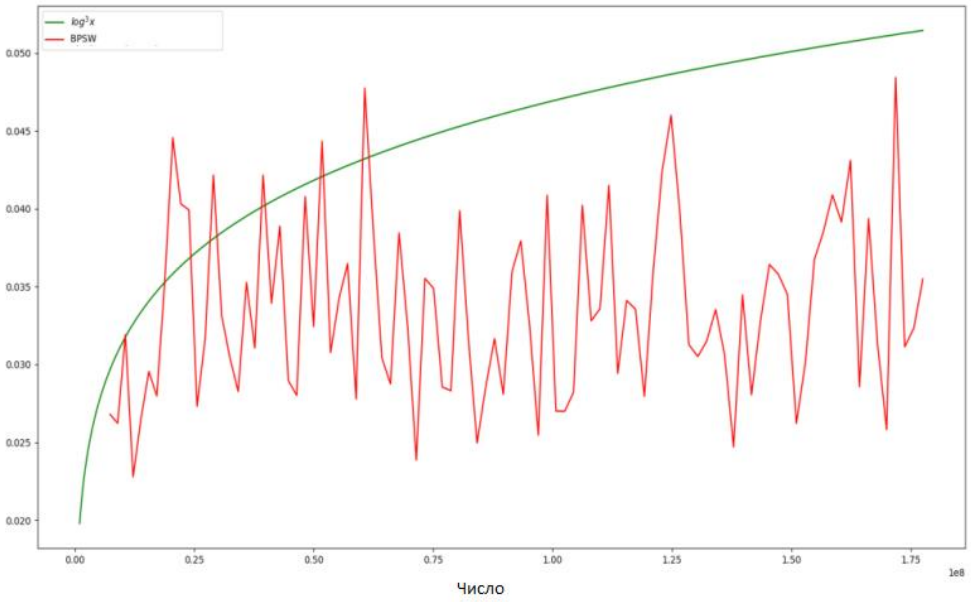

Рисунок 3.1 - Графік залежності швидкості роботи алгоритму BPSW від розміру простого числа

Приведені експериментальні дослідження дозволяють апроксимувати залежності та оцінити швидкодію запропонованого рішення. Запропонований алгоритм на основі BPSW тесту приведений на рисунку 3.2.

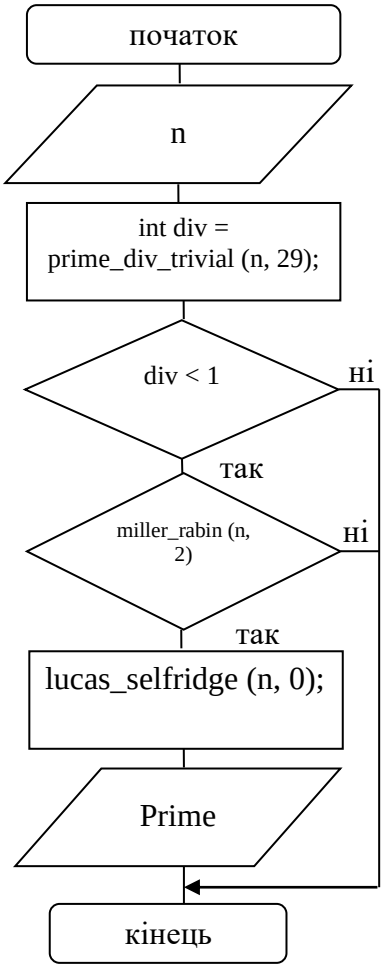

Рисунок 3.2 – Алгоритм визначення простити числа на основі BPSW тесту простоти

Дана методика дозволяє досягти зменшення обчислювальної складності. Для чисел розрядністю 64 біти обчислювальна складність складатиме $O(\log n)$, проте якщо використовувати довгу арифметику обчислювальна складність складатиме $O(\log^3(n))$.

3.2 Розробка програмних модулів

Для роботи з цілими числами довільного розміру в мові **C++** є бібліотека BigInteger . Серед методів цього класу, крім стандартних арифметичних операцій, виділимо такі:

- modInvers - знаходження зворотного за модулем.
- modPow - зведення в ступінь за модулем.
- isProbablePrime – перевірка числа на простоту.

Для реалізації програмного засобу було обрано середовище Rad Studio та мову C++, оскільки вони дозволяють ефективно реалізовувати математичні перетворення, та швидко реалізовувати додатки для ОС Windows.

На рисунку 3.3 приведено головне вікно середовища розробки.

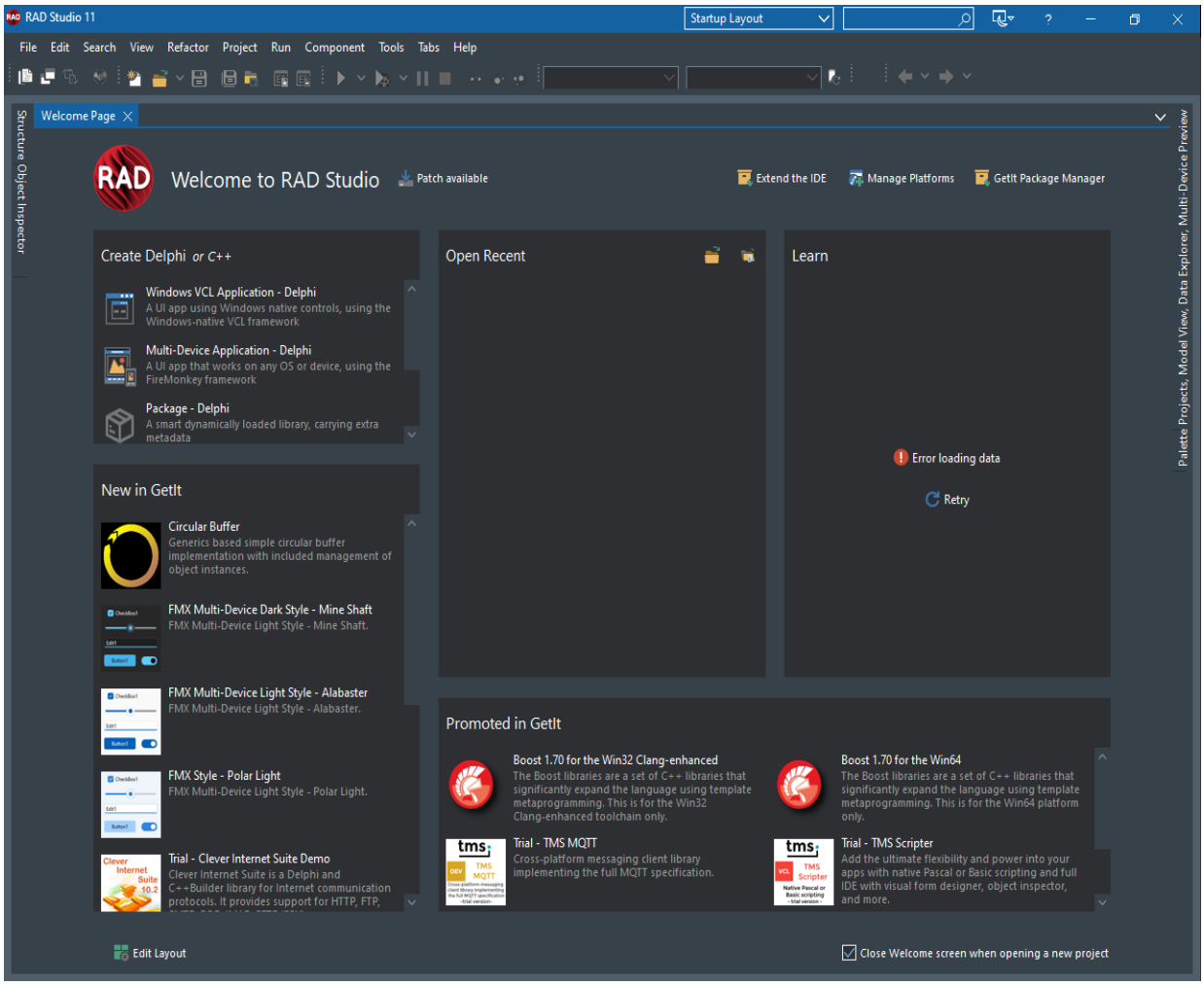

Рисунок 3.3 – Головне вікно середовища розробки

Для тестування запропонованого удосконалення на основі тесту BPSW створено інтерфейс користувача на базі однієї форми. Форма складається з трьох вкладок, на кожній з яких можна протестувати роботу удосконаленого та складових алгоритмів. Екранна форма приведена на рисунку 3.4.

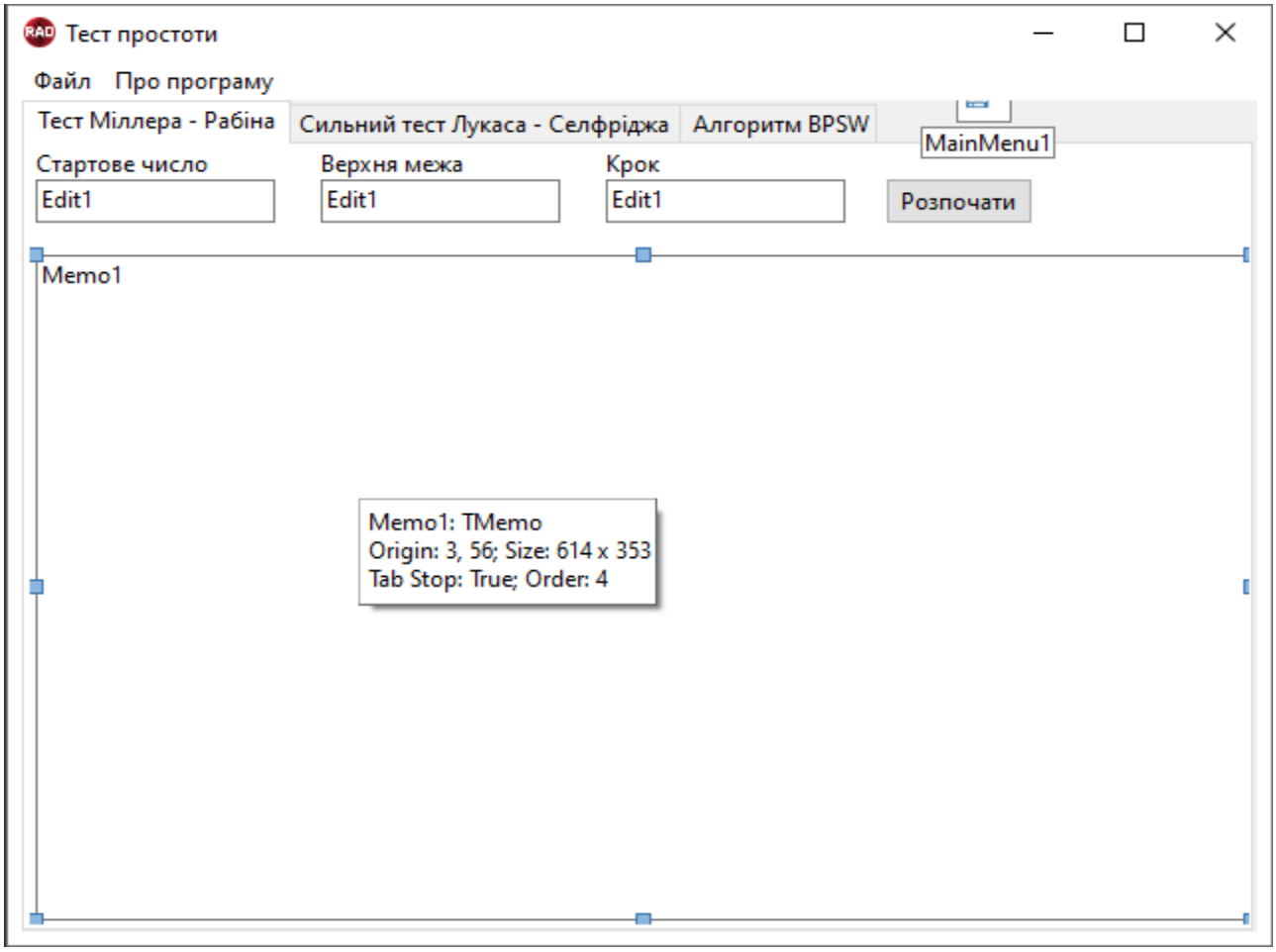

Рисунок 3.4 – Екранна форма для тестування алгоритму

Для роботи з класом BigInteger не потрібно ніяких додаткових бібліотек, що є істотним фактором при виборі інструментів. Наведемо час виконання ключової операції modPow, на процесорі Intel i5 із тактовою частотою 2.27 ГГц (Win-64) залежно від розрядності чисел (таблиця 3.1).

В бібліотеці є готова функціє що виконує перевірку на простоту. Вона працює в такий спосіб. Якщо число, що перевіряється, має довжину не більше 100 бітів, то виконується certainty/2 (але не більше 50) тестів Міллера-Рабіна з випадкової основи a .

Таблиця 3.1 – Час операції modPow

Довжина, біт	Час, мс
64	0.008
128	0.034
256	0.14
512	0.75
1024	4.7
2048	35

Якщо число має довжину більше 100 біт, то перевірка виконується у два кроки. На першому знову ж таки виконується certainty/2 тестів Міллера-Рабіна з випадкової основи a , але їх гранична кількість залежить від довжини числа в бітах, як показано на рисунку 3.5.

```
58      if (sizeInBits < 256) rounds = 27;
59      else if (sizeInBits < 512) rounds = 15;
60      else if (sizeInBits < 768) rounds = 8:
61      else if (sizeInBits < 1024)rounds = 4;
62      else rounds = 2;
63
```

Рисунок 3.5 – Код перевірки

Після цих перевірок, на другому кроці виконується ще й перевірка методом Люка-Лемера, як показано на рисунку 3.6.

```
67      bool passesLucasLehmer();
```

Рисунок 3.6 – Перевірка Люка-Лемера.

Створимо шаблон функції для BPSW тесту простоти. Спочатку перевіримо числа на прості дільники потім виконаємо алгоритм покроково(рисунок 3.7).

```
template <class T>
bool baillie_pomerance_selfridge_wagstaff (T n)
{


    // спочатку перевіряємо на тривіальні дільники -наприклад, до 29
    int div = prime_div_trivial (n, 29);
    if (div == 1)
        return true;
    if (div > 1)
        return false;

    // тест Міллера-Рабіна по основі 2
    if (!miller_rabin (n, 2))
        return false;

    // сильний тест Лукаса-Селфріджа
    return lucas_selfridge (n, 0);

}
```

Рисунок 3.7 – Шаблон функції для тесту

Для роботи алгоритму розроблено додаткові функції, наприклад обчислення найменшого спільного дільника та піднесення до степеня за залишком(рисунок 3.8).

```
const int trivial_limit = 50;
int p[1000];

int gcd (int a, int b) {
    return a ? gcd (b%a, a) : b;
}

int powmod (int a, int b, int m) {
    int res = 1;
    while (b)
        if (b & 1)
            res = (res * 1ll * a) % m,  --b;
        else
            a = (a * 1ll * a) % m,  b >>= 1;
    return res;
}
```

Рисунок 3.8 – Розроблені функції gcd та powmod

Для запропонованого тесту простоти створено функцію для виконання перевірки Міллера Рабіна, яка приведена на рисунку 3.9.

```
bool miller_rabin (int n) {
    int b = 2;
    for (int g; (g = gcd (n, b)) != 1; ++b)
        if (n > g)
            return false;
    int p=0, q=n-1;
    while ((q & 1) == 0)
        ++p,  q >>= 1;
    int rem = powmod (b, q, n);
    if (rem == 1 || rem == n-1)
        return true;
    for (int i=1; i<p; ++i) {
        rem = (rem * 1ll * rem) % n;
        if (rem == n-1)  return true;
    }
    return false;
}
```

Рисунок 3.9 – Функція для перевірки числа за допомогою тесту Міллера Рабіна

В тестах простоти використовуються символи Якобі, за допомогою яких визначається квадратичність залишку. Для цих цілей створена функція, що приведена на рисунку 3.10.

```
int jacobi (int a, int b)
{
    if (a == 0)  return 0;
    if (a == 1)  return 1;
    if (a < 0)
        if ((b & 2) == 0)
            return jacobi (-a, b);
        else
            return - jacobi (-a, b);
    int a1=a,  e=0;
    while ((a1 & 1) == 0)
        a1 >>= 1,  ++e;
    int s;
    if ((e & 1) == 0 || (b & 7) == 1 || (b & 7) == 7)
        s = 1;
    else
        s = -1;
    if ((b & 3) == 3 && (a1 & 3) == 3)
        s = -s;
    if (a1 == 1)
        return s;
    return s * jacobi (b % a1, a1);
}
```

Рисунок 3.10 – Функція обчислення символів Якобі

Створено також спеціальну функцію для перевірки числа модифікованим алгоритмом BPSW(рисунок 3.11). Функція виконанна для стандартного типу int що дозволяє оцінити її швидкодію, оскільки використання бібліотек довгої арифметики має додаткову обчислювальну складність.

```
bool bpsw (int n) {
    if ((int)sqrt(n+0.0) * (int)sqrt(n+0.0) == n)  return false;
    int dd=5;
    for (;;) {
        int g = gcd (n, abs(dd));
        if (1<g && g<n)  return false;
        if (jacobi (dd, n) == -1)  break;
        dd = dd<0 ? -dd+2 : -dd-2;
    }
    int p=1,  q=(p*p-dd)/4;
    int d=n+1,  s=0;
    while ((d & 1) == 0)
        ++s,  d>>=1;
    long long u=1, v=p, u2m=1, v2m=p, qm=q, qm2=q*2, qkd=q;
    for (int mask=2; mask<=d; mask<<=1) {
        u2m = (u2m * v2m) % n;
        v2m = (v2m * v2m) % n;
        while (v2m < qm2)   v2m += n;
        v2m -= qm2;
        qm = (qm * qm) % n;
        qm2 = qm * 2;
        if (d & mask) {
            long long t1 = (u2m * v) % n,  t2 = (v2m * u) % n,
                t3 = (v2m * v) % n,  t4 = (((u2m * u) % n) * dd) % n;
            u = t1 + t2;
            if (u & 1)  u += n;
            u = (u >> 1) % n;
            v = t3 + t4;
            if (v & 1)  v += n;
            v = (v >> 1) % n;
            qkd = (qkd * qm) % n;
        }
    }
    if (u==0 || v==0)  return true;
    long long qkd2 = qkd*2;
    for (int r=1; r<s; ++r) {
        v = (v * v) % n - qkd2;
        if (v < 0)  v += n;
        if (v < 0)  v += n;
        if (v >= n)  v -= n;
        if (v >= n)  v -= n;
        if (v == 0)  return true;
        if (r < s-1) {
            qkd = (qkd * 1ll * qkd) % n;
            qkd2 = qkd * 2;
        }
    }
    return false;
}
```

Рисунок 3.11 – Код функції модифікованого алгоритму BPSW

Також для перевірки на тривіальні дільники створено спеціальну функцію, що приведена на рисунку 3.12

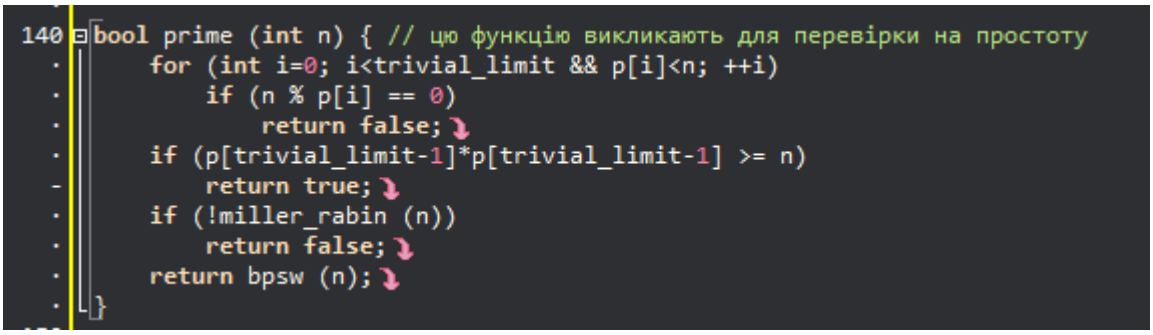

Рисунок 3.12 – Функція для перевірки на тривіальні дільники

Перед запуском функції для перевірки на тривіальні дільники потрібно створити початковий список простих чисел, які будуть використані в подальших перевірках. Для цього створено функцію, що приведена на рисунку 3.13.

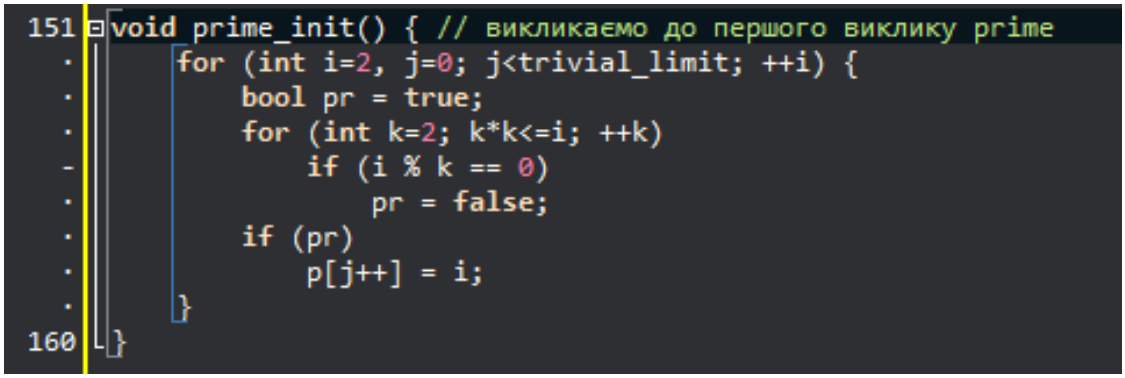

Рисунок 3.13 – Функція створення списку простих дільників

В результаті експериментів встановлено що в бібліотеці неможливо отримати помилку в методі isProbablePrime класу BigInteger для чисел довше 100 бітів за жодного рівня надійності більшого за 0.

3.3 Результати тестування програмного засобу

Після розробки алгоритму тесту простоти та проектування програмного засобу було проведено тестування розроблених теоретичних положень. На рисунку 3.14 приведена головна форма додатку з переліком простих чисел обчислених з заданого діапазону.

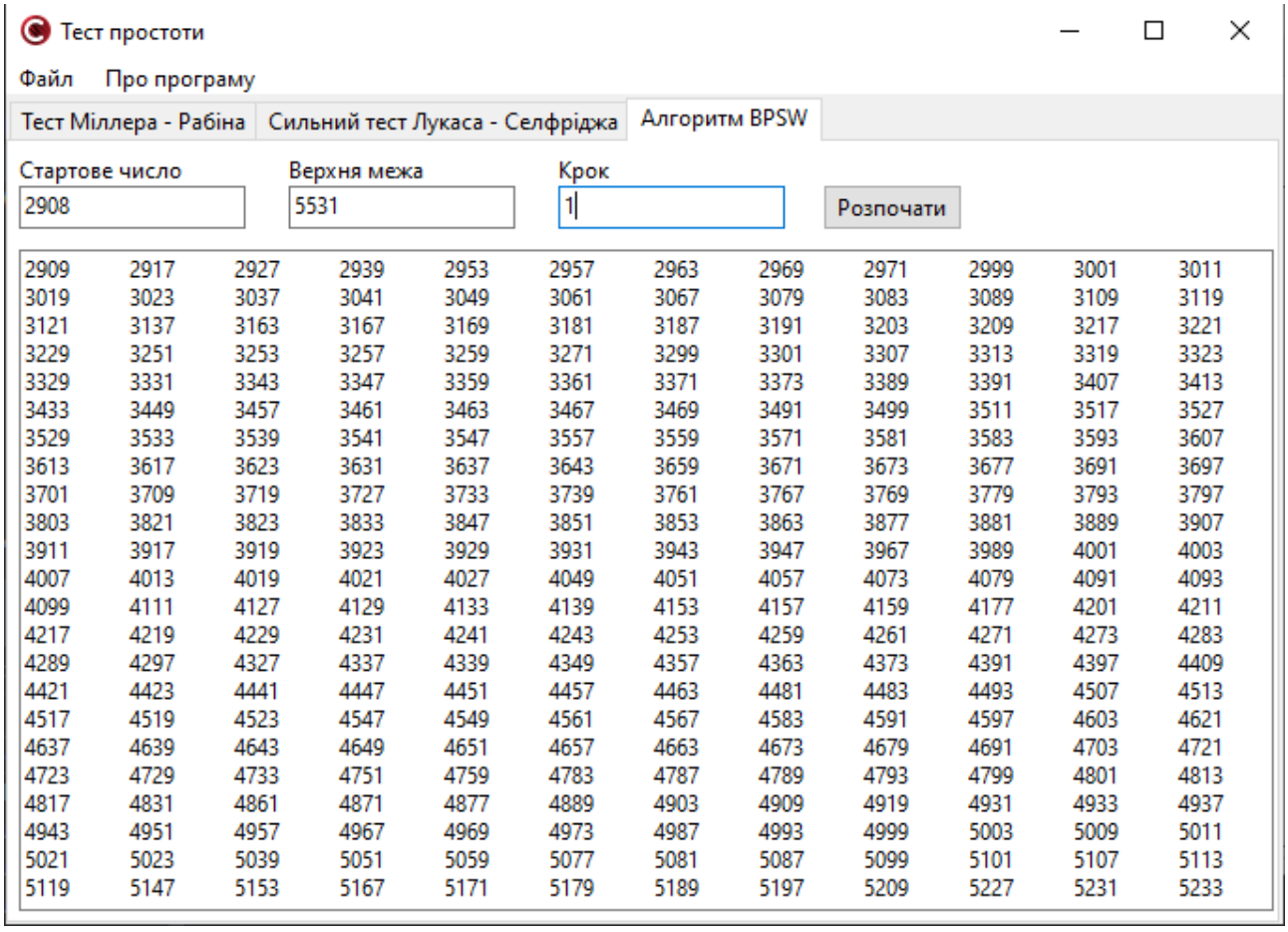

Рисунок 3.14 – Розроблений додаток тестування удосконаленого алгоритму BPSW

Розглянемо результати, отримані в результаті тестування реалізації тесту BPSW. Всі випробування проводилися на вбудованому типі – 64-бітному числі long long. Довга арифметика не тестувалася.

Тестування проводили на комп'ютері з процесором i5-7440HQ 2.8 GHz. Результати буде представлено у мікросекундах (10^{-6} сек). Середній час роботи на відрізку чисел залежно від межі тривіального перебору.

Мається на увазі параметр, що передається функції prime_div_trivial(), який описаний кодом вище.

Якщо запускати тест на всіх непарних числах із відрізка, то результати виходять такими, як показано в таблиці 3.2.

Таблиця 3.2 – Результати експерименту

Початок відрізка	Кінець відрізка	0	10^2	10^3	10^4	10^5
1	10^5	7.01	3.2	0.2	0.3	0.5
10^6	$10^6 + 10^5$	11.7	5.7	4.0	0.8	1.2
10^9	$10^9 + 10^5$	26.3	10.1	11.2	15.0	16.1
10^{12}	$10^{12} + 10^5$	39.3	14.1	13.1	17.4	53.2
10^{15}	$10^{15} + 10^5$	65.2	22.4	17.1	22.6	55.7

Якщо запускати тест тільки на простих числах із відрізка, то швидкість роботи буде такою як наведено в таблиці 3.3.

Таблиця 3.3 – Результати експерименту на простих числах

Початок відрізка	Кінець відрізка	0	10^2	10^3	10^4	10^5
1	10^5	42.9	40.8	3.1	4.2	4.2
10^6	$10^6 + 10^5$	75.0	76.4	88.8	13.9	15.2
10^9	$10^9 + 10^5$	186.5	188.5	201.0	294.3	283.9
10^{12}	$10^{12} + 10^5$	288.3	288.3	302.2	387.9	1069.5
10^{15}	$10^{15} + 10^5$	485.6	489.1	496.3	585.4	1267.4

Отже, оптимально вибирати межу тривіального перебору рівним 100 чи 1000 . Для всіх наступних тестів було обрано межу 1000. Середній час роботи на відрізку чисел оцінюється наступним чином.

Тепер, коли ми вибрали межу тривіального перебору, можна точніше протестувати швидкість роботи на різних відрізках чисел(таблиця 3.4).

Таблиця 3.4 – Результати експерименту на обраному відрізку

Початок відрізка	Кінець відрізка	Час роботи на непарних числах	Час роботи на простих числах
1	10^5	1.2	4.2
10^6	10^6+10^5	13.8	88.8
10^7	10^7+10^5	16.8	115.5
10^8	10^8+10^5	21.2	164.8
10^9	10^9+10^5	24.0	201.0
10^{10}	$10^{10}+10^5$	25.2	225.5
10^{11}	$10^{11}+10^5$	28.4	266.5
10^{12}	$10^{12}+10^5$	30.4	302.2
10^{13}	$10^{13}+10^5$	33.0	352.2
10^{14}	$10^{14}+10^5$	37.5	424.3
10^{15}	$10^{15}+10^5$	42.3	499.8
10^{16}	$10^{15}+10^5$	46.5	553.6
10^{17}	$10^{15}+10^5$	48.9	621.1

Або у вигляді графіка приблизний час роботи тесту BPSW на одному числі можна представити у ви, як це показано на рисунку 3.15.

Тобто ми отримали, що на практиці, на невеликих числах (до 10^{17}), алгоритм працює за $O(\log N)$. Це обумовлено тим, що з вбудованого типу int64 операція розподілу виконується за $O(1)$, тобто. складність поділу залежить від кількості бітів в числі.

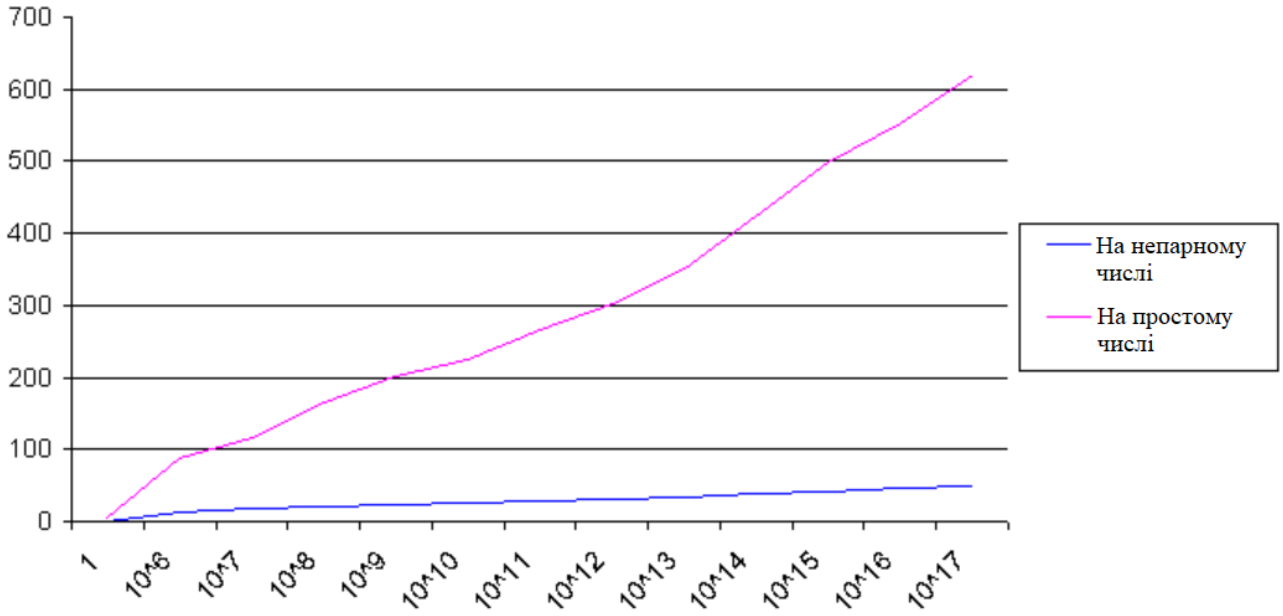

Рисунок 3.15 – Результати тесту на одному числі

Якщо ж застосувати тест BPSW до довгої арифметики, то очікується, що він працюватиме за $O(\log 3(N))$ [11].

ВИСНОВКИ

Досліджено математичні основи існуючих тестів простоти, що дало можливість виявити їхні переваги та недоліки для використання в системах захисту інформації.

Досліджено властивості простих та псевдопростих чисел, що дало можливість оцінити ефективність використання ймовірнісних тестів простоти та їхню обчислювальну складність.

Проведено дослідження властивостей символів Якобі та алгоритму їх обчислення в тестах простоти, як однієї з трудомістких операцій при побудові алгоритму перевірки на простоту.

Проаналізовано математичні основи BPSW тесту простоти та його елементів для виявлення слабких та сильних сторін складових алгоритму.

Удосконалено тест простоти на основі алгоритму BPSW, що дало можливість знизити обчислювальну складність алгоритму та побудувати програмний засіб для перевірки чисел на простоту.

Реалізовано та протестовано програмний засіб для тестування чисел на простоту мовою C++ за допомогою розробленого алгоритму на основі BPSW тесту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ігнатєв І.В., Кондратюк В.М., Алгоритм перевірки числа на простоту. Збірник матеріалів проблемно-наукової міжгалузевої конференції «Автоматизація та комп'ютерно – інтегровані технології» (АКІТ -2022), Тернопіль, 2022. 70-73 с.

2. Посвятовська О.Б., Стефурак Н.А., Кондратюк В.М., Дослідження властивостей простих чисел для BPSW. Збірник матеріалів науково-практичної конференції молодих вчених, аспірантів та студентів» (КБКІТ-2020), Тернопіль, 2022, С. 73-79.

3. Ivasiev S., Yakymenko I., Kasianchuk M., Shevchuk R., Tymoshenko L. The Method of Factorizing Multi-Digit Numbers Based on the Operation of Adding Odd Numbers. Advanced Computer Information Technology (ACIT–2018): Proceedings of the International Conference. Ceske Budejovice (Czech Republic). 2018. P. 232-235.

4. Persson A., Bengtsson L. Forward and reverse converters and moduli set selection in signed-digit Residue Number Systems. J. Signal Process. Syst. 2009. Vol. 56 (1). P. 1–15.

5. Краснобаев В.А., Янко А.С., Кошман С.А., Курчанов В.Н., Бендес Ю.П. Расчет и сравнительный анализ производительности компьютерной системы обработки целочисленных данных, представленных в системе остаточных классов. Системи обробки інформації. 2015. В 3 (128). С. 57-61.

6. Касянчук М., Карпінський М., Казмірчук С. Методологія опрацювання багаторозрядних чисел в асиметричних криптосистемах/ Захист інформації. 2019. Т.21, №2. С. 65- 73.

7. Касянчук М.М., Якименко І.З., Івасьєв С.В., Момотюк О.В. Експериментальне дослідження програмної реалізації методів пошуку оберненого елемента за модулем. Інформатика та математичні методи в моделюванні. 2017. Т.7, №3. С. 178–186.

8. Persson A., Bengtsson L. Forward and reverse converters and moduli set selection in signed-digit Residue Number Systems. J. Signal Process. Syst. 2009. Vol. 56 (1). P. 1–15.

9. Краснобаев В.А., Янко А.С., Кошман С.А., Курчанов В.Н., Бендес Ю.П. Расчет и сравнительный анализ производительности компьютерной системы обработки целочисленных данных, представленных в системе остаточных классов. Системи обробки інформації. 2015. В 3 (128). С. 57-61.

10. Zadiraka V., Yakymenko I., Kasianchuk M., Ivasiev S. Theoretical and numerical Krestenson's basis and its application to problems of cryptographic protection and factorization of multidigit numbers, Computer technologies in information security: collective monograph, By edited V.Zadiraka, Ya.Nykolaichuk, Ternopil: Kart-blansh, 2015. P. 216-260. Ch. 5.

11. Ivasiev S., Yakymenko I., Kasianchuk M., Shevchuk R., Karpinski M., Gomotiuk O. Effective algorithms for finding the remainder of multi-digit numbers. Advanced Computer Information Technology (ACIT–2019): Proceedings of the International Conference. Ceske Budejovice (Czech Republic). 2019. P. 175-178

12. Ivasiev S., Yakymenko I., Kasianchuk M., Shevchuk R., Tymoshenko L. The Method of Factorizing Multi-Digit Numbers Based on the Operation of Adding Odd Numbers. Advanced Computer Information Technology (ACIT–2018): Proceedings of the International Conference. Ceske Budejovice (Czech Republic). 2018. P. 232-235.

13. Rajba T. Klos-Witkowska A., Ivasiev S., Yakymenko I., Kasianchuk M. Research of Time Characteristics of Search Methods of Inverse Element by the Module. Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications: Proceedings of the 2017 IEEE 9 th International Conference. Bucharest, Romania. V.1. September, 2017. P.82– 85 (Scopus).

14. Касянчук М., Карпінський М., Казмірчук С. Методологія опрацювання багаторозрядних чисел в асиметричних криптосистемах/ Захист інформації. 2019. Т.21, №2. С. 65- 73.

15. Касянчук М.М., Якименко І.З., Івасьєв С.В., Момотюк О.В. Експериментальне дослідження програмної реалізації методів пошуку

оберненого елемента за модулем. Інформатика та математичні методи в моделюванні. 2017. Т.7, №3. С. 178–186.

16. Івасьєв С.В., Лісковецький Д.В., Шпак В.Б. Метод збереження простих великорозрядних чисел у базисі Радемахера**.** Збірник матеріалів проблемно-наукової міжгалузевої конференції «Автоматизація та комп'ютерно-інтегровані технології» (АКІТ - 2019), Тернопіль, 2019. С. 156-160.

17. Глинська М.Л., Лісковецький Д.В., Івасьєв С.В. Збірник матеріалів наукової конференції «Кібербезпека та комп'ютерноінтегровані технології» (КБКІТ - 2019). –Тернопіль. –2019. С. 21-24.

18. Ачасова С. М., Бандман О. Л. Корректность параллельных вычислительных процессов. Новосибирск: Наука. Сиб. отд-ние, 1990. – 253 с.

19. Factorization of a 768-bit RSA Modulus / Th. Kleinjung [et al.] // Advances in Cryptology — CRYPTO 2010 : 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15—19, 2010. — Santa Barbara, 2010. — P. 333—350.

20. Бараш, Л. Ю. Алгоритм AKS проверки чисел на простоту и поиск констант генераторов псевдослучайных чисел / Л. Ю. Бараш // Безопасность информационных технологий. — 2005. — №2. — С. 27—38.

21. Шнайер, Б. Прикладная криптография / Б. Шнайер ; пер. с англ. — М. : Триумф, 2013. — 816 с.

22. Вербіцький О.В. Вступ до криптології. – Львів: ВНТЛ, 1998 г. – 248 с.

23. Гулямов С.С., Зайнидинов Х.Н. Синтез параллельно-конвейерных вычислительных структур для выполнения быстрых преобразований Хаара. Методы и модели систем обработки. Сб. научн. трудов НПО "Кибернетика" АН РУЗ. Ташкент. -1994. - С.124-127.

24. Карацуба А.А. Сложность вычислений. Тр.МИАМ, 1995.Т.211. С.186- 202.

25. Касянчук М. М., Николайчук Я. М., Якименко І. З. Теорія алгоритмів перетворень китайської теореми про залишки в матрично-розмежованому базисі Радемахера-Крестенсона. Вісник національного університету «Львівська політехніка». 2011. № 688. С. 118–124.

26. Айерлэнд К., Роузен М. Классическое введение в современную теорию чисел, М.: Мир.– 1987- 416 с.

27. Акушский И.Я., Юдицкий Д.И., Машинная арифметика в остаточных классах. М: Сов.радио, 1968. – 440 с.

28. Касянчук М.М. Якименко І.З., Волинський О.І., Івасьєв С.В. Теоретичні основи аналітики та алгоритми оптимізації обчислень простих чисел. Поступ в науку. Збірник наукових праць Бучацького інституту менеджменту і аудиту. Матеріали Міжнародної проблемно-наукової міжгалузевої конференції «Інформаційні проблеми комп'ютерних систем, юриспруденції, енергетики, економіки, моделювання та управління (ПНМК-2010)». В.6, т.1.- с.33-36.

29. Касянчук М.М. Теорія та математичні закономірності досконалої форми системи залишкових класів. Праці Міжнародного симпозіуму „Питання оптимізації обчислень (ПОО–XXXV)". Т.1. Київ–Кацивелі. 2009.– С. 306–310.

30. Николайчук Я.М., Волинський О.І., Кулина С.В. Теоретичні основи побудови спецпроцесорів у базисі Крестенсона. Вісник Хмельницького національного університету. 2007. № 3. Т.І(93). С. 85-90.

31. Стенли Б. Липпман. С++ для начинающих, Пер. с англ. 2тт. Москва: Унитех. Рязань: Гэлион, 1992, 304-345 с.

32. Николайчук Я.М. Теорія джерел інформації. Тернопіль: ТзОВ„Терно–граф", 2010. – 536 с.

33. Якименко І.З., Касянчук М.М., Тимошенко Л.М., Івасьєв С.В., Николайчук Я.М. Алгоритм знаходження системи модулів модифікованої досконалої форми системи залишкових класів, Матеріали МНПК СІЕТ.- Одеса. 2014. С. 115-117.

34. Kozaczko D., Kasianchuk M., Yakymenko I., Ivasiev S.Vector Module Exponential in the Remaining Classes System. Proceedings of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS–2015). Warsaw, Poland. V.1, September, 2015. P.161–163.

35. Lakhani G. Some Fast Residual Arithmetic Adders. International Journal of

Electronics. - 1994. - P. 225-240.

36. Lenstra H. W. Divisors in residue classes. Math. Comput. 1984. Vol. 42. N 165. P.331-340.

37. Nykolaychuk Ya. M., Kasianchuk M.M., Yakymenko I.Z., Theoretical Foundations of the Modified Perfect form of Residue Number System. Cybernetics and Systems Analysis, 2016, P. 219-223.

38. Tsmots I., Teslyuk V., Teslyuk T., Ihnatyev I. Basic Components of Neuronetworks with Parallel Vertical Group Data Real-Time Processing. Advances in Intelligent Systems and Computing II. CSIT 2017. Advances in Intelligent Systems and Computing, vol. 689, Springer, Cham. P. 558 – 576.

39. Yakymenko I., Kasyanchuk M., Nykolajchuk Ya. Matrix Algorithms of Processing of the Information Flow in Computer Systems Based on Theoretical and Numerical Krestenson's Basis. Proceedings of the X–th International Conference "Modern Problems of Radio Engineering, Telecommunications and Computer Science" (TCSET–2010). L'viv–Slavske. 2010. – P.241.

ДОДАТОК А

Код програмного засобу

```
//! Модуль 64-бітного числа
long long abs (long long n);
unsigned long long abs (unsigned long long n);
//! Повертає true, якщо n парне
template <class T>
bool even (const T&n);
//! Поділяє число на 2
template <class T>
void bisect (T&n);
//! Помножує число на 2
template <class T>
void redouble (T&n);
//! Повертає true, якщо n – точний квадрат простого числа
template <class T>
bool perfect_square (const T&n);
//! Обчислює корінь із числа, округляючи його вниз
template <class T>
T sq_root (const T&n);
//! Повертає кількість біт у числі
template <class T>
unsigned bits_in_number (T n);
//! Повертає значення k-го біта числа (біти нумеруються з нуля)
template <class T>
bool test_bit (const T&n, unsigned k);
//! Помножує a * = b (mod n)
template <class T>
void mulmod (T&a, Tb, const T&n);
//! Обчислює a^k (mod n)
template <class T, class T2>
T powmod (T a, T2 k, const T&n);
//! Переводить число n у форму q*2^p
template <class T>
void transform_num (Tn, T&p, T&q);
//! Алгоритм Евкліда
template <class T, class T2>
T gcd (const T&a, const T2&b);
//! Обчислює jacobi(a,b) - символ Якобі
template <class T>
T jacobi (T a, T b)
//! Обчислює pi(b) перших простих чисел. Повертає вектор з простими і в pi - pi(b)
```

```
template <class T, class T2>
const std::vector & get_primes (const T&b, T2&pi);
//! Тривіальна перевірка n простоту, перебираються всі дільники до m.
//! Результат: 1 - якщо n точно просте, p - його знайдений дільник; 0 - якщо невідомо
template <class T, class T2>
T2 prime_div_trivial (const T&n, T2 m);
template <class T, class T2>
bool miller_rabin (Tn, T2b)
{
// Спершу перевіряємо тривіальні випадки
if (n == 2)
return true;
if (n < 2 | | even (n))
return false;
// перевіряємо, що n і b взаємно прості (інакше це призведе до помилки)
// якщо вони не взаємно прості, то або n не просто, або потрібно збільшити b
if (b < 2)
b = 2;
for (T g; (g = gcd (n, b))! = 1; ++b)
if (n > g)
return false;
// Розкладаємо n-1 = q * 2 ^ p
T n_1 = n;
--n_1;
T p, q;
transform_num (n_1, p, q);
// обчислюємо b^q mod n, якщо воно дорівнює 1 або n-1, то n просте (або псевдопросте)
T rem = powmod (T(b), q, n);
if (rem == 1 || rem == n_1)
return true;
// Тепер обчислюємо b ^ 2q, b ^ 4q, ... , b ^ ((n-1) / 2)
// якщо якесь із них дорівнює n-1, то n просте (або псевдопросте)
for (T i=1; i<p; i++)
{
mulmod (rem, rem, n);
if (rem == n_1)
return true;
}
return false;
}
template <class T, class T2>
bool lucas_selfridge (const T&n, T2 unused)
{
// Спершу перевіряємо тривіальні випадки
```

```
if (n == 2)
return true;
if (n < 2 | | even (n))
return false;
// Перевіряємо, що n не є точним квадратом, інакше алгоритм дасть помилку
if (perfect_square (n))
return false;
// Алгоритм Селфріджа: знаходимо перше число d таке, що:
// jacobi(d,n)=-1 і він належить ряду { 5,-7,9,-11,13,... }
T2 dd;
for (T2 d_abs = 5, d_sign = 1; ; d_sign = -d_sign, ++++d_abs)
{
dd = d_abs * d_sign;
T g = gcd (n, d_abs);
if (1 < g && g < n)
// знайшли дільник - d_abs
return false;
if (jacobi (T(dd), n) == -1)
break;
}
// Параметри Селфріджа
T2
p = 1,
q = (p * p - dd) / 4;
// Розкладаємо n+1 = d*2^s
T n_1 = n;
++n_1;
Ts, d;
transform_num (n_1, s, d);
// алгоритм Лукаса
T
u = 1,
v = p,
u2m = 1,
v2m = p,
qm = q,
qm2 = q*2,
qkd = q;
for (unsigned bit = 1, bits = bits_in_number(d); bit < bits; bit++)
{
mulmod (u2m, v2m, n);
mulmod (v2m, v2m, n);
while (v2m < qm2)
v2m + = n;
v2m -= qm2;
mulmod (qm, qm, n);
```

```
qm2 = qm;
redouble (qm2);
if (test_bit (d, bit))
{
T t1, t2;
t1 = u2m;
mulmod (t1, v, n);
t2 = v2m;
mulmod (t2, u, n);
T t3, t4;
t3 = v2m;
mulmod (t3, v, n);
t4 = u2m;
mulmod (t4, u, n);
mulmod (t4, (T)dd, n);
u = t1 + t2;
if (! even (u))
u += n;
bisect (u);
u %= n;
v = t3 + t4;
if (!even (v))
v + = n;
bisect (v);
v % = n;
mulmod (qkd, qm, n);
}
}
// точно просте (чи псевдо-просте)
if (u == 0 || v == 0)
return true;
// Довираховуємо члени, що залишилися
T qkd2 = qkd;
redouble (qkd2);
for (T2 r = 1; r < s; ++r)
{
mulmod (v, v, n);
v -= qkd2;
if (v < 0) v + = n;
if (v < 0) v + = n;
if (v >= n) v -= n;
if (v >= n) v -= n;
if (v == 0)
return true;
if (r < s-1)
{
```

```
mulmod (qkd, qkd, n);
qkd2 = qkd;
redouble (qkd2);
}
}
return false;
}
template <class T>
bool baillie_pomerance_selfridge_wagstaff (Tn)
{
// Спершу перевіряємо на тривіальні дільники - наприклад, до 29
int div = prime_div_trivial (n, 29);
if (div == 1)
return true;
if (div > 1)
return false;
// тест Міллера-Рабіна на підставі 2
if (!miller_rabin (n, 2))
return false;
// сильний тест Лукаса-Селфріджа
return lucas_selfridge (n, 0);
}
const int trivial_limit = 50;
int p[1000];
int gcd (int a, int b) {
return a? gcd (b%a, a): b;
}
int powmod (int a, int b, int m) {
int res = 1;
while (b)
if (b & 1)
res = (res * 1ll * a) % m, - b;
else
a = (a * 1ll * a) % m, b >> = 1;
return res;
}
bool miller_rabin (int n) {
int b = 2;
for (int g; (g = gcd (n, b))! = 1; ++b)
if (n > g)
return false;
int p=0, q=n-1;
while ((q & 1) == 0)
++p, q >>= 1;
int rem = powmod (b, q, n);
if (rem == 1 || rem == n-1)
```

```
return true;
for (int i=1; i<p; ++i) {
rem = (rem * 1ll * rem) % n;
if (rem == n-1) return true;
}
return false;
}
int jacobi (int a, int b)
{
if (a == 0) return 0;
if (a == 1) return 1;
if (a < 0)
if ((b & 2) == 0)
return jacobi (-a, b);
else
return - jacobi (-a, b);
int a1=a, e=0;
while ((a1 & 1) == 0)
a1 >>= 1, ++e;
int s;
if ((e & 1) == 0 || (b & 7) == 1 || (b & 7) == 7)
s = 1;
else
s = -1;
if ((b & 3) == 3 && (a1 & 3) == 3)
s = -s;
if (a1 == 1)
return s;
return s * jacobi (b% a1, a1);
}
bool bpsw (int n) {
if ((int)sqrt(n+0.0) * (int)sqrt(n+0.0) == n) return false;
int dd=5;
for(;;) {
int g = gcd (n, abs (dd));
if (1<g && g<n) return false;
if (jacobi (dd, n) == -1) break;
dd = dd <0? -dd+2: -dd-2;
}
int p=1, q=(p*p-dd)/4;
int d=n+1, s=0;
while ((d & 1) == 0)
++s, d>>=1;
long long u=1, v=p, u2m=1, v2m=p, qm=q, qm2=q*2, qkd=q;
for (int mask=2; mask<=d; mask<<=1) {
u2m = (u2m * v2m) % n;
```

```
v2m = (v2m * v2m) % n;
while (v2m <qm2) v2m += n;
v2m -= qm2;
qm = (qm * qm) % n;
qm2 = qm*2;
if (d & mask) {
long long t1 = (u2m * v) % n, t2 = (v2m * u) % n,
t3 = (v2m * v) % n, t4 = (((u2m * u) % n) * dd) % n;
u = t1 + t2;
if (u & 1) u += n;
u = (u >> 1) % n;
v = t3 + t4;
if (v&1) v += n;
v = (v >> 1) % n;
qkd = (qkd * qm) % n;
}
}
if (u==0 || v==0) return true;
long long qkd2 = qkd*2;
for (int r=1; r<s; ++r) {
v = (v * v) % n - qkd2;
if (v < 0) v + = n;
if (v < 0) v + = n;
if (v >= n) v -= n;
if (v >= n) v -= n;
if (v == 0) return true;
if (r < s-1) {
qkd = (qkd * 1ll * qkd) % n;
qkd2 = qkd * 2;
}
}
return false;
}
bool prime (int n) { // Цю функцію потрібно викликати для перевірки простоти
for (int i=0; i<trivial_limit && p[i]<n; ++i)
if (n% p [i] == 0)
return false;
if (p[trivial_limit-1]*p[trivial_limit-1] >= n)
return true;
if (!miller_rabin (n))
return false;
return bpsw (n);
}
void prime_init() { // викликати до першого виклику prime()!
for (int i=2, j=0; j<trivial_limit; ++i) {
bool pr = true;
```

```
for (int k=2; k*k<=i; ++k)
if (i % k == 0)
pr = false;
if (pr)
p[j++] = i;
}
}
```

ДОДАТОК Б

Світокопія публікацій

ЗАХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ
ГАЛИЦЬКИЙ ФАХОВИЙ КОЛЕДЖ ІМ. В. ЧОРНОВОЛА

КІБЕРБЕЗПЕКА ТА КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ (КБКІТ – 2022)

науково-практична конференція
молодих вчених, аспірантів та студентів

29–31 серпня 2022
Тернопіль

СЕКЦІЯ 3

КРИПТОГРАФІЧНІ МЕТОДИ ЗАХИСТУ ІНФОРМАЦІЇ

УДК 004.56.5(043.2)

Посвятовська О.Б.[1], Стефурак Н.А.[1], Кондратюк В.М.[1]

[1]Галицький фаховий коледж імені В'ячеслава Чорновола

[2]Тернопільський національний університет

ДОСЛІДЖЕННЯ ВЛАСТИВОСТЕЙ ПРОСТИХ ЧИСЕЛ ДЛЯ BPSW ТЕСТУ

Вступ. Тести простоти мають велике значення в комп'ютерній техніці та зокрема в криптографії. На основі використання простих чисел базуються більшість крипто перетворення для шифрування та крипто аналізу даних. Доводяться деякі властивості BPSW-псевдопростих чисел [1,3,4]. Як результат, доводиться відсутність BPSW-псевдопростих, менших за 2^{60}. Основними результатами є дослідження математичних основ та роботи запропонованих тверджень.

Мета роботи полягає у дослідженні властивостей простих чисел та тестів прости на яких базується BPSW тест простоти. Проведені дослідження BPSW псевдопростих чисел дозволять побудувати ефективний тест простоти.

1. Математичні основи тестів простоти

Для перевірки чисел на простоту дуже ефективним є такий метод [2,5,6,7]. Нехай n – непарне натуральне число, в якому хочемо перевірити простоту. Це пропонується робити наступним чином.

Крок 1. Тест Міллера - Рабіна на підставі 2.

Крок 2. Тест Лукаса - Сельфідж. Зазначимо, що якщо у тесті із послідовності 5, −7, 9, . . . вибрано параметр D, то фактично при цьому перевіряється, що число n взаємно-просте з усіма числами, не перевищують $|D|$.

Числа, які проходять ці два тести, називатимемо BPSW псевдопростими.

Для натурального n і цілих (P, Q) позначимо через

$R(n) = R(n, P, Q)$ функцію (тут $D = P^2 - 4 * Q$):

$$R(n,P,Q) = \begin{cases} 0, J(D,n) = 0 \\ \gcd(2^{n-1} - 1, U_{n-1}(P,Q)), J(D,n) = 1 \\ \gcd(2^{n-1} - 1, U_{n+1}(P,Q)), J(D,n) = -1 \end{cases}$$

Якщо p – просте і взаємно-просте з $PQ(1 - 4Q)$, то $R(p)$ ділиться на p.

Значення функції R не дуже великі і можуть бути обчислені при не надто великих n (≈ до 2^{18}).

Нехай n - BPSW-псевдопростий, $n = pq$, де p - просте. Тоді p – простий дільник $R(q)$.

Так як $2^{pq-1} = 2^{(p-1)q+(q-1)} \equiv 2^{q-1} \equiv 1 \bmod p$, тобто $2^{n-1} - 1$ ділиться на p.

Нехай тест Лукаса виконається з параметром $Q, D = 1\text{-}4Q$. По умові $J(D/q) = -1$. Нехай $J(D/q) = +1$. Тоді $J(D/q) = -1$. Так як:

$$U_{pq+1} = U_{(p-1)q+q+1} \equiv U_{q+1} \equiv 0 \bmod p,$$

так як U_{q+1} ділится на p. Таким чином, число p повинне ділити $2^{q-1}-1$ і U_{q+1}, тобто p- простий дільник $R(q)$.

Нехай $J(D/q)=-1$. Тоді $J(D/q)=+1$.

Так як: $U_{pq+1}=U_{(p+1)q-q+1}=U_{(p+1)-(q-1)}=0 \bmod p$.

Звідси слідує, що $U_{q-1}=0 \bmod p$, так як p простий дільник U_{q-1}, а значить, p знову буде дільником $R(q)$.

Ця теорема дозволяє ефективно перевірити на псевдопростоту числа виду pq, де q – довільне натуральне, яке не перевищує деякої межі, а p – просте. Наприклад, нап $q < 100\ 000$ і $|D| \le 21$ таких чисел не виявлено. За допомогою бібліотеки GMP можна перевірити всі числа в межах до 218 і навіть дещо більше.

Крім того, немає жодного простого числа q, меншого за 220 такого, що $J(5, q)=-1$ і у R-функції якого $R(q) = R(q, P, Q)$ був би простий дільник, більший q. Також можна перевірити і таке твердження. За $|D| < 128$ серед чисел $q < 218$, $J(D, q)=-1$ немає жодного, який мав би простий дільник числа $R(q)$ більший q.

Можна дещо уточнити. Нехай n - BPSW-псевдопростий, $n = pq$, де p - просте при деякому $D = 1 - 4Q$. Тоді: якщо $J(D/p) = +1$, то p – простий дільник одного із чотирьох чисел.

$$\gcd(2^k-1,U_k),\ \gcd(2^k-1,V_k),\ \gcd(2^k+1,U_k),\ \gcd(2^k+1,V_k);$$

якщо $J(D/p) = -1$, то p – простий дільник одного із чотирьох чисел $\gcd(2^k-1,U_{k+1})$, $\gcd(2^k-1,V_{k+1})$, $\gcd(2^k+1,U_{k+1})$, $\gcd(2^k+1,V_{k+1})$, де $k=(q-1)/2$.

Це слідує з того, що $2^{2k}-1=(2^k-1)(2^k+1)$ і $U_{2k}=U_kV_k$.

При конкретних обчисленнях ця теорема зручніша, ніж (7), оскільки має справу з числами вдвічі меншою за довжину.

Простий множник p з $J(D/p)=+1$.

Нехай n – BPSW-псевдопросте, $n = pq$, де p – просте та $J(D/p) = 1$.

Теорема (7) стверджує у разі, що p – простий дільник $2^{q-1}-1$ і U_{q+1}, та:

$$\begin{aligned} &a)\ 2^{q-1} \equiv 1 \bmod p, \\ &b)\ U_{q+1} \equiv 0 \bmod p. \end{aligned} \tag{3}$$

Позначимо $(q-1)/2$ через k. Тоді (нагадаємо, $U_{2n}=U_nV_n$):

$$\begin{aligned} &a)\ (2^k-1)(2^k+1) = 0 \bmod p, \\ &b)\ U_{k+1}V_{k+1} = 0 \bmod p. \end{aligned} \tag{4}$$

Нехай γ :

$$\gamma = \frac{\alpha}{\beta} = \frac{P+\sqrt{Q}}{P-\sqrt{Q}} = \frac{P}{Q}\alpha - 1.$$

Оскільки $J(D/p)=1$, то $\sqrt{Q}$ належить Z_p, отже і $\gamma \in Z_p$. Тому умови (4) можна записати так (нагадаю, $k = (q-1)/2$):

$$\begin{aligned} &a)\ 2^k \equiv \pm 1 \bmod p, \\ &b)\ \gamma^{k+1} = \pm 1 \bmod p. \end{aligned} \tag{5}$$

Нехай n - BPSW-псевдопростий, $n = pq$, де p - просте і $J(D/p) = 1$. Тоді

а) число p задовольняє умову:

$$\gcd(ord(2,p),\rho(p,P,Q)) \le 2,$$

б) при фіксованому p число q задовольняє умові:

$$q = q_0 \bmod LCM\,(ord(2,p),\rho(p,P,Q)),$$

де q_0 знаходиться з умов:

$$a)\, q_0 = 1 \bmod ord(2,p),$$
$$b)\, q_0 = -1 \bmod \rho(p,P,Q).$$

Нагадаємо, що в цьому випадку і $ord(2,p)$, і $\rho(p,P,Q)$ є дільниками $p-1$, а LCM($ord(2,p)$, $\rho(p,P,Q)$) буде дільником $2(p-1)$.

Доведення . Нехай $n = pq$. Оскільки $2^{p-1} = 1 \bmod p$ то $2^{q-1} = 1 \bmod p$ або $q - 1 = 0 \bmod \text{ord}(2, p)$ або:

$$q \equiv 1 \bmod ord(2,p). \qquad (6)$$

Далі,

$$U_{pq+1}(p,Q) = 0 \bmod n \Rightarrow U_{pq+1}(P,Q) = 0 \bmod p.$$

Маємо: $U_{p-1} = 0 \bmod p$, тому:

$$U_{pq+1} = U_{(p-1)q+q+1} = U_{q+1} = 0 \bmod p,$$

Тобто $q+1$ ділиться на $\rho(p,P,Q)$ або:

$$q = -1 \bmod \rho(p,P,Q). \qquad (7)$$

Виходить q має задовольняти умовам:

$$a)\, q \equiv 1 \bmod ord(2,p),$$
$$b)\, q = -1 \bmod \rho(p,P,Q).$$

що доводить (б), причому і ord(2, p), і $\rho(p,P,Q)$ є дільниками $p-1$.

Припустимо, що:

$$d = GCD(ord(2,p),\rho(p,P,Q)) > 2.$$

Тоді:

$$a)\, q = 1 \bmod d,$$
$$b)\, q = -1 \bmod d.$$

Числа, які відповідають умовам теореми, існують, але їх не так багато. Для їхнього пошуку можна скористатися наступним твердженням. Нехай n - BPSW-псевдопростий, p - простий дільник n і $J(D/p) = 1$. Тоді

$$ord(2,p)\cdot\rho(p,P,Q) \le 2(p-1).$$

Оскільки ми маємо обмеження знизу на $\rho(n)$ (наслідок 4), можна написати таку нерівність.

Нехай n - BPSW-псевдопросте, $|Q| < 100$, p – простий дільник n та $J(D/p) = 1$. Тоді:

$$\frac{p-1}{ord(2,p)} > \ln(p)/4 + 1/2.$$

Нехай n - BPSW-псевдопростий, p - простий дільник n і $J(D/p) = 1$. Тоді:

а) якщо $p-1$ ділиться на 2^k, $k > 1$, або $2^s = 1$, або $U_s = 0$, де $s = (p-1)/2^{k-1}$;

б) якщо r – непарний простий дільник $p-1$ кратності $k \ge 1$, то або $2^s = 1$, або $U_s = 0$, де $s = (p-1)/r^k$.

Доведення випливає з того, що:

$$\gcd(ord(2,p),\rho p),\rho(Q))\le 2.$$

Числа p, менші 234 (17.17 млрд), які відповідають умовам теореми (10), знайдені при $|D|<128$.

За простих D і $D=-15$ їх виявилося 170, їх 5 – при $Q=-1$. Це 61681, 363 101449, 4 278 255 361, 4 562 284 561 і 4 582 537 681.

Простий множник p, з $J(D/p)=-1$. При $|Q|<32$ серед чисел $q<2^{18}$, $J(1-4Q, q)=-1$ немає жодного, який мав би простий дільник числа $R(q)$ більший q.

Для простого p і цілих P, Q через $L(p)=L(p, P, Q)$ позначимо число:

$$L = LCM(ord(2, p), \rho p) P, Q)).$$

Нехай n - BPSW-псевдопростий, $n=pq$, де p - просте і $J(D, p)=-1$. Тоді:

$$q \equiv 1 \bmod L(p, P, Q))$$

Нагадаємо, що в цьому випадку $\mathrm{ord}(2, p)$ – дільник $p-1$, а $\rho(p,P,Q)$ – дільник $p+1$. Оскільки $J(D/p)=-1$, то $U_{p+1}\equiv 0 \bmod p$. Ми знаємо, що:

$$U_{pq+1}=0 \bmod pq, 2^{pq-1}=1 \bmod pq,$$

або, враховуючи, що $U_{p+1}=0 \bmod p$,

$$U_{p-1}=0 \bmod p, 2^{q-1}=1 \bmod p, \text{ або } q-1=0 \bmod \rho(p,P,Q), q-1=0 \bmod ord(2,p),$$

$$\text{або } q=1 \bmod \rho(p,P,Q), q=1 \bmod ord(2,p).$$

Ці дві умови можна поєднати в одну:

$$q=1 \bmod LCM(ord(2, p), \rho p) P, Q)).$$

Нехай n - BPSW-псевдопростий, $n=pq$, де p - просте і $J(D/p)=-1$. Позначимо $LCM(ord(2, p), \rho p) P, Q))$ через L. Тоді

а) якщо L ділиться на 5 та $J(p/5)=-1$, то $D=5$;

б) якщо L ділиться на 5 та $J(p/5)=+1$, то $|D|>5$;

в) якщо L ділиться на просте $s>5$ та $J(p/s)=-1$, то $|D|\le s$;

г) якщо L ділиться на 3 та $J(p/3)=-1$, то $|D|\le 15$.

Сформулюємо трохи інакше.

Нехай n - BPSW-псевдопростий, $n=pq$, де p - просте і $J(D/p)=-1$. Позначимо $LCM(ord(2, p), \rho p) P, Q))$ через L. Тоді

а) якщо L ділиться на просте більше 3 і $J(p/s)=-1$, то $|D|\le s$;

б) якщо L ділиться на 3 та $J(p/3)=-1$, то $|D|\le 15$.

Нехай n - BPSW-псевдопростий, $n=pq$, де p - просте і $J(D/p)=-1$. Тоді

а) якщо $\mathrm{ord}(2, p)$ ділиться на просте більше 3 і $J(p/s)=-1$, то $|D|\le s$;

б) якщо $\mathrm{ord}(2, p)$ ділиться на 3 та $J(p/3)=-1$, то $|D|\le 15$.

2. Властивості добутку двох простих чисел

Зберемо разом все, що відомо, у випадку $n=pq$ – добуток двох простих чисел. Оскільки $J(D/n)=-1$, то нехай $J(D/p)=-1$, $J(D/q)=+1$. Вважатимемо також, що $|Q|<32$. Тоді:

(1) $J(D_0/n)=J(D_0/p)J(D_0/q)=1$ за всіх $D_0=5, -7, \ldots$ менших за модулю $|D|$.

(2) p – простий дільник $R(q)=\gcd(2^{q-1}-1, U_{q-1}(P,Q))$.

Це ж можна сформулювати так:

а) p – простий дільник $2^{q-1}-1$,

б) p – простий дільник $U_{q+1}(P,Q)$.

(3) q – простий дільник $R(p)=\gcd(2^{p-1}-1, U_{p+1}(P,Q))$.

Це ж можна сформулювати так:

а) q – простий дільник $2^{p-1}-1$,

б) q – простий дільник $U_{p+1}(P,Q)$.

(4) $p, q > 2^{18}$ (кордон може бути розширений).

(5) $\gcd(ord(2,q), \rho(q,P,Q)) \le 2$.

(6) Для q < 2^{34} є фіксований список (331 число).

(7) $p \equiv 1 \bmod ord(2,q)$, (тут $ord(2,q)$ буде дільником $q-1$).

(8) $p \equiv -1 \bmod \rho(q,P,Q)$, (тут $\rho(q,P,Q)$ буде дільником $q-1$).

(9) $p \equiv 1 \bmod ord(2,p)$, (тут ord(2, p) буде дільником $p-1$).

(10) $q \equiv 1 \bmod \rho(p,P,Q)$, (тут $\rho(p,P,Q)$) буде дільником $p+1$).

(11) $q > ord(2,p)\cdot\rho(p,P,Q)$.

(12) $(p-1)/ord(2,p) > \ln(p)/4 + 1/2$.

2.1 Добуток кількох простих з $J(D/p_i) = -1$

Нагадаємо (визначення 6), що через $L(p) = L(p, P, Q)$ позначено число LCM(ord(2,p), ρ(p,P,Q)).

Нехай n - BPSW-псевдопростий, $n = p_1 \dots p_k q$, де p_i – прості та $J(D, p_i) = -1$ при $i = 1, \dots, k$ а q – довільне. Позначимо через S_i добуток всіх p_j, крім p_i:

$$S_i = p_1 \dots p_k / p_i = n/(qp_i).$$

Тоді:

а) $qS_i \equiv 1 \bmod L(p_i, P, Q)$ $i = 1, \dots, k$;

б) нехай d_{ij} - найбільший спільний дільник $d_{ij} = \gcd(L(p_i), L(p_j))$ довільних i, j. Тоді $(p_i - p_j)S_{ij}$ ділиться на d_{ij}, де $S_{ij} = S_i/p_j = S_j/p_i$ – добуток всіх p_m, крім p_i та p_j.

Наслідок 5. Нехай n – BPSW-псевдопростий, $n = p_1p_2q$, де p_i – прості та $J(D, p_i) = -1$ при $i = 1, 2$ а q – довільне. Тоді:

а1) $qp_2 \equiv 1 \bmod L(p_1)$,

а2) $qp_1 \equiv 1 \bmod L(p_2)$,

б) $p_1 - p_2$ ділиться на $d_{12} \equiv \gcd(L(p_1), L(p_2))$.

Нехай n – BPSW-псевдопростий, $n = p_1p_2q$, де p_i – прості та $J(D, p_i) = -1$ при $i = 1, 2$ а q – довільне. Тоді $p_1 - p_2$ ділиться на gcd(ord(2, p_1), ord(2, p_2)).

3. BPSW-псевдопрості до 2^{64}

У цьому розділі розглядаються лише числа n, менші 2^{64}, BPSWпсевдопрості з параметром D, за модулем меншим 128. Усі обчислення проводилися на процесорі AMD Athlon II X2240, 2.8 GHz.

Відповідно до теореми (5), параметр D або простий за модулем, або дорівнює -15.

Алгоритм перевірки такий.

Крок 1. Немає BPSW-псевдопростих чисел виду $n = pq$, де p - просте, і $q < 2^{19}$.

Перевірка за допомогою теореми (9). Перевіряти можна на

C++ за допомогою GMP. Час – близько доби.

Крок 2. Немає BPSW-псевдопростих чисел виду $n = pq$, де p - просте, $J(D / p) = +1$ і $p < 2^{34}$.

Числа p, менші за 2^{34} (17.17 млрд), що задовольняють умовам теореми (10) знайдено при $|D| < 128$ та наведені у файлі GCD_le_2bb.txt.

За простих D і D = −15 їх виявилося 170. Ці 170 чисел треба перевіряти окремо.

Крок 3. Немає BPSW-псевдопростих чисел виду $n = pq$, де p, q – прості, $J(D/p) = -1$, $J(D/q) = +1$, та $2^{19} < p < 2^{30}$.

Перевірка за допомогою теореми (13). Для кожного простого p із зазначеного інтервалу перебираємо всі D, для кожної пари (p, D) знаходимо $L = LCM(ord(2, p), ord(p, P, Q))$ і перебираємо всі q, що задовольняють порівнянню $q = 1 \mod L(p, P, Q)$ та менші $2^{64}/p$.

Висновок. При великих p величина L зазвичай виявляється великою і кількість допустимих або взагалі не буде, або їх буде мало. Тому перебір виявляється не надто великим. Час – близько двох діб. Після цих кроків можна дійти невтішного висновку, що немає BPSWпсевдопростых чисел, менших 2^{64}, є добутком двох простих чисел.

Перелік використаних джерел.

1. Chen Z., Greene J. Some Comments on Baillie–PSW Pseudoprimes // Fib. Quart. 2003. V. 41, № 4. P. 334–344.

2. Crandall R. E.,Pomerance *C*. Prime Numbers: a computational perspective, second edition. Springer : New York, 2005. 597 p.

3. Jameson G.J.O. Carmichael numbers and pseudoprimes. Lancaster Univ. UK. URL: http://www.maths.lancs.ac.uk/ jameson/carpsp.pdf.

4. Lehmer D. H. On Fermat's quotient, base two // Math. Comput., 1981. V. 36, № 153. P. 289–290.

5. Ribenboim P. My numbers, my friends: popular lectures on number theory. Springer, 2000. 392 p.

6. *Galway W.* Tables of pseudoprimes and related data. URL: http://www.cecm.sfu.ca/Pseudoprimes/index-2-to-64.html.

7. Martin R. Albrecht, Jake Massimo, Kenneth G. Paterson, and Juraj Somorovsky, Prime and Prejudice: Primality Testing Under Adversarial Conditions, ACM SIGSAC Conference on Computer and Communications Security, October 15-19, 2018, Toronto, Ontario, Canada, pp. 281–298, DOI 10.1145/3243734.3243787. Available at https: //eprint.iacr.org/2018/749.pdf


АВТОМАТИЗАЦІЯ
ТА КОМП'ЮТЕРНО-ІНТЕГРОВАНІ
ТЕХНОЛОГІЇ
проблемно-наукова міжгалузева
конференція молодих науковців
аспірантів та студентів
м. Тернопіль
2022

ЗАХІДНОУКРАЇНСЬКИЙНАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИКАРПАТСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТЕФАНИКА
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ
НАЦІОНАЛЬНИЙ ТРАНСПОРТНИЙ УНІВЕРСИТЕТ
НАДВІРНЯНСЬКИЙ КОЛЕДЖ НТУ
ГАЛИЦЬКИЙ КОЛЕДЖ ІМ. В. ЧОРНОВОЛА

Проблемно-наукова міжгалузева конференція

АВТОМАТИЗАЦІЯ ТА КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ

(АКІТ – 2022)

21—23 лютого 2022 року

Тернопіль

БЕЗПЕКА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

УДК 004.56.5(043.2)

Ігнатєв І.В.[1], Кондратюк В.М.[1]

[1]Західноукраїнський національний університет

АЛГОРИТМИ ПЕРЕВІРКИ ЧИСЛА НА ПРОСТОТУ

Вступ. Великі прості числа мають вагоме практичне застосування в криптографічних алгоритми. Ефективність багатьох сучасних шифрів залежить від того, наскільки просте число велике. Наприклад, для шифру RSA, який на даний момент використовує відкритий 2048-бітний ключ, що складається із добутку двох простих чисел. Збільшення потужності технічних засобів може дозволити факторизувати відкритий ключ (на даний момент максимальне факторизоване число RSA-768 містить 232 десяткових знаків) [1], що призведе до небезпеки шифрів, заснованих на відкритому ключі. Тому ефективний пошук великих простих чисел здатний покращити вже існуючі шифри.

Мета: розгляд різних видів алгоритмів визначення простоти чисел, виявлення оптимальних алгоритмів для конкретних наборів даних, знаходження оптимізаційних кроків для конкретних тестів простоту. Основні об'єкти вивчення: достовірні та ймовірнісні тести на простоту, їх відмінності та специфіка отриманих результатів для споріднених тестів.

1. Аналіз алгоритмів тестів простоти

Тести на простоту - вид алгоритмів, метою яких є визначення простоти заданого числа. Серед тестів розрізняють два підвиди: справжні та імовірнісні тести.

Справжні випробування дозволяють точно визначити простоту числа. Однак вони мають ряд суттєвих недоліків, що обмежують їхнє застосування. Серед них: умовність – тест працює лише для заздалегідь визначених чисел (тести для чисел Мерсенна, Ферма, Прота), мала швидкість обробки даних – поліноміальний час. Серед цих тестів у роботі були розглянуті та оптимізовані наступні тести:

- Тест Люка, що базується на однойменній теоремі, працює за поліноміальний час. У даному тесті визначається простота числа $2^p - 1$ за поліноміальний час від бітової довжини числа p. Саме завдяки цьому тесту числа Мерсенна майже завжди були самими великими знайдених простих чисел.
- Тест Адлемана - Померанса - Румелі - універсальний тест, що працює за $O(\log n^{c*\log\log n})$. Даний алгоритм може бути застосований на практиці у зв'язку з повільнозростаючою складністю.
- Тест Агравала - Каяла - Саксени - універсальний тест, заснований на узагальненій теоремі Ферма, що працює за $O((\log n)^6)$. Головна особливість даного алгоритму полягає у тому, що він поліноміальний, універсальний, детермінований та безумовний на відміну від усіх попередніх алгоритмів [2].
- Тести за допомогою еліптичних кривих. Дані тести є одними з найбільш швидких із відомих методів перевірки числа на простоту. У гіршому випадку складність такого тесту становитиме $O((\log n)^{5+\Sigma})$. І для деяких випадків

цей показник може бути покращено до $4+\sum$.

У свою чергу результатом імовірнісних тестів є лише ймовірність простоти числа. Головною перевагою таких тестів над дійсними є їх швидкість обробки даних (наприклад алгоритм Міллера - Рабіна зі складністю $O((\log n)^3)$ [3].

Однак через ймовірність результатом відповіддю можуть стати так звані псевдопрості числа - числа, які пройшли тест на простоту, але не є. Для зменшення частоти появи подібних чисел використовуються раунди - повторні перевірки числа цим же тестом. Прикладом псевдопростих чисел є числа Кармайкла. У роботі [2] було розглянуто такі ймовірні тести:

- Тест Ферма, заснований на однойменній теоремі, що працює за $O(k \cdot \log n \cdot \log n))$.
- Тест Міллера — Рабіна, який використовує теорему Рабіна про свідків простоти, мінімальною швидкістю роботи $O(k \cdot (\log n)^3)$.
- Тест Соловея — Штрассена, заснований на символі Якобі, зі швидкістю роботи $O((\log n)^3)$.

2. Алгоритм BPSW

Проаналізуємо універсальний оптимізаційний алгоритм, отриманий у процес аналізу специфіки кожного тесту для певних числових даних. Даний алгоритм аналогічний тесту BPSW, основу якого лежить використання кількох типів тестів.

На рисунку 1 наведено графік залежності швидкості роботи тесту BPSW від розміру простого числа.

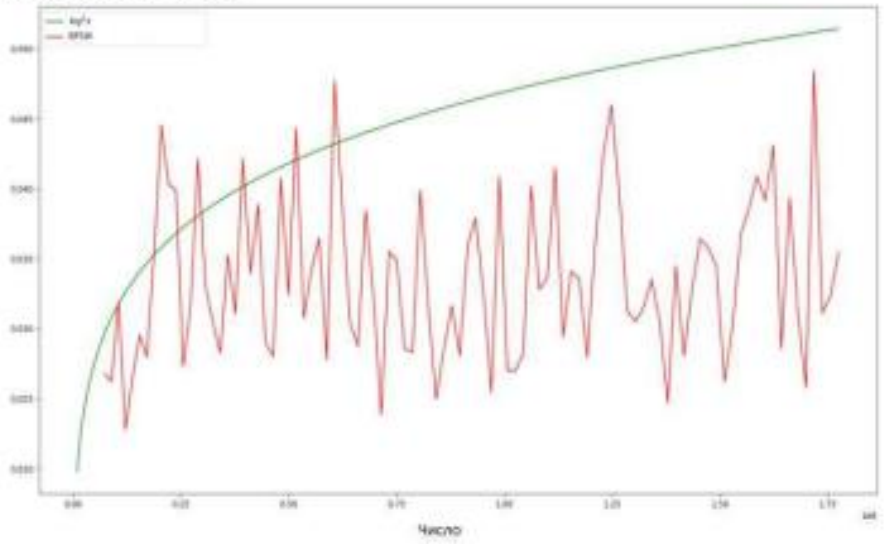

Рисунок 1 - Графік залежності швидкості роботи алгоритму BPSW від розміру простого числа

Приведені експериментальні дослідження дозволяють апроксимувати залежності та оцінити швидкодію запропонованого рішення. Запропонований алгоритм на основі BPSW тесту приведений на рисунку 2.

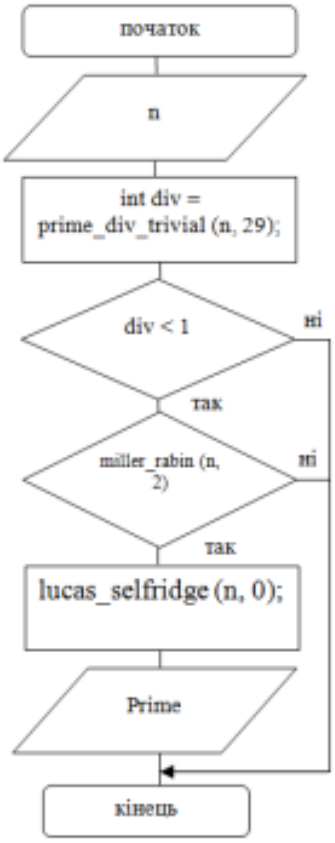

Рисунок 2 – Алгоритм визначення простити числа на основі BPSW тесту простоти

Дана методика дозволяє досягти зменшення обчислювальної складності. Для чисел розрядністю 64 біти обчислювальна складність складатиме $O(\log n)$, проте якщо використовувати довгу арифметику обчислювальна складність складатиме $O(\log^3(n))$.

Висновки. Проведені дослідження дозволяють зробити висновок, що використання алгоритму побудованого на основі тесту BPSW в криптографічних перетвореннях приведуть до зменшення обчислювальної складності та підвищать ефективність.

Перелік використаних джерел.

1. Factorization of a 768-bit RSA Modulus / Th. Kleinjung [et al.] // Advances in Cryptology - CRYPTO 2010 : 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. - Santa Barbara, 2010. - P. 333-350.

3. Бараш, Л. Ю. Алгоритм AKS проверки чисел на простоту и поиск констант генераторов псевдослучайных чисел / Л. Ю. Бараш // Безопасность информационных технологий. - 2005. - №2. - С. 27-38.

4. Шнайер, Б. Прикладная криптография / Б. Шнайер ; пер. с англ. — М. : Триумф, 2013. — 816 с.