В. А. ЛУЖЕЦЬКИЙ
Г. В. ШЕЛЕПАЛО

ПРОСТІ ЧИСЛА:

властивості та застосування в криптографії

Міністерство освіти і науки України
Вінницький національний технічний університет

В. А. Лужецький, Г. В. Шелепало

ПРОСТІ ЧИСЛА:
властивості та застосування в криптографії

Електронний навчальний посібник
комбінованого (локального та мережного) використання

Вінниця
ВНТУ
2024

УДК 512′.72
Л82

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 8 від 25.01.2024 р.)

Рецензенти:
В. М. Михалевич, доктор технічних наук, професор
Ф. М. Сохацький, доктор фізико-математичних наук, професор
В. П. Майданюк, кандидат технічних наук, доцент

Лужецький, В. А.

Л82 Прості числа: властивості та застосування в криптографії : електронний навчальний посібник комбінованого (локального та мережного) використання [Електронний ресурс] / В. А. Лужецький, Г. В. Шелепало. – Вінниця: ВНТУ, 2024. – 161 с.

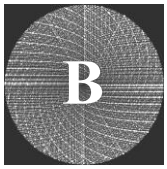
У навчальному посібнику розглядаються властивості простих чисел, які використовуються для розв’язання задач криптографії. Наведено методи та алгоритми факторизації цілих чисел, тестування на простоту і генерування великих простих чисел. Навчальний посібник рекомендується студентам і аспірантам, що навчаються за спеціальністю «Кібербезпека та захист інформації», а також він буде корисним студентам, що навчаються за іншими спеціальностями галузі знань «Інформаційні технології».

УДК 512′.72

ЗМІСТ

ВСТУП 5
1 ДЕЯКІ ВІДОМОСТІ ПРО ПРОСТІ ЧИСЛА 9
1.1 Поняття простого числа 9
1.2 Скільки існує простих чисел? 10
1.3 Яким чином розташовані прості числа в послідовності натуральних чисел? 12
1.4 Графічні подання розташування простих чисел 16
1.5 *k*-кортежі простих чисел 19
Контрольні питання 24
Вправи 25
Джерела для поглибленого вивчення 26
2 ПОШУК ПРОСТИХ ЧИСЕЛ 27
2.1 Алгоритм «Решето Ератосфена». 27
2.1.1 Базовий алгоритм 27
2.1.2 Решето, що просіює, починаючи з квадратів простих чисел 28
2.1.3 Решето для непарних чисел 29
2.2 Решето Ейлера 29
2.3 Алгоритм «Решето Сундарама» 30
2.4 Решето Аткіна 31
2.5 Формули для простих чисел 32
2.6 Послідовності Люка 36
2.7 Спеціально сконструйовані прості числа 42
2.8 Прості числа з властивостями їх представлення в системі числення 50
2.9 Центровані *k*-кутні прості числа 55
2.10 Взаємно прості числа 58
Контрольні питання 61
Вправи 62
Джерела для поглибленого вивчення 63
3 ФАКТОРИЗАЦІЯ ЦІЛИХ ЧИСЕЛ 66
3.1 Поняття факторизації 66
3.2 Класи натуральних чисел. 68
3.3 Пробне ділення 72
3.4 Метод колісної факторизації 72
3.5 Метод факторизації Ферма 74
3.6 Метод факторизації Ейлера 77
3.7 ρ алгоритм Полларда 77
3.8 $p-1$ алгоритм Полларда 79
3.9 $p+1$ алгоритм Вільямса 81
3.10 Спеціальне решето числового поля 83
3.11 Метод квадратичного решета 83

3.12 Метод раціонального решета. 86
3.13 Алгоритм загального решета числового поля 88
Контрольні питання 90
Вправи 91
Джерела для поглибленого вивчення 91
4 ТЕСТИ ПРОСТОТИ 93
4.1 Істинні тести простоти 93
4.1.1 Перевірка простоти пробним діленням. 93
4.1.2 Тест простоти AKS 96
4.1.3 Тест Люка-Лемера 99
4.2 Сертифікати про простоту числа 101
4.2.1 Сертифікат Пратта 101
4.2.2 Алгоритм Голдвассера–Кіліана 103
4.2.3 Алгоритм Аткіна-Морейна 104
4.2.4 Сертифікат Голдвассера-Кіліана-Аткіна-Морейна 115
4.3 Імовірнісні тести простоти 117
4.3.1 Тест Ферма 118
4.3.2 Імовірнісний тест простоти Соловея-Штрассена 120
4.3.3 Імовірнісний тест простоти Міллера-Рабіна 124
4.3.4 Імовірнісний тест простоти Люка 128
4.4 Псевдопрості числа 130
Контрольні питання 132
Вправи 133
Джерела для поглибленого вивчення 134
5 ГЕНЕРУВАННЯ ВЕЛИКИХ ПРОСТИХ ЧИСЕЛ 136
5.1 Підходи до генерування великих простих чисел 136
5.2 Функціональна модель RBG 137
5.3 Генерування псевдовипадкових бітів за допомогою механізму DRBG на основі геш-функції 140
5.4 Генерування випадкових імовірно простих чисел 141
5.5 Побудова доказово простих чисел 145
5.6 Незаконні великі прості числа 149
5.7 Поняття асиметричного бекдору 150
Контрольні питання 154
Вправи 155
Джерела для поглибленого вивчення 155
ПРЕДМЕТНИЙ ПОКАЖЧИК 157



ВСТУП

Тривалий час прості числа вважалися малозастосовними поза чистою математикою. Ситуація суттєво змінилася після появи концепцій криптографії з відкритим ключем. Поняття криптографії з відкритим ключем виникло після публікації у 1976 році Діффі та Хеллманом фундаментальної статті «Нові напрямки в криптографії», в який вони описали протокол узгодження захищених ключів, що реалізується в загальнодоступному каналі зв'язку (Угода про ключ Діффі–Хеллмана). Операції цього протоколу базуються на арифметиці за модулем великого простого числа.

З того часу було запропоновано багато інших криптографічних систем, основаних на використанні великих простих чисел і пов'язаних з ними математичних результатах. Наприклад, великі прості числа використовуються в схемах шифрування з відкритим ключем, узгодженні ключів, схемах цифрового підпису та генераторах псевдовипадкових чисел.

Конкретним прикладом є вимога наявності простого числа p для визначення скінченного поля $\mathbf{Z}_p$, що використовується в протоколі узгодження ключів Діффі-Хеллмана та його похідних. В цьому випадку елемент високого порядку в $\mathbf{Z}_p^*$ також є обов'язковим.

Іншим прикладом є вимога наявності простих чисел p і q для модуля RSA $n = p \cdot q$. У цьому випадку прості числа мають бути достатньої розрядності та бути випадковими. Крім того, потрібно щоб прості числа мали певні додаткові властивості, аби криптосистема не була чутлива до спеціалізованих атак.

Третім прикладом є вимога, що стосується незвідного полінома $f(x)$ степеня m над скінченним полем $\mathbf{Z}_p$ для побудови скінченного поля $\mathbf{F}_{p^m}$. У цьому випадку елемент високого порядку в $\mathbf{F}_{p^m}^*$ також потрібен.

Наведені приклади показують, що прості чисел відіграють важливу роль у захисті конфіденційної інформації в сучасну цифрову епоху.

Незважаючи на активне вивчення простих чисел протягом багатьох століть, їх властивості ще недостатньо розуміються науковим співтовариством. Досі не встановлено простої закономірності розподілу простих чисел, немає ефективного методу визначення простоти числа, немає задовільної формули кількості простих чисел, і взагалі, сума знань про властивості, ознаки, характер поведінки простих чисел є дуже малою і тому немає повної картини цього явища.

Завдяки розвитку криптографії і поширенню основаних на теорії чисел алгоритмів, на передовому краю математики перебувають дослідження,

пов'язані з перевіркою числа на простоту і з розкладанням на прості множники (факторизацією).

Результати досліджень показують, що перевірка на простоту набагато простіше розкладання на прості множники. Це дивує нематематиків, адже в школі вчать перевіряти число на простоту тим самим методом, що й шукати його прості множники: перебором усіх можливих дільників. Але, виявляється, існують хитрі способи довести простоту числа без цього. Ці методи дозволяють довести, що число є складеним, без знаходження будь-яких його дільників. Достатньо показати, що це число не проходить тест на простоту.

Саме складність розкладання великих чисел на прості множники гарантує безпеку шифрування RSA. Якщо зловмисник може розрахувати число, яке використовується в шифруванні, він зможе обчислити закритий ключ і розшифрувати повідомлення.

Безпека криптосистем безпосередньо пов’язана з довжиною використовуваних ключів. У міру того, як комп’ютери стають потужнішими, довжина ключів також має збільшуватися для підтримки безпеки. Тому виникає запитання: Як генерувати великі прості числа? Пошук «великих» простих чисел викликає інтерес математиків багато століть. Однак останнім часом ці дослідження набули прикладного значення, оскільки для генерування великих простих чисел, що використовуються в криптографії з відкритим ключем, потрібні методи, які є достатньо ефективними.

У навчальному посібнику розглядаються саме сформульовані вище питання стосовно простих чисел.

Розділ 1 присвячено опису властивостей простих чисел, які використовуються для розв’язання певних задач криптографії. Наведено відповіді на питання: «Скільки існує простих чисел?» і «Яким чином розташовані прості числа в послідовності натуральних чисел?».

Закономірності розташування простих чисел у послідовності натуральних чисел шукали багато поколінь математиків, але спроби були марні через те, що закон їх розподілу дуже складний. Описуються графічні подання розташування простих чисел у вигляді двовимірного подання набору натуральних чисел: піфагограма, скатертина Улама, трикутне подання Клаубера і спіраль Сакса.

У теорії чисел властивості розподілу простих чисел вивчають зокрема з використанням k-кортежів, тому розглядаються прості-близнюки, двоюрідні прості числа, прості триплети та інші k-кортежі.

У розділі 2 «Пошук простих чисел» описуються детерміновані алгоритми знаходження початкового набору простих чисел, що не перевищують певного числа n, які називають решето Ератосфена, решето Сундарама, решето Аткіна та ін. Крім таких алгоритмів, ще з давніх часів математики прагнули знайти «формулу для простих чисел». Наведено поліноміальні та степеневі формули, для яких з використанням сучасної комп’ютерної техніки сьогодні можна емпірично аналізувати ймовірність отримати прості значення на різноманітних інтервалах.

Існує кілька числових послідовностей, що мають властивості, корисні для розв'язання певних задач криптографії. Ці послідовності утворюються на основі базових послідовностей, які запропонував Едуард Люка. Описуються послідовності Люка, Фібоначчі та Пелла.

Розглядаються класи спеціально сконструйованих простих чисел: числа Евкліда, числа Куммера, числа Софі Жермен, числа Мерсенна, числа Ферма числа Солінаса та ін.

Певний інтерес становлять прості числа, властивості яких пов'язані з їх поданням у певній системі числення, та центровані k-кутні прості числа (фігурні числа). На цей час вони мають лише теоретичне значення, однак у майбутньому вони можуть бути поштовхом до нових ідей у криптографії, наприклад, як вимірність простору породила ідею двовимірної та тривимірної криптографії.

Метою розділу 3 «Факторизація цілих чисел» є розгляд алгоритмів факторизації спеціального та загального призначення. Криптостійкість алгоритмів шифрування та протоколів з відкритим ключем, зокрема алгоритму RSA, базується саме на передбачуваної складності задачі факторизації.

Виходячи з розкладання натуральних чисел на прості множники розглядаються k-майже прості числа (напівпрості і сфенічні) та B−гладкі і B−степенево гладкі числа.

В розділі «Тести простоти» розглядаються істинні та ймовірнісні тести простоти, а також сертифікати про простоту числа. За допомогою істинних тестів можна довести, що додатні цілі числа є простими, а тому їх часто називають алгоритмами доведення простоти. Описано перевірку простоти пробним діленням, тест простоти Агравала-Каяли-Саксени, тест простоти Люка-Лемера для чисел Мерсенна, сертифікати Пратта і Голдвассера-Кіліана-Аткіна-Морейна. Розглянуто найпопулярніші ймовірнісні тести простоти: тест Ферма, тест Соловея-Штрассена, тест Міллера-Рабіна і тест Люка. Наводяться вимоги до тестів, сформульовані відповідними стандартами.

Генеруванню великих простих чисел присвячено окремий розділ, в якому описуються алгоритми генерування випадкових імовірно простих чисел та побудови доказово простих чисел, що рекомендуються кількома стандартами.

Наприкінці кожного розділу наведено контрольні питання, вправи та джерела для поглибленого вивчення. Своїм завданням ми вважали забезпечення посилань на велику кількість джерел, що містять детальний опис тих аспектів, які не розкрито в достатньому обсязі.

Більшість наведених алгоритмів деталізовано до рівня, який забезпечує можливість легко створювати програмні додатки.

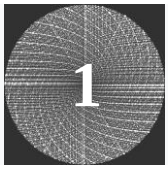
В процесі викладення матеріалу наводиться багато послідовностей чисел. Достатньо вичерпну інформацію про найрізноманітніші числові послідовності, які зустрічаються в математиці, computer science та суміжних науках, можна отримати, скориставшись онлайн-енциклопедією

послідовностей цілих чисел OEIS (The On-Line Encyclopedia of Integer Sequences), яка доступна за адресою: https://oeis.org/.

Кожна послідовність у OEIS має нумероване позначення, яке починається з літери A і далі містить шість десяткових цифр. Опис кожної послідовності містить відомості про її властивості, формули та програми для її обчислення, посилання на літературу та веб-сторінки, текстовий файл зі списком членів послідовності.

Навчальний посібник буде корисним студентам і аспірантам, що навчаються за спеціальністю «Кібербезпека та захист інформації», а також за іншими спеціальностями галузі знань «Інформаційні технології».

Автори висловлюють вдячність рецензентам за корисні поради, які сприяли покращенню змісту навчального посібника.



ДЕЯКІ ВІДОМОСТІ ПРО ПРОСТІ ЧИСЛА

1.1 Поняття простого числа

Означення 1.1. Ціле число p називають *простим числом*, якщо воно:

1) відмінне від 0 і ± 1;
2) має дільником лише ± 1 і $\pm p$.

Наприклад, числа 2, 3, 5, 7, 11, 13, 17 ... – прості. Хоча означення простоти числа надзвичайно просте, наведена послідовність простих чисел є нетривіальним об'єктом уваги багатьох поколінь математиків.

Перші результати досліджень властивостей простих чисел було наведено ще в роботах Евкліда.

Евклід (грец. Ευκλείδης; близько 325 - 270 до н.е.). Основна праця Евкліда «Начала» (у латинізованому варіанті *Elementa*, «Елементи») складається із серії книжок, у яких міститься систематизований виклад геометрії, а також деяких питань теорії чисел.

«Начала» складаються з тринадцяти книг. VII-IX книги присвячені теорії чисел. У цих книгах розглядаються теореми про пропорції і геометричні прогресії, вводиться метод для знаходження найбільшого спільного дільника двох чисел (відомий нині як алгоритм Евкліда), подається метод побудови ряду парних досконалих чисел, доводиться нескінченність множини простих чисел.

«Начала» відіграли винятково важливу роль у подальшому розвитку математичної науки.

Лема Евкліда. *Якщо просте число p ділить добуток ab двох цілих чисел a і b, то p ділить принаймні одне з цих чисел a або b.*

Наприклад, якщо p=17, a=119, b=91, тоді ab=119×91=10829, і оскільки це ділиться на 17, лема передбачає, що число 17 ділить одне або обидва числа. Дійсно, 119=17×7.

Ця властивість є ключовою в доведенні фундаментальної теореми арифметики.

У сучасній теорії чисел використовують таку еквівалентну форму леми Евкліда.

Теорема 1.1. *Якщо p – просте число, яке ділить добуток ab і не ділить a, то воно ділить b.*

Також існує узагальнення леми Евкліда на випадок будь-яких цілих чисел.

Теорема 1.2. *Якщо ціле число n ділить добуток ab двох цілих чисел і є взаємно простим з a, то n ділить b.*

Незважаючи на те, що означення простих чисел дуже просте, питання щодо простих чисел можуть бути дуже складними.

1. Скільки існує простих чисел?
2. Яким чином розташовані прості числа в послідовності натуральних чисел?
3. Чи існує формула, що генерує прості числа?
4. Яким чином можна дізнатися для даного числа, чи просте воно?

Розглянемо відомі на цей час відповіді на ці питання.

1.2 Скільки існує простих чисел

Сучасні технології та алгоритми можуть отримати вражаюче великі прості числа, однак жодне відкриття простого числа не може бути кінцем історії. Дійсно, простих чисел існує нескінченно багато, як було доведено Евклідом у 300 році до нашої ери. Це досягнення вважають початком абстрактної теорії простих чисел. Знаменитий доказ наступної теореми, по суті, належить Евкліду.

Теорема Евкліда (Книга IX, твердження 20). *Простих чисел нескінченно багато.*

Сучасне трактування доведення твердження, яке запропонував Евклід, має такий вигляд.

Розглянемо будь-який скінченний список простих чисел $p_1, p_2, \ldots, p_n$. Буде показано, що існує хоча б одне додаткове просте число, якого немає в цьому списку. Нехай P – добуток усіх простих чисел зі списку: $P = p_1 p_2 \ldots p_n$. Нехай $q = P + 1$. Тоді q або просте, або ні.

Якщо q просте, то це ще одне просте число, якого немає в списку, а саме q.

Якщо q не просте, то деякий простий множник p ділить q. Якби цей множник p був у нашому списку, то він би ділив P (оскільки P є добутком кожного числа в списку); але p також ділить $P + 1 = q$, як було сказано вище. Якщо p ділить P, а також q, тоді p також має поділити різницю двох чисел, яка дорівнює $(P + 1) - P$ або число 1. Оскільки жодне просте число не ділить 1, то p не може бути в списку. Це означає, що існує принаймні ще одне просте число, окрім тих, що у списку.

Це доводить, що для кожного скінченного списку простих чисел існує просте число, якого немає в списку.

В оригінальній роботі Евклід використав метод, який він часто застосовував, тобто метод узагальненого прикладу. Він вибирає лише три прості числа і, використовуючи загальний метод, описаний вище, доводить, що завжди можна знайти додаткове просте число.

Математики пропонували й інші доведення теореми Евкліда. Ось одна з варіацій.

Факторіал $n!$ натурального числа n ділиться на кожне ціле число від 2 до n, оскільки він є добутком усіх них. Число $n!+1$ не ділиться на жодне з цілих чисел від 2 до n включно (залишок від ділення на кожне з них дорівнює 1). Отже, $n!+1$ є або простим числом, або ділиться на просте число, більше за n. У будь-якому випадку для кожного натурального числа n існує принаймні одне просте число, більше за n. Оскільки кількість натуральних чисел нескінченна, то кількість простих чисел також нескінченна.

У 1737 році Леонард Ейлер посилив результат Евкліда, використавши для доведення достатньо потужний математичний апарат.

Леонард Ейлер (Ойлер) (нім. *Leonhard Euler*; 1707 – 1783).

Ейлер вважається найвидатнішим математиком 18-го століття, а, можливо, навіть усіх часів. Він також є одним з найбільш плідних математиків. Ейлер є автором 866 наукових публікацій, зокрема у галузях математичного аналізу, диференціальної геометрії, теорії чисел, теорії графів, наближених обчислень, небесної механіки, математичної фізики, оптики, балістики, кораблебудування, теорії музики, що мали значний вплив на розвиток науки.

Ейлер пов'язав характер розподілу простих чисел з ідеями з аналізу. Він довів, що сума обернених до простих чисел розходиться. У цей спосіб він виявив зв'язок між дзета-функцією Рімана і простими числами, результат відомий як «тотожність Ейлера у теорії чисел». Він також винайшов функцію Ейлера $\varphi(N)$, з використанням властивостей якої він узагальнив малу теорему Ферма до того, що зараз називається теоремою Ейлера. Він зробив значний внесок у теорію досконалих чисел, якою математики були зачаровані з часів Евкліда. Ейлер також досяг прогресу в напрямку теореми про розподіл простих чисел і висунув гіпотезу квадратичної взаємності. Ці два поняття розглядаються як основні теореми теорії чисел, а його ідеї підготували ґрунт для робіт Гауса.

Ейлер показав, що сума чисел, обернених до простих, необмежено зростає, тобто ряд

$$\sum_{p \text{ prime}} \frac{1}{p} = \frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \frac{1}{11} + \ldots. = \infty$$

є розбіжним.

Для цього він розглянув гармонічний ряд:

$$\sum_{n=1}^{\infty}\frac{1}{n}=1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+....=\infty,$$

а також тотожність, за допомогою якої він показав, що множина простих чисел нескінченна:

$$\sum_{n=1}^{\infty}\frac{1}{n}=\prod_{p}\left(1+\frac{1}{p}+\frac{1}{p^2}+...\right)=\prod_{p}\frac{1}{1-p^{-1}}.$$

Тут добуток береться за всіма простими числами. Такі нескінченні добутки сьогодні називають *добутками Ейлера*. Ейлер зауважив, що якби кількість простих чисел була скінченною, то добуток праворуч мав би збігатися, що суперечить розбіжності гармонічного ряду.

1.3 Яким чином розташовані прості числа в послідовності натуральних чисел

Відомо не лише, що існує нескінченна кількість простих чисел, а також відомо, що в будь-якому достатньо великому інтервалі їх є велика кількість, і можна приблизно визначити цю кількість. Це важливо для приблизної оцінки ймовірності того, що велике, навмання вибране число, буде простим.

Першим статистичну закономірність у розташуванні простих чисел помітив Гаусс. У листі Енке (1849) він повідомив, що ще в 1792 або 1793, емпірично, виявив, що щільність простих чисел «в середньому близька до величини, обернено пропорційної логарифму».

Карл Фрідріх Гаусс (нім. *Johann Carl Friedrich Gauß*; 1777 – 1855) – німецький математик, астроном, геодезист та фізик.

Вважається одним з найвидатніших математиків всіх часів, «королем математиків».

Праці Гаусса мали великий вплив на весь подальший розвиток вищої алгебри, теорії чисел, диференціальної геометрії, класичної теорії електрики і магнетизму, геодезії, теоретичної астрономії.

Перший великий твір Гаусса «Арифметичні дослідження» присвячений окремим питанням теорії чисел і вищої алгебри. Тут Гаусс розвинув теорію квадратичних лишків, уперше довів квадратичний закон взаємності – одну з центральних теорем теорії чисел, розробив теорію квадратичних форм, яку раніше побудував Лагранж, виклав теорію поділу кола, яка багато в чому була прообразом теорії Галуа. Гаусс склав величезні таблиці простих чисел, квадратичних лишків і нелишків.

Нехай $\pi(x)$ – функція підрахунку простих чисел, яка дає кількість простих чисел, менших або таких, що дорівнюють x, для будь-якого дійсного числа x. Зауважимо, що позначення $\pi(x)$ не пов’язане з числом π. Наприклад, $\pi(20)=8$,

оскільки є вісім простих чисел (2, 3, 5, 7, 11, 13, 17 і 19), які менші за 20.

Теорема 1.3. (Теорема простих чисел – Prime Number Theorem). *Маємо*

$$\pi(x) \sim x / \ln x . \qquad (1.1)$$

Ця теорема описує асимптотичний розподіл простих чисел, який дає нам загальне уявлення про те, як прості числа розподіляються між додатними цілими числами.

Асимптотична нотація ~, використана в теоремі 1.3, нічого не говорить про межу різниці двох функцій за необмеженого зростання x. Натомість вона вказує на те, що $x / \ln x$ наближає $\pi(x)$ у тому сенсі, що відносна похибка цього наближення прагне до 0, коли $x \to \infty$.

З теореми 1.3 випливає, що кількість простих чисел нескінченна:

$$\lim_{x\to\infty} \frac{x}{\ln x} = \infty .$$

Результат теореми 1.3 дозволяє також зробити висновок, що ймовірність того, що будь-яке додатне ціле число, вибране випадково в діапазоні від нуля до x, буде простим, становить приблизно $1/\ln x$.

Крім того, теорема 1.3 еквівалентна твердженню, що n-е просте число p_n є наближенням $p_n \sim n\ln(n)$. Тут асимптотична нотація також означає, що відносна похибка цього наближення прагне до 0, коли $n \to \infty$.

Теорема 1.3 була незалежно доведена Адамаром і Валле-Пуссеном у 1896 році з використанням ідей, висунутих Ріманом.

Ріман висловив думку, що існує взаємозв'язок між розподілом на критичній прямій нетривіальних нулів дзета-функції Рімана та асимптотикою розподілу простих чисел.

Георг Фрідріх Бернгард Ріман (нім. *Georg-Friedrich-Bernhard Riemann,* 1826-1866, німецький математик, механік і фізик.

Дослідження Рімана відносяться до теорії функцій комплексного змінного, теорії чисел, геометрії, математичної і теоретичної фізики, теорії диференціальних рівнянь.

У серпні 1859 року Бернгард Ріман став членом-кореспондентом Берлінської академії наук; це була велика честь для 32-річного математика. Згідно з традицією Ріман подав академії роботу на тему досліджень, якими він був у той час зайнятий. Вона називалася «Про кількість простих чисел, що не перевищують даної величини». Він дав інтегральне подання дзета-функції (ζ-функції Рімана), розподіл нулів якої пов'язаний з розподілом простих чисел. Це припущення назвали гіпотезою Рімана.

Гіпотеза Рімана входить до списку семи «проблем XX тисячоліття». За доведення цієї гіпотези Математичний інститут Клея (Кембридж, Массачусетс) обіцяв виплатити приз розміром 1 млн доларів США. Цікаво, що спростування гіпотези (тобто обчислення нетривіального нуля поза «критичною лінією») не дає права отримання призу.

Дзета-функція Рімана – це функція $\zeta(s)$ комплексної змінної $s=\sigma+it$, за умови $\sigma>1$, що визначається за допомогою ряду Діріхле:

$$\zeta(s)=\sum_{n=1}^{\infty}\frac{1}{n^s}.$$

Цей ряд збігається у разі $\mathrm{Re}\,s>1$, є аналітичною функцією від s і має аналітичне продовження на всю комплексну площину, крім особливої точки $s=1$.

$\zeta(s)$ набуває нулевих значень за s, що є від'ємними парними числами: $\zeta(-2)=\zeta(-4)=\zeta(-6)=\ldots=0$ (такі нулі називають «тривіальними» нулями) і за комплексних чисел з $\mathrm{Re}\,s=\frac{1}{2}$ («нетривіальні» нулі дзета-функції Рімана).

Гіпотеза Рімана стосується розташування таких нетривіальних нулів і формулюється таким чином: «Усі нетривіальні нулі дзета-функції мають $\mathrm{Re}\,s=\frac{1}{2}$».

Якщо гіпотеза правильна, то всі нетривіальні нулі дзета-функції Рімана (кількість яких нескінченна) лежать на критичній прямій $\mathrm{Re}\,s=\frac{1}{2}$ комплексних чисел $s=\frac{1}{2}+it$.

Хоча не було знайдено якоїсь закономірності у розподілі простих чисел серед натуральних, Ріман виявив, що функція розподілу простих чисел $\pi(x)$ пов'язана з розподілом нетривіальних нулів дзета-функції.

Якщо гіпотезу Рімана буде доведено, то це приведе до революційних змін наших знань в галузі шифрування та до небаченого прориву в галузі безпеки Інтернету.

В 1901 році Хельге фон Кох показав, що гіпотеза Рімана еквівалентна такому твердженню про розподіл простих чисел:

$$\pi(x)=\mathrm{Li}(x)+O(\sqrt{x}\ln x)\ \text{ при }\ x\to\infty, \qquad (1.2)$$

де $\mathrm{Li}(x)=\int_2^x\frac{1}{\ln t}dt$ – інтегральний логарифм.

Це є більш точним наближенням $\pi(x)$ і відповідає більш регулярному розподілу простих чисел.

Результати більш точного наближення $\pi(x)$ можна побачити в табл. 1.1. Наприклад, для $x=10^7$ формула (1.1) дає результат, який відрізняється від справжньої кількості простих чисел на 44 158, тоді як формула (1.2) забезпечує відмінність лише 339.

Таблиця 1.1 – Значення функцій $\pi(x)$, $x / \ln x$ і $\mathrm{Li}(x)$

x	$\pi(x)$	$\pi(x)$ - $x / \ln x$	$\mathrm{Li}(x) - \pi(x)$	$x / \pi(x)$	$\pi(x) / x$ (частка простих чисел), %
10	4	- 0,3	2,2	2,500	40
10^2	25	3,3	5,1	4,000	25
10^3	168	23	10	5,952	16,8
10^4	1 229	143	17	8,137	12,3
10^5	9 592	906	38	10,425	9,59
10^6	78 498	6 116	130	12,740	7,85
10^7	664 579	44 158	339	15,047	6,65

Одним із застосувань теореми 1.3 є те, що вона дає уявлення про час, який буде потрібний, щоб знайти просте число певного розміру шляхом випадкового пошуку. Багато криптосистем (наприклад, RSA) потребують простих чисел $p \approx 2^{512}$. Виходячи з результату теореми, ймовірність того, що випадково вибране число такого розміру є простим, становить приблизно

$$\frac{1}{\ln 2^{512}} \approx \frac{1}{355} .$$

Якщо пошук простого числа обмежити непарними числами, то приблизно 177 чисел доведеться перевірити на простоту.

З теореми про розподіл простих чисел випливає, що в послідовності простих чисел існують розриви.

Розрив простих чисел (*prime gap*) – це різниця між двома послідовними простими числами.

n-й розрив (позначається g_n) – це різниця між (*n*+1)-м і *n*-м простими числами, тобто $g_n = p_{n+1} - p_n$. Іноді замість g_n розглядають функцію $g(p_n)$.

За визначенням g_n кожне просте число можна записати як

$$p_{n+1} = 2 + \sum_{i=1}^{n} g_i .$$

Перший, найменший і єдиний непарний простий розрив є 1 (різниця між простими числами 3 і 2). Усі інші розриви є парними. Існує лише одна пара послідовних розривів довжини 2: розриви g_2 і g_3 між простими числами 3, 5 і 7.

Для будь-якого цілого числа *n* факторіал *n*! є добутком усіх натуральних чисел до *n* включно. Далі в послідовності

$$n!+2, n!+3, \ldots, n!+n$$

перший доданок ділиться на 2, другий доданок ділиться на 3 і так далі. Таким чином, це послідовність з $n-1$ складених цілих чисел, і вона належить розриву між простими числами довжини не менше *n*. З цього випливає, що між простими числами існують як завгодно великі розриви, тобто для будь-

якого цілого числа N існує ціле число m з $g_m \geq N$.

Приклади розривів між простими числами:

p_{n+1}	3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67
g_n	1, 2, 2, 4, 2, 4, 2, 4, 6, 2, 6, 4, 2, 4, 6, 6, 2, 6.

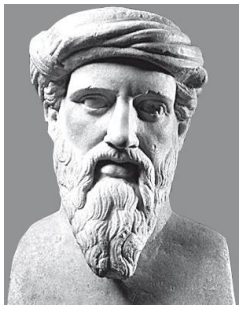
Послідовність розривів у OEIS має номер A001223.

1.4 Графічні подання розташування простих чисел

Математики довго шукали закономірності розташування простих чисел у послідовності натуральних чисел, але спроби були марні. В той самий час, окремі факти змушували думати, що прості числа все-таки якось упорядковані, але закон їх розподілу дуже складний.

У школі Піфагора для дослідження властивостей натуральних чисел використовували двовимірне подання набору чисел, яке нині називають *піфагограмою*.

Для побудови піфагограми послідовність чисел розбивають на рядки певної довжини і записують їх один під одним.



Піфагор (дав.-гр. Πυθαγόρας, 570 – 497 до н. е.) – давньогрецький філософ, релігійний та політичний діяч, засновник піфагореїзму.

Піфагор займає почесне місце в історії математики. Основним змістом піфагорійської математики є вчення про число. Піфагорійці вважали надзвичайно важливими різні властивості чисел і відношення між ними. Вони ввели багато фундаментальних теоретико-числових понять, виявили і дослідили глибокі властивості чисел і поставили такі питання, які й сьогодні залишаються предметом досліджень багатьох учених і все ще чекають свого розв'язання. Найважливішою властивістю чисел піфагорійці вважали парність і непарність та першими ввели поняття парного і непарного числа, простого й складеного, розробили теорію подільності на два, дали кілька класифікацій натуральних чисел.

За життя Піфагор не писав книг і трактатів. Він ділився всіма своїми знаннями з учнями в усній формі, вважаючи, що довіряти таємниці книгам, які можуть бути викрадені, не можна.

Нехай послідовність натуральних чисел розбивається на рядки, що містять шість чисел. Маємо таку піфагограму:

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36
37	38	39	40	41	42
43	44	45	46	47	48

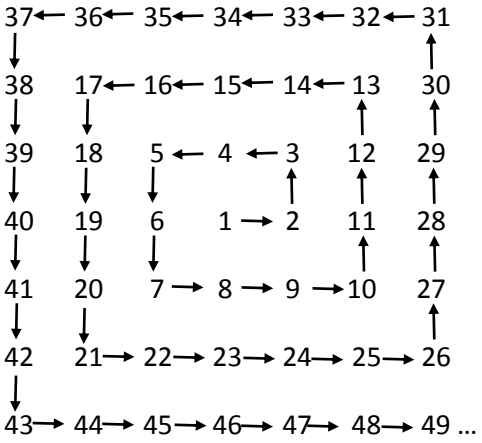
Тут прості числа виділено сірим кольором. Така візуальна картина може сприяти виявленню певних закономірностей. З цієї піфагограми випливає, що прості числа можна обчислювати за формулами: $n = 6x + 1$ і $n = 6x + 5.$

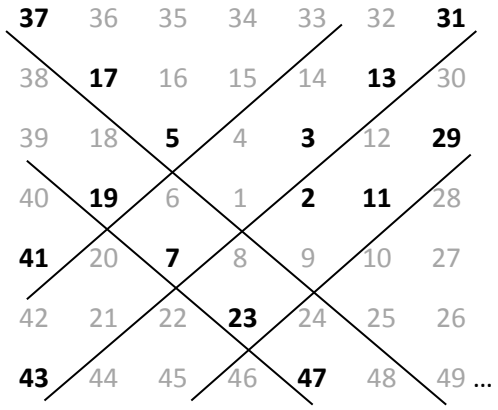
Станіслав Улам запропонував оригінальний спосіб графічного подання послідовності натуральних чисел у вигляді спіралі, який зараз називають *скатертина Улама*.

Якщо відзначити прості числа, то можна побачити, що вони розташовані вздовж діагональних прямих. Ці діагоналі описуються поліномами виду:

$$ax^2 + bx + c,$$

де коефіцієнти *a*, *b*, *c* – цілі числа. Тому скатертина Улама дозволяє візуально визначити поліноми, значення яких найчастіше є простими числами.





Станіслав Улам (1909 – 1984) – польсько-американський математик.

Народився у Львові та був одним з найактивніших представників Львівської математичної школи, які збирались у «Шкотській кав'ярні». Незадовго до початку Другої світової війни вченому вдалося виїхати до США. З 1944 року працював у Лос-Аламоській лабораторії (США), де долучився до створення водневої бомби (разом з Едвардом Теллером).

Він одним із перших почав використовувати комп'ютери для наукових досліджень і всіляко їх пропагувати. Улам сформулював важливі гіпотези в теорії чисел, теорії графів та теорії пакування.

Станіслав Улам написав автобіографічну книгу «Adventures of a Mathematician» (Пригоди математика), український переклад якої опубліковано львівським видавництвом «Літопис».

На рис. 1.1 наведено приклад скатертини Улама, де прості числа, що є значеннями полінома $4x^2 - 2x + 41,$ позначені жирними точками.

Клаубер запропонував трикутне подання послідовності натуральних чисел (рис. 1.2), у якому *n*-й рядок містить числа від $(n-1)^2 + 1$ до n^2. Як і на скатертині Улама, прості числа, що є значеннями поліномів другого степеня, розташовані вздовж прямих ліній. Вертикальні лінії відповідають

поліномам виду $k^2 - k + M$. Висока густина простих чисел відповідає поліному $x^2 - x + 41$.

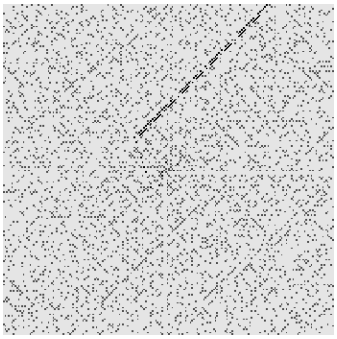


Рисунок 1.1 – Приклад скатертини Улама

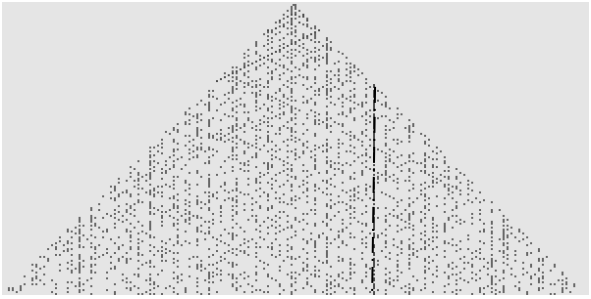


Рисунок 1.2 – Трикутне подання послідовності натуральних чисел

Роберт Сакс запропонував подання послідовності натуральних чисел, у якому числа розташовані по Архімедовій спіралі (рис. 1.3). У спіралі Сакса в кожну спіраль входить така кількість чисел, яка дорівнює подвоєному номеру спіралі. Завдяки цій властивості всі значення поліномів другого степеня повністю розташовані вздовж одного промінця, тоді як на спіралі Улама вони займають два променя.

Розглянуті подання послідовності натуральних чисел на площині створюють певні графічні картини розташування простих чисел, що можуть бути використані для пошуку функцій, значеннями яких є прості числа. Однак їх практичне застосування обмежено відносно невеликими числами.

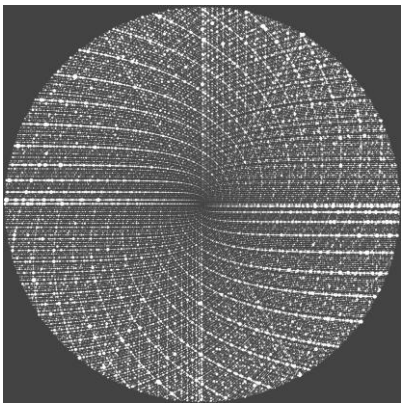


Рисунок 1.3 – Спіраль Сакса

1.5 *k*-кортежі простих чисел

У теорії чисел властивості розподілу простих чисел вивчають зокрема з використанням кортежів.

Означення 1.2. Скінченну сукупність значень, що є повторюваним шаблоном розривів між простими числами, називають *k-кортежем простих чисел* (*prime k-tuples*).

Для *k*-кортежу (*a*,*b*,…) позиції задані набором цілих чисел, усі значення $(n+a, n+b, \ldots)$ яких є простими числами. Зазвичай перше значення в *k*-кортежі дорівнює 0 (тобто $a=0$), а решта – це різні додатні парні числа, які є значеннями g_n.

Наприклад, 3-кортежу з шаблоном (0, 4, 6) відповідає такий набір простих чисел: $p, p+4, p+6$.

У табл. 1.2 наведено *k*-кортежі, які відомі під загальними назвами.

Таблиця 1.2 – *k*-кортежі

k	Назва	Шаблон	Послідовність в OEIS
2	twin primes	(0, 2)	A001359
2	cousin primes	(0, 4)	A023200 і A046132
2	sexy primes	(0, 6)	A023201 і A046117
3	prime triplets	(0, 2, 6) (0, 4, 6)	A022004 A022005
3	sexy prime triplets	(0, 6, 12)	A046118, A046119 і A046120
4	prime quadruplets, prime decade	(0, 2, 6, 8)	A007530
4	sexy prime quadruplets	(0, 6, 12, 18)	A046118
5	prime quintuplets	(0, 2, 6, 8, 12) (0, 4, 6, 10, 12)	A022006 A022007
6	prime sextuplets	(0, 4, 6, 10, 12, 16)	A022008
7	prime septuplets	(0, 2, 6, 8, 12, 18, 20) (0, 2, 8, 12, 14, 18, 20)	A022009 A022010
8	prime octuplets	(0, 2, 6, 8, 12, 18, 20, 26) (0, 2, 6, 12, 14, 20, 24, 26) (0, 6, 8, 14, 18, 20, 24, 26)	A022011 A022012 A022013

Означення 1.3. Пари простих чисел виду $(p, p+2)$ називають *простими−близнюками* (*twin primes*).

Перші кілька пар простих-близнюків:

(3, 5), (5, 7), (11, 13), (17, 19), (29, 31), (41, 43), (59, 61), (71, 73), (101, 103), (107, 109), (137, 139), ... (послідовність A077800 в OEIS).

Кожне третє непарне число ділиться на 3, і тому жодні три послідовних непарних числа не можуть бути простими, якщо одне з них не дорівнює 3. Таким чином, число 5 є єдиним простим числом, яке належить двом парам простих-близнюків.

Усі пари простих-близнюків, крім (3, 5), мають вигляд $(6n-1,6n+1)$. З цього випливає, що сума будь-якої пари простих-близнюків (крім 3 і 5) ділиться на 12.

Доведено, що для будь-якого цілого $m \geq 2$ пара $(m, m+2)$ є парою простих-близнюків тоді і лише тоді, коли

$$4((m-1)!+1) \equiv -m \bmod m(m+2).$$

Відповідь на питання про кількість пар простих-близнюків дає така гіпотеза.

Гіпотеза про прості-близнюки. *Існує нескінченна кількість простих чисел p таких, що p*+2 *також є простим.*

Незважаючи на зусилля відомих математиків наразі ця гіпотеза залишається недоведеною.

Сильніша форма цієї гіпотези постулює закон розподілу для простих-близнюків, подібний до теореми про прості числа.

Нехай $\pi_2(x)$ позначає кількість простих чисел $p \leq x$ таких, що p+2 також є простим.

Гіпотеза Харді–Літтлвуда

$$\pi_2(x) \sim 2C_2 \frac{\mathrm{x}}{(\ln x)^2} \sim 2C_2 \int_2^x \frac{1}{(\ln t)^2} dt.$$

Тут константа C_2 пари простих чисел визначається як

$$C_2 = \prod_{\substack{p \text{ prim}, \\ p \geq 3}} \left(1 - \frac{1}{(p-1)^2}\right) \approx 0{,}660161815846869573927812110014 \ldots .$$

Друге $\sim$ не є частиною гіпотези та підтверджується інтегруванням за частинами.

⚠ **Означення 1.4.** Просте число p таке, що ані p−2, ані p+2 не є простими, називають *ізольованим простим числом* (*isolated prime*).

Іншими словами, p не є частиною пари простих-близнюків. Наприклад, 23 є ізольованим простим числом, оскільки 21 і 25 є складовими.

Перші ізольовані прості числа:

2, 23, 37, 47, 53, 67, 79, 83, 89, 97, 113, 127, 131, 157, 163, 167, 173, 211, 223,… (послідовність A007510 в OEIS).

Майже всі прості числа є ізольованими в тому сенсі, що відношення кількості ізольованих простих чисел, менших за n, і кількості всіх простих чисел, менших за n, прагне до 1, коли $n \to \infty$.

Ізольоване просте число також називають *одиночне просте* (*single prime*) або просте-недвійник (*non-twin prime*).

⚠ **Означення 1.5.** Пари простих чисел виду $(p, p+4)$ називають *двоюрідними простими числами* (*cousin primes*).

Перші кілька пар двоюрідних простих чисел:

(3, 7), (7, 11), (13, 17), (19, 23), (37, 41), (43, 47), (67, 71), (79, 83), (97, 101), (103, 107), (109, 113), (127, 131), (163, 167), (193, 197), (223, 227), (229, 233).

Одне з чисел n, $n + 4$, $n + 8$ завжди буде ділитися на 3. Ось приклади.

$n = 1$: 1, 5, 9 (число 9 ділиться на 3).

$n = 2$: 2, 6, 10 (число 6 ділиться на 3).

$n = 3$: 3, 7, 11 (число 3 ділиться на 3).

$n = 4$: 4, 8, 12 (число 12 ділиться на 3).

$n = 5$: 5, 9, 13 (число 9 ділиться на 3).

Є єдиний випадок $n = 3$, коли всі три числа є простими. З цього випливає, що число 7 є єдиним простим числом, яке належить двом парам двоюрідних простих чисел (3, 7) і (7, 11).

В OEIS двоюрідні прості числа подано двома послідовностями.

Послідовність A023200 «Прості числа p такі, що p+4 також є простим числом»:

3, 7, 13, 19, 37, 43, 67, 79, 97, 103, 109, 127, 163, 193, 223, 229, 277, 307, 313, 349, 379, 397, 439, 457, 463, 487, 499, 613, 643, 673, 739, 757, 769, 823, 853,…

Послідовність A046132 «Більше число p+4 двоюрідних простих чисел»:

7, 11, 17, 23, 41, 47, 71, 83, 101, 107, 113, 131, 167, 197, 227, 233, 281, 311, 317, 353, 383, 401, 443, 461, 467, 491, 503, 617, 647, 677, 743, 761, 773, 827, 857,…

З першої гіпотези Харді–Літтлвуда випливає, що двоюрідні прості числа мають таку саму асимптотичну щільність, як прості-близнюки.

⚠ **Означення 1.6.** Пари простих чисел виду $(p, p+6)$ називають *сексуальними простими числами* (*sexy primes*).

Термін «сексуальні» – це каламбур, що походить від латинського слова «шість»: *sex*.

Перші кілька пар таких простих чисел:

(5,11), (7,13), (11,17), (13,19), (17,23), (23,29), (31,37), (37,43), (41,47), (47,53), (53,59), (61,67), (67,73), (73,79), (83,89), (97,103), (101,107), (103,109),…

В OEIS сексуальні прості числа подано двома послідовностями.

Послідовність A023201 «Прості числа p такі, що p + 6 також є простим числом»:

5, 7, 11, 13, 17, 23, 31, 37, 41, 47, 53, 61, 67, 73, 83, 97, 101, 103, 107, 131, 151, 157, 167, 173, 191, 193, 223, 227, 233, 251, 257, 263, 271, 277, 307, 311, 331,…

Перші кілька пар з використанням чисел цієї послідовності:

(5,11), (7,13), (11,17), (17,23), (31,37), (41,47), (47,53), (61,67), (67,73), (97,103), (101,107), (151,157), (167,173),...

Послідовність A046117 «Прості числа p такі, що p-6 також є простим числом»:

11, 13, 17, 19, 23, 29, 37, 43, 47, 53, 59, 67, 73, 79, 89, 103, 107, 109, 113, 137, 157, 163, 173, 179, 197, 199, 229, 233, 239, 257, 263, 269, 277, 283, 313, 317,…

Перші кілька пар з використанням чисел цієї послідовності:

(11,17), (13,19), (17,23), (23,29), (37,43), (47,53), (53,59), (67,73), (73,79), (103,109), (107,113), (157,163), (173,179),…

⚠ **Означення 1.7.** Трійки простих чисел виду $(p, p+2, p+6)$ або $(p, p+4, p+6)$ називають *простий триплет* (*prime triplets*).

Винятком з цього означення є трійки (2, 3, 5) і (3, 5, 7), оскільки жодні три послідовних непарних числа не можуть бути простими, якщо одне з них не дорівнює 3.

Початкові числа триплетів подано в OEIS двома послідовностями.

Послідовність A022004 «Початкові числа простих триплетів $(p, p+2, p+6)$»:

5, 11, 17, 41, 101, 107, 191, 227, 311, 347, 461, 641, 821, 857, 881, 1091, 1277, 1301, 1427, 1481, 1487, 1607, 1871, 1997, 2081, 2237, 2267, 2657, 2687, 3251,…

Ось кілька триплетів з використанням чисел цієї послідовності:

(5, 7, 11), (11, 13, 17), (17, 19, 23), (41, 43, 47), (101, 103, 107), (107, 109, 113), (191, 193, 197), (227, 229, 233), (311, 313, 317),…

Послідовність A022005 «Початкові числа простих триплетів $(p, p+4, p+6)$»:

7, 13, 37, 67, 97, 103, 193, 223, 277, 307, 457, 613, 823, 853, 877, 1087, 1297, 1423, 1447, 1483, 1663, 1693, 1783, 1867, 1873, 1993, 2083, 2137, 2377, 2683,…

Перші триплети з використанням чисел цієї послідовності:

(7, 11, 13), (13, 17, 19), (37, 41, 43), (67, 71, 73), (97, 101, 103), (103, 107, 109), (193, 197, 199), (223, 227, 229), (277, 281, 283), (307, 311, 313),…

Подібно до гіпотези про пари простих-близнюків, передбачається, що існує нескінченна кількість простих трійок.

⚠ **Означення 1.8.** Четвірку простих чисел виду $(p, p+2, p+6, p+8)$ називають *проста четвірка* (*prime quadruplets*).

Послідовність A007530 в OEIS «Прості четвірки: числа k такі, що k+2, k+6, k+8 є простими»:

5, 11, 101, 191, 821, 1481, 1871, 2081, 3251, 3461, 5651, 9431, 13001, 15641,

15731, 16061, 18041, 18911, 19421, 21011, 22271, 25301, 31721, 34841,…

Ось кілька перших простих четвірок:

{5, 7, 11, 13}, {11, 13, 17, 19}, {101, 103, 107, 109}, {191, 193, 197, 199}, {821, 823, 827, 829}, {1481, 1483, 1487, 1489}, {1871, 1873, 1877, 1879},…

Усі прості четвірки, крім {5, 7, 11, 13}, мають форму $\{30n+11,\ 30n+13,\ 30n+17,\ 30n+19\}$ для деякого цілого числа *n*. Така структура гарантує, що жодне з чотирьох простих чисел не ділиться на 2, 3 або 5.

Просту четвірку цієї форми також називають *простою декадою* (*prime decade*).

Невідомо, чи існує нескінченна кількість простих четвірок. Доведення того, що їх нескінченно багато, означатиме гіпотезу про пари простих-близнюків. Однак поточні знання стверджують, що може існувати нескінченно багато пар простих-близнюків і лише скінченна кількість простих четвірок.

⚠ **Означення 1.9.** 5-кортежі простих чисел виду $(p, p+2, p+6, p+8, p+12)$ або $(p, p+4, p+6, p+10, p+12)$ називають *прості п'ятірки* (*prime quintuplets*).

Початкові числа простих п'ятірок подано в OEIS двома послідовностями.

Послідовність A022006 «Початкові числа простих 5-кортежів $(p, p+2, p+6, p+8, p+12)$»:

5, 11, 101, 1481, 16061, 19421, 21011, 22271, 43781, 55331, 144161, 165701, 166841, 195731, 201821, 225341, 247601, 268811, 326141, 347981, 361211,…

Ось кілька простих п'ятірок з використанням чисел цієї послідовності:

{5, 7, 11, 13, 17}, {11, 13, 17, 19, 23}, {101, 103, 107, 109, 113},
{1481, 1483, 1487, 1489, 1493}, {16061, 16063, 16067, 16069, 16073},
{19421, 19423, 19427, 19429, 19433}, {21011, 21013, 21017, 21019, 21023}.

Усі такі прості п'ятірки, крім {5, 7, 11, 13, 17}, мають форму $\{30n+11, 30n+13, 30n+17, 30n+19, 30n+23\}$ для деякого цілого числа *n*.

Послідовність A022007 «Початкові числа простих 5-кортежів $(p, p+4, p+6, p+10, p+12)$»:

7, 97, 1867, 3457, 5647, 15727, 16057, 19417, 43777, 79687, 88807, 101107, 257857, 266677, 276037, 284737, 340927, 354247, 375247, 402757, 419047,…

Перші прості п'ятірки з використанням чисел цієї послідовності:

{7, 11, 13, 17, 19}, {97, 101, 103, 107, 109}, {1867, 1871, 1873, 1877, 1879},
{3457, 3461, 3463, 3467, 3469}, {5647 , 5651, 5653, 5657, 5659},…

Усі такі прості п'ятірки мають форму $\{30n+7,\ 30n+11,\ 30n+13,\ 30n+17, 30n+19\}$ для деякого цілого числа *n*.

Невідомо, чи існує нескінченна кількість простих п'ятірок. Знову ж

таки, доведення гіпотези про пари простих-близнюків може необов'язково доводити, що існує також нескінченна кількість простих п'ятірок. Крім того, доведення існування нескінченної кількості простих четвірок, необов'язково означатиме, що існує нескінченна кількість простих п'ятірок.

⚠ **Означення 1.10.** 6-кортежі простих чисел виду $(p, p+4, p+6, p+10, p+12, p+16)$ називають *прості шістки* (*prime sextuplets*).

Послідовність A022008 «Початкові числа простих 6-кортежів $(p, p+4, p+6, p+10, p+12, p+16)$»:

7, 97, 16057, 19417, 43777, 1091257, 1615837, 1954357, 2822707, 2839927, 3243337, 3400207, 6005887, 6503587, 7187767, 7641367, 8061997, 8741137,…

Ось кілька простих шісток:

{7, 11, 13, 17, 19, 23}, {97, 101, 103, 107, 109, 113},
{16057, 16061, 16063, 16067, 16069, 16073},
{19417, 19421, 19423, 19427, 19429, 19433},
{43777, 43781, 43783, 43787, 43789, 43793}.

Усі прості шістки, крім {7, 11, 13, 17, 19, 23}, мають форму $\{210n+97, 210n+101, 210n+103, 210n+107, 210n+109, 210n+113\}$ для деякого цілого числа *n*. Така структура гарантує, що жодне з шести простих чисел не ділиться на 2, 3, 5 або 7.

Невідомо, чи існує нескінченна кількість простих шісток. Як і в попередніх випадках, доведення гіпотези про пари простих-близнюків необов'язково може довести, що існує також нескінченна кількість простих шісток. Крім того, доведення того, що існує нескінченна кількість простих п'ятірок, необов'язково може довести, що існує нескінченна кількість простих шісток.

? КОНТРОЛЬНІ ПИТАННЯ

1. Хто заклав основи теорії чисел?
2. Яке ціле число називають простим?
3. Що стверджує лема Евкліда?
4. Яке твердження є еквівалентною формою леми Евкліда?
5. Сформулюйте узагальнення леми Евкліда.
6. Скільки існує простих чисел?
7. Який розподіл простих чисел описує теорема простих чисел?
8. Що таке асимптотична нотація для простих чисел?
9. Хто висловив ідею обчислення асимптотики розподілу простих чисел?
10. Що лежить в основі гіпотези Рімана для простих чисел?

11. Як обчислюється функція розподілу простих чисел за інтегральним логарифмом?

12. Яке основне застосування теореми розподілу простих чисел в криптосистемах?

13. Що називають розривом простих чисел?

14. Як обчислюють розрив між простими числами?

15. Що називають піфагорамою?

16. Чим відрізняється скатертина Улама від піфагограми?

17. Назвіть графічні подання розташування простих чисел.

18. Які закономірності розташування простих чисел простежуються в графічних поданнях?

19. Сформулюйте означення *k*-кортежу простих чисел.

20. Які прості числа називають близнюками?

21. Назвіть просте число, яке єдине одночасно належить двом парам простих-близнюків.

22. Яку форму мають усі пари простих-близнюків, крім пари (3,5)?

23. На яке число ділиться сума будь-якої пари простих-близнюків, крім пари (3,5)?

24. За яких умов пара (m, m+2) є парою простих-близнюків?

25. Скільки існує пар простих-близнюків?

26. Яке число називають ізольованим простим числом?

27. Чи є ізольоване просте число частиною пари простих-близнюків?

28. Які числа називають двоюрідними простими числами?

29. Що таке простий триплет?

30. Яку форму мають трійки простих чисел?

31. Що означає шаблон (0, 2, 6) 3-кортежу простих чисел?

32. Які трійки простих чисел є винятком з означення простого триплету?

33. Яке число одночасно входить до трійок послідовних непарних чисел?

34. Яка форма чисел простої четвірки?

35. Яку форму мають прості п’ятірки?

36. Яку форму мають прості шістки?

ВПРАВИ

1. Назвіть прості числа за функцією $\pi(20)$.

2. Скільки простих чисел вказує функція $\pi(100)$?

3. Назвіть перший, найменший і єдиний непарний простий розрив.

4. Назвіть пари простих чисел, між якими існує розрив довжини 2.

5. Назвіть прості числа, між якими існує два послідовних розриви довжини 2.

6. Для чисел від 1 до 48 побудуйте піфагограму з рядками, що містять чотири числа.

7. Для чисел від 1 до 50 побудуйте піфагограму з рядками, що містять чотири числа і шість чисел.

8. Побудуйте скатертину Улама для чисел від 1 до 100.

9. Побудуйте трикутне подання послідовності натуральних чисел від 1 до 50.

ДЖЕРЕЛА ДЛЯ ПОГЛИБЛЕНОГО ВИВЧЕННЯ

1. Goldfeld, Dorian. «The elementary proof of the prime number theorem: an historical perspective». In Chudnovsky, David; Chudnovsky, Gregory; Nathanson, Melvyn (eds.). Number theory (New York, 2003). New York: Springer-Verlag. pp. 179-192.

2. Cornaros, Charalambos; Dimitracopoulos, Costas. «The prime number theorem and fragments of PA». *Archive for Mathematical Logic*. 33 (4): (1994), pp. 265-281.

3. D. Goldston, J. Pintz and C. Yıldırım, «Primes in tuples». *Ann. of Math*. 170 (2009), no. 2, pp. 819–862.

4. D. Hensley and I. Richards, «Primes in intervals», *Acta Arith*. 25 (1973/74), pp. 375–391.

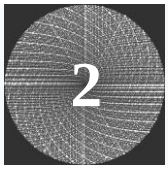
5. M. Wolf, «On the Twin and Cousin primes», http://www.ift.uni.wroc.pl/~mwolf/

6. R. P. Brent, «Irregularities in the distribution of primes and twin primes», *Math. Comp.*, 29 (1975) pp. 43-56.

7. Twin prime https://t5k.org/glossary/page.php?sort=TwinPrime

8. R. P. Brent, «The distribution of small gaps between succesive primes», *Math. Comp.*, 28 (1974) pp. 315-324.

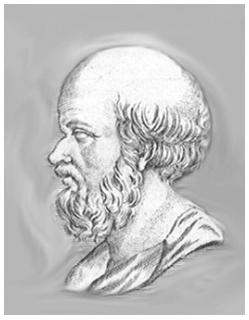
9. T. Forbes. Prime k-tuplets. http://www.ltkz.demon.co.uk/ktuplets.htm.



ПОШУК ПРОСТИХ ЧИСЕЛ

Для знаходження початкового набору простих чисел, що не перевищують певного числа *n*, використовують детерміновані алгоритми, які називають решето Ератосфена, решето Сундарама, решето Аткіна та ін.

Назва «решето» походить від того, що за часів Ератосфена числа писали на дошці, покритій воском, і проколювали дірочки в тих місцях, де були написані складені числа. Тому дощечка нагадувала решето, яким «просіювалися» всі складені числа, а залишалися лише прості числа.



Ератосфен (грец. Ἐρατοσθένης; 276 – 194 до н. е.) – грецький математик, астроном, географ, філолог і поет. Перший відомий вчений, який обчислив розміри Землі. За багатогранність наукової діяльності сучасники дали йому прізвисько *Пентатл*, тобто *Багатоборець*. Широко відомий трактат *Решето* (*Koskonon*). У ньому науковець виклав спрощену методику визначення простих чисел (так зване «решето Ератосфена»).

2.1 Алгоритм «Решето Ератосфена»

2.1.1 Базовий алгоритм

Початок. Виписати послідовно всі числа від 2 до *n*.

Кроки алгоритму.

1. Число 2 є простим.
2. Присвоїти змінній *p* значення, що є простим числом.
3. Закреслити в послідовності всі числа від 2*p* до *n* з кроком *p* (числа 2*p*, 3*p*, 4*p*, …).
4. Перше число більше за *p*, яке залишилося незакресленим, є простим.
5. Повторювати кроки 2,3 і 4 поки можливо.

Результат. Числа, які залишилися незакресленими є простими.

Приклад 2.1. Знайти набір простих чисел, що не перевищують *n* = 29.

Розв'язання. Початкова послідовність:

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29.

p = 2. Закреслюємо всі числа, які більші за 2 і кратні 2:
4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28.

2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ 9 ~~10~~ 11 ~~12~~ 13 ~~14~~ 15 ~~16~~ 17 ~~18~~ 19 ~~20~~ 21 ~~22~~ 23 ~~24~~ 25 ~~26~~ 27 ~~28~~ 29.

p = 3. Закреслюємо всі числа, які більші за 3 і кратні 3:
6, 9, 12, 15, 18, 21, 24, 27.

2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~ ~~21~~ ~~22~~ 23 ~~24~~ 25 ~~26~~ ~~27~~ ~~28~~ 29.

p = 5. Закреслюємо всі числа, які більші за 5 і кратні 5:
10, 15, 20, 25.

2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~ ~~21~~ ~~22~~ 23 ~~24~~ ~~25~~ ~~26~~ ~~27~~ ~~28~~ 29.

p = 7. Чисел, які більші за 7 і кратні 7, немає. Звершення просіювання.

Відповідь: 2 3 5 7 11 13 17 19 23 29.

У наведеному прикладі деякі складені числа були закреслені кілька разів. Наприклад, числа 12, 15, 18 і 20 закреслені двічі. Зауважимо, що кількість закреслень складеного числа – це кількість простих дільників без урахування їх кратності.

Решето Ератосфена є достатньо ефективним алгоритмом, оскільки не передбачає використання «важких» операцій, на кшталт ділення і множення. Визначення чисел, що закреслюються, здійснюється з кроком p.

Існує кілька модифікацій решета Ератосфена.

2.1.2 Решето, що просіює, починаючи з квадратів простих чисел

Це алгоритм, в якому на кроці 3 числа можна закреслювати, починаючи відразу з числа p^2, оскільки всі менші числа, кратні p, обов'язково мають простий дільник менше p, а вони вже попередньо закреслені. І, відповідно, зупиняти алгоритм можна, коли $p^2 > n$. Крім того, всі прості числа, крім 2, є непарними, і тому для них можна використовувати кроки по $2p$, починаючи з p^2. Отже, закреслюються числа: p^2, p^2+2p, p^2+4p,…

Приклад 2.2. Знайти набір простих чисел, що не перевищують n = 31.

Розв'язання. Початкова послідовність:

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

p = 2. Закреслюємо числа:
4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30.

2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ 9 ~~10~~ 11 ~~12~~ 13 ~~14~~ 15 ~~16~~ 17 ~~18~~ 19 ~~20~~ 21 ~~22~~ 23 ~~24~~ 25 ~~26~~ 27 ~~28~~ 29 ~~30~~ 31

p = 3. Закреслюємо числа:
9, 15, 21, 27.

2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~ ~~21~~ ~~22~~ 23 ~~24~~ 25 ~~26~~ ~~27~~ ~~28~~ 29 ~~30~~ 31

p = 5. Закреслюємо число 25.

2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~ ~~21~~ ~~22~~ 23 ~~24~~ ~~25~~ ~~26~~ ~~27~~ ~~28~~ 29 ~~30~~ 31

p = 7. 7^2 >31. Звершення просіювання.

Відповідь: 2 3 5 7 11 13 17 19 23 29 31.

2.1.3 Решето для непарних чисел

Оскільки серед простих чисел є лише одне парне число 2, то можна просіювати тільки непарні числа. У послідовності 3, 5, 7, 9, 11, 13, 15,… закреслюють числа 3n, 5n, 7n, 9n, 11n, 13n, 15n,… для n = 3, 5, 7, 9, 11, 13, 15,…

Приклад 2.3. Знайти набір простих чисел, що не перевищують n=47.

Розв'язання. Початкова послідовність:

3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47.

p = 3. Закреслюємо числа:
9, 15, 21, 27, 33, 39, 45.

3 5 7 ~~9~~ 11 13 ~~15~~ 17 19 ~~21~~ 23 25 ~~27~~ 29 31 ~~33~~ 35 37 ~~39~~ 41 43 ~~45~~ 47.

p = 5. Закреслюємо числа:
15, 25, 35, 45.

3 5 7 ~~9~~ 11 13 ~~15~~ 17 19 ~~21~~ 23 ~~25~~ ~~27~~ 29 31 ~~33~~ ~~35~~ 37 ~~39~~ 41 43 ~~45~~ 47.

p = 7. Чисел, які більші за 7 і кратні 7, немає. Звершення просіювання.

Відповідь: 3 5 7 11 13 17 19 23 29 31 37 41 43 47.

2.2 Решето Ейлера

Це решето відрізняється від решета Ератосфена тим, що кожне закреслене (складене) число видаляється з послідовності.

На кожному етапі алгоритму перше число в послідовності береться як просте число і всі кратні йому числа позначаються для наступного видалення. Після цього з послідовності видаляють перше число і всі позначені числа. Процес завершується коли для першого числа не буде знайдено кратних йому чисел.

Приклад 2.4. Знайти набір простих чисел, що не перевищують n=29, використовуючи решето Ейлера.

Розв'язання. Початкова послідовність:

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29.

Виділяємо перше число 2. Позначаємо всі числа, які кратні 2:

(2) 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ 9 ~~10~~ 11 ~~12~~ 13 ~~14~~ 15 ~~16~~ 17 ~~18~~ 19 ~~20~~ 21 ~~22~~ 23 ~~24~~ 25 ~~26~~ 27 ~~28~~ 29.

Видаливши всі позначені числа, отримаємо послідовність:

3 5 7 9 11 13 15 17 19 21 23 25 27 29.

Виділяємо перше число 3. Позначаємо всі числа, які кратні 3:

(3) 5 7 ~~9~~ 11 13 ~~15~~ 17 19 ~~21~~ 23 25 ~~27~~ 29.

Видаливши всі позначені числа, отримаємо послідовність:

5 7 11 13 17 19 23 25 29.

Виділяємо перше число 5. Позначаємо всі числа, які кратні 5:

(5) 7 11 13 17 19 23 ~~25~~ 29.

Видаливши всі позначені числа, отримаємо послідовність:

7 11 13 17 19 23 29.

Виділяємо перше число 7.

(7) 11 13 17 19 23 29.

Відсутні числа, які кратні 7. Завершення процесу просіювання.

Відповідь: 2 3 5 7 11 13 17 19 23 29.

2.3 Алгоритм «Решето Сундарама»

Початок. Виписати послідовно всі натуральні числа від 1 до *N*.

Кроки алгоритму.

1. Вилучити з послідовності всі числа виду: $i + j + 2ij$,

де $i \le j$; $i = 1,2,\ldots,\left\lfloor \frac{\sqrt{2N+1}-1}{2} \right\rfloor$; $j = i, i+1,\ldots,\left\lfloor \frac{N-i}{2i+1} \right\rfloor$.

2. Для кожного числа *n*, що залишилося, обчислити просте число за формулою:

$$p = 2n + 1.$$

Числа, що вилучаються, можна отримати, побудувавши таблицю в такий спосіб (табл. 2.1).

Таблиця 2.1 – Числа, що вилучаються

a=4; *d*=3	4	7	10	13	16	19	…
a=7; *d*=5	7	12	17	22	27	32	…
a=10; *d*=7	10	17	24	31	38	45	…
a=13; *d*=9	13	22	31	40	49	58	…
a=16; *d*=11	16	27	38	49	60	71	…
a=19; *d*=13	19	32	45	58	71	84	
…	…	…	…	…	…		…

Перший рядок (і стовпець) складаються з членів арифметичної

прогресії, яка починається з a=4 з кроком d=3. Решта рядків також складаються з членів арифметичних прогресій, для яких a визначаються вихідною послідовністю. Для другого і всіх наступних рядків d є непарним цілим числом у порядку зростання, починаючи з d=5.

Приклад для N =24.

Розв'язання. Початкова послідовність:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24.

Вилучення чисел, наведених в таблиці:

1 2 3 ~~4~~ 5 6 ~~7~~ 8 9 ~~10~~ 11 ~~12~~ ~~13~~ 14 15 ~~16~~ ~~17~~ 18 ~~19~~ 20 21 ~~22~~ 23 ~~24~~.

Послідовність чисел, що залишилися:

1 2 3 5 6 8 9 11 14 15 18 20 21 23.

Для кожного числа n, що залишилося, обчислюємо просте число за формулою $p = 2n + 1$ і маємо послідовність простих чисел.

Відповідь: 3 5 7 11 13 17 19 23 29 31 37 41 43 47.

Решето Ератосфена та його модифікації, а також решето Сундарама, базуються на використанні подання чисел у вигляді добутку xy. На відміну від цього, Аткін запропонував алгоритм, що базується на використанні квадратичних форм подання чисел ($ax^2 + by^2$).

2.4 Решето Аткіна

Алгоритм передбачає обчислення залишку від ділення чисел на 60 (обчислення за модулем 60) і визначення кількості розв'язків рівнянь певного виду.

Складеними є всі числа, що дорівнюють (за модулем 60):

0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,

44, 46, 48, 50, 52, 54, 56 і 58 (діляться на 2);

3, 9, 15, 21, 27, 33, 39, 45, 51 і 57 (діляться на 3);

5, 25, 35 і 55 (діляться на 5).

Числа, що дорівнюють (за модулем 60):

1, 13, 17, 29, 37, 41, 49 і 53

є простими тоді і лише тоді, коли кількість розв'язків рівняння $4x^2 + y^2 = n$ непарна і саме число n не кратне жодному квадрату простого числа.

Числа, що дорівнюють (за модулем 60): 7, 19, 31 і 43, є простими тоді і лише тоді, коли кількість розв'язків рівняння $3x^2 + y^2 = n$ непарна і саме число n не кратне жодному квадрату простого числа.

Числа, що дорівнюють (за модулем 60): 11, 23, 47 і 59, є простими тоді і лише тоді, коли кількість розв'язків рівняння $3x^2 - y^2 = n$ (для $x > y$) непарна і саме число n не кратне жодному квадрату простого.

Окремий крок алгоритму викреслює числа, кратні квадратам простих чисел. Оскільки жодне з цих чисел не ділиться на 2, 3 і 5, то перевірка, що число не кратне квадрату простого числа, не містить 2^2, 3^2 і 5^2.

2.5 Формули для простих чисел

Прості числа розташовані в послідовності натуральних чисел незрозумілим до цього часу чином. Тому ще з давніх часів математики прагнули знайти «формулу для простих чисел».

Адрієн-Марі Лежандр (фр. *Adrien-Marie Legendre*; 1752–1833) – французький математик, член Французької академії наук.

Дослідження Лежандра присвячені математичному аналізу, теорії чисел, небесній механіці та теорії геодезичних вимірів. В галузі математичного аналізу ввів так звані багаточлени Лежандра, досліджував інтеграли першого і другого роду, першим відкрив і застосував в обчисленнях метод найменших квадратів, широко застосовуваний нині.

Своїми роботами Лежандр зробив основний внесок у створення теорії чисел. Сформулював (1808) закон розподілу простих чисел, дещо пізніше дав перший послідовний і повний виклад теорії чисел.

Французький математик Лежандр висунув гіпотезу, що якщо a і b взаємно прості, то в арифметичній прогресії з першим членом b і різницею a знаходиться нескінченна кількість простих чисел. Ця гіпотеза була доведена німецьким математиком Діріхле.

Формула, за якою обчислюються такі прості числа, має вигляд:

$$n = ax + b,$$

де a, b і x – натуральні числа.

Наведемо приклади послідовностей чисел, серед яких є прості числа (виділені жирним шрифтом).

$n = 2x + 1.$

3 5 7 9 **11 13** 15 **17 19** 21 **23** 25 27 **29 31** 33 35 **37** 39 **41**…

$n = 4x + 1.$

5 9 **13 17** 21 25 **29** 33 **37 41** 45 49 **53** 57 **61** 65 69 **73** 77…

$n = 4x + 3.$

3 7 11 15 **19 23** 27 **31** 35 39 **43 47 51** 55 **59** 63 **67 71** 75…

$n = 6x + 1$.

7 13 19 25 **31 37 43** 49 55 **61 67 73 79** 85 91 **97 103**…

Йоганн Петер Густав Лежен Діріхле (нім. *Johann Peter Gustav Lejeune Dirichlet*; 1805 – 1859) – німецький математик.

Діріхле зробив значний внесок у математичний аналіз, теорію функцій комплексної змінної та теорію чисел.

У теорії чисел широко використовуються ряд Діріхле, найвідомішим прикладом якого є дзета-функція Рімана, а також L-функція Діріхле, згортка Діріхле для арифметичних функцій, формула Діріхле для кількості дільників числа.

Принцип Діріхле (принцип голубів і кліток – pigeonhole principle) застосовується в інформатиці. Наслідками цього принципу є те, що алгоритм гешування, незалежно від того, як він працює, не може уникнути однакових значень індексів, а для будь-якого алгоритму ущільнення без втрат, знайдеться файл, який не може бути ущільнений.

⚠ **Означення 2.1.** Прості числа виду $4n+1$ називають *простими числами Піфагора* (*Pythagorean primes*).

Свою назву ці числа отримали через те, що, за аналогією з теоремою Піфагора, вони можуть бути подані в вигляді суми двох квадратів.

Ось декілька перших простих чисел Піфагора:

5, 13, 17, 29, 37, 41, 53, 61, 73, 89, 97, 101, 109, 113, … (послідовність A002144 в OEIS).

Приклади подання простих чисел Піфагора в вигляді суми двох квадратів:

$$5 = 2^2 + 1^2,\ 13 = 3^2 + 2^2,\ 17 = 4^2 + 1^2,\ 29 = 5^2 + 2^2,\ 37 = 6^2 + 1^2.$$

Леонард Ейлер вивчав поліном $x^2 + x + 41$, який для x = 0, 1, 2, …, 39 дає прості числа. Наприклад, для $x = 20$ маємо просте число $20^2 + 20 + 41 = 461$. Однак для $x = 40$ його значенням є складене число $1681 = 41 \cdot 41$.

В OEIS ці прості числа становлять послідовність A005846:

41, 43, 47, 53, 61, 71, 83, 97, 113, 131, 151, 173, 197, 223, 251, 281, 313, 347,

383, 421, 461, 503, 547, 593, 641, 691, 743, 797, 853, 911, 971, 1033, 1097, …

Поліном $x^2 - x + 41$ дає такі саме прості числа для x = 1, 2, …, 40.

Після того, як Ейлер показав таку можливість, пошук формул для обчислення простих чисел став популярною математичною розвагою.

Багато математиків досліджували поліноми виду:

$$ax^2 + bx + c,$$

де a, b, c – цілі числа.

Ось приклади таких поліномів:

$36x^2 - 810x + 2753$, $x = 0,1,2,\ldots,44$ (послідовність A050268 в OEIS);

$47x^2 - 1701x + 10181$, $x = 0,1,2,\ldots,42$ (послідовність A050267 в OEIS).

Досліджувалися також поліноми третього, четвертого, п'ятого і шостого степенів.

Означення 2.2. Прості числа, які є різницею двох послідовних кубів чисел, називають *кубові прості числа* (c*uban primes*).

Перші кілька кубових простих чисел виду 1:

7, 19, 37, 61, 127, 271, 331, 397, 547, 631, 919, 1657, 1801, 1951, 2269, 2437,...

(послідовність A002407 в OEIS).

Якщо для обчислення цих чисел використовується формула $n^3 - (n-1)^3$, то n набуває значення:

2, 3, 4, 5, 7, 10, 11, 12, 14, 15, 18, 24, 25, ... (послідовність A002504 в OEIS).

Також ці числа можуть бути обчислені за формулою $3y^2 + 3y + 1$ для $y = 1,2,3,\ldots$

Кубові прості числа виду 2 є розв'язком рівняння:

$$p = \frac{x^3 - y^3}{x - y},\ x = y + 2,\ y > 0,$$

що спрощується до $3y^2 + 6y + 4$. За допомогою заміни $y = n - 1$ це можна записати як $3n^2 + 1$, де n набуває значення з послідовності A002504 в OEIS.

Перші кілька кубових простих чисел виду 2:

13, 109, 193, 433, 769, 1201, 1453, 2029, 3469, 3889, 4801, 10093, 12289,...

(послідовність A002648 в OEIS).

Використовуючи сучасну комп'ютерну техніку, сьогодні можна емпірично аналізувати ймовірність отримати для різних поліномів прості значення на різноманітних інтервалах.

До цього часу не відомо, чи існує поліном (крім лінійного), серед значень якого є нескінченна кількість простих чисел. Доведено, що не існує поліномів, значення яких є тільки прості числа.

Крім поліномів запропоновано ряд інших формул.

Формула Вільсона:

$$f(n) = \left\lfloor \frac{n! \bmod (n+1)}{n} \right\rfloor (n-1) + 2,$$

де $\lfloor\ \rfloor$ – округлення до найближчого меншого цілого числа;

n – додатне ціле число.

Згідно з теоремою Вільсона, коли $n+1$ є простим числом, перший множник у добутку стає одиницею, і формула дає просте число $n+1$. Але коли $n+1$ не є простим числом, перший множник стає нулем і формула дає просте число 2.

Нехай $n=10$.

$$f(10)=\left\lfloor \frac{10!\bmod(11)}{10} \right\rfloor(10-1)+2=\left\lfloor \frac{10}{10} \right\rfloor(10-1)+2=1\cdot 9+2=11.$$

Маємо просте число 11.

Нехай $n=14$.

$$f(14)=\left\lfloor \frac{14!\bmod(15)}{14} \right\rfloor(14-1)+2=\left\lfloor \frac{0}{14} \right\rfloor(14-1)+2=0\cdot 13+2=2.$$

Ця формула не є ефективним способом генерування простих чисел, оскільки обчислення $n!\bmod(n+1)$ потребує приблизно $n-1$ множень і обчислень за модулем $n+1$.

Міллс (W. H. Mills) довів, що існує таке дійсне число A, що, кщо

$$d_n=A^{3^n},$$

то

$$\lfloor d_n\rfloor=\left\lfloor A^{3^n} \right\rfloor$$

є простим числом для всіх натуральних чисел n.

Якщо гіпотеза Рімана справджується, то найменше таке A має значення приблизно 1,306377883863080690468614492 6... (послідовність A051021 в OEIS) і відоме як константа Міллса. Це значення дає початок простим числам $\lfloor d_1\rfloor=2$, $\lfloor d_2\rfloor=11$, $\lfloor d_3\rfloor=1361$,... (послідовність A051254 в OEIS).

Варто відзначити, що ця формула не має практичної цінності, оскільки не існує відомого способу обчислення константи без пошуку простих чисел.

Іншу формулу, що породжує прості числа, подібну до формули Міллса, запропонував Райт (E. M. Wright). Він довів, що існує таке дійсне число α, що, якщо

$$g_0=\alpha \text{ і } g_{n+1}=2^{g_n} \text{ для } n\geq 0,$$

то

$$\lfloor g_n\rfloor=\left\lfloor 2^{\cdot\cdot 2^{2^{\alpha}}} \right\rfloor$$

є простим для всіх $n\geq 1$.

Райт наводить константу $\alpha=1,9287800$, яка дає початок простим числам $\lfloor g_1\rfloor=\lfloor 2^{\alpha}\rfloor=3$, $\lfloor g_2\rfloor=13$ і $\lfloor g_3\rfloor=16381$. $\lfloor g_4\rfloor$ є парним числом.

Цю послідовність простих чисел не можна продовжити, не знаючи більше цифр константи α. Як і формула Міллса, і з тих самих причин,

формула Райта не має практичної цінності для знаходження простих чисел.

Плуфф (Simon Plouffe) запропонував набір формул, подібних до формули Міллса, які мають такий вигляд:

$$\left\lfloor a_0^{r^n} \right\rfloor,$$

де $\lfloor\ \rfloor$ означає округлення до найближчого цілого числа.

Наприклад, якщо $a_0 \approx 43{,}80468771580293481$ і $r = 5/4$, то це дає 113, 367, 1607, 10177, 102217...

Використовуючи $a_0 = 10^{500} + 961 + \varepsilon$ і r = 1,01 з $0 < \varepsilon < 0{,}5$, Плуфф виявив, що можна створити послідовність з 50 ймовірно простих чисел. Імовірно, існує таке ε, що ця формула дасть нескінченну послідовність справжніх простих чисел.

Наведені формули Міллса, Райта і Плуффа підтверджують, що пошук формул для обчислення простих чисел став популярною математичною розвагою.

2.6 Послідовності Люка

Означення 2.3. Функцію $f : \mathbf{N} \to \mathbf{X}$, яка визначена на множині натуральних чисел і набуває значення на об'єктах довільної природи, називають *послідовністю*.

У загальному випадку послідовність записується у вигляді $\{x_1, x_2, \ldots, x_n, \ldots\}$ або $\{x_n\}$.

Елементи $x_1, x_2, \ldots$ називають *членами послідовності*.

Кожний індекс вказує порядковий номер члена послідовності.

Залежно від виду елементів, послідовності поділяють на числові та функціональні.

Означення 2.4. Послідовність дійсних чисел, тобто відображення, яке кожному натуральному числу n ставить у відповідність дійсне число x_n, називають ч*исловою послідовністю*.

Означення 2.5. Послідовність, яка має скінченну кількість членів, називають *скінченною послідовністю*.

Записом такої послідовності може бути:

$$\{x_1, x_2, \ldots, x_n\} \text{ або } \{x_i\}_{i=1}^{n}.$$

Означення 2.6. Послідовність, яка має нескінченну кількість членів, називають *нескінченною послідовністю*.

Для запису нескінченної послідовності використовують таке:

$$\{x_1, x_2, ..., x_n, ...\} \text{ або } \{x_i\}_{i=1}^{\infty}.$$

Щоб задати послідовність, потрібно вказати спосіб, за допомогою якого можна знайти будь-який її член.

1. Послідовність можна задати описом знаходження її членів.
2. Скінченну послідовність можна задати переліком її членів.
3. Послідовність можна задати таблицею, в якій навпроти кожного члена послідовності вказують його порядковий номер.
4. Послідовність можна задати формулою, за якою можна знайти будь-який член послідовності, знаючи його номер.
5. Спочатку вказати перший або кілька перших членів послідовності, а потім умову, за якою можна визначити будь-який член послідовності за попередніми. Такий спосіб задання послідовності називають рекурентним. Інакше кажучи, для таких послідовностей окрім формули, яка виражає x_{n+1} через $x_1, x_2, ..., x_n$ необхідно вказати один або декілька перших членів. За обчислення таких членів відбувається «повернення назад» (рекурсія).

⚠ **Означення 2.7.** Числові послідовності, члени яких є цілими числами, називають *цілочисловими послідовностями*.

Існує величезна кількість таких послідовностей, але є кілька послідовностей, що мають властивості, корисні для розв'язання певних задач криптографії. Ці послідовності утворюються на основі базових послідовностей, які запропонував Едуард Люка.

Франсуа Едуар Анатоль Люка
(фр. *François Édouard Anatole Lucas*; 1842-1891) – французький математик.
Свої найважливіші роботи Едуар Люка зробив у теорії чисел. Він розробив методи перевірки простоти чисел. У 1857 році, у віці 15 років, Люка почав перевіряти простоту числа 2^{127}-1 вручну, використовуючи послідовності Люка. У 1876 році, після 19 років перевірок, він нарешті довів, що 2^{127}-1 є простим числом. Це залишиться найбільшим відомим простим числом Мерсенна, перевіреним вручну.
Люка відомий своїми дослідженнями послідовності Фібоначчі. Пов'язані з нею послідовності Люка і числа Люка названі його ім'ям.

⚠ **Означення 2.8.** Сімейство пар лінійних рекурентних цілочислових послідовностей $\{U_n(P,Q)\}$ і $\{V_n(P,Q)\}$, що задовольняють рекурентне співвідношення:

$$x_n = P \cdot x_{n-1} - Q \cdot x_{n-2};\ n \geq 2,$$

де коефіцієнти P і Q – фіксовані цілі числа, називають *послідовностями Люка*.

Будь-яку послідовність, що задовольняє це рекурентне співвідношення,

можна подати як лінійну комбінацію послідовностей Люка $\{U_n(P,Q)\}$ і $\{V_n(P,Q)\}$.

За наявності двох параметрів P і Q послідовності Люка першого роду $\{U_n(P,Q)\}$ і другого роду $\{V_n(P,Q)\}$ визначаються рекурентними співвідношеннями:

$U_0(P,Q)=0$, $U_1(P,Q)=1$,

$U_n(P,Q)=P\cdot U_{n-1}(P,Q)-Q\cdot U_{n-2}(P,Q)$ для $n>1$;

$V_0(P,Q)=2$, $V_1(P,Q)=P$, $V_n(P,Q)=P\cdot V_{n-1}(P,Q)-Q\cdot V_{n-2}(P,Q)$ для $n>1$.

Для практичних задач корисними є такі співвідношення ($n > 0$):

$$U_n(P,Q)=\frac{P\cdot U_{n-1}(P,Q)+V_{n-1}(P,Q)}{2};$$

$$V_n(P,Q)=\frac{(P^2-4Q)\cdot U_{n-1}(P,Q)+P\cdot V_{n-1}(P,Q)}{2}.$$

Ці співвідношення можна подати у матричній формі таким чином:

$$\begin{bmatrix} U_n(P,Q) \\ U_{n+1}(P,Q) \end{bmatrix}=\begin{bmatrix} 0 & 1 \\ -Q & P \end{bmatrix}\cdot\begin{bmatrix} U_{n-1}(P,Q) \\ U_n(P,Q) \end{bmatrix};$$

$$\begin{bmatrix} V_n(P,Q) \\ V_{n+1}(P,Q) \end{bmatrix}=\begin{bmatrix} 0 & 1 \\ -Q & P \end{bmatrix}\cdot\begin{bmatrix} V_{n-1}(P,Q) \\ V_n(P,Q) \end{bmatrix};$$

$$\begin{bmatrix} U_n(P,Q) \\ V_n(P,Q) \end{bmatrix}=\begin{bmatrix} P/2 & 1/2 \\ (P^2-4Q)/2 & P/2 \end{bmatrix}\cdot\begin{bmatrix} U_{n-1}(P,Q) \\ V_{n-1}(P,Q) \end{bmatrix}.$$

Початкові члени послідовностей $\{U_n(P,Q)\}$ і $\{V_n(P,Q)\}$ наведено в табл. 2.2.

Таблиця 2.2 – Початкові члени послідовностей $\{U_n(P,Q)\}$ і $\{V_n(P,Q)\}$

n	$U_n(P,Q)$	$V_n(P,Q)$
0	0	2
1	1	P
2	P	P^2-2Q
3	P^2-Q	P^3-3PQ
4	P^3-2PQ	$P^4-4P^2Q+2Q^2$
5	$P^4-3P^2Q+Q^2$	$P^5-5P^3Q+5PQ^2$

Для розв'язання деяких задач доцільно використовувати рекурентні співвідношення з метою послідовного обчислення членів послідовностей, а розв'язання певних задач потребує безпосереднього обчислення конкретних членів послідовностей, заданих індексом *n*. Останнє забезпечується в такий спосіб.

Рекурентним співвідношенням для елементів послідовностей Люка відповідає характеристичне рівняння:

$$x^2 - Px + Q = 0,$$

яке має дискримінант $D = P^2 - 4Q$ і корені:

$$a = \frac{P + \sqrt{D}}{2} \text{ і } b = \frac{P - \sqrt{D}}{2}.$$

Враховуючи це, маємо:

$$a + b = P;$$

$$a - b = \sqrt{D};$$

$$ab = \frac{1}{4}(P^2 - D) = Q.$$

Варто відзначити, що послідовність $\{a^n\}$ і послідовність $\{b^n\}$ також задовольняють рекурентне співвідношення. Однак це можуть бути не цілочислові послідовності.

Якщо $D \neq 0$, то маємо різні значення *a* і *b* та можемо визначити:

$$a^n = \frac{V_n + U_n\sqrt{D}}{2},\ b^n = \frac{V_n - U_n\sqrt{D}}{2}.$$

З цього випливає, що члени послідовностей Люка можна обчислити безпосередньо (не використовуючи рекурсію) таким чином:

$$U_n = \frac{a^n - b^n}{a - b} = \frac{a^n - b^n}{\sqrt{D}};$$

$$V_n = a^n + b^n.$$

Випадок $D = 0$ виникає тоді, коли $P = 2S$ і $Q = S^2$ для деякого цілого *S*. У цьому випадку $a = b = S$, а отже, нескладно обчислити:

$$U_n(P,Q) = U_n(2S, S^2) = nS^{n-1};$$

$$V_n(P,Q) = V_n(2S, S^2) = 2S^n.$$

Для розв'язання практичних задач корисними є такі властивості членів послідовностей.

$U_{km}(P,Q)$ є кратним $U_m(P,Q)$, тобто послідовність $\{U_m(P,Q)\}_{m\geq 1}$ є послідовністю подільності. Це означає, зокрема, що $U_n(P,Q)$ може бути

простим лише тоді, коли n є простим.

Існує алгоритм швидкого обчислення $U_n(P,Q)$ і $V_n(P,Q)$ для великих значень n, який є аналогічним алгоритму піднесення до степеня за допомогою піднесення до квадрата.

Одним із застосувань послідовностей Люка в криптографії є перевірка простоти чисел. Комбінація ймовірнісного тесту Міллера-Рабіна та ймовірнісного тесту Люка є досить ефективною, оскільки жодне складене число не пройде обидва тести, а сумарний час тестування менший за час тестування будь-яким іншим тестом. Крім цього, послідовності Люка використовуються в деяких методах доведення простоти.

Послідовності Люка також використовуються для побудови систем шифрування та підпису з відкритим ключем, в яких замість піднесення до степеня використовується обчислення за модулем складеного числа певного члена цієї послідовності.

Криптосистема з відкритим ключем LUC реалізує аналоги ElGamal (LUCELG), Diffie–Hellman (LUCDIF) і RSA (LUCRSA).

LUCRSA має приблизно таку саму безпеку, як і RSA, для такого самого розміру ключа, але приблизно вдвічі повільніша. LUCDIF і LUCELG порівняно з Diffie–Hellman та ElGamal, для того самого рівня безпеки потребують удвічі менший модуль, оскільки їх безпека базується на дискретному логарифмі $GF(p^2)$, а не $GF(p)$. Через менший застосовуваний модуль вони на 50 – 100 відсотків швидші. Однак криптосистеми, основані на послідовності Люка, не отримали такої ретельної перевірки, як більш популярні, основані на піднесенні до степеня, тому рекомендують використовувати їх з обережністю.

Послідовності Люка для деяких значень P і Q мають спеціальні назви:

$\{U_n(1,-1)\}$ – послідовність чисел Фібоначчі (Fibonacci numbers);

$\{V_n(1,-1)\}$ – послідовність чисел Люка (Lucas numbers);

$\{U_n(2,-1)\}$ – послідовність чисел Пелла (Pell numbers);

$\{U_n(3,2)\}$ – послідовність чисел Мерсенна (Mersenne numbers) $2^n - 1$;

$\{U_n(x,-1)\}$ – послідовність поліномів Фібоначчі (Fibonacci polynomials);

$\{V_n(x,-1)\}$ – послідовність поліномів Люка (Lucas polynomials).

Числа Люка визначаються за рекурентною формулою

$$L_n = L_{n-1} + L_{n-2}$$

з початковими значеннями $L_0 = 2$ і $L_1 = 1$.

Кілька перших чисел Люка:

2, 1, 3, 4, 7, 11, 18, 29, 47, 76, 123, 199, 322, 521, 843, 1364, 2207, 3571, 5778,...

(послідовність A000032 в OEIS).

Числа Фібоначчі визначаються за рекурентною формулою

$$F_n = F_{n-1} + F_{n-2}$$

з початковими значеннями $F_0 = 0$ і $F_1 = 1$.

Леонардо Пізанський (італ. *Leonardo Pisano*, близько 1170 – 1250), відоміший як **Фібоначчі** (*Fibonacci*) – італійський математик.

Фібоначчі вважають одним з найвидатніших західних математиків Середньовіччя. Він є автором математичного трактату «Книга абака» (Liber abaci, 1202). Ця книга містить майже всі арифметичні й алгебраїчні відомості того часу, викладені з винятковою повнотою і глибиною. Вона відіграла значну роль у розвитку математики в Західній Європі протягом кількох наступних століть. Саме за цією книгою європейці знайомилися з десятковою системою числення. У «Книзі абака» Фібоначчі дослідив властивості чисел рекурентної послідовності, яка була відома ще в Стародавній Індії. Саме завдяки цьому дослідженню отримали назви – послідовність Фібоначчі та числа Фібоначчі.

Основну роль у своїх книгах Фібоначчі віддає задачам, їх розв'язкам і коментарям. Ці задачі він складав для математичних турнірів, які в ті часи культивував імператор Священної Римської імперії Фрідріх II замість кривавих лицарських турнірів. На таких математичних змаганнях супротивники обмінювалися не ударами, а задачами.

Кілька перших чисел Фібоначчі:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181,…

(послідовність A000045 в OEIS).

У деяких старих визначеннях чисел Фібоначчі початкове значення $F_0 = 0$ вилучено, тому послідовність починається з $F_1 = F_2 = 1$.

Числа Пелла визначаються за рекурентною формулою

$$P_n = 2P_{n-1} + P_{n-2}$$

з початковими значеннями $P_0 = 0$ і $P_1 = 1$.

Джон Пелл (англ. *John Pell*, застаріле – **Пелль** або **Пель**; 1611 – 1685) – англійський математик, член Лондонського королівського товариства з 1663 р.

Колом його інтересів були теорія чисел, математика і мовознавство. Він займався дослідженнями алгебраїчних рівнянь та складанням математичних таблиць квадратів. У 1638 році Пелл звернув на себе увагу математичного співтовариства своєю книгою «Ідея математики» і почав жваве листування з Мерсенном та іншими видатними вченими. Улюбленою темою Пелла було розв'язання діофантових рівнянь – цій темі він присвятив цикл лекцій в Амстердамському університеті. В теорій чисел відомими є рівняння Пелла та числа Пелла.

Кілька перших чисел Пелла:

0, 1, 2, 5, 12, 29, 70, 169, 408, 985, 2378, 5741, 13860, 33461, 80782, 195025, …

(послідовність A000129 в OEIS).

2.7 Спеціально сконструйовані прості числа

Існують класи цілих додатних чисел, які визначають за допомогою певних математичних співвідношень і в яких серед множини всіх чисел є прості числа. Такі прості числа називають *спеціально сконструйованими простими числами*.

⚠ **Означення 2.9.** Прості числа виду $p_n\#\pm1$, де $p_n\#$ – прайморіал, називають *прайморіальними простими числами* (*primorial prime*).

Прайморіал (*Primorial*) – це функція простого числа p_n, значення якої дорівнює добутку перших *n* простих чисел і яка позначається $p_n\#$.

$$p_n\# = \prod_{k=1}^{n} p_k ,$$

де p_k – *k*-е просте число.

Наприклад, значення $p_6\#$ дорівнює:

$$p_6\# = 2\cdot3\cdot5\cdot7\cdot11\cdot13 = 30030.$$

В OEIS прайморіали подано послідовністю A002110:

1, 2, 6, 30, 210, 2310, 30030, 510510, 9699690, 223092870, 6469693230,…

Прайморіали відіграють важливу роль у пошуку простих чисел в арифметичних прогресіях із простих чисел. Наприклад, сума 2236133941+23# є простим числом, з якого починається послідовність з тринадцяти простих чисел, які можна отримати, послідовно додаючи 23#, і закінчується числом 5136341251.

⚠ **Означення 2.10.** Числа виду $p_n\#+1$ (не обов'язково прості) називають *числами Евкліда* (*Euclid number*).

Тести простоти показують, що числа:

$p_n\#+1$ є простими для *n* = 1, 2, 3, 4, 5, 11, … (послідовність A014545 в OEIS).

Наприклад, для *n* = 3 і $p_3 = 5$ маємо:

$$p_n\#+1 = 5\#+1 = 2\cdot3\cdot5+1 = 30+1 = 31.$$

Ось декілька перших чисел Евкліда:

3, 7, 31, 211, 2311, 30031, 510511 (послідовність A006862 в OEIS).

⚠ **Означення 2.11.** Числа виду $p_n\#-1$ називають *числами Куммера* (*Kummer number*).

Іноді числа Куммера називають числами Евкліда другого роду.

Тести простоти показують, що числа $p_n\#-1$ є простими для n = 2, 3, 5, 6, 13, 24, … (послідовність A057704 в OEIS).

Наприклад, для $n = 5$ і $p_5 = 11$ маємо:

$$p_n\#-1 = 11\#-1 = 2\cdot 3\cdot 5\cdot 7\cdot 11-1 = 2310-1 = 2309.$$

В OEIS числа Куммера подано послідовністю A057588:

1, 5, 29, 209, 2309, 30029, 510509, 9699689, 223092869, 6469693229,…

Як і у випадку з числами Евкліда, невідомо, чи існує нескінченна кількість простих чисел Куммера. Перше з цих чисел, яке є складеним, це 209.

Софі Жермен (фр. *Marie-Sophie Germain*; 1776 – 1831) – французька вчена, математик, механік і філософ.

Зробила вагомий внесок у диференціальну геометрію, теорію чисел і механіку. За дослідження згинання пластинок у теорії пружності їй, першій з жінок, присуджено премію Паризької академії наук, а результати її досліджень використано під час будівництва Ейфелевої вежі.

Вона довела окремий випадок Великої теореми Ферма для простих чисел, які сьогодні називають числами Софі Жермен.

⚠ **Означення 2.12.** Просте число p таке, що число $2p+1$ є також простим, називають *простим числом Софі Жермен* (*Sophie Germain prime*).

⚠ **Означення 2.13.** Просте число виду $2p+1$, де p – просте число Софі Жермен, називають *безпечним простим числом* (*safe prime*).

Наприклад, 23 – це просте число Софі Жермен, а $2\cdot 23+1 = 47$ – пов’язане з ним безпечне просте число.

Ось кілька простих чисел Софі Жермен:

2, 3, 5, 11, 23, 29, 41, 53, 83, 89, 113, 131, 173, 179, 191, 233, 239, 251, 281, 293,

359, 419, 431, 443, 491, 509, 593, 641, 653, 659, 683, 719, 743, 761, 809, ...

(послідовність A005384 в OEIS), які породжують такі безпечні прості числа:

5, 7, 11, 23, 47, 59, 83, 107, 167, 179, 227, 263, 347, 359, 383, 467, 479, 503, 563,

587, 719, 839, 863, 887, 983, 1019, 1187, 1283, 1307, 1319, 1367, 1439, 1487, ...

(послідовність A005385 в OEIS).

Безпечні прості числа більші за 7 мають вигляд $12k-1$. Наприклад, $59=12\cdot 5-1$.

Було припущення, що простих чисел Софі Жермен нескінченно багато, але це залишається недоведеним.

Прості числа Софі Жермен і безпечні прості числа застосовуються в криптографії з відкритим ключем та тестуванні на простоту.

Безпечні прості числа використовують у підходах, основаних на дискретних логарифмах, зокрема в алгоритмі Діффі-Хеллмана. Якщо $2p+1$ безпечне просте число, то мультиплікативна група чисел за модулем $2p+1$ має підгрупу високого порядку. Саме така підгрупа забезпечує високий рівень криптографічної стійкості.

Прості числа виду $2p+1$ називають безпечними через те, що вони мають зв'язок із сильними простими числами.

У стандарті ANSI X9.31 висувається вимога щодо використання сильних простих чисел в алгоритмах RSA. Криптосистема RSA використовує модуль $n=pq$, де p і q непарні прості числа. Ці числа мають бути достатньої розрядності, щоб було практично неможливо розкласти модуль n на множники. Крім того, вони мають вибиратися випадковим чином. Водночас ймовірність будь-якого окремого простого числа, що вибирається, має бути достатньо малою, щоб не дати супротивнику можливості сформувати стратегію пошуку на основі такої ймовірності.

Криптосистема RSA висуває ще додаткові обмеження на вибір p і q, які забезпечують її стійкість до криптоаналітичних атак. Ці обмеження сформульовано через поняття сильного простого числа.

⚠ **Означення 2.14.** Просте число p називають *сильним простим* (*strong prime*), якщо існують такі цілі числа r, s і t, що виконуються три умови:

1. $p-1$ має великий простий множник r.
2. $p+1$ має великий простий множник s.
3. $r-1$ має великий простий множник t.

Для безпечного простого числа $q=2p+1$ число $q-1$ має великий дільник, а саме p, тому q задовольняє частково критерію сильного простого числа.

⚠ **Означення 2.15.** Просте число p, таке що $p+2$ є або простим, або напівпростим числом, називають *простим числом Ченя* (*Chen prime*).

Кілька перших простих чисел Ченя:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 47, 53, 59, 67, 71, 83, 89, 101, … (послідовність A109611 в OEIS) .

Прості числа Ченя названі на честь Чень Цзінжуня, який довів у 1966

році, що таких простих чисел нескінченно багато. Цей результат також випливає з істинності гіпотези про числа прості-близнюки, оскільки молодший член пари простих чисел-близнюків за означенням є простим числом Ченя.

Кілька перших простих чисел Ченя, які не є молодшими членами пари простих чисел-близнюків:
2, 7, 13, 19, 23, 31, 37, 47, 53, 67, 83, 89, 109, 113, 127, ... (послідовність A063637 в OEIS).

У теорії чисел теорема Ченя (Chen's theorem) стверджує, що кожне достатньо велике парне число можна записати як суму або двох простих чисел, або простого і напівпростого числа (добуток двох простих чисел).

Наприклад, $100 = 47 + 53 = 23 + 7 \cdot 11.$

Це послаблена форма гіпотези Гольдбаха, яка стверджує, що кожне парне число є сумою двох простих чисел.

⚠ **Означення 2.16.** Просте число p виду $p = \frac{2^q + 1}{3}$, де q – просте число, називають *простим числом Вагстафа* (*Wagstaff prime*).

Перші три прості числа Вагстафа дорівнюють 3, 11 і 43, тому що:

$$\frac{2^3 + 1}{3} = 3;\ \frac{2^5 + 1}{3} = 11 \text{ і } \frac{2^7 + 1}{3} = 43.$$

Ось кілька значень числа q:

3, 5, 7, 11, 13, 17, 19, 23, 31, 43, 61, … (послідовність A000978 в OEIS),

які породжують прості числа Вагстафа:

3, 11, 43, 683, 2731, 43691, 174763, 2796203, 715827883, 2932031007403,

768614336404564651, … (послідовність A000979 в OEIS).

⚠ **Означення 2.17.** Числа виду $M_n = 2^n - 1$, де n – натуральне число, називають *числами Мерсенна* (*Mersenne numbers*).

Якщо n складене число, $n = kl$ і $k, l > 1$, то й M_n теж складене.
Це випливає з такого розкладання:
$2^n - 1 = 2^{kl} - 1 = (2^k - 1)(2^{k(l-1)} + 2^{k(l-2)} + \ldots + 1).$

⚠ **Означення 2.18.** Прості числа виду $M_p = 2^p - 1$, де p – просте число, називають *простими числами Мерсенна* (*Mersenne primes*).

Однак не всі прості p породжують прості числа Мерсенна M_p.
Наприклад, для p=11 маємо $M_{11} = 2047 = 23 \cdot 89.$
Будь-який дільник складеного числа M_n для простого n має вигляд

$2nk+1$, де k – натуральне число. В наведеному прикладі $23 = 2 \cdot 11 \cdot 1 + 1$ (k =1) і $89 = 2 \cdot 11 \cdot 4 + 1$(k =4).

Марен Мерсенн (фр. *Marin Mersenne*; 1588 – 1648) – французький математик, фізик, філософ і теолог.

Марен Мерсенн був талановитим вченим-універсалом: йому належать дослідження в області математики, фізики і теорії музики. Він богослов і філософ, який поєднував життя в монастирі з координуванням наукового життя Європи першої половини XVII століття активно переписуючись практично з усіма видатними вченими того часу.

В історію математики вчений увійшов завдяки названим на його честь «числам Мерсенна», які сьогодні відіграють важливу роль в теорії чисел, криптографії та для побудови генераторів псевдовипадкових чисел.

Для того, щоб підкреслити, що мова йде саме про прості числа Мерсенна, використовують таке позначення: $M_p = 2^p - 1$, де p – просте число.

Приклад значень p для простих чисел Мерсенна:

2, 3, 5, 7, 13, 17, 19, 31, … (послідовність A000043 в OEIS)

і отриманих з їх використанням простих чисел Мерсенна:

3, 7, 31, 127, 8191, 131 071, 524 287, 2 147 483 647, …(послідовність A000668 в OEIS).

Перевірка простоти чисел Мерсенна виконується досить ефективним тестом Люка-Лемера. Тому прості числа Мерсена є найбільш відомими простими числами.

Арифметика за модулем числа Мерсенна особливо ефективна на двійковому комп'ютері, що робить їх популярними, коли потрібен простий модуль, наприклад генератор випадкових чисел Парка–Міллера. Щоб знайти примітивний поліном порядку чисел Мерсенна, потрібно знати факторизацію цього числа, тому прості числа Мерсенна дозволяють знайти примітивні поліноми дуже високого порядку. Такі примітивні триноми використовуються в генераторах псевдовипадкових чисел з дуже великими періодами, таких як «вихор Мерсенна», узагальнений регістр зсуву та генератори Фібоначчі з відставанням.

Прості числа Мерсенна тісно пов'язані з досконалими числами. Евклід показав, що число виду $\frac{M_p \cdot (M_p + 1)}{2} = (2^p - 1) \cdot 2^{p-1}$, де M_p – просте число Мерсенна, є досконалим.

⚠ **Означення 2.19.** Числа виду $F_n = 2^{2^n} + 1$, де $n \geq 0$ – ціле число, називають *числами Ферма* (*Fermat numbers*).

Перші числа Ферма:

3, 5, 17, 257, 65537, 4294967297, 18446744073709551617,… (послідовність A000215 в OEIS).

Лише для n = 0, 1, 2, 3, 4 числа Ферма є простими (3, 5, 17, 257, 65537), а для $n > 4$ – складеними.

П'єр де Ферма (фр. *Pierre de Fermat*; 1601 – 1665) – французький математик, засновник аналітичної геометрії і теорії чисел.

П'єр де Ферма – найзагадковіша постать у науковому світі XVII століття. Він вільно володів шістьма мовами (французькою, латинською, окситанською, класичною грецькою, італійською та іспанською). Ферма був юристом за освітою, тому математика була радше хобі, ніж професією. Проте, він вніс важливий внесок в аналітичну геометрію, теорію ймовірності, теорію чисел. Ферма передавав більшість своїх робіт у листах друзям, часто практично без доведень своїх теорем. У деяких з цих листів він досліджував багато фундаментальних ідей числення ще до Ньютона або Лейбніца. Працюючи над «Арифметикою» Діофанта, він суттєво розвинув теорію чисел, поклавши початок розділу математики – теорії алгебраїчних чисел, яка виникла внаслідок спроб довести деякі сформульовані, але не доведені самим П'єром Ферма теореми.

Прості числа Ферма особливо корисні для генерування псевдовипадкових послідовностей чисел у діапазоні $1,...,2^l$, де $l > 1$.

Обчислення виконуються за формулою:

$$V_{j+1} = AV_j \bmod p,$$

де p – просте число Ферма;

$A > \sqrt{p}$ і є примітивним коренем за модулем p.

Початкове значення V_0 вибирають з діапазону $1,...,(p-1)$.

Це корисно в інформатиці, оскільки більшість структур даних мають елементи з 2^l можливими значеннями. Наприклад, байт має 256 (2^8) можливих значень (0 – 255). Таким чином, щоб заповнити байт або байти псевдовипадковими значеннями, можна використовувати генератор псевдовипадкових чисел, який виробляє значення від 1 до 256, початковим значенням байта є 1. З цієї причини дуже великі прості числа Ферма становлять особливий інтерес для шифрування даних. Цей метод створює лише псевдовипадкові значення, оскільки після $p-1$ кроків послідовність повторюється. Невдало вибраний множник A може призвести до повторення послідовності раніше, ніж $p-1$.

⚠ **Означення 2.20.** Прості числа виду $2^u 3^v + 1$ для деяких додатних цілих чисел u і v називають *простими числами Пірпонта* (*Pierpont primes*).

Вони названі на честь математика Джеймса Пірпонта (James Pierpont).

Кілька перших простих чисел Пірпонта:

2, 3, 5, 7, 13, 17, 19, 37, 73, 97, 109, 163, 193, 257, 433, 487, 577, 769,… (послідовність A005109 в OEIS).

Якщо $u > 0$ і $v > 0$, то просте число Пірпонта має вигляд $6k + 1$.

Якщо $u = 0$ і $v > 0$, то $3^v + 1$ є парним числом, більшим за 2, а отже, складеним.

Якщо $v = 0$ і u є степенем числа 2, то це робить просте число Пірпонта простим числом Ферма.

У рамках триваючого всесвітнього пошуку множників чисел Ферма деякі прості числа Пірпонта були оголошені множниками. Наприклад, просте число Пірпонта $2^{67}3^2 + 1$ є дільником числа Ферма $2^{2^{63}} + 1$.

Існує 36 простих чисел Пірпонта, менших за 10^6, 59 менших за 10^9, 151 менших за 10^{20} і 789 менших за 10^{100}. Гіпотетично існує $O(\log N)$ простих чисел Пірпонта, менших за N, на відміну від припущеного $O(\log\log N)$ простих чисел Мерсенна в цьому діапазоні. Тобто простих чисел Пірпонта більше ніж простих чисел Мерсенна.

Відомі також прості числа виду $2^u3^v - 1$ для деяких цілих чисел $u, v \geq 0$. Ось кілька таких простих чисел:

2, 3, 5, 7, 11, 17, 23, 31, 47, 53, 71, 107, 127, 191, 383, 431, 647, 863, 971,…

(послідовність A005105 в OEIS).

⚠ **Означення 2.21.** Цілі числа, що обчислюються за формулою $4^n + 2^{n-1} - 1$, називають *числами Кінея* (*Kynea number*).

Еквівалентною формулою є $(2^n + 1)^2 - 2$.

⚠ **Означення 2.22.** Числа Кінея, які є простими, називають *простими числами Кінея* (*Kynea primes*).

Перші прості числа Кінея:

2, 7, 23, 79, 1087, 66047, 263167, 16785407, 1073807359, 17180131327,…

(послідовність A091514 в OEIS).

⚠ **Означення 2.23.** Прості числа, що обчислюються за формулою $2^a \pm 2^b \pm 1$, де $0 < b < a$, називають *простими числами Солінаса* (*Solinas Primes*).

Перші прості числа Солінаса:

3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 47, 59, 61, 67, 71, 73, 79, 97,… (послідовність A165255 в OEIS).

Ці прості числа дозволяють використовувати швидкі алгоритми модульної редукції та широко використовуються в криптографії.

Чотири з рекомендованих простих чисел у документі NIST «Рекомендовані еліптичні криві для використання федеральним урядом» є простими числами Солінаса:

- (*p*-192) $2^{192} - 2^{64} - 1;$
- (*p*-224) $2^{224} - 2^{96} + 1;$
- (*p*-256) $2^{256} - 2^{224} + 2^{192} + 2^{96} - 1;$
- (*p*-384) $2^{386} - 2^{128} - 2^{96} + 2^{32} - 1.$

⚠ **Означення 2.24.** Прості числа, розташовані в послідовності простих чисел на позиціях, номери яких є простими числами, називають *суперпростими числами* (*superprimes*) або *простими числами вищого порядку* (*higher-order prime numbers*).

Їх також називають простими числами з простими індексами $a(n) = p_{p_n}$.

Наприклад, у послідовності простих чисел:

2, **3**, **5**, 7, **11**, 13, **17**, 19, 23, 29, **31**, 37, **41**, 43, 47, 53, **59**, 61, **67**, 71, 73, 79, **83**,

на позиціях з номерами 2, 3, 5, 7, 11, 13, 17, 19, 23 розташовані такі суперпрості числа:

3, 5, 11, 17, 31, 41, 59, 67, 83, … (послідовність A006450 в OEIS).

Усі прості числа Фібоначчі мають індекс *n*, який є простим числом, за винятком $F_4 = 3$. Однак не кожен простий індекс *p* дає просте число Фібоначчі F_p. Наприклад, $F_{19} = 4181 = 37 \times 113.$

Перші індекси простих чисел Фібоначчі:

3, 4, 5, 7, 11, 13, 17, 23, 29, 43, 47, …(послідовність A001605 в OEIS)

і відповідні їм прості числа Фібоначчі:

2, 3, 5, 13, 89, 233, 1597, 28657, 514229, 433494437, 2971215073,…

(послідовність A005478 в OEIS).

Кілька перших простих чисел Люка:

2, 3, 7, 11, 29, 47, 199, 521, 2207, 3571, 9349, 3010349, 54018521, 370248451, ...

(послідовність A005479 в OEIS).

Індекси цих простих чисел:

0, 2, 4, 5, 7, 8, 11, 13, 16, 17, 19, 31, 37, 41, 47, ... (послідовність A001606 в OEIS).

Перші кілька простих чисел Пелла:

2, 5, 29, 5741, 33461, 44560482149, 1746860020068409,

6848040646216128746 9, ... (послідовність A086383 в OEIS).

Індекси цих простих чисел:

2, 3, 5, 11, 13, 29, 41, 53, ... (послідовність A096650 в OEIS).

Як і у випадку з простими числами Фібоначчі, прості числа Пелла мають індекс, який є простим числом.

2.8 Прості числа з властивостями їх подання в системі числення

⚠ **Означення 2.25.** Прості числа, сума цифр яких є простим числом, називають *адитивними простими числами* (*additive primes*).

Наприклад, сума цифр простого числа 43 дорівнює 4+3=7, і це є просте число.

Перші адитивні прості числа:

2, 3, 5, 7, 11, 23, 29, 41, 43, 47, 61, 67, 83, 89, 101, 113, 131, 137, 139, 151,…

(послідовність A046704 в OEIS).

⚠ **Означення 2.26.** Прості числа, добуток цифр яких є простим числом, називають *мультиплікативними простими числами* (*multiplicative primes*).

Наприклад, добуток цифр простого числа 71 дорівнює $7 \cdot 1 = 7$, і це є просте число.

Перші мультиплікативні прості числа:

2, 3, 5, 7, 13, 17, 31, 71, 113, 131, 151, 211, 311, 1117, 1151, 1171, 1511,…

(послідовність A046703 в OEIS).

Нехай суму квадратів цифр натурального числа s_0 подано s_1. Подібним чином нехай суму квадратів цифр s_1 буде подано s_2 і так далі. Така ітерація завжди врешті-решт досягає одного з 10 чисел 0, 1, 4, 16, 20, 37, 42, 58, 89 або 145 (послідовність A039943 в OEIS).

Якщо $s_i = 1$ для деякого $i \geq 1$, то вихідне число s_0 називають *щасливим числом* (*happy number*).

Наприклад, починаючи з 23, отримаємо: $2^2 + 3^2 = 13;\ \ 1^2 + 3^2 = 10;$ $1^2 + 0^2 = 1$. Отже, 23 – щасливе число.

З іншого боку, 5 не є щасливим числом, оскільки маємо послідовність: $5^2 = 25;\ \ 2^2 + 5^2 = 29;\ \ 2^2 + 9^2 = 85;\ \ 8^2 + 5^2 = 89$, в який число 89 є ознакою цього. На певному кроці процес обчислень почне циклічно повторюватися і ніколи не досягне 1.

Число, яке не є щасливим, називають *сумним* (*sad*) або *нещасним* (*unhappy*).

Якщо число щасливе, то всі члени його послідовності щасливі числа, а

для нещасного числа всі члени послідовності є нещасними числами.

У загальному випадку розглядають *b*-щасливі числа, тобто числа подані в системі числення з основою *b*.

Ось кілька перших 10 *b*-щасливих чисел:

1, 7, 10, 13, 19, 23, 28, 31, 32, 44, 49, 68, 70, 79, 82, 86, 91, 94, 97, 100, 103,…

(послідовність A007770 в OEIS).

b-щасливі прості числа (*b-happy primes*) є одночасно *b*-щасливими і простими числами.

Перші 10 *b*-щасливі прості числа:

7, 13, 19, 23, 31, 79, 97, 103, 109, 139, 167, 193, 239, 263, 293, 313, 331, 367, …

⚠ **Означення 2.27.** Просте число, квадрат якого більший за добуток будь-яких двох простих чисел на однаковій кількості позицій перед і після нього в послідовності простих чисел, називають *гарним простим числом* (*good prime*).

Гарне просте число задовольняє нерівність: $p_n^2 > p_{n-i} \cdot p_{n+i}$ для всіх $1 \le i \le n-1$.

Наприклад, перші прості числа – 2, 3, 5, 7 і 11. Оскільки для числа 5 виконуються умови:

$$5^2 > 3 \cdot 7;$$
$$5^2 > 2 \cdot 11,$$

то 5 є гарним простим числом.

Перші гарні прості числа:

5, 11, 17, 29, 37, 41, 53, 59, 67, 71, 97, 101, 127, 149, 179, 191, 223, 227, 251,...

(послідовність A028388 в OEIS).

⚠ **Означення 2.28.** Число, яке залишається простим за кожної перестановки його цифр, називають *переставним простим числом* (*permutable prime*).

Наприклад, число 337 є переставним, тому що кожне з чисел 337, 373 і 733 є простими.

У десятковій системі числення єдиними переставними простими числами є 2, 3, 5, 7, 13, 17, 37, 79, 113, 199, 337 та їхні перестановки:

(13→31), (17→71), (37→73), (79→97), (113→131→311), (199→991→919),

(337→373→733).

Переставні прості числа не можуть мати цифри 0, 2, 4, 6, 8 або 5. Якщо наймолодша цифра числа є 0, 2, 4, 6 або 8, то це парне число, а якщо цифра 5, то таке число ділиться на 5.

Кожне переставне просте число є майже повторюваною цифрою, тобто це перестановка цілого числа виду:

$$B_n(a,b) = aaa...ab,$$

де a і b – різні цифри з набору {1, 3, 7, 9}.

Якщо зняти обмеження на те, що переставні прості числа мають принаймні дві різні цифри, то всі одноцифрові прості числа, а також усі повторні прості числа будуть тривіально переставними.

Цікавим є пошук переставних простих чисел в інших системах числення.

⚠ **Означення 2.29.** Прості числа, які залишаються простими незалежно від того, скільки цифр скорочено зліва, називаються *простими числами, що скорочуються зліва* (*left-truncatable primes*).

Наприклад, число 4632647 можна скоротити зліва, оскільки воно та кожне його скорочення (632647, 32647, 2647, 647, 47 і 7) є простими числами.

Під час скорочення цифру нуль опускають, щоб уникнути тривіальних прикладів, таких як просте число $10^{60}+7$:

10000000000000000000000000000000000000000000000000000000007.

Кожне його скорочення є простим числом 7.

Ось три найбільші прості числа, які можна скоротити зліва:

95 918 918 997 653 319 693 967,

96 686 312 646 216 567 629 137,

357 686 312 646 216 567 629 137.

У десятковій системі числення є 4260 простих чисел, які можна скоротити зліва:

2, 3, 5, 7, 13, 17, 23, 37, 43, 47, 53, 67, 73, 83, 97, 113, 137, 167, 173, 197, 223,

283, 313, 317, 337, 347, 353, 367, 373, 383, 397, 443, 467, 523, 547, 613, 617,

643, 647, 653, 673, 683, 743, 773, 797, 823, 853, 883, 937, ... (A024785 в OEIS).

Цікавим є пошук простих чисел, що скорочуються зліва, в інших системах числення.

⚠ **Означення 2.30.** Прості числа, які залишаються простими незалежно від того, скільки цифр скорочено справа, називають *простими числами, що скорочуються справа* (*right-truncatable primes*).

Прикладом простого числа, що скорочується справа, є 7393, оскільки всі числа 7393, 739, 73 і 7 є простими.

У десятковій системі числення є 83 простих чисел, які можна скоротити справа:

2, 3, 5, 7, 23, 29, 31, 37, 53, 59, 71, 73, 79, 233, 239, 293, 311, 313, 317, 373, 379,

593, 599, 719, 733, 739, 797, 2333, 2339, 2393, 2399, 2939, 3119, 3137, 3733,

3739, 3793, 3797, 5939, 7193, 7331, 7333, 7393, 23333, 23339, 23399, 23993,

29399, 31193, 31379, 37337, 37339, 37397, 59393, 59399, … (A024770 в OEIS).

Найбільшим простим числом, що скорочується справа, є 73939133. Усі прості числа, більші за 5, закінчуються цифрою 1, 3, 7 або 9, тому просте число, що скорочується справа, може містити лише ці цифри після першої цифри.

Якщо дозволити 1 вважати простим числом, то найбільшими числами є 1979339333 і 1979339339.

У табл. 2.3 наведено найбільші прості числа, що скорочуються справа, для різних систем числення.

Варто відзначити, що за строгого підходу не існує простих чисел, поданих у двійковій системі числення, які можна скоротити справа, оскільки немає однорозрядних двійкових простих чисел. Однак, для цього особливого випадку, можна дозволити завершувати послідовність на числі два $(10)_2$ або три $(11)_2$.

Зі збільшенням основи системи числення з'являється більше можливостей для пошуку розширень з кожного простого числа, тому очікується, що кількість простих чисел, що скорочуються справа, необмежено зростатиме зі збільшенням основи.

Таблиця 2.3 – Найбільші прості числа, що скорочуються справа

Основа системи числення	Найбільші прості числа, що скорочуються справа	Кількість чисел
2	1011	5
3	2122	4
4	2333 (або 133313)	7
5	34222	14
6	2155555	36
7	25642 (або 166426)	19
8	2117717	68
9	3444224222	68
10	73939133	83

⚠ **Означення 2.31.** Прості числа, які залишаються простими, коли початкова («ліва») і остання («права») цифри одночасно видаляються, називають *простими числами, що скорочуються зліва і справа* (*left-and-right-truncatable primes*).

Прикладом простого числа, яке можна скоротити зліва і справа є 1825711, оскільки 1825711, 82571, 257 і 5 усі прості.

У десятковій системі числення є 920 720 315 простих чисел, які можна скоротити зліва і справа:

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, 127, 131, 137, 139, 151, 157, 173, 179, 223, 227, 229, 233, 239, 251, 257,

271, 277, 331, 337, 353, 359, 373, 379, 421, 431, 433, 439, 457, 479, 521, 523, ...

(послідовність A077390 в OEIS).

Зауважимо, що скорочення відбувається до отримання одної або двох цифр.

⚠ **Означення 2.32.** Прості числа, які залишаються простими незалежно від того, скільки цифр скорочено зліва або справа, називають *двосторонніми простими числами* (*two-sided primes*).

Існує 15 таких простих чисел:

2, 3, 5, 7, 23, 37, 53, 73, 313, 317, 373, 797, 3137, 3797, 739397 (послідовність A020994 в OEIS).

Наприклад, з простого числа 3797 утворюються такі числа:

797, 97, 7 і 379, 37, 3.

Зауважимо, що простота числа не залежить від використовуваної системи числення, тоді як прості числа, що скорочуються, визначаються лише для системи числення з конкретною основою.

Розглянуті класи простих чисел передбачають скорочення лише у строго визначеному порядку. А чи може бути довільний порядок скорочення? Тобто, чи існують прості числа, в яких можна видаляти будь-яку цифру і все одно отримувати просте число на кожному кроці? Якщо так, то кожна цифра має бути простою, і жодна цифра не може зустрічатися двічі, отже, це буде короткий список: 2, 3, 5, 7, 23, 37, 53 і 73.

Щоб зробити пошук більш цікавим, Кріс Калдвелл (*Chris Caldwell*) увів таке поняття.

⚠ **Означення 2.33.** Просте число, з якого можна видаляти по одній цифрі у певному порядку та отримувати просте число на кожному кроці, називають *простим числом з видаленням* (*deletable prime*).

Одним із прикладів є число 410256793, оскільки наведені нижче числа є простими:

$$410256793 \xrightarrow{0} 41256793 \xrightarrow{9} 4125673 \xrightarrow{2} 415673 \xrightarrow{1} 45673 \xrightarrow{3} 4567$$

$$4567 \xrightarrow{5} 467 \xrightarrow{4} 67 \xrightarrow{6} 7.$$

Кожен грав у гру, де береться певне слово, з якого потрібно утворити якомога більше інших слів. Подібну гру можна організувати з простими числами. Наприклад, комбінуванням цифр числа 1379 можна утворити такі прості числа:

3, 7, 13, 17, 19, 31, 37, 71, 73, 79, 97, 137, 139, 173, 179, 193, 197, 317, 379, 397, 719, 739, 937, 971, 1973, 3719, 3917, 7193, 9137, 9173, 9371.

Зауважте, що можна використовувати стільки кожної цифри, скільки

міститься у вихідному числі.

⚠ **Означення 2.34.** Натуральне число, з якого утворюється більше простих чисел, ніж з будь-якого меншого натурального числа, називають *первісним числом* (*primeval number*).

У табл. 2.4 наведено первісні числа, менші за 100 000 (з веб-сторінки Mike Keith).

Таблиця 2.4 – Первісні числа, менші за 100 000

Первісне число	Кількість утворюваних простих чисел	Первісне число	Кількість утворюваних простих чисел
2	1	1379	31
13	3	10079	33
37	4	10123	35
107	5	10136	41
113	7	10139	53
137	11	10237	55
1013	11	10279	60
1037	19	10367	64
1079	21	10379	89
1237	26	12379	96
1367	29	13679	106

Можливо така гра з простими числами в майбутньому буде поштовхом до нових ідей у криптографії, наприклад, як гра «Кубик Рубика», що породила ідею тривимірної криптографії.

2.9 Центровані *k*-кутні прості числа

Розглянемо послідовності $\{C_{k,n}\}$, які складаються з центрованих *k*-кутних чисел.

⚠ **Означення 2.35.** Центроване фігурне число, яке подає трикутник із точкою в центрі та всіма іншими точками, що оточують центральну точку в послідовних трикутних шарах, називають *центрованим трикутним числом* (*centered triangular number*).

Центроване трикутне число задається формулою:

$$C_{3,n} = \frac{3(n^2+n)+2}{2},\ n = 0,1,2,\ldots$$

Перші центровані трикутні числа:

1, 4, 10, 19, 31, 46, 64, 85, 109, 136, 166, 199, 235, 274, 316, 361, 409, 460,…

(послідовність A005448 в OEIS).

На рис. 2.1 наведено зображення центрованих трикутних чисел 1, 4, 10, 19 і 31.

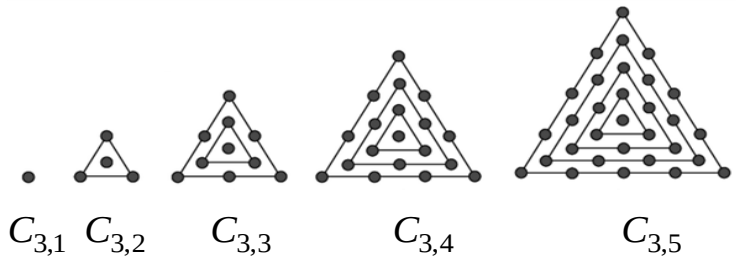


Рисунок 2.1 – Зображення центрованих трикутних чисел

Означення 2.36. Центровані трикутні числа, які є простими, називають *центрованими трикутними простими числами* (*centered triangular primes*).

Перші центровані трикутні прості числа:

19, 31, 109, 199, 409, 571, 631, 829, 1489, 1999, 2341, 2971, 3529, 4621, 4789,…

(послідовність A125602 в OEIS).

Означення 2.37. Центроване фігурне число, яке подає квадрат із точкою в центрі та всіма іншими точками, що оточують центральну точку в послідовних квадратних шарах, називають *центрованим квадратним числом* (*centered square number*).

Центроване квадратне число задається формулою:

$$C_{4,n} = n^2 + (n+1)^2,\ \boldsymbol{n = 0,1,2,\ldots}$$

Перші кілька центрованих квадратних чисел:

1, 5, 13, 25, 41, 61, 85, 113, 145, 181, 221, 265, 313, 365, 421, 481, 545, 613,…

(послідовність A001844 в OEIS).

На рис. 2.2 наведено зображення центрованих квадратних чисел 1, 5, 13, 25 і 41.

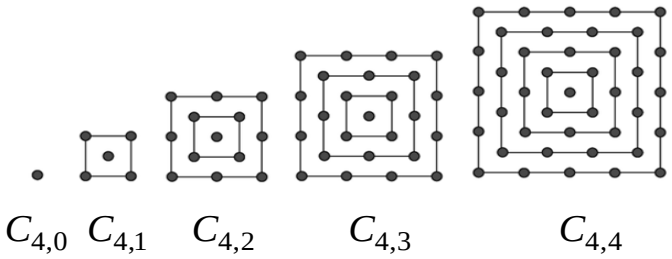


Рисунок 2.2 – Зображення центрованих квадратних чисел

Означення 2.38. Центровані квадратні числа, які є простими, називають *центрованими квадратними простими числами* (*centered square primes*).

Перші центровані квадратні прості числа:

5, 13, 41, 61, 113, 181, 313, 421, 613, 761, 1013, 1201, 1301, 1741, 1861, 2113,…

(послідовність A027862 в OEIS).

⚠ **Означення 2.39.** Центроване фігурне число, яке подає семикутник із точкою в центрі та всіма іншими точками, що оточують центральну точку в послідовних семикутних шарах, називають *центрованим семикутним числом* (*centered heptagonal number*).

Центроване семтикутне число задається формулою:

$$C_{7,n} = \frac{7(n^2+n)+2}{2},\ n = 0,1,2,\ldots$$

Перші кілька центрованих семикутних чисел:

1, 8, 22, 43, 71, 106, 148, 197, 253, 316, 386, 463, 547, 638, 736, 841, 953,…

(послідовність A069099 в OEIS).

Центровані семикутні числа змінюють парність у шаблоні непарний-парний-парний-непарний.

На рис. 2.3 наведено зображення центрованих семикутних чисел 8, 22, 43 і 71.

Рисунок 2.3 – Зображення центрованих семикутних чисел

⚠ **Означення 2.40.** Центровані семикутні числа, які є простими, називають *центрованими семикутними простими числами* (*centered heptagonal primes*).

Перші центровані семикутні прості числа:

43, 71, 197, 463, 547, 953, 1471, 1933, 2647, 2843, 3697, 4663, 5741, 8233,…

(послідовність A144974 в OEIS).

Завдяки парності центровані семикутні прості числа мають форму $C_{7,(4n)}$ або $C_{7,(4n+1)}$.

Центровані семикутні прості числа-близнюки:

43, 71, 197, 463, 1933, 5741, 8233, 9283, 11173, 14561, 34651, ...

(послідовність A144975 в OEIS).

⚠ **Означення 2.41.** Центроване фігурне число, яке подає десятикутник із точкою в центрі та всіма іншими точками, що оточують центральну

точку в послідовних десятикутних шарах, називають *центрованим десятикутним числом* (*centered decagonal number*).

Центроване десятикутне число задається формулою:

$$C_{10,n} = 5(n^2 + n) + 1,\ n = 0,1,2,\ldots$$

Перші центровані десятикутні числа:

1, 11, 31, 61, 101, 151, 211, 281, 361, 451, 551, 661, 781, 911, 1051, ...

(послідовність A062786 в OEIS).

На рис. 2.4 наведено зображення центрованих десятикутних чисел 11, 31, 61 і 101.

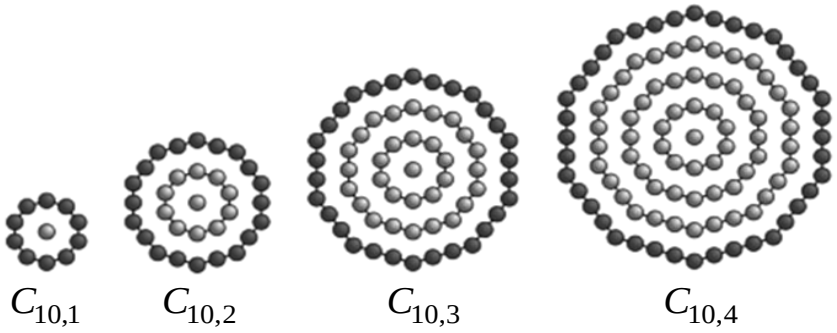


Рисунок 2.4 – Зображення центрованих десятикутних чисел

⚠ **Означення 2.42.** Центровані десятикутні числа, які є простими, називають *центрованими десятикутними простими числами* (*centered decagonal primes*).

Перші центровані десятикутні прості числа:

11, 31, 61, 101, 151, 211, 281, 661, 911, 1051, 1201, 1361, 1531, 1901, 2311,…

(послідовність A090562 в OEIS).

Центровані фігурні числа вивчають у рекреаційній математиці через їхні елегантні геометричні та арифметичні властивості, але для них ще не знайшли практичного застосування. Можливо з ними станеться те саме, що сталося з кубиком Рубіка, коли його модель було використано для побудови тривимірної криптографії.

> Рекреаційна математика або математичне дозвілля – це загальний термін для математичних занять, якими займаються з метою дозвілля, а не для науково-прикладного професійного застосування. Однак рекреаційна математика не є виключно сферою аматорів. Вона часто розглядає математичні головоломки та ігри.
>
> Прикладами є геометричні фігури та конструкції: танграм, пентаміно, ханойська вежа, кубик Рубіка; ігри з використанням теорії чисел та комбінаторики: магічний квадрат, судоку.

2.10 Взаємно прості числа

Навіщо потрібні знання про взаємно прості числа? Є достатньо причин для цього. Широко використовувана в криптографії китайська теорема про

лишки передбачає використання модулів, які є взаємно простими числами. Для побудови генераторів псевдовипадкових чисел та деяких шифрів також використовуються взаємно прості числа.

⚠ **Означення 2.43.** Два цілі числа, для яких не існує цілого числа, більшого за одиницю, що ділить їх обидва, називають *взаємно простими числами* (*relatively prime* or *coprime*).

Тобто, два цілі числа є взаємно прості, якщо їхній найбільший спільний дільник дорівнює одиниці. Тобто правило перевірки взаємної простоти цілих чисел полягає у визначенні їхнього найбільшого спільного дільника. Якщо $\text{НСД}(a,b)=1$, то числа a і b є взаємно прості, інакше – ні.

Наприклад, числа 22 і 35 є взаємно простими.

Дільники числа 22 – це **1**, 2, 11 і 22.

Дільники числа 35 – це **1**, 5, 7 і 35.

Єдиний спільний дільник 1, тому НСД(22,35)=1.

Наприклад, числа 21 і 54 не є взаємно простими.

Дільники числа 21 – це **1, 3**, 7 і 21.

Дільники числа 54 – це **1,** 2, **3**, 6, 9, 18, 27 і 54.

Є два спільні дільники 1 і 3, тому НСД(21,54) = 3.

$\text{НСД}(a,b)$ можна знайти або використовуючи канонічний розклад числа на прості множники, або за допомогою алгоритму Евкліда. Оскільки розкладання на прості множники великих чисел є досить складною процедурою, то визначення $\text{НСД}(a,b)$ на основі цього доцільне лише для невеликих чисел. Алгоритм Евкліда має придатну складність реалізації як для невеликих, так і для великих чисел.

Необхідною та достатньою умовою взаємної простоти чисел a і b є існування цілих чисел u і v, $|u|<|b|$, $|v|<|a|$, таких, що $au+bv=1$.

Зауважимо, що пара взаємно простих чисел може складатися з двох простих чисел, одного простого і одного складеного, а також з двох складених чисел.

Для визначення пари взаємно простих чисел достатньо виконати такі дії.

1. Взяти два будь-яких цілих числа m і n.
2. Обчислити $\text{НСД}(m,n)$.
3. Обчислити взаємно прості числа a і b:

$$a=\frac{m}{\text{НСД}(m,n)};\ b=\frac{n}{\text{НСД}(m,n)}.$$

Ось кілька властивостей взаємно простих чисел.

1. Число 1 є взаємно простим для кожного числа, навіть самого себе.

2. Число 0 є взаємно простим лише для числа 1. Оскільки число 0 ділиться на будь-яке ціле число, то НСД(0,1)=1.

3. Будь-які два послідовних числа є взаємно простими.

Наприклад, (5, 6); (14, 15); (27, 28); (88, 89) і так далі.

4. Будь-які два простих числа є взаємно простими. Оскільки кожне просте число має лише два дільники 1 і саме число, єдиним спільним дільником двох простих чисел буде 1.

Наприклад, 11 і 13 – два прості числа. Дільники числа 11 дорівнюють 1 і 11, а дільники числа 13 дорівнюють 1 і 13. Єдиним спільним дільником є 1, а отже, вони взаємно прості.

5. Просте число є взаємно простим з будь-яким іншим числом, якщо інше число не є кратним цього простого числа.

Наприклад, НСД(7,9)=1, НСД(7,10)=1, НСД(7,12)=1, НСД(7,13)=1 і напроти, НСД(7,21)=3.

6. Два парних числа ніколи не можуть утворити взаємно прості пари, оскільки всі парні числа мають спільний дільник 2.

7. Якщо два числа мають цифри 0 і 5 в наймолодшому розряді, то вони не є взаємно простими.

Наприклад, 20 і 45 не є взаємно простими, оскільки їх НСД дорівнює 5.

8. Сума двох взаємно простих чисел завжди взаємно проста з їх добутком.

Наприклад, 5 і 6 є взаємно простими числами. Число 11=5+6 є взаємно простим з числом $30 = 5 \cdot 6$, оскільки НСД(11,30)=1.

9. Найменше спільне кратне (НСК) двох взаємно простих чисел дорівнює їх добутку.

Виходячи з того, що $\text{НСК}(a,b) = \dfrac{|a| \cdot |b|}{\text{НСД}(a,b)}$ і $\text{НСД}(a,b) = 1$, маємо $\text{НСК}(a,b) = |a| \cdot |b|$.

Наприклад, найменше спільне кратне взаємно простих чисел 5 і 6 дорівнює $30 = 5 \cdot 6$.

⚠ **Означення 2.44.** Множину цілих чисел, для яких не існує цілого числа, більшого за одиницю, що ділить їх усі, називають *множиною спільно взаємно простих чисел* (*mutually relatively prime* or *mutually coprime*).

Інакше кажучи, числа множини $(a_1, a_2, \ldots, a_k)$ є спільно взаємно простими, якщо $\text{НСД}(a_1, a_2, \ldots, a_k) = 1$.

Для визначення $\text{НСД}(a_1, a_2, \ldots, a_k)$ використовують рекурсивну формулу:

$$\text{НСД}(a_1, a_2, \ldots, a_k) = \text{НСД}(\text{НСД}(\ldots(\text{НСД}(\text{НСД}(a_1, a_2), a_3), \ldots), a_k).$$

Наприклад, цілі числа 36, 48, 54 і 71 є спільно взаємно прості, оскільки

$$\text{НСД}(36,48) = 12;\ \text{НСД}(12,54) = 6;\ \text{НСД}(6,71) = 1.$$

Наприклад, цілі числа 36, 48, 54 і 81 не є спільно взаємно прості, оскільки

$$\text{НСД}(36,48) = 12;\ \text{НСД}(12,54) = 6;\ \text{НСД}(6,81) = 3.$$

⚠ **Означення 2.45.** Множину цілих чисел, у який для кожної пари різних чисел не існує цілого числа, більшого за одиницю, що ділить їх, називають *множиною попарно взаємно простих чисел* (*pairwise relatively prime* or *pairwise coprime*).

Інакше кажучи, числа множини $(a_1, a_2, \ldots, a_k)$ є попарно взаємно простими, якщо $\prod_{i=1}^{k}\prod_{j=1}^{i-1} \text{НСД}(a_i, a_j) = 1$.

Властивість попарно взаємно простих чисел є сильнішою за властивість спільно взаємно простих – попарно взаємно прості числа також будуть спільно взаємно простими, але протилежне – хибно.

Наприклад, цілі числа 48, 54 і 71 є спільно взаємно простими, оскільки $\text{НСД}(48,54) = 6;\ \text{НСД}(6,71) = 1$. Однак вони не попарно взаємно прості, тому що $\text{НСД}(48,54) = 6$.

Наприклад, цілі числа 49, 54 і 71 є попарно взаємно простими, оскільки $\text{НСД}(49,54) = 1;\ \text{НСД}(49,71) = 1$ і $\text{НСД}(54,71) = 1$. Крім того, вони є спільно взаємно простими, тому що $\text{НСД}(49,54) = 1$ і $\text{НСД}(1,71) = 1$.

Очевидно, що лише для двох цілих чисел поняття «взаємно прості» і «попарно взаємно прості» збігаються.

Концепція попарної взаємної простоти є важливою як гіпотеза в багатьох результатах теорії чисел зокрема для китайської теореми про залишки.

Множина попарно взаємно простих чисел може бути нескінченною. Прикладом є множина всіх чисел Ферма.

Для обчислення НСД і генерування взаємно простих чисел пропонуються сервіси:

«Online calculators» (https://planetcalc.com/9543/) і

«Co-Prime Calculator» **(**https://www.mymathtables.com/numbers/co-prime-numbers-calculator.html**).**

? КОНТРОЛЬНІ ПИТАННЯ

1. Які детерміновані алгоритми використовують для знаходження початкового набору простих чисел?
2. Чим відрізняється решето Ератосфена від решета Ейлера?
3. Назвіть модифікації решета Ератосфена.
4. В чому суть алгоритму решета Сундарама?
5. На чому базується алгоритм решета Аткіна?
6. Яка загальна формула для обчислення простих чисел?
7. Які числа називаються простими числами Піфагора?
8. Який поліном для обчислення простих чисел використовував Ейлер?
9. Що зображають кубові прості числа?
10. В чому суть формули Вільсона для обчислення простих чисел?

11. Які ще формули для обчислень простих чисел вам відомі?

12. Які послідовності називають послідовностями Люка?

13. Які послідовності Люка відомі для деяких значень?

14. Яка відмінність між рекурентними формулами чисел Люка та чисел Фібоначчі?

15. За якою рекурентною формулою обчислюються числа Пелла?

16. Які числа називають спеціально сконструйованими?

17. В чому відмінність між числами Евкліда та числами Куммера?

18. Як в криптографії застосовуються безпечні прості числа?

19. Дайте означення сильного простого числа.

20. В чому суть гіпотези Голдбаха і чим вона відрізняється від теореми Ченя?

21. Для чого використовують прості числа Пірпонта?

22. Яка відмінність між адитивними простими числами та мультиплікативними?

23. Яку форму зображення мають центровані десятикутні числа?

24. Назвіть властивості взаємно простих чисел.

25. В чому суть концепції попарної взаємної простоти цілих чисел?

ВПРАВИ

1. За алгоритмом решета Ератосфена виписати набір простих чисел, що не перевищує n=19.

2. За алгоритмом, що просіює квадрати простих чисел, сформуйте набір простих чисел, що не перевищують n=29.

3. Користуючись алгоритмом решета для непарних чисел, встановіть набір простих чисел, що не перевищують n=37.

4. Використовуючи решето Ейлера, випишіть набір простих чисел, що не перевищують n=41.

5. Обчисліть набір перших 10 простих чисел Піфагора.

6. Випишіть прості числа першої сотні за поліномом Ейлера.

7. Обчисліть числа Люка першої тисячі.

8. Встановіть послідовність чисел Фібоначчі першої сотні.

9. Скільки чисел Пелла є в послідовності першої сотні?

10. Скільки чисел Куммера є в першій тисячі?

11. Випишіть послідовність простих чисел Софі Жермен першої сотні.

12. Обчисліть прості числа Ченя в межах першої сотні.

13. Скільки простих чисел Вагстафа є в першій сотні?

14. Обчисліть послідовність простих чисел Мерсенна в межах першої тисячі.

15. Випишіть перших 4 простих чисел Ферма.

16. З послідовності простих чисел 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 випишіть суперпрості числа.

17. Випишіть перших 20 простих щасливих чисел.

18. З десяткової системи числення випишіть двоцифрові переставні прості числа.

19. Скільки існує двосторонніх простих чисел?

20. Випишіть прості центровані трикутні числа першої тисячі.

21. Скільки простих центрованих квадратних чисел є в першій сотні?

22. Скільки простих центрованих семикутних чисел є в першій тисячі?

23. Довести, що числа 9 і 25 є взаємно простими.

24. Перевірити чи є цілі числа 12, 36, 54 та 71 спільно взаємно простими?

25. Довести, що цілі числа 36, 54 і 71 не є попарно взаємно простими.

ДЖЕРЕЛА ДЛЯ ПОГЛИБЛЕНОГО ВИВЧЕННЯ

1. Agoh, Takashi, «On Sophie Germain primes», *Tatra Mt. Math. Publ.*, 20 (2000), pp. 65-73.

2. I. O. Angell and H. J. Godwin, «On truncatable primes», *Math. Comp.*, 31 (1977), pp. 265-267

3. Bicknell, Marjorie. «A primer on the Pell sequence and related sequences». *Fibonacci Quarterly.* 13 (4): (1975), pp. 345-349.

4. Binbin Zhou, «The Chen primes contain arbitrarily long arithmetic progressions», *Acta Arithmetica* 138:4 (2009), pp. 301-315.

5. J. Brillhart, P. Montgomery and R. Silverman, «Tables of Fibonacci and Lucas factorizations», *Math. Comp.*, 50 (1988), pp. 251-260.

6. C. Caldwell and Y. Gallot, «On the primality of $n! \pm 1$ and $2 \times 3 \times 5 \times \cdots \times p \pm 1$», *Math. Comp.*, 71:237 (2002), pp. 441-448.

7. C. Caldwell, «Permutable primes», *J. Recreational Math.*, 19:2 (1987), pp. 135-138.

8. C. Caldwell, «Truncatable primes», *J. Recreational Math.*, 19:1 (1987), pp. 30-33.

9. C. Caldwell and H. Dubner, «Primorial, factorial and multifactorial primes», *Math. Spectrum*, 26:1 (1993/4), pp. 1-7.

10. Cheng, Yuan-You Fu-Rui, «Explicit estimate on primes between consecutive cubes», *Rocky Mountain J. Math.*, 40:1 (2010), pp. 117-153.

11. R. Crandall, K. Dilcher and C. Pomerance, «A search for Wieferich and Wilson primes», *Math. Comp.*, 66:217 (1997), pp. 433-449.

12. deletable prime https://t5k.org/glossary/page.php?sort=DeletablePrime

13. H. Dubner, «Factorial and primorial primes», *J. Recreational Math.*, 19:3 (1987), pp. 197-203.

14. H. Dubner and W. Keller, «New Fibonacci and Lucas primes», *Math. Comp.*, 68:225 (1999), pp. 417-427.

15. H. Dubner and R. Ondrejka, «A primer on palindromes», *J. Recreational*

Math., 26:4 (1994) pp. 256-267.

16. Fibonacci number https://t5k.org/glossary/page.php?sort=FibonacciNumber

17. Fibonacci prime https://t5k.org/glossary/page.php?sort=FibonacciPrime

18. E. Fouvry and H. Iwaniec, «Primes in arithmetic progressions», *Acta Arith*. 42 (1983), no. 2, pp. 197-218.

19. J. Friedlander and H. Iwaniec. «The polynomial $X^2 + Y^4$ captures its primes», *Ann. of Math.*, 148: **(**1998), pp. 945-1040.

20. Granville, Andrew; Martin, Greg, «Prime number races», *The American Mathematical Monthly,* 113 (1): (January 2006), pp. 1-33.

21. K. Indlekofer and A. Járai, «Largest known twins and Sophie Germain primes», *Math. Comp.,* 68:227 (1999), pp. 1317-1324.

22. Jones, James P.; Sato, Daihachiro; Wada, Hideo; Wiens, Douglas, «Diophantine representation of the set of prime numbers», *American Mathematical Monthly, Mathematical Association of America*, 83 (6): (1976), pp. 449-464.

23. M. Joye and J-J. Quisquater, «Efficient Computation of Full Lucas Sequences», *Electronics Letter*s, Vol. 32 (1996), pp. 537-538. Corrected version available at http://www.dice.ucl.ac.be/crypto/publications.html.

24. M. Křížek and L. Somer, «Euclidean primes have the minimum number of primitive roots», *JP J. Algebra Number Theory Appl.,* 12:1 (2008), pp. 121-127.

25. left-truncatable prime https://t5k.org/glossary/page.php?sort=LeftTruncatablePrime

26. Lucas number https://t5k.org/glossary/page.php?sort=LucasNumber

27. Lucas Sequences in Cryptography http://www.weidai.com/lucas.html

28. Y. Matijasevic. «Diophantine representations of the set of prime numbers», *Dokl. Akad. Nauk SSSR,* 12: (1971), pp. 354-358.

29. Y. Matiyasevich, «Formulas for prime numbers», In *Tabachnikov, Serge (ed.). Kvant Selecta: Algebra and Analysis. Vol. II. American Mathematical Society.* (1999), pp. 13-24.

30. Mersenne Primes: History, Theorems and Lists https://t5k.org/mersenne/

31. N. Mackinnon, «Prime number formulae», *The Mathematical Gazette*. 71 (456): (June 1987), pp. 113-114.

32. W. H. Mills, «A prime-representing function», *Bull. Amer. Math. Soc.*, 53 (1947), p. 604.

33. permutable prime https://t5k.org/glossary/page.php?sort=PermutablePrime

34. J. Pierpont, «On an undemonstrated theorem of the Disquisitiones Aritmeticae», *American Mathematical Society Bulletin*,:2 (1895-1896), pp. 77 - 83.

35. Prime Number Types https://prime-numbers.info/#numberTypes

36. primeval number https://t5k.org/glossary/page.php?sort=Primeval

37. P. Ribenboim, «The new book of prime number records», 3rd edition, Springer-Verlag, pp. xxiv+541, New York, NY, 1995**.**

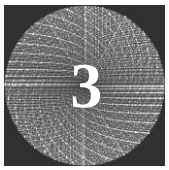
38. J. Rosser and L. Schoenfeld. «Approximate formulas for some functions of prime numbers», *Illinois J. Math.*, 6: **(**1962), pp. 64-94.

39. J. Selfridge and A. Hurwitz. «Fermat numbers and Mersenne numbers», *Math. Camp.*, 18: (1964), pp. 146-148.

40. Sieve of Sundaram
https://artofproblemsolving.com/wiki/index.php/Sieve_of_Sundaram

41. Solinas, Jerome A., «Generalized Mersenne Prime». In Tilborg, Henk C. A. van; Jajodia, Sushil (eds.) (2011). Encyclopedia of Cryptography and Security. Springer US. pp. 509-510.

42. E. M. Wright, «A class of representing functions», *J. London Math. Soc.*, 29 (1954), pp. 63-71.



ФАКТОРИЗАЦІЯ ЦІЛИХ ЧИСЕЛ

3.1 Поняття факторизації

Фундаментальна теорема арифметики стверджує:

Теорема 3.1. *Кожне додатне ціле число* $n > 1$ *може бути подано єдиним способом, окрім перестановки, як добуток простих чисел:*

$$n = p_1 \cdot p_2 \cdot \ldots \cdot p_k .$$

Як наслідок, кожне натуральне число n єдиним чином подається у вигляді:

$$n = p_1^{a_1} p_2^{a_2} \ldots p_m^{a_m} ,$$

де $a_1, a_2, \ldots, a_m$ – натуральні числа.

Таке подання числа n називають *канонічним розкладом* на прості множники.

Ця теорема дає початок тому, що можна назвати «Основна задача арифметики»: *розкласти задане ціле число* $n > 1$ *на прості множники*.

> Проблема відмінності простих чисел від складених і розкладання останніх на прості множники, як відомо, є однією з найважливіших і корисніших в арифметиці. Вона залучила працьовитість і мудрість стародавніх та сучасних геометрів до такої міри, що було б зайвим довго обговорювати цю проблему... Крім того, гідність самої науки, здається, потребує, щоб усі можливі засоби були досліджені для вирішення такої елегантної і такої знаменитої проблеми.
>
> Карл Фрідріх Гаусс
> «Disquisitiones Arithmeticae» (1801)

⚠ **Означення 3.1.** Частину величини (як правило, ціле число або поліном), яка за множення на інші множники дає всю величину, називають *фактором* (*factor*) або *множником*.

⚠ **Означення 3.2.** Визначення факторів величини називають *факторизацією* (*factorization* або *factorisation*).

Зазвичай потрібно розбити величину на найменш можливі частини, щоб жоден фактор сам по собі не був факторним.

⚠ **Означення 3.3.** Розкладання цілого числа на прості множники називають *факторизацією цілого числа* (*integer factorization*).

Виходячи з цього означення теорему 3.1 також називають теоремою єдиної факторизації.

Існує багато алгоритмів цілочисельної факторізації, але для всіх них завдання формулюється таким чином.

Дано складене ціле число n, знайти всі прості множники p_i для n, тобто знайти p_i такі, що $p_i \mid n$.

Проблема розкладання на множники є набагато складнішою, ніж проблема визначення того, чи є ціле число складеним або простим. Отже, перш ніж розкладати ціле число на множники, його потрібно перевірити чи справді воно є складеним.

⚠ **Означення 3.4.** Факторизацію виду $n = a \cdot b$, де $1 < a < n$ і $1 < a < n$, називають *нетривіальною факторізацією* (*non-trivial factorization*), а числа a і b називають *нетривіальними множниками* (*non-trivial factors*) числа n.

Числа a і b не обов'язково прості, тому знайдені множники a і b потрібно перевірити на простоту. Далі, алгоритм факторизації застосовується до a та/або b, якщо будь-яке з них виявиться складеним. Така рекурсивна процедура забезпечує отримання всіх простих множників числа n.

Для великих цілих чисел визначення всіх факторів зазвичай є дуже складним, за винятком певних видів чисел.

Передбачувана складність розв’язання задачі факторизації лежить в основі криптостійкості алгоритмів шифрування та протоколів з відкритим ключем, зокрема алгоритму RSA.

Відомі алгоритми факторизації поділяють на два основних типи: спеціального призначення та загального призначення.

Час роботи алгоритмів факторизації спеціального призначення залежить від властивостей числа, яке розкладається на множники, або від одного з його невідомих множників. Ці алгоритми швидко знаходять малі прості множники незалежно від величини n. Через це такі алгоритми зазвичай застосовують для видалення малих множників перед реалізацією алгоритму загального призначення. Основна проблема з алгоритмами спеціального призначення полягає в тому, що якщо n не має малого множника, то вони є неефективними.

До алгоритмів цього типу належать:

- пробне ділення;
- метод колесної факторизації;
- метод факторизації Ферма;
- метод факторизації Ейлера;
- алгоритм ρ Полларда;
- алгоритм $p - 1$ Полларда;
- алгоритм $p + 1$ Вільямса;
- спеціальне решето числового поля.

Алгоритми факторізації загального призначення знаходять прості множники незалежно від їх величини, але вони мають експоненціальний або субекспоненціальний час реалізації.

Алгоритми цього типу реалізують такі методи:

- метод раціонального решета;
- метод квадратичного решета;
- метод загального решета числового поля та ін.

3.2 Класи натуральних чисел

Виходячи з розкладання натуральних чисел на прості множники, розглядають певні класи чисел.

⚠ **Означення 3.5.** Натуральне число *n*, що утворюється як добуток *k* простих чисел, називають *k-майже простим числом* (*k-almost prime*).

Більш формально, число *n* є *k*-майже простим тоді і тільки тоді, коли $\tau(n) = k$, де $\tau(n)$ – кількість простих чисел у розкладенні числа *n* на прості множники.

Множину *k*-майже простих чисел зазвичай позначають P_k. У табл. 3.1 наведено приклади послідовностей *k*-майже простих чисел.

Таблиця 3.1 – Послідовності *k*-майже простих чисел

k	Елементи послідовності	Послідовність в OEIS
1	2, 3, 5, 7, 11, 13, 17, 19, 23,…	A000040
2	4, 6, 9, 10, 14, 15, 21, 22, …	A001358
3	8, 12, 18, 20, 27, 28, 30,…	A014612
4	16, 24, 36, 40, 54, 56, 60,…	A014613
5	32, 48, 72, 80, 108, 112,…	A014614
6	64, 96, 144, 160, 216, 224,…	A046306
7	128, 192, 288, 320, 432, 448,…	A046308

Таким чином, натуральне число є простим тоді і тільки тоді, коли воно 1-майже просте.

⚠ **Означення 3.6.** Складене число, яке є добутком двох (можливо однакових) простих чисел, називають *напівпростим числом* (*semiprime*) або *pq-числом*.

Такі числа утворюють клас 2-майже простих чисел. Наведена в табл. 3.1 послідовність складається з чисел, серед яких є квадрати (добуток однакових простих чисел).

Напівпрості числа, множники яких є різними (тобто безквадратні напівпрості числа): 6, 10, 14, 15, 21, 22, 26, 33, 34, ... (послідовність A006881 в OEIS).

⚠ **Означення 3.7.** Натуральне число, яке є добутком трьох різних простих чисел *p, q* і *r*, називають *сфенічним числом* (*sphenic number*).

Найменшими сфенічними числами є:

30, 42, 66, 70, 78, 102, 105, 110, 114, 130, 138, 154, 165, ... (послідовність A007304 в OEIS).

Зокрема: $30 = 2 \cdot 3 \cdot 5$, $42 = 2 \cdot 3 \cdot 7$, $66 = 2 \cdot 3 \cdot 11$, $70 = 2 \cdot 5 \cdot 7$ і $78 = 2 \cdot 3 \cdot 13$.

Напівпрості і сфенічні числа використовуються у криптосистемах з відкритим ключем, таких як RSA. Причиною цього є те, що обчислювальна складність факторизації числа, утвореного добутком великих простих чисел є значно більшою, ніж обчислювальна складність множення цих чисел.

Алгоритми шифрування RSA передбачають використання спеціальних великих простих чисел як множників. У табл. 3.2 наведено деякі спеціальні напівпрості числа, які є добутком двох великих (різних) простих чисел. Тут: L_n, L_p і L_q – кількість десяткових розрядів чисел *n, p* і *q*, відповідно.

⚠ **Означення 3.8.** Натуральне число називають *B-гладким числом* (*B-smooth number*), якщо всі його прості множники не більші за число *B*.

Варто відзначити, що саме число *B* не обов'язково може бути серед множників *B*-гладкого числа. Якщо найбільший простий множник числа дорівнює *p*, то число є *B*-гладким для будь-якого $B \geq p$. У багатьох алгоритмах *B* є простим числом, але також використовують і складені числа.

Таблиця 3.2 – Спеціальні великі напівпрості числа

$n = pq$	L_n	L_p	L_q
$10^{48}+19$	49	21	28
$10^{50}+27$	51	22	29
$10^{54}-3$	54	23	32
$10^{63}+19$	64	32	32
RSA-129	129	64	65
RSA-140	140	70	70

Натуральне число є *B*-гладким тоді і лише тоді, коли воно є *p*-гладким.

Наприклад, число 120 розкладається на такі множники: $120 = 2^3 \cdot 3 \cdot 5$. Отже, це число є 5-гладким.

Приклади послідовностей *B*-гладких чисел наведено в табл. 3.3. Як бачимо, 2-гладкі числа є степенями 2.

3-гладкі числа називають *гармонічними числами* (*harmonic numbers*), хоча ця назва має й інші, більш широко використовувані значення.

5-гладкі числа характеризуються тим, що мають прості множники лише 2, 3 або 5. Тобто вони обчислюються за формулою:

$$regular = 2^i \cdot 3^j \cdot 5^k, \text{ де } i, j, k \geq 0.$$

Через це їх також називають *регулярними числами* (*regular numbers*) або числами Хеммінга на честь Річарда Хеммінга, який запропонував проблему пошуку комп'ютерних алгоритмів для генерування цих чисел у порядку зростання. Ця задача була використана як тестовий приклад для функціонального програмування.

Таблиця 3.3 – *B*-гладкі числа

B	*B*-гладкі числа	Номер послідовності в OEIS
2	1, 2, 4, 8, 16, 32, 64, 128, 256, 512, ...	A000079
3	1, 2, 3, 4, 6, 8, 9, 12, 16, 18, 24, ...	A003586
5	1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, ...	A051037
7	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 15, ...	A002473

7-гладкі числа мають прості множники лише 2, 3, 5 або 7 і обчислюються за формулою:

$$2^i \cdot 3^j \cdot 5^k \cdot 7^m, \text{ де } i, j, k, m \geq 0.$$

Їх називають *скромними числами* (*humble numbers*) та іноді високоскладеними (*highly composite*), хоча це суперечить іншому означенню високоскладених чисел.

⚠ **Означення 3.9.** Натуральне число *n* називають *B-степенево гладким числом* (*B-powersmooth number*), якщо всі степені простих чисел p^v, що ділять *n*, задовольняють умову $p^v \leq B$.

Наприклад, $120 = 2^3 \cdot 3 \cdot 5$ є 5-гладким числом, але не 5-степенево гладким, оскільки є 2^3, яке більше за 5. Це 8-степенево гладке число, оскільки його найбільший простий множник дорівнює $2^3 = 8$.

B-гладкі та *B*-ступенево гладкі числа мають ряд застосувань у криптографії. Наприклад, в алгоритмі Поліга–Хеллмана для обчислення дискретних логарифмів для груп *B*-гладкого порядку, а також в алгоритмах цілочисельної факторизації, зокрема в алгоритмі загального решета числового поля (*general number field sieve*, GNFS) і в $p - 1$ алгоритмі Полларда. Про такі алгоритми часто кажуть, що вони працюють із гладкими числами без вказівки *B*; це означає, що використовувані числа мають бути *B*-ступенево гладкими для деякого невизначеного невеликого числа *B*. Зі збільшенням *B* продуктивність відповідного алгоритму або методу швидко зменшується. Ще одним прикладом конструктивного використання гладкості є геш-функція VSH (Very Smooth Hash) з доведеною криптографічною стійкістю.

⚠ **Означення 3.10.** Число, яке не ділиться на жоден квадрат, крім 1, називають *вільним від квадратів* (*squarefree number*) або *безквадратним*.

Наприклад, число 21 – вільне від квадратів, а число 27 – ні, оскільки 27 ділиться на $9 = 3^2$.

Початок послідовності вільних від квадратів чисел такий:

1, 2, 3, 5, 6, 7, 10, 11, 13, 14, 15, 17, 19, 21, 22, 23, 26, 29, 30, 31, 33, 34, 35,…

(послідовність A005117 в OEIS).

Додатне число n вільне від квадратів тоді і тільки тоді, коли в розкладанні цього числа на прості множники жодне просте число не зустрічається більше, ніж один раз. Або додатне число n вільне від квадратів тоді і тільки тоді, коли значення функції Мебіуса $\mu(n) \neq 0$.

Якщо подати вільне від квадратів число як нескінченний добуток виду:

$$\prod_{n=0}^{\infty} p_{n+1}^{a_n},$$

де $a_n \in \{0,1\}$, а p_n – n-е просте число, то можна використати a_n як біти двійкового числа:

$$\sum_{n=0}^{\infty} a_n 2^n.$$

Наприклад, вільне від квадратів число 70 має розкладання $2 \cdot 5 \cdot 7$, або як нескінченний добуток:

$$2^1 \cdot 3^0 \cdot 5^1 \cdot 7^1 \cdot 11^0 \cdot 13^0 \cdot \ldots$$

У двійковому коді біти a_n записуються у зворотному порядку і тому число 70 кодується послідовністю ...001101 або 13 в десятковій системі числення.

Оскільки розкладання на прості множники кожного числа є єдиним, то єдиним є й двійковий код кожного вільного від квадратів числа.

Зворотне також істинне: оскільки у кожного додатного числа є єдиний двійковий код, то його можна декодувати, отримуючи число, вільне від квадратів.

Наприклад, десятковому числу 105 відповідає двійковий код 1101001. Використовуючи біти цього коду, маємо $2^1 \cdot 3^0 \cdot 5^0 \cdot 7^1 \cdot 11^0 \cdot 13^1 \cdot 17^1 = 3094$.

Безквадратним числам послідовності A005117 в OEIS відповідає послідовність A048672:

0, 1, 2, 4, 3, 8, 5, 16, 32, 9, 6, 64, 128, 10, 17, 256, 33, 512, 7, 1024, 18, 65, 12,…

Зауважимо, що потужність множини чисел, вільних від квадратів, збігається з потужністю множини всіх натуральних чисел. Це означає, що кодування вільних від квадратів чисел по порядку є перестановкою множини натуральних чисел.

3.3 Пробне ділення

Пробне ділення (*trial division*) є найлегшим для розуміння з алгоритмів розкладання числа *N* на множники. Такий алгоритм вперше описав Фібоначчі у книзі «Liber Abaci» (1202).

Основна ідея пробного ділення полягає в перевірці, чи ділить ціле число *N* кожен з можливих дільників *d*.

Відомо, що будь-яке складене число є добутком двох або більше простих чисел. Нехай множники числа *N* дорівнюють n_1, n_2 і так далі. Множники є найбільшими лише тоді, коли для числа *N* існують два множники n_1 і n_2. Числа n_1 і n_2 є найбільшими лише тоді, коли вони однакові за значенням. Нехай $n_1 = n_2 = n$. Тоді $N = n \cdot n = n^2$. Отже, найбільший можливий множник для *N* – це квадратний корінь з *N*. Таким чином, алгоритм пробного ділення передбачає використання як можливих дільників простих чисел з діапазону від 2 до $\sqrt{N}$. Якщо $\sqrt{N}$ є цілим числом, тоді він є множником, а N є повним квадратом.

Приклад 3.1. Розкласти число 630 на прості множники.

Розв'язання. Як можливі дільники використовуються прості числа з діапазону від 2 до $\lfloor\sqrt{630}\rfloor$=25, тобто 2, 3, 5, 7, 11, 13, 17, 19, 23.

630:2=315;
2 не ділить 315;
315:3=105;
105:3=35;
3 не ділить 35;
35:5=7;
5 не ділить 7;
7:7=1. Зупинка процесу пробного ділення.

Відповідь: $630 = 2 \cdot 3^2 \cdot 5 \cdot 7$

Цей метод добре працює для розкладання невеликих цілих чисел, але він є неефективним у разі великих цілих чисел. Наприклад, Ферма не зміг виявити, що 6-е число Ферма не є простим числом. Навіть за допомогою сучасних потужних комп'ютерів, число з 500 десяткових цифр, яке є добутком двох навмання вибраних простих чисел, практично неможливо розкласти на прості множники.

3.4 Метод колісної факторизації

Метод колісної факторизації (WFM – wheel factoring method) передбачає використання двох списків. Перший список $B = \{b_1, b_2, ..., b_r\}$, де

b_i – просте число, а r – число 3 або 4, називається *базис* (*basis*) і складається з перших r послідовних простих чисел. Другий список T, що називається *колесо* (*wheel*), складається з чисел, які є взаємопростими з усіма числами базису B. Коло колеса дорівнює добутку базисних чисел $s = b_1 \times b_2 \times \ldots \times b_r$. Числа базису і колеса використовуються для пошуку найменшого дільника числа, яке потрібно розкласти на множники.

На рис. 3.1 показано колесо факторизації, коли $B = \{2,3,5\}$ і коло оберту дорівнює $s = 30 = 2 \times 3 \times 5$. Перший виток $T_1 = \{1, 7, 11, 13, 17, 19, 23, 29\}$, а другий виток $T_2 = \{31, 37, 41, 43, 47, 49, 53, 59\}$. Усі елементи T є взаємопростими з елементами B, тобто $НСД(b_i, t_j) = 1$, $b_i \in B$, $t_j \in T$.

Не обов'язково починати WFM з 1. Можна почати, наприклад, з 11. У цьому випадку спочатку перевіряється подільність числа n на числа базису $B = \{2,3,5,7\}$ і числа першого витка $\{11,13,17,19,23,29,31,37\}$. На наступному кроці перевіряється подільність числа n на числа наступного витка $\{41,43,47,49,53,59,61,67\}$. Такі кроки виконуються поки не буде визначено множник числа n.

Перше число j-го витка дорівнює $i = 11 + 30(j-1)$, а наступними є елементи $i+2$, $i+6$, $i+8$, $i+12$, $i+18$, $i+20$ та $i+26$. Різниці між двома послідовними числами в будь-якому витку дорівнюють 2, 4, 2, 4, 6, 2 і 6, відповідно, тоді як різниця між останнім числом у будь-якому витку та першим числом у наступному витку дорівнює 4. Для обчислення всіх витків ці різниці зберігаються в масиві *inc*[8]={2, 4, 2, 4, 6, 2, 6, 4}.

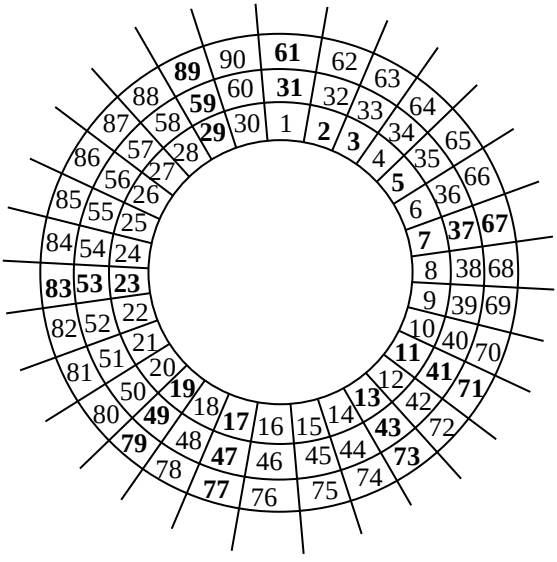


Рисунок 3.1 – Колесо факторизації

Алгоритм 3.1. Пробне ділення на елементи базису

***Вхід*:** складене ціле число n.
***Вихід*:** ціле число p.

***Процес*:**

1. Покласти $p = 1$.

2. Якщо $n \bmod 2 = 0$, то покласти $p = 2$ і повернути p.
3. Якщо $n \bmod 3 = 0$, то покласти $p = 3$ і повернути p.
4. Якщо $n \bmod 5 = 0$, то покласти $p = 5$ і повернути p.
5. Якщо $n \bmod 7 = 0$, то покласти $p = 7$ і повернути p.
6. Повернути p.

Алгоритм 3.2. Колісна факторизація

***Вхід*:** складене ціле число n.
***Вихід*:** множники p і q числа n.

***Процес*:**

1. Визначити p за алгоритмом 3.1.
2. Якщо $p \neq 1$, то покласти $q = n / p$ та повернути p і q.
3. Обчислити $m = \lfloor \sqrt{n} \rfloor$.
4. Покласти $k = 11,\ i = 1$.
5. Створити $inc[8] = \{2,4,2,4,6,2,6,4\}$.
6. Поки $k \le m$, виконувати:
   6.1. Якщо $n \bmod k = 0$, то покласти $p = k$ і $q = n / p$ та повернути p і q, інакше:
      6.1.1 Обчислити $k = k + inc[i]$.
      6.1.2 Якщо $i < 8$, то покласти $i = i + 1$, інакше покласти $i = 1$.
7. Завершити «Поки».

У разі використання цього методу для пошуку простих чисел або для просіювання зменшується кількість чисел-кандидатів, які можна вважати можливими простими числами. З основою {2, 3} скорочення до 1/3 < 34 % усіх чисел. Це означає, що повністю 2/3 усіх чисел-кандидатів пропускаються автоматично. Більші основи ще більше зменшують цю частку. Наприклад, з основою {2, 3, 5} скорочення до 8/30 < 27 %, а з основою {2, 3, 5, 7} – до 48/210 < 23%. Чим більше колесо, тим більше задіяних обчислювальних ресурсів і менші додаткові покращення.

3.5 Метод факторизації Ферма

Метод факторизації Ферма базується на поданні непарного цілого числа як різниці двох квадратів:

$$N = a^2 - b^2.$$

Ця різниця алгебраїчно розкладається як $(a+b)(a-b)$. Тому кожне непарне число має подання $N = c \cdot d$, де $c = (a+b)$ і $d = (a-b)$.

Метод передбачає випробування різних значень числа a з метою отримання результату: $a^2 - N = b^2$.

Як перше значення a для випробування потрібно використати $a_1 = \lceil \sqrt{N} \rceil$. Потім перевірити чи є число $\Delta a_1 = a_1^2 - N$ повним квадратом.

Для цього можна скористатися такими властивостями квадратів чисел у десятковому записі.

1. Остання цифра квадрата може дорівнювати 0, 1, 4, 5, 6 або 9 (квадратичні залишки за модулем 10).
2. Дві останні цифри квадрата можуть набувати значень:
   00, 01, 04, 09, 16, 21, 24, 25, 29, 36, 41, 44, 49, 56, 61, 64, 69, 76, 81, 84, 89 або 96 (квадратичні залишки за модулем 100).
3. Квадрат не може закінчуватися непарною кількістю нулів.
4. Квадрат або ділиться на 4, або у разі ділення на 8 дає залишок 1.
5. Квадрат або ділиться на 9 або у разі ділення на 3 дає залишок 1.

Властивість 2 вказує на те, що існує лише 22 комбінації останніх двох цифр, які може мати квадрат числа, тому інші комбінації можна виключити.

Якщо Δa_1 не є квадратом, то спробувати число $a_2 = a_1 + 1$ і перевірити $\Delta a_2 = a_2^2 - N$. Варто врахувати таке:

$$\Delta a_2 = a_2^2 - N = (a_1 + 1)^2 - N = a_1^2 + 2a_1 + 1 - N = \Delta a_1 + 2a_1 + 1.$$

Тобто для обчислення Δa_2 не потрібно виконувати піднесення до квадрата.

Продовжити з $\Delta a_3 = a_3^2 - N$.

$$\Delta a_3 = a_3^2 - N = \Delta a_2 + 2a_2 + 1 = \Delta a_2 + 2a_1 + 3.$$

З цього випливає, що наступні різниці отримують простим додаванням 2.

Якщо результатом реалізації методу є $N = 1 \cdot N$, то це показує, що N є простим.

Нехай c буде найбільшим множником для $N = c \cdot d$. Виходячи з того, що $a = (c + d)/2$, для реалізації методу потрібно виконати приблизно $(c + d)/2 - \sqrt{N} = (\sqrt{d} - \sqrt{c})^2/2 = (\sqrt{N} - c)^2/2c$ кроків.

Якщо N просте (коли $c = 1$), то потрібно виконати $O(N)$ кроків. Це неефективний спосіб доведення простоти числа. Але якщо N має множник, близький до квадратного кореня, то метод реалізується швидко. Точніше, якщо c відрізняється менше ніж на $(4N)^{1/4}$ від $\sqrt{N}$, то для реалізації методу потрібен лише один крок незалежно від величини N.

Приклад 3.2. Розкласти на множники число 5959.

Розв'язання.

Для першого випробування використовуємо $a_1 = \lceil \sqrt{N} \rceil = \lceil \sqrt{5959} \rceil = 78.$

Обчислюємо $\Delta a_1 = a_1^2 - N = 78^2 - 5959 = 6084 - 5959 = 125.$

Перевіряємо, чи число 125 є повним квадратом. За властивістю 2, це може бути квадрат. Умови властивостей 4 і 5 не виконуються. Тому число 125 не є квадратом.

Для другої спроби використовуємо $a_2 = a_1 + 1 = 78 + 1 = 79.$

Обчислюємо $\Delta a_2 = \Delta a_1 + 2a_1 + 1 = 125 + 2 \cdot 78 + 1 = 282.$

Число 282 не є повним квадратом за властивістю 5.

Для третьої спроби використовуємо $a_3 = a_2 + 1 = 79 + 1 = 80.$

Обчислюємо $\Delta a_3 = \Delta a_2 + 2a_1 + 3 = 282 + 2 \cdot 78 + 3 = 441.$

Число 441 є повним квадратом (властивості 2 і 4). Отже, $a = 80$ і $b = \sqrt{441} = 21.$

Таким чином, множниками числа 5959 є:

$c = a + b = 80 + 21 = 101,\ d = a - b = 80 - 21 = 59.$

Відповідь: $c = 101,\ d = 59.$

Якщо *N* має більше двох простих множників, то спочатку отримується факторізація з найменшими значеннями *a* і *b*. Тобто найменший множник $(a+b) \ge \sqrt{N}$, тому $(a-b) = N/(a+b)$ є найбільшим множником $\le \sqrt{N}$. Далі кожен з отриманих множників піддається черговій факторизації.

Приклад 3.3. Розкласти на множники число 5467.

Розв'язання.

Для першого випробування використовуємо $a_1 = \lceil \sqrt{N} \rceil = \lceil \sqrt{5467} \rceil = 74.$

Обчислюємо $\Delta a_1 = a_1^2 - N = 74^2 - 5467 = 5476 - 5467 = 9.$

Число 441 є повним квадратом. Отже, $a = 74$ і $b = \sqrt{9} = 3.$

Таким чином, множниками числа 5467 є:

$c = a + b = 74 + 3 = 77,\ d = a - b = 74 - 3 = 71.$

Число 71 є простим. Це найбільший множник $< \sqrt{5467}$.

Множник 77 не є простим, тому його потрібно факторизувати.

Для першого випробування використовуємо $a_1 = \lceil \sqrt{N} \rceil = \lceil \sqrt{77} \rceil = 9.$

Обчислюємо $\Delta a_1 = a_1^2 - N = 9^2 - 77 = 81 - 77 = 4.$

Число 4 є повним квадратом. Отже, $a = 9$ і $b = \sqrt{4} = 2.$

Таким чином, множниками числа 77 є:

$c = a + b = 9 + 2 = 11,\ d = a - b = 9 - 2 = 7.$

Числа 7 і 11 є простими.

Відповідь: прості множники 7, 11 і 71.

У своїй найпростішій формі алгоритм Ферма може бути навіть повільнішим, ніж пробне ділення (найгірший випадок). Однак, поєднання пробного ділення та алгоритму Ферма є більш ефективним, ніж кожен з них окремо.

3.6 Метод факторизації Ейлера

Метод Ейлера реалізує розкладання числа на множники шляхом запису його у вигляді суми двох квадратів двома різними способами. Наприклад, число 1000009 можна записати як $1000^2 + 3^2$ або як $972^2 + 235^2$, а метод Ейлера дає факторізацію $1000009 = 293 \cdot 3413.$

Метод факторизації Ейлера є більш ефективним, ніж метод Ферма, для цілих чисел, множники яких не розташовані близько один до одного, і потенційно набагато ефективніший, ніж пробне ділення, якщо можна досить легко знайти подання чисел у вигляді суми двох квадратів. Методи, які використовуються для знаходження подання чисел у вигляді суми двох квадратів, по суті, такі ж, як і для знаходження різниць квадратів у методі факторизації Ферма.

Суттєвим недоліком методу Ейлера є те, що він не може бути застосований до розкладання цілого числа на множники з будь-яким простим множником у формі $4k + 3$, що входить до непарного степеня в його розкладанні на прості множники, оскільки таке число ніколи не може бути сумою двох квадратів. Навіть непарні складені числа виду $4k + 1$ часто є добутком двох простих чисел виду $4k + 3$ (наприклад, $713 = 23 \cdot 31$) і тому не можуть бути розкладені на множники методом Ейлера.

Ця обмежена застосовність зробила метод факторизації Ейлера неприйнятним для комп'ютерних алгоритмів розкладання на множники, оскільки будь-який користувач, який намагається розкласти на множники випадкове ціле число, навряд чи дізнається, чи можна застосувати метод Ейлера до відповідного цілого числа.

3.7 ρ алгоритм Полларда

Метод використовується для розкладання на множники числа $n = pq$, де p – нетривіальний множник.

Формується псевдовипадкова послідовність $\{x_k\}$, така що $x_{i+1} = g(x_i) \bmod n$ для початкового значення x_0. Рекомендують вибирати $g(x) = (x^2 \pm 1)$ або $g(x) = (x^2 \pm a)$. Однак не потрібно застосовувати функції $g(x) = (x^2 - 2)$ та $g(x) = x^2$.

Послідовність $\{x_k\}$ пов'язана з іншою послідовністю $\{x_k \bmod p\}$. Оскільки p не відомо заздалегідь, то послідовність $\{x_k \bmod p\}$ не може бути явно обчислена. Але в цьому полягає основна ідея методу.

Оскільки кількість можливих значень для цих послідовностей є скінченною і послідовність $\{x_k\}$, і послідовність $\{x_k \bmod p\}$ зрештою повторюватимуться, навіть якщо ці значення невідомі. Якщо послідовності складаються з випадкових чисел, то *парадокс дня народження* (*birthday paradox*) стверджує, що число x_k буде повторюватися через кожні $O(\sqrt{N})$ значень, де N – кількість можливих значень. Тому послідовність $\{x_k \bmod p\}$,

ймовірно, повториться набагато раніше, ніж послідовність $\{x_k\}$. Коли знайдуться такі k_1 і k_2, що $x_{k_1} \neq x_{k_2}$, але $x_{k_1} \equiv x_{k_2} \bmod p$, то число $|x_{k_1} - x_{k_2}|$ буде кратним p, і тому буде знайдено множник p.

Факт $x_{k_1} \equiv x_{k_2} \bmod p$ виявляється за допомогою алгоритму Флойда (Floyde) пошуку циклу. Цей алгоритм використовує два покажчики, що переміщуються по послідовності з різною швидкістю. Через це його також називають «алгоритм черепахи та зайця», натякаючи на байку Езопа «Черепаха та заєць».

На кожному кроці алгоритму обчислюється трійка значень (x_i, x_{2i}, d_i), $i = 1, 2, \ldots$ де

$$x_i = g(x_{i-1}) \bmod n,$$

$$x_{2i} = g(g(x_{i-1}) \bmod n) \bmod n,$$

$$d_i = \text{НСД}(n, |x_i - x_{2i}|).$$

Якщо $1 < d_i < n$, то знайдено часткове розкладання числа n, причому $n = d_i \cdot (n / d_i)$. Знайдений дільник d_i може бути складеним, тому його також необхідно факторизувати. Якщо число n / d_i складене, то алгоритм продовжується з модулем $n' = n / d_i$.

Під час реалізації алгоритму може трапитись таке, що послідовності повторюються одночасно. У цьому (нечастому) випадку алгоритм не вдається, і його потрібно повторити з іншим параметром. З урахуванням цього, час реалізації алгоритму пропорційний $n^{1/4}$.

Свою ідею алгоритму факторизації чисел Поллард назвав методом факторизації Монте-Карло через те, що в процесі обчислення генерується псевдовипадкова послідовність чисел. Проте пізніше метод назвали ρ-алгоритмом Полларда. Це було обґрунтовано таким чином. Якщо послідовність має повторювані значення, то послідовність буде циклічно змінюватись, оскільки кожне значення залежить лише від попереднього. Ця структура скінченного циклу дає підстави для назви «алгоритм ρ» через подібність до форми грецької літери ρ, коли значення $\{x_k \bmod p\}$ подано як вузли в орієнтованому графі (рис. 3.2).

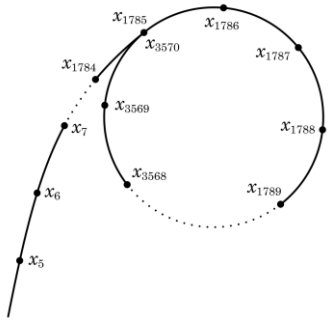


Рисунок 3.2 – Циклічна діаграма, що нагадує грецьку літеру ρ

Приклад 3.4. Розкласти на множники число 2047.

Розв'язання. Нехай $g(x) = (x^2 + 1) \bmod 2047$ і $x_0 = 2$.

Значення x_i для $i = 1,2,3,..$ обчислюються за формулою:

$$x_i = (x_{i-1}^2 + 1) \bmod 2047,$$

а значення x_{2i} – за формулою:

$$x_{2i} = g(g(x_{i-1}) \bmod 2047) \bmod 2047.$$

Результати обчислень для кожної ітерації наведено в табл. 3.4.

Таблиця 3.4 – Результати обчислень

i	x_i	x_{2i}	$\lvert x_i - x_{2i} \rvert$	$d_i = \text{НСД}(2047, \lvert x_i - x_{2i} \rvert)$
0	2	2		
1	5	26	21	1
2	26	1849	1823	1
3	677	1136	365 459	1
4	1849	722	1127	23

Отримано значення $d_3 = 23 > 1$ і відповідно, маємо значення $n / d_3 = 2047 / 23 = 89$. Оскільки отримані числа є простими, то виконання алгоритму завершено.

Відповідь: $p = 23$ і $q = 89$.

Алгоритм дуже швидкий для чисел із малими множниками, але повільніший у випадках, коли всі множники великі.

3.8 *p* - 1 алгоритм Полларда

Це алгоритм спеціального призначення, тобто він призначений лише для цілих чисел із певними типами множників. Знаходяться множники, для яких число, що стоїть перед множником, $p-1$, є степеневим. Алгоритм приводить до концепції безпечних простих чисел.

Нехай *n* – складене ціле число з простим множником *p*. Згідно з малою теоремою Ферма для всіх цілих чисел *a* взаємно простих з *p* і для всіх натуральних чисел *K* маємо:

$$a^{K(p-1)} \equiv 1 \bmod p.$$

Якщо число *x* дорівнює 1 за модулем множника *n*, то $\text{НСД}((x-1), n)$ буде ділитися на цей множник.

Ідея полягає в тому, щоб зробити показник степеня великим кратним $p-1$, зробивши його числом із дуже великою кількістю простих множників. Загалом, береться добуток усіх простих степенів, менших за деяку границю *B*. Починаючи із випадкового *x* відбувається неодноразово заміна його на

$x^w \bmod n$ для *w*, що набуває значення простих степенів. На кожному етапі або один раз наприкінці перевіряється чи не дорівнює 1 значення $\text{НСД}((x-1),n)$.

Можливо, що для всіх простих множників *p* числа *n*, $p-1$ ділиться на маленькі прості числа, тоді $p-1$ алгоритм Полларда просто повертає *n*.

Алгоритм 3.3. *p* -1 алгоритм Полларда

Вхід: складене число *n*.
Вихід: нетривіальний множник числа *n* або «невдача».

***Процес*:**

1. Вибрати границю гладкості *B*.
2. Обчислити $M = \prod_{primes\ q<B} q^{\lfloor \log_q B \rfloor}$.
3. Вибрати випадковим чином *a* взаємно просте з *n*.

   Коментар: Випадковий вибір не є обов'язковим.
   Наприклад, якщо *n* непарне, то завжди можна вибрати *a* = 2.
4. Обчислити $\text{g} = \text{НСД}((a^M - 1),n)$.

   Коментар: Піднесення до степеня можна виконати за модулем n.
5. Якщо $1 < \text{g} < \text{n}$, то повернути *g*.
6. Якщо *g* = 1, то вибрати більше значення *B* і перейти до кроку 2 або повернути «невдача».
7. Якщо *g* = *n*, то вибрати менше значення *B* і перейти до кроку 2 або поверніть «невдача».

Якщо *g* = 1 на кроці 6, то це означає, що немає простих множників *p*, для яких $p-1$ є *B*-ступенево гладким.

Якщо *g* = *n* на кроці 7, то це вказує на те, що всі множники були *B*-ступенево гладкими, але в рідкісних випадках це може вказувати на те, що *a* мав малий порядок за модулем *n*. Крім того, коли максимальні прості множники $p-1$ для кожного простого множника *p* числа *n* однакові в деяких рідкісних випадках, цей алгоритм не працюватиме.

Часова складність цього алгоритму дорівнює $O(B \cdot \log B \cdot \log^2 n)$. Зауважимо, що більші значення *B* сповільнюють його роботу, але з більшою ймовірністю створюють множник.

Приклад 3.5. Розкласти на множники число *n* = 253.

Розв'язання. Вибираємо *B* = 5.

Обчислюємо $M = \prod_{primes\ q<B} q^{\lfloor \log_q B \rfloor} = 2^2 \cdot 3^1 \cdot 5^1 = 60.$

Вибираємо *a* = 2.

Обчислюємо $a^M \bmod n = 2^{60} \bmod 253 = 78$.

Обчислюємо $\text{g} = \text{НСД}((a^M - 1), n) = \text{НСД}(77, 253) = 11.$

Оскільки $1 < 11 < 253$, повертаємо g =11.

$253/11 = 23$ є простим числом.

Відповідь: $253 = 11 \cdot 23.$

3.9 $p+1$ алгоритм Вільямса

$p+1$ алгоритм Вільямса (Williams) є алгоритмом факторизації, який добре працює, коли число *n*, яке потрібно розкласти, містить один або кілька простих множників *p*, таких що $p+1$ є гладким, тобто $p+1$ містить лише малі множники. Він використовує послідовності Люка для виконання піднесення до степеня в квадратичному полі.

Для реалізації алгоритму спочатку потрібно вибрати деяке ціле число *A* більше 2, яке характеризує послідовність Люка:

$$V_0 = 2, V_1 = A, V_j = AV_{j-1} - V_{j-2},$$

де всі операції виконуються за модулем *n*.

Тоді будь-яке непарне просте число *p* ділить $\text{НСД}((V_M - 2), n)$ кожного разу, коли *M* є кратним $p - \left(\frac{D}{p}\right)$, де $D = A^2 - 4$ і $\left(\frac{D}{p}\right)$ є символом Якобі.

Потрібно, щоб $\left(\frac{D}{p}\right) = -1$, тобто *D* має бути квадратичним незалишком за модулем *p*. Але оскільки заздалегідь не відомо значення *p*, то для пошуку рішення може знадобитися більше одного значення *A*. Якщо $\left(\frac{D}{p}\right) = +1$, то цей алгоритм вироджується в повільну версію $p-1$ алгоритму Полларда.

Для різних значень *M* обчислюється $g = \text{НСД}((V_M - 2), n)$, і коли $1 < g < n$, то визначається нетривіальний множник числа *n*.

Використовувані значення *M* є послідовними факторіалами 1!, 2!, 3!, 4!,…, а V_M є *M*-м елементом послідовності seq (V_{M-1}), що характеризується значенням V_{M-1}.

Приклад 3.6. Знайти множник числа 112729, використовуючи *A* = 5.

Розв'язання. Обчислюємо послідовні значення V_M :

V_1 з seq (5) = $V_{1!}$ з seq (5) = 5;

V_2 з seq (5) = $V_{2!}$ з seq (5) = 23;

V_3 з seq (23) = $V_{3!}$ з seq (5) = 12098;

V_4 з seq (12098) = $V_{4!}$ з seq (5) = 87680;

V_5 з seq (87680) = $V_{5!}$ з seq (5) = 53242;

V_6 з seq (53242) = $V_{6!}$ з seq (5) = 27666;

V_7 з seq (27666) = $V_{7!}$ з seq (5) = 110229.

На сьомому кроці маємо

$$g = \text{НСД}((V_M - 2), n) = \text{НСД}((110229 - 2), 112729) = 139,$$

тобто $1 < g < 112729$.

Відповідь: 139 є нетривіальним множником числа 112729.

Зауважте, що $p + 1 = 140 = 2^2 \cdot 5 \cdot 7$. Число $7! = 5040 = 36 \cdot 140$ є найменшим факторіалом, кратним 140, тому правильний множник 139 знайдено на сьомому кроці.

Приклад 3.7. Знайти множник числа 112729 використовуючи $A = 9$.

Розв'язання. Обчислюємо послідовні значення V_M:

V_1 з seq (9) = $V_{1!}$ з seq (9) = 9;

V_2 з seq (9) = $V_{2!}$ з seq (9) = 79;

V_3 з seq (79) = $V_{3!}$ з seq (9) = 41886;

V_4 з seq (41886) = $V_{4!}$ з seq (9) = 79378;

V_5 з seq (79378) = $V_{5!}$ з seq (9) = 1934;

V_6 з seq (1934) = $V_{6!}$ з seq (9) = 10582;

V_7 з seq (10582) = $V_{7!}$ з seq (9) = 84241;

V_8 з seq (84241) = $V_{8!}$ з seq (9) = 93973;

V_9 з seq (93973) = $V_{9!}$ з seq (9) = 91645.

На сьомому кроці маємо

$$g = \text{НСД}((V_M - 2), n) = \text{НСД}((91645 - 2), 112729) = 811,$$

тобто $1 < g < 112729$.

Відповідь: 811 є нетривіальним множником числа 112729.

Зауважте, що $p - 1 = 810 = 2 \cdot 3^4 \cdot 5$. Число $9! = 362880 = 448 \cdot 810$ є найменшим факторіалом, кратним 810, тому правильний множник 811 знайдено на дев'ятому кроці.

Множник 139 цього разу не знайдено, тому що $p - 1 = 138 = 2 \cdot 3 \cdot 23$ не є дільником 9!.

Наведені прикладів показують, що наперед не відомо, чи знайдене просте число має гладке число $p + 1$ чи $p - 1$.

3.10 Спеціальне решето числового поля

Спеціальне решето числового поля (special number field sieve – SNFS) – це алгоритм цілочисельної факторизації спеціального призначення, який є ефективним для цілих чисел виду $r^e \pm s$, де r і s мають малі значення (наприклад, числа Мерсенна). Він також ефективний для будь-яких цілих чисел, які можна подати як поліном з невеликими коефіцієнтами. Це містить цілі числа більш загальної форми $ar^e \pm bs^h$, а також багато цілих чисел, двійкове подання яких має низьку вагу Хеммінга (кількість одиниць у коді).

Алгоритм SNFS широко використовувався NFSNet (волонтерська організація розподілених обчислень), NFS@Home та іншими для розкладання чисел проекту Каннінгема.

Каннінгемський проект (Cunningham Project) – це проект, названий на честь англійського математика Аллана Джозефа Чампніса Каннінгема, який спрямований на розкладання чисел виду $b^n \pm 1$ для b = 2, 3, 5, 6, 7, 10, 11, 12 і великих n. Такі числа називають *числами Каннінгема* і позначають $C^{\pm}(b,n)$. Двома особливо відомими групами чисел Каннінгема в цьому відношенні є числа Ферма, які мають форму $C^{+}(2,2^m)$, і числа Мерсенна, які мають форму $C^{-}(2,n)$.

Встановлення того, чи задане число Каннінгема є простим, було головним напрямком досліджень навколо цього виду чисел. Каннінгем працював над тим, щоб зібрати всі відомі дані про те, які з цих чисел є простими. У 1925 році він опублікував таблиці, які підсумовували його висновки з Г. Дж. Вудолом. Існує три друковані версії таблиці, остання опублікована в 2002 році, а також онлайн-версія Семюеля Вагстаффа.

Національний науковий фонд США (NSF – National Science Foundation) заснував мережу NFSNET, що поєднала кафедри інформатики та індустріальні науково-дослідні лабораторії з ARPANET через комутований доступ та орендовані лінії, об'єднуючи в собі шість суперкомп'ютерних центрів. Також було створено близько двадцяти регіональних мереж, з'єднаних з NFSNET, що дозволило користувачам у тисячах університетів, дослідних лабораторіях, бібліотеках та музеях отримати доступ до суперкомп'ютерів мережі.

3.11 Метод квадратичного решета

Метод квадратичного решета (*quadratic sieve*) – це метод факторизації, запропонований Карлом Померанцем (Carl Pomerance), який на практиці є найшвидшим для чисел до 10^{100}.

Метод базується на такій ідеї. Нехай x і y є такими цілими, що $x^2 \equiv y^2 \bmod n$, але $x \not\equiv \pm y \bmod n$. Тоді n ділить $x^2 - y^2 = (x-y)(x+y)$, але не ділить ані $(x-y)$, ані $(x+y)$. Звідси НСД$((x-y),n)$ має бути нетривіальним дільником n.

Припустимо, що потрібно розкласти ціле число n. Нехай $m = \sqrt{n}$.

Розглянемо багаточлен $q(x) = (x+m)^2 - n$. Зауважимо, що

$$q(x) = x^2 + 2mx + m^2 - n \approx x^2 + 2mx$$

є малим порівняно з *n*, якщо абсолютне значення *x* мале.

Алгоритм квадратичного решета вибирає $a_i = (x+m)$ і перевіряє чи $b_i = (x+m)^2$ є p_t-гладким. Зауважимо, що $a_i = (x+m)^2 \equiv b_i \bmod n$, звідси *n* є квадратичним залишком за модулем *p*. Отже база дільників має складатись з простих чисел *p*, для яких символ Лежандра $\left(\frac{n}{p}\right) = 1$.

Щоб зрозуміти, як працює квадратичне решето, потрібно зрозуміти концепцію вектора степенів. Візьмемо для прикладу число 504. Розкладання на прості множники має вигляд $504 = 2^3 \cdot 3^2 \cdot 5^0 \cdot 7^1$. Це можна подати у вигляді вектора степенів (3,2,0,1), який фіксує степені простих чисел, що беруть участь у розкладанні.

Число є квадратом, якщо кожен елемент у його векторі степенів парний. Коли виконується множення двох чисел, то степені їх розкладань додаються. Наприклад, якщо кожен елемент векторів степенів (3,1,0,1) і (1,3,0,1) обчислити за модулем 2 і виконати додавання відповідних елементів за модулем 2, то отримаємо: $(1,1,0,1) \oplus (1,1,0,1) = (0,0,0,0)$. Тут вектори степенів (3,1,0,1) і (1,3,0,1) відповідають числам 168 і 378, добуток яких 63504 є квадратом числа 252.

Алгоритм 3.4. Квадратичне решето

Вхід: ціле складене число *n*, яке не є степенем простого числа.
Вихід: нетривіальний дільник *d* числа *n*.

***Процес*:**

1. Сформувати базу дільників $S = \{p_1, p_2, \ldots, p_t\}$, де $p_1 = -1$ і p_j $(j \geq 2)$ є $(j-1)$-е просте *p*, для якого *n* є квадратичним залишком за модулем *p*.
2. Обчислити $m = \lfloor \sqrt{n} \rfloor$.
3. Покласти *i* = 1.
4. Поки $i \leq t+1$, виконувати:

   Коментар: Збираються $t+1$ пар (a_i, b_i).

   4.1. Обчислити $b = q(x) = (x+m)^2 - n$.

   Коментар: Значення *x* вибирати в порядку $0, \pm 1, \pm 2, \ldots$

   4.2. Якщо *b* не є p_t-гладким, то вибрати нове значення *x* і перейти до кроку 4.1.

   Коментар: Гладкість перевірити діленням на елементи *S*.

4.3. Якщо b є p_t-гладким ($b = \prod_{j=1}^{t} p_i^{e_{ij}}$), то покласти $a_i = (x+m)$,

$b_i = b$, $v_i = (v_{i1}, v_{i2}, \ldots, v_{it},)$, де $v_{ij} = e_{ij} \bmod 2$ для $i \le j \le t$.

4.4. Покласти $i = i+1$.

5. Вибрати непусту підмножину $T \subseteq 1,2,\ldots,t+1$ таку, що $\sum_{i \in T} v_i = 0 \bmod 2$.

6. Обчислити $x = \left(\prod_{i \in T} a_i \right) \bmod n$

7. Для кожного $1 \le j \le t$ обчислити $l_j = \left(\sum_{i \in T} e_{ij} \right) / 2$.

8. Обчислити $y = \left(\prod_{j=1}^{t} p_j^{l_j} \right) \bmod n$.

9. Якщо $x \equiv \pm y \bmod n$, то перейти до кроку 5, інакше обчислити $d = \text{НСД}((x-y), n)$ і повернути d.

Приклад 3.8. Знайти нетривіальний дільник числа n = 24961.

Розв'язання. Обираємо базу дільників $S = \{-1,2,3,5,13,23\}$ розміру $t = 6$. Прості числа 7, 11, 17 і 19 не входять до складу S, тому що для них символ Лежандра $\left(\frac{n}{p} \right) = -1$.

Обчислюємо $m = \lfloor \sqrt{n} \rfloor = \lfloor \sqrt{24961} \rfloor = 157$.

Результати виконання кроків 4.1 – 4.4 наведено в табл. 3.5.

Таблиця 3.5 – Значеня x, для яких $q(x)$ є 23-гладкими

i	x	$q(x)$	Факторизація $q(x)$	a_i	b_i
1	0	-312	$-2^3 \cdot 3 \cdot 13$	157	(1, 1, 1, 0, 1, 0)
2	1	3	3	158	(0, 0, 1, 0, 0, 0)
3	-1	-625	-5^4	156	(1, 0, 0, 0, 0, 0)
4	2	320	$2^6 \cdot 5$	159	(0, 0, 0, 1, 0, 0)
5	-2	-936	$-2^3 \cdot 3^2 \cdot 13$	155	(1, 1, 0, 0, 1, 0)
6	4	960	$2^6 \cdot 3 \cdot 5$	161	(0, 0, 1, 1, 0, 0)
7	-6	-2160	$-2^4 \cdot 3^3 \cdot 5$	151	(1, 0, 1, 1, 0, 0)

Візьмемо $T = \{1,2,5\}$, який відповідає $(v_1 + v_2 + v_5) = 0 \bmod 2$.

$\oplus$ (1, 1, 1, 0, 1, 0)

$$\begin{array}{r} (0,\,0,\,1,\,0,\,0,\,0) \\ (1,\,1,\,0,\,0,\,1,\,0) \\ \hline (0,\,0,\,0,\,0,\,0,\,0) \end{array}$$

Обчислюємо $x = \left(\prod_{i \in T} a_i \right) \bmod n$:

$$x = (a_1 \cdot a_2 \cdot a_5) \bmod n = (157 \cdot 158 \cdot 155) \bmod 24961 = 936.$$

Обчислюємо $l_1 = 1, l_2 = 3, l_3 = 2, l_4 = 0, l_5 = 1, l_6 = 0$.

Обчислюємо $y = \left(\prod_{j=1}^{t} p_j^{l_j} \right) \bmod n$:

$$y = ((-1)^1 \cdot 2^3 \cdot 3^2 \cdot 5^0 \cdot 13^1 \cdot 23^0) \bmod 24961 = 24025.$$

Оскільки $x \equiv -y \bmod n$ $(936 \equiv -24025 \bmod 24961)$, то потрібно взяти наступну підмножину T. Нехай це буде $T = \{3,6,7\}$, якій відповідає $v_3 + v_6 + v_7 = 0$.

Обчислюємо $x = \left(\prod_{i \in T} a_i \right) \bmod n$:

$$x = (a_3 \cdot a_6 \cdot a_7) \bmod n = (156 \cdot 161 \cdot 151) \bmod 24961 = 23405.$$

Обчислюємо $l_1 = 1, l_2 = 5, l_3 = 2, l_4 = 3, l_5 = 0, l_6 = 0$.

Обчислюємо $y = \left(\prod_{j=1}^{t} p_j^{l_j} \right) \bmod n$:

$$y = ((-1)^1 \cdot 2^5 \cdot 3^2 \cdot 5^3 \cdot 13^0 \cdot 23^0) \bmod 24961 = 13922.$$

Тепер $23405 \not\equiv \pm 13922 \bmod 24961$, тому обчислюємо:
$d = \text{НСД}((x - y), n) = \text{НСД}(9483, 24961) = 109$.
Відповідь: 109 є нетривіальним множником числа 24961.

3.12 Метод раціонального решета

Метод раціонального решета (*rational sieve*) є окремим випадком методу загального решета числового поля. Хоча він менш ефективний, але його суть допомогає зрозуміти, як працює метод загального решета числового поля.

Нехай потрібно розкласти на множники складене число n. Спочатку визначаємо границю гладкості B і базу множників (яку позначатимемо **P**), тобто множину всіх простих чисел, менших або рівних B. Потім шукаємо додатне ціле число z, таке, що як z, так і $z + n$ є B-гладкими, тобто всі їх прості дільники належать **P**. Використовуючи визначені прості дільники можна записати:

$$z = \prod_{p_i \in \mathbf{P}} p_i^{a_i} \; ; \; z + n = \prod_{p_i \in \mathbf{P}} p_i^{b_i} \; ,$$

де a_i і b_i – цілі додатні числа.

Але z і $z+n$ порівнянні за модулем n, і тому кожне таке ціле число z, яке ми знаходимо, дає мультиплікативне відношення (mod n) між елементами **P**, тобто

$$\prod_{p_i \in \mathbf{P}} p_i^{a_i} \equiv \prod_{p_i \in \mathbf{P}} p_i^{b_i} \bmod n \, .$$

Коли буде згенеровано достатню кількість цих співвідношень (зазвичай достатньо, щоб кількість співвідношень була трохи більшою за розмір **P**), то буде можливість використати методи лінійної алгебри, щоб помножити разом ці різні співвідношення таким чином, аби усі показники степеня були парні числа. Це дасть порівняння квадратів $a^2 \equiv b^2 \bmod n$, яке можна перетворити на факторизацію $n = \text{НСД}((a-b),n) \cdot \text{НСД}((a+b),n)$. Ця факторизація може виявитися тривіальною (тобто $n = n \cdot 1$), і в цьому випадку потрібно спробувати добуток іншої комбінації співвідношень. Якщо буде отримано нетривіальну пару множників числа n, то алгоритм припинить роботу.

Приклад 3.9. Розкласти на множники число 187.

Розвязання. Нехай границя гладкості B=7. Маємо таку базу множників $\mathbf{P} = \{2,3,5,7\}$.

Першим кроком є перевірка n на подільність на кожен із членів **P**. Якщо n ділиться на одне з цих простих чисел, то визначено простий множник. Однак 187 не ділиться на 2, 3, 5 і 7.

Далі шукаємо ціле число z, таке, що як z, так і $z+187$ є 7-гладкими. Ось чотири значення z: 2, 5, 9 і 56. Маємо такі чотири мультиплікативні співвідношення (mod 187):

$$2^1 3^0 5^0 7^0 = 2 \equiv 189 = 2^0 3^3 5^0 7^1 ;$$
$$2^0 3^0 5^1 7^0 = 5 \equiv 192 = 2^6 3^1 5^0 7^0 ;$$
$$2^0 3^2 5^0 7^0 = 9 \equiv 196 = 2^2 3^0 5^0 7^2 ;$$
$$2^3 3^0 5^0 7^1 = 56 \equiv 243 = 2^0 3^5 5^0 7^0 .$$

Існує кілька різних способів утворити їх добутки і отримати показники степеня, що є парними числами. Наприклад, 2×56 і відповідно: 189×243.

$$2 \times 56 = (2^1 3^0 5^0 7^0) \times (2^3 3^0 5^0 7^1) = 2^4 3^0 5^0 7^1 ;$$
$$189 \times 243 = (2^0 3^3 5^0 7^1) \times (2^0 3^5 5^0 7^0) = 2^0 3^8 5^0 7^1 .$$

Поділивши на спільний множник 7, отримаємо $2^4 \equiv 3^8 \bmod 187$ або $4^2 \equiv 81^2 \bmod 187$. Отже, маємо таке розкладання на множники:

$$187 = \text{НСД}((81-4),187)\cdot\text{НСД}((81+4),187) = 11\cdot 17.$$

Як альтернатива може бути $2^0 3^2 5^0 7^0 \equiv 2^2 3^0 5^0 7^2 \bmod 187$, що має правильну форму. Це означає, що $3^2 \equiv 14^2 \bmod 187$. Отже, маємо таке розкладання на множники:

$$187 = \text{НСД}((14-3),187)\cdot\text{НСД}((14+3),187) = 11\cdot 17.$$

Відповідь: $187 = 11\cdot 17$.

Недоліком методу раціонального решета є неможливість розкласти числа виду p^m, де p – просте число, а m – ціле додатне число. Однак це не є великою проблемою, оскільки такі числа статистично рідкісні, і, крім того, існує простий та швидкий процес перевірки, чи це число має таку форму.

Найбільшою проблемою є пошук достатньої кількості z, таких, що і z, і $z+n$ є B-гладкими Для будь-якого заданого B частка чисел, які є B-гладкими, швидко зменшується зі збільшенням розміру числа. Отже, якщо n велике (скажімо, сто і більше цифр), буде важко або неможливо знайти достатню кількість z для роботи алгоритму.

3.13 Алгоритм загального решета числового поля

Алгоритм загального решета числового поля (general number field sieve - GNFS) є найефективнішим серед алгоритмів факторизації цілих чисел, більших за 10^{100}. Це робить його найкращим алгоритмом для спроби «зламати» ключі криптосистеми RSA. Алгоритм GNFS був використаний для факторизації 130-значного «проблемного» числа, опублікованого RSA. Це найбільше криптографічне число, яке було коли-небудь факторизовано.

Тоді як алгоритм спеціального решета числового поля забезпечує факторизацію лише чисел певної спеціальної форми, алгоритм загального решета числового поля може розкладати на множники будь-яке число, окрім степенів простих чисел (які тривіально розкласти на множники шляхом вилучення коренів).

Алгоритм решета числового поля (як спеціального, так і загального) базується на вдосконаленні більш простих алгоритмів раціонального решета або квадратичного решета. Використовуючи такі алгоритми для факторизації великого числа n, необхідно шукати гладкі числа (тобто числа з малими простими множниками) порядку $\sqrt{n}$. Значення цих чисел експоненційно зростає зі збільшенням n. З іншого боку, решето числового поля потребує знаходження гладких чисел, що є субекспоненціальними відносно n. Оскільки ці числа менші, тому ймовірність того, що число такого розміру виявиться гладким, вище, що і є причиною ефективності методу решета числового поля.

Є деякі особливості реалізації цього алгоритму.

Просіювання виконується подібно до решета Ератосфена (звідки метод

і отримав свою назву). Решетом слугують прості числа бази множників та їх степені. Під час просіювання число не «викреслюється», а ділиться на число з решета. Якщо внаслідок просіювання число виявиться одиницею, то воно *B*-гладке.

Основна ідея полягає в тому, щоб замість перебору чисел та перевірки, чи діляться їх квадрати за модулем n на прості числа з бази множників, перебираються прості числа з бази і відразу для всіх чисел виду $x^2 - n$ перевіряється, діляться вони на це просте число чи його степень.

Щоб досягти пришвидшення, алгоритм загального решета числового поля передбачає виконання обчислень у числових полях. Це призводить до багатьох досить складних аспектів алгоритму порівняно з більш простим раціональним решетом.

Не всі числа заданої довжини однаково важко розкласти на множники. Найважчими прикладами цих проблем (для відомих на цей момент методів) є напівпрості числа, добуток двох простих чисел. Коли вони обидва великі, наприклад більше двох тисяч бітів, випадково вибрані та приблизно однакового розміру (але не дуже близько, наприклад, щоб уникнути ефективної факторизації методом Ферма).

Для факторизації великих чисел доступні такі ресурси.

https://escatter11.fullerton.edu/nfs/

Дослідницький проект NFS@Home, який використовує комп'ютери, підключені до Інтернету, для виконання кроку решета під час розкладання великих цілих чисел за допомогою решета числового поля. За допомогою NFS@Home можна брати участь у найсучасніших факторизаціях, завантаживши та запустивши безкоштовну програму на своєму комп'ютері. Найновіші великі факторизації проводилися переважно великими кластерами в університетах.

https://cado-nfs.gitlabpages.inria.fr/

CADO-NFS – це повна реалізація в C/C++ алгоритму Number Field Sieve (NFS) для розкладання цілих чисел на множники та обчислення дискретних логарифмів у скінченних полях. Він складається з різних програм, що відповідають усім етапам алгоритму, і загального сценарію, який виконує їх, можливо, паралельно в мережі комп'ютерів. CADO-NFS поширюється за ліцензією Gnu Lesser General Public License (LGPL) версії 2.1 (або будь-якої пізнішої версії).

https://sourceforge.net/projects/msieve/

Msieve – це бібліотека C, яка реалізує набір алгоритмів для розкладання великих цілих чисел. Він містить реалізацію алгоритмів SIQS (Self-Initializing Quadratic Sieve) і GNFS; останнє допомогло завершити деякі з найбільших відомих публічних факторизацій.

Крім практичної цінності, алгоритм GNFS також цікавий з наукового погляду. Алгоритм використовує ідеї та результати з різних галузей математики та інформатики. Алгебраїчна теорія чисел, теорія графів,

скінченні поля, лінійна алгебра і комплексний аналіз – усе це відіграє важливі ролі в GNFS.

> Досі немає жодного алгоритму факторизації P-класу складності. Більшість фахівців сходяться на думці, що такого алгоритму не існує, але останнім часом їхня впевненість трохи похитнулася. Оскільки десь за лаштунками, зовсім поряд, можуть ховатися й інші відкриття, подібні до тесту Агравала-Каяла-Саксени і основані на таких самих простих ідеях, як поліноміальна версія теореми Ферма (і не важливо, що поки про них ніхто навіть не підозрює), може виявитися, що системи шифрування, основані на розкладанні числа на прості множники, не настільки надійні, як хочеться вірити.
>
> Іен Стюарт

? КОНТРОЛЬНІ ПИТАННЯ

1. Який вигляд має канонічний розклад цілого числа на прості множники?
2. Сформулюйте основну задачу арифметики.
3. Як називають дію розкладання цілого числа на прості множники?
4. Чому фундаментальну теорему арифметики також називають теоремою єдиної факторизації.
5. Що таке нетривіальна факторизація?
6. Складність розв'язання якої задачі лежить в основі криптостійкості алгоритмів RSA?
7. На які два типи поділяють алгоритми факторизації?
8. Наведіть приклади алгоритмів факторизації спеціального призначення.
9. Назвіть алгоритми факторизації, які належать до типу загального призначення.
10. Як називають натуральні числа, що є добутком k простих чисел?
11. Як називають натуральне число, що є добутком двох простих чисел?
12. Як називають натуральне число, що є добутком трьох простих чисел?
13. В яких криптосистемах використовують сфенічні числа?
14. Які натуральні числа називають B-гладкими числами?
15. Які натуральні числа називають B-степенево гладкими числами?
16. Які числа називають вільними від квадратів?
17. В чому полягає основна ідея пробного ділення?
18. Хто вперше описав алгоритм пробного ділення?
19. Назвіть кроки процедури пробного ділення.
20. Якому діапазону чисел належать прості числа, що використовуються для пробного ділення?
21. Яким стандартом рекомендована процедура пробного ділення?

22. Чим принципово відрізняється метод колісної факторизації від пробного ділення?

23. На якому поданні непарного цілого числа базується метод факторизації Ферма?

24. Що є суттєвим недоліком методу факторизації Ейлера?

25. Яка відмінність між алгоритмами p-1 Полларда та p+1 Вільямса?

26. Яка відмінність розкладання на множники за методом квадратичного решета від методу факторизації Ферма?

27. Чим відрізняється спеціальне решето числового поля від загального решета числового поля?

ВПРАВИ

1. Випишіть послідовність з 10 елементів, що є 9-майже простими числами.
2. Випишіть послідовність з 20 елементів, що є 11-гладкими числами.
3. Чи є 5-гладким число 60?
4. Чи є число 60 5-степенево гладким числом?
5. Методом пробного ділення визначити прсті множники числа 189.
6. Назвіть найменше сфенічне число.
7. Обчисліть сфенічне число, множниками якого є прості числа 7,11,13.
8. Назвіть приклади безквадратних напівпростих чисел.
9. Розкладіть число 437 на прості множники методом колісної факторизації.
10. Розкладіть число 437 на прості множники методом Ферма.
11. Розкладіть число 437 на прості множники за ρ алгоритмом Полларда.
12. Розкладіть число 437 на прості множники за $p-1$ алгоритмом Полларда.
13. Знайдіть множник числа 32387 за $p+1$ алгоритмом Вільямса для A=9.
14. Знайдіть нетривіальний дільник числа 32387 за алгоритмом квадратичного решета.
15. Розкладіть число 341 на прості множники за алгоритмом раціонального решета.

ДЖЕРЕЛА ДЛЯ ПОГЛИБЛЕНОГО ВИВЧЕННЯ

1. H. M. Bahig, D. I. Nassr, M. A Mahdi,. et al. «Speeding up wheel factoring method». *The Journal of Supercomputing* 78: (2022), pp. 15730-15748.

2. J. P. Buhler, H. W. Lenstra and C. Pomerance, «Factoring Integers with

the Number Field Sieve», In *The Development of the Number Field Sieve*, Springer-Verlag, vol. 1554, (1993), pp. 50-94.

3. D. M. Bressoud, «Factorizations and primality testing», Springer-Verlag, 1989. New York, NY.

4. D. V. Chudnovsky and G. V. Chudnovsky, «Sequences of Numbers Generated by Addition in Formal Groups and New Primality and Factorizations Tests», *Advances in Applied Mathematics*, vol. 7, (1987), pp. 385-434.

5. Fundamental Theorem of Arithmetic https://t5k.org/glossary/page.php?sort=FundamentalTheorem

6. N. Koblitz, «A Course in Number Theory and Cryptography», 2nd edition, Springer-Verlag, 1994, 245 p..

7. R. Lehman. «Factoring large integers». *Math. Comp.*, 28: (1974), pp. 637-646.

8. H. Lenstra Jr. «Factoring integers with elliptic curves». *Ann. of Math.,* 2: (1987), pp. 649-673.

9. A. Lenstra and M. Manasse. «Factoring with two large primes». *Math. Comp.*, 63: (1994), pp. 785-798.

10. Метод факторизації : навч. посіб. для студентів ВНЗ / Г. Я. Попов, В. І. Острик; М-во освіти і науки України, Одес. нац. ун-т ім. І. І. Мечникова. – Одеса : ОНУ, 2014. – 118 с.

11. J. M. Pollard, «Theorems on Factorization and Primality Testing», *Proceedings of the Cambridge Philosophical Society* 76 (1974), pp. 521-528.

12. J. M. Pollard, «A Monte Carlo Method for Factorization», *BIT* 15 (1975), pp. 331-334.

13. C. Pomerance, «Analysis and Comparison of Some Integer Factoring Algorithms», In *Computational Methods in Number Theory*, *Part I*, H.W. Lenstra, Jr. and R. Tijdeman, eds., Math. Centre Tract 154, Amsterdam, 1982, pp 89-139.

14. C. Pomerance, «The quadratic sieve factoring algorithm». In *Advances in cryptology, Proc. Eurocrypt* '84, volume 209 of *Lecture Notes in Computer Science*, pp. 169-182. Springer-Verlag, 1985.

15. H. Reisel, «Prime Numbers and Computer Methods for Factorization». – 2nd edition, Springer, 2012. - 464 p.

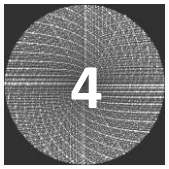
16. Sieve of Eratosthenes https://t5k.org/glossary/page.php?sort=SieveOfEratosthenes

17. Song Y. Yan, «Primality Testing and Integer Factorization in Public-Key Cryptography». – Springer Science & Business Media, 2004. – 236 p.

18. trial division https://t5k.org/glossary/page.php?sort=TrialDivision

19. H. C. Williams, «A p+1 method of factoring», *Mathematics of Computation*, 39 (159): (1982), pp. 225–234.

20. wheel factorization https://t5k.org/glossary/page.php?sort=WheelFactorization



ТЕСТИ ПРОСТОТИ

У загальному розумінні перевірка простоти додатного цілого числа n (тест простоти **T**) – це процедура, яка виводить $\mathbf{T}(n) \in \{true, false\}$, щодо того, число n є простим чи складеним.

Існує два види тестів простоти: істинні та ймовірнісні. Для цілого числа кандидата істинні тести простоти (*true primality tests*) з математичним доведенням роблять висновок, що кандидат є простим числом, тоді як імовірнісні тести простоти (*probabilistic primality tests*) оголошують слабший результат, що кандидат, ймовірно, просте число з певною ймовірністю помилки.

4.1 Істинні тести простоти

Тести простоти, за допомогою яких можна довести (*proven*), що додатні цілі числа є простими, часто називають алгоритмами доведення простоти (*primality proving algorithms*). Ці тести, як правило, мають більшу обчислювальну складність, ніж імовірнісні тести простоти.

Означення 4.1. Ціле число n, яке визначено як просте на основі алгоритму доведення простоти, називають *доказово простим числом* (*provable prime*).

4.1.1 Перевірка простоти пробним діленням

Щоб перевірити, чи є ціле додатне число n простим, можна використати пробне ділення. Для цього потрібно поділити число n на всі прості числа, що менші або дорівнюють $\sqrt{n}$. Якщо дільник знайдено, то число n є складеним, а інакше – простим.

Для перевірки простоти будь-якого цілого числа n, довжина якого не перевищує 10 цифр, стандартом NIST.FIPS.186-4 рекомендовано процедуру пробного ділення, що передбачає виконання таких дій.

1. Підготувати набір простих чисел $\le \sqrt{n}$, застосувавши процедуру просіювання.

2. Поділити n на кожне просте число з набору. Якщо n ділиться на одне з простих чисел, то оголосити, що n є складеним, і завершити процедуру.

3. В іншому випадку оголосити, що n є простим числом і завершити процедуру.

Процедура просіювання ідентифікує цілі числа послідовності $Y_0, Y_0 + 1, \ldots, Y_0 + J$, які діляться на прості числа p_j від 2 до L (база

факторів). Значення L є довільним. Стандарт NIST.FIPS.186-4 рекомендує типове значення L в діапазоні від 10^3 до 10^5. Якщо база факторів складається з простих чисел $p_1, p_2, ..., p_k$, то $J = \prod_{j=1}^{k} p_j$. Число Y_0 дорівнює або 1, або p_{k+1}.

Алгоритм 4.1. Решето +1

Вхід: ціле число n.

Вихід: набір простих чисел $\le \sqrt{n}$.

***Процес*:**

1. Визначити базу факторів $\{p_1, p_2, ..., p_k\}$, що задовольняють умову: $\prod_{j=1}^{k} p_j \le \sqrt{n}$.
2. Вибрати значення Y_0.
3. Обчислити $S_j = Y_0 \bmod p_j$ для всіх p_j бази факторів.
4. Ініціалізувати масив довжиною J+1, кожен елемент якого є 0.
5. Починаючи з $Y_0 - S_j + p_j$, кожному наступному p_j-му елементу масиву присвоїти значення 1. Зробити це для всієї довжини масиву та для кожного j.
6. Після завершення кожний елемент масиву, який має значення 0, вказує на те, що відповідне число є простим.

Масив може бути або бітовим масивом для компактності, коли пам'ять невелика, або байтовим масивом для швидкості, коли пам'ять легко доступна. Немає необхідності просіювати відразу весь інтервал решета. Масив можна розділити на частини, просіюючи кожну частину перед тим, як переходити до наступної частини.

Приклад 4.1. Підготувати набір простих чисел для перевірки на простоту чисел, що не перевищують 1000.

Розв'язання. Обчислюємо $\lfloor \sqrt{1000} \rfloor = 31$.

Визначаємо базу факторів: $p_1 = 2, p_2 = 3, p_3 = 5$, оскільки $2 \cdot 3 \cdot 5 = 30 < 31$. Цьому набору відповідає $J = 30$.

Нехай Y_0=1. Обчислюємо $S_j = Y_0 \bmod p_j$ для всіх p_j бази факторів.

$S_1 = 1 \bmod 2;\ S_2 = 1 \bmod 3;\ S_3 = 1 \bmod 5.$

Створюємо масів довжиною $J + 1 = 30 + 1 = 31$.

Номер елемента	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Значення елемента	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Номер елемента	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Значення елемента	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Починаючи з $Y_0 - S_1 + p_1 = 1 - 1 + 2 = 2$, кожному наступному 2-му елементу масиву (тобто елементам з номерами 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28 і 30) присвоюємо значення 1.

Номер елемента	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Значення елемента	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Номер елемента	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Значення елемента	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

Починаючи з $Y_0 - S_2 + p_2 = 1 - 1 + 3 = 3$, кожному наступному 3-му елементу масиву (тобто елементам з номерами 6, 9, 12, 15, 18, 21, 24, 27 і 30) присвоюємо значення 1.

Номер елемента	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Значення елемента	0	0	0	1	0	1	0	1	1	1	0	1	0	1	1	1

Номер елемента	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Значення елемента	0	1	0	1	1	1	0	1	0	1	1	1	0	1	0

Починаючи з $Y_0 - S_3 + p_3 = 1 - 1 + 5 = 5$, кожному наступному 5-му елементу масиву (тобто елементам з номерами 10, 15, 20, 25 і 30) присвоюємо значення 1.

Номер елемента	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Значення елемента	0	0	0	1	0	1	0	1	1	1	0	1	0	1	1	1

Номер елемента	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Значення елемента	0	1	0	1	1	1	0	1	1	1	1	1	0	1	0

Тут значення 0 мають елементи 1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 і 31, які є простими числами, окрім 1.

Відповідь: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31.

Приклад 4.2. Чи є число 233 простим?

Розв'язання. $\lfloor\sqrt{233}\rfloor$=15. Отже, маємо такий набір простих чисел для пробного ділення: 2, 3, 5, 7, 11 і 13.

2 не ділить 233;
3 не ділить 233;
5 не ділить 233;
7 не ділить 233;
11 не ділить 233;
13 не ділить 233.
Зупинка процесу пробного ділення.

Відповідь: Число 233 є простим.

Приклад 4.3. Чи є число 1001 простим?

Розв'язання. $\lfloor\sqrt{1001}\rfloor = 31$. Отже, маємо такий набір простих чисел для пробного ділення: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31.

2 не ділить 1001;
3 не ділить 1001;
5 не ділить 1001;
101:7 = 143; 7 ділить 1001.
Зупинка процесу пробного ділення.
Відповідь: Число 1001 є складеним.

Пробне ділення можна реалізовувати паралельно, оскільки наступний крок ділення не використовує результат попереднього ділення. Це забезпечує можливість пришвидшення процесу перевірки на простоту.

У разі перевірки простоти великого числа пробне ділення може бути використане для попереднього відбору. Тобто спочатку виконується пробне ділення на певну кількість малих простих чисел, а потім застосовується певний тест простоти.

4.1.2 Тест простоти AKS

У серпні 2002 року М. Агравал та його колеги оголосили про детермінований алгоритм перевірки простоти, який виконується за поліноміальний час. Цей тест, відомий як AKS (Agrawal-Kayal-Saxena), базується на такому твердженні.

Твердження 4.1. *Нехай* $n \geq 2$ *– ціле число,* $a < n$ *– ціле число, таке що* $\text{НСД}(a,n) = 1$. *Тоді n є простим тоді і тільки тоді, коли* $(x+a)^n \equiv (x^n + a) \bmod n$.

Відзначимо, що твердження 4.1 само собою забезпечує перевірку простоти з поліноміальною складністю. Однак, складність обчислення коефіцієнтів є експоненціальною за значенням *n*, що робить цей тест надто повільним, щоб використовувати його для тестування великих чисел, необхідних для криптографічних систем. Фактично, перевірка пробним діленням є швидшою.

Приклад 4.4. Перевірити, чи є простим число 9.

Розв'язання. Обчислюємо $(x+1)^9$:

$$(x+1)^9 = x^9 + 9x^8 + 36x^7 + 84x^6 + 126x^5 + 126x^4 +$$

$+84x^3+36x^2+9x+1.$

Коефіцієнти 84 не діляться на 9, тому

$$(x+1)^9 \equiv (x^9+3x^6+3x^3+1) \bmod 9,$$

а отже, число 9 є складеним.

Відповідь: число 9 є складеним.

Зверніть увагу, що відмінні від нуля коефіцієнти відповідають степеням *x*, які діляться на простий множник числа 9.

Приклад 4.5. Перевірити, чи є простим число 11.

Розв'язання. Для зручності вибираємо $a=1.$

Обчислюємо $(x+1)^{11}$:

$$(x+1)^{11} = x^{11}+11x^{10}+55x^9+165x^8+330x^7+462x^6+462x^5+330x^4+$$

$$+165x^3+55x^2+11x+1.$$

Коефіцієнти 11, 55, 165, 330 і 462 діляться на 11, тому

$$(x+1)^{11} \equiv (x^{11}+1) \bmod 11.$$

Відповідь: число 11 є простим.

Основним проривом тесту AKS є оцінювання поліномів з обох сторін рівняння в твердженні 4.1 за модулем іншого полінома x^r-1 і перевірка кількох різних *a*. Тобто перевірка конгруентності

$$(x+a)^n \equiv (x^n+a) \bmod ((x^r-1), n).$$

Це знижує степінь поліномів і зменшує кількість коефіцієнтів, що покращує ефективність алгоритму.

Алгоритм 4.2. Тест простоти AKS

***Вхід*:** ціле число $n \geq 2$.

***Вихід*:** *composite* або *prime*.

***Процес*:**

1. Якщо $n = a^b$ для $a \in \mathbf{N}$ і $b > 1$, то повернути *composite*.
2. Знайти найменше *r* таке, що $ord_r(n) > (\log_2 n)^2$.
3. Якщо $1 < \text{НСД}(a,n) < n$ для деякого $a \le r$, то повернути *composite*.
4. Якщо $n \le r$, то повернути *prime*.
5. Для *a* від 1 до $\lfloor \log_2 n\sqrt{\phi(r)} \rfloor$ виконувати:
6. Якщо $(x+a)^n \not\equiv (x^n + a) \bmod((x^r - 1), n)$, то повернути *composite*.
7. Повернути *prime*.

На кроці 1 використовується алгоритм 4.3, що реалізує тестування на досконалі степені.

Означення 4.2. Ціле число *n* є *досконалим степенем* (*perfect power*) тоді і тільки тоді, коли існують $a, b \in \mathbf{N}$ з $b > 1$ такі, що $n = a^b$.

Приклади досконалих степенів: $81 = 3^4$ і $117649 = 7^6$.

На кроці 2 для послідовних значень *r*, починаючи з *r* = 2, перевіряється умова $n^k \equiv 1 \bmod r$ для кожного $1 \le k \le (\log_2 n)^2$. Якщо жодне зі значень *k* не відповідає 1, то знайдено найменше *r*, інакше збільшити *r* на 1 і повторити спробу.

Алгоритм 4.3. Тестування на досконалі степені

***Вхід*:** ціле число $n \ge 2$.

***Вихід*:** *perfect power* ($p = m^b$) або *not a perfect power*.

***Процес*:**

1. Покласти *b* = 2.
2. Поки $2^b \le n$ виконувати:
   2.1. Покласти *a* = 1, *c* = *n*.
   2.2. Поки $c - a \ge 2$ виконувати:
      2.2.1. Обчислити $m = \lfloor (a + c)/2 \rfloor$.
      2.2.2. Обчислити $p = \min\{m^b, n+1\}$.
      2.2.3. Якщо *p* = *n*, то повернути *perfect power* ($p = m^b$).
      2.2.4. Якщо *p* < *n* , то покласти *a* = *m*, інакше покласти *c* = *m*.
   2.3. Завершити «Поки».
   2.4. Покласти *b* = *b* + 1.
3. Завершити «Поки».
4. Повернути *not a perfect power*.

Тест AKS став великим теоретичним проривом у галузі тестування на простоту. Це є елегантним підходом, який довів, що проблема PRIMES має поліноміальну складність, тобто знаходиться в класі складності P. Через це тест AKS завжди згадують, коли мова йде про тестування на простоту. Фахівці з математики та інформатики вважають його маяком надії на те, що нерозв'язані проблеми одного разу можуть бути вирішені.

4.1.3 Тест Люка-Лемера

Відомі ефективні алгоритми для перевірки простоти деяких спеціальних класів чисел, таких як числа Мерсенна та числа Ферма. Прості числа Мерсенна *n* є корисними, оскільки арифметику в полі $\mathbf{Z}_n$ для такого *n* можна реалізувати достатньо ефективно.

⚠ **Означення 4.3.** Ціле число виду $M_p = 2^p - 1$, де $p \geq 2$, називають *числом Мерсенна* (*Mersenne number*).

⚠ **Означення 4.4.** Просте число виду $2^p - 1$ називають *простим числом Мерсенна* (*Mersenne prime*).

Число Мерсенна $M_p = 2^p - 1$ є простим тоді і тільки тоді, коли виконуються умови:

- *p* є простим;
- у послідовності цілих чисел, що визначена $u_{k+1} = (u_k^2 - 2) \bmod M_p$ для $u_0 = 4$, $(s-2)$-й елемент задовольняє $u_{s-2} \equiv 0 \bmod M_p$.

Простоту числа *p* можна ефективно перевірити, наприклад, за допомогою алгоритму пробного ділення, оскільки *p* експоненціально менше, ніж M_p.

Алгоритм 4.4. Тест простоти Люка-Лемера для чисел Мерсенна

***Вхід*:** число Мерсенна $M_p = 2^p - 1$, де $p \geq 3$.

***Вихід*:** *prime* або *composite*.

***Процес*:**

1. Виконати пробне ділення для перевірки простоти *s*, використовуючи всі прості числа, що менші або дорівнюють $\sqrt{s}$. Якщо є просте число, яке ділить *s*, то повернути *composite*.
2. Покласти *u* = 4.
3. Для *k* від 1 до *s* − 2 обчислювати $u = (u^2 - 2) \bmod M_p$.
4. Якщо $u = 0$, то повернути *prime*, в іншому випадку повернути *composite*.

Приклад 4.6. Нехай $s = 7$. Чи є простим число Мерсенна $M_7 = 2^7 - 1 = 127$?

Розв'язання. Початкове значення *u* = 4.

Обчислення $u = (u^2 - 2) \bmod 127$:

$k = 1,\ u = (4^2 - 2) \bmod 127 = 14;$

$k = 2,\ u = (14^2 - 2) \bmod 127 = 67;$

$k = 3,\ u = (67^2 - 2) \bmod 127 = 42;$

$k = 4,\ u = (42^2 - 2) \bmod 127 = 111;$

$k = 5,\ u = (111^2 - 2) \bmod 127 = 0.$

Відповідь: Оскільки останнє значення *u* дорівнює 0, то число Мерсенна 127 є простим.

Приклад 4.7. Нехай $p = 11$. Чи є простим число Мерсенна $M_{11} = 2^{11} - 1 = 2047$?

Розв'язання. Початкове значення *u* = 4.

Обчислення $u = (u^2 - 2) \bmod 2047$:

$k = 1,\ u = 4^2 - 2) \bmod 2047 = 14;$

$k = 2,\ u = (14^2 - 2) \bmod 2047 = 194;$

$k = 3,\ u = (194^2 - 2) \bmod 2047 = 788;$

$k = 4,\ u = (788^2 - 2) \bmod 2047 = 701;$

$k = 5,\ u = (701^2 - 2) \bmod 2047 = 119;$

$k = 6,\ u = (119^2 - 2) \bmod 2047 = 1877;$

$k = 7,\ u = (1877^2 - 2) \bmod 2047 = 240;$

$k = 8,\ u = (240^2 - 2) \bmod 2047 = 282;$

$k = 9,\ u = (282^2 - 2) \bmod 2047 = 1736;$

Відповідь: Оскільки останнє значення *u* не дорівнює 0, то число Мерсенна 2047 не є простим ($2047 = 23 \times 89$).

Тест Люка-Лемера є одним із основних тестів на простоту, який використовується широкомасштабним Інтернет-проєктом добровольчих обчислень з пошуку великих простих чисел Мерсенна GIMPS (Great Internet Mersenne Prime Search) https://www.mersenne.org/. Цей пошук був успішним у виявленні багатьох найбільших простих чисел, відомих на сьогоднішній день. Тест вважається цінним, оскільки він може перевірити на простоту великий набір дуже великих чисел протягом прийнятного періоду часу.

4.2 Сертифікати про простоту числа

У повсякденному житті сертифікат – це документ, що засвідчує правдивість твердження. Наприклад, диплом – це сертифікат про те, що ви здобули певний ступінь освіти. Він містить таку інформацію, як назва навчального закладу, що видав сертифікат, і підписи, за допомогою яких можна перевірити сертифікат.

Сертифікат про простоту числа p (*certificate of primality*) є коротким викладом доведення того, що p є простим.

Сертифікат має містити достатньо інформації, щоб відтворити доведення. Крім того, він має бути коротким (або принаймні суттєво коротшим за вихідний алгоритм). Наприклад, можна показати за допомогою пробного ділення, що число є простим. Однак це не забезпечує сертифікат простоти, оскільки для перевірки доказу потрібно відтворити всі обчислення. Хоча пробне ділення надає сертифікат складеності (якщо число є складеним): сертифікат буде найменшим знайденим простим дільником. Потім легко перевірити цей сертифікат, поділивши число на цей дільник.

4.2.1 Сертифікат Пратта

Концепцію сертифікатів простоти запропонував Пратт у 1975 р. Він описав структуру сертифіката і довів, що його розмір поліноміально залежить від довжини запису числа, а також, що він може бути верифікований за поліноміальний час. Сертифікат ґрунтується на теоремі Люка, яка випливає з малої теореми Ферма.

Теорема Люка. *Число p є простим тоді і лише тоді, коли в кільці залишків за модулем p існує елемент $a \in \mathbf{Z}_p$ такий, що*:

1. $a^{p-1} \equiv 1 \bmod p$.
2. $a^{\frac{p-1}{q}} \not\equiv 1 \bmod p$ *для всіх простих чисел q, які ділять* $p-1$.

Зауважимо, що, крім наведених умов, потрібно також перевірити, що числа q, подані в сертифікаті як прості, дійсно такими є. Отже, сертифікат простоти числа p крім свідка простоти a має також містити сертифікати простоти всіх зазначених у ньому співмножників q. Зручним поданням такого сертифіката є дерево, у вузлах якого знаходяться прості числа та відповідні їм свідки простоти, а нащадками вузла, в якому зберігається число p, є прості дільники q.

Враховуючи, що кожен вузол дерева зберігає $O(\log p)$ бітів для запису чисел у ньому, загальний розмір дерева не перевищує $O(\log^2 p)$. Загальний час перевірки оцінюється як $O(\log^5 p)$.

Приклад 4.8. Створити сертифікат про простоту числа n=683.

Розв'язання. Перевірка умов для числа 683.
Простими дільниками числа 682=683-1 є 2, 11 і 31.

a=2: $2^{682} \equiv 1 \bmod 683$; 682/2=341, $2^{341} \equiv 682 \bmod 683$; 682/11=62, $2^{62} \equiv 555 \bmod 683$; 682/31=22, $2^{22} \equiv 1 \bmod 683$. Тут умова 2 не виконується.

a=3: $3^{682} \equiv 1 \bmod 683$; $3^{341} \equiv 1 \bmod 683$. Тут умова 2 не виконується.

a=5: $5^{682} \equiv 1 \bmod 683$; $5^{341} \equiv 682 \bmod 683$; $5^{62} \equiv 651 \bmod 683$; $5^{22} \equiv 76 \bmod 683$. Тут виконуються всі умови. Тому число 5 є свідком числа 683.

Число 2 є так званим «самосвідком», тобто його визнають простим числом без доведення.

Перевірка умов для числа 11.
Простими дільниками числа 10=11-1 є 2 і 5.

a=2: $2^{10} \equiv 1 \bmod 11$; 10/2=5, $2^5 \equiv 10 \bmod 11$; 10/5=2, $2^2 \equiv 4 \bmod 11$.

Тут виконуються всі умови. Тому число 2 є свідком числа 11.
Перевірка умов для числа 31.
Простими дільниками числа 30=31-1 є 2, 3 і 5.

a=2: $2^{30} \equiv 1 \bmod 31$; 30/2=15, $2^{15} \equiv 1 \bmod 31$. Тут умова 2 не виконується.

a=3: $3^{30} \equiv 1 \bmod 31$; $3^{15} \equiv 30 \bmod 31$; 30/3=10, $3^{10} \equiv 25 \bmod 31$; 30/5=6, $3^6 \equiv 16 \bmod 31$. Тут виконуються всі умови. Тому число 3 є свідком числа 31.

Крім цього потрібно перевірити умови для чисел 3 і 5, які також є дільниками.

Перевірка умов для числа 3.
Простим дільником числа 2=3-1 є 2.

a=2: $2^2 \equiv 1 \bmod 3$; 2/2=1, $2^1 \equiv 2 \bmod 3$. Виконуються всі умови. Тому число 2 є свідком числа 3.

Перевірка умов для числа 5.
Простим дільником числа 4=5-1 є 2.

a=2: $2^4 \equiv 1 \bmod 5$; 4/2=2, $2^2 \equiv 4 \bmod 5$. Виконуються всі умови. Тому число 2 є свідком числа 5.

Відповідь: Дерево, що є графічним зображенням сертифіката про простоту числа n=683, наведено на рис. 4.1.

Варто відзначити, що сертифікати Пратта корисні в теорії та легко перевіряються, однак знаходження сертифіката для числа n потребує

факторизації *n*-1 та інших потенційно великих чисел. Це нескладно зробити в деяких окремих випадках, наприклад, для простих чисел Ферма, але в загальному випадку це завдання набагато складніше звичайної перевірки числа на простоту.

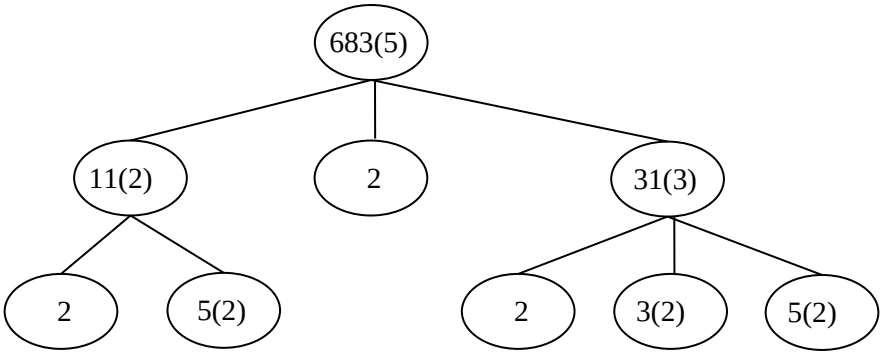


Рисунок 4.1 – Сертифікат про простоту числа *n*=683

4.2.2 Алгоритм Голдвассера–Кіліана

Голдвассер (Goldwasser) і Кіліан (Kilian) запропонували підхід до побудови сертифіката про простоту числа *N*, який базується на використанні еліптичних кривих. Цей підхід передбачає виконання таких дій.

Вибрати навмання три цілі числа: *a*, *x*, *y* і обчислити:

$$b \equiv (y^2 - x^3 - ax) \bmod N .$$

Тепер $P(x, y)$ – це точка на еліптичній кривій *E*, що описується як $y^2 = x^3 + ax + b$. Далі виконується алгоритм підрахунку кількості точок на кривій *E* над $\mathbf{F}_N$, який завершується результатом *m*, якщо *N* є просте число.

Якщо цей алгоритм зупиняється, то це означає, що може бути отриманий нетривіальний множник числа *N*.

Якщо алгоритм завершується вдало, то далі застосовується критерій щодо прийнятності кривої *E*.

Якщо $m = kq$, де $k \geq 2$ – невелике ціле число, а *q* – велике ймовірно просте число (число, яке проходить перевірку ймовірнісної простоти), то крива *E* не відкидається. Інакше ця крива відкидається та навмання вибирається інша трійка (*a*, *x*, *y*), щоб почати спочатку.

Ідея полягає в тому, щоб знайти *m*, яке ділиться на велике просте число *q*. Це просте число на кілька цифр менше, ніж *m* (або *N*), тому буде легше довести, що *q* є простим числом, ніж *N*.

Коли знайдено криву, що відповідає критерію, обчислюються добутки *mP* і *kP*. Якщо будь-який з них дає невизначений результат, то може бути отриманий нетривіальний множник числа *N*. Якщо обидва обчислення вдалі, то перевіряються результати.

Якщо $mP \neq 0$, то *N* не є простим. (Якщо *N* є простим, то крива *E* має порядок *m*, і тому добуток будь-якої точкі *P* на *m* дорівнює 0).

Якщо $kP = 0$, то відкидається крива E і все починається з іншої трійки (a, x, y).

Якщо $mP = 0$ і $kP \neq 0$, то N є простим числом.

Однак ще потрібно довести простоту числа q. Це доводиться за таким самим алгоритмом.

Таким чином, ми маємо справу з рекурсивним алгоритмом, де простота числа N залежить від простоти q і справді менших «ймовірно простих чисел», доки не буде досягнуто деякого порогу, коли q вважається достатньо малим щоб мати можливість застосувати нерекурсивний детермінований алгоритм.

Алгоритм Голдвассера–Кіліана має два недоліки: складність підрахування всіх точок на кривій E і важко знайти криву E, кількість точок якої має вигляд kq.

4.2.3 Алгоритм Аткіна-Морейна

Для усунення першого недоліку алгоритму Голдвассера–Кіліана Аткін (Atkin) і Морейн (Morain) запропонували будувати криву E, для якої можна легко обчислити кількість точок на кривій E.

Їхня ідея полягає в тому, щоб замість випадкового генерування еліптичних кривих і пошуку правильного значення m, побудувати криву E, для якої кількість точок легко обчислити. Побудова такої кривої базується на спеціальній операції, яку називають *комплексне множення* (*Complex Multiplication* – CM).

Перш ніж розглянути безпосередньо комплексне множення, наведемо кілька алгоритмів, які будуть потрібні в подальшому.

Алгоритм 4.5. Піднесення до степеня за модулем

***Вхід*:** додатне ціле число v, що має двійкове подання $v = v_r v_{r-1} \ldots v_1 v_0$, де старший біт v_r дорівнює 1, модуль m і ціле число g за модулем m.

***Вихід*:** $g^v \bmod m$.

***Процес*:**

1. Покласти $x = g$.
2. Для i від $r - 1$ до 0 виконати:
   2.1. Обчислити $x = x^2 \bmod m$.
   2.2. Якщо $v_i = 1$, то обчислити $x = g \cdot x \bmod m$.
3. Вивести x.

Алгоритм 4.6. Створення послідовностей Люка

***Вхід*:** ціле непарне число $n > 2$, цілі числа P і Q і ціле додатне число k, що має двійкове подання $k = k_r k_{r-1} \ldots k_1 k_0$, де старший біт k_r дорівнює 1.

***Вихід*:** $v_0 = V_k \bmod n$ і $q_0 = Q^{\lfloor k/2 \rfloor} \bmod n$.

***Процес*:**

1. Покласти $v_0 = 2$, $v_1 = P$, $q_0 = 1$, $q_1 = 1$.

2. Для i від r до 0 виконати:

2.1. Обчислити $q_0 = q_0 q_1 \bmod n$.

2.2. Якщо $k_i = 1$, то обчислити:

$q_1 = q_0 Q \bmod n$;

$v_0 = (v_0 v_1 - P q_0) \bmod n$;

$v_1 = (v_1^2 - 2q_1) \bmod n$.

В іншому випадку обчислити:

$q_1 = q_0$;

$v_0 = (v_0^2 - 2q_0) \bmod n$;

$v_1 = (v_0 v_1 - P q_0) \bmod n$.

3. Вивести (v_0, q_0).

Алгоритм 4.7. Обчислення квадратного коріня z за модулем p

***Вхід*:** непарне просте число p і ціле число g з інтервалу $0 < g < p$.

***Вихід*:** квадратний корінь за модулем p з g, якщо він існує. У разі відсутності квадратного кореня повертається «квадратних коренів не існує».

***Процес*:**

I. $p \equiv 3 \bmod 4$; тобто $p = 4k + 3$ для деякого цілого додатного числа k.

1. Обчислити і вивести $z = g^{k+1} \bmod p$.

Коментар: Застосувати алгоритм 4.5.

II. $p \equiv 5 \bmod 8$; тобто $p = 8k + 5$ для деякого цілого додатного числа k.

1. Обчислити $\gamma = (2g)^k \bmod p$.

Коментар: Застосувати алгоритм 4.5.

2. Обчислити $i = 2g\gamma^2 \bmod p$.

3. Обчислити та вивести $z = g\gamma(i - 1) \bmod p$.

III. $p \equiv 1 \bmod 8$.

1. Покласти $Q = g$.

2. Вибрати навмання значення P з інтервалу $0 < P < p$.

3. Обчислити $V = V_{(p+1)/2} \bmod p$ і $Q_0 = Q^{(p-1)/4} \bmod p$.

Коментар: Застосувати алгоритм 4.6.

4. Обчислити $z = (V / 2) \bmod p$.

5. Якщо $(z^2 \bmod p) = g$, то вивести z і завершити.

6. Якщо $1 < Q_0 < p - 1$, то вивести повідомлення «квадратних коренів не існує» і завершити.

7. Перейти до кроку 2.

Реалізуючи цей алгоритм потрібно враховувати такі його особливості.

1. У випадку I та випадку II алгоритм повертає розв'язок z за умови, що він існує (це відомо з умов розв'язуваної задачі). Якщо невідомо чи існує розв'язок z, то вихід z потрібно перевірити шляхом порівняння $w = z^2 \bmod p$

з g. Якщо $w = g$, то z є розв'язком, інакше не існує рішень. У випадку III алгоритм визначає існування чи відсутність розв'язку.

2. Ділення цілого числа V на 2 за модулем p, що реалізується на кроці 4 випадку III, можна виконати для числа V або $V + p$ (залежно від того, яке з них парне) шляхом зсуву вправо на один розряд двійкового коду цього числа.

3. У випадку III заданий вибір P дасть розв'язок тоді і тільки тоді, коли $P^2 - 4Q$ не є квадратичним залишком за модулем p. Якщо P вибирається навмання, ймовірність цього становить не менше 1/2. Таким чином, буде потрібно лише кілька значень P. Тому для пришвидшення процесу достатньо використання малих значень P. У такому випадку множення на P в алгоритмі 4.6. можна виконати шляхом повторного додавання.

Алгоритм 4.8. Піднесення полінома до степеня за модулем полінома

***Вхід*:** ціле додатне число k, що має двійкове подання $k = k_r k_{r-1} \ldots k_1 k_0$, де старший біт k_r дорівнює 1; поле $GF(q)$; поліноми $f(t)$ і $m(t)$ з коефіцієнтами в $GF(q)$.

***Вихід*:** поліном $f(t)^k \bmod m(t)$.

***Процес*:**

1. Покласти $u(t) = f(t) \bmod m(t)$.
2. Для i від $r - 1$ до 0 виконати:
   2.1. Обчислити $u(t) = u(t)^2 \bmod m(t)$.
   2.2. Якщо $k_i = 1$, то обчислити $u(t) = u(t) f(t) \bmod m(t)$.
3. Вивести $u(t)$.

Алгоритм 4.9. Розкладання поліномів над $GF(p)$

***Вхід*:** просте число $p > 2$, ціле додатне число d і поліном $f(t)$, що розкладається за модулем p на різні незвідні поліноми степеня d.

***Вихід*:** випадковий множник степеня d полінома $f(t)$.

***Процес*:**

1. Покласти $g(t) = f(t)$.
2. Поки $\deg(g) > d$ виконувати:
   2.1. Виберіть $u(t)$ як випадковий монічний поліном степеня $2d - 1$.
   2.2. Обчислити $c(t) = u(t)^{(p^d - 1)/2} \bmod g(t)$.
   Коментар: Застосувати алгоритм 4.8
   2.3. Покласти $h(t) = \text{НСД}((c(t) - 1), g(t))$.
   2.4. Якщо $h(t)$ константа або $\deg(g) = \deg(h)$, то перейти до кроку 2.1.
   2.5. Якщо $2\deg(g) > \deg(h)$, то покласти $g(t) = g(t)/h(t)$; інакше $g(t) = h(t)$.
3. Вивести $g(t)$.

Найважча процедура генерування параметрів еліптичної кривої – це знаходження базової точки простого порядку. Для цього потрібно знати порядок кривої $n = \#E(GF(q))$ і розв'язати таку задачу: «Задано поле $\mathbf{F} = GF(q)$, знайти еліптичну криву, визначену над **F**, порядок якої ділиться на достатньо велике просте число *r*. (Поняття «достатньо велике» визначається в термінах бажаного рівня безпеки).

Якщо *n* є порядком еліптичної кривої $y^2 = x^3 + ax + b$ над $GF(q)$, то межа Хассе:

$$q - 2\sqrt{q} + 1 \le n \le q + 2\sqrt{q} + 1. \tag{4.1}$$

З цього випливає, що порядок еліптичної кривої над $GF(q)$ дорівнює приблизно *q*.

⚠ **Означення 4.5.** Число *u* називають *майже простим* (*nearly prime*), якщо $u = kr$ для деякого простого *r* з інтервалу $r_{\min} \le r \le r_{\max}$ і деякого гладкого цілого числа *k*.

За умови пробного ділення $l_{\max}$ ціле додатне число *k* називають *гладким*, якщо кожен простий дільник *k* не перевищує $l_{\max}$.

Оскільки всі еліптичні криві над $GF(q)$ мають порядок не більше ніж $u_{\max} = q + 2\sqrt{q} + 1$, то $r_{\max}$ не має бути більшим за $u_{\max}$. Часто покладають $r_{\max} = u_{\max}$, коли немає певного бажаного максимуму. Крім того, якщо $r_{\min}$ близьке до $u_{\max}$, то буде невелика кількість можливих кривих для вибору, а отже, буде складніше знайти потрібну криву. Якщо потрібна крива простого порядку, зручним вибором є $r_{\min} = q + \sqrt{q}$.

Нехай $K = \lfloor u / r_{\min} \rfloor$.

Якщо *K* = 1, то *u* майже просте тоді і тільки тоді, коли воно просте.

Якщо $K \ge 2$, то наступний алгоритм перевіряє *u* на майже простоту.

Зауважте, що завжди можна взяти $l_{\max} \le K$.

Алгоритм 4.10. Пошук випадкової точки на еліптичній кривій

***Вхід*:** просте число $p > 3$; параметри *a, b* еліптичної кривої *E* за модулем *p*.
***Вихід*:** випадково згенерована точка (крім *O*) на *E*.

***Процес*:**

1. Вибрати випадкове число *x* з інтервалу $0 \le x < p$.
2. Обчислити $\alpha = (x^3 + ax + b) \bmod p$.
3. Якщо α = 0, то вивести (*x*, 0) і завершити.
4. Знайти квадратний корінь за модулем *p* з α або визначити, що не існує жодного кореня.

   Коментар: Застосувати алгоритм 4.7
5. Якщо результат кроку 4 вказує на те, що квадратних коренів не існує,

перейти до кроку 1; інакше прейти до кроку 4 з цілим числом β таким, що $\beta^2 \equiv a \bmod p$.

6. Згенерувати випадковий біт μ і покласти $y = (-1)^{\mu}\beta$.
7. Вивести (x, y).

Алгоритм 4.11. Знаходження точки великого простого порядку

***Вхід*:** просте число r; додатне ціле число k, яке не ділиться на r; еліптична крива E над полем $GF(q)$

***Вихід*:** якщо $\#E(GF(q)) = kr$, то точка G на E порядку r;

якщо $\#E(GF(q)) \neq kr$, то повідомлення «неправильний порядок».

***Процес*:**

1. Згенерувати випадкову точку P (не O) на E.

Коментар: Застосувати алгоритм 4.10.

2. Обчислити $G = kP$.
3. Якщо $G = O$, то перейти до кроку 1.
4. Обчислити $Q = rG$.
5. Якщо $Q \neq O$, то вивести «неправильний порядок» і завершити.
6. Вивести G.

Одним із важливих питань є обчислення скорочених поліномів класу. Нехай

$$F(z) = 1 + \sum_{j=1}^{\infty}(-1)^j (z^{(3j^2-j)/2} + z^{(3j^2+j)/2}) = 1 - z - z^2 + z^5 + z^7 - z^{12} - z^{15} + \ldots$$

і $\theta = \exp\left(\dfrac{-\sqrt{D} + Bi}{A}\pi\right)$.

Нехай

$$\mathbf{f}_0(A,B,C) = \theta^{-1/24} F(-\theta) / F(\theta^2);$$

$$\mathbf{f}_1(A,B,C) = \theta^{-1/24} F(\theta) / F(\theta^2);$$

$$\mathbf{f}_2(A,B,C) = \sqrt{2}\theta^{-1/12} F(\theta^4) / F(\theta^2).$$

Оскільки

$$|\theta| < e^{-\pi\sqrt{3}/2} \approx 0{,}0658287,$$

то ряд F (z), який використовується для обчислення чисел $\mathbf{f}_J(A,B,C)$, збігається так само швидко, як і степеневий ряд для $e^{-\pi\sqrt{3}/2}$.

Якщо [A, B, C] є матрицею детермінанта D, то її інваріант класу є

$$\mathbf{C}(A,B,C) = (N\lambda^{-BL} 2^{-I/6} (\mathbf{f}_J(A,B,C))^K)^G,$$

де $G = \text{НСД}(D,3)$;

$$I = \begin{cases} 3 & \text{якщо } D \equiv 1,2,6,7 \bmod 8; \\ 0 & \text{якщо } D \equiv 3 \bmod 8 \text{ і } D \neq 0 \bmod 3; \\ 2 & \text{якщо } D \equiv 3 \bmod 8 \text{ і } D \equiv 0 \bmod 3; \\ 6 & \text{якщо } D \equiv 5 \bmod 8. \end{cases}$$

$$J = \begin{cases} 0 & \text{для } AC \text{ непарного}; \\ 1 & \text{для } C \text{ парного}; \\ 2 & \text{для } A \text{ парного}. \end{cases}$$

$$K = \begin{cases} 2 & \text{якщо } D \equiv 1,2,6 \bmod 8; \\ 1 & \text{якщо } D \equiv 3,7 \bmod 8; \\ 4 & \text{якщо } D \equiv 5 \bmod 8. \end{cases}$$

$$L = \begin{cases} A - C + A^2C & \text{якщо } C \text{ непарне або } D \equiv 5 \bmod 8 \text{ і } C \text{ парне}; \\ A + 2C - AC^2 & \text{якщо } D \equiv 1,2,3,6,7 \bmod 8 \text{ і } C \text{ парне}; \\ A - C + 5AC^2 & \text{якщо } D \equiv 3 \bmod 8 \text{ і } A \text{ парне}; \\ A - C - AC^2 & \text{якщо } D \equiv 1,2,5,6,7 \bmod 8 \text{ і } A \text{ парне}. \end{cases}$$

$$M = \begin{cases} (-1)^{(A^2-1)/8} & \text{якщо } A \text{ непарне}; \\ (-1)^{(C^2-1)/8} & \text{якщо } A \text{ парне}. \end{cases}$$

$$N = \begin{cases} 1 & \text{якщо } D \equiv 5 \bmod 8 \text{ або } D \equiv 3 \bmod 8 \text{ і } AC \text{ непарне}, \\ & \text{або } D \equiv 7 \bmod 8 \text{ і } AC \text{ парне}; \\ M & \text{якщо } D \equiv 1,2,6 \bmod 8 \text{ або } D \equiv 7 \bmod 8 \text{ і } AC \text{ парне}; \\ -M & \text{якщо } D \equiv 3 \bmod 8 \text{ і } AC \text{ парне}. \end{cases}$$

Якщо $[A_1, B_1, C_1], \ldots, [A_h, B_h, C_h]$ є скороченими симетричними матрицями додатного безквадратного визначника D, то скорочений поліном класу для D є:

$$w_D(t) = \prod_{j=I}^{h} (t - \mathbf{C}(A_j, B_j, C_j)).$$

Скорочений поліном класу має цілі коефіцієнти, тому обчислення за наведеними вище формулами необхідно виконувати з достатньою точністю для визначення кожного коефіцієнта полінома $w_D(t)$. Це означає, що помилка, яка виникає під час обчислення кожного коефіцієнта має бути менше 0,5.

Нарешті, розглянемо особливості комплексного множення.

Якщо E – несуперсингулярна еліптична крива над $GF(q)$ порядку u, то число

$$Z = 4q - (q + 1 - u)^2$$

є додатним з межею Хассе (4.1), і існує унікальна факторизація

$$Z = DV^2,$$

де D є безквадратним числом (тобто не містить множників, що є квадратами чисел). Таким чином, для кожної несуперсингулярної еліптичної кривої над $GF(q)$ порядку u, існує єдине безквадратне ціле число D таке, що:

$$4q = W^2 + DV^2 \tag{4.2}$$

$$u = q + 1 \pm W \tag{4.3}$$

для деяких W і V.

Кажуть, що E «володіє» комплексним множенням на D (або, точніше, на $\sqrt{-D}$). Водночас D називають *дискримінантом* комплексного множення для q.

Якщо відоме D для заданої кривої E, то можна обчислити її порядок за допомогою (4.2) і (4.3).

Існує можливість отримати криві, порядки яких u задовольняють (4.2) і (4.3) для малих значень D. Майже простих чисел досить багато, щоб можна було знайти криві майже простого порядку з достатньо малим значенням D для їх побудови.

Метод комплексного множення над $GF(p)$ також називають методом Аткіна-Морейна. Цей метод передбачає два основних кроки: пошук відповідного порядку та побудова кривої з таким порядком. Напочатку здійснюється вибір розміру поля q, мінімального порядку точок $r_{\min}$ і пробного ділення з межею $l_{\max}$. З урахуванням цих величин D називають *відповідним*, якщо існує еліптична крива над $GF(q)$ з комплексним множенням на D, і яка має майже простий порядок.

Крок 1. Пошук майже простого порядку.

Знайти відповідне D. Коли воно знайдено, записати D, велике просте число r і ціле число k такі, що $u = kr$ – майже простий порядок кривої.

Крок 2. Побудова кривої та точки.

Використовуючи D, k і r побудувати еліптичну криву над $GF(q)$ і точку порядку r.

Пошук майже простого порядку над $GF(p)$ здійснюється таким чином.

Безквадратне число D може бути дискримінантом комплексного множення для p, лише якщо воно задовольняє такі конгруентні умови.

Нехай $K = \left\lfloor \dfrac{(\sqrt{p} + 1)^2}{r_{\min}} \right\rfloor$.

Якщо $p \equiv 3 \bmod 8$, то $D \equiv 2, 3$ або $7 \bmod 8$.

Якщо $p \equiv 5 \bmod 8$, то D непарне.

Якщо $p \equiv 7 \bmod 8$, то $D \equiv 3, 6$ або $7 \bmod 8$.

Якщо $K = 1$, то $D \equiv 3 \bmod 8$.

Якщо $K = 2$ або 3, то $D \not\equiv 7 \bmod 8$.

З урахуванням цих умов можливі такі безквадратні значення D.

Якщо $K = 1$, то

D = 3, 11, 19, 35, 43, 51, 59, 67, 83, 91, 107, 115, ….

Якщо $p \equiv 1 \bmod 8$ і $K = 2$ або 3, то

D = 1, 2, 3, 5, 6, 10, 11, 13, 14, 17, 19, 21, ….

Якщо $p \equiv 1 \bmod 8$ і $K \geq 4$, то

D = 1, 2, 3, 5, 6, 7, 10, 11, 13, 14, 15, 17, ….

Якщо $p \equiv 3 \bmod 8$ і $K = 2$ або 3, то

D = 2, 3, 10, 11, 19, 26, 34, 35, 42, 43, 51, 58, ….

Якщо $p \equiv 3 \bmod 8$ і $K \geq 4$, то

D = 2, 3, 7, 10, 11, 15, 19, 23, 26, 31, 34, 35, ….

Якщо $p \equiv 5 \bmod 8$ і $K = 2$ або 3, то

D = 1, 3, 5, 11, 13, 17, 19, 21, 29, 33, 35, 37, ….

Якщо $p \equiv 5 \bmod 8$ і $K \geq 4$, то

D = 1, 3, 5, 7, 11, 13, 15, 17, 19, 21, 23, 29, ….

Якщо $p \equiv 7 \bmod 8$ і $K = 2$ або 3, то

D = 3, 6, 11, 14, 19, 22, 30, 35, 38, 43, 46, 51, ….

Якщо $p \equiv 7 \bmod 8$ і $K \geq 4$, то

D = 3, 6, 7, 11, 14, 15, 19, 22, 23, 30, 31, 35, ….

Алгоритм 4.12. Тестування дискримінантів D

***Вхід*:** просте число p і безквадратне число D, що задовольняє конгруентні умови.

***Вихід*:** якщо D є дискримінантом, то ціле число W таке, що $4p = W^2 + DV^2$ для деякого V. (У випадках $D = 1$ або 3 також число V); якщо ні, то повідомлення «недискримінант».

***Процес*:**

1. Знайти $\sqrt{-D} \bmod p$ або визначити, що не існує жодного значення.

   Коментар: Застосувати алгоритм 4.7.

2. Якщо результат кроку 1 вказує на відсутність квадратних коренів, то повернути «недискримінант» і завершити.

В іншому випадку результатом кроку 1 є ціле число B за модулем p.

3. Покласти $A = p$ і $C = (B^2 + D)/p$.

4. Покласти $\mathbf{S} = \begin{vmatrix} A & B \\ B & C \end{vmatrix}$ і $\mathbf{U} = \begin{vmatrix} 1 \\ 0 \end{vmatrix}$.

5. Поки $|2B| \le A \le C$, виконувати:

5.1. Покласти $\delta = \left\lfloor \frac{B}{C} + \frac{1}{2} \right\rfloor$.

5.2. Покласти $\mathbf{T} = \begin{vmatrix} 0 & -1 \\ 1 & \delta \end{vmatrix}$.

5.3. Обчислити $\mathbf{U} = \mathbf{T}^{-1}\mathbf{U}$.

5.4. Обчислити $\mathbf{S} = \mathbf{T}^t\mathbf{S}\mathbf{T}$.

Коментар: $\mathbf{T}^t$ позначає транспонування $\mathbf{T}$.

6. Якщо $D = 11$ і $A = 3$, то покласти $\delta = 0$ і повторити кроки 5.2, 5.3 і 5.4.

7. Якщо $D = 1$ або 3, то повернути $W = 2X$ і $V = 2Y$ і завершити.

Коментар: X і Y - елементи $\mathbf{U}$, тобто $\mathbf{U} = \begin{vmatrix} X \\ Y \end{vmatrix}$.

8. Якщо $A = 1$, то повернути $W = 2X$ і завершити.

9. Якщо $A = 4$, то повернути $W = 4X + BY$ і завершити.

10. Повернути «недискримінант».

Алгоритм 4.13. Пошук майже простого порядку

***Вхід*:** просте число p; межа пробного ділення $l_{\max}$; межі $r_{\min}$ і $r_{\max}$ для порядку базової точки.

***Вихід*:** безквадратне ціле число D; просте число r в інтервалі $r_{\min} \le r \le r_{\max}$; гладке ціле k таке, що $u = kr$ є порядком еліптичної кривої за модулем p із комплексним множенням на D.

***Процес*:**

1. Вибрати безквадратне число D, що задовольняє конгруентні умови.

2. Обчислити символ Якобі $J = \left(\frac{-D}{p}\right)$. Якщо $J = -1$, то перейти до кроку 1.

Коментар: Застосувати алгоритм 4.20.

3. Скласти список непарних простих чисел l, що ділять D.

4. Для кожного l обчислити символ Якобі $J = \left(\frac{p}{l}\right)$. Якщо $J = -1$ для деякого l, то перейти до кроку 1.

5. Перевірити, чи D є дискримінантом. Якщо результат «недискримінант», то перейти до кроку 1. В іншому випадку результатом є ціле число W і разом із V, якщо $D = 1$ або 3.
6. Скласти список можливих порядків таким чином:
   - якщо $D = 1$, то порядки $p + 1 \pm W$ і $p + 1 \pm V$;
   - якщо $D = 3$, то порядки $p + 1 \pm W$, $p + 1 \pm (W + 3V)/2$ і $p + 1 \pm (W - 3V)/2$;
   - в іншому випадку порядки $p + 1 \pm W$.
7. Перевірити кожний порядок на майже простоту. Якщо будь-який порядок майже простий, то повернути (D, k, r) і завершити.

   Коментар: Застосувати алгоритм 4.16.
8. Перейти до кроку 1.

Еліптичну криву $y^2 \equiv (x^3 + a_0 x + b_0) \bmod p$ з комплексним множенням на D можна записати відразу використовуючи коефіцієнти, наведені в табл. 4.1.

Таблиця 4.1 – Коефіцієнти еліптичної кривої E для певних значень D

D	a_0	b_0	D	a_0	b_0
1	1	0	19	-152	722
2	-30	56	43	-3440	77658
3	0	1	67	-29480	1948226
7	-35	98	163	-8697680	9873093538
11	-264	1694			

Для інших значень D стандарт IEEE Std 1363-2000 рекомендує використовувати такий алгоритм.

Алгоритм 4.14. Побудова кривої із заданим комплексним множенням

***Вхід*:** просте число p і дискримінант $D > 3$ для p.

***Вихід*:** a_0 і b_0 такі, що еліптична крива $y^2 \equiv (x^3 + a_0 x + b_0) \bmod p$ має комплексне множення на D.

***Процес*:**
1. Обчислити $w(t) = w_D(t) \bmod p$.

   Коментар: Застосувати формули для визначення скороченого поліному класу.
2. Виконати алгоритм 4.12 і отримати значення W.
3. Якщо W є парним числом, то:

   3.1. Обчислити лінійний коефіцієнт $t - s$ виразу $w_D(t) \bmod p$ з $d = 1$.

   Коментар: Застосувати алгоритм 4.9.

   3.2 Обчислити $V = (-1)^D 2^{4IK} s^{24/(GK)} \bmod p$, де G, I і K такі, як у разі визначення скороченого поліному класу.

3.3 Обчислити:

$$a_0 = -3(V+64)(V+16) \bmod p;$$

$$b_0 = 2(V+64)^2(V-8) \bmod p.$$

4. Якщо *W* непарне, то:

4.1. Обчислити кубічний множник $g(t)$ виразу $w_D(t) \bmod p$ з $d = 3$.

Коментар: Застосувати алгоритм 4.9.

4.2. Виконати обчислення, в яких коефіцієнти поліномів є цілими числами за модулем *p*:

$$V(t) = \begin{cases} -t^{24} \bmod g(t) & \text{якщо 3 не ділить } D; \\ -256t^8 \bmod g(t) & \text{якщо 3 ділить } D. \end{cases}$$

$$a_1(t) = -3(V(t)+64)(V(t)+256) \bmod g(t);$$

$$b_1(t) = 2(V(t)+64)^2(V(t)-512) \bmod g(t);$$

$$a_3(t) = a_1(t)^3 \bmod g(t);$$

$$b_2(t) = b_1(t)^2 \bmod g(t).$$

4.3 Обчислити $a_0 = \sigma\tau \bmod p$ і $b_0 = \sigma\tau^2 \bmod p$, де σ – ненульовий коефіцієнт з $a_3(t)$, а τ – відповідний коефіцієнт з $b_2(t)$.

5. Повернути (a_0, b_0) і завершити.

Алгоритм 4.15. Вибір кривої та точки

***Вхід*:** параметри еліптичної кривої *p*, *k* і *r*, а також коефіцієнти a_0 і b_0, отримані за допомогою алгоритму 4.14.

***Вихід*:** еліптична крива *E* за модулем *p* і точка *G* на *E* порядку *r* або повідомлення «неправильний порядок».

***Процес*:**

1. Вибрати ціле число ξ таке, що $0 < \xi < p$.
2. Якщо $D = 1$, то покласти $a = a_0\xi \bmod p$ і $b = 0$; якщо $D = 3$, то покласти $a = 0$ і $b = b_0\xi \bmod p$. В іншому випадку покласти $a = a_0\xi^2 \bmod p$ і $b = b_0\xi^3 \bmod p$.
3. Знайти точку *G* порядку *r* на кривій $y^2 \equiv (x^3 + ax + b) \bmod p$.

Коментар: Застосувати алгоритм 4.11.

4. Якщо результатом виконання алгоритму 4.11 є «неправильний порядок», тоді повернути повідомлення «неправильний порядок» і завершити.
5. Повернути коефіцієнти *a*, *b* і точку *G*.

Спосіб вибору ξ на першому кроці цього алгоритму залежить від типу бажаних коефіцієнтів.

Якщо $D \neq 1$ або 3, і бажано, наприклад, щоб $a = -3$, то можна взяти

значення ξ, що є розв'язком порівняння $a_0\xi^2 \equiv -3 \bmod p$, якщо він існує.

Якщо розв'язок не існує, або якщо таке значення ξ призводить до повідомлення «неправильний порядок», то потрібно вибрати іншу криву.

Якщо $p \equiv 3 \bmod 4$ і результат був «неправильний порядок», то вибрати $p - \xi$ замість ξ; що веде до кривої з $a = -3$ і правильним порядком.

Якщо розв'язку ξ не існує, або якщо $p \equiv 1 \bmod 4$, то повторити алгоритм 4.14 з іншим коренем скороченого поліному.

Якщо немає обмежень на коефіцієнти, то значення ξ вибирається навмання.

Якщо результатом є повідомлення «неправильний порядок», то повторювати алгоритм, поки не буде отримано набір параметрів *a*, *b*, *G*.

Алгоритм 4.16. Тестування на майже простоту

***Вхід*:** цілі додатні числа u, $l_{\max}$, $r_{\min}$ і $r_{\max}$.

***Вихід*:** якщо u майже просте, то (r, k); якщо u не є майже простим, то повідомлення «не майже просте».

***Процес*:**

1. Покласти $r = u$, $k = 1$.
2. Для l від 2 до $l_{\max}$ виконувати:
   2.1. Якщо l є складеним, то перейти до кроку 2.3.
   2.2. Поки l ділить r виконувати:
      2.2.1. Покласти $r = r / l$ і $k = k\, l$.
      2.2.2. Якщо $r < r_{\min}$, то повернути «не майже просте» і завершити.
   2.3. Покласти $l = l + 1$.
3. Якщо $r > r_{\max}$, повернути «не майже просте» і завершити.
4. Перевірити r на простоту за допомогою алгоритму Міллера-Рабіна.
5. Якщо r просте, то повернути (r, k) і завершити.
6. Повернути «не майже просте».

4.2.4 Сертифікат Голдвассера-Кіліана-Аткіна-Морейна

Результатом спільної реалізації алгоритму Голдвассера–Кіліана і алгоритму Аткіна-Морейна є сертифікат простоти Голдвассера-Кіліана-Аткіна-Морейна (GKAM).

Стандарт IEEE Std 1363-2000 рекомендує алгоритм доведення простоти GKAM, що створює сертифікат простоти $\mathbf{C} = \{C_1, \ldots, C_s\}$, в якому кожен компонент C_i складається з додатних цілих чисел:

$$C_i = (p_i, r_i, a_i, b_i, x_i, y_i),$$

де для всіх i (x_i, y_i) – точка порядку r_i на еліптичній кривій

$$y^2 \equiv (x^3 + a_i x + b_i) \bmod p_i;$$

$\sqrt{r_i} > \sqrt[4]{p_i} + 1 -$ для всіх i;

$p_1 = n$;

$p_{i+1} = r_i$ – для $1 \le i \le s$;

$r_s < l_{\max}^2$;

r_s є простим числом, що підтверджено пробним діленням.

Алгоритм 4.17. Доведення простоти GKAM

***Вхід*:** велике непарне додатне ціле число n, яке перевірено на простоту алгоритмом Міллера-Рабіна і пробним діленням з межею $l_{\max}$.

***Вихід*:** сертифікат простоти для n або повідомлення про помилку.

***Процес*:**

1. Покласти **C** = {}.
2. Покласти $i = 0$.
3. Покласти $r = n$.
4. Поки $r_s > l_{\max}^2$ виконувати:
   4.1. Покласти $i = i + 1$.
   4.2. Покласти $p = r$.
   4.3. Покласти $r_{\min} = \left(\sqrt[4]{p_i} + 1\right)^2$:
   4.4. Знайти цілі числа D, k, r, такі що:
   - $r \ge r_{\min}$;
   - r перевірене на простоту алгоритмом Міллера-Рабіна;
   - kr є порядком еліптичної кривої над $GF(p)$ з комплексним множенням згідно з D.

   Коментар: Застосувати алгоритм 4.13.

   4.5. Покласти $C_i = (p, r, D, k)$.
   4.6. Додати C_i до **C**.
5. Покласти $s = i$.
6. Підтвердити простоту r шляхом пробного ділення. (Якщо буде виявлено, що r складене, то повернути повідомлення «помилка» та завершити).
7. Для i від 1 до s виконувати:
   7.1. Покласти $(p, r, D, k) = C_i$
   7.2. Знайти еліптичну криву $E : y^2 \equiv (x^3 + ax + b) \bmod p$ над $GF(p)$ і точку (x, y) на E порядку r.

   Коментар: Застосувати алгоритм 4.15.

   7.3. Покласти $C_i = (p, r, a, b, x, y)$.
8. Повернути **C** та завершити.

Важливо пам'ятати, що сертифікат Пратта рекомендують генерувати для невеликих чисел, а сертифікат Голдвассера-Кіліана-Аткіна-Морейна – для чисел, більших за 10^{10}.

Зауважте також, що сертифікати простоти не потрібні, коли просте число генерується та зберігається в безпечному середовищі, керованому стороною, яка згенерувала це число.

4.3 Імовірнісні тести простоти

Найпопулярнішими тестами простоти є ймовірнісні тести. Вони використовують, крім тестованого числа *n*, деякі інші числа *a*, які випадково вибирають з певного набору чисел. Базова структура цих тестів є такою:

1. Випадково вибрати число *a*.
2. Перевірити певний критерій, що містить *a* та задане число *n*. Якщо критерій не виконується, то число *n* є складеним, і тест завершується.
3. Виконувати кроки 1 і 2 поки не буде досягнуто потрібної впевненості в результаті тестування.

Якщо після певної кількості перевірок не отримано відповідь, що *n* є складене число, то його можна оголосити імовірно простим числом.

Імовірнісні тести простоти базуються на таких положеннях. Для кожного непарного цілого числа *n* набір $W(n) \subset \mathbf{Z}_n$ визначається таким чином, що виконуються властивості:

- для заданого $a \in \mathbf{Z}_n$ можна перевірити за поліноміальний час, чи $a \in W(n)$;
- якщо *n* просте, то $W(n) = \varnothing$ (порожній набір);
- якщо *n* складене, то $\#W(n) \geq \frac{n}{2}$.

⚠ **Означення 4.6.** Якщо непарне ціле число *n* є складеним, то елементи $W(n)$ називають *свідками складеності* (*witnesses to the compositeness*) числа *n*, а елементи додаткової множини $L(n) = \mathbf{Z}_n - W(n)$ називають *брехунами* (*liars*).

Імовірнісний тест простоти використовує ці властивості множини $W(n)$ у такий спосіб.

Припустимо, що *n* є цілим числом, простоту якого потрібно визначити. Ціле число $a \in \mathbf{Z}_n$ вибирається навмання, і перевіряється, чи $a \in W(n)$. Тест видає *composite* (складене) якщо $a \in W(n)$, і виводить *prime* (просте), якщо $a \notin W(n)$.

Якщо $a \in W(n)$, то кажуть, що *n* не відповідає перевірці простоти для бази *a*; у цьому випадку *n* обов'язково є складеним.

Якщо $a \notin W(n)$, то кажуть, що *n* пройшло перевірку простоти для бази *a*. Однак у цьому випадку немає висновку з абсолютною впевненістю про простоту *n*, і оголошення *prime* може бути неправильним.

Будь-яке одноразове виконання ймовірнісного тесту, яке оголошує *composite*, підтверджує це з упевненістю. З іншого боку, послідовні незалежні запуски тесту, всі з яких повертають відповідь *prime*, лише забезпечують збільшення впевненості, що *n* справді є простим.

Якщо тест виконується *t* разів незалежно для складеного числа *n*, ймовірність того, що *n* оголошується *prime* всі *t* разів (тобто ймовірність помилки) не більше 2^{-t}. З цієї причини такі тести правильніше називати *тестами складеності* (*compositeness tests*), ніж імовірнісними тестами простоти. Чим більше значення *t*, тим менша ймовірність помилки, а отже, вище рівень безпеки. Тому значення *t* називають *параметром безпеки*.

4.3.1 Тест Ферма

Перевірка простоти виконується на основі певних критеріїв простоти, одним із яких є критерій Ферма, що випливає з малої теореми Ферма.

Мала теорема Ферма. *Якщо p - просте число і* $\text{НСД}(a, p) = 1$, *то*

$$a^{p-1} \equiv 1 \bmod p.$$

Ця теорема дає нам потужний тест на складеність числа *n*.

Якщо $a^n - 1$ не ділиться на *n*, то *n* є складеним, і число *a* називають *свідком Ферма* (*Fermat witness*) складеності числа *n*.

Якщо $\text{НСД}(a,n) = 1$ і $a^{n-1} \equiv 1 \bmod n$, то *n* може бути як простим, так і складеним. У цьому випадку число *a* називають *брехуном Ферма* (*Fermat liar*) простоти числа *n*.

Алгоритм 4.18. Тест Ферма

***Вхід*:** ціле непарне число $n \geq 3$ і параметр безпеки $t \geq 1$.

***Вихід*:** *prime* або *composite*.

***Процес*:**

1. Для *i* від 1 до *t* виконати:
   1.1. Вибрати навмання ціле число $2 \leq a \leq n-2$.
   1.2. Обчислити $r = a^{n-1} \bmod n$.
   1.3. Якщо $(r \neq 1)$, то повернути *composite*.
2. Повернути *prime*.

Якщо цей алгоритм повертає *composite*, то ціле непарне число *n*, безумовно, є складеним. Якщо алгоритм повертає *prime*, то не надається доказу того, що *n* справді є простим.

Приклад 4.9. Визначити чи є n =133 простим.

Розв'язання. Випадково оберемо $1 \le a \le 133$, наприклад a =58.

Перевіряємо: $58^{132} \equiv 1 \bmod 133$.

Результат свідчить про те, що або число 133 є простим, або число 58 є брехуном. Тому беремо інше число a, наприклад a =23.

Перевіряємо: $23^{132} \equiv 106 \bmod 133$.

Отже число 133 є складеним і a =58 є брехуном.

Відповідь: число 133 є складеним.

Тест Ферма досить швидкий і простий у виконанні. Однак він має один суттєвий недолік. Твердження про те, що чим більше значення t, тим менша ймовірність помилки, для теста Ферма не зовсім коректне, оскільки існують числа, для яких усі значення a є брехунами.

⚠ **Означення 4.7.** Складене число n, таке що $a^{n-1} \equiv 1 \bmod n$ для кожного цілого числа a, взаємно простого з n, називають *числом Кармайкла* (*Carmichael number*).

З цього означення випливає, що всі a, для яких $\text{НСД}(a,n) = 1$, є брехунами. Через це числа Кармайкла також називають *абсолютно псевдопростими числами Ферма*.

Ось кілька чисел Кармайкла:

561, 1105, 1729, 2465, 2821, 6601, 8911, 10585, 15841, 29341, 41041, 46657,…

(послідовність A002997 в OEIS).

У 1939 році Джон Черник довів теорему, яка забезпечує можливість побудови підмножини чисел Кармайкла. Ця теорема стверджує, що якщо $6m+1$, $12m+1$ і $18m+1$ є простими числами для одного натурального числа m, то їх добуток $M_3(m) = (6m+1)(12m+1)(18m+1)$ є числом Кармайкла. Число m має задовольняти умову: $m \equiv 0 \bmod 5$ або $m \equiv 1 \bmod 5$.

Наприклад, для m=1 маємо такі множники:

$$6m+1 = 6 \cdot 1 + 1 = 7;\ 12m+1 = 12 \cdot 1 + 1 = 13;\ 18m+1 = 18 \cdot 1 + 1 = 19.$$

Числа Кармайкла також можуть утворюватися як добуток $k > 3$ простих чисел:

$$M_3(m) = (6m+1)(12m+1)\prod_{i=1}^{k-2}(9 \cdot 2^i m + 1),$$

за умови $m \equiv 0 \bmod 2^{k-4}$.

Приклади розкладів чисел Кармайкла на прості множники наведено в табл. 4.2.

Таблиця 4.2 – Розклади чисел Кармайкла на прості множники

k	Розклади чисел Кармайкла
3	561=3·11·17
4	41041=7·11·13·41
5	825265=5·7·17·19·73
6	321197185=5·19·23·29·37·137
7	5394826801=7·13·17·23·31·67·73
8	232250619601=7·11·13·17·31·37·73·163
9	9746347772162=7·11·13·17·19·31·37·41·641

Повторні тести Ферма числа Кармайкла не зможуть показати, що воно складене, доки не буде використано один із його простих множників.

Чим більше числа Кармайкла, тим рідше вони зустрічаються. Наприклад, у діапазоні від 1 до 10^{21} міститься 20138200 чисел Кармайкла (приблизно одне на 50 трильйонів чисел). Проте, доведено, що їх кількість нескінченна.

Якщо відомо прості множники числа n, то можна легко визначити, чи є n числом Кармайкла. Складене ціле число n є числом Кармайкла тоді і тільки тоді, коли задовольняються такі дві умови:

– n не має квадратів, тобто n не ділиться на квадрат будь-якого простого числа;

– $p-1$ ділить $n-1$ для кожного простого дільника p числа n.

Недолік тесту Ферма усувається використанням критеріїв, які є сильнішими, ніж критерій Ферма.

4.3.2 Імовірнісний тест простоти Соловея-Штрассена

Імовірнісний тест простоти Соловея-Штрассена (Solovay-Strassen) завжди коректно визначає, що просте число є простим, але для складених чисел з деякою ймовірністю він може дати неправильну відповідь. Основна перевага тесту полягає в тому, що він, на відміну від тесту Ферма, розпізнає числа Кармайкла як складені.

Цей тест базується на критерії Ейлера.

Нехай n – непарне просте число. Тоді

$$a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \bmod n, \text{ де } \left(\frac{a}{n}\right) - \text{символ Якобі,}$$

для всіх цілих чисел a, таких що $\text{НСД}(a,n)=1$.

Означення 4.8. Нехай n – непарне складене ціле число і нехай a – ціле число, $2 \le a \le n-2$.

1. Якщо $\text{НСД}(a,n) > 1$ або $\left(\frac{a}{n}\right) \neq a^{(n-1)/2} \bmod n$, то a називають *свідком Ейлера* (*Euler witness*) складеності числа n.

2. Якщо $\text{НСД}(a,n) = 1$ і $a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \bmod n$, то число a називають *брехуном Ейлера* (*Euler liar*) для n, а число n називають *псевдопростим числом Ейлера* для основи a.

Псевдопрості Ейлера за основою 2 утворюють послідовність:

561, 1105, 1729, 1905, 2047, 2465, 3277, 4033, 4681, 6601, 8321, 8481, …

(послідовність A047713 в OEIS),

а за основою 3 – послідовність:

121, 703, 1729, 1891, 2821, 3281, 7381, 8401, 8911, 10585, 12403, 15457, …

(послідовність A048950 в OEIS).

Алгоритм 4.19. Тест Соловея-Штрассена

***Вхід*:** ціле непарне число $n \ge 3$ і параметр безпеки $t \ge 1$.

***Вихід*:** *prime* або *composite*.

***Процес*:**

1. Для i від 1 до t виконати:
   1.1. Вибрати навмання ціле число a, $2 \le a \le (n-2)$.
   1.2. Обчислити $r = a^{(n-1)/2} \bmod n$.
   1.3. Якщо ($r \neq 1$ і $r \neq n-1$), то повернути *composite*.
   1.4. Обчислити символ Якобі $s = \left(\frac{a}{n}\right)$.
   1.5. Обчислити $b = r \bmod n$.
   1.6. Якщо ($b \neq s$), то повернути *composite*.
2. Повернути *prime*.

Якщо $\text{НСД}(a,n) = d$, то d є дільником $r = a^{(n-1)/2} \bmod n$. Отже, перевірка чи $r \neq 1$ (крок 1.3) усуває необхідність перевірки $\text{НСД}(a,n) \neq 1$.

Якщо цей алгоритм повертає *composite*, то ціле непарне число n, безсумнівно, складене, оскільки прості числа не порушують критерій Ейлера.

Якщо n насправді просте, то алгоритм завжди повертає *prime*. Інакше, якщо n насправді є складеним, то, враховуючи, що на кроці 1.1 основа a вибирається незалежно під час кожної ітерації кроку, ймовірність

помилкового повернення алгоритмом *prime* дорівнює 2^{-t}.

Обчислення символу Якобі $\left(\frac{a}{n}\right)$ здійснюється за таким алгоритмом.

Алгоритм 4.20. Jacobi ( )

***Вхід*:** *a* – будь-яке ціле число, *n* – будь-яке ціле число.

***Вихід*:** *result* – розрахований символ Якобі.

***Процес*:**

1. Покласти $a = a \bmod n$.

Коментар: *a* буде в діапазоні $0 \le a < n$.

2. Якщо (*a* = 1 або *n* = 1), то повернути *result*=1.
3. Якщо (*a* = 0), то повернути *result*=0.
4. Визначити *e* і a_1 так, щоб $a = 2^e a_1$, де a_1 – непарне.
5. Якщо *e* парне, то *s* = 1.
   Інакше, якщо ($n \equiv 1 \bmod 8$ або $n \equiv 7 \bmod 8$), то *s* = 1.
   Інакше, якщо ($n \equiv 3 \bmod 8$ або $n \equiv 5 \bmod 8$), то *s* = -1.
6. Якщо ($n \equiv 3 \bmod 4$ і $a_1 \equiv 3 \bmod 4$), то *s* = - *s*.
7. Обчислити $n_1 = n \bmod a_1$.
8. Повернути ($s \times$ Jacobi (n_1, a_1)).

Коментар: Викликати цей процес рекурсивно.

Приклад 4.10. Обчислити символ Якобі для *a* = 9 і *n* = 59.

Розв'язання.
Процес 1.
Умова (*a* = 1 або *n* = 1) не виконується. Умова (*a* = 0) не виконується.

Визначення *e* і a_1. $9 = 2^0 \times 9$, тому *e*=0 і $a_1 = 9$.

Оскільки *e* парне, тому *s* = 1.

$59 \equiv 3 \bmod 4$ і $9 \equiv 1 \bmod 4$, тобто умова ($n \equiv 3 \bmod 4$ і $a_1 \equiv 3 \bmod 4$) не виконується, а тому значення *s* не змінюється.

Обчислення $n_1 = n \bmod a_1$: $59 \equiv 5 \bmod 9$, $n_1 = 5$.

Обчислити і повернути ($s \times$ Jacobi (n_1, a_1)), тобто ($1 \times$ Jacobi $(5,9)$).

Обчислення символу Якобі для *a* = 9 і *n* = 5.
Процес 2.
Обчислення $a = a \bmod n$: $9 \equiv 4 \bmod 5$, тому *a* = 4.
Умова (*a* = 1 або *n* = 1) не виконується. Умова (*a* = 0) не виконується.

Визначення *e* і a_1. $4 = 2^2 \times 1$, тому *e* = 2 і $a_1 = 1$.

Оскільки *e* парне, тому *s* = 1.

Умова ($n \equiv 3 \bmod 4$ і $a_1 \equiv 3 \bmod 4$), не виконується, а тому значення *s* не змінюється.

Обчислення $n_1 = n \bmod a_1 : 5 \equiv 0 \bmod 1,\ n_1 = 0$.

Обчислити і повернути ($s \times \text{Jacobi}\ (n_1, a_1)$), тобто ($1 \times \text{Jacobi}\ (0,1)$).

Обчислення символу Якобі для *a* = 1 і *n* = 0.

Процес 3.

Оскільки *a* = 1, тому повернути 1.

Отже, Jacobi(0,1)=1, Jacobi(5,9)=1×1=1 і Jacobi(59,9)=1×1=1.

Відповідь: Jacobi (59,9) =1.

Приклад 4.11. Обчислити символ Якобі для *a* = 11 і *n* = 59.

Розв'язання.

Процес 1.

Умова (*a* = 1 або *n* = 1) не виконується. Умова (*a* = 0) не виконується.

Визначення *e* і a_1. $11 = 2^0 \times 11$, тому *e*=0 і $a_1 = 11$.

Оскільки *e* парне, тому *s* = 1.

$59 \equiv 3 \bmod 4$ і $11 \equiv 3 \bmod 4$, тобто виконується умова ($n \equiv 3 \bmod 4$ і $a_1 \equiv 3 \bmod 4$), а тому *s* = -1.

Обчислення $n_1 = n \bmod a_1 : 59 \equiv 4 \bmod 11,\ n_1 = 4$.

Обчислити і повернути ($s \times \text{Jacobi}\ (n_1, a_1)$), тобто ($(-1) \times \text{Jacobi}\ (4,11)$).

Обчислення символу Якобі для *a* = 11 і *n* = 4.

Процес 2.

Обчислення $a = a \bmod n : 11 \equiv 3 \bmod 4$, тому *a* = 3.

Умова (*a* = 1 або *n* = 1) не виконується. Умова (*a* = 0) не виконується.

Визначення *e* і a_1. $3 = 2^0 \times 3$, тому *e* = 0 і $a_1 = 3$.

Оскільки *e* парне, тому *s* = 1.

Умова ($n \equiv 3 \bmod 4$ і $a_1 \equiv 3 \bmod 4$), не виконується, а тому значення *s* не змінюється.

Обчислення $n_1 = n \bmod a_1 : 4 \equiv 1 \bmod 3,\ n_1 = 1$.

Обчислити і повернути ($s \times \text{Jacobi}\ (n_1, a_1)$), тобто ($1 \times \text{Jacobi}\ (1,3)$).

Обчислення символу Якобі для *a* = 3 і *n* = 1.

Процес 3.

Оскільки *n* = 1, тому повернути 1.

Отже, Jacobi(1,3)=1;

Jacobi(4,11)=$1 \times (-1) = -1$;

Jacobi(59,11)=$(-1) \times 1 = -1$.

Відповідь: Jacobi (59,11)=-1.

Приклад 4.12. Перевірити число $n = 59$ на простоту.

Розв'язання.

Нехай параметр безпеки $t = 2$.

$t = 1$.

Виберемо навмання ціле число a, $2 \le a \le (n-2)$. Нехай $a = 9$.

Обчислюємо $r = a^{(n-1)/2} \bmod n$: $r = 9^{29} \bmod 59 = 1$.

Оскільки $r = 1$, то обчислюємо символ Якобі $s = \left(\frac{9}{59}\right)$. З прикладу 4.10 маємо $s = 1$.

Обчислюємо $b = r \bmod n$: $b = 1 \bmod 59 = 1$

Оскільки $b = s$, то число 59 є простим. Переходимо до наступної ітерації.

$t = 2$.

Виберемо навмання ціле число a, $2 \le a \le (n-2)$. Нехай $a = 11$.

Обчислюємо $r = a^{(n-1)/2} \bmod n$: $r = 11^{29} \bmod 59 = 58$.

Оскільки $r = n - 1 = 58$, то обчислюємо символ Якобі $s = \left(\frac{11}{59}\right)$. З прикладу 4.11 маємо $s = -1$.

Обчислюємо $b = r \bmod n$: $b = 58 \bmod 59 = -1$

Оскільки $b = s$, то число 59 є простим.

Це остання ітерація.

Відповідь: 59 – просте число з імовірністю помилки 2^{-2}.

Тест Соловея-Штрассена був першим популярним тестом з моменту появи криптографії з відкритим ключем, зокрема криптосистеми RSA. Більше немає будь-яких причин використовувати цей тест, оскільки доступний більш ефективний тест Міллера-Рабіна. Опис тесту Соловея-Штрассена наведено лише через те, що в багатьох наукових працях продовжують посилатися на цей тест.

4.3.3 Імовірнісний тест простоти Міллера-Рабіна

Цей тест базується на критерії, що випливає з такого твердження.

Твердження 4.2. *Нехай p – непарне просте число таке, що* $p - 1 = 2^k q$, *де* q *– непарне число. Нехай a – будь-яке число, яке не ділиться на p. Тоді справджується одна з умов:*

- $a^q \equiv 1 \bmod p$,
- *одне зі значень* a^q, a^{2q}, a^{4q}, ..., $a^{2^{k-1}q}$ *конгруентне* -1 *за модулем p.*

Випадкове значення *a*, для якого складене число *n* задовольняє критерії простоти Міллера–Рабіна, називають *сильним брехуном* (до простоти) для *n*,

тоді як таке *n* називають *сильним псевдопростим для бази a*.

Одна ітерація тесту Міллера-Рабіна призведе до помилки в оголошенні складеного цілого числа простим з імовірністю менше 4^{-1}, тоді як *t* ітерацій будуть помилятися з імовірністю менше ніж 4^{-t}.

Стандарт NIST.FIPS.186-4 пропонує дві альтернативи для перевірки простоти: або використання кількох ітерацій лише тесту Міллера-Рабіна, або використання ітерованого тесту Міллера-Рабіна з наступною одною ітерацією теста Люка (Lucas test). Кількість ітерацій залежить від алгоритму, міцності безпеки, ймовірності помилки, довжини (у бітах) кандидата простого числа та типу випробувань, які необхідно виконати.

У таблицях 4.3, 4.4 і 4.5 наведено мінімальну кількість ітерацій випробувань Міллера-Рабіна, які мають бути виконані. Комбінація двох тестів забезпечує додаткову гарантію простоти числа. Для DSA комбінація двох тестів може забезпечити кращу продуктивність. Проте тест Люка не потрібен під час перевірки значень p_1, p_2, q_1 і q_2 на простоту під час генерації простих чисел RSA.

Таблиця 4.3 – Мінімальна кількість ітерацій тесту Міллера-Рабіна для DSA

Параметри	Тільки тест Міллера-Рабіна	Тест Міллера-Рабіна, а потім один тест Люка	Імовірність помилки
p: 1024 біт	40	3	2^{-80}
q: 160 біт		19	
p: 2048 біт	56	3	2^{-112}
q: 224 біт		24	
p: 2048 біт	56	3	2^{-112}
q: 256 біт		27	
p: 3072 біт	64	2	2^{-128}
q: 256 біт		27	

Таблиця 4.4 – Мінімальна кількість ітерацій тесту Міллера-Рабіна під час генерування простих чисел для використання в цифрових підписах RSA

Параметри	Мінімальна кількість ітерацій	Імовірність помилки
p_1, p_2, q_1 і q_2 >100 біт	28	2^{-80}
p і *q*: 512 біт	5	2^{-80}
p_1, p_2, q_1 і q_2 >140 біт	38	2^{-112}
p і *q*: 1024 біт	5	2^{-112}
p_1, p_2, q_1 і q_2 >170 біт	41	2^{-128}
p і *q*: 1536 біт	4	2^{-128}

Таблиця 4.5 – Мінімальна кількість ітерацій тесту Міллера-Рабіна під час генерування простих чисел для використання в цифрових підписах RSA з імовірністю помилки 2^{-100}

Параметри	Мінімальна кількість ітерацій
p_1, p_2, q_1 і q_2 >100 біт	38
p і q: 512 біт	7
p_1, p_2, q_1 і q_2 >140 біт	32
p і q: 1024 біт	4
p_1, p_2, q_1 і q_2 >170 біт	27
p і q: 1536 біт	3

Стандарт NIST.FIPS.186-4 рекомендує два алгоритми, що реалізують імовірнісний тест простоти Міллера-Рабіна. Ці алгоритми зокрема передбачають використання затверджених генераторів випадкових бітів (Random Bit Generator – RBG).

Алгоритм 4.21. Перевірка ймовірнісної простоти Міллера-Рабіна

***Вхід*:** *w* – непарне ціле число, яке потрібно перевірити на простоту;

iterations – кількість ітерацій тесту (значення має відповідати таблицям 4.3, 4.4 або 4.5).

***Вихід*:** *status* – статус, який повертається алгоритмом, де *status* – PROBABLY PRIME (ймовірно просте) або COMPOSITE (складене).

Процес:

1. Нехай *a* – велике ціле число таке, що 2^a ділить $w-1$.
2. Обчислити $m=(w-1)/2^a$
3. Покласти *wlen* = **len** (*w*).
4. Для *i* від 1 до *iterations* виконати:
4.1. Отримати з використанням RBG рядок *b* довжиною *wlen* бітів.

Коментар: Переконатися, що $1<b<(w-1)$.

4.2. Якщо ($b\le 1$ або $b\ge(w-1)$), то перейти до кроку 4.1.

4.3. Обчислити $z=b^m \bmod w$.

4.4. Якщо ($z=1$ або $z=w-1$), то перейти до кроку 4.7.

4.5. Для *j* від 1 до *a* – 1 виконати:

4.5.1. Обчислити $z=z^2 \bmod w$.

4.5.2. Якщо ($z=w-1$), то перейти до кроку 4.7.

4.5.3. Якщо ($z=1$), то перейти до кроку 4.6.

4.6. Повернути COMPOSITE.

4.7. Збільшити *i* для циклу на кроці 4.

5. Повернути PROBABLY PRIME.

Наступний алгоритм надає додаткову інформацію у разі виявлення помилки, яка може бути корисною під час створення або перевірки модулів RSA.

Алгоритм 4.22. Покращена перевірка простоти Міллера-Рабіна

***Вхід*:** *w* – непарне ціле число, яке потрібно перевірити на простоту. Це буде або *p*, або *q*, або одне з допоміжних простих чисел p_1, p_2, q_1 чи q_2.

iterations – кількість ітерацій тесту (значення має відповідати таблицям 4.3, 4.4 або 4.5).

***Вихід*:** *status* – статус, який повертається алгоритмом, де *status*:

- PROBABLY PRIME (ймовірно просте),
- PROVABLY COMPOSITE WITH FACTOR (доведено складене з множником; повертається з множником),
- PROVABLY COMPOSITE AND NOT A POWER OF A PRIME (доведено складене, але не степень простого числа).

Процес:

1. Нехай *a* – велике ціле число таке, що 2^a ділить *w*-1.
2. Обчислити $m = (w-1)/2^a$
3. Покласти *wlen* = **len** (*w*).
4. Для *i* від 1 до *iterations* виконати:
   4.1. Отримати з використанням RBG рядок *b* довжиною *wlen* бітів.
   Коментар: Переконатися, що $1 < b < (w-1)$.
   4.2. Якщо ($b \le 1$ або $b \ge (w-1)$), то перейти до кроку 4.1.
   4.3. Обчислити *g* = НСД (*b*, *w*).
   4.4. Якщо ($g > 1$), то повернути PROVABLY COMPOSITE WITH FACTOR і значення *g*.
   4.5. Обчислити $z = b^m \bmod w$.
   4.6. Якщо ($z = 1$ або $z = w-1$), то перейти до кроку 4.15.
   4.7. Для *j* від 1 до *a* – 1 виконати:
      4.7.1. Покласти *x* = *z*.
      Коментар: $x \ne 1$ і $x \ne w-1$.
      4.7.2. Обчислити $z = x^2 \bmod w$.
      4.7.3. Якщо ($z = w-1$), то перейти до кроку 4.15.
      4.7.4. Якщо ($z = 1$), то перейти до кроку 4.12.
   4.8. Покласти *x* = *z*.
   Коментар: $x = b^{(w-1)/2} \bmod w$ і $x \ne w-1$.
   4.9. Обчислити $z = x^2 \bmod w$.
   4.10. Якщо ($z = 1$), то перейти до кроку 4.12.
   4.11. Покласти *x* = *z*.
   Коментар: $x = b^{(w-1)} \bmod w$ і $x \ne 1$.
   4.12. Обчислити *g* = НСД (*x*-1, *w*).
   4.13. Якщо ($g > 1$), то повернути PROVABLY COMPOSITE WITH FACTOR і значення *g*.

4.14. Повернути PROVABLY COMPOSITE AND NOT A POWER OF A PRIME.

4.15. Збільшити *i* для циклу на кроці 4.

5. Повернути PROBABLY PRIME.

4.3.4 Імовірнісний тест простоти Люка

Цей тест базується на використанні чисел послідовності Люка, для якої символ Якобі $\left(\frac{D}{n}\right) = -1$. Якщо число *n*, що перевіряється на простоту, не є сильним ймовірно простим числом Люка, то *n* є складеним. В іншому випадку *n* майже напевно просте.

Враховуючи значення *n*, одним з методів вибору *D* є використання методу проб і помилок для знаходження першого *D* у послідовності 5, −7, 9, −11, ... такого, що $\left(\frac{D}{n}\right) = -1$. Якщо *D* і *n* мають спільний простий множник, то $\left(\frac{D}{n}\right) = 0$. Така послідовність значень *D* забезпечує те, що середня кількість значень *D*, які потрібно перевірити, перш ніж потрапити на одне, символ Якобі якого дорівнює −1, становить приблизно 1,79. Коли отримано потрібне значення *D*, встановлюються $P = 1$ і $Q = (1 - D)/4$. Варто відзначити, що пошук потрібного значення *D* не буде вдалим, якщо *n* є повним квадратом. Тому, перш ніж почати перевірку ймовірно простого числа, зазвичай перевіряють чи не є число *n* повним квадратом.

Для значень *D*, *P* і *Q* існують рекурентні співвідношення, які дозволяють обчислити U_{n+1} і V_{n+1} за $O(\log_2 n)$. Щоб почати, покладають $U_1 = 1$ і $V_1 = P = 1$.

Є можливість подвоїти індекс від *k* до 2*k* за один крок, використовуючи рекурентні співвідношення:

$$U_{2k} = U_k \cdot V_k;\ V_{2k} = V_k^2 - Q^k = \frac{V_k^2 + DU_k^2}{2}.$$

Також є можливість збільшити індекс на 1 за допомогою обчислень:

$$U_{2k+1} = (U_{2k} + V_{2k})/2;\ V_{2k+1} = (DU_{2k} + V_{2k})/2.$$

Для визначення елементів послідовності U_k, які потрібно обчислювати, використовуються біти двійкового коду числа *n*+1. Наприклад, якщо *n*+1=38 (у двійковій системі числення – 100110), то, беручи біти по одному зліва направо, отримаємо послідовність індексів для обчислення: $1_2 = 1$, $10_2 = 2$, $100_2 = 4$, $1000_2 = 8$, $1001_2 = 9$, $10010_2 = 18$, $10011_2 = 19$, $100110_2 = 38$. Отже, потрібно обчислити елементи U_1, U_2, U_4, U_8, U_9, U_{18}, U_{19} і U_{38}. Також обчислюються елементи з однаковими номерами послідовності V разом із

Q^1, Q^2, Q^4, Q^8, Q^9, Q^{18}, Q^{19} і Q^{38}.

Зауважимо, що всі обчислення виконуються за модулем n. Якщо $U_{n+1} \equiv 0 \bmod n$, то число n є ймовірно протим, інакше – складене.

Розглянуті основні складові тесту Люка реалізуються алгоритмом, що пропонується стандартом NIST.FIPS.186-4.

Алгоритм 4.23. Перевірка ймовірнісної простоти Люка

***Вхід*:** C – непарне ціле число-кандидат, яке буде перевірено на простоту.
***Вихід*:** *status* – статус, який повертається алгоритмом, де *status* – PROBABLY PRIME (ймовірно просте), COMPOSITE (складене).
***Процес*:**

1. Перевірити, чи C є повним квадратом (див. Алгоритм 4.24). Якщо так, то повернути COMPOSITE.
2. Знайти перше D в послідовності {5, -7, 9, -11, 13, -15, 17, …}, для якого символ Якобі $\left(\frac{D}{C}\right) = -1$. (див. Алгоритм 4.20). Якщо $\left(\frac{D}{C}\right) = 0$ для будь-якого D в послідовності, то повернути COMPOSITE.
3. Покласти $K = C+1$.
4. Нехай $K_r, K_{r-1}, ..., K_0$ є двійковим кодом числа K з $K_r = 1$.
5. Покласти $U_r = 1$ і $V_r = 1$.
6. Для i від r-1 до 0 виконати:
   6.1. Обчислити $U_{temp} = U_{i+1} V_{i+1} \bmod C$.
   6.2. Обчислити $V_{temp} = \frac{V_{i+1}^2 + D U_{i+1}^2}{2} \bmod C$.
   6.3. Якщо ($K_i = 1$), то виконати:
      6.3.1. Обчислити $U_i = \frac{U_{temp} + V_{temp}}{2} \bmod C$.
      6.3.2. Обчислити $V_i = \frac{V_{temp} + D U_{temp}}{2} \bmod C$.
      Інакше:
      6.3.3. Покласти $U_i = U_{temp}$.
      6.3.4. Покласти $V_i = V_{temp}$.
7. Якщо ($U_0 = 0$), то повернути PROBABLY PRIME. В іншому випадку повернути COMPOSITE.

Кроки 6.2, 6.3.1 і 6.3.2 містять вирази виду $A/2 \bmod C$, де A є цілим числом, а C є непарним цілим числом. Якщо $A/2$ не є цілим числом (тобто A є непарним), тоді $A/2 \bmod C$ можна обчислити як $(A+C)/2 \bmod C$. Крім того, $A/2 \bmod C = A \cdot (C+1)/2 \bmod C$ для будь-якого цілого числа A, незважаючи на те, що A є парним чи непарним.

Щоб визначити, чи є n-розрядне натуральне число C повним

квадратом, стандарт NIST.FIPS.186-4 пропонує такий алгоритм.

Алгоритм 4.24. Перевірка повного квадрата

***Вхід*:** C – ціле число, яке потрібно перевірити.
***Вихід*:** *status* – статус, який повертається алгоритмом, де *status* – PERFECT SQUARE (повний квадрат) або NOT A PERFECT SQUARE (неповний квадрат).
***Процес*:**

1. Вибрати n таке, що $2^n > C \geq 2^{n-1}$.
2. Покласти $m = \lceil n/2 \rceil$.
3. Покласти $i = 0$.
4. Вибрати X_0 таке, що $2^m > X_0 \geq 2^{m-1}$.
5. Поки $(X_i)^2 < 2^m + C$ виконувати:
   5.1. Покласти $i = i + 1$.
   5.2. Обчислити $X_i = ((X_{i-1})^2 + C)/(2X_{i-1})$.
6. Якщо $C = \lfloor X_i \rfloor^2$, то повернути PERFECT SQUARE.
   В іншому випадку повернути NOT A PERFECT SQUARE.

4.4 Псевдопрості числа

⚠ **Означення 4.9.** Ціле число $n > 1$, для якого $a^{n-1} \equiv 1 \bmod n$, називають *ймовірно простим за базою a* (*probable-prime base a* або *a*-PRP).

Існує багато інших визначень ймовірно простих чисел. Найчастіше вживають поняття «*псевдопрості числа*». Така назва випливає з того, що складені числа мають деякі властивості простих чисел.

Для прикладу в табл. 4.6 наведено псевдопрості числа, менші за 500 для основ 2, 3, ..., 10.

Таблиця 4.6 – Псевдопрості числа

База	Псевдопрості числа	Відсоток, %
2	341	0,6
3	91, 121	1,2
4	15, 85, 91, 341, 435, 451	3,8
5	217	0,6
6	35, 185, 217, 301, 481	3,2
7	25, 325	1,2
8	9, 21, 45, 63, 65, 105, 117, 133, 153, 231, 273, 341, 481	8,3
9	91, 121, 205	1,9
10	9, 33, 91, 99, 259, 451, 481	4,4

⚠ **Означення 4.10.** Складене число *n*, що проходить тест Ферма для числа *a* взаємно простого з *n*, називають *псевдопростим числом Ферма*.

Приклади псевдопростих чисел Ферма (ППЧФ) наведено в табл. 4.7.

Таблиця 4.7 – Псевдопрості числа Ферма

a	ППЧФ	*a*	ППЧФ	*a*	ППЧФ
2	341=11·31	60	341=11·31	153	209=11·19
3	91=7·13	63	341=11·31	156	217=7·31
58	133=7·19	102	133=7·19	159	247=13·19

Псевдопрості Ферма за базою 2 утворюють послідовність:

341, 561, 645, 1105, 1387, 1729, 1905, 2047, 2465, 2701, 2821, 3277, 4033, …

(послідовність A001567 в OEIS),
а за базою 3 – послідовність:

91, 121, 286, 671, 703, 949, 1105, 1541, 1729, 1891, 2465, 2665, 2701, 2821, …

(послідовність A005935 в OEIS).

⚠ **Означення 4.11.** Псевдопрості числа Ферма для $a = 2$ називають *числами Пуле*.

Приклади чисел Пуле наведено в табл. 4.8.

Таблиця 4.8 – Числа Пуле

n	Розклад на множники	*n*	Розклад на множники	*n*	Розклад на множники
341	11·31	2701	37·73	6601	7·23·41
561	3·11·17	2821	7·13·31	7957	73·109
1387	19·73	4681	31·151	8321	53·157
2047	23·89	5461	43·127	8481	3·11·257

Для кожної бази *a* існує нескінченна кількість складених чисел, які є *a*-PRP. Однак, PRP доволі часто є простими.

У табл. 4.9 наведено результати, які отримали Кім і Померанс, обчислюючи ймовірність *p*(*x*) того, що PRP число, менше за *x*, є складеним. Використано такі вхідні дані:

– ціле число *n* вибирається навмання з $1 < n \le x$;
– ціле число *a* вибирається навмання з $1 < a < n - 1$;
– $a^{n-1} \equiv 1 \bmod n$ (тобто *n* є PRP за базою *a*).

Аналіз цієї таблиці показує, що зі збільшенням цифр у числі ймовірність отримання складеного числа значно зменшується. З цього можна зробити висновок, що число вважається простим з дуже високою, але не з абсолютною впевненістю.

Таблиця 4.9 – Імовірність того, що випадкове PRP число є складеним

Кількість цифр у числі x	Верхня границя для $p(x)$	Кількість цифр у числі x	Верхня границя для $p(x)$	Кількість цифр у числі x	Верхня границя для $p(x)$
60	$7{,}16\cdot10^{-2}$	150	$1{,}49\cdot10^{-17}$	500	$2{,}3\cdot10^{-55}$
80	$8{,}46\cdot10^{-5}$	200	$3{,}85\cdot10^{-27}$	700	$1{,}8\cdot10^{-82}$
100	$2{,}77\cdot10^{-8}$	300	$5{,}8\cdot10^{-29}$	1000	$1{,}2\cdot10^{-123}$
120	$5{,}28\cdot10^{-12}$	400	$5{,}7\cdot10^{-42}$	2000	$8{,}6\cdot10^{-262}$

Для таких чисел Генрі Коен увів поняття «прості числа індустріального класу», тому що вони досить часто є достатньо придатними для багатьох задач криптографії.

⚠ **Означення 4.12.** Цілі числа, для яких простота не була сертифікована, але вони пройшли ймовірнісні тести на простоту, називають *простими числами індустріального класу* (*industrial-grade primes*).

Прості числа індустріального класу іноді використовуються замість сертифікованих простих чисел у таких алгоритмах, як шифрування RSA, які мають використовувати великі прості числа. Засвідчення простоти великих чисел (наприклад, понад 100 цифр) значно важче, ніж доведення, що вони є простими числами індустріального класу.

> В тестах простоти для дуже великих чисел, вибраних навмання, шанс потрапити на значення, яке обманює тест Ферма менший, ніж шанс того, що космічна радіація спричинить помилку в перебігу коректного алгоритму. Сприймання алгоритму є неадекватним через першу причину, але не через другу, що показує різницю між математикою та інженерією.
>
> Hal Abelson and Gerald J. Sussman
> «Structure and Interpretation
> Of Computer Programs»

? КОНТРОЛЬНІ ПИТАННЯ

1. Чим принципово відрізняються істинні тести простоти від імовірнісних тестів простоти?
2. Якої довжини цілі числа рекомендують перевіряти на простоту використанням пробного ділення?
3. Чи можна розпаралелити процес пробного ділення?
4. Яка причина того, що детермінований алгоритм перевірки простоти AKS є повільнішим за алгоритм пробного ділення?
5. Який тест є ефективним для перевірки простоти чисел Мерсенна?
6. Які вимоги має задовольняти сертифікат про простоту числа?
7. На якій теоремі ґрунтується сертифікат Пратта?

8. Для яких чисел рекомендують генерувати сертифікат Пратта?

9. На чому базується побудова сертифіката про простоту числа Голдвассера-Кіліана?

10. Назвіть недоліки алгоритму побудови сертифіката про простоту числа Голдвассера–Кіліана.

11. На чому базується побудова еліптичних кривих в алгоритмі Аткіна-Морейна?

12. Назвіть складові сертифікату простоти Голдвассера-Кіліана-Аткіна-Морейна.

13. Для яких чисел рекомендують генерувати сертифікат Голдвассера-Кіліана-Аткіна-Морейна?

14. Який результат імовірнісного тесту простоти називають свідком складеності?

15. Який результат імовірнісного тесту простоти називають брехуном простоти?

16. Який критерій простоти є основою теста Ферма?

17. Який суттєвий недолік має тест Ферма?

18. Яку властивість мають числа Кармайкла стосовно теста Ферма?

19. Чи розпізнає ймовірнісний тест простоти Соловея-Штрассена числа Кармайкла як складені?

20. На якому критерії ґрунтується ймовірнісний тест простоти Соловея-Штрассена?

21. Яка ймовірність оголошення складеного цілого числа простим за результатом виконання *t* ітерацій тесту Соловея-Штрассена?

22. На якому критерії ґрунтується ймовірнісний тест простоти Міллера-Рабіна?

23. Яка ймовірність оголошення складеного цілого числа простим за результатом виконання *t* ітерацій тесту Міллера-Рабіна?

24. На чому ґрунтується ймовірнісний тест простоти Люка?

25. Які складені числа називають псевдопростими числами Ферма?

26. Які складені числа називають псевдопростими числами Пуле?

27. Які складені числа називають простими числами індустріального класу?

ВПРАВИ

1. Підготувати набір простих чисел для перевірки на простоту чисел, що не перевищують 10000.

2. Використовуючи пробне ділення визначити, чи є число 2047 простим.

3. Використовуючи пробне ділення визначити, чи є число 2131 простим.

4. Використовуючи тест простоти AKS визначити, чи є число 21 простим.

5. Використовуючи тест простоти AKS визначити, чи є число 19 простим.

6. Використовуючи алгоритм 4.3 визначити, чи є число 243 досконалим степенем.

7. Використовуючи алгоритм 4.3 визначити, чи є число 143 досконалим степенем.

8. Використовуючи алгоритм 4.4 визначити, чи є число 8191 простим числом Мерсенна.

9. Створити сертифікат Пратта про простоту числа n=89.

10. Використовуючи тест Ферма визначити, чи є число 179 простим.

11. Використовуючи тест Ферма визначити, чи є число 187 простим.

12. Обчисленнями доведіть, що число 12025 є числом Кармайкла.

13. Обчислити символ Якобі для $a = 7$ і $n = 79$.

14. Обчислити символ Якобі для $a = 13$ і $n = 79$.

15. Використовуючи тест Соловея-Штрассена визначити, чи є число 79 простим.

16. Використовуючи тест Міллера-Рабіна визначити, чи є число 79 простим.

ДЖЕРЕЛА ДЛЯ ПОГЛИБЛЕНОГО ВИВЧЕННЯ

1. M. Agrawal and S. Biswas, «Primality and identity testing via Chinese remaindering2. In «*40th Annual Symposium on Foundations of Computer Science* (*New York*, 1999)», *IEEE Computer Soc*., Los Alamitos, CA, 1999, pp. 202-208.

2. M. Agrawal, N. Kayal, and N. Saxena, «PRIMES is in P,» *Ann. of Math*. 160 (2004), pp. 781-793. http://www.cse.iitk.ac.in/news/primality.pdf

3. W. R. Alford, A. Granville and C. Pomerance, «There are infinitely many Carmichael numbers», *Ann. of Math.* (2), 139 (1994), pp. 703-722.

A. Atkin and F. Morain, «Elliptic curves and primality proving,» *Math. Comp.*, 61:203 (July 1993), pp. 29-68.

4. J. Buchmann, J. Loho and J. Zayer, «An Implementation of the General Number Field Sieve» in *Advances in Cryptology*, Springer-Verlag, vol. 773, (1994), pp. 159-165.

5. R. J. Burthe, «Further Investigations with the Strong Probable Prime Test», *Mathematics of Computation*, vol. 65, (1996), pp. 373-381.

6. R. Bhattacharjee and P. Pandey, «Primality testing,» IIT Kanpur, (2001) http://www.cse.iitk.ac.in/research/btp2001/primality.html.

7. Certificate of primality https://t5k.org/glossary/page.php?sort=Certificate

8. D. V. Chudnovsky and G. V. Chudnovsky, «Sequences of Numbers Generated by Addition in Formal Groups and New Primality and Factorizations Tests», *Advances in Applied Mathematics*, vol. 7, (1987), pp. 385-434.

9. H. Cohen and Lenstra, Jr., H. W., «Primality testing and Jacobi sums», *Math. Comp.*, 42 (1984), pp. 297-330.

10. R. Crandall and C. Pomerance, «Prime numbers: a computational perspective», 2nd ed., Springer, New York, 2005. 612 p.

11. S. Goldwasser and J. Kilian, «Almost all primes can be quickly certified», In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing* (1986), pp. 316-329.

12. J. Grantham, «A probable prime test with high confidence», *J. Number Theory*, 72 (1998), pp. 32-47

13. G. Jaeschke, «On strong pseudoprimes to several bases», *Math. Comp.*, 61 (1993), pp. 915-926.

14. N. Koblitz, «A Course in Number Theory and Cryptography», 2nd edition, Springer-Verlag, 1994, 245 p.

15. G. Miller, «Riemann's hypothesis and tests for primality». *J Comput Syst Sci* 13: (1976), pp. 300-317.

16. M. Morrison, «A note on primality testing using Lucas sequences», *Math. Comp.*, 29 (1975), pp. 181-182.

17. R. Pinch, «The Carmichael numbers up to 10^{15}», *Math. Comp.*, 61:203 (1993), pp. 381-391.

18. C. Pomerance. «Very short primality proofs». *Math. Comp.*,48: (1987), pp. 315-322.

19. C. Pomerance, «Lecture notes on primality testing and factoring (notes by G. M. Gagola Jr.)», Notes Vol, 4, *Mathematical Association of America*, 1984. 34 pages.

20. V. Pratt. «Every prime has a succinct certificate». *SIAM J. Comput.*,4: (1975), pp. 214-220.

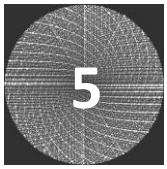
21. M. O. Rabin, «Probabilistic algorithm for testing primality», *J. Number Theory*, 12 (1980), pp. 128-138.

22. R. D. Silverman, «The Multiple Polynomial Quadratic Sieve», *Mathematics of Computation*, vol. 48, (1987), pp. 329-339.

23. R. Solovay and V. Strassen, «A fast Monte-Carlo test for primality». *SIAM J Comput* 6(1): (Mar 1977), pp. 84-85.

24. H. C. Williams, «Primality testing on a computer», *Ars Combin.*, 5 (1978), pp. 127-185.

25. H. C. Williams, «Édouard Lucas and primality testing», *Canadian Math. Soc. Series of Monographs and Adv*. Texts volume 22, John Wiley & Sons, pp. x+525, New York, NY, 1998.



ГЕНЕРУВАННЯ ВЕЛИКИХ ПРОСТИХ ЧИСЕЛ

5.1 Підходи до генерування великих простих чисел

Відповідно до теореми простих чисел (теорема 1.3), імовірність того, що випадково вибране число x є простими, дорівнює приблизно $1/\ln x$. Це припускає, що розумною стратегією отримання випадкового k-бітового ймовірно простого числа є багаторазове випадкове вибирання k-бітових непарних цілих чисел (кандидатів) n та перевірка їх на простоту, доки не буде знайдено одне, яке буде оголошено «*prime*».

Загальний підхід щодо генерування простих чисел реалізується за допомогою такого алгоритму.

Алгоритм 5.1. Узагальнене генерування простих чисел

***Вхід*:** тест на простоту **T**, обмежувальні властивості Ꝑ.

***Вихід*:** просте ціле число n.

***Процес*:**

1. Згенерувати випадкового кандидата n, перевіряючи властивості Ꝑ.
2. Поки $\mathbf{T}(n) = false$, виконувати:
   2.1. Оновити n зі збереженням властивостей Ꝑ.
3. Повернути n.

Кожен кандидат n має задовольняти деяким властивостям Ꝑ. Метою цієї вимоги є зменшення середньої кількості звернень до тесту **T**, що є найскладнішою процедурою алгоритму, уникаючи кандидатів, які є складеними. Без цієї вимоги середня кількість звернень до тесту **T** під час генерування l-розрядного простого числа близька до $\ln(2^l)$.

Найпростіша властивість Ꝑ полягає в тому, щоб дозволити n бути непарним числом. У цьому випадку оновлення кандидата n відбувається шляхом додавання до нього 2 (послідовністю пошуку є n, $n+2$, $n+4$,…, $n+2(s-1)$) і середня кількість звернень до тесту **T** зменшується до $\ln(2^l)/2$.

Узагальнення цієї ідеї полягає в тому, щоб прийняти для Ꝑ властивість, що $\text{НСД}(n, M) = 1$, де M – добуток k найменших простих чисел $p_1,...,p_k$, тобто $M = p_1 \cdot p_2 \cdot ... \cdot p_k$. У цьому випадку можна виконати пробне ділення на кожне просте число достатньо ефективно таким чином. Спочатку створюється масив значень $\omega_i = n \bmod p_i$ для i = 1,…, k. Кожного разу, коли 2 додається до поточного кандидата, значення масиву оновлюються як $\omega_i = (\omega_i + 2) \bmod p_i$. Кандидат проходить етап пробного ділення тоді і тільки

тоді, коли жодне зі значень ω_i не дорівнює 0.

Якщо M велике, альтернативним методом виконання пробного ділення є ініціалізація масиву елементами $S[i]=0$ для $0 \le i \le (s-1)$; елемент $S[i]$ відповідає кандидату $n+2i$. Для кожного p_i обчислюється $n \bmod p_i$. Нехай j є найменшим індексом, для якого $(n+2j) \equiv 0 \bmod p_i$. Тоді $S[j]$ і кожному елементу після нього встановлюються значення 1. Кандидат $n+2i$ проходить етап пробного діленням тоді і тільки тоді, коли $S[i]=0$

Незалежно від вибраних методів реалізації кроку 1 алгоритму 5.1, середня кількість звернень до **T** близька до $N(l,M)=\ln(2^l)\dfrac{\varphi(M)}{M}$, де $\varphi(M)$ – функція Ейлера.

Методи генерування простих чисел розрізняють за використанням випадковості. Кандидати *n* зазвичай генеруються на основі функції випадкового входу, а процедура визначення простоти кандидата може або використовувати, або не використовувати випадкові числа. Якщо не використовуються випадкові числа, то метод називають *детермінованим* (*deterministic*). У разі використання випадкових чисел метод називають *рандомізованим* (*randomized*).

Функцію випадкового входу реалізують генератори випадкових бітів. Існують дві принципово різні стратегії генерування випадкових бітів (*generating random bits*).

Одна з них полягає в тому, щоб виробляти біти недетерміновано (*non-deterministically*), де кожен біт виводу базується на непередбачуваному фізичному процесі. Цей клас генераторів випадкових бітів (RBGs – Random Bit Generators) відомий як недетерміновані генератори випадкових бітів (NRBGs – Nondeterministic Random Bit Generators).

Інша стратегія полягає в обчисленні бітів за допомогою детермінованого алгоритму. Цей клас RBGs відомий як детерміновані генератори випадкових бітів (DRBGs – Deterministic Random Bit Generators). Через детерміновану природу процесу, кажуть, що DRBG виробляє псевдовипадкові біти, а не випадкові. DRBG базується на механізмі DRBG і містить джерело вхідної ентропії. Механізм DRBG використовує алгоритм, що створює послідовність бітів із початкового значення, яке визначається на основі вхідної ентропії. Ця послідовність бітів потім може використовуватися безпосередньо або перетворюватися на випадкові числа, потрібні криптографічним алгоритмам.

5.2 Функціональна модель RBG

Методи генерування випадкових бітів визначено Рекомендацією NIST SP 800-90 **(**NIST Special Publication 800-90. Revised Recommendation for Random Number Generation Using Deterministic Random Bit Generators).

На рис. 5.1 подано функціональну модель RBG, який використовує механізм DRBG.

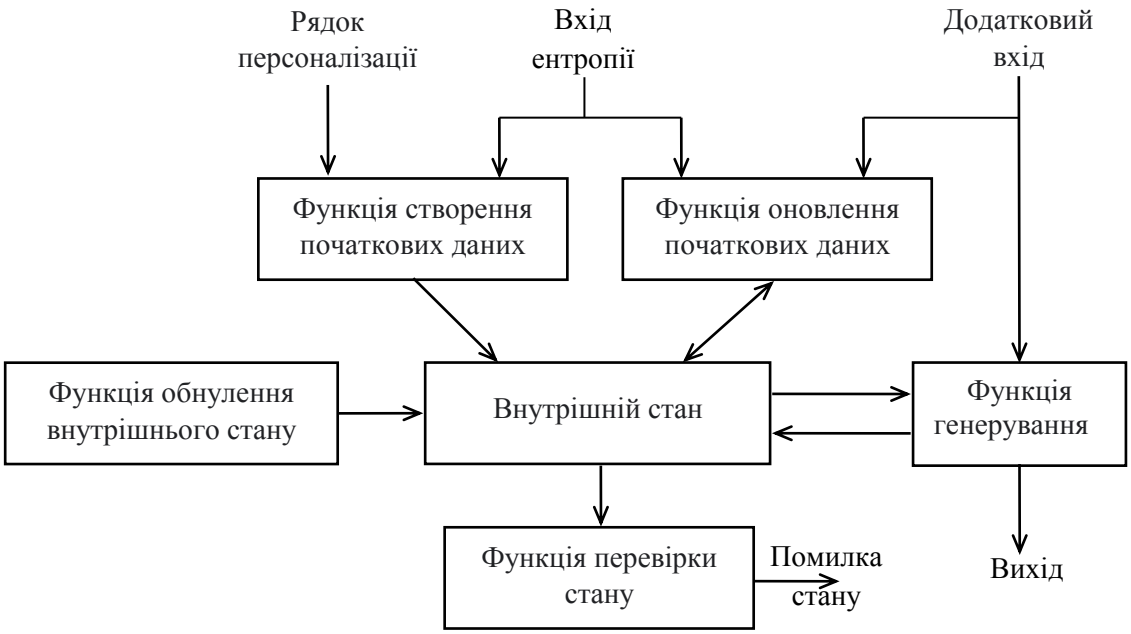


Рисунок 5.1 – Функціональна модель RBG

RBG має три входи: вхід ентропії, вхід рядка персоналізації та додатковий вхід.

Джерелом введення ентропії має бути або затверджений NRBG, або затверджений DRBG, таким чином утворюючи ланцюжок принаймні двох DRBG.

Введена ентропія та початкове значення мають зберігатися в секреті. Секретність цієї інформації є основою для безпеки RBG. Вхід ентропії може бути визначений як змінної довжини (в заданих межах), так і фіксованої довжини.

Рядок персоналізації – це бітовий рядок, який є максимально унікальним і який може містити секретну інформацію.

Метою рядка персоналізації є забезпечення відмінності цього DRBG від усіх інших, які можуть бути коли-небудь створені.

До складу рядка персоналізації можуть входити:

- серійний номер пристрою;
- відкритий ключ;
- ідентифікатор користувача;
- приватний ключ;
- PIN-код та пароль;
- мітка часу;
- мережева адреса;
- ідентифікатор програми;
- ідентифікатор версії протоколу;
- випадкове число.

Додатковий вхід використовується для введення додаткових даних під час кожного запиту бітів із DRBG і під час оновлення початкових даних. Однак це введення є необов’язковим. Додаткові дані можуть бути таємними або відкритими. Використання додаткових даних може забезпечити більшу ентропію для внутрішнього стану DRBG, що підвищить гарантію виконання вимог до ентропії.

Залежно від методу отримання вхідних даних їх значення може бути або може і не бути відомим користувачу чи програмі-споживачу. Наприклад, вхідні значення можуть бути отримані безпосередньо зі значень, введених користувачем або програмою-споживачем, або можуть бути виходом іншого RBG.

Внутрішній стан є пам'яттю DRBG і складається з усіх параметрів, змінних та інших збережених значень, які використовує механізм DRBG або на які діє. Внутрішній стан змінюється, коли генератор отримує запит на надання нових псевдовипадкових бітів. Внутрішній стан DRBG існує виключно в межах механізму DRBG і є недоступним функціям, що не належать до DRBG. Інформація про секретні частини внутрішнього стану DRBG та проміжні значення в обчисленнях із залученням цих секретних частин не впливає на будь-яку інформацію, яка залишає межі механізму DRBG, за винятком псевдовипадкових бітів.

Функції механізму DRBG обробляють внутрішній стан DRBG.

Функція створення початкових даних отримує ентропію і рядок персоналізації, на основі яких утворює дані, що є початковим внутрішнім станом.

Функція генерування генерує псевдовипадкові біти на запит, використовуючи поточний внутрішній стан і створює новий внутрішній стан для наступного запиту.

Ця функція:

1. Перевіряє правильність вхідних параметрів.
2. Викликає функцію оновлення початкових даних, щоб отримати достатню ентропію для створення внутрішнього стану.
3. Генерує запитані псевдовипадкові біти за допомогою алгоритму генерування.
4. Оновлює внутрішній стан.
5. Повертає запитані псевдовипадкові біти програмі-споживачу.

Функція оновлення початкових даних отримує новий вхід ентропії та поєднує його з поточним внутрішнім станом і даними додаткового входу, які надаються для створення нових початкових даних та нового внутрішнього стану. Повторне заповнення є засобом відновлення секретності виведення DRBG, якщо початковий або внутрішній стан стає відомим. Періодичне оновлення початкових даних усуває загрози, що є наслідком компрометації початкового значення, ентропії або внутрішнього стану в деякий реалізації (наприклад, смарт-картка).

Функція обнулення внутрішнього стану скасовує внутрішній стан.

Функція перевірки стану визначає, чи правильно продовжує працювати механізм DRBG.

5.3 Генерування псевдовипадкових бітів за допомогою механізму DRBG на основі геш-функції

Максимальний ступінь безпеки, який може підтримувати кожен DRBG на основі геш-функції визначається ступенем безпеки використаної геш-функції. Перелік геш-функцій, які рекомендується використовувати для генерування псевдовипадкових чисел, наведено в SP 800-57. Це геш-функції SHA-224, SHA-256, SHA-384 і SHA-512.

Складовими внутрішнього стану (*internal_state*) для Hash DRBG є:

1. Робочий стан (*working_state*):
   а) значення *V* довжини *seedlen* бітів, які оновлюються під час кожного виклику DRBG.
   б) константа *C* довжини *seedlen* бітів, яка залежить від початкового значення *seed*.
   в) лічильник (*reseed_counter*), що вказує кількість запитів для генерування псевдовипадкових бітів.
2. Адміністративна інформація:
   а) ступінь безпеки (*Security_strength*) екземпляра DRBG.
   б) прапорець *Prediction_resistance_flag*, який вказує, передбачена чи ні потрібна стійкість для створення екземпляра DRBG.

Значення *V* і *C* є секретними значеннями внутрішнього стану, тобто безпека цього механізму DRBG залежить від них.

Механізми DRBG підтримують чотири ступеня безпеки: 112, 128, 192 або 256 біт. Ступінь безпеки забезпечується відповідною кількістю ентропії.

Довжина *seedlen* початкового значення за використання SHA-224 і SHA-256 дорівнює 440 біт і 888 біт – за використання SHA-384 та SHA-512.

Інтервал оновлення початкового значення (*reseed_interval*) $\leq 2^{48}$.

Нехай **Hash** буде обраною геш-функцією.

Алгоритм 5.2. Hash DRBG Generate algorithm

***Вхід*:** 1. *working_state* (робочий стан): поточні значення для *V*, *C* і *reseed_counter* (лічильник оновлень початкових значень);
2. *requested_number_of_bits* (запитана кількість бітів): кількість псевдовипадкових бітів, які має повернути функція генерування;
3. *additional_input* (додатковий вхід): додатковий вхідний рядок, отриманий від споживача додатку.

***Вихід*:** 1. *status*: Статус, що повертає функція генерування. Буде вказано статус SUCCESS або вказано, що потрібне повторне заповнення перед запитом.
2. *returned_bits*: Псевдовипадкові біти, які повертає функція генерування.
3. new_working_*state*: нові значення для *V*, *C* і *reseed_counter*.

***Процес*:**

1. Якщо *reseed_counter* > *reseed_interval*, то повернути вказівку про

повторний запуск.

2. Якщо (*additional_input* ≠ *Null*), то виконати:
   2.1. Покласти *w* = **Hash** (0x02 || *V* || *additional_input*).
   2.2. Обчислити $V = (V + w)2^{seedlen}$.
3. Покласти (*returned_bits*) = **Hashgen** (*requested_number_of_bits*, *V*).
   Коментар: Використати алгоритм 5.3.
4. Покласти *H* = **Hash** (0x03 || *V*).
5. Обчислити $V = (V + H + C + reseed\_counter)2^{seedlen}$.
6. Покласти *reseed_counter* = *reseed_counter* + 1.
7. Повернути: SUCCESS, *returned_bits* і нові значення *V*, *C* і *reseed_counter* для *new_working_state*.

Алгоритм 5.3. Hashgen (...)

***Вхід*:** 1. *requested_no_of_bits*: кількість бітів, які потрібно повернути.
2. поточне значення *V*.

***Вихід*:** 1. *returned_bits*: згенеровані біти, які повертаються функції генерування.

***Процес*:**

1. Обчислити $m = \left\lceil \frac{requested\_no\_of\_bits}{outlen} \right\rceil$
2. Покласти *data* = *V*.
3. Покласти *W* = рядок *Null*.
4. Для *i* від 1 до *m*: виконувати:
   4.1. Покласти w_i = **Hash** (*data*).
   4.2. Покласти $W = W \| w_i$.
   4.3. Обчислити $data = (data + 1)2^{seedlen}$.
5. Покласти *returned_bits* = крайні ліві (*requested_no_of_bits*) біти *W*.
6. Повернути *returned_bits*.

5.4 Генерування випадкових імовірно простих чисел

Стандарт IEEE Std 1363-2000 рекомендує алгоритми генерування випадкових імовірно простих чисел у заданому інтервалі, які також задовольняють умову, що $p-1$ є взаємно простим із заданим непарним додатним цілим числом *f*. Така умова необхідна для формування параметрів сімейства криптографічних методів IF (Integer Factorization). У випадку RSA (Rivest–Shamir–Adleman) *f* дорівнює публічному показнику. У випадку RW (Rabin–Williams) *f* дорівнює найбільшому непарному дільнику публічної експоненти. Для програм, де така умова не потрібна, береться *f* =1.

Алгоритм 5.4. Генерування випадкових простих чисел

***Вхід*:** нижня межа $p_{\min}$ для *p*; верхня межа $p_{\max}$ для *p*; непарне додатне ціле число *f*.

***Вихід*:** випадкове просте число p з інтервалу $p_{\min} \le p \le p_{\max}$, таке що НСД((p – 1), *f*) = 1.

***Процес*:**

1. Обчислити $k_{\min} = (p_{\min} - 1)/2$ і $k_{\max} = (p_{\max} - 1)/2$.
2. Згенерувати випадкове ціле число k з інтервалу $k_{\min} \le k \le k_{\max}$.
3. Покласти $p = 2k + 1$.
4. Обчислити d = НСД((p – 1), *f*).
5. Якщо $d = 1$, то:
   5.1. Перевірити p на простоту..
   5.2. Якщо p є простим числом, то вивести p і зупинитися.
6. Перейти до кроку 2.

Наступний алгоритм відрізняється від попереднього лише тим, що на просте число p накладає додаткову умову $p \equiv a \bmod r$ для деяких заданих a і r, які є взаємно простими.

Алгоритм 5.5. Генерування випадкових простих чисел з умовами конгруентності

***Вхід*:** цілі додатні числа $r > 2$ і a, які є взаємно простими; нижня межа $p_{\min}$ для p; верхня межа $p_{\max}$ для p; непарне додатне ціле число f.

***Вихід*:** випадкове просте число p з інтервалу $p_{\min} \le p \le p_{\max}$, що задовольняє $p \equiv a \bmod r$, і для якого $p - 1$ є взаємно простим з f.

***Процес*:**

1. Якщо a непарне, то покласти $b = a$, інакше покласти $b = a + r$. Якщо r непарне, то покласти $s = 2r$, інакше покласти $s = r$.
2. Обчислити $k_{\min} = \lceil (p_{\min} - b)/s \rceil$ і $k_{\max} = \lfloor (p_{\max} - b)/s \rfloor$.
3. Згенерувати випадкове ціле число k з інтервалу $k_{\min} \le k \le k_{\max}$.
4. Покласти $p = sk + b$.
5. Обчислити d = НСД(($p - 1$), *f*).
6. Якщо $d = 1$, то:
   6.1 Перевірити p на простоту..
   6.2 Якщо p є простим числом, то вивести p і зупинитися.
7. Перейти до кроку 3.

Криптосистема RSA використовує модуль $n = pq$, де p і q непарні прості числа. Ці числа мають бути достатньої розрядності, щоб було практично неможливо розкласти модуль n на множники. Крім того, вони мають вибиратися випадковим чином. Водночас ймовірність будь-якого окремого простого числа, що вибирається, має бути достатньо малою, щоб не дати супротивнику можливість сформувати стратегію пошуку на основі такої ймовірності.

Криптосистема RSA висуває ще додаткові обмеження на вибір p і q, які

забезпечують її стійкість до криптоаналітичних атак. Ці обмеження сформульовано через поняття сильного простого числа.

Алгоритм 5.6. Створення сильного простого числа

***Вхід*:** нижня межа $p_{\min}$ для p; верхня межа $p_{\max}$ для p; бажані бітові довжини $L(r)$, $L(s)$ і $L(t)$; непарне додатне ціле число z.

***Вихід*:** випадкове сильне просте число p з інтервалу $p_{\min} \le p \le p_{\max}$, таке що НСД((p – 1), z) = 1.

***Процес*:**

1. Згенерувати випадкове просте число t з інтервалу

$$2^{L(t)-1} \le t \le 2^{L(t)} - 1.$$

Коментар: Застосувати алгоритм 5.4 з $f = 1$.

2. Згенерувати випадкове просте число r з інтервалу

$2^{L(r)-1} \le r \le 2^{L(r)} - 1$, що задовольняє $r \equiv 1 \bmod t$.

Коментар: Застосувати алгоритм 5.4 з $f = 1$.

3. Згенерувати випадкове просте число s з інтервалу

$$2^{L(s)-1} \le s \le 2^{L(s)} - 1.$$

Коментар: Застосувати алгоритм 5.4 з $f = 1$.

4. Обчислити $u = 1/s \bmod r$.
5. Обчислити $v = 1/r \bmod s$.
6. Обчислити $a = su - rv \bmod rs$.
7. Згенерувати випадкове просте число p з інтервалу $p_{\min} \le p \le p_{\max}$, що задовольняє $p \equiv a \bmod rs$.

Коментар: Застосувати алгоритм 5.5 з $f = z$.

8. Вихід: p.

Якщо сильне просте число p також має задовольняти додаткову умову конгруентності $p \equiv h \bmod m$, то останні кроки алгоритму потрібно замінити такими кроками:

7. Обчислити $b = 1/(rs) \bmod m$.
8. Обчислити $k = 1/m \bmod rs$.
9. Обчислити $c = akm + bhrs \bmod mrs$.
10. Згенерувати випадкове просте число p з інтервалу $p_{\min} \le p \le p_{\max}$, що задовольняє $p \equiv c \bmod mrs$.

Коментар: Застосувати алгоритм 5.5 з $f = z$.

11. Вихід: p.

Для створення ймовірно простого числа (кандидат на p або q криптосистеми RSA) стандарт FIPS PUB 186-4 рекомендує алгоритм, що передбачає використання двох допоміжних простих чисел і китайської теореми про залишки.

Алгоритм 5.7. Генерування ймовірно простого числа

***Вхід*:** 1) r_1 і r_2 – непарні прості числа, для яких
$\log_2(r_1 \cdot r_2) \le (nlen/2) - \log_2(nlen/2) - 6$.
2) *nlen* – бажана довжина *n*, модуль RSA.
3) *e* – експонента публічної перевірки.
4) *security_strength* – мінімальний рівень безпеки, необхідний для генерування випадкових чисел.

***Вихід*:** 1) *status* – статус, який повертається алгоритмом, де статус – SUCCESS (успішно) або FAILURE (помилка). Якщо повертається FAILURE, тоді як інші вихідні значення повертаються нулі.
2) *private_prime_factor* – простий множник числа *n*.
3) *X* – випадкове число, що використовується під час генерування *private_prime_factor*.

***Процес*:**

1. Якщо ($\text{НСД}(2r_1, r_2) \ne 1$), то повертається (FAILURE, 0, 0).

2. Обчислити $R = ((r_2^{-1} \bmod 2r_1) \cdot r_2) - (((2r_1)^{-1} \bmod r_2) \cdot 2r_1)$.

Коментар: Застосувати китайську теорему про залишки таким чином, щоб $R \equiv 1 \bmod 2r_1$ і $R \equiv -1 \bmod r_2$.

3. Згенерувати випадкове число *X* за допомогою схваленого генератора випадкових чисел, який забезпечує рівень безпеки *security_strength*, таке, що $\left(\sqrt{2}\right)\left(2^{(nlen/2)-1}\right) \le X \le \left(2^{nlen/2} - 1\right)$.

4. Обчислити $Y = X + ((R - X) \bmod (2r_1 \cdot r_2))$.

Коментар: *Y* – перше непарне ціле число $\ge X$, таке, що r_1 простий множник числа *Y*–1, а r_2 – простий множник числа *Y*+1.
Коментар: Визначити необхідне просте число шляхом побудови кандидатів із послідовності та виконання тестів на простоту.

5. Покласти $i = 0$.

6. Якщо $\left(Y \ge 2^{(nlen/2)}\right)$, то перейти до кроку 3.

7. Якщо ($\text{НСД}(Y - 1_1, e) = 1$) = 1), то виконати:
7.1. Перевірити простоту числа *Y* використовуючи алгоритм Міллера-Рабіна. Якщо PROBABLY PRIME не повертається, то перейти до кроку 8.
7.2. Покласти *private_prime_factor* = *Y*.

7.3. Повернути (SUCCESS, *private_prime_factor*, *X*).

8. Покласти $i = i + 1$.

9. Якщо ($i \geq 5(nlen/2)$), то повернути (FAILURE, 0, 0).

10. Обчислити $Y = Y + (2r_1r_2)$.

11. Перейти до кроку 6.

5.5 Побудова доказово простих чисел

Розглянемо алгоритми, що не генерують прості числа, застосовуючи тест на простоту до випадково вибраних кандидатів, а конструюють особливим чином цілі числа, які гарантовано є простими.

Для побудови доказово простого числа (*provable prime*) стандарт FIPS PUB 186-4 рекомендує алгоритм, який реалізує процедуру Шоу-Тейлора (Shawe-Taylor). Ця процедура є рекурсивною і передбачає використання геш-функції.

Нехай Hash() буде вибраною геш-функцією, а *outlen* – довжиною в бітах вихідного блоку геш-функції.

Алгоритм 5.8. ST_Random_Prime

***Вхід*:** 1) *length* – довжина простого числа, яке буде згенеровано.

2) *input_seed* – початкове число, яке буде використано для генерування простого числа.

***Вихід*:** 1) *status* – статус є SUCCESS або FAILURE. Якщо повертається FAILURE, то як інші вихідні значення повертаються нулі.

2) *prime* – запитуване просте число.

3) *prime_seed* – просте число, визначене під час генерування.

4) *prime_gen_counter* – (необов'язково) лічильник, визначений під час генерування простого числа.

***Процес*:**

1. Якщо *length* < 2, то повернути (FAILURE, 0, 0, {0}).

2. Якщо *length* ≥ 33, то перейти до кроку 11.

3. Покласти *prime_seed* = *input_seed*.

4. Покласти *prime_gen_counter* = 0.

Коментар: Створення псевдовипадкового цілого числа *c* довжини *length* біт.

5. Обчислити *c* = Hash(*prime_seed*) $\oplus$ Hash(*prime_seed* + 1).

6. Обчислити $c = 2^{length-1} + (c \bmod 2^{length-1})$.

7. Обчислити $c = 2 \cdot \lfloor c/2 \rfloor + 1$.

Коментар: Вибір як простого числа найменшого непарного цілого числа, яке більше або дорівнює *c*.

8. Покласти *prime_gen_counter* = *prime_gen_counter* + 1.

9. Покласти *prime_seed* = *prime_seed* + 2.

10. Виконати детерміновану перевірку простоти числа *c*. (Наприклад, шляхом пробного ділення).

11. Якщо *c* – просте число, то:

11.1. Покласти *prime* = *c*.

11.2. Повернути (SUCCESS, *prime*, *prime_seed* {*prime_gen_counter*}).

12. Якщо *prime_gen_counter* > 4 *length*, то повернути (FAILURE, 0, 0, {0}).

13. Перейти до кроку 5.

14. Покласти (*status*, c_0, *prime_seed*, *prime_gen_counter*) = = (**ST_Random_Prime** (($\lceil length/2 \rceil + 1$), *input_seed*).

15. Якщо повернено FAILURE, то повернути (FAILURE, 0, 0, {0}).

16. Покласти *iterations* = $\lceil length/outlen \rceil - 1$.

17. Покласти *old_counter* = *prime_gen_counter*.

Коментар: Створення псевдовипадкового цілого числа *x* в інтервалі $[2^{length-1}, 2^{length}]$.

18. Покласти *x* = 0.

19. Для *i* від 0 до *iterations* обчислити:

$$x = x + (\mathrm{Hash}(prime\_seed + i)\, 2^{i \times outlen}$$

20. Покласти *prime_seed* = *prime_seed* + *iterations* + 1.

21. Обчислити $x = 2^{length-1} + (x \bmod 2^{length-1})$.

Коментар: Створення кандидата простого числа *c* в інтервалі $[2^{length-1}, 2^{length}]$.

22. Обчислити $t = \lceil x/2c_0 \rceil$.

23. Якщо $2tc_0 > 2^{length}$, то $t = \lceil 2^{length-2}/c_0 \rceil$.

24. Обчислити $c = 2tc_0 + 1$.

25. Покласти *prime_gen_counter* = *prime_gen_counter* + 1.

Коментар: Перевірка на простоту кандидата простого числа *c*; спочатку вибрати ціле число $2 \le a \le (c-2)$.

26. Покласти *a* = 0.

27. Для *i* від 0 до *iterations* обчислити:

$$a = a + (\mathrm{Hash}(prime\_seed + i)\, 2^{i \times outlen}.$$

28. Покласти *prime_seed* = *prime_seed* + *iterations*+1.

29. Обчислити $a = 2 + a \bmod (c-3)$.

30. Обчислити $z = a^{2t} \bmod c$.

31. Якщо НСД((*z* – 1), *c*)=1 і $z^{c_0} \equiv 1 \bmod c$, то

31.1. Покласти *prime* = *c*.

31.2. Повернути (SUCCESS, *prime*, *prime_seed*,{*prime_gen_counter*}).

32. Якщо (*prime_gen_counter* ≥ ((4·*length*) + *old_counter*)), то повернути (FAILURE, 0, 0,{0}).

33. Покласти $t = t + 1$.
34. Перейти до кроку 23.

Для створення L-розрядного доказово простого числа (кандидат на p або q криптосистеми RSA) стандарт FIPS PUB 186-4 рекомендує алгоритм, який базується на використанні одночасно побудованих допоміжних доказово простих чисел.

Якщо потрібне сильне просте число, то цей алгоритм може генерувати прості числа p_1 і p_2 (довжин бітів N_1 і N_2), які ділять $p-1$ і $p+1$, відповідно.

Цей алгоритм потребує, щоб $N_1 + N_2 \le L - \lceil L/2 \rceil - 4$. Значення для N_1 і N_2 потрібно вибирати такі, що $N_1 + N_2 \le (L/2) - \log_2(L) - 7$.

Нехай Hash буде вибраною геш-функцією з вихідним значенням довжиною *outlen* біт.

Алгоритм 5.9. Створення доказово простого числа

***Вхід*:** 1. L – додатне ціле число, що дорівнює запитуваній довжині бітів для p.
2. N_1 – додатне ціле число, що дорівнює запитуваній довжині бітів для p_1. Якщо $N_1 \ge 2$, то p_1 є непарним простим числом довжини N_1 бітів; інакше p_1=1.
3. N_2 – додатне ціле число, що дорівнює запитуваній довжині бітів для p_2. Якщо $N_2 \ge 2$, то p_2 є непарним простим числом довжини N_2 бітів; інакше p_2=1.
4. *firstseed* – бітовий рядок, що дорівнює першому початковому значенню, яке буде використано.
5. e – експонента публічної перевірки.

***Вихід*:** 1. *status* – статус, який повертається алгоритмом, де статус – SUCCESS (успішно) або FAILURE (помилка). Якщо повертається FAILURE, то як інші вихідні значення повертаються нулі.
2. p, p_1, p_2 – шукане просте число p разом із p_1 і p_2 має властивість, що p_1 ділить $p-1$, а p_2 ділить $p+1$.
3. *pseed* – значення, отримане під час генерування.

***Процес*:**
1. Якщо L, N_1 і N_2 неприйнятні, то повернути (FAILURE, 0, 0, 0, 0).
   Коментар: Згенерувати p_1 і p_2, а також просте число p_0.
2. Якщо $N_1 = 1$, то:
   2.1. Покласти $p_1 = 1$.
   2.2. Покласти $p_2seed = firstseed$.
3. Якщо $N_1 \ge 2$, то:
   3.1. Використовуючи N_1 як довжину та *firstseed* як *input_seed*, згенерувати p_1 і p_2seed .
   Коментар: Застосувати алгоритм5.8.
   3.2. Якщо повертається FAILURE, то повернути (FAILURE, 0, 0, 0, 0).
4. Якщо $N_2 = 1$, то:
   4.1. Покласти $p_2 = 1$.

4.2. Покласти $p_0seed = p_2seed$.

5. Якщо $N_2 \geq 2$, то:

5.1. Використовуючи N_2 як довжину та p_2seed як *input_seed*, згенерувати p_2 і p_0seed .

Коментар: Застосувати алгоритм5.8.

5.2. Якщо повертається FAILURE, то повернути (FAILURE, 0, 0, 0, 0).

6. Використовуючи $\lceil L/2 \rceil + 1$ як довжину та p_0seed як *input_seed*, згенерувати p_0 і *pseed* за алгоритмом 5.8. Якщо повертається FAILURE, то повернути (FAILURE, 0, 0, 0, 0).

Коментар: Згенерувати сильне просте число p в інтервалі $[(\sqrt{2})(2^{L-1}), 2^L - 1]$.

7. Покласти $iterations = \lceil L/outlen \rceil - 1$.

8. Покласти *pgen_counter* = 0.

Коментар: Згенерувати псевдовипадкове число x в інтервалі $[(\sqrt{2})(2^{L-1}) - 1, 2^L - 1]$.

9. Покласти $x = 0$.

10. Для i від 0 до $iterations$ виконати:

$x = x + (Hash(pseed + i)) \times 2^{i \times outlen}$.

11. Покласти $pseed = pseed + iterations + 1$.

12. Обчислити $x = \lfloor (\sqrt{2})(2^{L-1}) \rfloor + (x \bmod (2^L - \lfloor (\sqrt{2})(2^{L-1}) \rfloor))$.

Коментар: Згенерувати кандидата на просте число p.

13. Якщо ($НСД(p_0 p_1, p_2) \neq 1$), то повернути (FAILURE, 0, 0, 0, 0).

14. Обчислити y в інтервалі $[1, p_2]$ так, щоб $yp_0p_1 \equiv 1 \bmod p_2$.

15. Обчислити $t = \lceil ((2yp_0p_1) + x)/(2p_0p_1p_2) \rceil$.

16. Якщо ($(2(tp_2 - y)p_0p_1 + 1) > 2^L$), то обчислити

$t = \lfloor ((2yp_0p_1) + \lfloor (\sqrt{2})(2^{L-1}) \rfloor)/(2p_0p_1p_2) \rfloor$.

Коментар: p задовольняє

$p - 1 \equiv 0 \bmod (2p_0p_1)$ і $p + 1 \equiv 0 \bmod p_2$

17. Обчислити $2(tp_2 - y)p_0p_1 + 1$.

18. Покласти *pgen_counter* = *pgen_counter* + 1.

19. Якщо ($НСД(p - 1, e) = 1$), то

Коментар. Вибір цілого числа a з інтервалу $[2, p-2]$.

19.1. Покласти $a = 0$.

19.2. Для i від 0 до $iterations$ виконати:

$a = a + (Hash(pseed + i)) \times 2^{i \times outlen}$.

19.3. Покласти $pseed = pseed + iterations + 1$.

19.4. Обчислити $a = 2 + (a \bmod (p - 3))$.

Коментар: Перевірка p на простоту:

19.5. Обчислити $z = a^{2(tp_2 - y)p_1} \bmod p$.

19.6. Якщо ((НСД$(z-1, p) = 1$) і ($z^{p_0} \equiv 1 \bmod p$)), то повернути (SUCCESS, p, p_1, p_2, *pseed*).

20. Якщо (*pgen_counter* $\geq 5L$), то повернути (FAILURE, 0, 0, 0, 0).

21. Покласти $t = t+1$.

22. Перейти до кроку 16.

Прийнятні значення для $L=nlen/2$, N_1 і N_2 наведено в табл. 5.1. Мінімальна довжина для кожного з допоміжних простих чисел p_1 і p_2 залежить від *nlen*, де *nlen* – це довжина модуля n в бітах. Зверніть увагу, що *nlen* також називають розміром ключа. Максимальна довжина визначається *nlen* і тим, що просте число p є ймовірним чи доказовим.

Таблиця 5.1 – Мінімальна та максимальна довжини p_1 і p_2

nlen (біт)	Мінімальна довжина p_1 і p_2 (біт)	Максимальна довжина p_1 і p_2 (біт)	
		Імовірно просте число	Доказово просте число
1024	>100	<496	<239
2048	>140	<1007	<494
3072	>170	<1518	<750

5.6 Незаконні великі прості числа

Пошук великих простих чисел є дослідженнями з використанням певного математичного апарату і сучасних комп'ютерних систем. Публікація знайдених великих простих чисел, з одного боку, є висвітленням певних досягнень у теорії чисел, а з іншого боку, це може бути оприлюдненням інформації, що становить державну або комерційну таємницю.

Будь-яка інформація може розглядатися як велике двійкове число. В сучасній юриспруденції є поняття незаконного зображення (через непристойність чи секретний статус). Отже, якщо інформація є незаконною, то і число саме собою може бути незаконним.

⚠ **Означення 5.1.** Число, що подає інформацію, володіння, оприлюднення, розповсюдження чи передачу якої заборонено в певній правовій юрисдикції, називають *незаконним числом* (*illegal number*).

Незаконні прості числа – це підмножина незаконних чисел.

Одне з перших незаконних простих чисел було створено у травні 2001 року Філом Кармоді (Phil Carmody). Його двійковий код пов'язаний з ущільненою версією програми мовою С, що виконує алгоритм дешифрування DeCSS, який може бути використаний для захисту від копіювання DVD. Публікація такого числа вважається незаконною у США згідно з законом про авторське право у цифрову епоху.

Ключ шифрування AACS (09 F9 11 02 9D 74 E3 5B D8 41 56 C5 63 56 88 C0), який став відомим у травні 2007 року, є прикладом простого числа, який, як стверджується, є таємним, і чия публікація або неналежне володіння є незаконним у Сполучених Штатах. Він нібито допомагає розшифровувати будь-які HD DVD або Blu-ray диски, випущені до цієї дати.

Один зі шляхів подолання юридичної заборони полягає в поданні незаконного коду у формі, яка має деяку властивість, що робить його дозволеним до публікації незалежно від законності самого коду. Оскільки бітам, що становлять комп'ютерну програму, відповідає число, то потрібно підібрати число, що має спеціальну властивість, яка робить його цікавим для публікації (одним із способів був друк числа на футболках, що популярно, наприклад, для простих чисел Мерсенна).

Простота числа – це фундаментальна властивість у теорії чисел, яка не залежить від юридичних визначень будь-якої країни.

Нехай ущільнений код програми є числом *k*. Відповідно до теореми Діріхле про прості числа в арифметичній прогресії, кожна арифметична прогресія з першим членом *b* і різницю *a*, де *a* і *b* – взаємно прості числа, містить нескінченну кількість простих чисел.

Використовуючи той факт, що програма для ущільнення gzip ігнорує байти після символу кінця (null terminated) ущільненого файлу, можна покласти $a = 256^i$ для такого *i*, що $a > b$. Тоді під час розархівовування числа $k \cdot a + b$ буде отримано число *k*. Це означає, що існує безліч простих чисел, які після розархівування дають один і той самий код. Таким чином, були згенеровані кандидати в прості числа, кожне з яких дає під час розархівовування код DeCSS мовою програмування C. Кармоді довів, що до них належать $k \cdot 256^2 + 2083$ і $k \cdot 256^{211} + 99$. Друге число є достатньо великим, щоб поміститися в список найбільших відомих простих чисел (через метод доведення).

Пізніше Чарльз М. Ханум (діючи за пропозицією Кармоді) знайшов варіант своїх коротких програм мовою C, для яких сам код (розглядається як число) є простим. Це можна зробити шляхом перейменування змінних без подовження програми. Оскільки кількість усіх символів ASCII у коді програми не перевищує 128, то він знайшов інший варіант, який є меншим, використовуючи 7 біт на символ.

Очевидно, немає кінця тому, що можна зробити!

5.7 Поняття асиметричного бекдору

Публікація «NIST Special Publication 800-90. Revised Recommendation for Random Number Generation Using Deterministic Random Bit Generators» містить специфікацію для трьох нібито криптографічно захищених генераторів псевдовипадкових чисел для використання в криптографії: Hash DRBG (на основі геш-функцій), HMAC DRBG (на основі HMAC) і CTR

DRBG (на основі блокових шифрів у режимі лічильника).

Генератори випадкових бітів NIST Hash_DRBG і HMAC DRBG мають підтвердження безпеки для одного виклику генерування псевдовипадкових чисел. Тому саме вони рекомендуються для використання в алгоритмах безпечного генерування великих простих чисел.

Стосовно генератора випадкових бітів NIST CTR DRBG було показано, що він має теоретичну недосконаліть у разі використання з певними параметрами, оскільки криптографи не враховували розмір блока шифру під час розробки цього генератора. Якщо AES використовується як базовий блоковий шифр і понад 128 біт беруться з цього генератора псевдовипадкових чисел, тоді рівень безпеки обмежується розміром блока, а не розміром ключа, і тому фактичний рівень безпеки набагато нижчий за рівень безпеки, що визначається розміром ключа. Наразі не існує відомого методу вирішення цієї проблеми. Тому генератор випадкових бітів NIST CTR DRBG може бути використаний лише для генерування послідовності бітів довжиною ≤ 128.

Попередні версії публікації включали четвертий генератор Dual EC DRBG (на основі криптографії еліптичної кривої). Пізніше повідомлялося, що Dual EC DRBG, ймовірно, містить клептографічний бекдор (асиметричний бекдор – asymmetric backdoor), вставлений Агентством національної безпеки США (NSA).

Традиційний бекдор – це симетричний бекдор: будь-хто, хто знайде бекдор, може ним також скористатися. Асиметричний бекдор може використовувати лише зловмисник, який його встановлює, навіть якщо повна реалізація бекдора стає загальнодоступною (наприклад, через публікацію, виявлення та розкриття шляхом зворотного проектування тощо). Крім того, обчислювально важко виявити наявність асиметричного бекдора запитами чорної скриньки. Для генерування ключів RSA існує експериментальний асиметричний бекдор OpenSSL RSA, розроблений Янгом і Юнгом, який використовує кручену пару еліптичних кривих і став доступним.

Клептографія (*Kleptography*) – це наука про безпечне та підсвідоме викрадення інформації.

Термін був введений Адамом Янгом і Моті Юнгом у Proceedings of Advances in Cryptology – Crypto '96.

Клептографія охоплює безпечні та приховані комунікації за допомогою криптосистем та криптографічних протоколів. Це нагадує стеганографію, яка вивчає приховані комунікації за допомогою графіки, відео, цифрових аудіоданих тощо.

Клептографія є продовженням теорії сублімінальних каналів (*subliminal channels*), яку започаткував Густавус Сіммонс (Gustavus Simmons).

У криптографії сублімінальні канали – це приховані канали (*covert channels*), які можна використовувати для таємного спілкування в звичайному вигляді по незахищеному каналу (*insecure channel*).

Сублімінальні канали в криптосистемах цифрового підпису були виявлені Сіммонсом у 1984 році. Він показав, як можна використовувати для

сублімінального надсилання повідомлення параметри алгоритмів цифрового підпису ElGamal і DSA, що мають бути встановлені за допомогою випадкової інформації. Оскільки процедура створення підпису не змінюється, то підпис залишається перевіреним і не відрізняється від звичайного підпису. Тому важко визначити, чи використовується сублімінальний канал.

Існують широкосмугові і вузькосмугові сублімінальні канали. Широкосмуговий канал використовує майже всі біти, які доступні каналу {≥50 % але ≤90 %}. Кожен канал, який використовує менше бітів, називають вузькосмуговим. Дослідження фахівців з криптографії показали, що алгоритм цифрового підпису має один сублімінальний широкосмуговий і три сублімінальні вузькосмугові канали.

Простим прикладом вузькосмугового сублімінального каналу для звичайного тексту людською мовою є передавання біту «0», коли кількість слів у реченні парна, а біту «1» в разі непарної кількості слів. Наприклад, питанню «Привіт, як справи?» буде відповідати сублімінальне повідомлення «1».

Під час підписання параметр k має бути встановлений випадковим чином. Замість цього, для широкосмугового каналу цей параметр встановлюється за допомогою сублімінального повідомлення m'. Покажемо це на прикладі.

1. Генерування ключів.
   1.1. Вибирають просте число $p = 2347$.
   1.2. Вибирають просте число $q = 23$.
   1.3. Обчислюють генератор групи $g = 266$.
   1.4. Вибирають ключ автентифікації $x = 1468$ і безпечно надсилають його одержувачу.
   1.5. Обчислюють відкритий ключ $y = g^x \bmod p = 2100$.
2. Підписання (обчислення підпису).
   2.1. Вибирають повідомлення $m = 1337$.
   2.2. (Обчислення значення геш-функції $H(m)$ тут замінено на обчислення за модулем 107) $h = 1337 \bmod 107 = 53$.
   2.3. Замість випадкового значення $k = ?$ вибрано сублімінальне повідомлення $m' = 17$, тобто $k = m'$.
   2.4. Обчислюють $(k)^{-1} \bmod q = 17^{-1} \bmod 23 = 19$.
   2.5. Обчислюють значення підпису
   $$r = (g^k \bmod p) \bmod q = (266^{17} \bmod 2347) \bmod 23 = 12.$$
   2.6. Обчислюють значення підпису
   $$s = k^{-1}(h + x \cdot r) \bmod q = 19 \cdot (53 + 1468 \cdot 12) \bmod 23 = 3.$$
   2.7. Надсилають повідомлення з підписом (1337; 12 , 3).
3. Перевірка підпису.
   3.1. Одержувачу надійшло повідомлення з підписом $(m; r, s) = (1337;\ 12, 3)$.
   3.2. Обчислюють значення геш-функції

$h = m \bmod 107 = 1337 \bmod 107 = 53.$

3.3. Обчислюють $w = s^{-1} \bmod q = 3^{-1} \bmod 23 = 8.$

3.4. Обчислюють $u_1 = (h \cdot w) \bmod q = (53 \cdot 8) \bmod 23 = 10.$

3.5. Обчислюють $u_2 = (r \cdot w) \bmod q = (12 \cdot 8) \bmod 23 = 4.$

3.6. Обчислюють підпис

$v = ((g^{u_1} \cdot y^{u_2}) \bmod p) \bmod q = ((266^{10} \cdot 2100^4) \bmod 2347) \bmod 23 = 12.$

3.7. Оскільки $v = r$, то підпис є дійсним.

4. Виділення повідомлення m' на стороні одержувача.

Формула для виділення повідомлення m' виводиться з формули для обчислення значення підпису $s = k^{-1}(h + x \cdot r) \bmod q$. Оскільки $k = m'$, то $s = (m')^{-1}(h + x \cdot r) \bmod q$, $s \cdot m' = (h + x \cdot r) \bmod q$, $m' = (s)^{-1}(h + x \cdot r) \bmod q$.

$m' = (3)^{-1}(53 + 1468 \cdot 12) \bmod 23 = 17.$

Прикладом вузькосмугового сублімінального каналу є використання замість модуля RSA $n = p \cdot q$ модуля $n = p \cdot q \cdot r$. Розрахунок показує, що в повідомленні з цифровим підписом може бути прихований один зайвий біт. Для виявлення цього криптологи з Centrum Wiskunde & Informatica в Амстердамі розробили алгоритм доведення з нульовим знанням того, що $n = p \cdot q$.

Клептографія вивчає асиметричні бекдори в генераторах псевдовипадкових чисел, алгоритмах генерування ключів RSA, обміну ключами Діффі-Хеллмана, алгоритмах цифрового підпису, алгоритмах шифрування та інших криптографічних алгоритмах.

Генератор випадкових бітів NIST Dual EC DRBG має асиметричний бекдор. Алгоритм EC-DRBG використовує клептограму дискретного логарифму, що за означенням робить EC-DRBG криптотрояном. Як і програми-вимагачі, криптотроян EC-DRBG містить і використовує відкритий ключ зловмисника для атаки на хост-систему.

Протоколи SSL, SSH і IPsec вразливі до клептографічних атак. У кожному разі зловмисник може скомпрометувати конкретний криптографічний алгоритм або протокол, перевіряючи інформацію, в якій закодована інформація про бекдор (наприклад, відкритий ключ, цифровий підпис, повідомлення обміну ключами тощо), а потім використовуючи логіку асиметричного бекдора з використанням секретного ключа (зазвичай приватного ключа).

Практичні приклади клептографічних атак можна знайти в JCrypTool, платформно-незалежній версії проекту CrypTool з відкритим кодом https://www.cryptool.org/en/. Також в JCrypTool реалізована демонстрація запобігання клептографічним атакам за допомогою методу KEGVER.

Проект CrypTool (CT) розробляє найпоширеніші у світі безкоштовні програми електронного навчання в галузі криптографії та криптоаналізу. Усі навчальні програми в проекті CT мають відкритий код і доступні безкоштовно.

JCrypTool — це платформа електронного навчання з відкритим вихідним кодом, розроблена, щоб не тільки дозволити кожному експериментувати з криптографією, але й розробляти та розширювати платформу JCrypTool різними способами за допомогою власних криптографічних плагінів (алгоритми, аналіз, ігри, візуалізації).

JCrypTool входить у комплект із розширеними криптографічними бібліотеками BouncyCastle і FlexiProvider; інші бібліотеки можна легко додати до проекту. FlexiProvider є початковим постачальником за замовчуванням, але користувачі можуть вільно змінити постачальника за замовчуванням. Наскрізна функція шукає реалізацію будь-якого даного криптографічного алгоритму зверху вниз, тому навіть спеціалізований постачальник із лише невеликою підмножиною алгоритмів може бути обраний за замовчуванням.

? КОНТРОЛЬНІ ПИТАННЯ

1. Яка теорема є основою загального підходу до генерування великих простих чисел?
2. Які методи генерування простих чисел називають детермінованими?
3. Які методи генерування простих чисел називають рандомізованими?
4. Назвіть функції, що реалізує генератор RBG.
5. Що може використовуватися як джерело введення ентропії для генератора RBG?
6. Назвіть складові рядка персоналізації для генератора RBG.
7. Назвіть складові внутрішнього стану для Hash DRBG.
8. Які складові внутрішнього стану для Hash DRBG є секретними?
9. Які ступені безпеки підтримують механізми DRBG?
10. Які стандарти рекомендують алгоритми генерування випадкових імовірно простих чисел?
11. Чим принципово відрізняються алгоритми генерування випадкових імовірно простих чисел від алгоритмів побудови доказово простих чисел?
12. Яка наука вивчає безпечне та підсвідоме викрадення інформації?.
13. Чим принципово відрізняються криптографія, клептографія і стеганографія?
14. Скільки доступних бітів використовує широкосмуговий сублімінальний канал?
15. Скільки доступних бітів використовує вузькосмуговий сублімінальний канал?

ВПРАВИ

1. Показати на прикладі реалізацію алгоритму 5.4.
2. Показати на прикладі реалізацію алгоритму 5.5.
3. Показати на прикладі реалізацію алгоритму 5.6.
4. Показати на прикладі алгоритму цифрового підпису реалізацію сублімінального широкосмугового каналу для передавання повідомлення $m' = 14$.
5. Підібрати значення *k*, *a* і *b*, для яких значення виразу $k \cdot a + b$ є простим числом.

ДЖЕРЕЛА ДЛЯ ПОГЛИБЛЕНОГО ВИВЧЕННЯ

1. ANS X9.80, Prime Number Generation, Primality Testing and Primality Certificates.
2. M. Agrawal, N. Kayal and N. Saxena. «PRIMES is in P». *Annals of Mathematics*, 2: (2002), pp. 781-793.
3. A. O. L. Atkin and F. Morain. «Elliptic Curves And Primality Proving». *Mathematics of Computation*, 61: (1993), pp. 29-68.
4. L. Blum, M. Blum and M. Shub, «A Simple Unpredictable Pseudo-random Number Generator», *SIAM Journal on Computing* 15 (1986), pp. 364-383.
5. M. Blum and S. Micali, «How to Generate Cryptographically Strong Sequences of Pseudo-random Bits», *SIAM Journal on Computing* 13 (1984), pp. 850-864.
6. Easttom, Chuck. «A Study of Cryptographic Backdoors in Cryptographic Primitives». *Electrical Engineering* (*ICEE*), *Iranian Conference on*. pp. 1664-1669.
7. Esslinger, Bernhard; Vacek, Patrick. «The Dark Side of Cryptography: Kleptography in Black-Box Implementations». *Infosecurity Magazine*. Retrieved 18 March 2014.
8. J. Hastad, R. Impagliazzo, L. Levin, and M. Luby. «A pseudorandom generator from anyone-way function». *SIAM J. Computing*, 28: (1999), pp. 1364-1396.
9. IEEE Std. 1363-2000, Standard Specifications for Public Key Cryptography.
10. Illegal prime http://primes.utm.edu/glossary/page.php?sort=Illegal.
11. J. Lagarias. «Pseudorandom number generators in cryptography and number theory». In C. Pomerance, editor, *Cryptology and computational number theory*, volume 42 of *Pmc. Sympos. Appl. Math.*, pages 115-143. Amer. Math. Soc., 1990.
12. U. M. Maurer, «A Universal Statistical Test for Random Bit Generators», A. J. Menezes and S. A. Vanstone, Eds., *Advances in Cryptology*,

CRYPTO ‘90, *Lecture Notes in Computer Science* 537 (1991), Springer-Verlag, pp. 409-420.

13. U. M. Maurer, «A Universal Statistical Test for Random Bit Generators» in Advances in Cryptology, Springer-Verlag, vol. 537, (1991), pp. 409-420.

14. S. Micali and C. P. Schnorr, «Efficient Perfect Polynomial Random Number Generators», *Journal of Cryptology*, vol. 3, (1991), pp. 157-172.

15. P. Mihailescu, «Fast Generation of Provable Primes Using Search in Arithmetic Progressions», Yvo G. Desmedt, Ed., *Advances in Cryptology*, CRYPTO ‘94, *Lecture Notes in Computer Science* 839 (1994), Springer-Verlag, pp. 282-293.

16. NIST Special Publication 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators.

17. NIST FIPS 186-4 Digital Signature Standard (DSS).

18. NIST FIPS 202, SHA-3 Standard: Permutation-based Hash and Extendable Output Functions.

19. NIST Special Publication 800-175B, Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms.

20. NIST FIPS180 Secure Hash Standard (SHS).

21. NIST Special Publication 800-57 Revision 5 Recommendation for Key Management: Part 1 – General.

22. S. Park and K. Miller. «Random number generators: good ones are hard to find». *Comm. ACM*, 31: (1988), pp. 1192-1201.

23. Public Key Cryptography Standard (PKCS) #1, RSA Encryption Standard.

24. G. J. Simmons, «The Subliminal Channel and Digital Signatures». In Beth, T.; Cot, N.; Ingemarsson, I. (eds.). *Proceedings of Eurocrypt* '84. *Lecture Notes in Computer Science*. Vol. 209. Springer-Verlag. pp. 364-378.

25. G. J. Simmons, «Subliminal Communication is Easy Using the DSA». In Helleseth, T. (ed.). *Proceedings of Eurocrypt* '93. *Lecture Notes in Computer Science*. Vol. 765. Springer-Verlag. pp. 218-232.

26. J. Shawe-Taylor, «Generating Strong Primes», *Electronics Letters*, vol. 22, (July 1986), pp. 875-877.

27. A. Young and M. Yung, «The Dark Side of Black-Box Cryptography, or: Should we trust Capstone?». In Koblitz, Neal (ed.). *Advances in Cryptology* - CRYPTO '96: *16th Annual International Cryptology Conference, Santa Barbara, California, USA,* August 18–22, 1996, *Proceedings. Lecture Notes in Computer Science*. Springer Berlin Heidelberg. pp. 89-103.

28. Zagórski, Filip; Kutyłowski, Mirosław. «Bezpieczeństwo protokołów SSL/TLS i SSL w kontekście ataków kleptograficznych» [Security of SSL/TLS and SSL protocols in the context of kleptographic attacks]. kleptografia.im.pwr.wroc.pl (in Polish).

↓ ПРЕДМЕТНИЙ ПОКАЖЧИК

адитивні прості числа, 50
алгоритм
Аткіна-Морейна, 104
Голдвассера–Кіліана, 103
загального решета числового поля, 88
$p+1$ алгоритм Вільямса, 81
ρ алгоритм Полларда, 77
p - 1 алгоритм Полларда, 79

брехуни простоти, 117
Ферма, 118
Ейлера, 121

взаємно прості числа, 59

гарне просте число, 51
гіпотеза про прості-близнюки, 20
гіпотеза Рімана, 14

двоюрідні прості числа, 21
детерміновані генератори випадкових бітів (DRBG), 137
дзета-функція Рімана, 14
добутки Ейлера, 12
доказово прості числа, 145
досконалі степені, 98

ізольовані прості числа, 20
імовірнісний тест простоти
Люка, 128
Міллера-Рабіна, 124
Соловея-Штрассена, 120
імовірно просте число за базою a, 130
інтегральний логарифм,14

канонічний розклад на прості множники, 66
клептографія, 151
k-кортежі простих чисел, 19
k-майже просте число, 68
колесо факторизації, 73
критерії Ейлера, 120

лема Евкліда, 9

майже просте, 107
мала теорема Ферма, 118
Метод
квадратичного решета, 83
колесної факторизації, 72
раціонального решета, 86
факторизації Ферма, 73
факторизації Ейлера, 77
множина попарно взаємно простих чисел, 61
множина спільно взаємно простих чисел, 60

незаконне число, 149
нетривіальна факторізація, 67

первісне число, 55
перевірка повного квадрата, 130
переставне просте число, 51
Піфагограма, 16
послідовності
числові, 36
скінченні, 36
нескінченні, 36
цілочислові, 37
Люка, 37
прайморіал, 42
пробне ділення, 72, 93
прості-близнюки, 19
прості п’ятірки, 23
простий триплет,22
прості четвірки, 22
прості числа
безпечні, 43
Вагстафа,45
прайморіальні,42
індустріального класу, 132
Кінея, 48
кубові, 34
Люка, 49
Мерсенна, 45, 99
мультиплікативні, 50
Пелла, 49
Пірпонта, 47
Піфагора, 33

сильні, 44
Солінаса, 48
Софі Жермен,43
Фібоначчі, 49
Ченя, 44
просте число з видаленням, 54
прості числа, що скорочуються
зліва, 52
справа, 52
зліва і справа, 53
прості шістки, 24
псевдопрості числа, 130
Ейлера, 121
Ферма, 131

решето
Аткіна, 31
Ейлера, 29
Ератосфена, 27
Сундарама, 30
розрив простих чисел, 15
рядок персоналізації, 138

свідки складеності, 117
Ферма, 118
Ейлера, 121
сертифікат
Голдвассера-Кіліана-Аткіна-Морейна, 115
Пратта, 101
символ Якобі, 112, 120, 122, 123
скатертина Улама, 17
спеціальне решето числового поля, 83
спіраль Сакса, 18
сублімінальні канали, 151
суперпрості числа, 49

теорема
Евкліда, 10
Люка, 101
простих чисел, 13
тест
AKS, 96
Люка-Лемера, 99
Ферма, 118

фундаментальна теорема арифметики, 66
формула простих чисел
Вільсона, 34
Міллса, 35
Плуффа, 36
функції механізму DRBG, 139
функція підрахунку простих чисел, 12

центровані числа
десятикутні, 58
квадратні, 56
семикутні, 57
трикутні, 55

центровані прості числа
десятикутні, 58
квадратні, 56
семикутні, 57
трикутні, 56

числа
B-гладкі, 69
вільні від квадратів, 70
B-степенево гладкі, 70
Евкліда, 42
Каннінгема, 83
Кармайкла, 119
Кінея, 48
Куммера, 43
Люка, 40
Мерсенна, 45, 99
напівпрості, 68
Пелла, 41
Пуле, 131
сфенічні, 69
Ферма, 46
Фібоначчі, 41

щасливі числа, 50
b-щасливі прості числа, 51

Навчальне електронне видання
комбінованого використання.
Можна використовувати в локальному та мережному режимах

Володимир Андрійович Лужецький
Галина Василівна Шелепало

ПРОСТІ ЧИСЛА: властивості та застосування в криптографії

Навчальний посібник

Рукопис оформлено *В. Лужецьким*

Редактор *Т. Старічек*

Оригінал-макет виготовлено *Т. Старічек*

Підписано до видання 19.03.2024 р.
Гарнітура Times New Roman.
Зам. № Р2024-069.

Видавець та виготовлювач
Вінницький національний технічний університет,
Редакційно-видавничий відділ.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95,
м. Вінниця, 21021.
press.vntu.edu.ua;
Email: irvc.vntu@gmail.com
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.