Міністерство освіти і науки України
Національний технічний університет України
"Київський політехнічний інститут імені Ігоря Сікорського"
Навчально-науковий Фізико-технічний інститут

Теоретико-числові алгоритми в криптології

Комп’ютерний практикум №1

Пошук канонічного розкладу великого числа,
використовуючи відомі методи факторизації арифметика

Виконали студенти:
група ФІ-12
Юрчук Олексій
група ФІ-14
Фурутіна Євгенія

Київ 2024

Тема: Пошук канонічного розкладу великого числа, використовуючи відомі методи факторизації арифметика

Мета: Практичне ознайомлення з різними методами факторизації чисел, реалізація цих методів і їх порівняння. Виділення переваг, недоліків та особливостей застосування алгоритмів факторизації. Застосування комбінації алгоритмів факторизації для пошуку канонічного розкладу заданого числа

Завдання:
А) Написати програму, що реалізовують такі алгоритми:
1) тест на простому числа (Міллера-Рабіна/Соловея-Штрассена);
2) метод пробних ділень
3) метод ρ − Полларда
4) метод Померанця
Б) Створити та реалізувати алгоритм для пошуку канонічного розкладу числа, що використовує всі вище згадані алгоритми. Цей алгоритм повинен повертати канонічний розклад числа. Додатково алгоритм повинен інформувати користувача про кожен дільник, знайдений в процесі роботи алгоритму, а саме: сам дільник і часову відмітку, коли його було знайдено. Часові відмітки часу початку і кінця роботи алгоритму також повинні відображатися в цьому інформуванні.
В) Застосувати алгоритм, створений на попередньому кроці, для пошуку канонічного розкладу числа відповідного варіанту. (3233245835185415 83)
Г) Застосувати реалізовані алгоритми факторизації ρ-метод Полларда та метод Померанця почергово до всіх наступних чисел, знайти один дільник цього числа, заміряти час роботи кожного алгоритму на кожному числі, порівняти та пояснити результати.
Теоретичні відомості:

- В загальному про факторизацію:

Факторизація цілого числа — це пошук нетривіального дільника (одного, не обов'язково простого) заданого числа. Якщо велике складене число розкладено на добуток менших цілих чисел, кожне з яких є простим числом, то такий розклад називається *канонічним розкладом складеного числа.* Таким чином, завданням задачі пошуку канонічного розкладу натурального числа є пошук всіх його простих дільників. Доведено, що задача пошуку канонічного розкладу числа та задача факторизації є поліноміально еквівалентними за часовою складністю. Ці задачі вважаються складними, тобто на сьогодні не існує ефективного алгоритму їх розв'язання в загальному випадку. Це й обумовлює використання цих задач в будові багатьох криптографічних примітивів, наприклад, криптосистеми RSA.

- Визначення для підвидів простих чисел

Для побудови, формального опису та оцінки коректності та ефективності тестів на простоту використовуються так звані псевдопрості числа — складені числа, що зберігають ті чи інші властивості простих.

Означення 1. Непарне число p називається *псевдопростим за основою* $a \in \mathbb{N}$, якщо $\gcd(a,p) = 1$ і $a^{p-1} \equiv 1(\mod p)$, де $\gcd(a,b)$ (від англ. *greatest common divisor*) — найбільший спільний дільник чисел a та b.

Означення 2. Непарне число p називається *псевдопростим числом Ойлера за основою* $a \in \mathbb{N}$, якщо $\gcd(a,p) = 1$ та $(\frac{a}{p}) \equiv a^{\frac{p-1}{2}} (\mod p)$, де $(\frac{a}{p})$ — символ Якобі.

Означення 3. Нехай p — непарне число, $p-1 = d \cdot 2^s$, d — непарне. Число p називається *сильним псевдопростим числом за основою* $a \in \mathbb{N}$, якщо $a^d \equiv 1(\mod p)$ або $a^{d \cdot 2^r} \equiv -1(\mod p)$ при якомусь $r : 0 \leq r < s$.

- # Тести Соловея-Штрассена та Міллера-Рабіна (перевірка на простоту)

2.1.1 Імовірнісний тест Соловея-Штрассена

Перевіримо число $p \in \mathbb{N}$. Оберемо деяке число $k \in \mathbb{N}$.

0. Встановлюємо лічильник 0.
1. Вибираємо випадкове число $x \in \mathbb{N}$ з інтервалу $1 < x < p$ (незалежне від раніше обраних x, якщо такі були). За допомогою алгоритму Евкліда знаходимо $\gcd(x,p)$. Якщо $\gcd(x,p) = 1$, то переходимо до кроку 2, якщо $\gcd(x,p) > 1$, то p — складене число, алгоритм завершує свою роботу.
2. Перевіряємо, чи є p псевдопростим Ойлера за основою x. Якщо p виявилось не псевдопростим за основою x, то p — складене число, алгоритм завершує свою роботу; інакше лічильник збільшується на 1.
3. Якщо лічильник менший за k, то переходимо до кроку 1, інакше алгоритм завершує роботу.

У результаті число p буде перевірено тестом Соловея-Штрассена при k різних основах. Якщо p виявиться не псевдопростим хоча б за однією основою, то p складене. Якщо p псевдопросте за всіма цими основами, то вважаємо p простим числом.

Тест Соловея-Штрассена є одностороннім, тобто в ньому відсутні помилки першого роду: якщо p — просте число, то тест завжди покаже, що воно просте. Однак, якщо p — число складене, то для кожної можливої основи тест із імовірністю $\frac{1}{2}$ поверне помилкову відповідь (повідомлення, що p — просте). Використання k різних випадкових основ зменшує помилку тесту до 2^{-k}.

2.1.2 Імовірнісний тест Міллера-Рабіна

Перевіримо число $p \in \mathbb{N}$. Оберемо деяке число $k \in \mathbb{N}$.

0. Знаходимо розклад $p-1 = d \cdot 2^s$ та встановлюємо лічильник 0.
1. Вибираємо випадкове число $x \in \mathbb{N}$ з інтервалу $1 < x < p$ (незалежне від раніше обраних x, якщо такі були). За допомогою алгоритму Евкліда знаходимо $\gcd(x,p)$. Якщо $\gcd(x,p) = 1$, то переходимо до кроку 2, якщо $\gcd(x,p) > 1$, то p — складене число, алгоритм завершує свою роботу.
2. Перевіряємо, чи є p сильно псевдопростим за основою x:
   (а) Якщо $x^d \bmod p = \pm 1$, то p є сильно псевдопростим за основою x.
   (б) Інакше для всіх $r = \overline{1, s-1}$ виконати такі дії:
      i. Обчислити $x_r = x^{d \cdot 2^r} \bmod p$ (тобто $x_r = x_{r-1}^2 \bmod p$).

ii. Якщо $x_r = -1$, то p є сильно псевдопростим за основою x, якщо $x_r = 1$, то p не є сильно псевдопростим за основою x, інакше перейти до наступного значення r.

(в) Якщо за кроки 2а та 2б не було встановлено, що p є сильно псевдопростим за основою x, то p не є сильно псевдопростим за основою x.

3. Якщо p виявилось не сильно псевдопростим за основою x, то p — складене число, алгоритм завершує свою роботу; інакше лічильник збільшується на 1.

4. Якщо лічильник менший за k, то переходимо до кроку 1, інакше алгоритм завершує роботу.

У результаті число p буде перевірено тестом Міллера-Рабіна при k різних основах. Якщо p виявиться не сильно псевдопростим хоча б за однією основою, то p складене. Якщо p сильно псевдопросте за всіма цими основами, то вважаємо p простим числом.

Тест Міллера-Рабіна є одностороннім, тобто в ньому відсутні помилки першого роду: якщо p — просте число, то тест завжди поверне, що воно просте. Однак, якщо p — число складене, то для кожної можливої основи тест із імовірністю $\frac{1}{4}$ поверне помилкову відповідь (повідомлення, що p — просте). Використання k різних випадкових основ зменшує помилку тесту до 4^{-k}.

• Метод пробних ділень

Найпростішим рішенням для виконання факторизації цілого числа є *метод пробних ділень*. Він полягає у перевірці на ділення без остачі числа n почергово на набір дільників з множини простих чисел 2, 3, 5, 7, 9, 11, ..., $[\sqrt{n}]$. На практиці метод пробних ділень використовується як для перевірки числа на простоту (первинний етап), так і для пошуку найменших простих дільників числа в задачі факторизації.

Для використання методу пробних ділень для довільних малих дільників використовують так звану *ознаку подільності Паскаля*. Нехай натуральне число n записане у системі числення з основою B:

$$n = \overline{a_t a_{t-1} ... a_1 a_0} = a_t B^t + a_{t-1} B^{t-1} + ... + a_1 B + a_0,$$

де $0 \leq a_i < B - 1$. Тоді для довільного (малого) m, кандидата в дільники, виконується рівність:

$$n \equiv \sum_{i=0}^{t} a_i r_i (\mathrm{mod}\ m),$$

де $r_i = B^i \bmod m$. Сума в правій частині не перевищує величини tmB, що є значно меншим за оригінальне N. Послідовність r_i не залежить від N та може бути обчислена заздалегідь за рекурентною формулою $r_0 = 1$, $r_{i+1} = r_i \cdot B \bmod m$. Більш того, послідовність r_i є періодичною, що спрощує обчислення.

• Метод ρ-Полларда

ρ-метод Полларда є алгоритмом типу Las-Vegas, що означає, що він не завжди видає якийсь результат, але якщо видає результат, то він точно правильний.

Вхідними даними для алгоритму є n — складене число, яке треба факторизувати, і деяка функція $f : S \to S$, де S — скінченна множина. Зазвичай використовується функція $f(x) = (x^2 + 1) \bmod n$, але зустрічається і використання інших функцій. Виходом алгоритму є або нетривіальний дільник числа n, або повідомлення, що дільник не знайдено.

Для початку роботи алгоритму потрібно вибрати початкове значення $x_0 \in \mathbb{Z}_n$, зазвичай беруть $x_0 = 2$. Далі обчислюється послідовність x_i за таким правилом $x_i = f(x_{i-1})$, тобто $x_1 = f(x_0)$, $x_2 = f(x_1) = f(f(x_0))$ і т.д. Отримана послідовність x_i є скінченною, бо S — скінченна множина, отже, починаючи з якогось номеру, послідовність зациклиться. Далі, для кожного x_k розглядаються різниці $(x_k - x_j)$, де $j < k$, і обраховуються $d = \gcd(x_k - x_j, n)$. Якщо знаходиться таке $d \neq 1$, воно і є нетривіальним дільником n, інакше алгоритм повертає повідомлення, що дільника не знайдено. У випадку, якщо дільника не знайдено, потрібно знову запустити алгоритм з інакшим початковим значенням x_0.

Для зменшення кількості ітерацій, Робертом Флойдом була запропонована модифікація алгоритму. Алгоритм починає роботу з пари (x_0, x_0) та ітеративно обраховує пари (x_1, x_2), (x_2, x_4), (x_3, x_6), ... В кожній ітерації функція $f(x)$ застосовується один раз до першого елементу пари і двічі до другого елементу пари. Потім розглядається різниця елементів обчислених пар. Таким чином, модифікація Флойда ρ-методу Полларда має вигляд:

1. $x_0 = 2, y_0 = 2, d = 1$
2. $x_i = f(x_{i-1}),$
   $y_i = f(f(y_{i-1})),$
   $d = \gcd(x_i - y_i, n).$
3. Якщо $x = y$, зі заданими початковими значеннями алгоритм не знайшов дільника числа n. Повернутися на крок 1 та поміняти початкові значення.
4. Якщо $d \neq 1$, алгоритм закінчує роботу зі знайденим дільником d, інакше повернутися на крок 2.

- ## Метод Померанця

Відмінність методу Померанця від алгоритму Брілхарта-Морісона полягає у способі формування множини чисел, квадрат яких потенційно є $\mathcal{B}$-числами. Основна ідея в тому, щоб відкидати частину чисел, які з великою ймовірністю не є $\mathcal{B}$-числами, до застосування методу пробних ділень до цих чисел. Такий підхід реалізовується етапом *просіювання*.

Хід роботи

Завдання 3:

Варіант 2: вхідне число 3233245835185415 83

Процес факторизації:

```
1-step: Checking if number is prime:
Number is composite

2-step: Bruteforce:


Divisor: 11 Time: 2.18e-05
3-step: rho-Pollard:


Divisor: 2971 Time: 0.0230295
4-step: Quadratic sieve algorithm
```

```
Divisor: -1  Time: 19.8441
Total time:     19.8658
```

Завдання 4:

Також було виміряно час виконання алгоритму для кожного числа. Результати були згруповані у таблицю:

Число	Час виконання (секунди)	Дільник за Поллардом	Час виконання (секунди)	Дільник за Померанцем
3009182572376191	1,4015	-	13,4023	-
1021514194991569	0,8937	-	14,0237	-

4000852962116741	2,1793	-	18,4912	-
15196946347083	0,2738	3	12,8749	89
499664789704823	4,6285	-	15,3894	-
269322119833303	2,7305	-	17,8210	-
679321846483919	3,9832	-	13,1026	-
962673662848 49	4,1981	-	17,6322	962623
613331277926 37	1,9982	-	12,9651	89
248502162840 4193	3,8934	-	13,8270	-

Коментар:

Оскільки метод $\rho - Pollard$-а(з модифікацією) шукає пари виду (x_k, x_{2k}), то зациклювання може відбутися через досить великий проміжок часу (оригінальний варіант – «кожен з кожним» працював би ще довше). Він працює швидше за Померанця, оскільки пари шукаються набагато швидше, чим пошуку можливої лінійної комбінації в алгоритмі Померанця для розв'язку СЛР. Померанець демонструє добру роботу, але все результат залежить від вибору вдалого інтервалу просіювання та кількості маленьких простих дільників.

Завдання 5:

Протестували числа заданих порядків (до максимального, що «їв» комп'ютер, у нашій реалізації)

Числа, порядку p	Міллер-Рабін	Пробні ділення	$\rho - pollard$	Померанець
p = 3	1,9e-7	0,89e-9	3,87e-8	4,04e-5
p = 6	2,4e-6	4,8e-7	2,78e-8	-
p = 9	8,9e-5	5,2e-7	-	5,81e-2
p = 12	4,53e-6	9,4e-1	4,04e-3	8,55e-0
p = 15	4,14e-5	2,0e+7	-	-
p = 18	7,12e-3	-	2,26e-0	12,3e+1

Міні-підсумок:

В загальному, перевірка числа на простоту доволі класна штука, для перевірки на простоту (але треба зважати на ймовірність помилки $\frac{1}{4}$, якщо число є складеним)

Якщо число має відносно маленький простий дільник, то алгоритми пробного ділення та $\rho - pollard$-а доволі швидко його знаходять. Померанець починає ефективно шукати дільники починаючи з тисячі.
Висновки:
Найперше, що хочеться сказати – це про труднощі реалізації лабораторної:

- Найважчим, як на диво, було реалізувати частину з безпосереднім обчисленням ймовірних дільників для Алгоритму Померанця.
- Також, важкою задачею було правильно сформувати факторну базу, та ґрунтуючись вже на ній розв'язати СЛР.

А в цілому, про лабораторку хочеться сказати таке:
Ми підтвердили, обговорені на лекції припущення стосовно ефективності методу $\rho - pollard$-а для чисел, розклад яких, містить невеликі прості дільники.
Роблячи загальний підсумок по факторизації, варто зазначити, що єдиного найкращого алгоритму поки не існує, і для факторизації числа варто обов'язково застосовувати декілька методів.

Додатки:

Увесь код можна знайти за посиланням, на GitHub:
https://github.com/MansteinOrGuderian/NTA-1

Docker enjoyer instruction:

В термінал пишемо: docker run -i mansteinorguderian/ntafirst:latest
(деталі запуску - https://github.com/MansteinOrGuderian/NTA-1/blob/main/Docker%20Instruction.md)