МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский Авиационный Институт»
(Национальный Исследовательский Университет)

Институт: №8 «Информационные технологии и прикладная математика» Кафедра: 806 «Вычислительная математика и программирование»

Лабораторная работа № 3 по курсу «Криптография»

Группа: М8О-306Б-20

Студент: И. П. Попов

Преподаватель: А. В. Борисов

Оценка:

Дата:

Москва, 2022

Факторизация числа

Задача:

Разложить число на нетривиальные сомножители.

Вариант выбрать следующим образом: свое ФИО подать на вход в хеш-функцию, являющуюся стандартом, выход хеш-функции представить в шестнадцатеричном виде и рассматривать младший байт как номер варианта. В отчете привести подробности процесса вычисления номера варианта.

Применение в криптографии

Предполагаемая большая вычислительная сложность задачи факторизации лежит в основе криптостойкости некоторых алгоритмов шифрования с открытым ключом, таких как RSA. Более того, если известен хотя бы один из параметров ключей RSA, то система взламывается однозначно, кроме того, существует множество алгоритмов восстановления всех ключей в системе, обладая какими-то данными.

Общий метод решета числового поля

Общий метод решета числового поля (англ. *general number field sieve*, GNFS) — метод факторизации целых чисел. Является наиболее эффективным алгоритмом факторизации чисел длиной более 110 десятичных знаков. Сложность алгоритма оценивается эвристической формулой

$$\exp\left(\left(\sqrt[3]{\frac{64}{9}}+o(1)\right)(\log n)^{\frac{1}{3}}(\log\log n)^{\frac{2}{3}}\right)=L_n\left[\frac{1}{3},\sqrt[3]{\frac{64}{9}}\right]$$

Метод является обобщением специального метода решета числового поля: тогда как последний позволяет факторизовать числа только некоторого специального вида, общий метод работает на множестве целых чисел, за исключением степеней простых чисел (которые факторизуются тривиально извлечением корней).

Гладкие числа

В теории чисел **гладким числом** называется целое число, все простые делители которого малы. Поскольку понятие «делители малы» может быть истрактовано вольно, чаще всего гладким числом называют такое, чьи простые делители не превосходят 10 (то есть, по сути равны 2, 3, 5 или 7).

Натуральное число называется *B*-**гладким**, если все его простые делители не превосходят *B*.

Число 2000 имеет следующее разложение на множители: $2^4 \times 5^3$. Поэтому 2000 — это 5-гладкое число, а также 6-гладкое число и так далее, но не 4-гладкое.

Суть метода

Метод решета числового поля (как специальный, так и общий) можно представить как усовершенствование более простого метода — метода рационального решета либо метода квадратичного решета. Подобные им алгоритмы требуют нахождения гладких чисел порядка $\sqrt{n}$. Размер этих чисел экспоненциально растёт с ростом n. Метод решета числового поля, в свою очередь, требует нахождения гладких чисел субэкспоненциального относительно n размера. Благодаря тому, что эти числа меньше, вероятность того, что число такого размера окажется гладким, выше, что и является причиной эффективности метода решета числового поля. Для достижения ускорения вычислений в рамках метода проводятся в числовых полях, что усложняет алгоритм, по сравнению с более простым рациональным решетом.

Основные принципы

- Метод факторизации Ферма для факторизации натуральных нечетных чисел n, состоящий в поиске таких целых чисел x и y, что $x^2 - y^2 = n,$ что ведет к разложению $n = (x - y) \cdot (x + y).$
- Нахождение подмножества множества целых чисел, произведение которых — квадрат

- Составление факторной базы: набора $\{-1, p_1, p_2, \ldots, p_n\}$, где p_i — простые числа, такие, что $p_i <= B$ для некоторого B.
- Просеивание выполняется подобно решету Эратосфена (откуда метод и получил своё название). Решетом служат простые числа факторной базы и их степени. При просеивании число не «вычеркивается», а делится на число из решета. Если в результате число оказалось единицей, то оно B-гладкое.
- Основная идея состоит в том, чтобы вместо перебора чисел и проверки, делятся ли их квадраты по модулю n на простые числа из факторной базы, перебираются простые числа из базы и сразу для всех чисел вида $x^2 - n$ проверяется, делятся ли они на это простое число или его степень.

Ход работы

Для вычисление номера своего варианта я подал свое ФИО на вход в хеш-функцию, являющуюся стандартом языка Java, выход хеш-функции представил в шестнадцатеричном виде

```
import java.util.Objects;
public class CR_lab2 {
    private String str;
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
          if (o == null || getClass() != o.getClass()) return
false;
        CR_lab2 cr_lab2 = (CR_lab2) o;
        return Objects.equals(str, cr_lab2.str);
    }
    @Override
    public int hashCode() {
        return Objects.hash(str);
    }
    public void setStr(String str) {
        this.str = str;
    }
    public static void main(String[] args) {
        CR_lab2 cr_lab2 = new CR_lab2();
```

```
        cr_lab2.setStr("Попов Илья Павлович");

        int hash = cr_lab2.hashCode();
        System.out.println(hash);
        System.out.println(Integer.toHexString(hash));
    }
}
```

```
D:\w_dev\jdk-19.0.2\bin\java.exe
"-javaagent:D:\w_dev\IntelliJ                              IDEA
2022.3.2\lib\idea_rt.jar=62545:D:\w_dev\IntelliJ           IDEA
2022.3.2\bin"                                -Dfile.encoding=UTF-8
-Dsun.stdout.encoding=UTF-8       -Dsun.stderr.encoding=UTF-8
-classpath                                        "D:\w_dev\java
projects\untitled\out\production\untitled" CR_lab2

-464930166
e449ba8a
```

Первой строкой вывода я распечатал сам хэш-код моей строки, а второй - этот хэш-код, представленный в 16-ричной системе.
Младшие разряды - 8a => вариант:

n(8a)=47015977796927775654580672918436492909565272168418449913648256023287266549571873540818274437304878516138351974723740391511336230645461001352990692336170037741630246719797239275825718151505109575789330406461323895666813566600944893086865327973774858127589696937850632664380779036097163873642418174442115662253 1

Следующий:

n(8b)=59966062771401421819147908219087595808224410077459847929292207430324158904887839057322172351564624164117789127000788299180261851453288865673628730928442633173837850292292752610852909373999688377541145082539759824267446209842592259157945668974592322820638103363720867612149302326377029995845950331172021680500 7

Прямая факторизация настолько больших чисел невозможна, поэтому была применена следующая хитрость: был найден НОД (число A) чисел из моего варианта и следующего за ним. После этого мое число было поделено на этот самый НОД и частным оказалось второе число(число B).

Числа A и B были проверены на простоту тестом Ферма и оба оказались простыми. Следовательно, они и будут являться ответом задачи.

Код на python:

```
from math import gcd
import random


def bin_pow(a, n):
    res = 1
    while n > 0:
        if n % 2 == 1:
            res = res * a
        a = a * a
        n //= 2
    return res


def is_prime(num, test_count):
    for i in range(test_count):
        rnd = random.randint(1, num - 1)
        if gcd(num, rnd) != 1:
            return False
    return True


myVar =
4701597779692777565458067291843649290956527216841844991364825602328726654957187354081827443730487851613835197472374039151133623064546100135299069233617003774163024671979723927582571815150510957578933040646132389566681356660094489308686532797377485812758969693785063266438077903609716387364241817444211566225 31
nextVar =
5996606277140142181914790821908759580822441007745984792929220743032415890488783905732217235156462416411778912700078829918026185145328886567362
```

```
873092844263317383785029229275261085290937399968837754114508253975982426744620984259225915794566897459232282063810336372086761214930232637702999584595033117202168050070
d1 = gcd(myVar, nextVar)
d2 = myVar // d1
print("Проверка на простоту")
print("Первый делитель:", is_prime(d1, 100))
print("Второй делитель:", is_prime(d2, 100))
print("\nВывод делителей")
print("Первый делитель:", d1)
print("Второй делитель:", d2)
```

Вывод в терминал:

Проверка на простоту

A: True

B: True

Ответ:

A:
217451550740883846561507857729403011932884658895029895125998624235271671431311

B:
216213577860165303078848277059201648559970936790831983491196190195371052058436574856233275059372129066520174348840408295858099836647552424644450913212537878436684499663233086361611676342944363831797531158030216216517458943893791502 1

Выводы

В ходе выполнения данной лабораторной работы я узнал границы работы алгоритмов факторизации и, изучил метод Ферма для проверки числа на простоту, асимптотика которого O(k * log n), где k – число тестов, n – проверяемое число.