Лабораторная работа 6

Разложение чисел на множители

Баранов Иван

2022 Москва

RUDN University, Moscow, Russian Federation

Цель работы

Цель работы

- Ознакомиться с задачей разложения простого числа на множители
- Рассмотреть каноническое представление числа
- Реализовать метод нахождения нетривиальных сомножителей

Описание задачи

Каноническое разложение

Задача разложения на множители - одна из первых задач, использованных для построения криптосистем с открытым ключом.

Задача разложения составного числа на множители формулируется следующим образом:

для данного положительного целого числа n найти его каноническое разложение $n = p_1^{a_1} p_2^{a_2} \ldots p_s^{a_s}$, где p_i - попарно различные простые числа, $a_i >= 1$

Задача нахождения сомножителей

На практике не обязательно находить каноническое разложение числа n.

Достаточно найти его разложение на два нетривиальных сомножителя: $n = pq, 1 <= p <= q < n$.

Далее будем понимать задачу разложения именно в этом смысле. Для её решения воспользуемся р-Методом Полларда (рис. 1) и реализуем его посредством Python (рис. 2)

Описание алгоритма

Описание алгоритма

р–Метод Полларда. Пусть n – нечетное составное число, $S = \{0, 1, \dots, n-1\}$ и $f\colon\ S \to S$ – случайное отображение, обладающее сжимающими свойствами, например $f(x) \equiv x^2 + 1 \ (mod\ n)$. Основная идея метода состоит в следующем. Выбираем случайный элемент $x_0 \in S$ и строим последовательность $x_0, x_1, x_{2,\dots}$, определяемую рекуррентным соотношением

$$x_{i+1} = f(x_i),$$

где $i \geq 0$, до тех пор, пока не найдем такие числа i, j, что $i < j$ и $x_i = x_j$. Поскольку множество S конечно, такие индексы i, j существуют (последовательность «зацикливается»). Последовательность $\{x_i\}$ будет состоять из «хвоста» $x_0, x_1, \dots, x_{i-1}$ длины $O\left(\sqrt{\frac{\pi n}{8}}\right)$ и цикла $x_i = x_j, x_{i+1}, \dots, x_{j-1}$ той же

Figure 1: р-Метод Полларда

Реализация алгоритма

Реализация алгоритма

```python
import math
from random import randint

def pollard(n: int) -> int:
    f = lambda x: (x**2 + 1) % n
    c = randint(0,n-1)
    a = c
    b = c
    while True:
        a = f(a)
        b = f(f(b))
        d = math.gcd(a-b, n)

        if 1 < d < n:
            return d
        elif d == n:
            return None

for i in range(10000, 10100):
    p = pollard(i)
    if p: print(i, p, i//p)
```

Figure 2: Реализация алгоритма

Выводы

Выводы

- Ознакомились с задачей разложения простого числа на множители
- Рассмотрели каноническое представление числа
- Реализовали метод нахождения нетривиальных сомножителей