Отчёт по лабораторной работе №6

Разложение чисел на множители

Бакундукизе Эжид Принц НФИмд-01-21

Содержание

List of Figures

1 Цель работы

Изучение задачи разложения на множители, изучение р-алгоритма Полларда.

2 Теоретические сведения

Разложение на множители — предмет непрерывного исследования в прошлом; и такие же исследования, вероятно, продолжатся в будущем. Разложение на множители играет очень важную роль в безопасности некоторых криптосистем с открытым ключом.

Согласно Основной теореме арифметики любое положительное целое число больше единицы может быть уникально записано в следующей главной форме разложения на множители, где $p_1, p_2, ..., p_k$ — простые числа и $e_1, e_2, ..., e_k$ — положительные целые числа [1].

$$n = p_1^{e_1} * p_2^{e_2} * ... * p_k^{e_k}$$

Поиск эффективных алгоритмов для разложения на множители больших составных чисел ведется давно. К сожалению, совершенный алгоритм для этого пока не найден. Хотя есть несколько алгоритмов, которые могут разложить число на множители, ни один не способен провести разложение достаточно больших чисел в разумное время. Позже мы увидим, что это хорошо для криптографии, потому что современные криптографические системы полагаются на этот факт. В этой секции мы даем несколько простых алгоритмов, которые проводят разложение составного числа. Цель состоит в том, чтобы сделать процесс разложения на множители менее трудоёмким.

В 1974 г. Джон Поллард разработал метод, который находит разложение числа p на простые числа. Метод основан на условии, что p–1 не имеет сомножителя, большего, чем заранее определенное значение B, называемое границей. Алгоритм Полларда показывает, что в этом случае

$$p = GCD(2^{B!} - 1, n)$$

[2]

Сложность. Заметим, что этот метод требует сделать $B–1$ операций возведения в степень $a = a^e modn$. Есть быстрый алгоритм возведения в степень, который выполняет это за $2 * 1og_2 B$ операций. Метод также использует вычисления НОД, который требует n^3 операций. Мы можем сказать, что сложность — так или иначе больше, чем $O(B)$ или $O(2^n)$, где n_b — число битов в B. Другая проблема – этот алгоритм может заканчиваться сигналом об ошибке. Вероятность успеха очень мала, если B имеет значение, не очень близкое к величине $\sqrt{n}$.

2.1 p-алгоритм Полларда

- Вход. Число n, начальное значение c, функция f, обладающая сжимающими свойствами.
- Выход. Нетривиальный делитель числа n.

1. Положить $a = c, b = c$
2. Вычислить $a = f(a)(modn), b = f(b)(modn)$
3. Найти $d = GCD(a - b, n)$
4. Если $1 < d < n$, то положить $p = d$ и результат: p. При $d = n$ результат: ДЕЛИТЕЛЬ НЕ НАЙДЕН. При $d = 1$ вернуться на шаг 2.

3 Выполнение работы

3.1 Реализация алгоритма на языке Python

```
from math import gcd



def f(x, n):
    return pow((pow(x,2)+5),1,n)


def Pol(n, c):
    a = c
    b = c
    d = 1

    while (d==1):


        a = pow(f(a, n),1, n)
        b = pow(f(f(b, n), n),1, n)
        d = gcd(a-b, n)


    if 1< d <n:
        p = d
        return p
```

```
    if d == n:
        return d

n = 1359331
c = 1
p = Pol(n, c)

if  p !=1:
    print("Нетривиальный делитель числа {} - {}".format(n, p))
    print("Разложение на множители числа {} = {}*{}".format(n, p, int(n/p)))


import numpy as np

n = 15
sq_n = np.sqrt(n)

for i in range(1,n+1):
    if n%i == 0 and i >=sq_n:
        p = i
        q = int(n/p)
        print ("Разложение на множители числа {} = {}*{}".format (n,p,q))
        print()
```

3.2 Контрольный пример

Нахождение нетривиального делителя p-методом Полларда (Fig.1)

```
In [24]: n = 1359331
         c = 1
         p = Pol(n, c)

         if  p !=1:
             print("Нетривиальный делитель числа {} - {}".format(n, p))
             print("Разложение на множители числа {} = {}*{}".format(n, p, int(n/p)))

Нетривиальный делитель числа 1359331 - 1181
Разложение на множители числа 1359331 = 1181*1151
```

Figure 3.1: Нахождение нетривиального делителя р-методом Полларда

Разложение числа на простые множители (Fig.2)

```
import numpy as np

n = 15
sq_n = np.sqrt(n)

for i in range(1,n+1):
    if n%i == 0 and i >=sq_n:
        p = i
        q = int(n/p)
        print ("Разложение на множители числа {} = {}*{}".format (n,p,q))
        print()

Разложение на множители числа 15 = 5*3

Разложение на множители числа 15 = 15*1
```

Figure 3.2: Разложение числа на простые множители

4 Выводы

В ходе выполнения данной лабораторной работы мы изучили алгоритмы разложения чисел на простые множители, изучили и реализовали p-метод Полларда для нахождения нетривиального делителя, а также написали алгоритм для разложения числа на простые множители.

Список литературы

1. Алгоритмы тестирования на простоту и факторизации
2. P-метод Полларда