Отчет по лабораторной работе №6

Разложение чисел на множители

Асеинова Елизавета

25 ноября 2023

Содержание

Список иллюстраций

1 Цель работы

Целью данной работы является освоение *р-метода Полларда*, который является одним из алгоритмом разложения составного числа на множители.

2 Задание

1. Изучить алгоритм разложения чисел на множители.
2. Реализовать представленный алгоритм и разложить на множители заданное число.

3 Теоретическое введение

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 $n = p_1^{\alpha_1} p_2^{\alpha_2} ... p_s^{\alpha_s}$, где p_i - попарно различные простые числа, $\alpha_i \geqslant 1$.

На практике не обязательно находить каноническое разложение числа *n*. Достаточно найти его разложение на два *нетривиальных сомножителя*: $n = pq, 1 \leqslant p \leqslant q < n$. Далее будем понимать задачу разложения именно в этом смысле.

p-Метод Полларда. Пусть *n* - нечетное составное число, $S = 0, 1, ..., n-1$ и $f : S \to S$ - 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 $x_0, x_1, x_2, ...$, определяемую рекуррентным соотношением

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

где $i \geqslant 0$, 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”) [2]. Последовательность $\{x_i\}$ будет состоять из “хваста” $x_0, x_1, ..., x_{i-1}$ длины $O\left(\sqrt{\frac{\pi n}{8}}\right)$ и цикла $x_i = x_j, x_{i+1}, ..., x_{j-1}$ той же длины.

4 Выполнение лабораторной работы

Для реализации рассмотренного алгоритма разложения чисел на множители используется среда Google Colab.

1. Запишем алгоритм, реализующий *p-метод Полларда*, с помощью следующей функции:

```
[ ] from math import gcd
    def add_func(x, n):
      return (x**2 + 5) % n
```

```
[ ] def Pollard(n, a, b, d):
      a = add_func(a,n)
      b = add_func(add_func(b, n), n)
      d = gcd(a-b, n)
      if 1 < d < n:
        print(d)
        exit()
      if d == n:
        print("Делитель не найден")
      if d == 1:
        Pollard(n, a, b, d)
```

Реализация метода Полларда

Здесь на вход подается целое число n, начальные значения a и b, вычисляемые через вышеописанную функцию. Необходимо выполнить следующее:

- положить $a \leftarrow c, b \leftarrow c$
- вычислить $a \leftarrow f(a)(mod\, n), b \leftarrow f(b)(mod\, n)$
- найти $d \leftarrow (a - b, n)$
- если $1 < d < n$, то положить $p \leftarrow d$ и результат: p. При $d = n$ результат: "Делитель не найден"; при $d = 1$ вернуться на шаг 2

По итогу при вызове функции мы получим нетривиальный делитель числа n.

2. Проверим корректность работы алгоритма для заданных сведений. Для этого запишем условие примера с помощью следующей функции:

```
#пример
def example():
  n = 1359331
  c = 1
  a = add_func(c, n)
  b = add_func(a, n)
  d = gcd(a-b, n)
  if 1 < d < n:
    print(d)
    exit()
  if d == n:
    pass
  if d == 1:
    Pollard(n,a,b,d)
```

```
[ ] example()
```

```
1181
```

Пример алгоритма

При вызове данной функции видим, что получаем то же число, что было описано в примере. То есть 1181 является нетривиальным делителем числа 1359331.

5 Выводы

В ходе работы мы изучили и реализовали вероятностные алгоритмы проверки чисел на простоту.

6 Список литературы

1. Фороузан Б. А. Криптография и безопасность сетей. - М.: Интернет-Университет Информационных Технологий : БИНОМ. Лаборатория знаний, 2010. - 784 с. [1]

2. Методические материалы курса [2]