Отчёт по лабораторной работе №6

Разложение чисел на множители

Милёхин Александр НПМмд-02-21

Содержание

[1 Цель работы 1](#_Toc95056553)

[2 Теоретические сведения 1](#_Toc95056554)

[2.1 p-алгоритм Полларда 2](#_Toc95056555)

[3 Выполнение работы 2](#_Toc95056556)

[3.1 Реализация алгоритмов на языке Python 2](#_Toc95056557)

[3.2 Контрольный пример 3](#_Toc95056558)

[4 Выводы 4](#_Toc95056559)

[Список литературы 4](#_Toc95056560)

# 1 Цель работы

Изучение задачи разложения на множители, изучение p-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 — простые числа и — положительные целые числа.

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 на простые числа. Метод основан на условии, что не имеет сомножителя, большего, чем заранее определенное значение , называемое границей. Алгоритм Полларда показывает, что в этом случае

Сложность. Заметим, что этот метод требует сделать операций возведения в степень . Есть быстрый алгоритм возведения в степень, который выполняет это за операций. Метод также использует вычисления НОД, который требует операций. Мы можем сказать, что сложность — так или иначе больше, чем или , где — число битов в . Другая проблема – этот алгоритм может заканчиваться сигналом об ошибке. Вероятность успеха очень мала, если имеет значение, не очень близкое к величине .

## 2.1 p-алгоритм Полларда

* Вход. Число , начальное значение , функция , обладающая сжимающими свойствами.
* Выход. Нетривиальный делитель числа .

1. Положить
2. Вычислить
3. Найти
4. Если , то положить и результат: . При результат: “Делитель не найден”. При вернуться на шаг 2.

# 3 Выполнение работы

## 3.1 Реализация алгоритмов на языке Python

from math import gcd  
  
ag = 1  
bg = 1  
  
def f(x, n):  
 return (x\*x+5)%n  
  
def method(n, a, b, d):  
 a = f(a, n)%n  
 b = f(f(b,n), n)%n  
 d = gcd(a-b, n)  
 if 1 < d < n:  
 p = d  
 print(p)  
 exit()  
 if d == n:  
 print("Делитель не найден")  
 if d == 1:  
 global ag  
 ag = b  
 method(n, a, b, d)  
  
def main():  
 n = 1359331  
 c = 1  
 a = c  
 b = c  
 a = f(a, n)%n  
 b = f(a, n)%n  
 d = gcd(a-b, n)  
 if 1 < d < n:  
 p = d  
 print(p)  
 exit()  
 if d == n:  
 pass  
 if d == 1:  
 method(n, a, b, d)  
  
main()

## 3.2 Контрольный пример

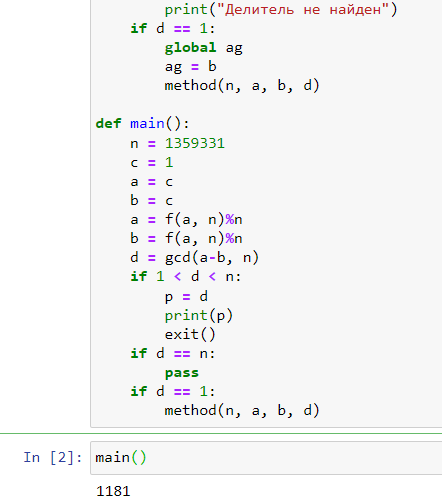


Figure 1: Пример работы алгоритма

Таким образом, число 1181 является нетривиальным делителем числа 1359331.

# 4 Выводы

Я изучил задачу разложения на множители и p-алгоритм Полларда, а также реализовал данный алгоритм программно на языке Python.

# Список литературы

1. [Алгоритмы тестирования на простоту и факторизации](https://habr.com/ru/post/521876/)
2. [P-метод Полларда](https://ru.bmstu.wiki/P-метод_Полларда)