Отчёт по лабораторной работе №6.  
Разложение чисел на множители

Студент: Аронова Юлия Вадимовна, 1032212303

Группа: НФИмд-01-21

Преподаватель: Кулябов Дмитрий Сергеевич,

д-р.ф.-м.н., проф.

Москва 2021

Содержание

[1 Цель работы 1](#_Toc90659116)

[2 Задание 1](#_Toc90659117)

[3 Теоретическое введение 2](#_Toc90659118)

[3.1 Факторизация чисел 2](#_Toc90659119)

[3.2 -метод Полларда 2](#_Toc90659120)

[4 Выполнение лабораторной работы 4](#_Toc90659121)

[4.1 Алгоритм, реализующий -метод Полларда 5](#_Toc90659122)

[5 Выводы 6](#_Toc90659123)

[Список литературы 6](#_Toc90659124)

# 1 Цель работы

Целью данной лабораторной работы является краткое ознакомление с -методом Полларда для нахождения нетривиального делителя целого числа, а также его последующая программная реализация.

# 2 Задание

Рассмотреть и реализовать на языке программирования Python -метод Полларда для нахождения нетривиального делителя целого числа.

# 3 Теоретическое введение

## 3.1 Факторизация чисел

*Факторизацией целого числа* называется его разложение в произведение простых сомножителей [1]. Такое разложение, согласно основной теореме арифметики, всегда существует и является единственным (с точностью до порядка следования множителей).

Мы будем ограничиваться поиском разложения на два (*нетривиальных*) множителя: . Если алгоритм находит такое разложение за арифметических операций, то полное разложение на простые множители будет найдено за арифметических операций, поскольку состоит из произведения не более чем простых чисел [2].

## 3.2 -метод Полларда

Этот метод был разработан Джоном Поллардом в 1975 г. Пусть – число, которое следует разложить. -метод Полларда работает следующим образом [2]:

1 шаг:

Выбрать отображение . Обычно – многочлен степени большей или равной 2, например, .

2 шаг:

Случайно выбрать и вычислять члены рекуррентной последовательности по правилу .

3 шаг:

Для некоторых номеров проверять условие до тех пор, пока не будет найден делитель числа .

Сложность алгоритма оценивается как [3]. Метод строит числовую последовательность, элементы которой образуют цикл, начиная с некоторого номера , что может быть проиллюстрировано расположением чисел в виде греческой буквы (см. Рис. 1).

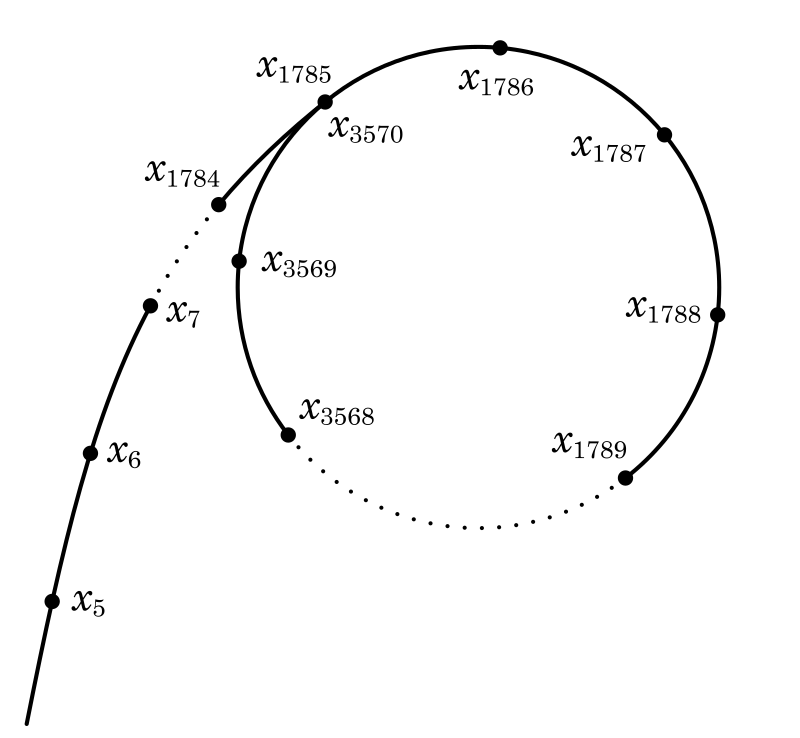


Figure 1: Зацикливание числовой последовательности, получаемой методом -методом Полларда

**Алгоритм 1. Алгоритм, реализующий -метод Полларда**

*Вход.* Число , начальное значение , функция , обладающая сжимающими свойствами.

*Выход.* Нетривиальный делитель числа .

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

Пример 1.

Найдём -методом Полларда нетривиальный делитель числа . Положим . Работа алгоритма проиллюстрирована в Таблице 1.

Table 1: Пример применения -метода Полларда для числа

| i | a | b | d |
| --- | --- | --- | --- |
| 0 | 1 | 1 | - |
| 1 | 6 | 41 | 1 |
| 2 | 41 | 123,939 | 1 |
| 3 | 1,686 | 391,594 | 1 |
| 4 | 123,939 | 438,157 | 1 |
| 5 | 435,426 | 582,738 | 1 |
| 6 | 391,594 | 1,144,026 | 1 |
| 7 | 1,090,062 | 885,749 | 1,181 |
| - | Ответ: | 1,181 |  |

Пример 2.

Повторим процедуру для числа при и (см. Табл. 2) или (см. Табл. 3).

Table 2: Пример применения -метода Полларда для числа (1)

| i | a | b | d |
| --- | --- | --- | --- |
| 0 | 2 | 2 | - |
| 1 | 5 | 26 | 1 |
| 2 | 26 | 7,474 | 1 |
| 3 | 677 | 871 | 97 |
| - | Ответ: | 97 |  |

Table 3: Пример применения -метода Полларда для числа (2)

| i | a | b | d |
| --- | --- | --- | --- |
| 0 | 2 | 2 | - |
| 1 | 7 | 52 | 1 |
| 2 | 52 | 1,442 | 1 |
| 3 | 2,707 | 778 | 1 |
| 4 | 1,442 | 3,932 | 83 |
| - | Ответ: | 83 |  |

# 4 Выполнение лабораторной работы

Реализуем описанный выше алгоритм на языке **Python** в среде Jupyter Notebook. Для работы нам понадобится функция нахождения наибольшего общего делителя. Возьмем функцию, реализующую алгоритм Евклида, реализованную в рамках 4-ой лабораторной работы:

def euclidean\_algorithm(a, b):  
 """  
 Находит НОД чисел a и b с помощью алгоритма Евклида  
 """  
 (a, b) = (abs(int(a)), abs(int(b)))  
  
 if b > a:  
 (a, b) = (b, a)  
  
 r = [a, b] # шаг 1; задаем r0 и r1  
  
 # шаги 2-3  
 while r[1] != 0:  
 (r[0], r[1]) = (r[1], r[0] % r[1])  
  
 return r[0] # шаг 4

## 4.1 Алгоритм, реализующий -метод Полларда

Создадим функцию pollard\_rho\_method(n, f, c) следующего вида:

def pollard\_rho\_method(n, f, c = 1):  
 """  
 Находит нетривиальный делитель числа n ро-методом Полларда  
 на основе начального значения c и сжимающей функции f  
 """  
 a = c; b = c # шаг 1  
  
 while True:  
 x = a #  
 a = eval(f) % n #  
 # шаг 2  
 x = b #  
 x = eval(f) #  
 b = eval(f) % n #  
  
 d = euclidean\_algorithm(abs(a - b), n) # шаг 3  
  
 if d > 1 and d < n: #  
 return d #  
 # шаг 4  
 if d == n: #  
 print("Делитель не найден") #  
 return 0 #

Теперь с помощью данной функции найдём нетривиальные делители некоторых чисел (см. Рис. 2).

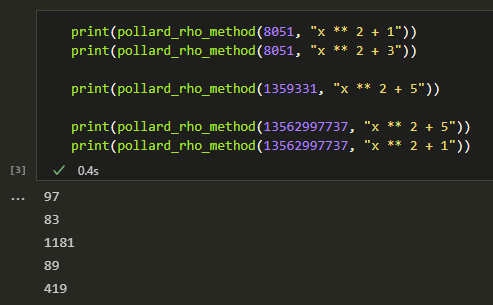


Figure 2: Примеры нахождения нетривиальных делителей чисел посредством программной реализации -метода Полларда

# 5 Выводы

Таким образом, была достигнута цель, поставленная в начале лабораторной работы: было проведено краткое знакомство с алгоритмом, реализующим -метод Полларда для нахождения нетривиального делителя целого числа, после чего алгоритм был успешно реализован на языке программирования **Python**.

# Список литературы

1. Ишмухаметов Ш.Т. Методы факторизации натуральных чисел: учебное пособие. Казань: Казанский университет, 2011. С. 190.

2. Василенко О.Н. Теоретико-числовые алгоритмы в криптографии. Москва: МЦНМО, 2003.

3. Википедия. [Ро-алгоритм Полларда — Википедия, свободная энциклопедия](\url%7bhttps://ru.wikipedia.org/?curid=259057&oldid=116665965%7d). 2021.