Шаблон отчёта по лабораторной работе №6

Разложение чисел на множители

Коне Сирики

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Задание

Рассмотреть и реализовать на языке программирования Python ρ-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: $n = ab, 1 < a \leq b < n$. Если алгоритм находит такое разложение за $O(f(n))$ арифметических операций, то полное разложение n на простые множители будет найдено за $O(f(n) \log n)$ арифметических операций, поскольку n состоит из произведения не более чем $\log_2 n$ простых чисел [2].

3.2 ρ-метод Полларда

Этот метод был разработан Джоном Поллардом в 1975 г. Пусть $n \in \mathbb{N}$ – число, которое следует разложить. ρ-метод Полларда работает следующим образом [2]:

1 шаг: Выбрать отображение $f : \mathbb{Z}_n \to \mathbb{Z}_n$. Обычно $f(x)$ – многочлен степени большей или равной 2, например, $f(x) = x^2 + 1$.

2 шаг: Случайно выбрать $x_0 \in \mathbb{Z}_n$ и вычислять члены рекуррентной последовательности $x_0, x_1, x_2, \ldots$ по правилу $x_i \equiv f(x_{i-1}) \pmod{n}$.

3 шаг: Для некоторых номеров j, k проверять условие $1 < \text{НОД}(x_j - x_k, n) < n$ до тех пор, пока не будет найден делитель числа n.

Сложность алгоритма оценивается как $O(n^{1/4})$ [3]. Метод строит числовую последовательность, элементы которой образуют цикл, начиная с некоторого номера n, что может быть проиллюстрировано расположением чисел в виде греческой буквы ρ (см. Рис. 3.1).

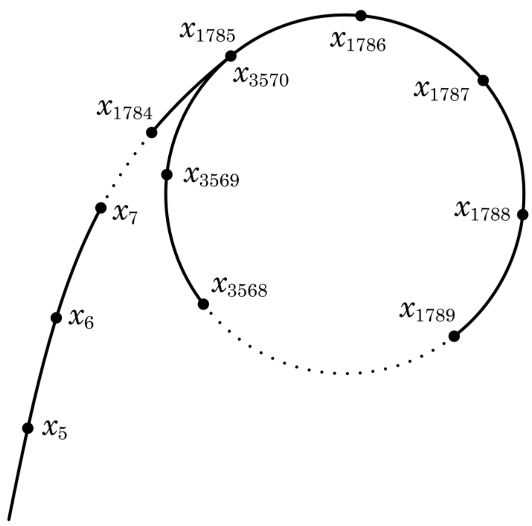

Рис. 3.1: Зацикливание числовой последовательности, получаемой методом ρ-методом Полларда

Алгоритм 1. Алгоритм, реализующий ρ-метод Полларда

Вход. Число n, начальное значение c, функция f, обладающая сжимающими свойствами.

Выход. Нетривиальный делитель числа n.

1. Положить $a \leftarrow c, b \leftarrow c$.
2. Вычислить $a \leftarrow f(a) \pmod n, b \leftarrow f(f(b)) \pmod n$.
3. Найти $d \leftarrow \text{НОД}(a - b, n)$.
4. При $1 < d < n$ положить $p \leftarrow d$ и результат: d. При $d = n$ результат: “Делитель не найден”. При $d = 1$ вернуться на шаг 2.

Пример 1. Найдём ρ-методом Полларда нетривиальный делитель числа $n = 1359331$. Положим $c = 1$, $f(x) = x^2 + 5 \pmod{n}$. Работа алгоритма проиллюстрирована в Таблице 3.1.

Таблица 3.1: Пример применения ρ-метода Полларда для числа $1,359,331$

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 $n = 8051$ при $c = 2$ и $f(x) = x^2 + 1$ (см. Табл. 3.2) или $f(x) = x^2 + 3$ (см. Табл. 3.3).

Таблица 3.2: Пример применения ρ-метода Полларда для числа $8,051$ (1)

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

Таблица 3.3: Пример применения ρ-метода Полларда для числа $8,051$ (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 ρ-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. 4.1).

```python
print(pollard_rho_method(8051, "x ** 2 + 1"))
print(pollard_rho_method(8051, "x ** 2 + 3"))

print(pollard_rho_method(1359331, "x ** 2 + 5"))

print(pollard_rho_method(13562997737, "x ** 2 + 5"))
print(pollard_rho_method(13562997737, "x ** 2 + 1"))
```

```
[3]  ✓ 0.4s
...  97
     83
     1181
     89
     419
```

Рис. 4.1: Примеры нахождения нетривиальных делителей чисел посредством программной реализации ρ-метода Полларда

5 Выводы

Таким образом, была достигнута цель, поставленная в начале лабораторной работы: было проведено краткое знакомство с алгоритмом, реализующим ρ-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. 2021.