# Лабораторна работа № 7

**Дисректное логарифмирование в конечном поле**

Покрас Илья Михайлович

# Содержание

# Список иллюстраций

# 1 Цель работы

Реализовать алгоритм дискретного логарифмирования программно

# 2 Задание

Реализовать $\rho$-метод Полларда для задач дискретного логарифмирования.

# 3 Теоретическое введение

$\rho$-Метод Полларда для дискретного логарифмирования — алгоритм дискретного логарифмирования в кольце вычетов по простому модулю, имеющий экспоненциальную сложность. Предложен британским математиком Джоном Поллардом в 1978 году, основные идеи алгоритма очень похожи на идеи $\rho$-алгоритма Полларда для факторизации чисел. Данный метод рассматривается для группы ненулевых вычетов по модулю p.

Постановка задачи:

Для заданного простого числа p и двух целых чисел a и b требуется найти целое число x удовлетворяющее сравнению: $a^x \equiv b(mod p)$

# 4 Выполнение лабораторной работы

Я создал функцию $\rho$-метод Полларда с входными параметрами a, b, p, v1, v2, u1, u2 (u, v могут быть как определнными числами, так и случайными значениями). Данная функция рассчитывает c и d, а так же их логарифмы. Далее вычисляется x, удовлетворяющее условия сравнения, что и является результатом (рис. 4.1).

```
function pollards_rho_log(p, a, b, u1, v1, u2, v2)


    c = (a^u1 % p) * (b^v1 % p) % p
    d = c

    println("c: $c - $u1 + $v1 x")
    println("d: $d - $u2 + $v2 x")

    u1, v1 = calc_func_uv(c, p, u1, v1)
    c = calc_func_x(c, p, a, b)

    u2, v2 = calc_func_uv(d, p, u2, v2)
    d = calc_func_x(d, p, a, b)

    u2, v2 = calc_func_uv(d, p, u2, v2)
    d = calc_func_x(d, p, a, b)

    println("c: $c - $u1 + $v1 x")
    println("d: $d - $u2 + $v2 x")

    while c != d

        u1, v1 = calc_func_uv(c, p, u1, v1)
        c = calc_func_x(c, p, a, b)

        u2, v2 = calc_func_uv(d, p, u2, v2)
        d = calc_func_x(d, p, a, b)

        u2, v2 = calc_func_uv(d, p, u2, v2)
        d = calc_func_x(d, p, a, b)
        println("c: $c - $u1 + $v1 x")
        println("d: $d - $u2 + $v2 x")
    end

    x = 1

    while mod((v1-v2)*x, p ÷ 2) != mod((u2-u1), p ÷ 2)
        x+=1
    end

    println("x = $x(mod$(p÷ 2))")
end
```

Рис. 4.1: Функция $\rho$-метода для дискретного логарифмирования

Мною были созданы функции для расчета функции c и d, а также u и v в зависимости от того, к какому множеству относится проверяемое значение (рис. 4.2).

```
function calc_func_x(x, p, a, b)
    if x < p ÷ 2
        return mod(a*x, p)
    else
        return mod(b*x, p)
    end
end

function calc_func_uv(c, p, u, v)
    if c < p ÷ 2
        u=u+1
    else
        v=v+1
    end
    return u, v
end
```

Рис. 4.2: Функции расчета

Далее были инциализированы переменные и вызвана функция с данными, взятыми из примера для большей наглядности (рис. 4.3).

```
p = 107
a = 10
b = 64

pollards_rho_log(p, a, b, 2, 2, 2, 2)
```

Рис. 4.3: Инициализация переменных и вызов функции

И получил следующие значения (рис. 4.4):

```
c: 4 - 2 + 2 x
d: 4 - 2 + 2 x
c: 40 - 3 + 2 x
d: 79 - 4 + 2 x
c: 79 - 4 + 2 x
d: 56 - 5 + 3 x
c: 27 - 4 + 3 x
d: 75 - 5 + 5 x
c: 56 - 5 + 3 x
d: 3 - 5 + 7 x
c: 53 - 5 + 4 x
d: 86 - 7 + 7 x
c: 75 - 5 + 5 x
d: 42 - 8 + 8 x
c: 92 - 5 + 6 x
d: 23 - 9 + 9 x
c: 3 - 5 + 7 x
d: 53 - 11 + 9 x
c: 30 - 6 + 7 x
d: 92 - 11 + 11 x
c: 86 - 7 + 7 x
d: 30 - 12 + 12 x
c: 47 - 7 + 8 x
d: 47 - 13 + 13 x
x = 20(mod53)
```

Рис. 4.4: Результат выполнения кода

# 5 Выводы

Я реализовал $\rho$-метод Полларда для задач дискретного логарифмирования.

# Список Литературы


1. Julia - Control Flow
2. Julia - Mathematical Operations
3. Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone - Handbook of Applied Cryptography