

Лабораторная работа №7

Тема: Дискретное логарифмирование в конечном поле

Выполнила: Исламова Сания Маратовна

Группа: НПИмд-01-24

Студ.билет: 1132249576

Задача лабораторной работы:

Реализовать рассмотренный алгоритм программно: Алгоритм, реализующий р-метод Полларда для задач дискретного логарифмирования

Описание хода выполнения лабораторной работы:

```
# Заголовок программы
println("р-метод Полларда для дискретного логарифмирования")
# Бесконечный цикл для многократного использования программы
while true
    # Запрос ввода данных от пользователя
    println("\nВведите p a b r через пробел (или 'выход' для завершения):")
    # Чтение введенной строки с клавиатуры
    input = readline()
    # Проверка команды выхода из программы
    input == "выход" && break
    # Блок обработки ошибок ввода
    try
        # Разделение строки на части и преобразование в BigInt
        p,a,b,r = parse.(BigInt, split(input))
        # Определение функции р-метода Полларда
        function ρ(p,a,b,r)
            # Инициализация: случайные u, v из [0, r-1]
            u,v = rand(0:r-1,2)
            # Вычисление начальной точки c = a^u * b^v mod p
            c = powermod(a,u,p)*powermod(b,v,p)%p
            # Черепаха и заяц: начальные точки одинаковы
            d = c
            # Инициализация логарифмов: Log(c) = u + v*x, Log(d) = u + v*x
            α1,β1,α2,β2 = u,v,u,v
            # Цикл поиска коллизии (метод Флойда)
            while (c = (c<p÷2 ? a*c : b*c)%p) != d
                # Обновление d (два шага)
                (d = (d<p÷2 ? a*d : b*d)%p; d = (d<p÷2 ? a*d : b*d)%p)
                # Обновление логарифмов для c (один шаг)
                α1,β1 = (α1+(c<p÷2))%r, (β1+(c≥p÷2))%r
                # Обновление логарифмов для d (два шага)
                α2,β2 = (α2+2(d<p÷2))%r, (β2+2(d≥p÷2))%r
            end
            # После нахождения коллизии: решение уравнения
            # Находим коэффициенты (β1-β2)*x ≡ (α2-α1) (mod r)
```

```

g,x,_ = gcdx((β1-β2)%r, r)
# Проверка разрешимости и возврат решения
(Δα=(α2-α1)%g ≠ 0 ? nothing : (x*Δα÷g)%r
end
# Вызов функции и получение результата
x = p(p,a,b,r)
# Вывод результата
println(x==nothing ? "Нет решений" : "x = $x")
# Обработка ошибок при некорректном вводе
catch
    println("Ошибка: введите 4 числа или 'выход'")
end
end
# Сообщение о завершении программы
println("Программа завершена")

```

Результат реализации рассмотренного алгоритма

```

Lab7.jl
C: > Users > 4eka0 > Downloads > Lab7.jl > ...
1 #Лабораторная работа №7
2 #Тема: Дискретное логарифмирование в конечном поле
3 #Выполнила: Исламова Сания
4 #Группа НПИмд-01-24
5
6 # Заголовок программы
7 println("[P]-метод Полларда для дискретного логарифмирования")
8 # Бесконечный цикл для многократного использования программы
9 while true
10     # Запрос ввода данных от пользователя
11     println("Введите p a b r через пробел (или 'выход' для завершения):")
12     # Чтение введенной строки с клавиатуры
13     input = readline()
14     # Проверка команды выхода из программы
15     input == "выход" && break
16     # Блок обработки ошибок ввода
17     try
18         # Разделение строки на части и преобразование в BigInt
19         p,a,b,r = parse.(BigInt, split(input))
20         # Определение функции p-метода Полларда
21         function P(p,a,b,r)
22             # Инициализация: случайные u, v из [0, r-1]
23             u,v = rand(0:r-1,2)
24             # Вычисление начальной точки c = a^u * b^v mod p
25             c = powermod(a,u,p)*powermod(b,v,p)%p
26             # Черепаха и заяц: начальные точки одинаковы
27             d = c
28             # Инициализация логарифмов: log(c) = u + v*x, log(d) = u + v*x
29             β1,β2,β3 = u,v,u,v
30             # Цикл поиска коллизии (метод Флойда)
31             while (c == (c<p? a*c : b*c)%p) != (d == (d<p? a*d : b*d)%p)
32                 # Обновление d (два шага)
33                 (d = (d<p? a*d : b*d)%p; d = (d<p? a*d : b*d)%p)
34                 # Обновление логарифмов для c (один шаг)
35                 β1,β1 = (β1+(c*p+2))%r, (β1+(c*p+2))%r
36                 # Обновление логарифмов для d (два шага)
37                 β2,β2 = (β2+2(d*p+2))%r, (β2+2(d*p+2))%r
38             end
39             # После нахождения коллизии: решение уравнения
40             # Находим коэффициенты (β1-β2)*x = (α2-α1) (mod r)
41             g,x,_ = gcdx((β1-β2)%r, r)
42             # Проверка разрешимости и возврат решения
43             (Δα=(β2-β1)%r)≠0 ? nothing : (x*Δα÷g)%r
44         end
45         # Вызов функции и получение результата
46         x = P(p,a,b,r)
47         # Вывод результата
48         println(x==nothing ? "Нет решений" : "x = $x")
49         # Обработка ошибок при некорректном вводе
50         catch
51             println("Ошибка: введите 4 числа или 'выход'")
52         end
53     end
54     # Сообщение о завершении программы
55     println("Программа завершена")
56
57

```

[ПРОБЛЕМЫ](#) [ВЫХОДНЫЕ ДАННЫЕ](#) [КОНСОЛЬ ОТЛАДКИ](#) [TERMINAL](#) TERMINAL [ПОРТЫ](#)

p-метод Полларда для дискретного логарифмирования

Введите p a b r через пробел (или 'выход' для завершения):

julia> 1 2 3 4

1 2 3 4

x = 0

Введите p a b r через пробел (или 'выход' для завершения):

45 67 102 15

x = -2

Введите p a b r через пробел (или 'выход' для завершения):

ВыХод

Ошибка: введите 4 числа или 'выход'

Введите p a b r через пробел (или 'выход' для завершения):

выход

Программа завершена