

Лабораторная работа 7

р-метод Полларда.

```
Терминал  Оболочка  Правка  Вид  Окно  Справка
elizaveta — julia — julia — 210x56

Last login: Sat Dec 6 18:04:15 on ttys001
elizaveta@MacBook-Pro-Elizaveta: ~ % /Applications/Julia-1.12.app/Contents/Resources/julia/bin/julia ; exit;

Documentation: https://docs.julialang.org
Type "?>" for help, "]"? for Pkg help.
Version 1.12.1 (2025-10-17)
Official https://julialang.org release

julia> """
Р-метод Полларда для задачи дискретного логарифмирования в конечном поле GF(p)
Решает уравнение: a^x = b (mod p)
"""
function pollard_rho_discrete_log(a, b, p, max_iterations=1000)
    # Определяем порядок r (в данном случае p-1, так как p простое)
    # В общем случае нужно знать порядок элемента a
    r = p - 1

    # Функция f(c) с ветвлением
    function f(c)
        if c < p + 2
            return (a * c) % p
        else
            return (b * c) % p
        end
    end

    # Функция для обновления логарифмов
    # Логарифм представлен как (α, β) где log = a + βx
    function update_log(log_tuple, c_next, c_current)
        α, β = log_tuple

        if c_current < p + 2
            # f(c) = a*c → log(f(c)) = log(c) + 1
            return (α + 1, β)
        else
            # f(c) = b*c = a^x*c → log(f(c)) = log(c) + x
            return (α, β + 1)
        end
    end

    # Инициализация
    # Начинаем с c = a^u * b^v, где u, v случайные
    u = rand(0:r-1)
    v = rand(0:r-1)

    c = powermod(a, u, p) * powermod(b, v, p) % p
    d = c

    # Логарифмы: log_c = α_c + β_c * x
    log_c = (u, v) # log(c) = u + v*x
    log_d = (u, v) # log(d) = u + v*x

    println("Начальные значения:")
end
```

```
Терминал  Оболочка  Правка  Вид  Окно  Справка
elizaveta — julia — julia — 210x56

# Инициализация
# Начинаем с c = a^u * b^v, где u, v случайные
u = rand(0:r-1)
v = rand(0:r-1)

c = powermod(a, u, p) * powermod(b, v, p) % p
d = c

# Логарифмы: log_c = α_c + β_c * x
log_c = (u, v) # log(c) = u + v*x
log_d = (u, v) # log(d) = u + v*x

println("Начальные значения:")
println("c = $c, log(c) = $(log_c[1]) + $(log_c[2])*x")
println("d = $d, log(d) = $(log_d[1]) + $(log_d[2])*x")
println()

# Основной цикл
for i in 1:max_iterations
    # Обновляем c (медленный указатель)
    old_c = c
    c = f(c)
    log_c = update_log(log_c, c, old_c)

    # Обновляем d (быстрый указатель — два шага)
    old_d = d
    d = f(d)
    log_d = update_log(log_d, d, old_d)

    old_d2 = d
    d = f(d)
    log_d = update_log(log_d, d, old_d2)

    # Вывод для отладки
    if i <= 10
        println("Шаг $i:")
        println("c = $c, log(c) = $(log_c[1]) + $(log_c[2])*x")
        println("d = $d, log(d) = $(log_d[1]) + $(log_d[2])*x")
    end

    # Проверка на столкновение
    if c == d
        println("Найдено столкновение на шаге $i!")
        println("c = d = $c")
        println("log(c) = $(log_c[1]) + $(log_c[2])*x")
        println("log(d) = $(log_d[1]) + $(log_d[2])*x")

        # Решаем уравнение: α_c + β_c*x = α_d + β_d*x (mod r)
        α_c, β_c = log_c
        α_d, β_d = log_d

        # Уравнение: (β_c - β_d)*x = (α_d - α_c) (mod r)
        coeff = (β_c - β_d) % r
        const_term = (α_d - α_c) % r

        if coeff == 0
            println("Коэффициент равен 0, что означает, что уравнение не имеет решения в данном контексте.")
        else
            # Решение уравнения
            x = (const_term * invmod(coeff, r)) % r
            println("Найдено решение: x = $x")
        end
    end
end
```

```
Терминал Оболочка Правка Вид Окно Справка
elizaveta — julia — julia — 210x56

println("c = d = $c")
println("log(c) = $(log_c[1]) + $(log_c[2])*i")
println("log(d) = $(log_d[1]) + $(log_d[2])*i")

# Решаем уравнение: a_c + β_c*x = a_d + β_d*x (mod r)
a_c, β_c = log_c
a_d, β_d = log_d

# Уравнение: (β_c - β_d)*x = (a_d - a_c) (mod r)
coeff = (β_c - β_d) % r
const_term = (a_d - a_c) % r

if coeff == 0
    if const_term == 0
        println("Уравнение имеет бесконечно много решений")
        return nothing
    else
        println("Уравнение не имеет решений")
        return nothing
    end
end

# Находим обратный коэффициент по модулю r
try
    coeff_inv = invmod(coeff, r)
    x = (const_term * coeff_inv) % r

    # Проверяем решение
    if powermod(a, x, p) == b % p
        println("Найдено решение: x = $x")
        println("Проверка: $a*$x mod $p = $(powermod(a, x, p))")
        println("Ожидалось: $b")
        return x
    else
        # Проверяем x + k*r
        for k in 0:div(r, 10) # Проверяем несколько периодов
            x_test = (x + k*r) % (p-1)
            if powermod(a, x_test, p) == b % p
                println("Найдено решение: x = $x_test")
                return x_test
            end
        end
        println("Найденное x не удовлетворяет уравнению")
    end
catch e
    println("Невозможно найти обратный элемент для $coeff mod $r")
    return nothing
end

println("Достигнуто максимальное число итераций ($max_iterations)")
return nothing
end
pollard_rho_discrete_log
```

р-метод Полларда.

```
Терминал  Оболочка  Правка  Вид  Окно  Справка
elizaveta — julia — julia — 210x56

end
end

println("Достигнуто максимальное число итераций ($max_iterations)")
return nothing
end
pollard_rho_discrete_log
julia> """
Функция для демонстрации работы алгоритма на примере из лабораторной
"""
function demo_example()
    println("==")
    println("    ПРИМЕР ИЗ ЛАБОРАТОРНОЙ РАБОТЫ")
    println("==")

    # Параметры из примера
    p = 107
    a = 10
    b = 64

    println("Решаем уравнение: $a*x = $b (mod $p)")
    println("Ожидаемое решение: x = 20 (mod 53)")
    println()

    # Запускаем алгоритм
    result = pollard_rho_discrete_log(a, b, p)

    if result != nothing
        println("\n" * "==" * " ")
        println("    ПРОВЕРКА РЕШЕНИЯ")
        println("==")

        # Проверяем несколько значений
        for x_test in [result, result + 53, result + 106]
            lhs = powermod(a, x_test, p)
            println("$a*$x_test mod $p = $lhs")
            if lhs == b
                println("✅ Это правильное решение!")
            end
        end
    end
end
end
return result
end
demo_example
julia> """
Функция для тестирования алгоритма на различных примерах
"""
function test_various_examples()
    println("\n" * "==" * " ")
    println("    ТЕСТИРОВАНИЕ НА РАЗЛИЧНЫХ ПРИМЕРАХ")
    println("==")
end
```

```
Терминал  Оболочка  Правка  Вид  Окно  Справка
elizaveta — julia — julia — 210x56

end
end

println("Достигнуто максимальное число итераций ($max_iterations)")
return nothing
end
pollard_rho_discrete_log
julia> """
Функция для демонстрации работы алгоритма на примере из лабораторной
"""
function demo_example()
    println("==")
    println("    ПРИМЕР ИЗ ЛАБОРАТОРНОЙ РАБОТЫ")
    println("==")

    # Параметры из примера
    p = 107
    a = 10
    b = 64

    println("Решаем уравнение: $a*x = $b (mod $p)")
    println("Ожидаемое решение: x = 20 (mod 53)")
    println()

    # Запускаем алгоритм
    result = pollard_rho_discrete_log(a, b, p)

    if result != nothing
        println("\n" * "==" * " ")
        println("    ПРОВЕРКА РЕШЕНИЯ")
        println("==")

        # Проверяем несколько значений
        for x_test in [result, result + 53, result + 106]
            lhs = powermod(a, x_test, p)
            println("$a*$x_test mod $p = $lhs")
            if lhs == b
                println("✅ Это правильное решение!")
            end
        end
    end
end
end
return result
end
demo_example
julia> """
Функция для тестирования алгоритма на различных примерах
"""
function test_various_examples()
    println("\n" * "==" * " ")
    println("    ТЕСТИРОВАНИЕ НА РАЗЛИЧНЫХ ПРИМЕРАХ")
    println("==")
end
```

Результат работы.

```
Терминал  Оболочка  Правка  Вид  Окно  Справка
elizaveta — julia — julia — 210x56

julia> """
Функция для ввода параметров от пользователя
"""
function user_input_mode()
    println("==*60)
    println(" РЕЖИМ РУЧНОГО ВВОДА ПАРАМЕТРОВ")
    println("==*60)

    print("Введите простое число p: ")
    p = parse{Int, readline()}

    print("Введите основание a: ")
    a = parse{Int, readline()}

    print("Введите значение b: ")
    b = parse{Int, readline()}

    println("\nРешаем уравнение: $a*x = $b (mod $p)")

    result = pollard_rho_discrete_log(a, b, p)

    if result != nothing
        println("\n✅ Решение: x = $result")
        println("Проверка: $a*$result mod $p = $(powermod(a, result, p))")
    else
        println("\n❌ Решение не найдено")
    end

    return result
end
user_input_mode

julia> """
Основная функция для запуска лабораторной работы
"""
function main()
    println("ЛАБОРАТОРНАЯ РАБОТА №7")
    println("Дискретное логарифмирование в конечном поле")
    println("Реализация p-метода Полларда")
    println()

    while true
        println("\n" * "="*60)
        println(" МЕНЮ")
        println("==*60)
        println("1. Запустить пример из лабораторной работы")
        println("2. Протестировать на различных примерах")
        println("3. Ввести параметры вручную")
        println("4. Выход")
        print("\nВыберите вариант: ")

        choice = readline()

        if choice == "1"
            demo_example()
        elseif choice == "2"
```

```
Терминал  Оболочка  Правка  Вид  Окно  Справка
elizaveta — julia — julia — 210x56

        println("\n✅ Решение: x = $result")
        println("Проверка: $a*$result mod $p = $(powermod(a, result, p))")
    else
        println("\n❌ Решение не найдено")
    end

    return result
end
user_input_mode

julia> """
Основная функция для запуска лабораторной работы
"""
function main()
    println("ЛАБОРАТОРНАЯ РАБОТА №7")
    println("Дискретное логарифмирование в конечном поле")
    println("Реализация p-метода Полларда")
    println()

    while true
        println("\n" * "="*60)
        println(" МЕНЮ")
        println("==*60)
        println("1. Запустить пример из лабораторной работы")
        println("2. Протестировать на различных примерах")
        println("3. Ввести параметры вручную")
        println("4. Выход")
        print("\nВыберите вариант: ")

        choice = readline()

        if choice == "1"
            demo_example()
        elseif choice == "2"
            test_various_examples()
        elseif choice == "3"
            user_input_mode()
        elseif choice == "4"
            println("Выход из программы")
            break
        else
            println("Неверный выбор. Попробуйте снова.")
        end

        println("\nНажмите Enter для продолжения...")
        readline()
    end
end
main

julia> # Запуск программы
if abspath(PROGRAM_FILE) == @__FILE__
    main()
end

julia>
```

Выводы

- Познакомилась с дискретным логарифмированием в конечном поле
- Реализовала алгоритм р-метод Полларда

{.standout}

Спасибо за внимание!