

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5

дисциплина: Математические основы защиты информации и
информационной безопасности

Студент: Пиняева Анна Андреевна

Группа: НПИмд-01-24

МОСКВА

2025

Теоретическое введение

1. Алгоритм, реализующий тест Ферма.

Вход. Нечетное целое число $n \geq 5$. Выход. «Число n , вероятно, простое» или «Число n составное». 1. Выбрать случайное целое число a , $2 \leq a \leq n-2$. 2. Вычислить $r \leftarrow a^{(n-1)} \pmod{n}$. 3. При $r = 1$ результат: «Число n , вероятно, простое». В противном случае результат: «Число n составное».

2. Алгоритм вычисления символа Якоби.

Вход. Нечетное целое число $n \geq 3$, целое число a , $0 \leq a < n$. Выход. Символ Якоби (4). 1. Положить $g \leftarrow 1$. 2. При $a = 0$ результат: 0. 3. При $a = 1$ результат: g . 4. Представить a в виде $a = 2^k \cdot a_1$, где число a_1 нечетное. 5. При четном k положить $s \leftarrow 1$, при нечетном k положить $s \leftarrow 1$, если $n \equiv +1 \pmod{8}$; положить $s \leftarrow -1$, если $n \equiv \pm 3 \pmod{8}$. 6. При $a_1 = 1$ результат: $g \cdot s$. 7. Если $n \equiv 3 \pmod{4}$ и $a_1 \equiv 3 \pmod{4}$, то $s \leftarrow -s$. 8. Положить $a \leftarrow n \pmod{a_1}$, $n \leftarrow a_1$, $g \leftarrow g \cdot s$ и вернуться на шаг 2.

3. Алгоритм, реализующий тест Соловья-Штрассена.

Вход. Нечетное целое число $n \geq 5$. Выход. «Число n , вероятно, простое» или «Число n составное». 1. Выбрать случайное целое число a , $2 \leq a < n$. 2. Вычислить r . 3. При $r \neq 1$ и $r \neq n-1$ результат: «Число n составное». 4. Вычислить символ Якоби s . 5. При $r = s \pmod{n}$ результат: «Число n составное». В противном случае результат: «Число n , вероятно, простое».

** 4. Алгоритм, реализующий тест Миллера-Рабина.**

Вход. Нечетное целое число $n \geq 5$. Выход. «Число n , вероятно, простое» или «Число n составное». 1. Представить $n-1$ в виде $n-1 = 2^s \cdot g$, где число g нечетное. 2. Выбрать случайное целое число a , $2 \leq a < n$. 3. Вычислить $y \leftarrow a^g \pmod{n}$. 4. При $y \neq 1$ и $y \neq n-1$ выполнить следующие действия. - Положить $j \leftarrow 1$. - Если $j \leq s-1$ и $y \neq n-1$, то - Положить $y \leftarrow y^2 \pmod{n}$. - При $y = 1$ результат: «Число n составное». - Положить $j \leftarrow j+1$. - При $y \neq n-1$ результат: «Число n составное». 5. Результат: «Число n , вероятно, простое».

В задании лабораторной предлагается рассмотреть все алгоритмы. Исходный код написан на языке Julia [[@doc-julia](#)]. —————

Цель работы

Изучение и реализация вероятностных алгоритмов проверки чисел на простоту на языке Julia.

Ход работы

1. Алгоритм, реализующий тест Ферма

```
function fermat_test(n::Integer, k::Integer=10)
    # Проверка граничных случаев
    if n ≤ 3
        return n == 2 || n == 3
    end
    if iseven(n)
        return false
    end

    # k итераций теста
    for _ in 1:k
        a = rand(2:(n-2)) # Случайное основание
        if gcd(a, n) ≠ 1 # Проверка взаимной простоты
            return false
        end
        if powermod(a, n-1, n) ≠ 1 # Проверка малой теоремы Ферма
            return false
        end
    end
    return true
end
```

2. Алгоритм вычисления символа Якоби.

```
function jacobi_symbol(a::Integer, n::Integer)
    if n ≤ 0 || iseven(n)
        throw(DomainError(n, "n должно быть нечётным положительным числом"))
    end

    a = mod(a, n)
    result = 1

    while a ≠ 0
        # Удаление множителей 2 (свойство символа Якоби)
        while iseven(a)
            a ÷= 2
            mod8 = n % 8
            if mod8 == 3 || mod8 == 5
                result = -result
            end
        end

        # Закон квадратичной взаимности
    end
end
```

```

    a, n = n, a

    if a % 4 == 3 && n % 4 == 3
        result = -result
    end

    a = mod(a, n)
end

return n == 1 ? result : 0
end

```

3. Алгоритм, реализующий тест Соловья-Штрассена

```

function solovay_strassen_test(n::Integer, k::Integer=10)
    if n ≤ 3
        return n == 2 || n == 3
    end
    if iseven(n)
        return false
    end

    for _ in 1:k
        a = rand(2:(n-2))

        # Проверка НОД (если НОД > 1, то n составное)
        if gcd(a, n) > 1
            return false
        end

        # Вычисление  $a^{(n-1)/2} \bmod n$ 
        r = powermod(a, (n-1)÷2, n)

        if r ≠ 1 && r ≠ n-1
            return false
        end

        # Вычисление символа Якоби
        j = jacobi_symbol(a, n)

        # Приведение символа Якоби по модулю n
        j_mod = mod(j, n)
        if j_mod < 0
            j_mod += n
        end

        # Проверка критерия Эйлера
        if r ≠ j_mod
            return false
        end
    end
end

```

```

    return true
end

```

4. Алгоритм, реализующий тест Миллера-Рабина

```

function miller_rabin_test(n::Integer, k::Integer=10)
    if n ≤ 3
        return n == 2 || n == 3
    end
    if iseven(n)
        return false
    end

    # Представление n-1 в виде 2^s * r (r - нечетное)
    s = 0
    r = n - 1
    while iseven(r)
        r ÷= 2
        s += 1
    end

    # k итераций теста
    for _ in 1:k
        a = rand(2:(n-2))
        x = powermod(a, r, n)

        # Проверка первого условия: a^r ≡ 1 (mod n)
        if x ≠ 1 && x ≠ n-1
            j = 1
            # Проверка второго условия для j = 1..s-1
            while j ≤ s-1 && x ≠ n-1
                x = powermod(x, 2, n)
                if x == 1
                    return false # Найден нетривиальный квадратный корень из 1
                end
                j += 1
            end
            if x ≠ n-1
                return false # Не выполняется ни одно из условий
            end
        end
    end
    return true
end

```

- Восстановление НОД с общим множителем

5. Вывод результатов

```

function test_all_algorithms(n::Integer, iterations::Integer=10)
    println("Тестирование числа $n")
    println("="^50)

    # Запускаем все тесты

```

```

fermat_result = fermat_test(n, iterations)
solovay_result = solovay_strassen_test(n, iterations)
miller_result = miller_rabin_test(n, iterations)
enhanced_result = enhanced_fermat_test(n, iterations)

# Выводим результаты
println("Тест Ферма: ", fermat_result ? "Вероятно простое" : "Составное")
println("Тест Соловэя-Штрассена: ", solovay_result ? "Вероятно простое" : "Составное")
println("Тест Миллера-Рабина: ", miller_result ? "Вероятно простое" : "Составное")
println("Улучшенный тест Ферма: ", enhanced_result ? "Вероятно простое" : "Составное")
println()
end

# Демонстрация на двух числах
test_all_algorithms(17, 5)
test_all_algorithms(29, 5)

```

Результат тестирования представлен на рис.1

Рис. 1 Тестирование:

```

Тестирование числа 17
=====
Тест Ферма: Вероятно простое
Тест Соловэя-Штрассена: Вероятно простое
Тест Миллера-Рабина: Вероятно простое
Улучшенный тест Ферма: Вероятно простое

Тестирование числа 29
=====
Тест Ферма: Вероятно простое
Тест Соловэя-Штрассена: Вероятно простое
Тест Миллера-Рабина: Вероятно простое
Улучшенный тест Ферма: Вероятно простое

```

Вывод: В ходе данной работы мной были изучены и реализованы различные вероятностные алгоритмы проверки чисел на простоту. Написан программный код на языке Julia и протестирован.