Отчет по лабораторной работе №5

Вероятностные алгоритмы проверки чисел на чистоту

Арам Грачьяевич Саргсян

Содержание

1 Цель работы **5**

2 Выполнение лабораторной работы **6**

3 Выводы **9**

Список иллюстраций

2.1 Тест Ферма 6
2.2 Символ Якоби 7
2.3 Тест Соловэя-Штрассена 7
2.4 Тест Миллера-Рабина 8

Список таблиц

1 Цель работы

Изучить вероятностные алгоритмы проверки чисел на простоту.

2 Выполнение лабораторной работы

1. Я реализовал тест Ферма на языке julia (рис. 2.1).

```
using Random

function fermat_test(n::Int)
    # Проверка на условие входа
    if n < 5 || n % 2 == 0
        return "Входное число должно быть нечётным и больше или равно 5"
    end

    # Шаг 1: Выбрать случайное целое число a, 2 ≤ a ≤ n - 2
    a = rand(2:n-2)

    # Шаг 2: Вычислить r = a^(n-1) % n
    r = powermod(a, n-1, n)  # эффективное возведение в степень по модулю

    # Шаг 3: Проверка результата
    if r == 1
        return "Число n, вероятно, простое"
    else
        return "Число n составное"
    end
end
n=7
println(n,": ",fermat_test(n))
n=9
println(n,": ",fermat_test(n))
```

```
7: Число n, вероятно, простое
9: Число n составное
```

Рис. 2.1: Тест Ферма

2. Я реализовал алгоритм вычисления символа Якоби на julia (рис. 2.2).

```
function jacobi(a::Int, n::Int)

    if n < 3 || n % 2 == 0
        return "Число n должно быть нечётным и >= 3"
    end
    if a < 0 || a >= n
        return "Число a должно быть в диапазоне 0 ≤ a < n"
    end
    g = 1

    while a != 0
        if a == 0
            return 0
        end

        if a == 1
            return g
        end

        k = 0
        while a % 2 == 0
            a ÷= 2
            k += 1
        end

        if k % 2 == 1
            if n % 8 == 3 || n % 8 == 5
                g = -g
            end
        end

        if a == 1
            return g
        end

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

        a, n = n % a, a
    end

    return g
end
```

```
jacobi (generic function with 1 method)
```

Рис. 2.2: Символ Якоби

3. Я реализовал тест Соловэя-Штрассена на языке julia (рис. 2.3).

```
# Тест Соллобея-Штрассена
function solovay_strassen_test(n::Int, k::Int)
    if n < 2
        return false
    elseif n == 2
        return true
    elseif n % 2 == 0
        return false
    end

    for _ in 1:k
        a = rand(2:n-2)  # Случайное число в диапазоне [2, n-2]
        x = jacobi(a, n)
        if x == 0 || powermod(a, (n-1) ÷ 2, n) != (x % n + n) % n
            return "Число n составное"
        end
    end

    return "Число n, вероятно, простое"
end

n = 61  # Число для проверки
k = 5   # Количество итераций
println(n, ":", solovay_strassen_test(n, k))
```

```
61:Число n, вероятно, простое
```

Рис. 2.3: Тест Соловэя-Штрассена

4. Я реализовал тест Миллера-Рабина на языке julia

```
function miller_rabin_test(n::Int, k::Int)

    if n < 2
        return false
    elseif n == 2
        return true
    elseif n % 2 == 0
        return false
    end

    # Представить n - 1 в виде 2^s * d, где d нечетно
    d = n - 1
    s = 0
    while d % 2 == 0
        d ÷= 2
        s += 1
    end

    # Повторить тест k раз для повышения надежности
    for _ in 1:k
        a = rand(2:n-2)  # Случайное число a в диапазоне [2, n-2]
        x = powermod(a, d, n)  # a^d % n

        if x == 1 || x == n - 1
            continue  # Продолжить, если x == 1 или x == n - 1
        end

        composite = true
        for _ in 1:(s - 1)
            x = powermod(x, 2, n)  # x = x^2 % n
            if x == n - 1
                composite = false
                break
            end
        end

        if composite
            return "Число n составное"
        end
    end

    return "Число n, вероятно, простое"
end

n = 61
k = 5
println(n,":", miller_rabin_test(n, k))
```

```
61:Число n, вероятно, простое
```

Рис. 2.4: Тест Миллера-Рабина

3 Выводы

Я реализовал алгоритмы вычисления числа на простоту на языке julia.