marp: true css: custom-theme.css title: "Лабораторная работа №5"на простоту" subtitle: "Дисциплина: Математические основы защиты информации и информационной безопасности" author: Хосе Фернандо Леон Атупанья, НФИмд-01-24, 1032249918 institute: Российский Университет Дружбы Народов, Москва, Россия date: 26 октября 2024

Лабораторная работа №5

Дисциплина: Математические основы защиты информации и информационной безопасности

Тема: Вероятностные алгоритмы проверки чисел

Студент: Леон Фернандо Хосе Фернандо

Цель работы

Вычислить и проверить если входоющее чисел оставное или простое, используя алгоритмы, представленные в лабораторном рабочем материале 5.

Задание

1. Реализовать рассмотренные алгоритмы программно (4 коды)

2. Выполнение лабораторной работы

1. Алгоритм, реализующий тест Ферма

Реализуем тест Ферма на простоту, вероятностный алгоритм, используемый для проверки того, является ли число простым. Тест основан на Малой теореме Ферма, которая предполагает, что если число n составное.

1. Алгоритм Евклида (Код 1/2)

```
using Random

function fermat_primality_test(n::Int)
    if n < 5 || iseven(n)
        return "Input must be a odd integer greater than or equal 5"
    end

    a = rand(2:(n - 2))
    r = powermod(a, n - 1, n)
```

```
    if r == 1
        return "The number n is a probably prime"
    else
        return "The number n is composite"
    end
end
```

1. Алгоритм Евклида (Код 2/2)

```
function input(prompt:: AbstractString)
    print(prompt)
    return chomp(readline())
end

n = input("n = ")
num1 = parse(Int, n)

result = fermat_primality_test(num1)
println(result)
```

2. Алгоритм вычисления символа Якоби (1/3)

Вычислите символ Якоби (n/a), важную функцию в теории чисел, часто используемую в алгоритмах, связанных с проверкой на простоту и квадратичными вычетами. Символ Якоби обобщает символ Лежандра и может быть вычислен для любого целого числа a и любого положительного нечетного числа n.

```
function jacobi_symbol(a::Int, n::Int)
    g = 1

    if a == 0
        return 0
    end

    if a == 1
        return g
    end
```

2. Алгоритм вычисления символа Якоби (2/3)

```
            if iseven(k)
        s = 1
    else
        if n % 8 == 1 || n % 8 == 7
            s = 1
        else
            s = -1
        end
    end

    if a1 == 1
        return g * s
    end

    if n % 4 == 3 && a1 % 4 == 3
        s = -s
    end

    a = n % a1
    n = a1
    g *= s

    return jacobi_symbol(a, n) * g
end
```

2. Алгоритм вычисления символа Якоби (3/3)

```
a = input("a = ")
num1 = parse(Int, a)

n = input("n = ")
num2 = parse(Int, n)

result = jacobi_symbol(num1, num2)
println("Jacobi symbol (a/b): ", result)
```

3. Алгоритм, реализующий тест Соловэя-Штрасеена (1/3)

```
using Random

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

```
    g = 1

    while a != 0
        #Step 4: Factor out powers of 2 in a to find a1(odd part)
        k = 0
        while iseven(a)
            a = div(a, 2)
            k += 1
        end
        a1 = a

        #Determine s based on k and n mod 8
        s = 1
        if isodd(k)
            if n % 8 == 3 || n % 8 == 5
                s = -1
            end
        end
```

3. Алгоритм, реализующий тест Соловэя-Штрасеена (2/3)

```
        if n % 4 == 3 && a1 % 4 == 3
            s = -s
        end

        #Update g, a and n
        g *= s
        a, n = n % a1, a1
    end

    return g == 1 ? 1 : (n == 1 ? g : 0)
end

function solovay_strassen_test(n::Int)
    if n < 5 || iseven(n)
        return "Input must be an odd integer greater than or equal to 5"
    end
```

3. Алгоритм, реализующий тест Соловэя-Штрасеена (3/3)

```
        a = rand(2: n - 2)
    r = powermod(a, div(n - 1, 2), n)

    if r != 1 && r != n - 1
        return "The number n is composite"
    end

    s = jacobi_symbol(a, n)

    if r % n != s
        return "The number n is compisite"
    else
        return "The number n is probably prime"
    end
end
```

4. Алгоритм, реализующий тест Миллера-Рабина (1/2)

```
using Random

function miller_rabin_test(n::Int)
    r = n - 1
    s = 0
    while iseven(r)
        r = div(r, 2)
        s += 1
    end

    a = rand(2: n - 2)
    y = powermod(a, r, n)

    if y != 1 && y != n - 1
        j = 1
        while j <= s - 1 && y != n - 1
            y = powermod(y, 2, n)
            if y == 1
                return "The number n is composite"
            end
            j += 1
        end
```

4. Алгоритм, реализующий тест Миллера-Рабина (2/2)

```
        if y != n - 1
            return "The number n is composite"
        end
    end
    return "The number n is probably prime"
end

n = input("n = ")
num1 = parse(Int, n)

r = miller_rabin_test(num1)
println(r)
```

Вывод

Каждый алгоритм обеспечивает различный баланс скорости и точности, при этом алгоритм Миллера-Рабина, как правило, является наиболее надежным для практического использования, особенно когда надежность имеет решающее значение. Тест Ферма, хотя и быстрый, более уязвим к ошибкам при работе с определенными составными числами, и метод Соловея-Штрассена находится между ними, предлагая разумный компромисс.