

Лабораторна работа № 6

Вероятностные проверки чисел на простоту

Покрас Илья Михайлович

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
4.1	Алгоритм теста Ферма	7
4.2	Алгоритм вычисления символа Якоби	8
4.3	Алгоритм теста Соловья-Штрассена	10
4.4	Алгоритм теста Миллера-Рабина	11
5	Выводы	13
	Список Литературы	14

Список иллюстраций

4.1	Функция алгоритма теста Ферма	7
4.2	Результат выполненного кода (1)	7
4.3	Функция алгоритма вычисления символа Якоби	9
4.4	Результат выполненного кода (2)	9
4.5	Функция алгоритма теста Соловья-Штрассена	10
4.6	Результат выполненного кода (3)	10
4.7	Функция алгоритма теста Миллера-Рабина	11
4.8	Результат выполненного кода (4)	12

1 Цель работы

Реализовать алгоритмы вероятностной проверки чисел на простоту.

2 Задание

- Реализовать алгоритм теста Ферма;
- Реализовать алгоритм вычисления символа Якоби;
- Реализовать алгоритм теста Соловья-Штрассена;
- Реализовать алгоритм теста Миллера-Рабина.

3 Теоретическое введение

Тест простоты Ферма в теории чисел — это тест простоты натурального числа n , основанный на малой теореме Ферма. Малая теорема Ферма - теорема теории чисел, которая утверждает, что: если p - простое число и a - целое число, не делящееся на p , то $a^{(p-1)} - 1$ делится на p

Символ Якоби — теоретико-числовая функция двух аргументов, введенная К. Якоби в 1837 году. Символ Якоби обобщает символ Лежандра на все нечётные числа, большие единицы. Символ Кронекера — Якоби, в свою очередь, обобщает символ Якоби на все целые числа, но в практических задачах символ Якоби играет гораздо более важную роль, чем символ Кронекера — Якоби.

Тест Соловея — Штрассена — вероятностный тест простоты, открытый в 1970-х годах Робертом Мартином Соловеем совместно с Фолькером Штрассеном. Тест всегда корректно определяет, что простое число является простым, но для составных чисел с некоторой вероятностью он может дать неверный ответ. Основное преимущество теста заключается в том, что он, в отличие от теста Ферма, распознает числа Кармайкла как составные.

Тест Миллера — Рабина — вероятностный полиномиальный тест простоты. Тест Миллера — Рабина, наряду с тестом Ферма и тестом Соловея — Штрассена, позволяет эффективно определить, является ли данное число составным. Однако, с его помощью нельзя строго доказать простоту числа. Тем не менее тест Миллера — Рабина часто используется в криптографии для получения больших случайных простых чисел.

4 Выполнение лабораторной работы

4.1 Алгоритм теста Ферма

Я создал функцию алгоритма теста Ферма для проверки числа на простоту с входными данными в виде числа и с выводом сообщение о том, каким число является, в зависимости от того, выполняется ли условие. Далее вызвал функцию, входным аргументом которой будет являться переменная, которую я задал до этого (рис. 4.1).

```
function fermat_test(n)
    a = rand(2:n-2)
    if powermod(a,(n-1), n) != 1
        return "$n is composite"
    end
    return "$n is probably prime"
end

n = 17

fermat_test(n)
```

Рис. 4.1: Функция алгоритма теста Ферма

И получил следующее сообщение (рис. 4.2):

```
17 is probably prime
```

Рис. 4.2: Результат выполненного кода (1)

4.2 Алгоритм вычисления символа Якоби

Далее я создал функцию алгоритма вычисления символа Якоби для работы которого требуются входные параметры a и n , такие, что $n \geq 3$, $0 \leq a < n$. Далее я задал переменные, которые будут являться этими входными данными и используя их вызвал функцию (рис. 4.3).


```

function jacobi(a, n)
    if a == 0
        return 0
    elseif a == 1
        return 1
    end

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

    if e % 2 == 0
        s = 1
    else
        s = n % 8 == 1 || n % 8 == 7 ? 1 : -1
    end

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

    a1 = a
    n1 = n % a1

    if a1 == 1
        return s
    else
        return s * jacobi(n1, a1)
    end
end

a = 2
n = 10
println(jacobi(a, n))

```

Рис. 4.3: Функция алгоритма вычисления символа Якоби

И получил следующий результат (рис. 4.4):

-1

Рис. 4.4: Результат выполненного кода (2)

4.3 Алгоритм теста Соловья-Штрассена

После я создал функцию алгоритма теста Соловья-Штрассена, который так же вызывает функцию вычисления символа Якоби, который был реализован ранее. Будет выводиться сообщение о том, каким число является, в зависимости от того, выполняется ли условие. Далее вызвал функцию, входным аргументом которой будет являться переменная, которую я задал до этого (рис. 4.5).

```
> function jacobi(a, n) ...  
end  
  
function solovay_strassen(n)  
  
    a = rand(2:n-2)  
    r = a^((n-1)/2) % n  
    if r != 1 && r != n-1  
        return "$n is composite"  
    end  
    if r != jacobi(a, n) % n  
        return "$n is composite"  
    end  
  
    return "$n is probably prime"  
end  
  
n = 11  
println(solovay_strassen(n))
```

Рис. 4.5: Функция алгоритма теста Соловья-Штрассена

И получил следующий результат (рис. 4.6):

-1

Рис. 4.6: Результат выполненного кода (3)

4.4 Алгоритм теста Миллера-Рабина

Я создал функцию алгоритма теста Миллера-Рабина, требующую число $n \geq 5$, которое проверяется на простоту. Будет выводиться сообщение о том, каким число является, в зависимости от того, выполняется ли условие. Далее я создал переменную, которая будет являться входным параметром вызываемой функции (рис. 4.7):

```
function miller_rabin(n)
    s = 0
    r = n - 1
    while r % 2 == 0
        r = r ÷ 2
        s += 1
    end

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

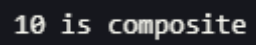
    if y != 1 && y != n - 1
        j = 1
        while j <= s - 1 && y != n - 1
            y = (y^2) % n
            if y == 1
                return "$n is composite"
            end
            j += 1
        end
        if y != n - 1
            return "$n is composite"
        end
    end

    return "$n is probably prime"
end

n = 10
println(miller_rabin(n))
```

Рис. 4.7: Функция алгоритма теста Миллера-Рабина

И получил следующий результат (рис. 4.8):

A dark rectangular box with the text "10 is composite" in a light-colored, monospaced font.

```
10 is composite
```

Рис. 4.8: Результат выполненного кода (4)

5 Выводы

Я реализовал алгоритмы вероятностной проверки чисел на простоту.

Список Литературы

1. Julia - Control Flow
2. Julia - Mathematical Operations
3. Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone - Handbook of Applied Cryptography