

Отчёт по лабораторной работе №5: Вероятностные алгоритмы проверки чисел на простоту

**Дисциплина: Математические основы защиты информации и
информационной безопасности**

Манаева Варвара Евгеньевна

Содержание

1	Общая информация о задании лабораторной работы	4
1.1	Цель работы	4
1.2	Задание [1]	4
2	Теоретическое введение [2]	5
2.1	Проверки чисел на простоту	5
2.2	Вероятностные проверки чисел на простоту	5
3	Выполнение лабораторной работы [1]	7
3.1	Реализовать тест Ферма	7
3.1.1	Проверка работы функции	8
3.2	Реализовать алгоритм вычисления символа Якоби	8
3.2.1	Проверка работы функции	10
3.3	Реализовать тест Соловья-Штрассена	10
3.3.1	Проверка работы функции	11
3.4	Реализовать тест Миллера-Робина	12
3.4.1	Проверка работы функции	13
4	Выводы	14
	Список литературы	15

Список иллюстраций

3.1	Результат работы реализованной функции теста Ферма	8
3.2	Результат работы реализованной функции алгоритма вычисления символа Якоби	10
3.3	Результат работы реализованной функции теста Соловья-Штрассена	11
3.4	Результат работы реализованной функции теста Миллера-Робина	13

1 Общая информация о задании лабораторной работы

1.1 Цель работы

Ознакомиться с алгоритмами вероятностной проверки чисел на простоту.

1.2 Задание [1]

1. Реализовать тест Ферма;
2. Реализовать алгоритм вычисления символа Якоби;
3. Реализовать тест Соловья-Штрассена;
4. Реализовать тест Миллера-Робина.

2 Теоретическое введение [2]

2.1 Проверки чисел на простоту

Существует два типа критериев простоты: детерминированные и вероятностные. Детерминированные тесты позволяют доказать, что тестируемое число - простое. Практически применимые детерминированные тесты способны дать положительный ответ не для каждого простого числа, поскольку используют лишь достаточные условия простоты.

Детерминированные тесты более полезны, когда необходимо построить большое простое число, а не проверить простоту, скажем, некоторого единственного числа.

На сегодня известно достаточно много алгоритмов проверки чисел на простоту. Несмотря на то, что большинство из таких алгоритмов имеет субэкспоненциальную оценку сложности, на практике они показывают вполне приемлемую скорость работы.

2.2 Вероятностные проверки чисел на простоту

В отличие от детерминированных, вероятностные тесты можно эффективно использовать для тестирования отдельных чисел, однако их результаты, с некоторой вероятностью, могут быть неверными. К счастью, ценой количества повторений теста с модифицированными исходными данными вероятность ошибки можно сделать как угодно малой.

На практике рассмотренные алгоритмы чаще всего по отдельности не применяются. Для проверки числа на простоту используют либо их комбинации, либо детерминированные тесты на простоту.

Детерминированный алгоритм всегда действует по одной и той же схеме и гарантированно решает поставленную задачу. Вероятностный алгоритм использует генератор случайных чисел и дает не гарантированно точный ответ. Вероятностные алгоритмы в общем случае не менее эффективны, чем детерминированные (если используемый генератор случайных чисел всегда дает набор одних и тех же чисел, возможно, зависящих от входных данных, то вероятностный алгоритм становится детерминированным).

3 Выполнение лабораторной работы

[1]

3.1 Реализовать тест Ферма

Информация о задаче.

Исходный код написан на языке Julia [3]. Код функции, осуществляющей тест Ферма, представлен ниже.

```
function testFerma(n)
    if n < 5
        return "Incorrect input."
    end
    a = rand(2:n-2)
    r = powermod(a, n-1, n)
    if r == 1
        return "Число " * string(n) * ", вероятно, простое."
    else
        return "Число " * string(n) * " составное."
    end
end
```

3.1.1 Проверка работы функции

```
display(testFerma(441))
```

```
display(testFerma(443))
```

Результат работы кода представлен ниже (рис. 3.1).

1. Тест Ферма

```
[1]: function testFerma(n)
      if n < 5
          return "Incorrect input."
      end
      a = rand(2:n-2)
      r = powermod(a, n-1, n)
      if r == 1
          return "Число " * string(n) * ", вероятно, простое."
      else
          return "Число " * string(n) * " составное."
      end
  end

[1]: testFerma (generic function with 1 method)

[2]: display(testFerma(441))
      display(testFerma(443))

      "Число 441 составное."
      "Число 443, вероятно, простое."
```

Рис. 3.1: Результат работы реализованной функции теста Ферма

3.2 Реализовать алгоритм вычисления символа Якоби

Информация о задаче.

Исходный код написан на языке Julia [3]. Код функции, осуществляющей алгоритм вычисления символа Якоби, представлен ниже.

```
function YacobySymbol(n, a)
    if n < 3 || a >= n || a < 0
        return "Incorrect input."
    end
    g = 1
    a1 = 0
    k = 0
    s = 0
    while a1 != 1
```



```

if a == 0
    return 0
elseif a == 1
    return 1
end
a1 = a
k = 0
while a1 % 2 == 0
    k += 1
    a1 = round(Int64, a1 / 2)
end

if k % 2 == 0 || (k % 2 == 1 && (n % 8 == 1 || n % 8 == 7))
    s = 1
elseif k % 2 == 1 && (n % 8 == 3 || n % 8 == 5)
    s = -1
end
if a1 == 1
    return g*s
end

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

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

```

end

3.2.1 Проверка работы функции

`YacobySymbol(443, 359)`

Результат работы кода представлен ниже (рис. 3.2).

```

a1 = round(1000 * a1 / 2)
end

if k % 2 == 0 || (k % 2 == 1 && (n % 8 == 1 || n % 8 == 7))
    s = 1
elseif k % 2 == 1 && (n % 8 == 3 || n % 8 == 5)
    s = -1
end

if a1 == 1
    return g*s
end

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

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

[3]: YacobySymbol (generic function with 1 method)

[4]: YacobySymbol(443, 359)

[4]: 1
```

Рис. 3.2: Результат работы реализованной функции алгоритма вычисления символа Якоби

3.3 Реализовать тест Соловья-Штрассена

Информация о задаче.

Исходный код написан на языке Julia [3]. Код функции, осуществляющей тест Соловья-Штрассена, представлен ниже.

```
function testSoloveyaShtrassena(n)
    if n < 5
        return "Incorrect input."
    end
    a = rand(2:n-2)
```

```

r = powermod(a, round(Int64, (n-1)/2), n)
if r != 1 && r != n-1
    return "Число " * string(n) * " составное."
else
    s = YacobySymbol(n, a)
    if r == s && r != NaN
        return "Число " * string(n) * " составное."
    end
    return "Число " * string(n) * ", вероятно, простое."
end
end
end

```

3.3.1 Проверка работы функции

```

display(testSoloveyaShtrassena(4463429))
display(testSoloveyaShtrassena(443))

```

Результат работы кода представлен ниже (рис. 3.3).

3. Тест Соловья-Штрассена

```

[5]: function testSoloveyaShtrassena(n)
    if n < 5
        return "Incorrect input."
    end
    a = rand(2:n-2)
    r = powermod(a, round(Int64, (n-1)/2), n)
    if r != 1 && r != n-1
        return "Число " * string(n) * " составное."
    else
        s = YacobySymbol(n, a)
        if r == s && r != NaN
            return "Число " * string(n) * " составное."
        end
        return "Число " * string(n) * ", вероятно, простое."
    end
end

[5]: testSoloveyaShtrassena (generic function with 1 method)

[6]: display(testSoloveyaShtrassena(4463429))
      display(testSoloveyaShtrassena(443))

      "Число 4463429 составное."
      "Число 443, вероятно, простое."

```

Рис. 3.3: Результат работы реализованной функции теста Соловья-Штрассена

3.4 Реализовать тест Миллера-Робина

Информация о задаче.

Исходный код написан на языке Julia [3]. Код функции, осуществляющей тест Миллера-Робина, представлен ниже.

```
function testMilleraRobina(n)
    if n < 5
        return "Incorrect input."
    end
    r = n-1
    s = 0
    while r % 2 == 0
        s += 1
        r = round{Int64}(r / 2)
    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
                "Число " * string(n) * " составное."
            end
            j += 1
        end
        if y != n-1
            "Число " * string(n) * " составное."
        else

```

```

        "Число " * string(n) * ", вероятно, простое."
    end
else
    return "Число " * string(n) * ", вероятно, простое."
end
end
end

```

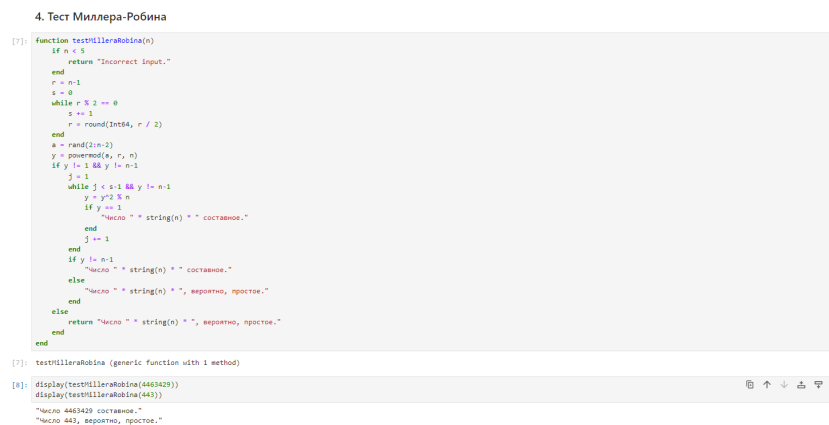
3.4.1 Проверка работы функции

```

display(testMilleraRobina(4463429))
display(testMilleraRobina(443))

```

Результат работы кода представлен ниже (рис. 3.4).



```

4. Тест Миллера-Робина
[7]: function testMilleraRobina(n)
    if n < 5
        return "Incorrect input."
    end
    r = n-1
    s = 0
    while r % 2 == 0
        s += 1
        r = round(int64, r / 2)
    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
                "число " * string(n) * " составное."
            end
            j += 1
        end
        if y != n-1
            "число " * string(n) * " составное."
        else
            "число " * string(n) * ", вероятно, простое."
        end
    else
        return "число " * string(n) * ", вероятно, простое."
    end
end

testMilleraRobina (generic function with 1 method)

[8]: display(testMilleraRobina(4463429))
display(testMilleraRobina(443))
"число 4463429 составное."
"число 443, вероятно, простое."

```

Рис. 3.4: Результат работы реализованной функции теста Миллера-Робина

4 Выводы

В результате работы мы ознакомились с вероятностными алгоритмами проверки чисел на простоту, а именно:

- Тестом Ферма;
- Алгоритмом вычисления символа Якоби;
- Тестом Соловья-Штрассена;
- Тестом Миллера-Робина.

Также были записаны скринкасты:

На RuTube:

- Весь плейлист
- Запись создания шаблона отчёта и презентации для заполнения
- Выполнения лабораторной работы
- Запись создания отчёта
- Запись создания презентации
- Защита лабораторной работы

На Платформе:

- Весь плейлист
- Запись создания шаблона отчёта и презентации для заполнения
- Выполнения лабораторной работы
- Запись создания отчёта
- Запись создания презентации
- Защита лабораторной работы

Список литературы

1. Лабораторная работа №5. Вероятностные алгоритмы проверки чисел на простоту [Электронный ресурс]. RUDN, 2024. URL: https://esystem.rudn.ru/pluginfile.php/2368514/mod_folder/content/0/lab05.pdf.
2. Математика криптографии и теория шифрования [Электронный ресурс]. URL: <https://intuit.ru/studies/courses/552/408/info>.
3. Julia 1.10 Documentation [Электронный ресурс]. 2024. URL: <https://docs.julialang.org/en/v1/>.