Лабораторная работа №5

Математические основы защиты информации и информационной безопасности

Николаев Дмитрий Иванович, НПМмд-02-24

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Изучить работу алгоритмов проверки чисел на простоту: алгоритм, реализующий тест Ферма; алгоритм вычисления символа Якоби; алгоритм, реализующий тест Соловэя-Штрассена; алгоритм, реализующий тест Миллера-Рабина; а также реализовать их программно.

2 Теоретическое введение

2.1 Вероятностные алгоритмы проверки чисел на простоту

Пусть a — целое число. Числа $ ±1, ±a$ называются *тривиальными делителями* числа a.

Целое число $p \in \mathbb{Z} \backslash \{0\}$ называется *простым*, если оно не является делителем единицы и не имеет других делителей, кроме тривиальных. В противном случае число $p \in \mathbb{Z} \backslash \{-1, 0, 1\}$ называется *составным*.

Например, числа $\pm 2, \pm 3, \pm 5, \pm 7, \pm 11, \pm 13, \pm 17, \pm 19, \pm 23, \pm 29$ являются простыми.

Пусть $m \in \mathbb{N}, m > 1$. Целые числа a и b называются сравнимыми по модулю m (обозначается $a \equiv b \pmod m$), если разность $a - b$ делится на m. Также эта процедура называется нахождением остатка от целочисленного деления a на m.

Проверка чисел на простоту является составной частью алгоритмов генерации простых чисел, применяемых в криптографии с открытым ключом. Алгоритмы проверки на простоту можно разделить на вероятностные и детерминированные.

Детерминированный алгоритм всегда действует по одной и той же схеме и гарантированно решает поставленную задачу (или не дает никакого ответа).

Вероятностный алгоритм использует генератор случайных чисел и дает негарантированно 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, зависящих от входных данных, то вероятностный алгоритм становится детерминированным).

Для проверки на простоту числа n вероятностным алгоритмом выбирают случайное число a ($1 < a < n$) и проверяют условия алгоритма. Если число n не проходит тест по основанию a, то алгоритм выдает результат «Число n составное», и число n действительно является составным.

Если же n проходит тест по основанию a, ничего нельзя сказать о том, действительно ли число n является простым. Последовательно проведя ряд проверок таким тестом для разных a и получив для каждого из них ответ «Число n вероятно, простое», можно утверждать, что число n является простым с вероятностью, близкой к 1. После t независимых выполнений теста вероятность того, что составное число n будет t раз объявлено простым (вероятность ошибки), не превосходит $\frac{1}{2^t}$.

2.1.1 Алгоритм, реализующий тест Ферма

Тест Ферма основан на малой теореме Ферма: для простого числа p и произвольного числа a, $1 \leq a \leq p-1$, выполняется сравнение

$$a^{p-1} \equiv 1 \pmod p$$

Следовательно, если для нечётного n существует такое целое a, что $1 < a < n$, НОД$(a, n) = 1$ и $a^{n-1} \not\equiv 1 \pmod n$, то число n составное. Отсюда получаем следующий вероятностный алгоритм проверки числа на простоту.

Вход: Нечётное 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».

На шаге 1 мы не рассматривали числа $a = 1$ и $a = n-1$, поскольку $1^{n-1} \equiv 1 \pmod{n}$ для любого целого n и $(n-1)^{n-1} \equiv (-1)^{n-1} \equiv 1 \pmod{n}$ для любого нечётного n.

2.1.2 Алгоритм вычисления символа Якоби

Вход: Нечётное целое число $n \geq 3$, целое число a, $0 \leq a < n$.
Выход: Символ Якоби $\left(\frac{a}{n}\right)$.

1. Положить $g \leftarrow 1$.
2. При $a = 0$ результат: 0.
3. При $a = 1$ результат: g.
4. Представить a в виде $a = 2^k a_1$, где число a_1 нечётное.
5. При четном k положить $s \leftarrow 1$, при нечётном k положить $s \leftarrow 1$, если $n \equiv \pm 1 \pmod{8}$; 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.

2.1.3 Алгоритм, реализующий тест Соловэя-Штрассена

Тест Соловэя-Штрассена основан на критерии Эйлера: нечётное число n является простым тогда и только тогда, когда для любого целого числа a, $1 \leq a \leq n-1$, взаимно простого с n, выполняется сравнение:

$$a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) \pmod{n},$$

где $\left(\frac{a}{n}\right)$ — *символ Якоби*.

Пусть $m, n \in \mathbb{Z}$, где $n = p_1 p_2 \dots p_r$ и числа $p_i \neq 2$ простые (не обязательно различные). *Символ Якоби* $\left(\dfrac{m}{n}\right)$ определяется равенством

$$\left(\frac{m}{n}\right) = \left(\frac{m}{p_1}\right)\left(\frac{m}{p_2}\right)\dots\left(\frac{m}{p_r}\right).$$

Вход: Нечётное целое число $n > 5$.

Выход: «Число n, вероятно, простое» или «Число n составное».

1. Выбрать случайное целое число a, $2 \leq a < n-2$.
2. Вычислить $r \leftarrow a^{\frac{n-1}{2}} \pmod{n}$.
3. При $r \neq 1$ и $r \neq n-1$ результат: «Число n составное».
4. Вычислить символ Якоби $s \leftarrow \left(\dfrac{a}{n}\right)$.
5. При $r \equiv s \pmod{n}$ результат: «Число n составное». В противном случае результат: «Число n вероятно, простое».

2.1.4 Алгоритм, реализующий тест Миллера-Рабина

На сегодняшний день для проверки чисел на простоту чаще всего используется тест Миллера-Рабина, основанный на следующем наблюдении. Пусть число n нечётное и $n-1 = 2^s r$, где r — нечётное. Если n простое, то для любого $a \geq 2$, взаимно простого с n, выполняется условие $a^{p-1} \equiv 1 \pmod{p}$.

Вход: Нечётное целое число $n \geq 5$.

Выход: «Число n, вероятно, простое» или «Число n составное».

1. Представить $n-1$ в виде $n-1 = 2^s r$, где число r нечётное.
2. Выбрать случайное целое число a, $2 \leq a < n-2$.
3. Вычислить $y \leftarrow a^r \pmod{n}$.
4. При $y \neq 1$ и $y \neq n-1$ выполнить следующие действия:
    1. Положить $j \leftarrow 1$.
    2. Если $j \leq s-1$ и $y \neq n-1$, то:
        1. Положить $y \leftarrow y^2 \pmod{n}$.

2. При $y = 1$ результат: «Число n составное».
3. Положить $j \leftarrow j + 1$.

3. При $y \neq n - 1$ результат: «Число n составное».

5. Результат: «Число n, вероятно, простое».

2.2 Резюме алгоритмов

1. **Тест Ферма**: Этот алгоритм проверяет, удовлетворяет ли случайное число a соотношению $a^{n-1} \equiv 1 \pmod{n}$, где $1 < a < n$. Если это условие не выполняется, число составное.
2. **Символ Якоби**: Функция вычисляет символ Якоби $\left(\frac{a}{n}\right)$ рекурсивно, используя свойства сравнения и перестановки чисел по модулю.
3. **Тест Соловэя-Штрассена**: Использует символ Якоби для проверки условия $a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) \pmod{n}$. Если это условие не выполняется для какого-либо a, число составное.
4. **Тест Миллера-Рабина**: Использует разложение $n-1$ на $2^s \cdot r$ и проверяет условия для случайных a, используя возведение в степень по модулю. Этот тест считается более надежным и чаще используется.

3 Выполнение лабораторной работы

Действуя согласно [1], реализуем все описанные алгоритмы проверки чисел на простоту на языке Julia.

Сначала реализуем функцию модульного экспоненциирования и алгоритм теста Ферма (Рис.[3.1]). Для реализации алгоритма теста Соловэя-Штрассена (Рис.[3.4]) предварительно определим функции вычисления числа делений некоторого заданного числа на 2 (Рис.[3.2]) и символа Якоби (Рис.[3.3]). После чего реализуем алгоритм Миллера-Рабина (Рис.[3.5]). Для следующего набора чисел: 13 - простое, 15 - составное, 17 - простое, 19 - простое, 561 - составное, 1105 - составное, 1729 - составное, 2143 - простое, 2399 - простое (Рис.[3.6]), проверим работу данных алгоритмов, в результате чего получим следующий вывод, представленный на Рис.[3.7].

```julia
using Random

"""Возведение в степень по модулю a^b mod c"""
function powermod(a::Int, b::Int, c::Int)::Int
    res = 1
    a = a % c
    while b > 0
        if b % 2 == 1
            res = (res * a) % c
        end
        b = div(b, 2)
        a = (a * a) % c
    end
    return res
end

"""Функция проверки числа на простоту с помощью теста Ферма"""
function Fermat_Test(n::Int, k::Int = 5)::Bool
    if n == 2
        return true
    elseif n < 2 || n % 2 == 0
        return false
    end
    if n < 5
        error("Число должно быть больше 4")
    end
    for _ in 1:k
        # Выбираем случайное число a, 2 <= a <= n-2
        a = rand(2:n-2)
        # Проверяем условие Ферма: a^(n-1) ≡ 1 (mod n)
        if gcd(a, n) != 1 || powermod(a, n-1, n) != 1
            return false # Число составное
        end
    end
    return true # Число, вероятно, простое
end
```

Рис. 3.1: Код алгоритма теста Ферма на Julia

```julia
"""Вычисление числа делений на 2"""
function Powers_of_Two(a::Int)::Int
    k = 0
    while a % 2 == 0
        k += 1
        a = div(a, 2)
    end
    return k
end
```

Рис. 3.2: Код вычисления количества делений некоторого числа на 2 на Julia

```julia
"""Вычисление символа Якоби (a/n)"""
function Jacobi_Symbol(a::Int, n::Int)::Int
    if n < 3 || n % 2 == 0
        error("n должно быть положительным нечётным числом большим 2")
    end
    if a >= n || a < 0
        error("a должно быть неотрицательным числом меньшим n")
    end
    if gcd(a, n) != 1 return 0 end
    g = 1
    while a > 1
        k = Powers_of_Two(a)
        s = 1
        if k % 2 == 1
            if n % 8 == 3 || n % 8 == 5
                s = -1
            end
        end
        a = div(a, 2^k)
        if a == 1
            return g*s
        end
        if a % 4 == 3 && n % 4 == 3
            s = -s
        end
        a, n, g = n % a, a, g*s
    end
    if a == 0 return 0 end
    return g
end
```

Рис. 3.3: Код вычисления символа Якоби на Julia

```julia
"""Функция проверки числа на простоту с помощью теста Соловэя-Штрассена"""
function Solovay_Strassen_Test(n::Int, k::Int = 5)::Bool
    if n < 2 || n % 2 == 0
        return false
    end
    if n < 5
        error("Число должно быть больше 4")
    end
    for _ in 1:k
        # Выбираем случ число a, 2 <= a < n-2
        a = rand(2:n-3)
        # Неотрицательный остаток
        r = powermod(a, div(n-1, 2), n)
        if r != 1 && r != n - 1
            return false # Число составное
        end
        jacobi = Jacobi_Symbol(a, n)
        if r != mod(jacobi, n)
            return false # Число составное
        end
    end
    return true # Число, вероятно, простое
end
```

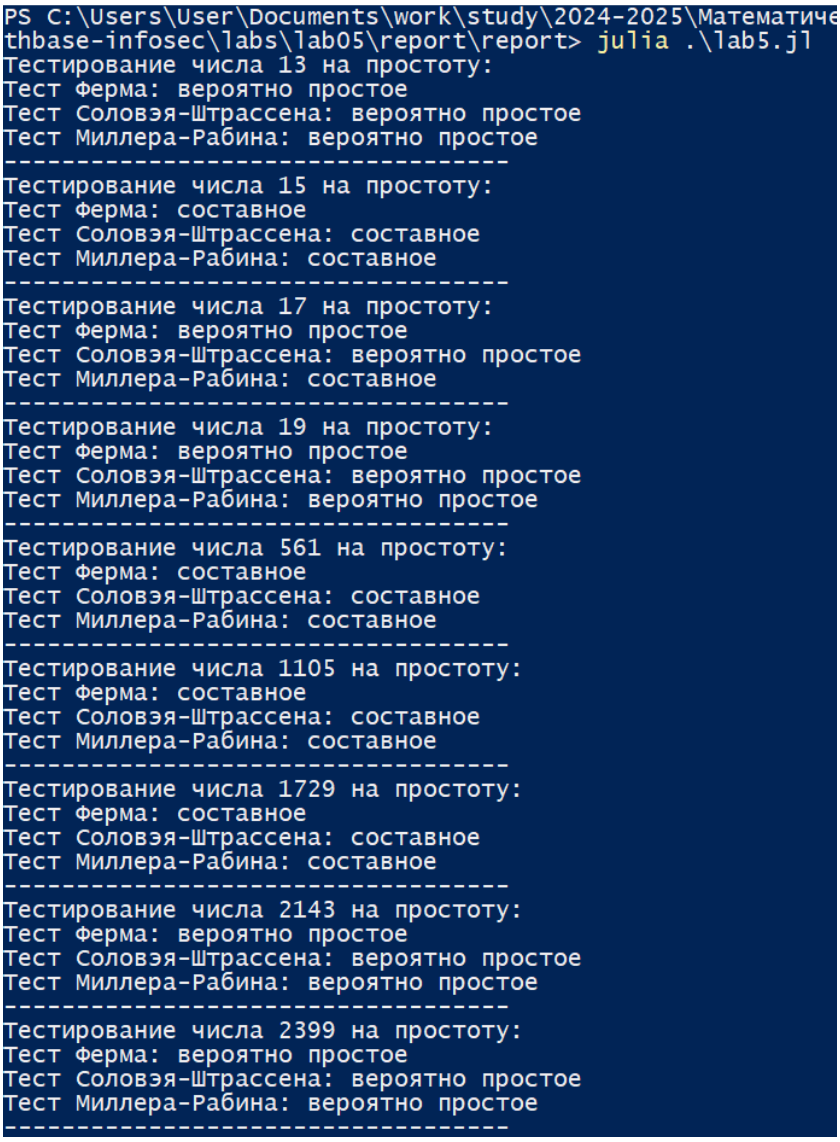
Рис. 3.4: Код алгоритма теста Соловэя-Штрассена на Julia

```julia
"""Функция проверки числа на простоту с помощью теста Миллера-Рабина"""
function Miller_Rabin_Test(n::Int, k::Int = 5)::Bool
    if n < 2 || n % 2 == 0
        return false
    end
    if n < 5
        error("Число должно быть больше 4")
    end
    # Представляем (n-1) в виде 2^s * r, где число r - нечётное
    s = 0
    r = n - 1
    while r % 2 == 0
        r = div(r, 2)
        s += 1
    end
    for _ in 1:k
        # Выбираем случ число a, 2 <= a < n-2
        a = rand(2:n-3)
        # Вычисляем y = a^r mod n
        y = powermod(a, r, n)
        if y != 1 && y != n - 1
            for _ in 1:(s-1)
                y = powermod(y, 2, n)
                if y == 1
                    return false # Число составное
                end
            end
            if y != n - 1
                return false # Число составное
            end
        end
    end
    return true # Число, вероятно, простое
end
```

Рис. 3.5: Код алгоритма теста Миллера-Рабина на Julia

```julia
# Тестирование каждого алгоритма на примерах
function Test_Algorithms(n)
    println("Тестирование числа $n на простоту:")

    # Тест Ферма
    fermat_result = Fermat_Test(n)
    println("Тест Ферма: $(fermat_result ? "вероятно простое" : "составное")")

    # Тест Соловэя-Штрассена
    solovay_result = Solovay_Strassen_Test(n)
    println("Тест Соловэя-Штрассена: $(solovay_result ? "вероятно простое" : "составное")")

    # Тест Миллера-Рабина
    miller_rabin_result = Miller_Rabin_Test(n)
    println("Тест Миллера-Рабина: $(miller_rabin_result ? "вероятно простое" : "составное")")
end

# Пример использования: (простое, составное, простое, простое,
# составное, составное, составное, простое, простое)
for n in [13, 15, 17, 19, 561, 1105, 1729, 2143, 2399]
    Test_Algorithms(n)
    println("----------------------------------")
end
```

Рис. 3.6: Начальные данные для сравнения алгоритмов проверки чисел на простоту на Julia



```
PS C:\Users\User\Documents\work\study\2024-2025\Математиче
thbase-infosec\labs\lab05\report\report> julia .\lab5.jl
Тестирование числа 13 на простоту:
Тест Ферма: вероятно простое
Тест Соловэя-Штрассена: вероятно простое
Тест Миллера-Рабина: вероятно простое
-----------------------------------
Тестирование числа 15 на простоту:
Тест Ферма: составное
Тест Соловэя-Штрассена: составное
Тест Миллера-Рабина: составное
-----------------------------------
Тестирование числа 17 на простоту:
Тест Ферма: вероятно простое
Тест Соловэя-Штрассена: вероятно простое
Тест Миллера-Рабина: составное
-----------------------------------
Тестирование числа 19 на простоту:
Тест Ферма: вероятно простое
Тест Соловэя-Штрассена: вероятно простое
Тест Миллера-Рабина: вероятно простое
-----------------------------------
Тестирование числа 561 на простоту:
Тест Ферма: составное
Тест Соловэя-Штрассена: составное
Тест Миллера-Рабина: составное
-----------------------------------
Тестирование числа 1105 на простоту:
Тест Ферма: составное
Тест Соловэя-Штрассена: составное
Тест Миллера-Рабина: составное
-----------------------------------
Тестирование числа 1729 на простоту:
Тест Ферма: составное
Тест Соловэя-Штрассена: составное
Тест Миллера-Рабина: составное
-----------------------------------
Тестирование числа 2143 на простоту:
Тест Ферма: вероятно простое
Тест Соловэя-Штрассена: вероятно простое
Тест Миллера-Рабина: вероятно простое
-----------------------------------
Тестирование числа 2399 на простоту:
Тест Ферма: вероятно простое
Тест Соловэя-Штрассена: вероятно простое
Тест Миллера-Рабина: вероятно простое
-----------------------------------
```

Рис. 3.7: Результат выполнения кода и сравнения алгоритмов проверки чисел на простоту на Julia

4 Выводы

В ходе выполнения лабораторной работы я изучил работу алгоритмов проверки чисел на простоту: алгоритма, реализующего тест Ферма; алгоритма вычисления символа Якоби; алгоритма, реализующего тест Соловэя-Штрассена; алгоритма, реализующего тест Миллера-Рабина; а также реализовал их программно на языке Julia.

Список литературы

1. Лабораторная работа № 5. Вероятностные алгоритмы проверки чисел на простоту [Электронный ресурс]. Саратовский государственный университет имени Н.Г. Чернышевского, 2024.