Лабораторная работа №5: Презентация.

вероятностные алгоритмы проверки чисел на простоту.

Евдокимов Максим Михайлович. Группа - НФИмд-01-24.[1]

02 октября, 2024, Москва, Россия

[1]Российский Университет Дружбы Народов

Цели и задачи работы

Цель лабораторной работы

Вероятностные алгоритмы проверки чисел на простоту

Задание

1. Реализовать алгоритм теста Ферма.
2. Реализовать алгоритм вычисления символа Якоби.
3. Реализовать алгоритм теста Соловэя-Штрассена.
4. Реализовать алгоритм теста Миллера-Рабина.

Тест Ферма

Тест Ферма основан на малой теореме Ферма, которая утверждает, что если n — простое число, то для любого целого a такого, что $1 \leq a < n$, выполняется: $a^{n-1} \equiv 1 \mod n$

Алгоритм действий:

1. Выбрать случайное число a такое, что $1 \leq a < n$.
2. Вычислить $a^{n-1} \mod n$.
3. Если $a^{n-1} \not\equiv 1 \mod n$, то n – составное.
4. Если $a^{n-1} \equiv 1 \mod n\$$, то n вероятно простое.

Сравнение:

- **Плюсы:** Простой и быстрый.
- **Минусы:** Подвержен “числам Кармайкла” составным числам, которые проходят тест для всех a.

Символ Якоби — это обобщение символа Лежандра на случай, когда знаменатель является нечетным составным числом. Символ Якоби $\left(\frac{a}{n}\right)$ определяется для целого числа a и нечетного натурального числа n.

Если n — простое число, то символ Якоби совпадает с символом Лежандра. Символ Лежандра $\left(\frac{a}{p}\right)$ определяется для целого числа a и простого числа p и указывает, является ли a квадратичным вычетом по модулю p.

Свойства символа Якоби

1. Мультипликативность:

$$\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right)$$

2. Симметрия:

$$\left(\frac{a}{n}\right) = \left(\frac{a \mod n}{n}\right)$$

3. **Квадратичный закон взаимности:**

   Для нечетных натуральных чисел a и b:

$$(\frac{a}{b})(\frac{b}{a}) = (-1)^{\frac{(a-1)(b-1)}{4}}$$

4. **Свойства для $a = -1$ и $a = 2$:**

$$(\frac{-1}{n}) = (-1)^{\frac{n-1}{2}}$$

$$(\frac{2}{n}) = (-1)^{\frac{n^2-1}{8}}$$

Тест Соловея-Штрассена использует символ Якоби и малую теорему Ферма для определения вероятности простоты числа.

Алгоритм действий:

1. Выбрать случайное число a такое, что $1 \leq a < n$.
2. Вычислить символ Якоби $\left(\frac{a}{n}\right)$.
3. Вычислить $a^{(n-1)/2} \mod n$.
4. Если $\left(\frac{a}{n}\right) \not\equiv a^{(n-1)/2} \mod n$, то n — составное.
5. Если $\left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \mod n$, то n вероятно простое.

Сравнение:

- **Плюсы:** Более надежный, чем тест Ферма, так как не подвержен “числам Кармайкла”.
- **Минусы:** Требует вычисления символа Якоби, что может быть сложнее.

Тест Миллера-Рабина — это вероятностный тест, основанный на расширении малой теоремы Ферма и использующий свойства квадратичных вычетов.

Алгоритм действий:

1. Представить $n-1$ как $2^s \cdot d$, где d – нечетное.
2. Выбрать случайное число a такое, что $1 \leq a < n$.
3. Вычислить $a^d \mod n$.
4. Если $a^d \equiv 1 \mod n$ или $a^d \equiv -1 \mod n$, то n вероятно простое.
5. Иначе, вычислить $a^{2^r \cdot d} \mod n$ для $r = 1, 2, ..., s-1$.
6. Если для какого-то r выполняется $a^{2^r \cdot d} \equiv -1 \mod n$, то n вероятно простое.
7. Если ни одно из условий не выполняется, то n – составное.

Сравнение:

- **Плюсы:** Один из самых надежных вероятностных тестов, не подвержен “числам Кармайкла”.
- **Минусы:** Требует больше вычислений, чем тест Ферма.

Вывод:

- **Тест Ферма** — простой, но подвержен “числам Кармайкла”.
- **Тест Соловея-Штрассена** — более надежный, чем Ферма, но требует вычисления символа Якоби.
- **Тест Миллера-Рабина** — самый надежный из трех, но требует больше вычислений.

Каждый из этих тестов дает вероятностный результат, и для подтверждения простоты числа обычно используют несколько итераций теста.

Ход работы

Подготовка

Так для тестирования работы кода я создал простой шаблон генерирующий наше случайное число для проверки, а также необходимые для каждого алгоритма коэффициент или параметр который гарантированно меньше исходного, но не меньше 1.

```julia
# Создание списка для теста
tests = Matrix(undef, 0, 2)
for j in 1:10
      p = rand(3:10000)
      k = rand(1:p)
      tests = vcat(tests, [p, k]')
end
tests
```

```
10×2 Matrix{Any}:
 3846  3655
 9603  4874
 2116  1347
  288     2
    5     3
   55     8
 3379  1591
 6637  2043
 8752  1625
   67    54
```

Рис. 1: Случайная тестовая группа

Алгоритм теста Ферма

```julia
# 1. Алгоритм теста Ферма
function fermat_test(n::Int, k::Int=5)
    if n <= 1
        return false
    elseif n <= 3
        return true
    end

    for _ in 1:k
        a = rand(2:n-2)
        if powermod(a, n-1, n) != 1
            return false
        end
    end
    return true
end

println("Тест Ферма:")
for i in 1:size(tests)[1]
    t = fermat_test(tests[i, 1], tests[i, 2])
    println("Число ", tests[i, 1], t ? "" : " НЕ", " простое.")
end
```

Рис. 2: Код теста Ферма

Результат 1

```
Тест Ферма:
Число 3846 НЕ простое.
Число 9603 НЕ простое.
Число 2116 НЕ простое.
Число 288 НЕ простое.
Число 5 простое.
Число 55 НЕ простое.
Число 3379 НЕ простое.
Число 6637 простое.
Число 8752 НЕ простое.
Число 67 простое.
```

Рис. 3: Результат шифра Цезаря

Если посмотреть на полученный результаты может показаться что при данных значениях нет ни одного число которое при использовании теста Соловэя-Штрассена. Для проверки рассмотрим случай номер 5 при значениях n = 5 (число для проверки) и a = 3 (случайное число) оба при этом оказались простыми.

1. Начнём с символа Якоби: 3/5; 5 - простое и совпадает с символом Лежандра который вычисляем как 5/3 => 2/3 а, так как 2 = mod(3) и не является квадратом по модулю 3 то 2/3 = -1 => 3/5 => -1
2. Теперь вычислим $a^{(n-1)/2} * \mod n$ => $3^{(5-1)/2} * \mod 5$ => $3^2 * mod 5 = 4$
3. Сравним: -1 и 4 не ровны значит чисто по тесту Соловея-Штрассена при a = 3 и n = 5, n не простое. При других значениях a возможно он даст правильный ответ, но не здесь.

```julia
# 2. Алгоритм вычисления символа Якоби
function jacobi_symbol(n::Int, a::Int=5)
    if n < 0 || iseven(n)
        return "Не подходит n четное."
    end
    a %= n
    g = 1

    while a != 0
        while iseven(a)
            a >>= 1
            if (n % 8) in [3, 5]
                g = -g
            end
        end
        a, n = n, a
        if a % 4 == 3 && n % 4 == 3
            g = -g
        end
        a %= n
    end
    return n == 1 ? g : 0
end

println("Символ Якоби:")
for i in 1:size(tests)[1]
    t = jacobi_symbol(tests[i, 1])
    println("При n = ", tests[i, 1], " и a = ", tests[i, 2], " символ Якоби = ", t)
end
```

Рис. 4: Код вычисления символа Якоби

Результат 2

```
Символ Якоби:
При n = 3846 и a = 3655 символ Якоби = Не подходит n четное.
При n = 9603 и a = 4874 символ Якоби = -1
При n = 2116 и a = 1347 символ Якоби = Не подходит n четное.
При n = 288 и a = 2 символ Якоби = Не подходит n четное.
При n = 5 и a = 3 символ Якоби = 0
При n = 55 и a = 8 символ Якоби = 0
При n = 3379 и a = 1591 символ Якоби = 1
При n = 6637 и a = 2043 символ Якоби = -1
При n = 8752 и a = 1625 символ Якоби = Не подходит n четное.
При n = 67 и a = 54 символ Якоби = -1
```

Рис. 5: Результат вычисления символа Якоби

Алгоритм теста Соловэя-Штрассена

```julia
# 3. Алгоритм теста Соловэя-Штрассена
function solovay_strassen__test(n::Int, k::Int=5)
    if n <= 1
        return false
    elseif n <= 3
        return true
    end

    for _ in 1:k
        a = rand(2:n-2)
        x = jacobi_symbol(n, a)
        if typeof(x) == String
            return x
        end
        if x == 0 || powermod(a, (n-1) ÷ 2, n) != x % n
            return false
        end
    end
    return true
end

println("Тест Соловэя-Штрассена:")
for i in 1:size(tests)[1]
    t = solovay_strassen__test(tests[i, 1], tests[i, 2])
    println("При n = ", tests[i, 1], " и k = ", tests[i, 2], ": ", t)
end
```

Рис. 6: Код теста Соловэя-Штрассена

```
Тест Соловэя-Штрассена:
При n = 3846 и k = 3655: Не подходит n четное.
При n = 9603 и k = 4874: false
При n = 2116 и k = 1347: Не подходит n четное.
При n = 288 и k = 2: Не подходит n четное.
При n = 5 и k = 3: false
При n = 55 и k = 8: false
При n = 3379 и k = 1591: false
При n = 6637 и k = 2043: false
При n = 8752 и k = 1625: Не подходит n четное.
При n = 67 и k = 54: false
```

Рис. 7: Результат теста Соловэя-Штрассена

Алгоритм теста Миллера-Рабина

```
# 4. Алгоритм теста Миллера-Рабина
function miller_rabin___test(n::Int, k::Int=5)
    if n <= 1
        return false
    elseif n <= 3
        return true
    elseif iseven(n)
        return false
    end

    # Разложение n-1 на d * 2^r
    r, d = 0, n - 1
    while iseven(d)
        d >>= 1
        r += 1
    end

    for _ in 1:k
        a = rand(2:n-2)
        x = powermod(a, d, n)
        if x == 1 || x == n - 1
            continue
        end

        for _ in 1:(r-1)
            x = powermod(x, 2, n)
            if x == n - 1
                break
            end
        end
```

Рис. 8: Код теста Миллера-Рабина 1

```
        end

        if x != n - 1
            return false
        end
    end
    return true
end

println("Тест Миллера-Рабина:")
for i in 1:size(tests)[1]
    t = miller_rabin__test(tests[i, 1], tests[i, 2])
    println("Число ", tests[i, 1], t ? "" : " НЕ", " простое.")
end
```

Рис. 9: Код теста Миллера-Рабина 2

Результат 4

```
Тест Миллера-Рабина:
Число 3846 НЕ простое.
Число 9603 НЕ простое.
Число 2116 НЕ простое.
Число 288 НЕ простое.
Число 5 простое.
Число 55 НЕ простое.
Число 3379 НЕ простое.
Число 6637 простое.
Число 8752 НЕ простое.
Число 67 простое.
```

Рис. 10: Результат теста Миллера-Рабина

Выводы по проделанной работе

Вывод

В ходе выполнения лабораторной работе выли изучены такие способы определение простоты числа как алгоритм теста Ферма, алгоритм теста Миллера-Рабина и алгоритм теста Соловэя-Штрассена, и алгоритм вычисления символа Якоби.