Front matter

title: “Отчет по Лабораторной работе №5 по предмету Математические основы защиты информации и кибер безопасности” author: “Лобов Михаил Сергеевич”

Generic otions

lang: ru-RU toc-title: “Содержание”

Bibliography

bibliography: bib/cite.bib csl: pandoc/csl/gost-r-7-0-5-2008-numeric.csl

Pdf output format

toc: true # Table of contents toc-depth: 2 lof: true # List of figures lot: true # List of tables fontsize: 12pt linestretch: 1.5 papersize: a4 documentclass: scrreprt ## I18n polyglossia polyglossia-lang: name: russian options: - spelling=modern - babelshorthands=true polyglossia-otherlangs: name: english ## I18n babel babel-lang: russian babel-otherlangs: english ## Fonts mainfont: IBM Plex Serif romanfont: IBM Plex Serif sansfont: IBM Plex Sans monofont: IBM Plex Mono mathfont: STIX Two Math mainfontoptions: Ligatures=Common,Ligatures=TeX,Scale=0.94 romanfontoptions: Ligatures=Common,Ligatures=TeX,Scale=0.94 sansfontoptions: Ligatures=Common,Ligatures=TeX,Scale=MatchLowercase,Scale=0.94 monofontoptions: Scale=MatchLowercase,Scale=0.94,FakeStretch=0.9 mathfontoptions: ## Biblatex biblatex: true biblio-style: “gost-numeric” biblatexoptions: - parentracker=true - backend=biber - hyperref=auto - language=auto - autolang=other* - citestyle=gost-numeric ## Pandoc-crossref LaTeX customization figureTitle: “Рис.” tableTitle: “Таблица” listingTitle: “Листинг” lofTitle: “Список иллюстраций” lotTitle: “Список таблиц” lolTitle: “Листинги” ## Misc options indent: true header-includes: -

- keep figures where there are in the text
    - # keep figures where there are in the text

Цель работы

Изучить “Вероятностные алгоритмы проверки чисел на простоту”.

Задание

Реализовать алгоритмы проверки чисел на простоту на языке Julia.

Теоретическое введение

Пусть a – целое число. Числа $\pm 1, \pm a$ называются тривиальными делителями числа a.

Целое число $p \in Z/\{0\}$ называется простым, если оно не является делителем единицы и не имеет других делителей, кроме тривиальных. В противном случае число $p \in Z/\{-1,0,1\}$ 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 $m \in N, m > 1$. 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 b.

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 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, зависящих от входных данных, то вероятностный алгоритм становится детерминированным).

Для проверки на простоту числа n вероятностным алгоритмом выбирают случайное число a, такое что $1 < a < n$, 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 $\frac{1}{2^t}$.

Выполнение лабораторной работы

Написаны программы на языке Julia.

Тест Ферма

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

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

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

```
using Random
n = 20
function is_prime(n::Int,k::Int=5)
    if n < 5 || n%2 == 0
        return false
    end
    for _ in 1:k
        a = rand(2:n-2)
        r = powermod(a,n-1,n)
        if r != 1
            return false
        end
    end
    return true
end
if is_prime(n)
    println("Число $n скорее всего простое")
else
    println("Число $n составное")
end
```

```
Число 20 составное
```

1. Пояснение програмного кода

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

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

1. Выбрать случайное целое число a, такое что $2 \leq a \leq n-2$.
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».

Вычисление символа Якоби

Необходим для **теста Соловея-Штрассена**

Вход: Нечетное целое число $n \geq 3$, целое число a, такое что $0 \leq a < n$.

Выход: Символ Якоби $\left(\frac{a}{n}\right)$.

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

```
function jacobi_symbol(a::Int, n::Int)
    g = 1
    if a == 0
        return 0
    end
    if a == 1
        return g
    end
    while a != 0
        k = 0
        while a % 2 == 0
            a ÷= 2
            k += 1
        end
        a1 = a
        if k % 2 == 0
            s = 1
        else
            if n % 8 == 1 || n % 8 == 7
                s = 1
            elseif n % 8 == 3 || n % 8 == 5
                s = -1
            end
        end
        g *= s
        if a1 == 1
```

```
            return g
        end
        if n % 4 == 3 && a1 % 4 == 3
            g = -g
        end
        a, n = n % a1, a1
    end
    return g
end

n = 19
a = rand(0:n-2)
result = jacobi_symbol(a, n)
println("Символ Якоби ($a/$n) = $result")
```

```
Символ Якоби (16/19) = 1
```

Тест Соловея-Штрассена

Тест Соловея–Штрассена основан на критерии Эйлера: нечетное число n является простым тогда и только тогда, когда для любого целого числа a, такого что $1 \leq a \leq n-1$ и 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 $t, n \in \mathbb{Z}$, где $n = p_1 p_2 \dots p_r$ и числа $p_i \neq 2$ простые (не обязательно различные). Символ Якоби $\left(\frac{m}{n}\right)$ определяется равенством

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

```
n = 27

function test_solovei_strassen(n::Int)
    a = rand(2:n-2)
    n_1 = (n-1)/2
    r = powermod(a, n_1, n) # power(a, (n-1)/2, n)
    if r != 1 && r !=(n-1)
        return false
    end
    s = jacobi_symbol(a,n)
    if r == s%n
        return true
    else
        return false
    end
end
```

```
if is_prime(n)
    println("Число $n скорее всего простое")
else
    println("Число $n не является простым")
end
```

```
Число 19 скорее всего простое
```

Алгоритм, реализующий тест Миллера–Рабина

Описание алгоритма:

Вход: Нечетное 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 нечетное.
2. Выбрать случайное целое число a, такое что $2 \leq a < n-2$.
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. Если $y = 1$, 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».

```
n = 19

function test_miller_rabin(n::Int, k::Int=5)
    s = 0
    r = n - 1
    while r % 2 == 0
        r ÷= 2
        s += 1
    end

    for _ in 1:k
        a = rand(2:n-2)
        y = powermod(a, r, n)
```

```
        if y == 1 || y == n - 1
            continue
        end

        for j in 1:s-1
            y = powermod(y, 2, n)
            if y == 1
                return false
            end
            if y == n - 1
                break
            end
        end

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

if test_miller_rabin(n)
    println("Число $n вероятно простое")
else
    println("Число $n составное")
end
```

```
Число 19 вероятно простое
```

Выводы

В ходе выполнения лабораторной работы №5 были изучены и реализованы на языке Julia вероятностные алгоритмы проверки чисел на простоту, такие как тест Ферма, тест Соловея — Штрассена и тест Миллера — Рабина. Эти алгоритмы играют важную роль в криптографии и других областях, где необходимо быстро определять простоту чисел. Вероятностные алгоритмы позволяют получать ответ с высокой степенью вероятности, однако, в отличие от детерминированных методов, не гарантируют абсолютной точности. Такой подход позволяет находить простые числа более эффективно, что особенно актуально при работе с большими числами. Алгоритм Миллера-Рабина является наиболее используемым в современных системах, потому что дает наибольшую точность.

Список литературы