Лабораторная работа №5

Вероятностные алгоритмы проверки чисел на простоту

Тазаева Анастасия Анатольевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Ознакомиться с алгоритмами проверки чисел на простоту. Реализовать их.

2 Задание

Реализовать на языке программирования Julia:

1. Алгоритм, реализующий тест Ферма.
2. Алгоритм вычисления числа Якоби.
3. лгоритм, реализующий тест Соловея-Штрассена.
4. Алгоритм, реализующий тест Миллера-Рабина.

3 Теоретическое введение

3.1 Алгоритм, реализующий тест Ферма

Вход. Нечетное целое число $n >= 5$

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

1. Выбрать случайное целое число $a, 2 \leq a \leq n-2$.
2. Вычислить $r = a^{n-1}(mod n)$.
3. При $r = 1$ результат: «Число n, вероятно. простое». В противном случае результат: «Число n составное».

3.2 Алгоритм вычисления числа Якоби

Вход. Нечетное целое число $n >= 3$, целое число $0 \leq a < n$

Выход. Символ Якоби $(\frac{a}{n})$

1. Положить $g = 1$
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 нечетное
5. При четном k положить $s = -1$, при нечетном k положить $s = 1$, если $n = +-1(mod 8)$; положить $s = -1$, если $n = +-3(mod 8)$
6. При $a_1 = 1$ результат: $g \cdot s$

7. Если $n = 3(mod4)$ и $a_1 = 3(mod4)$, то $s = -s$ 8.Положить $a = n(moda_1), n = a_1, g = g \cdot s$ и вернуться на шаг 2.

3.3 Алгоритм, реализующий тест Соловея-Штрассена

Вход. Нечетное целое число $n >= 5$

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

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

3.4 Алгоритм, реализующий тест Миллера-Рабина

Вход. Нечетное целое число $n >= 5$

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 $a, 2 \leq a \leq n-2$.
3. Вычислить $y = a^r(modn)$.
4. При $y \neq 1$ и $y \neq n-1$ выполнить следующие действия.
5. Положить $j = 1$.
6. Если $j \leq s-1$ и $y \neq n-1$, то 1. Положить $y = y^2(modn)$ 2. При $y = 1$ результат: «Число n составное». 3. Положить $j = j+1$
7. При $y \neq n-1$ результат: «Число n составное».
8. Результат: «Число n, вероятно. простое».

4 Выполнение лабораторной работы

4.1 Алгоритм, реализующий тест Ферма

Написан программный код на языке Julia [1]:

```
function test_Ferma(n)
    if n < 5
        return "incorrrect n.please try again \n"
    end
    a = rand(2:n-2)
    r = powermod(a, n-1,n)
    if r == 1
        return "4islo " * string(n) * ", veroyatno, prostoe\n"
    else
        return "4islo " * string(n) * " sostavnoe \n"
    end
end
```

Получен следующий результат выполнения программного кода (рис. 4.1).

```
print(test_Ferma(43))
print(test_Ferma(44))
print(test_Ferma(4))
```

```
4islo 43, veroyatno, prostoe
4islo 44 sostavnoe
incorrrect n.please try again
```

Рисунок 4.1: Алгоритм, реализующий тест Ферма. Пример отработки

4.2 Алгоритм вычисления числа Якоби

Написан программный код на языке Julia [1]:

```
function simvol_Yakobi(n,a)
    if n < 3 || a >= n || a < 0
        return "\nincorrrect n,a.please try again\n"
    end
    g = 1
    a1 = 0
    k = 0
    s = 0
    while a1 != 1
        if 1 == 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
```

Получен следующий результат выполнения программного кода (рис. 4.2).

```
print(simvol_Yakobi(44, 40))
print(simvol_Yakobi(44, 50))
```

```
0
incorrrect n,a.please try again
```

Рисунок 4.2: Алгоритм вычисления числа Якоби. Пример отработки

4.3 Алгоритм, реализующий тест Соловея-Штрассена

Написан программный код на языке Julia [1]:

```
function test_Soloveya_Shtrassena(n)
    if n < 5
        return "incorrrect n.please try again"
    end
    a = rand(2:n-2)
    r = powermod(a, round(Int64, (n-1)/2), n)
    if r != 1 && r != n-1
        return "4islo " * string(n) * " sostavnoe \n"
    else
        s = simvol_Yakobi(n,a)
        if r == s && r != NaN
            return "4islo " * string(n) * " sostavnoe \n"
        end
        return "4islo " * string(n) * ", veroyatno, prostoe\n"
    end
end
```

Получен следующий результат выполнения программного кода (рис. 4.3).

```
print(test_Soloveya_Shtrassena(43))
print(test_Soloveya_Shtrassena(44))
print(test_Soloveya_Shtrassena(4))
```

```
4islo 43, veroyatno, prostoe
4islo 44 sostavnoe
incorrrect n.please try again
```

Рисунок 4.3: Алгоритм, реализующий тест Соловея-Штрассена. Пример отработки

4.4 Алгоритм, реализующий тест Миллера-Рабина

Написан программный код на языке Julia [1]:

```
function test_Millera_Robina(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
                "4islo " * string(n) * " sostavnoe \n"
            end
            j += 1
        end
        if y != n-1
            "4islo " * string(n) * " sostavnoe \n"
        else
            "4islo " * string(n) * ", veroyatno, prostoe\n"
        end
    else
        return "4islo " * string(n) * ", veroyatno, prostoe\n"
    end
end
```

Получен следующий результат выполнения программного кода (рис. 4.4).

```
[80]:
test_Millera_Robina(44)

[80]:
"4islo 44 sostavnoe \n"

[81]:
test_Millera_Robina(43)

[81]:
"4islo 43, veroyatno, prostoe\n"
```

Рисунок 4.4: Алгоритм, реализующий тест Миллера-Рабина. Пример отработки

5 Выводы

В ходе лабораторной работы реализованы на языке программирования Julia:

1. Алгоритм, реализующий тест Ферма.
2. Алгоритм вычисления числа Якоби.
3. Алгоритм, реализующий тест Соловея-Штрассена.
4. Алгоритм, реализующий тест Миллера-Рабина.

Список литературы

[1] *Julia 1.10 Documentation*. Англ. 2024. URL: https://docs.julialang.org/en/v1/base/strings/.