Лабораторная работа №5

Математические основы защиты информации и информационной безопасности

Леонтьева Ксения Андреевна | НПМмд-02-23

Содержание

Список иллюстраций

1 Цель работы

Реализовать на языке программирования вероятностные алгоритмы проверки чисел на простоту.

2 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**.

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 $(1 < a < n)$ 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. При t независимых выполнений теста вероятность того, что составное число n будет t раз объявлено простым (вероятность ошибки), не превосходит $\frac{1}{2^t}$.

Более подробно см. в [1], [2], [3], [4].

3 Выполнение лабораторной работы

Тест Ферма реализуем по следующей схеме:

На вход подается нечетное целое число $n \geq 5$.

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

Код программы (рис. 3.1).

```
import numpy as np
import math
```

```
def Fermat(n):
    a = np.random.randint(2, (n - 2) + 1)
    r = (a ** (n - 1)) % n
    if r == 1:
        return "Число " + str(n) + ", вероятно, простое"
    else:
        return "Число " + str(n) + " составное"
```

Рис. 3.1: Тест Ферма

Вычисление символа Якоби реализуем по следующей схеме:

На вход подаются нечетное целое число $n \geq 3$ и целое число $a, 0 \leq a < n$.

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 нечетное.

5. При четном k положить $s \leftarrow 1$, при нечетном k положить $s \leftarrow 1$, если $n \equiv \pm 1 (mod\, 8)$; положить $s \leftarrow -1$, если $n \equiv \pm 3 (mod\, 8)$.

6. При $a_1 = 1$ результат: gs.

7. Если $n \equiv 3 (mod\, 4)$ и $a_1 \equiv 3 (mod\, 4)$, то $s \leftarrow -s$.

8. Положить $a \leftarrow n (mod\, a_1)$, $n \leftarrow a_1$, $g \leftarrow gs$ и вернуться на шаг 2.

Код программы (рис. 3.2).

```
def Jacobi(a,n):
    g = 1
    s = 0
    while (a != 0) and (a != 1):
        a1 = a
        k = 0
        if a1 % 2 != 0:
            k = 0
        if a1 % 2 == 0:
            while a1 % 2 == 0:
                a1 = int(a1 / 2)
            while a != (2 ** k) * a1:
                k = k + 1
        if k % 2 == 0:
            s = 1
        else:
            if (n % 8 == 1 % 8) or (n % 8 == -1 % 8):
                s = 1
            elif (n % 8 == 3 % 8) or (n % 8 == -3 % 8):
                s = -1
        if a1 == 1:
            return g * s

        if (n % 4 == 3 % 4) and (a1 % 4 == 3 % 4):
            s = -s
        a = n % a1
        n = a1
        g = g * s

    if a == 0:
        return 0
    else:
        return g
```

Рис. 3.2: Вычисление символа Якоби

Тест Соловэя-Штрассена реализуем по следующей схеме:

На вход подается нечетное целое число $n \geq 5$.

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

Код программы (рис. 3.3).

```
def S_SH(n):
    a = np.random.randint(2, (n - 2) + 1)
    r = (a ** ((n - 1) / 2)) % n
    if (r != 1) and (r != n - 1):
        return "Число " + str(n) + " составное"
    s = Jacobi(a,n)
    if r != s % n:
        return "Число " + str(n) + " составное"
    return "Число " + str(n) + ", вероятно, простое"
```

Рис. 3.3: Тест Соловэя-Штрассена

Тест Миллера-Рабина реализуем по следующей схеме:
На вход подается нечетное целое число $n \geq 5$.

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 $y \leftarrow a^r(mod\, n)$.
4. При $y \neq 1$ и $y \neq n-1$ выполнить следующие действия.

4.1. Положить $j \leftarrow 1$.
4.2. Если $j \leq s-1$ и $y \neq n-1$, то

4.2.1. Положить $y \leftarrow y^2 (mod\, n)$.

4.2.2. При $y = 1$ результат: “Число n составное”.

4.2.3. Положить $j \leftarrow j + 1$.

4.3. При $y \neq n - 1$ результат: “Число n составное”.

5. Результат: “Число n, вероятно, простое”.

Код программы (рис. 3.4).

```
def M_R(n):
    r = n - 1
    s = 0
    if r % 2 != 0:
        s = 0
    if r % 2 == 0:
        while r % 2 == 0:
            r = int(r / 2)
        while n - 1 != (2 ** s) * r:
            s = s + 1
    a = np.random.randint(2, (n - 2) + 1)
    y = (a ** r) % n
    if (y != 1) and (y != n - 1):
        j = 1
        if (j <= s - 1) and (y != n - 1):
            y = (y ** 2) % n
            if y == 1:
                return "Число " + str(n) + " составное"
            j = j + 1
        if y != n - 1:
            return "Число " + str(n) + " составное"
    return "Число " + str(n) + ", вероятно, простое"
```

Рис. 3.4: Тест Миллера-Рабина

В итоге были получены следующие результаты (рис. 3.5) - (рис. 3.8).

```
for n in range (5, 50 , 2):
    print(Fermat(n))
    print(S_SH(n))
    print(M_R(n))
    print('-------')
```

```
Число 5, вероятно, простое
Число 5, вероятно, простое
Число 5, вероятно, простое
-------
Число 7, вероятно, простое
Число 7, вероятно, простое
Число 7, вероятно, простое
-------
Число 9 составное
Число 9 составное
Число 9 составное
-------
Число 11, вероятно, простое
Число 11, вероятно, простое
Число 11, вероятно, простое
-------
Число 13, вероятно, простое
Число 13, вероятно, простое
Число 13, вероятно, простое
```

Рис. 3.5: Результаты выполнения алгоритмов

```
Число 15 составное
Число 15 составное
Число 15 составное
-------
Число 17, вероятно, простое
Число 17, вероятно, простое
Число 17 составное
-------
Число 19, вероятно, простое
Число 19, вероятно, простое
Число 19, вероятно, простое
-------
Число 21 составное
Число 21 составное
Число 21 составное
-------
Число 23, вероятно, простое
Число 23, вероятно, простое
Число 23, вероятно, простое
-------
Число 25 составное
Число 25, вероятно, простое
Число 25 составное
```

Рис. 3.6: Результаты выполнения алгоритмов

```
Число 27 составное
Число 27 составное
Число 27 составное
-------
Число 29, вероятно, простое
Число 29, вероятно, простое
Число 29, вероятно, простое
-------
Число 31, вероятно, простое
Число 31, вероятно, простое
Число 31, вероятно, простое
-------
Число 33 составное
Число 33 составное
Число 33 составное
-------
Число 35 составное
Число 35 составное
Число 35 составное
-------
Число 37, вероятно, простое
Число 37 составное
Число 37, вероятно, простое
```

Рис. 3.7: Результаты выполнения алгоритмов

```
Число 39 составное
Число 39 составное
Число 39 составное
-------
Число 41, вероятно, простое
Число 41, вероятно, простое
Число 41, вероятно, простое
-------
Число 43, вероятно, простое
Число 43, вероятно, простое
Число 43, вероятно, простое
-------
Число 45 составное
Число 45 составное
Число 45 составное
-------
Число 47, вероятно, простое
Число 47 составное
Число 47, вероятно, простое
-------
Число 49 составное
Число 49 составное
Число 49 составное
```

Рис. 3.8: Результаты выполнения алгоритмов

4 Выводы

В ходе выполнения данной лабораторной работы были реализованы вероятностные алгоритмы проверки чисел на простоту.

Список литературы

1. Тест Ферма [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82_%D0%A4%D0%B5%D1%80%D0%BC%D0%B0.
2. Символ Якоби [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/Jacobi_symbol.
3. Тест Соловэя-Штрассена [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82_%D0%A1%D0%BE%D0%BB%D0%BE%D0%B2%D0%B5%D1%8F_%E2%80%94_%D0%A8%D1%82%D1%80%D0%B0%D1%81%D1%81%D0%B5%D0%BD%D0%B0.
4. Тест Миллера-Рабина [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82_%D0%9C%D0%B8%D0%BB%D0%BB%D0%B5%D1%80%D0%B0_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D1%87%D0%B8%D1%81%D0%B5%D0%BB).