Отчёт по лабораторной работе №5. Вероятностные алгоритмы проверки чисел на простоту

Дисциплина: Математические основы защиты информации и информационной безопасности

Студент: Аронова Юлия Вадимовна, 1032212303
Группа: НФИмд-01-21
Преподаватель: д-р.ф.-м.н., проф. Кулябов Дмитрий Сергеевич

9 декабря, 2021, Москва

Цели и задачи работы

Целью данной лабораторной работы является ознакомление с тремя вероятностными алгоритмами проверки чисел на простоту, а также их последующая программная реализация.

Задачи: Рассмотреть и реализовать на языке программирования Python:

1. Алгоритм, реализующий тест Ферма;
2. Алгоритм, реализующий тест Соловея-Штрассена (включающий в себя алгоритм вычисления символа Якоби);
3. Алгоритм, реализующий тест Миллера-Рабина.

Теоретическое введение

Общие понятия

Простые числа

Натуральное $p > 1$ называется *простым*, если оно делится только на 1 и на p. Целое $a > 1$, имеющее другие делители кроме a и 1, называется *составным*.

Существует два типа критериев простоты: детерминированные и вероятностные.

Детерминированные тесты действуют по одной и той же схеме и гарантированно позволяют доказать, что тестируемое число – простое.

Вероятностные тесты не дают гарантированного ответа. Их можно эффективно использовать для тестирования отдельных чисел, однако их результаты с некоторой вероятностью могут быть неверными.

Тест Ферма

Малая теорема Ферма

Для простого числа p и $a : 1 \leq a \leq p-1$, выполняется сравнение $a^{p-1} \equiv 1 \pmod{p}$.

Если для нечётного n $\exists a \in \mathbb{Z} : 1 \leq a < n$, НОД$(a, n) = 1$ и $a^{n-1} \not\equiv 1 \pmod{n}$, то число n составное.

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”.

Figure 1: Алгоритм, реализующий тест Ферма

Символ Якоби (1 / 2)

Пусть p – простое число, $p > 2$, $a \in \mathbb{Z}$, НОД$(a, p) = 1$. Число a называется **квадратичным вычетом** по модулю p, если уравнение $x^2 \equiv a \pmod{p}$ разрешимо.

Символ Лежандра $\left(\frac{a}{p}\right)$ (где $a \in \mathbb{Z}$) равен: $+1$, если a – квадратичный вычет по модулю p; -1, если a – квадратичный невычет; и 0, если $a \equiv 0 \pmod{p}$.

Если $m \in \mathbb{N}$, m – нечётное составное число и $m = \prod_{j=1}^{k} p_j^{\alpha^j}$ есть разложение m на простые множители, то для $a \in \mathbb{Z}$ **символ Якоби** $\left(\frac{a}{m}\right)$ определяется равенством

$$\left(\frac{a}{m}\right) = \prod_{j=1}^{k} \left(\frac{a}{p_j}\right)^{\alpha^j}$$

Символ Якоби (2 / 2)

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 чётном 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.

Figure 2: Алгоритм вычисления символа Якоби

Тест Соловея-Штрассена

Критерий Эйлера

Нечётное число n является простым тогда и только тогда, когда для любого целого числа a, $2 \le a \le n-1$, взаимно простого с n, выполняется: $a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) \pmod{n}$, где $\left(\frac{a}{n}\right)$ – символ Якоби.

Вход. Нечётное целое число $n \ge 5$.

Выход. “Число n, вероятно, простое” или “Число n составное”.

1. Выбрать случайное число a, $2 \le a < n-2$.
2. Вычислить $r \leftarrow a^{\frac{n-1}{2}} \pmod{n}$.
3. При $r \ne 1$ и $r \ne n-1$ результат: “Число n составное”.
4. Вычислить символ Якоби $s \leftarrow \left(\frac{a}{n}\right)$.
5. При $r \not\equiv s \pmod{n}$ результат: “Число n составное”. В противном случае результат: “Число n, вероятно, простое”.

Figure 3: Алгоритм, реализующий тест Соловея-Штрассена

Тест Миллера-Рабина (1 / 2)

Пусть число n – нечётное и $n-1=2^s r$, где r – нечётное. Если n – простое, то для любого $2 \leq a \leq n-1$ выполняется хотя бы одно из условий:

1. $a^r \equiv 1 \pmod{n}$;
2. $\exists d < s : a^{2^d r} \equiv -1 \pmod{n}$.

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:

    4.1. Положить $j \leftarrow 1$.

    4.2. Пока $j \leq s - 1$ и $y \neq n - 1$:

        4.2.1. Положить $y \leftarrow y^2 \pmod{n}$

        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”.

Figure 4: Алгоритм, реализующий тест Миллера-Рабина

Ход выполнения и результаты

Тест Ферма. Реализация

```python
import numpy as np


def equal_by_modulo(a, b, m):
    return (True if (a - b) % m == 0 else False)
```

```python
def fermat_algorithm(n):
    if n < 5 or n % 2 == 0: return "Некорректное число n"
    a = np.random.randint(2, n - 1) # шаг 1
    r = (a ** (n - 1)) % n # шаг 2
    if r == 1: # шаг 3
        return "Число {}, вероятно, простое".format(n)
    else:
        return "Число {} составное".format(n)
```

Тест Ферма. Результаты

```
print(fermat_algorithm(37), ";", fermat_algorithm(239), ";", fermat_algorithm(877))
print(fermat_algorithm(63), ";", fermat_algorithm(755), ";", fermat_algorithm(1111111))
[3] ✓ 8.1s
... Число 37, вероятно, простое ; Число 239, вероятно, простое ; Число 877, вероятно, простое
    Число 63 составное ; Число 755 составное ; Число 1111111 составное
```

Figure 5: Примеры проверки чисел на простоту посредством программной реализации теста Ферма

Алгоритм вычисления символа Якоби. Реализация

```
def jacobi_symbol(a, n, g = 1): # шаг 1: значение g по умолчанию
    if a == 0: return 0 # шаг 2
    if a == 1: return g # шаг 3
    k = 0 # шаг 4
    while a % (2 ** k) == 0:
        k += 1
    k -= 1; a1 = int(a / (2 ** k))
    s = 1 # шаг 5
    if k%2==1 and (equal_by_modulo(n,3,8) or equal_by_modulo(n,-3,8)):
        s = -1
    if a1 == 1: return g * s # шаг 6
    if equal_by_modulo(n, 3, 4) and equal_by_modulo(a1, 3, 4): # шаг 7
        s = -s
    a = n % a1; n = a1; g = g * s # шаг 8
    return jacobi_symbol(a, n, g)
```

Алгоритм вычисления символа Якоби. Результаты

```
print("Символ Якоби ({}/{}) = {}".format(1001, 9907, jacobi_symbol(1001, 9907)))
print("Символ Якоби ({}/{}) = {}".format(19, 45, jacobi_symbol(19, 45)))
print("Символ Якоби ({}/{}) = {}".format(219, 383, jacobi_symbol(219, 383)))
```

[5] ✓ 0.4s

```
Символ Якоби (1001/9907) = -1
Символ Якоби (19/45) = 1
Символ Якоби (219/383) = 1
```

Figure 6: Примеры вычисления символа Якоби с помощью реализованной функции

Тест Соловея-Штрассена. Реализация

```
def solovay_strassen_algorithm(n):
    if n < 5 or n % 2 == 0: return "Некорректное число n"
    a = np.random.randint(2, n - 2) # шаг 1
    r = (a ** int((n - 1) / 2)) % n # шаг 2
    if r != 1 and r != (n - 1): # шаг 3
        return "Число {} составное".format(n)
    s = jacobi_symbol(a, n) # шаг 4
    if not equal_by_modulo(r, s, n): # шаг 5
        return "Число {} составное".format(n)
    else:
        return "Число {}, вероятно, простое".format(n)
```

Тест Соловея-Штрассена. Результаты

```
print(solovay_strassen_algorithm(37), ";", solovay_strassen_algorithm(239), ";", solovay_strassen_algorithm(877))
print(solovay_strassen_algorithm(63), ";", solovay_strassen_algorithm(755), ";", solovay_strassen_algorithm(1111111))
```

[7] ✓ 2.7s

```
Число 37, вероятно, простое ; Число 239, вероятно, простое ; Число 877, вероятно, простое
Число 63 составное ; Число 755 составное ; Число 1111111 составное
```

Figure 7: Примеры проверки чисел на простоту посредством программной реализации теста Соловея-Штрассена

Тест Миллера-Рабина. Реализация

```
def miller_rabin_algorithm(n): <...>
    s = 0 # шаг 1
    while (n - 1) % (2 ** s) == 0: s += 1
    s -= 1
    r = int((n - 1) / (2 ** s))
    a = np.random.randint(2, n - 2) # шаг 2
    y = (a ** r) % n # шаг 3
    if y != 1 and y != (n - 1): # шаг 4
        j = 1 # шаг 4.1
        while j <= (s - 1) and y != (n - 1): # шаг 4.2
            y = (y ** 2) % n # шаг 4.2.1
            if y == 1: return "Число {} составное".format(n) # шаг 4.2.2
            j += 1 # шаг 4.2.3
        if y != (n - 1): return "Число {} составное".format(n) # шаг 4.3
    return "Число {}, вероятно, простое".format(n) # шаг 5
```

Тест Миллера-Рабина. Результаты

```
print(miller_rabin_algorithm(37), ";", miller_rabin_algorithm(239), ";", miller_rabin_algorithm(877))
print(miller_rabin_algorithm(63), ";", miller_rabin_algorithm(755), ";", miller_rabin_algorithm(1111111))
```

[9] ✓ 2.4s

```
... Число 37, вероятно, простое ; Число 239, вероятно, простое ; Число 877, вероятно, простое
    Число 63 составное ; Число 755 составное ; Число 1111111 составное
```

Figure 8: Примеры проверки чисел на простоту посредством программной реализации теста Миллера-Рабина

Заключение

Таким образом, была достигнута цель, поставленная в начале лабораторной работы: было проведено краткое знакомство с алгоритмом вычисления символа Якоби и тремя вероятностными алгоритмами проверки чисел на простоту – на основе теста Ферма, теста Соловея-Штрассена, теста Миллера-Рабина, – после чего все четыре алгоритма были успешно реализованы на языке программирования **Python**.

Спасибо за внимание