лабораторной работе №5
Вероятностные тесты на простоту чисел
Студент: Хамза хуссен

---

1. Цель исследования

Ознакомление с принципами работы, реализацией и сравнительным анализом вероятностных алгоритмов проверки чисел на простоту: теста Ферма, теста Соловэя-Штрассена и теста Миллера-Рабина.

---

2. Краткая теоретическая справка

В современных криптографических системах часто требуется генерация больших простых чисел. В связи с этим задача эффективной проверки числа на простоту является крайне актуальной.

Критерии простоты делятся на два класса:

- **Детерминированные:** Позволяют строго доказать простоту числа, но для больших чисел могут быть вычислительно сложными. Чаще используются для целенаправленной генерации простых чисел.
- **Вероятностные:** Оперируют понятием «вероятно простое число». Не дают абсолютной гарантии, но позволяют быстро тестировать произвольные числа, причём вероятность ошибки может быть сделана сколь угодно малой за счёт многократного повторения теста с разными параметрами. На практике именно эти алгоритмы часто применяются на первом этапе проверки.

В основе всех рассматриваемых тестов лежит использование случайно выбранных «свидетелей» (witness) и проверка определенных условий, вытекающих из свойств простых чисел.

2.1 Тест Ферма (на основе малой теоремы Ферма)

- **Входящие данные:** Нечётное целое число $n \geq 5$.
- **Исход:** «Число n, вероятно, простое» или «Число n составное».
- **Алгоритм:**
    1. Случайным образом выбрать основание a, где $2 \leq a \leq n-2$.
    2. Вычислить значение $r = a^{n-1} (\mathrm{mod} n)$.
    3. Если $r = 1$, вернуть «Вероятно простое». Иначе — «Составное».

2.2 Тест Соловэя-Штрассена

- **Входящие данные:** Нечётное целое число $n \geq 5$.
- **Исход:** «Число n, вероятно, простое» или «Число n составное».
- **Алгоритм:**
    1. Случайным образом выбрать основание a, где $2 \leq a \leq n-2$.
    2. Вычислить $r = a^{(n-1)/2} (\mathrm{mod} n)$.
    3. Если $r \neq 1$ и $r \neq n-1$, вернуть «Составное».
    4. Вычислить символ Якоби $s = \left(\frac{a}{n}\right)$.
    5. Если $r \equiv s (\mathrm{mod} n)$, вернуть «Вероятно простое». Иначе — «Составное».

2.3 Тест Миллера-Рабина

- **Входящие данные:** Нечётное целое число $n \geq 5$.
- **Исход:** «Число n, вероятно, простое» или «Число n составное».
- **Алгоритм:**
    1. Представить $n-1$ в виде $n-1 = 2^s \cdot r$, где r — нечётное.
    2. Случайным образом выбрать основание a, где $2 \leq a \leq n-2$.
    3. Вычислить $y = a^r (\mathrm{mod} n)$.
    4. Если $y \neq 1$ и $y \neq n-1$, то:
        - Присвоить $j = 1$.
        - Пока $j \leq s-1$ и $y \neq n-1$, выполнять:
            - $y = y^2 (\mathrm{mod} n)$.
            - Если $y = 1$, вернуть «Составное».
            - $j = j + 1$.
        - Если $y \neq n-1$, вернуть «Составное».
    5. Вернуть «Вероятно простое».

---

3. Практическая реализация и анализ

(В данном разделе обычно следует описание реализованной программы, таблицы с результатами тестирования чисел, сравнение времени работы алгоритмов и анализ их надёжности.)

4. Выводы

(Здесь подводятся итоги: отмечается, что тест Миллера-Рабина является наиболее надёжным и широко используемым на практике, тогда как тест Ферма, будучи самым быстрым, пропускает наибольшее количество составных чисел (чисел Кармайкла). Тест Соловэя-Штрассена занимает промежуточное

положение, но вычисление символа Якоби делает его несколько менее эффективным по сравнению с тестом Миллера-Рабина. Делается вывод о целесообразности использования комбинации этих тестов или последовательного применения теста Миллера-Рабина с разными основаниями для достижения высокой достоверности при приемлемых временных затратах.)

5.Выполнение работы

5.1.Реализация алгоритмов на языке Python

```
import random

def fermat_test(n, trials=5):
    if n < 5:
        return False
    if n % 2 == 0:
        return False

    for _ in range(trials):
        a = random.randint(2, n - 2)
        if pow(a, n - 1, n) != 1:
            return False
    return True

def jacobi_symbol(a, n):
    if a == 0:
        return 0
    if a == 1:
        return 1

    result = 1
    if a < 0:
        a = -a
        if n % 4 == 3:
            result = -result

    while a != 0:
        while a % 2 == 0:
            a //= 2
            if n % 8 in (3, 5):
                result = -result
        a, n = n, a
        if a % 4 == 3 and n % 4 == 3:
            result = -result
        a %= n

    return result if n == 1 else 0

def solovay_strassen_test(n, trials=5):
    if n < 5:
        return False
```

```
    if n % 2 == 0:
        return False

    for _ in range(trials):
        a = random.randint(2, n - 2)
        x = pow(a, (n - 1) // 2, n)
        if x == 0 or x != (n + jacobi_symbol(a, n)) % n:
            return False
    return True

def miller_rabin_test(n, trials=5):
    if n < 5:
        return False
    if n % 2 == 0:
        return False

    s, d = 0, n - 1
    while d % 2 == 0:
        d //= 2
        s += 1

    for _ in range(trials):
        a = random.randint(2, n - 2)
        x = pow(a, d, n)

        if x == 1 or x == n - 1:
            continue

        composite = True
        for _ in range(s - 1):
            x = (x * x) % n
            if x == n - 1:
                composite = False
                break

        if composite:
            return False

    return True

def test_number(n):
    print(f"Тестируем число: {n}")
    print(f"Ферма: {'Простое' if fermat_test(n) else 'Составное'}")
    print(f"Соловэй-Штрассен: {'Простое' if solovay_strassen_test(n) else
'Составное'}")
    print(f"Миллер-Рабин: {'Простое' if miller_rabin_test(n) else
'Составное'}")
    print()

if __name__ == "__main__":
    test_numbers = [17, 25, 97, 100, 561]
```

```
    for num in test_numbers:
        test_number(num)
```

Контрольный пример

```
Тестируем число: 17
Ферма: Простое
Соловэй-Штрассен: Простое
Миллер-Рабин: Простое

Тестируем число: 25
Ферма: Составное
Соловэй-Штрассен: Составное
Миллер-Рабин: Составное

Тестируем число: 97
Ферма: Простое
Соловэй-Штрассен: Простое
Миллер-Рабин: Простое


Тестируем число: 100
Ферма: Составное
Соловэй-Штрассен: Составное
Миллер-Рабин: Составное

Тестируем число: 561
Ферма: Составное
Соловэй-Штрассен: Составное
Миллер-Рабин: Составное
```