МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»**

Кафедра теоретических основ компьютерной безопасности и криптографии

**Проверка чисел на простоту**

ОТЧЁТ

ПО ДИСЦИПЛИНЕ

«ТЕОРЕТИКО-ЧИСЛОВЫЕ МЕТОДЫ В КРИПТОГРАФИИ»

студента 5 курса 531 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Мызникова Сергея Анатольевича

|  |  |  |
| --- | --- | --- |
| Преподаватель, профессор | \_\_\_\_\_\_\_\_\_\_ | В.А. Молчанов |
|  | подпись, дата |  |

Саратов 2024

**СОДЕРЖАНИЕ**

[1 Постановка задачи 3](#_Toc181535770)

[2 Теоретические сведения 4](#_Toc181535771)

[2.1 Тест Ферма 4](#_Toc181535772)

[2.2 Тест Соловея-Штрассена 6](#_Toc181535773)

[2.3 Тест Миллера-Рабина 7](#_Toc181535774)

[3 Результаты работы 7](#_Toc181535775)

[3.1 Алгоритмы: описание и временная сложность 7](#_Toc181535776)

[3.1.1 Тест Ферма 7](#_Toc181535777)

[3.1.2 Тест Соловея-Штрассена 8](#_Toc181535778)

[3.1.3 Тест Миллера-Рабина 8](#_Toc181535779)

[3.2 Тестирование алгоритмов 9](#_Toc181535780)

[ЗАКЛЮЧЕНИЕ 10](#_Toc181535781)

[ПРИЛОЖЕНИЕ А 11](#_Toc181535782)

# **1 Постановка задачи**

Цель работы — изучение основных методов проверки простоты чисел и их программная реализация.

Порядок выполнения работы:

1. Рассмотреть тест Ферма проверки чисел на простоту и привести его программную реализацию.
2. Рассмотреть тест Соловея-Штрассе на проверки чисел на простоту и привести его программную реализацию.
3. Рассмотреть тест Миллера-Рабина и привести его программную реализацию.

# **2 Теоретические сведения**

## **2.1 Тест Ферма**

Плотность распределения простых чисел:

1 − 10 - 40%, 1 − 100 - 25%, 1 − 1000 - 17%, 1 – 105 - <10%.

Вероятностный алгоритм проверки числа n на простоту использует необходимые условия простоты P(a):

1. выбирается случайным образом и проверяется выполнимость теста – некоторого условного алгоритма,
2. если тест не проходит, то есть не выполняется, то вывод “число n, составное”
3. если тест проходит, то есть выполняется , то вывод “число n, вероятно, простое”

Если событие – “число n простое” имеет вероятность , то вероятность ошибки – получить для составного числа вывод “число возможно простое” и при t повторах теста вероятность ошибки .

Малая теорема Ферма. Если 𝑝 – простое число и – произвольное целое число, такое что 1 ≤ 𝑎 ≤ 𝑝 − 1, выполняется сравнение:

**Определение.** Число называется псевдопростым по основанию , если для чисел и выполняется сравнение .

**Лемма.** Пусть нечетное число. Тогда выполняются утверждения:

а) множество всех , относительно которых является псевдопростым, образует подгруппу в ;

б) если не является псевдопростым хотя бы по одно  
 му основанию , то не является псевдопростым относи  
 тельно по крайней мере половины чисел из .

**Вероятность успеха** – вероятность получить “Число составное” для составного числа равна .

Возможны 3 случая:

1) Число простое и тест всегда дает ответ “Число , вероятно, простое”

2) Число составное и , тогда тест дает ответ “Число n составное” с вероятность успеха

3) Число составное и , тогда тест дает ответ “Число составное” с вероятностью .

В случае 2) при повторах теста вероятность успеха

**Определение.** Нечетное составное число называется числом Кармайкла, если (и, значит, вероятность успеха Алгоритма Ферма будет )

**Лемма.** Для любого числа Кармайкла справедливы утверждения:

1) для простых различных чисел

2)

Плотность распределения чисел Кармайкла:

- 16 чисел: 561, 1105, 1729, …;

- 2163 чисел.

## **2.2 Тест Соловея-Штрассена**

**Критерий Эйлера.** Нечетное число *n* является простым тогда и только тогда, когда для любого a выполняется свойство:

.

простое число тогда и только тогда, когда , где

**Определение**: Число называется эйлеровым псевдопростым по основанию , если для чисел , выполняется сравнение .

**Лемма.** Пусть нечетное число. Тогда выполняются утверждения:

а) множество всех , относительно которых является эйлеровым псевдопростым, образует подгруппу в ;

б) если – составное число, то не является псевдопростым эйлеровым числом относительно по крайней мере половины чисел из .

**Вероятность успеха** теста Соловея-Штрассена для составного числа равна

Возможны 2 случая:

1. Число простое и тест всегда дает ответ “Число , вероятно, простое”
2. Число составное и тест дает ответ “Число составное” с вероятность успеха

В случае 2) при повторах теста вероятность успеха

## **2.3 Тест Миллера-Рабина**

**Критерий Миллера.** Пусть *n* нечётное и *n* – 1 = для нечетного *t*. Тогда *n* является простым тогда и только тогда, когда для любого *a*  выполняется свойство:

.

простое число тогда и только тогда, когда , где

**Определение.** Число псвдопростое по основанию называется сильно псевдопростым по основанию , если выполняется одно из условий:

1. Либо
2. Либо для некоторого

Для составного числа выполняется и, значит, вероятность успеха теста Миллера-Рабина для составного числа равна . При повторах теста вероятность успеха

# **3 Результаты работы**

## **3.1 Алгоритмы: описание и временная сложность**

### **3.1.1 Тест Ферма**

Вход: Нечетное число n, количество итераций k

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

Шаг 1: Выбрать случайно и вычислить

Если , то ответ “Число составное”;

Шаг 2: Если d = 1, то проверить условие: . Если оно не выполнено, то ответ “Число составное”, иначе “Число n, вероятно, простое”.

**Временная сложность алгоритма**. O(log3n)

### **3.1.2 Тест Соловея-Штрассена**

Вход: Нечетное число n, количество итераций k

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

Шаг 1: Выбрать случайно и вычислить

Если , то ответ “Число составное”;

Шаг 2: Если d = 1, то проверить условие: . Если оно не выполнено, то ответ “Число составное”, иначе “Число n, вероятно, простое”.

**Временная сложность алгоритма**. O(log3n)

### **3.1.3 Тест Миллера-Рабина**

Вход: Нечетное число n, количество итераций k

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

Шаг 1: Выбрать случайно и вычислить

Если , то ответ “Число составное”;

Шаг 2: Если d = 1, то вычислить для значений . Если или для некоторого , то ответ “Число n, вероятно, простое”. В противном случае ответ “Число n составное”.

**Временная сложность алгоритма**. O(log3n)

## **3.2 Тестирование алгоритмов**

Тест Ферма

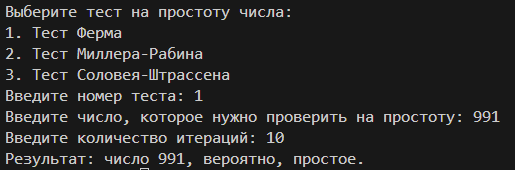


Рисунок 1. Тестирование проверки простоты с помощью теста Ферма

Тест Соловея-Штрассена

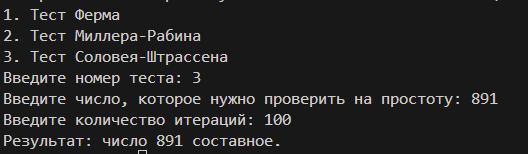


Рисунок 2. Тестирование проверки простоты с помощью теста Соловея-Штрассена

Тест Миллера-Рабина

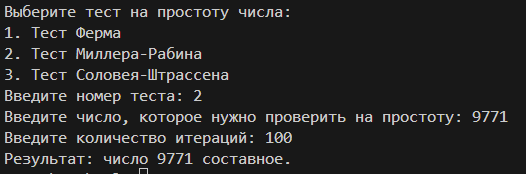


Рисунок 3. Тестирование проверки простоты с помощью теста Миллера-Рабина

# **ЗАКЛЮЧЕНИЕ**

В ходе работы били рассмотрены алгоритмы проверки чисел на простоту. В качестве тестов были рассмотрены тест Ферма, тест Миллера-Рабина, тест Соловея-Штрассена. Все эти тесты являются вероятностными тестами проверки на простоту, поэтому число, проверенное с помощью этих тестов, можно считать лишь вероятно простым.

Для проверки работоспособности этих алгоритмов была реализована программа на языке Python. Программы были протестированы и была подтверждена правильность их работы. Также была произведена оценка временной сложности алгоритмов.

# **ПРИЛОЖЕНИЕ А**

**Программа с реализованными алгоритмами**

import random

# Тест Ферма

def fermat\_primality\_test(n, k=5):

    if n <= 1:

        return False

    if n == 2:

        return True

    for \_ in range(k):

        a = random.randint(2, n - 2)

        if pow(a, n - 1, n) != 1:

            return False

    return True

# Тест Миллера-Рабина

def miller\_rabin(n, k):

    if n == 2 or n == 3:

        return True

    if n < 2 or n % 2 == 0:

        return False

    r, d = 0, n - 1

    while d % 2 == 0:

        r += 1

        d //= 2

    for \_ in range(k):

        a = random.randint(2, n - 2)

        x = pow(a, d, n)

        if x == 1 or x == n - 1:

            continue

        for \_ in range(r - 1):

            x = pow(x, 2, n)

            if x == n - 1:

                break

        else:

            return False

    return True

def jacobi\_symbol(a, n):

    if n <= 0 or n % 2 == 0:

        return 0

    result = 1

    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

# Тест Соловея-Штрассена

def solovay\_strassen\_test(n, k=5):

    if n <= 2:

        return n == 2

    if n % 2 == 0:

        return False

    for \_ in range(k):

        a = random.randint(2, n - 1)

        jacobi = jacobi\_symbol(a, n)

        mod\_exp = pow(a, (n - 1) // 2, n)

        if jacobi == 0 or mod\_exp != (jacobi % n):

            return False

    return True

def primality\_test\_interface():

    print("Выберите тест на простоту числа:")

    print("1. Тест Ферма")

    print("2. Тест Миллера-Рабина")

    print("3. Тест Соловея-Штрассена")

    choice = int(input("Введите номер теста: "))

    n = int(input("Введите число, которое нужно проверить на простоту: "))

    k = int(input("Введите количество итераций: "))

    if choice == 1:

        result = fermat\_primality\_test(n, k)

        if result:

            print(f"Результат: число {n}, вероятно, простое.")

        else:

            print(f"Результат: число {n} составное.")

    elif choice == 2:

        result = miller\_rabin(n, k)

        if result:

            print(f"Результат: число {n}, вероятно, простое.")

        else:

            print(f"Результат: число {n} составное.")

    elif choice == 3:

        result = solovay\_strassen\_test(n, k)

        if result:

            print(f"Результат: число {n}, вероятно, простое.")

        else:

            print(f"Результат: число {n} составное.")

    else:

        print("Неверный выбор.")

primality\_test\_interface()