Математические основы защиты информации и информационной безопасности. Отчет по лабораторной работе № 5

**Вероятностные алгоритмы проверки чисел на простоту**

Мохамед Либан Абдуллахи

Содержание

[Цель работы 2](#_Toc148189354)

[Задание 2](#_Toc148189355)

[Выполнение Работы 2](#_Toc148189356)

[Исходный код 2](#_Toc148189357)

[Результать Работы 6](#_Toc148189358)

[Выводы 6](#_Toc148189359)

[Список литературы 6](#_Toc148189360)

# Цель работы

# Освоить на практике применение алгоритмы проверки чисел на простоту.

# Задание

# 1 Алгоритм, реализующий тест ферма

# 2 алгоритм вычисления символа якоби

# 3 Алгоритм, реализующий тест Миллера-Рабина

# 4 алгоритм реализующий тест соловея-штрассена

# Выполнение Работы

# Для выполнения работы была написана программа с помощью языка программирования Python. Программа вычисляет алгоритма проверки чисел на простоту.

# Исходный код

import random

def is\_prime(n, k=5):

    """Проверка простоты числа методом Ферма-Рабина."""

    if n <= 1:

        return False

    if n <= 3:

        return True

    # Выполняем тест Ферма k раз

    for \_ in range(k):

        a = random.randint(2, n - 2)

        if pow(a, n - 1, n) != 1:

            return False

    return True

# Пример использования

number\_to\_test = 5

result = is\_prime(number\_to\_test)

if result:

    print(f"{number\_to\_test} - простое число")

else:

    print(f"{number\_to\_test} - составное число")

def jacobi\_symbol(a, n):

    if n <= 0 or n % 2 == 0:

        raise ValueError("Вторым аргументом должно быть положительное нечетное целое число")

    a = a % n

    result = 1

    while a != 0:

        while a % 2 == 0:

            a /= 2

            r = n % 8

            if r == 3 or r == 5:

                result = -result

        a, n = n, a

        if a % 4 == 3 and n % 4 == 3:

            result = -result

        a %= n

    if n == 1:

        return result

    else:

        return 0

# Пример использования

a = 3

n = 11

result = jacobi\_symbol(a, n)

print(f"Символ Якоби ({a}/{n}) = {result}")

def power\_mod(base, exponent, modulus):

    result = 1

    base = base % modulus

    while exponent > 0:

        if exponent % 2 == 1:

            result = (result \* base) % modulus

        exponent = exponent // 2

        base = (base \* base) % modulus

    return result

def jacobi\_symbol(a, n):

    if n <= 0 or n % 2 == 0:

        raise ValueError("Вторым аргументом должно быть положительное нечетное целое число")

    a = a % n

    result = 1

    while a != 0:

        while a % 2 == 0:

            a = a / 2

            r = n % 8

            if r == 3 or r == 5:

                result = -result

        a, n = n, a

        if a % 4 == 3 and n % 4 == 3:

            result = -result

        a = a % n

    if n == 1:

        return result

    else:

        return 0

def is\_prime\_spsp(n, k=5):

    if n == 2 or n == 3:

        return True

    if n == 1 or n % 2 == 0:

        return False

    # Выполняем тест Соловея-Штрассена k раз

    for \_ in range(k):

        a = random.randint(2, n - 2)

        x = jacobi\_symbol(a, n)

        y = power\_mod(a, (n - 1) // 2, n)

        if x == 0 or y != x % n:

            return False

    return True

# Пример использования

number\_to\_test = 11

result = is\_prime\_spsp(number\_to\_test)

if result:

    print(f"{number\_to\_test} - простое число")

else:

    print(f"{number\_to\_test} - составное число")

def power\_mod(base, exponent, modulus):

    result = 1

    base = base % modulus

    while exponent > 0:

        if exponent % 2 == 1:

            result = (result \* base) % modulus

        exponent = exponent // 2

        base = (base \* base) % modulus

    return result

def miller\_rabin\_test(n, k=5):

    if n == 2 or n == 3:

        return True

    if n == 1 or n % 2 == 0:

        return False

    # Представляем n - 1 как d \* 2^r, где d нечетное

    r, d = 0, n - 1

    while d % 2 == 0:

        r += 1

        d //= 2

    # Выполняем тест Миллера-Рабина k раз

    for \_ in range(k):

        a = random.randint(2, n - 2)

        x = power\_mod(a, d, n)

        if x == 1 or x == n - 1:

            continue

        for \_ in range(r - 1):

            x = (x \* x) % n

            if x == n - 1:

                break

        else:

            return False

    return True

# Пример использования

number\_to\_test = 17

result = miller\_rabin\_test(number\_to\_test)

if result:

    print(f"{number\_to\_test} - простое число")

else:

    print(f"{number\_to\_test} - составное число")

# Результать Работы

# 

# Выводы

# Освоено на практике применение алгоритма алгоритмы проверки чисел на простоту.

# Список литературы

1. Методические материалы курса