Отчет по лабораторной работе №5

Дисциплина: Математические основы защиты информации и информационной безопасности

Живцова Анна

Содержание

# 1 Цель работы

Изучить вероятностные алгоритмы проверки чисел на простоту.

# 2 Задание

Реализовать алгоритмы тестов Ферма, Соловэя-Штрассена, Миллера-Рабина.

# 3 Теоретическое введение

Простые числа широко применяются в криптографии с открытым ключем. Подробнее в источниках [1,2]. Алгоритмы проверки на простоту можно разделить на детерминированные и вероятностные. Вероятностные, возможно, вычислительно менее сложные, однако они позволяют дать ответ лишь с некоторой вероятностью. Иногда этого бывает достаточно. Рассмотрим несколько вероятностных тстов на простоту и приведем ключевые факты, на которых основаны тесты. Дале – простое число, и – произвольное число.

**Тест Ферма.** основан на следующем факте

**Тест Соловэя-Штрассена.** основан на критерии Эйлера  
>

**Тест Миллера-Рабина.** основан на тесте Ферма.

# 4 Выполнение лабораторной работы

В этом разделе – число, которое требуется проверить на простоту. нечетно и больше 5.

## 4.1 Тест Ферма

Для реализации теста Ферма на языке Python была написанна следующая функция.

def ferma(n):  
 a = np.random.randint(2, n-2)  
 if (a\*\*(n-1))%n == 1:  
 return 'Число {}, вероятно, простое'.format(n)  
 else:  
 return 'Число {} составное'.format(n)

## 4.2 Тест Соловэя-Штрассена

Для реализации теста отдельно была реализована функция рассчета символя Якоби

def yakoby(n, a, g = 1):  
 if a == 0:  
 return 0  
 if a == 1:  
 return g  
 k = 0  
 a1 = a  
 while a1%2 == 0:  
 k += 1  
 a1 /=2  
 if k%2 == 0:  
 s = 1  
 elif n%8 == 1 or n%8 == 7:  
 s = 1  
 elif n%8 == 3 or n%8 == 5:  
 s = -1  
 if a == 1:  
 return g\*s  
 if n%4 == 3 and a1%4 == 3:  
 s \*= -1  
 a = n%a1  
 n = a1  
 g \*= s  
 return yakoby(n, a, g)

Сам тест реализован с помощью функции

def s\_sh(n):  
 a = np.random.randint(2, n-2)  
 r = a\*\*((n-1)/2)  
 if r != 1 and r!= n-1:  
 return 'Число {} составное'.format(n)  
 else:  
 s = yakoby(n, a)  
 if r%n == s:  
 return 'Число {} составное'.format(n)  
 return 'Число {}, вероятно, простое'.format(n)

## 4.3 Тест Миллера-Рабина

Данный тест реализован с помощью функции

def m\_r(n):  
 s = 0  
 r = n-1  
 while r%2 == 0:  
 s += 1  
 r /=2  
 a = np.random.randint(2, n-2)  
 y = (a\*\*r)%n  
 if y != 1 and y != n - 1:  
 for j in range(1, s):  
 if y != n - 1 and (y\*y)%n == 1:  
 return 'Число {} составное'.format(n)  
 if y != n - 1:  
 return 'Число {} составное'.format(n)  
 return 'Число {}, вероятно, простое'.format(n)

## 4.4 Тестирование реализованных алгоритмов

Проведено тестирование реализованных алгоритмов. Видно, что в большинстве случаев ответы различных тестов совпадают, однако иногда делается ошибочное предположение о простоте (см. рис. 1).

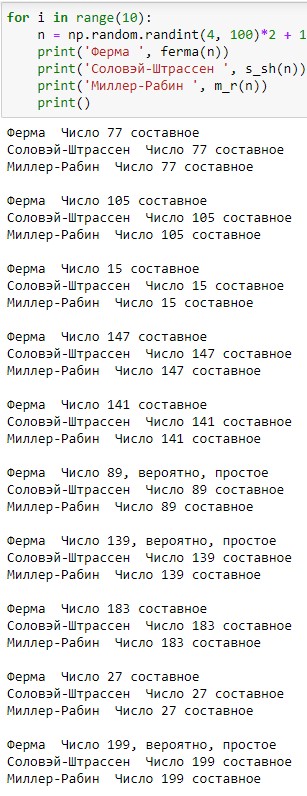


Рис. 1: Тестирование вероятностных алгоритмов проверки на простоту

# 5 Выводы

В данной работе я изучила вероятностные алгоритмы проверки чисел на простоту. Реализовала тесты Ферма, Соловэя-Штрассена, Миллера-Рабина. Протестировала реализованные функции.

Таким образом, задачи работы были выполнены, а цели достигнуты.

# Список литературы

1. Kulyabov D., Korolkova A., Gevorkyan M. Информационная безопасность компьютерных сетей: лабораторные работы. 2015.

2. Самуйлов К.Е. и др. Сети и телекоммуникации : Учебник и практикум. Издательство Юрайт, 2019. С. 1–363.