Лабораторная работа 5

Вероятностные алгоритмы проверки чисел на простоту

Пологов Владислав Александрович

Содержание

# Цель работы

Реализовать алгоритмы проверки числа на простоту:

1. Алгоритм, реализующий тест Ферма;  
2. Алгоритм вычисления символа Якоби;  
3. Алгоритм, реализующий тест Соловэя-Штрассена;  
4. Алгоритм, реализующий тест Миллера-Рабина.

# Описание реализации

Для реализации алгоритмов использовались средства языка Python.

# Реализация

## Алгоритм, реализующий тест Ферма

На вход мы подаём нечётное целое число n >= 5. На выходе получаем результат работы алгоритма и суждение о том, является ли число вероятно простым или составным. Алгоиртм, реализующий тест Ферма и его реализация на Python приведёны на рисунке 1. (рис. -fig. 1)

## Алгоритм, реализующий тест Ферма

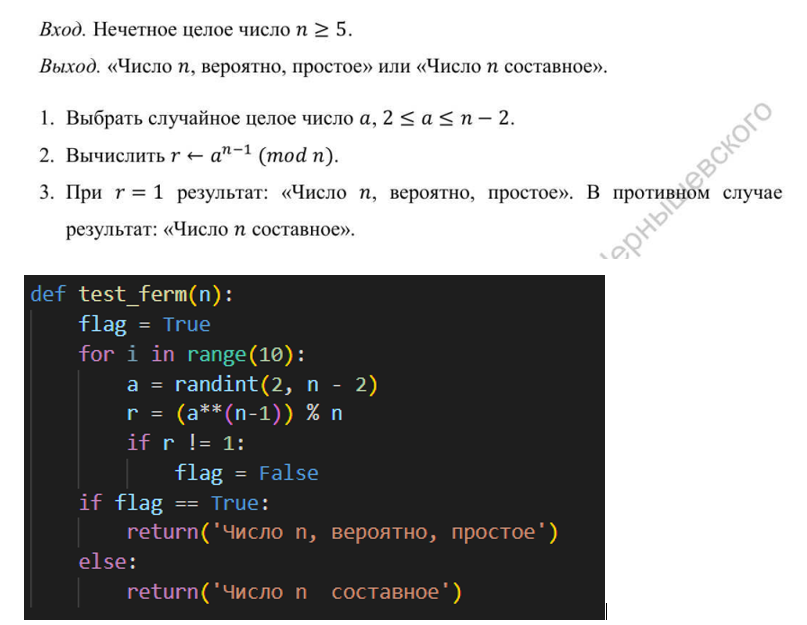


Figure 1: Алгоритм, реализующий тест Ферма

## Алгоритм вычисления символа Якоби

Для реализации алгоритма вычисления символа Якоби использовалась дополнительная переменная g.Символ Якоби практически никогда не вычисляют по определению. Чаще всего для вычисления используют свойства символа Якоби, главным образом — квадратичный закон взаимности. Ключевое используемое при вычислении свойство символа Якоби — квадратичный закон взаимности. Благодаря ему алгоритм похож на алгоритм Евклида нахождения наибольшего общего делителя двух чисел, в котором тоже аргументы на каждом шаге меняются местами. Аналогично алгоритму Евклида, при перестановке аргументов больший заменяется на остаток от деления на меньший. Это возможно благодаря периодичности символа Якоби. Однако, поскольку символ Якоби определён только при условии нечётности второго аргумента, то до перестановки выделяется чётная часть первого аргумента. Данный алгоритм и его реализация на Python представлены на рисунке 2. (рис. -fig. 2)

## Алгоритм вычисления символа Якоби

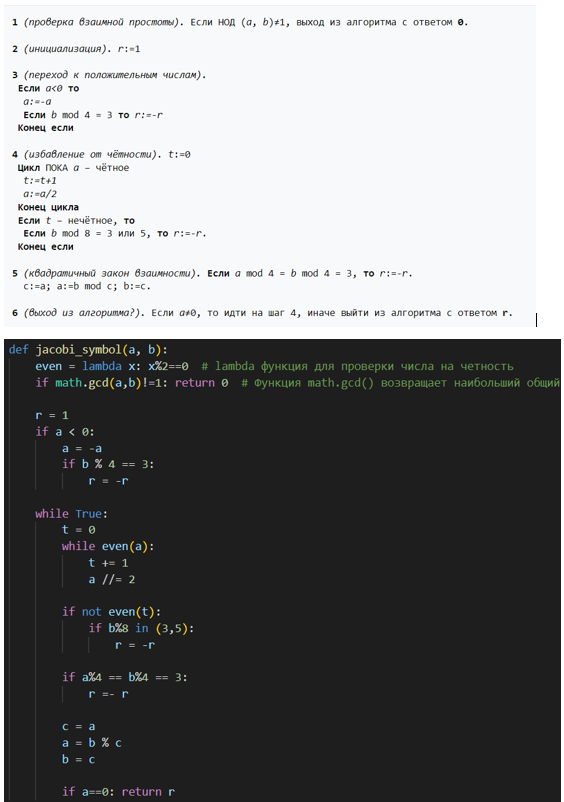


Figure 2: Алгоритм вычисления символа Якоби

## Алгоритм, реализующий тест Соловэя-Штрассена

Для его вычисления понадобится вызывать функцию нахождения символа Якоби. На вход мы подаём нечётное целое число n >= 5. На выходе получаем результат работы алгоритма и суждение о том, является ли число вероятно простым или составным. Алгоритм, реализующий тест Соловэя-Штрассена и его реализация на Python представлены на рисунке 3. (рис. -fig. 3)

## Алгоритм, реализующий тест Соловэя-Штрассена

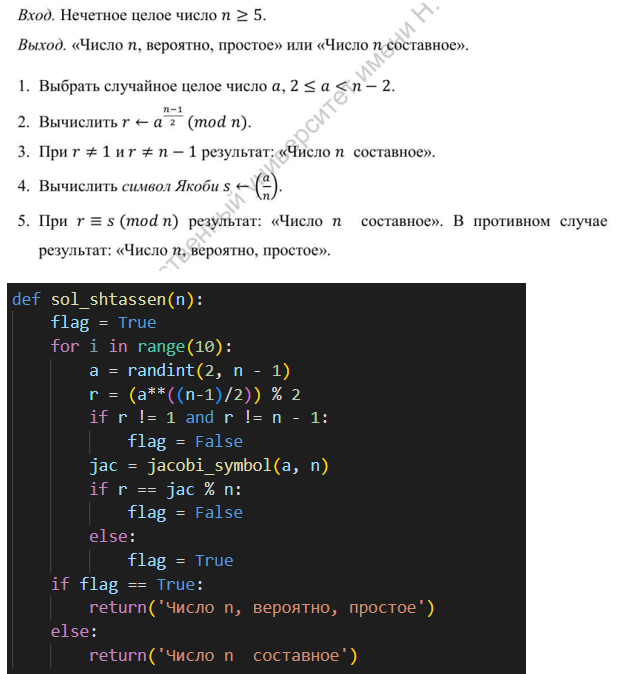


Figure 3: Алгоритм, реализующий тест Соловэя-Штрассена

## Алгоритм, реализующий тест Миллера-Рабина

На вход мы подаём нечётное целое число n >= 5. На выходе получаем результат работы алгоритма и суждение о том, является ли число вероятно простым или составным.

Алгоритм, реализующий тест Миллера-Рабина представлен на рисунке 4. (рис. -fig. 4)

Код алгоритма, реализующего тест Миллера-Рабина, представлен на рисунке 5. (рис. -fig. 5)

## Алгоритм, реализующий тест Миллера-Рабина

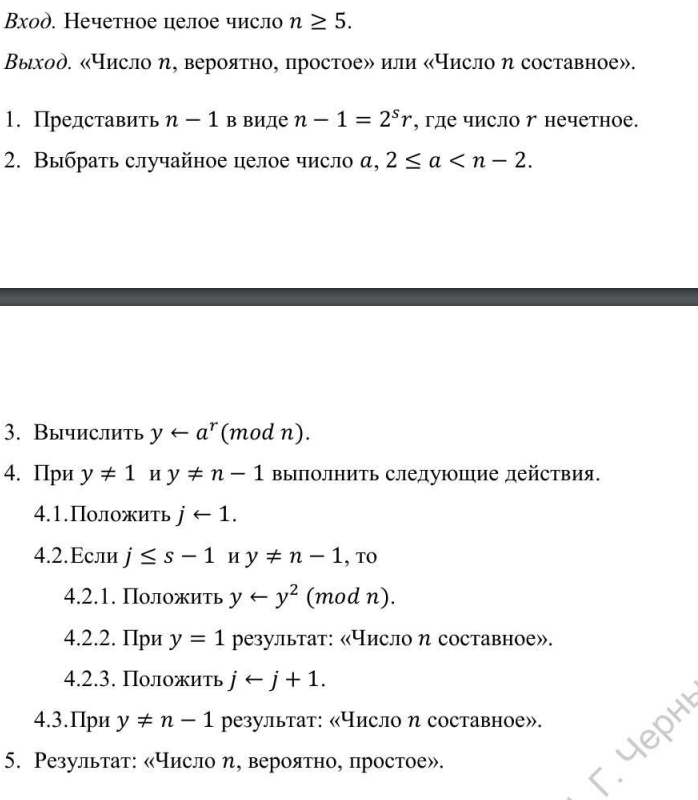


Figure 4: Алгоритм, реализующий тест Миллера-Рабина

## Алгоритм, реализующий тест Миллера-Рабина

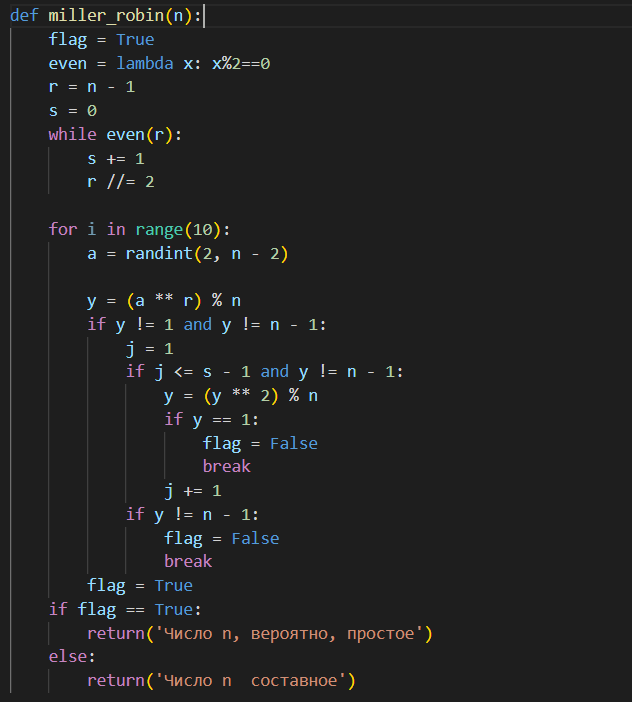


Figure 5: Алгоритм, реализующий тест Миллера-Рабина

# Вывод

* Реализовали следующие алгоритмы для проверки чисел на простоту:
  1. Алгоритм, реализующий тест Ферма;
  2. Алгоритм вычисления символа Якоби;
  3. Алгоритм, реализующий тест Соловэя-Штрассена;
  4. Алгоритм, реализующий тест Миллера-Рабина.