Математические основы защиты информации и информационной безопасности. Лабораторная работа №5

Вероятностные алгоритмы проверки чисел на простоту

Содержание

# 1 Цель работы

Изучение алгоритмов Ферма, Соловэя-Штрассена, Миллера-Рабина.

# 2 Теоретические сведения

Для построения многих систем защиты информации требуются простые числа большой разрядности. В связи с этим актуальной является задача тестирования на простоту натуральных чисел.

Существует два типа критериев простоты: детерминированные и вероятностные. Детерминированные тесты позволяют доказать, что тестируемое число - простое. Практически применимые детерминированные тесты способны дать положительный ответ не для каждого простого числа, поскольку используют лишь достаточные условия простоты. Детерминированные тесты более полезны, когда необходимо построить большое простое число, а не проверить простоту, скажем, некоторого единственного числа. В отличие от детерминированных, вероятностные тесты можно эффективно использовать для тестирования отдельных чисел, однако их результаты, с некоторой вероятностью, могут быть неверными. К счастью, ценой количества повторений теста с модифицированными исходными данными вероятность ошибки можно сделать как угодно малой. На сегодня известно достаточно много алгоритмов проверки чисел на простоту. Несмотря на то, что большинство из таких алгоритмов имеет субэкспоненциальную оценку сложности, на практике они показывают вполне приемлемую скорость работы. На практике рассмотренные алгоритмы чаще всего по отдельности не применяются. Для проверки числа на простоту используют либо их комбинации, либо детерминированные тесты на простоту. Детерминированный алгоритм всегда действует по одной и той же схеме и гарантированно решает поставленную задачу. Вероятностный алгоритм использует генератор случайных чисел и дает не гарантированно точный ответ. Вероятностные алгоритмы в общем случае не менее эффективны, чем детерминированные (если используемый генератор случайных чисел всегда дает набор одних и тех же чисел, возможно, зависящих от входных данных, то вероятностный алгоритм становится детерминированным).

## 2.1 Тест Ферма

* Вход. Нечетное целое число .
* Выход. «Число n, вероятно, простое» или «Число n составное».

1. Выбрать случайное целое число .
2. Вычислить
3. При результат: «Число n, вероятно, простое». В противном случае результат: «Число n составное».

Подробнее об алгоритме: [1]

## 2.2 Тест Соловэя-Штрассена

* Вход. Нечетное целое число .
* Выход. «Число n, вероятно, простое» или «Число n составное».

1. Выбрать случайное целое число .
2. Вычислить
3. При и результат: «Число n составное».
4. Вычислить символ Якоби
5. При результат: «Число n, вероятно, простое». В противном случае результат: «Число n составное».

Подробнее об алгоритме: [2]

## 2.3 Тест Миллера-Рабина

* Вход. Нечетное целое число .
* Выход. «Число n, вероятно, простое» или «Число n составное».

1. Представить в виде , где r - нечетное число
2. Выбрать случайное целое число .
3. Вычислить
4. При и выполнить действия
   * Положить
   * Если и то
     + Положить
     + При результат: «Число n составное».
     + Положить
   * При результат: «Число n составное».
5. Результат: «Число n, вероятно, простое».

Подробнее об алгоритме: [3]

# 3 Выполнение работы

## 3.1 Реализация алгоритмов

import random  
  
  
def fermat(n):  
 a = random.randint(2, n - 2)  
 r = a \*\* (n - 1) % n  
 return r == 1  
  
  
def jacoby(a, n):  
 if (a == 0):  
 return 0  
 result = 1  
 if (a < 0):  
 a = -a  
 if (n % 4 == 3):  
 result = -result  
 if (a == 1):  
 return result  
 while (a):  
 if (a < 0):  
 a = -a  
 if (n % 4 == 3):  
 result = -result  
 while (a % 2 == 0):  
 a = a // 2  
 if (n % 8 == 3 or n % 8 == 5):  
 result = -result  
 a, n = n, a  
 if (a % 4 == 3 and n % 4 == 3):  
 result = -result  
 a = a % n  
 if (a > (n // 2)):  
 a = a - n  
 if (n == 1):  
 return result  
 return 0  
  
  
def solovay\_strassen(n):  
 a = random.randint(2, n - 2)  
 r = a \*\* ((n - 1) / 2)  
 if (r != 1 and r != (n - 1)):  
 return False  
 s = jacoby(a, n)  
 return not ((r - s) % n == 0)  
  
  
def miller\_rabin(n):  
 a = random.randint(2, n - 2)  
 d = n - 1  
 s = 0  
 while (d % 2 == 0):  
 s = s + 1  
 d = int(d / 2)  
 x = a \*\* d  
 x = x % n  
 if (x == 1 or x == (n - 1)):  
 return True  
 r = 1  
 while (r < (s - 1)):  
 x = x \*\* 2  
 x = x % n  
 if (x == 1):  
 return False  
 if (x == (n - 1)):  
 return True  
 return False  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 while True:  
 try:  
 result\_code = int(input(  
 """  
Выберите алгоритм проверки числа на простоту  
 1 - Алгоритм Ферма;  
 2 - Алгоритм Соловэя-Штрассена;  
 3 - Алгоритм Миллера-Рабина;  
 -------------------------  
 0 - Выход из программы  
Введите номер операции: """  
 ))  
 if result\_code > 3:  
 print("Ошибка ввода!")  
 continue  
 if result\_code == 0:  
 break  
 except:  
 print("Ошибка ввода!")  
 continue  
  
 try:  
 n = int(input("Введите нечетное целое число > 5: "))  
 except:  
 print("Ошибка ввода!")  
 continue  
  
 result = False  
  
 if result\_code == 1:  
 result = fermat(n)  
  
 if result\_code == 2:  
 result = solovay\_strassen(n)  
  
 if result\_code == 3:  
 result = miller\_rabin(n)  
  
 if result:  
 print("Число {}, вероятно, простое".format(n))  
 else:  
 print("Число {} составное".format(n))

## 3.2 Пример работы алгоритма Ферма

На рис. 1 представлены результаты работы алгоритма Ферма:

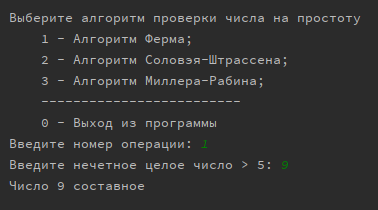


Figure 1: Пример работы алгоритма Ферма

## 3.3 Пример работы алгоритма Соловэя-Штрассена

На рис. 2 представлены результаты работы алгоритма Соловэя-Штрассена:

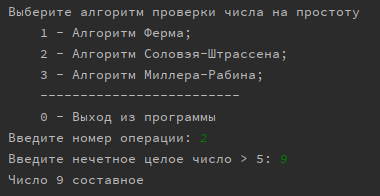


Figure 2: Пример работы алгоритмов для Соловэя-Штрассена

## 3.4 Пример работы алгоритма Миллера-Рабина

На рис. 3 представлены результаты работы алгоритма Миллера-Рабина:

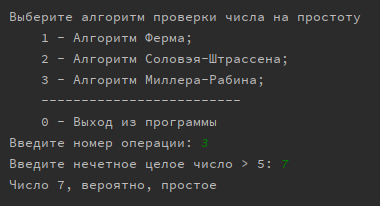


Figure 3: Пример работы алгоритмов для Миллера-Рабина

# 4 Выводы

В ходе выполнения работы были успешно изучены алгоритмы Ферма, Соловэя-Штрассена, Миллера-Рабина, а также реализованы данные алгоритмы программно на языке Python.

# Список литературы

1. Алгоритм Ферма [Электронный ресурс]. Википедия, 2021. URL: <https://ru.wikipedia.org/wiki/Метод_факторизации_Ферма>.

2. Алгоритм Соловэя-Штрассена [Электронный ресурс]. Википедия, 2020. URL: <https://ru.wikipedia.org/wiki/Тест_Соловея_—_Штрассена>.

3. Расширенный Миллера-Рабина [Электронный ресурс]. Википедия, 2021. URL: <https://ru.wikipedia.org/wiki/Тест_Миллера_—_Рабина>.