Отчет по лабораторной работе 5

По предмету мат. основы защиты информации

Студент: Дидусь Кирилл Валерьевич, 1132223499

Группа: НПМмд-02-22

Преподаватель: Кулябов Дмитрий Сергеевич,

д-р.ф.-м.н., проф.

Москва, 2022

Содержание

# 1 Цель работы

Целью данной лабораторной работы является ознакомление с алгоритмом для вероятностной проверки числа на простоту.

# 2 Задание

Реализовать алгоритмом для вероятностной проверки числа на простоту в программном виде.

# 3 Выполнение лабораторной работы

В ходе выполнения лабораторной работы было реализовано 3 вариации алгоритма для вероятностной проверки числа на простоту, использующих различные тесты.

Среди них:

* тест Ферма
* тест Соловэя-Штрассена
* тест Миллера-Рабина

Программный код представлен в качестве листинга в конце отчета.

# 4 Выводы

Таким образом, была достигнута цель, поставленная в начале лабораторной работы: я ознакомился с алгоритмом для вероятностной проверки числа на простоту, а так же мне удалось реализовать вариции этого алгоритма на языке программирования Python.

# 5 Листинг программы

# Проверка чисел на простоту является составной частью  
# алгоритмов генерации простых чисел, применяемых в криптографии с открытым ключом.  
# АПП бывают вероятностные и детерминированные. Рассмотрим вероятностные алгоритмы.  
  
import random as rnd  
  
def dividers\_count(n,divider):  
 i = 0  
 n\_copy = n  
 while (n\_copy % divider == 0):  
 i += 1  
 n\_copy /= divider  
 return i  
  
def jacobi(n,a):  
 if n<3:  
 print("n can't be lower than 5")  
 return  
 g = 1  
 res, s = 0,0  
 n\_copy = n  
 a\_copy = a  
 while(True):  
 if(a\_copy == 0):  
 res = 0  
 return res  
 elif(a\_copy == 1):  
 res = g  
 return res  
 k = dividers\_count(a\_copy,2)  
 a1 = a\_copy / pow(2,k)  
 if(k%2 ==0):  
 s = 1  
 else:  
 if ((n\_copy % 8 == 1) | (n\_copy % 8 == -1)):  
 s = 1  
 elif ((n\_copy % 8 == 3) | (n\_copy % 8 == -3)):  
 s = -1  
 if(a1==1):  
 res = g\*s  
 return res  
 if((n\_copy % 4 == 3) & (a1%4==3)):  
 s = -s  
 a\_copy = n\_copy % a1  
 n\_copy = a1  
 g = g\*s  
  
def ferma\_test(n): #n>=5  
 if n<5:  
 print("n can't be lower than 5")  
 return  
 a = rnd.randrange(2,n-2)  
 r = pow(a,(n-1))%n  
 if(r==1):  
 return True  
 else:  
 return False  
  
def solovayStrassen\_test(n):  
 if n<5:  
 print("n can't be lower than 5")  
 return  
 a = rnd.randrange(2,n-2)  
 r = pow(a,((n-1)/2)%n)  
 s = jacobi(n,a)  
 if (r % n == s):  
 return False  
 else:  
 return True  
   
def millerRabin\_test(n):  
 if n<5:  
 print("n can't be lower than 5")  
 return  
 s = dividers\_count(n-1,2)  
 r = (n-1) / pow(2,s)  
 a = rnd.randrange(2,n-2)  
 y = pow(a,r)%n  
 if((y != 1) & (y != n-1)):  
 j = 1  
 if ((j < s - 1) & (y != n - 1)):  
 y = y\*y % n  
 if (y == 1):  
 return False  
 j += 1  
 if (y != n-1):  
 return False  
 return True  
  
  
def is\_prime(n):  
 for i in range(20):  
 # если тест возрващают True 20 раз, n очень вероятно простое число.  
 if((ferma\_test(n)) & True):  
 continue  
 else:  
 return False  
 return True  
  
# if true - вероятно простое  
# if false - составное  
n = int(input("введите n: "))  
print(is\_prime(n))