Вероятностные алгоритмы проверки чисел на простоту

Тагиев Байрам Алтай оглы

Содержание

## 0.1 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».

## 0.2 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».

## 0.3 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».

# 1 Выполнение работы

## 1.1 Реализация алгоритмов на языке Python

import random  
  
def Ferma(n, count):  
 for i in range(count):  
 a = random.randint(2, n-1)  
 if ( a\*\*(n-1)%n != 1 ):  
 print("Complex")  
 return False  
 print("Simple")  
 return True

def modulo(base, exponent, mod):  
 x = 1  
 y = base  
 while (exponent > 0):  
 if (exponent%2 == 1):  
 x = (x\*y)%mod  
 y = (y\*y)%mod  
 exponent = exponent//2  
 return x%mod  
  
def calculateJacobian(a, n):  
 if (a == 0):  
 return 0  
 ans = 1  
 if (a < 0):  
 a = -a  
 if (n%4 == 3):  
 ans = -ans  
 if ( a == 1):  
 return ans  
 while (a):  
 if (a < 0):  
 a = -a  
 if (n%4 == 3):  
 ans = -ans  
 while (a%2 == 0):  
 a = a//2  
 if (n%8 == 3 or n%8 == 5):  
 ans = -ans  
 a, n = n, a  
 if (a%4 == 3 and n%4 == 3):  
 ans = -ans  
 a = a%n  
 if (a > n//2):  
 a = a - n  
 if (n == 1):  
 return ans  
 return 0  
  
def SoloveiStrassen(p, iterations):  
 if (p < 2):  
 print("Complex")  
 return False  
 if (p!=2 and p%2==0):  
 print("Complex")  
 return False  
 for i in range(iterations):  
 a = random.randrange(p-1) + 1  
 jacobian = (p + calculateJacobian(a, p))%p  
 mod = modulo(a, (p-1)/2, p)  
 if (jacobian == 0 or mod != jacobian):  
 print("Complex")  
 return False  
 return True

def MillerRabin(n):  
 if n != int(n):  
 print("Complex")  
 return False  
 n = int(n)  
 if n==0 or n==1 or n==4 or n==6 or n==8 or n==9:  
 print("Complex")  
 return False  
 if n==2 or n==3 or n==5 or n==7:  
 print("Simple")  
 return True  
 s = 0  
 d = n-1  
 while d%2 == 0:  
 d >>= 1  
 s += 1  
 assert(2\*\*s\*d == n-1)  
   
 def trial\_compose(a):  
 if pow(a, d, n) == 1:  
 print("Complex")  
 return False  
 for i in range(s):  
 if pow(a, 2\*\*i\*d, n) == n-1:  
 print("Complex")  
 return False  
 print("Simple")  
 return True  
   
 for i in range(8):  
 a = random.randrange(2, n)  
 if trial\_compose(a):  
 print("Complex")  
 return False  
 print("Simple")  
 return True

## 1.2 Результаты выполнения

n = 101  
print(Ferma(n, 25))  
print("==========")  
print(Ferma(n+1, 25))

Simple  
True  
==========  
Complex  
False

print(SoloveiStrassen(n, 25))  
print("==========")  
print(SoloveiStrassen(n+1, 25))

True  
==========  
Complex  
False

print(MillerRabin(n))  
print("==========")  
print(MillerRabin(n+1))

Complex  
Complex  
Complex  
Complex  
Complex  
Complex  
Complex  
Complex  
Simple  
True  
==========  
Simple  
Complex  
False

# 2 Выводы

Изучили алгоритмы Ферма, Соловэя-Штрассена, Миллера-Рабина.