Отчёт по лабораторной работе №5 Вероятностные алгоритмы проверки чисел на простоту

Дисциплина: Математические основы защиты информации и информационной безопасности

Студент: Леонова Алина Дмитриевна, 1032212306
Группа: НФИмд-01-21
Преподаватель: д-р.ф.-м.н., проф. Кулябов Дмитрий Сергеевич

11 декабря, 2021, Москва

Цель и задание работы

Цель работы

Целью данной работы является ознакомление и реализация на выбранном языке программирования трёх вероятностных алгоритмов проверки чисел на простоту, а также алгоритма вычисления символа Якоби.

Задание

Реализовать программно:

- алгоритм, реализующий тест Ферма
- алгоритм вычисления символа Якоби
- алгоритм, реализующий тест Соловэя-Штрассена
- алгоритм, реализующий тест Миллера-Рабина

Теоретическое введение

Тесты проверки простоты

Тестом простоты (или проверкой простоты) называется алгоритм, который, приняв на входе число N, позволяет либо не подтвердить предположение о составности числа, либо точно утверждать его простоту. Во втором случае он называется истинным тестом простоты.

Тест Ферма

Тест Ферма опирается на малую теорему Ферма:

Если n — простое число, то оно удовлетворяет сравнению $a^{n-1} \equiv 1(\; mod \; n)$ для любого a, которое не делится на n.

Тест Соловэя-Штрассена

Тест Соловея-Штрассена опирается на малую теорему Ферма и свойства символа Якоби $(\frac{a}{n})$:

- Если n — нечетное составное число, то количество целых чисел a, взаимнопростых с n и меньших n, удовлетворяющих сравнению $a^{(n-1)/2} \equiv (\frac{a}{n})(\ mod\ n)$, не превосходит $n/2$.

Выполнение лабораторной работы

Алгоритм, реализующий тест Ферма

```
2
3   # 1. Тест Ферма
4   def test_Ferma(n):
5       print('')
6       if n % 2 == 0:
7           return print('Ошибка: число ', n,' чётное')
8       if n < 5:
9           return print('Ошибка: число ', n,' < 5')
10
11      a = random.randint(2, n - 2)
12      r = ( a ** (n - 1) ) % n
13
14      if r == 1:
15          return print('Число ', n,' , вероятно, простое')
16      else:
17          return print('Число ', n,' составное')
```

Figure 1: Функция test_Ferma

Функция, реализующая тест Ферма (см. рис. 1).

Алгоритм вычисления символа Якоби

```python
# 2. Символ Якоби
def Yakobi(n, a):
    print('# символ Якоби (', a,'/',n,')')
    if n % 2 == 0:
        return print('Ошибка: число ', n,' чётное')
    if n < 3:
        return print('Ошибка: число n (', n,') < 3')
    if a < 0 or a >= n:
        return print('Ошибка: число a (', a,') некорректно')

    g = 1

    while True:
        if a == 0:
            return 0
        if a == 1:
            return g

        k = 0
        while a % (2**k) == 0:
            k += 1
        k -= 1

        a1 = a / (2**k)
        #print(a, ' = 2 ^', k, ' * ', a1)

        s = 0
        if k % 2 == 0:
            s = 1
        else:
            if (n - 1) % 8 == 0 or (n + 1) % 8 == 0:
                s = 1
            if (n - 3) % 8 == 0 or (n + 3) % 8 == 0:
                s = -1

        if a1 == 1:
            return g*s

        if (n - 3) % 4 == 0 and (a1 - 3) % 4 == 0:
            s = -s

        a = n % a1
        n = a1
        g = g*s
```

Figure 2: Функция Yakobi

Алгоритм, реализующий тест Соловэя-Штрассена

```
65
66  # 3. Тест Соловэя-Штрассена
67  def test_Sol_Shtr(n):
68      print('')
69      if n % 2 == 0:
70          return print('Ошибка: число ', n,' чётное')
71      if n < 5:
72          return print('Ошибка: число ', n,' < 5')
73
74      a = random.randint(2, n - 3)
75      r = ( a ** ((n - 1)/2) ) % n
76
77      if r != 1 and r != n-1:
78          return print('Число ', n,' составное')
79
80      s = Yakobi(n, a)
81
82      if (r - s) % n != 0:
83          return print('Число ', n,' составное')
84      else:
85          return print('Число ', n,' , вероятно, простое')
86
```

Figure 3: Функция test_Sol_Shtr

Функция, реализующая тест Соловэя-Штрассена (см. рис. 3).

Алгоритм, реализующий тест Миллера-Рабина

```
88  # 4. Тест Миллера-Рабина
89  def test_Mil_Rab(n):
90      print('')
91      if n % 2 == 0:
92          return print('Ошибка: число ', n,' чётное')
93      if n < 5:
94          return print('Ошибка: число ', n,' < 5')
95
96      s = 0
97      while n - 1 % (2**s) == 0:
98          s += 1
99      s -= 1
100     r = (n - 1) / (2**s)
101     #print(n - 1, ' = 2 ^', s, ' * ', r)
102
103     a = random.randint(2, n - 3)
104     y = ( a ** r ) % n
105
106     if y != 1 and y != n-1:
107         j = 1
108         while j <= s - 1 and y != n - 1:
109             y = (y*y) % n
110             if y == 1:
111                 return print('Число ', n,' составное')
112             j += 1
113         if y != n - 1:
114             return print('Число ', n,' составное')
115
116     return print('Число ', n,' , вероятно, простое')
```

Figure 4: Функция test_Mil_Rab

Функция, реализующая тест Миллера-Рабина (см. рис. 4).

Проверка

```
# Функция проверки функций тестов
def check(f):
    f(1)
    f(1122)
    f(11)
    f(27)
    f(31)


print('Тест Ферма')
check(test_Ferma)

print('\n-----------------------------------')
print('Символ Якоби')
print('Результат:', Yakobi(27, 5))
print('Результат:', Yakobi(27, 12))
print('Результат:', Yakobi(51, 13))

print('\n-----------------------------------')
print('Тест Соловэя-Штрассена')
check(test_Sol_Shtr)

print('\n-----------------------------------')
print('Тест Миллера-Рабина')
check(test_Mil_Rab)
```

Figure 5: Функция check и проверка всего

Функция проверок работы всех реализованных функций на пяти разных вариантах входных параметров и вызов проверки всего (см. рис. 5).

Результат работы реализованных алгоритмов

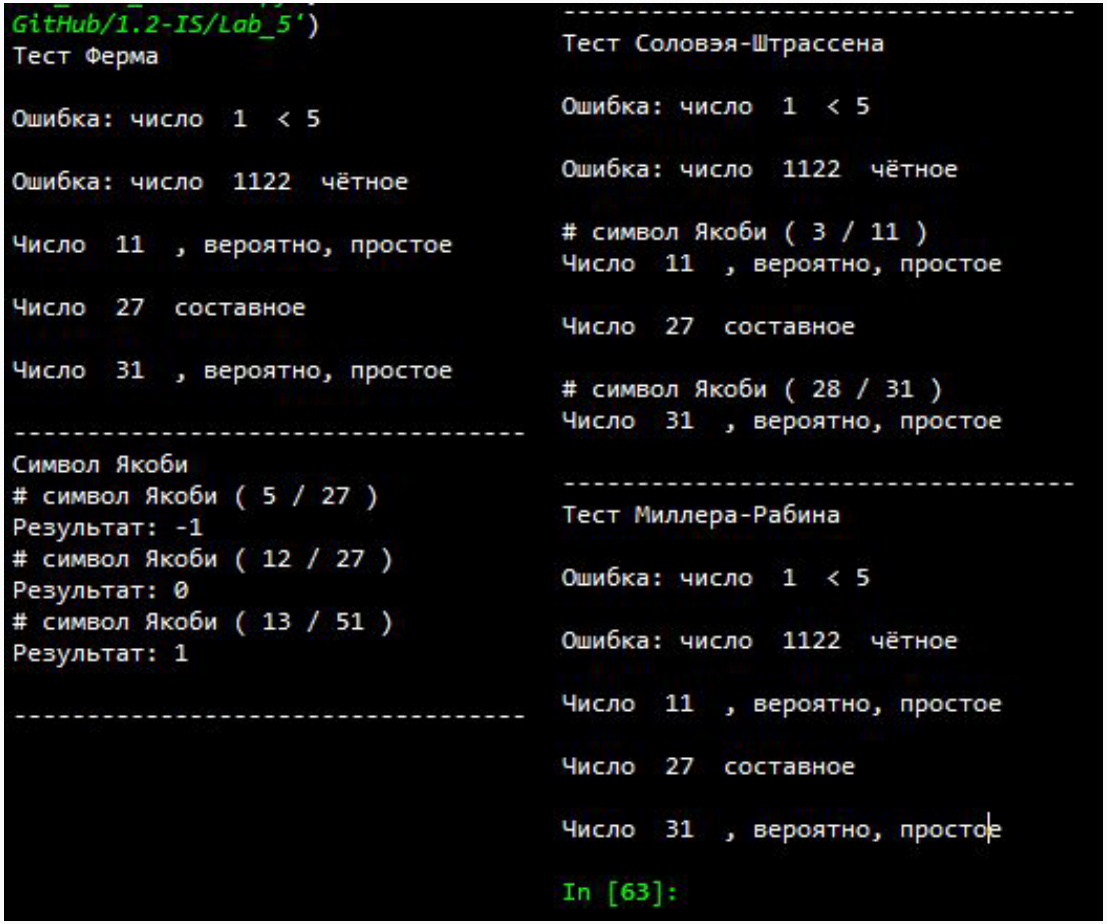

Figure 6: Результат выполнения L5_Leonova.py

Результат выполнения программы (см. рис. 6).

Выводы

Цель лабораторной работы была достигнута, три вероятностных алгоритмов проверки чисел на простоту и алгоритм вычисления символа Якоби были реализованы на языке программирования Python.