Отчёт

по лабораторной работе 5

Кочетов Андрей Владимирович

Содержание

List of Figures

List of Tables

1 Цель работы

Реализовать различные тесты.

2 Задание

Лабораторная работа подразумевает написание программ на языке python, которая реализует тесты, приведенные в методичке.

3 Выполнение лабораторной работы

1. Реализация теста Ферма

```
import random
```

```
def ferma(n, count):
    for i in range(count):
        a = random.randint(2, n-1)
        if (a**(n-1)) % n!=1):
            print('Число n составное')
            return False
    print('Число n, вероятно, простое')
    return True
```

Figure 3.1: рис.1. Тест Ферма

2. Реализация вычисления символа Якоби.

```
    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
```

Figure 3.2: рис.2. Символ Якоби

3. Реализация теста Соловзя-Штрассена.

```
]: def modul(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 solovay_strassen(p, iter):
       if (p < 2):
           return False
       if (p != 2 and p % 2 == 0):
           return False
       for i in range(iter):
           a = random.randrange(p-1) + 1
           jacobian = (p + find_jacobian(a, p)) % p
           mod = modul(a, (p-1) / 2, p)
           if (jacobian == 0 or mod != jacobian):
               return False
       return
```

Figure 3.3: рис.3. Тест Соловзя-Штрассена

4. Реализация теста Миллера-Рабина.

```
In [ ]: def millet_rabin(n):
            if n!= int(n):
                print('Число составное')
                return False
            n = int(n)
            if n == 0 or n == 1 or n == 4 or n==6 or n==8 or n ==9:
                print('Число n составное')
            if n == 2 or n == 3 or n == 5 or n == 7:
                print('Числа n, вероятно, простое')
                return True
            s = 0
            d = n-1
            while d%2 == 0:
                d >>= 1
                s+=1
            assert(2** s * d == n-1)
            def probn_sost(a):
                if pow(a,d,n) == 1:
                    print('Число n составное')
                    return False
                for i in range(s):
                    if pow(a, 2**i*d, n) == n - 1:
                        print('Число n составное')
                        return False
                print('Число n, вероятно, простое')
                return True
            for i in range(8):
                a = random.randrange(2, n)
                if probn_sost(a):
                    print('Число n составное')
                    return False
            print('Число n, вероятно, простое')
            return
```

Figure 3.4: рис.4. Теста Миллера-Рабина

5. Запуск алгоритмов.

```
: def main():
      n = int(input('Введите число для теста Ферма: '))
      print('Тест Ферма для числа: ', n)
      ferma(n, 500)
      print('Тест Миллера-Рабина')
      n = int(input('Введите число для теста Миллера-Рабина: '))
      miller_rabin(n)
      n = int(input('Введите число для теста Соловзя-Штрассена: '))
      if (solovay_strassen(n, 500)):
          print(n, 'Число n, вероятно, простое')
      else:
          print(n, 'Число n составное')
```

```
: main()
```

```
Введите число для теста Ферма: 7
Тест Ферма для числа:  7
Число n, вероятно, простое
Тест Миллера-Рабина
Введите число для теста Миллера-Рабина: 7
Числа n, вероятно, простое
Введите число для теста Соловзя-Штрассена: 7
7 Число n составное
```

```
: main()
```

Figure 3.5: рис.5. Запуск

4 Выводы

Я написал программный код, который реализует различные тесты.