Отчет по лабораторной работе №5

Вероятностные алгоритмы проверки чисел на простоту

Асеинова Елизавета Валерьевна

8 ноября 2023

Содержание

Список иллюстраций

1 Цель работы

Цель данной работы: научиться реализовывать алгоритмы проверки чисел на простоту.

2 Задание

1. Реализовать алгоритмы проверки чисел на простоту.

3 Теоретическое введение

Пусть a - целое число. Числа ‡1, ‡a называются тривиальными делителями числа a. Целое число p €Z/{0) называется простым, если оно не является делителем единицы и не имеет других делителей, кроме тривиальных. В противном случае число p €Z/{-1, 0, 1) называется составным.

4 Выполнение лабораторной работы

1. Прописываем функцию для алгоритма теста Ферма (рис. 4.1).

```
def ferma(a, n):
  r = (a ** (n-1)) % n
  if r == 1:
    print('Число n =', n, 'вероятно, простое')
  else:
    print('Число n =', n, 'составное')
```

```
14] ferma(12, 41)
```

```
Число n = 41 вероятно, простое
```

Рис. 4.1: Программная реализация алгоритма теста Ферма.

2. Прописывается функция для алгоритма вычисления символа Якоби. (рис. 4.2).

```
def Jakobi_symbol(a,n):
  g = 1
  while True:
    if a==0:
      res = 0
      break
    else:
      k = primefactors(a)[0]
      a1 = primefactors(a)[1]
      if k % 2 == 0:
        s = 1
      if k % 2 != 0:
        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:
      res = g * s
      break

    if ((n-3) % 4) == 0 or ((a1-3) % 4) == 0:
      s = -s
    a = n % a1
    n = a1
    g = g * s
  return -res
```

```
[17] Jakobi_symbol(10, 21)
```

```
1
```

Рис. 4.2: Алгоритм вычисления символа Якоби

3. Программная реализация алгоритма Соловэй-Штрассена. (рис. 4.3).

```
def solovey_strassen(a, n):
  r = (a ** ((n-1) / 2)) % n
  if r != 1 and r != (n-1):
    print('Число n =', n, 'составное')
  s = Jakobi_symbol(a,n)
  if (r - s) % n != 0:
    print('Число n =', n, 'составное')
  else:
    print('Число n =', n, 'вероятно, простое')
```

```
[24] solovey_strassen(12,41)
```

```
Число n = 41 вероятно, простое
```

Рис. 4.3: Программная реализация алгоритма Соловэй-Штрассена

4. Программная реализация алгоритма Миллера-Рабина. (рис. 4.4).

```
def miller_rabin(a, n):
  s = primefactors(n-1)[0]
  r = primefactors(n-1)[1]
  y = (a ** r) % n

  if y != 1 and y != n-1:
    j = 1
    while j <= s-1 and y != n-1:
      y = (y **2) % n
      if y == 1:
        return 'Число n =', n, 'составное'
      j += 1
    if y != n - 1:
      return 'Число n =', n, 'составное'
  return 'Число n =', n, 'вероятно, простое'
```

```
[28] miller_rabin(12, 40)
```

```
('Число n =', 40, 'составное')
```

Рис. 4.4: Программная реализация алгоритма Миллера-Рабина.

5 Выводы

В ходе работы были реализованы алгоритмы проверки чисел на простоту.