

Вероятностные алгоритмы проверки чисел на простоту

Гебриал Ибрам Есам Зекри ¹

2022 Moscow, Russia

¹RUDN University, Moscow, Russian Federation

Цель работы

Реализация алгоритмов Ферма, Соловья-Штрассена, Миллера-Рабина и вычисления Якоби.

Задачи

1. Реализовать алгоритм Ферма.
2. Реализовать алгоритм Соловья-Штрассена.
3. Реализовать алгоритм Миллера-Рабина.
4. Реализовать алгоритм вычисления Якоби.

Реализация

Функция `ferma` для алгоритма ферма. (рис. 1)

```
3 def ferma(n):  
4     print("Тесты Ферма")  
5     a = random.randint(2, n - 2)  
6     r = a ** (n - 1) % n  
7     if r==1:  
8         print("Число n, вероятно, простое")  
9     else:  
10        print("Число n составное")  
11  
12 n= int(input("enter n(Odd number): "))  
13 ferma(n)  
14
```

Figure 1: Функция для алгоритма ферма

Функция modul для вычисления бинарного эксп. (рис. 2)

```
15 # функция для бинарного эксп
16 def modul(base, exponent, mod):
17     x = 1
18     y = base
19     while (exponent > 0):
20         if (exponent % 2 == 1):
21             x = (x * y) % mod
22
23             y = (y * y) % mod
24             exponent = exponent // 2
25
26     return x % mod
27
28
```

Figure 2: Функция для вычисления бинарного эксп

Реализация алгоритма вычисления Якоби.

Функция jacobian для вычисления Якоби. (рис. 3)

```
29 def jacobian(a, n):
30     if (a == 0):
31         return 0
32     ans = 1
33     if (a < 0):
34         a = -a
35     if (n % 4 == 3):
36         ans = -ans
37     if (a == 1):
38         return ans
39     while (a):
40         if (a < 0):
41             a = -a
42         if (n % 4 == 3):
43             ans = -ans
44         while (a % 2 == 0):
45             a = a // 2
46         if (n % 8 == 3 or n % 8 == 5):
47             ans = -ans
48         a, n = n, a
49         if (a % 4 == 3 and n % 4 == 3):
50             ans = -ans
51         a = a % n
52         if (a > n // 2):
53             a = a - n
54     if (n == 1):
55         return ans
56     return 0
57
```

Figure 3: Функция для вычисления Якоби

Функция solovoy для алгоритма Соловья-Штрассена. (рис. 4)

```
59 def solovoy(n):  
60     print("Тест Соловья-Штрассена")  
61     a = random.randrange(2,n-2)  
62     r= (a**(n-1/2))%n  
63     if (r != 1 and r!=n-1):  
64         print("Число n составное")  
65     s=jacobian(a,n)  
66     if modul(r,s,n) == 1:  
67         print("Число n составное")  
68     else:  
69         print("Число n, вероятно, простое")  
70
```

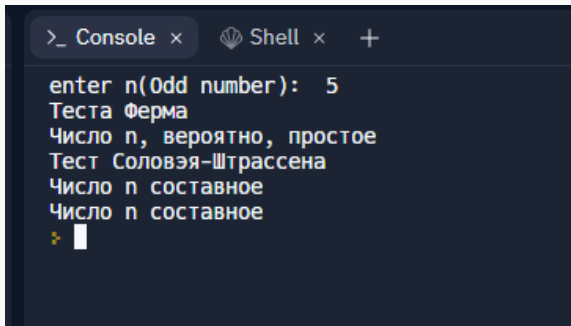
Figure 4: Функция для алгоритма Соловья-Штрассена

Реализация алгоритма Миллера-Рабина

Функция MillerRabin для алгоритма Миллера-Рабина. (рис. 5)

```
72 def toBinary(n):
73     r = []
74     while (n > 0):
75         r.append(n % 2)
76         n = n / 2
77     return r
78
79 def MillerRabin(n, s = 10):
80
81     for j in range(1, s + 1):
82         a = random.randint(1, n - 1)
83         b = toBinary(n - 1)
84         d = 1
85         for i in range(len(b) - 1, -1, -1):
86             x = d
87             d = (d * d) % n
88             if d == 1 and x != 1 and x != n - 1:
89                 print("Число n составное") # Составное
90             if b[i] == 1:
91                 d = (d * a) % n
92                 if d != 1:
93                     print("Число n составное") # Составное
94                 print("Число n, вероятно, простое")
95
96 solovoy(n)
97 MillerRabin(n)
```

Figure 5: Функция для алгоритма Миллера-Рабина



```
>_ Console x Shell x +
enter n(Odd number): 5
Теста Ферма
Число n, вероятно, простое
Тест Соловья-Штрассена
Число n составное
Число n составное
❏
```

Figure 6: Результат алгоритмов

Реализовал алгоритмы Ферма, Соловья-Штрассена, Миллера-Рабина и вычисления Якоби.

Спасибо за внимание