

# Математические основы защиты информации и информационной безопасности. Отчет по лабораторной работе №5

## Вероятностные алгоритмы проверки чисел на простоту

Серенко Данил Сергеевич 1132236895

### Содержание

Цель работы .....	1
Выполнение лабораторной работы .....	1
Алгоритм, реализующий тест Ферма.....	1
Символ Якоби .....	2
Тест Соловья-Штрассена .....	3
Тест Миллера-Рабина.....	4
Результат работы программы .....	5
Выводы .....	6
Список литературы .....	6

### Цель работы

Освоить на практике алгоритмы проверки чисел на простоту.

### Выполнение лабораторной работы

Требуется реализовать:

1. Алгоритм, реализующий тест Ферма
2. Алгоритм вычисления символа Якоби
3. Алгоритм, реализующий тест Соловья-Штрассена
4. Алгоритм, реализующий тест Миллера-Рабина.

### Алгоритм, реализующий тест Ферма

Алгоритм основан на малой теореме Ферма, которая утверждает, что если  $p$  - простое число, то для любого целого числа  $a$ , не являющегося кратным  $p$ , выполняется  $a^{p-1} \equiv 1 \pmod{p}$ . Алгоритм выбирает случайные значения  $a$  и проверяет условие. Если оно не выполняется для какого-либо  $a$ , то  $p$  считается составным. Если оно выполняется для всех выбранных  $a$ , то  $p$  вероятно является простым.

Реализация на Python представлена на рисунке 1.

```
4 def is_prime_fermat(n, k=5):
5     if n <= 1:
6         return False
7     if n <= 3:
8         return True
9     for _ in range(k):
10        a = random.randint(2, n - 2)
11        if pow(a, n - 1, n) != 1:
12            return False
13    return True
```

*fermat*

### Символ Якоби

Символ Якоби обобщает символ Лежандра и используется для определения вычетов в кольце вычетов по модулю  $n$ . Для нечетного простого числа  $p$  и целого числа  $a$ , символ Якоби  $Jacobi(a, p)$  равен 1, если  $a$  является квадратичным вычетом по модулю  $p$ , -1, если  $a$  является квадратичным невычетом, и 0, если  $a$  кратно  $p$ . Символ Якоби используется в различных алгоритмах для проверки простоты и для решения квадратичных уравнений по модулю.

Реализация на Python представлена на рисунке 2.

```

15
16 def jacobi_symbol(a, n):
17     if n % 2 == 0 or n <= 0:
18         raise ValueError("n должно быть нечетным и положительным")
19     a = a % n
20     t = 1
21     while a != 0:
22         while a % 2 == 0:
23             a /= 2
24             r = n % 8
25             if r == 3 or r == 5:
26                 t = -t
27
28         a, n = n, a
29         if a % 4 == 3 and n % 4 == 3:
30             t = -t
31         a = a % n
32     if n == 1:
33         return t
34     else:
35         return 0
36

```

*jacobi*

## Тест Соловея-Штрассена

Этот алгоритм использует символ Якоби и проверяет, является ли число простым. Алгоритм выбирает случайное целое число  $a$  и проверяет два условия: 1)  $a$  не делится на  $n$ , и 2) символ Якоби  $\text{Jacobi}(a, n)$  равен результату вычисления с использованием символа Лежандра. Если оба условия выполняются для всех выбранных  $a$ , то  $n$  вероятно является простым числом.

Реализация на Python представлена на рисунке 3.

```

37
38 def is_prime_solovay_strassen(n, k=5):
39     if n <= 1:
40         return False
41     if n <= 3:
42         return True
43
44     def legendre(a, p):
45         return pow(a, (p - 1) // 2, p)
46
47     for _ in range(k):
48         a = random.randint(2, n - 2)
49         x = legendre(a, n)
50         y = jacobi_symbol(a, n)
51         if x != y % n:
52             return False
53     return True
54

```

*solovay\_strassen*

## Тест Миллера-Рабина

Этот алгоритм также использует вероятностный метод для проверки простоты числа. Алгоритм выбирает случайное целое число  $a$  и разлагает  $n - 1$  на  $2^s * d$ , где  $s$  - четное, и  $d$  нечетное. Затем алгоритм проверяет условия Миллера-Рабина: 1)  $a^d \equiv 1 \pmod{n}$ , и 2) для всех  $i$  от 0 до  $s-1$ ,  $a^{2^i * d} \equiv -1 \pmod{n}$  или  $a^{2^i * d} \equiv 1 \pmod{n}$ . Если оба условия выполняются для всех выбранных  $a$ , то  $n$  вероятно является простым числом.

Реализация на Python представлена на рисунке 4.

```

56 def is_prime_miller_rabin(n, k=5):
57     if n <= 1:
58         return False
59     if n <= 3:
60         return True
61
62     def miller_rabin_test(a, s, d, n):
63         x = pow(a, d, n)
64         if x == 1 or x == n - 1:
65             return True
66
67         for _ in range(s - 1):
68             x = (x * x) % n
69             if x == n - 1:
70                 return True
71         return False
72     s, d = 0, n - 1
73     while d % 2 == 0:
74         s += 1
75         d //= 2
76
77     for _ in range(k):
78         a = random.randint(2, n - 2)
79         if not miller_rabin_test(a, s, d, n):
80             return False
81
82     return True

```

*miller\_rabin*

## Результат работы программы

функция запуска и выходные значения программы.

```
84 n = 23
85 print("тест Ферма: ")
86 if is_prime_fermat(n):
87     print(f"{n} вероятно простое")
88 else:
89     print(f"{n} составное")
90 b = 13
91 a = 6
92 symbol = jacobi_symbol(a, b)
93 print(f"Символ Якоби ({a}/{b}) = {symbol}")
94 print("тест соловья-Штрассена: ")
95 if is_prime_solovay_strassen(n):
96     print(f"{n} вероятно простое")
97 else:
98     print(f"{n} составное")
99 print("тест Миллера-Рабина: ")
100 if is_prime_miller_rabin(n):
101     print(f"{n} вероятно простое")
102 else:
103     print(f"{n} составное")
```

C:\Users\Nitro\AppData\Local\Programs\Python\Python39\python.exe C:\Users\Nitro\lab\_5.py

тест Ферма:  
23 вероятно простое  
Символ Якоби (6/13) = -1  
тест соловья-Штрассена:  
23 вероятно простое  
тест Миллера-Рабина:  
23 вероятно простое

*main*

Выходные значения программы.

## Выводы

В результате выполнения работы я освоил на практике применение алгоритмов проверки чисел на простоту.

## Список литературы

1. Методические материалы курса