

Лабораторная работа 5

Попов Дмитрий Павлович, НФИмд-01-23

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра математического моделирования и искусственного интеллекта

ПРЕЗЕНТАЦИЯ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

дисциплина: Математические основы защиты информации и информационной безопасности

Преподаватель: Кулябов Дмитрий Сергеевич

Студент: Попов Дмитрий Павлович

Группа: НФИмд-01-23

МОСКВА

2023 г.

Прагматика выполнения лабораторной работы

Прагматика выполнения 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.

Цель работы

Цель работы

Освоение на практике алгоритмов проверки чисел на простоту

Выполнение лабораторной работы

1.1 Алгоритм, реализующий тест Ферма

Алгоритм основан на малой теореме Ферма, которая утверждает, что если n - простое число, то для любого целого числа a , не являющегося кратным n , выполняется $a^{(n-1)} \not\equiv 1 \pmod{n}$. Алгоритм выбирает случайные значения a и проверяет условие. Если оно не выполняется для какого-либо a , то n считается составным. Если оно выполняется для всех выбранных a , то n вероятно является простым.

1.2 Алгоритм, реализующий тест Ферма

```
def ferma(n, k=10):  
    if n <= 1:  
        return False  
    if n <= 3:  
        return True  
    for _ in range(k):  
        a = random.randint(2, n - 2)  
        if pow(a, n - 1, n) != 1:  
            return False  
    return True
```

Figure 1: ferma

2.1 Символ Якоби

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

2.2 Символ Якоби

```
def jacobi(n, a):  
    if n % 2 == 0 or n <= 0:  
        raise ValueError("условие n >= 3 и n нечетное - не выполнено")  
    a = a % n  
    g = 1  
    while a != 0:  
        while a % 2 == 0:  
            a /= 2  
            r = n % 8  
            if r == 3 or r == 5:  
                g = -g  
        a, n = n, a  
        if a % 4 == 3 and n % 4 == 3:  
            g = -g  
        a = a % n  
    if n == 1:  
        return g  
    else:  
        return 0
```

Figure 2: jacobi

3.1 Тест Соловья-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 $Jacobi(a, n)$ равен результату вычисления с использованием символа Лежандра. Если оба условия выполняются для всех выбранных a , то n вероятно является простым числом.

3.2 Тест Соловья-Штрассена

```
def strassen(n, k=10):  
    if n <= 1:  
        return False  
    if n <= 3:  
        return True  
    for _ in range(k):  
        a = random.randint(2, n - 2)  
        r = pow(a, (n - 1) // 2, n)  
        s = jacobi(n=n, a=a)  
        if r != s % n:  
            return False  
    return True
```

Figure 3: solovay_strassen

4.1 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) $a^d \not\equiv 1 \pmod{n}$, и 2) для всех i от 0 до $s-1$, $a^{2^i * d} \not\equiv -1 \pmod{n}$ или $a^{2^i * d} \equiv 1 \pmod{n}$. Если оба условия выполняются для всех выбранных a , то n вероятно является простым числом.

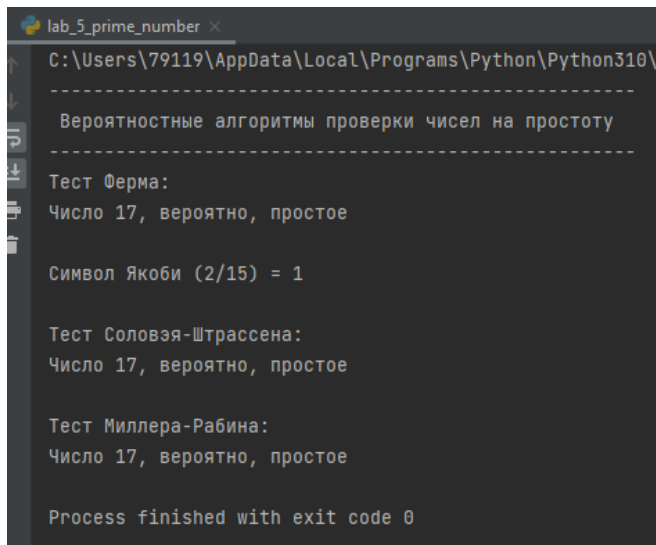
4.2 Тест Миллера-Рабина

```
def miller_rabin(n, k=10):  
    def miller_rabin_test(a, s, d, n):  
        x = pow(a, d, n)  
        if x == 1 or x == n - 1:  
            return True  
        for _ in range(s - 1):  
            x = (x * x) % n  
            if x == n - 1:  
                return True  
        return False  
    if n <= 1:  
        return False  
    if n <= 3:  
        return True  
    s, d = 0, n - 1  
    while d % 2 == 0:  
        s += 1  
        d //= 2  
    for _ in range(k):  
        a = random.randint(2, n - 2)  
        if not miller_rabin_test(a, s, d, n):
```

5.1 Результат работы программы

```
print("Тест Ферма: ")
n = 17
if ferma(n):
    print(f"Число {n}, вероятно, простое")
else:
    print(f"Число {n} составное")
print()
n1 = 2
n2 = 15
symbol = jacobi(n=n2, a=n1)
print(f"Символ Якоби ({n1}/{n2}) = {symbol}")
print()
print("Тест Соловья-Штрассена: ")
if strassen(n):
    print(f"Число {n}, вероятно, простое")
else:
    print(f"Число {n} составное")
print()
print("Тест Миллера-Рабина: ")
if miller_rabin(n):
    print(f"Число {n}, вероятно, простое")
else:
```


5.2 Результат работы программы



```
lab_5_prime_number x
C:\Users\79119\AppData\Local\Programs\Python\Python310\
-----
Вероятностные алгоритмы проверки чисел на простоту
-----
Тест Ферма:
Число 17, вероятно, простое

Символ Якоби (2/15) = 1

Тест Соловья-Штрассена:
Число 17, вероятно, простое

Тест Миллера-Рабина:
Число 17, вероятно, простое

Process finished with exit code 0
```

Figure 6: output

Вывод

Вывод

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

