

Лабораторная работа №5.

Вероятностные алгоритмы проверки чисел на простоту

Дисциплина: Математические основы защиты информации и информационной безопасности

Студентка: Царитова Нина Аведиковна

Группа: НФИмд-02-23

Преподаватель: д-р.ф.-м.н., проф. Кулябов Дмитрий Сергеевич

Цели и задачи работы

Целью данной лабораторной работы является ознакомление с вероятностными алгоритмами проверки чисел на простоту.

Реализовать все рассмотренные в инструкции к лабораторной работе алгоритмы проверки чисел на простоту программно.

Ход выполнения и результаты

Входные данные

```
#ОБЩИЙ СЛУЧАЙ  
#вообще говоря, число a задается случайно (в зависимости от алгоритма)  
a=12  
n=17#должно удовлетворять определенным условиям (в зависимости от алгоритма)
```

Figure 1: Входные данные для реализации алгоритмов проверки чисел на простоту

Алгоритм, реализующий тест Ферма. Реализация

```
def test_ferma(a,n):  
    '''  
    Функция, реализующая тест Ферма по соотв. алгоритму  
    '''  
    r=(a**(n-1))%n  
    if r==1:  
        print('Число n=',n,', вероятно, ПРОСТОЕ')  
    else:  
        print('Число n=',n,', вероятно, СОСТАВНОЕ')  
test_ferma(a,n)
```

Figure 2: Реализация алгоритма теста Ферма

Алгоритм, реализующий тест Ферма. Результаты

⇒ Число $n = 17$, вероятно, ПРОСТОЕ

Figure 3: Результат реализации алгоритма теста Ферма

Алгоритм вычисления символа Якоби. Реализация

```
def primefactors(n):  
    '''  
    Функция для расчета k,a1  
    (предварительно разложить число на множители с использованием 2)  
    '''  
    list_2=[]  
    m=n  
  
    while m%2==0:  
        list_2.append(2)  
        m=m/2  
    k=len(list_2)  
    a1=int(n/(2**k))  
    return k,a1
```

Figure 4: Реализация алгоритма вычисления символа Якоби 1 часть

Алгоритм вычисления символа Якоби. Реализация

```
def Jakobi_symbol(a,n):  
    '''  
    Функция, реализующая поиск символа Якоби по соотв. алгоритму  
    '''  
    g=1  
    while True:  
        if a==0:  
            res=0  
            break  
        if a==1:  
            res=g  
            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
```

Figure 5: Реализация алгоритма вычисления символа Якоби 2 часть

Алгоритм вычисления символа Якоби. Реализация

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

Figure 6: Реализация алгоритма вычисления символа Якоби 3 часть

Алгоритм вычисления символа Якоби. Результат



Figure 7: Результат реализации алгоритма вычисления символа Якоби

Алгоритм, реализующий тест Соловья-Штрассена. Реализация

```
def solovey_strassen(a,n):  
    '''  
    Функция, реализующая тест Соловья-Штрассена  
    '''  
  
    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')  
    solovey_strassen(a,n)
```

Figure 8: Реализация алгоритма теста Соловья-Штрассена

Алгоритм, реализующий тест Соловья-Штрассена. Результат

Число $n = 17$, вероятно, ПРОСТОЕ

Figure 9: Результат реализации алгоритма теста Соловья-Штрассена

Алгоритм, реализующий тест Миллера-Рабина. Реализация

```
def miller_rabin(a,n):  
    '''  
    Функция, реализующая тест Миллера-Рабина  
    '''  
    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=j+1  
        if (y!=n-1):  
            return 'Число n=',n, 'СОСТАВНОЕ'  
    return 'Число n=',n, 'вероятно, ПРОСТОЕ'  
miller_rabin(a,n)
```

```
☞ ( 'Число n=', 17, ', вероятно, ПРОСТОЕ' )
```

Figure 11: Результат реализации алгоритма теста Миллера-Рабина

В результате выполнения данной лабораторной работы нам удалось осуществить программно алгоритмы, рассмотренные в описании к лабораторной работе.