

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук
Кафедра прикладной информатики и теории вероятностей

Отчёт по лабораторной работе №5

*Дисциплина: Математические основы защиты информации и
информационной безопасности*

Студент: Майорова О.А., НФИмд-02-21
Преподаватель: д.ф.-м.н. Кулябов Д.С.

Москва 2021

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Выводы	14
	Список литературы	15

List of Figures

4.1	Проверка функции теста Ферма	10
4.2	Проверка функции вычисления символа Якоби	11
4.3	Проверка функции теста Соловья-Штрассена	12
4.4	Проверка функции теста Миллера-Рабина	13

List of Tables

1 Цель работы

Цель: Ознакомиться с некоторыми вероятностными алгоритмами проверки чисел на простоту.

2 Задание

Программно реализовать следующие вероятностные алгоритмы проверки чисел на простоту: тест Ферма, тест Соловья-Штрассена, тест Миллера-Рабина. А также алгоритм вычисления символа Якоби.

3 Теоретическое введение

Нахождение алгоритма, который правильно и эффективно проверяет очень большое целое число и устанавливает: данное число – простое или же составное, — всегда было проблемой в теории чисел и, следовательно, в криптографии. Алгоритмы, которые решают эту проблему, могут быть разделены на две обширные категории — детерминированные и вероятностные. Детерминированный алгоритм всегда даёт точный ответ. А вероятностные тесты в большинстве своём однозначно определяют, если число составное, но не обеспечивают математического доказательства простоты, когда оно определяется вероятно простым. Несмотря на это, детерминированные алгоритмы обычно менее эффективны, чем вероятностные. Это обусловлено тем, что детерминированные, как правило, требуют больших вычислительных мощностей, и в то же время для вероятностных можно получить вероятность ошибки настолько маленькую, что это почти гарантирует, что алгоритм вырабатывает правильный ответ [1,2].

Первый вероятностный метод, который мы рассмотрим, — испытание простоты чисел тестом Ферма. В основе теста лежит малая теорема Ферма:

$$a^{n-1} \equiv 1 \pmod{n}.$$

Если n — целое число, чья простота под вопросом, нахождение такого a в интервале $1 \leq a \leq n - 1$, что это равенство не выполняется, докажет, что n — составное. И наоборот, если найти целое число a , что равенство выполняется, утверждается, что n — простое, в том смысле, что оно удовлетворяет теореме Ферма для основания a . Отсюда вытекает следующее определение. Составное n такое, что равенство выполняется, называется псевдопростым по основанию a

[1,2].

Другой тест - тест Соловья-Штрассена использует критерий Эйлера для определения значения символа Лежандра (квадратичного характера числа по простому модулю). В самом тесте, естественно, вычисляется символ Якоби по модулю n . Нечётное целое число n является простым тогда и только тогда, когда для всех чисел $a : 1 \leq a \leq n - 1$ выполняется сравнение вида:

$$a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \pmod{n}.$$

Если n простое, то данное сравнение, очевидно, выполнено в силу свойств символа Лежандра [3].

Рассмотрим ещё один метод – тест Миллера-Рабина. Пусть n – нечётное число большее 1. Число $n - 1$ однозначно представляется в виде $n - 1 = 2^s t$, где s – целое, t – нечётное. Целое число $a \in (1; n)$, называется свидетелем простоты числа n , если выполняется одно из условий:

$$a^t \equiv 1 \pmod{n},$$

или существует целое число $k : 0 \leq k < s$, такое, что

$$a^{2^k t} \equiv n - 1 \pmod{n}.$$

Таким образом, для данного n находятся подходящие s и t , выбирается случайное число $a \in (1; n)$. Если $a \in (1; n)$ не является свидетелем простоты числа n , то выдается ответ «составное», и алгоритм завершается. Иначе, выбирается новое случайное число a , и процедура проверки повторяется. После нахождения заданного числа свидетелей простоты выдаётся ответ «вероятно простое», и алгоритм завершается. Данный алгоритм интересен также тем, что для r прошедших проверку a , вероятность того, что кандидат окажется составным не превосходит 4^{-r} [2].

4 Выполнение лабораторной работы

Для выполнения лабораторной работы был выбран язык Python. Далее реализуем представленные алгоритмы в виде функций в соответствии псевдокоду из задания к лабораторной работе. Перед началом работы подключим библиотеку `numpy`:

```
import numpy as np
```

Сначала реализуем тест Ферма:

```
# n - нечётное целое число >= 5
```

```
def Fermat(n):  
    a = np.random.randint(2, n-1)  
    r = a**(n-1) % n  
    if r == 1:  
        return 'Число n, вероятно, простое'  
    else:  
        return 'Число n составное'
```

Результатом запуска функции будет рис. 4.1.

1	Fermat(47)
'Число n, вероятно, простое'	

1	Fermat(69)
'Число n составное'	

Figure 4.1: Проверка функции теста Ферма

Так как для теста Соловья-Штрассена необходимо вычисление символа Якоби, реализуем алгоритм его вычисления:

n - нечётное целое число ≥ 3

a - целое число $0 \leq a < n$

```
def Jacobi(a, n):
    g = 1
    while True:
        if a == 0:
            return 0
        if a == 1:
            return g

    k = 0
    a1 = a
    while a1 % 2 == 0:
        a1 = int(a1 / 2)
        k += 1

    if k % 2 == 0:
        s = 1
    else:
        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:
        return g*s

    if (n-3) % 4 == 0 and (a1-3) % 4 == 0:
        s = -s

    a = n % a1
    n = a1
    g = g*s

```

Результатом запуска функции будет рис. 4.2.

1	Jacobi(1, 3)
1	
1	Jacobi(2, 3)
-1	
1	Jacobi(3, 3)
0	

Figure 4.2: Проверка функции вычисления символа Якоби

Теперь реализуем тест Соловья-Штрассена:

n - нечётное целое число ≥ 5

```

def SolStr(n):
    a = np.random.randint(2, n-2)

```

```

r = a**int((n-1)/2) % n

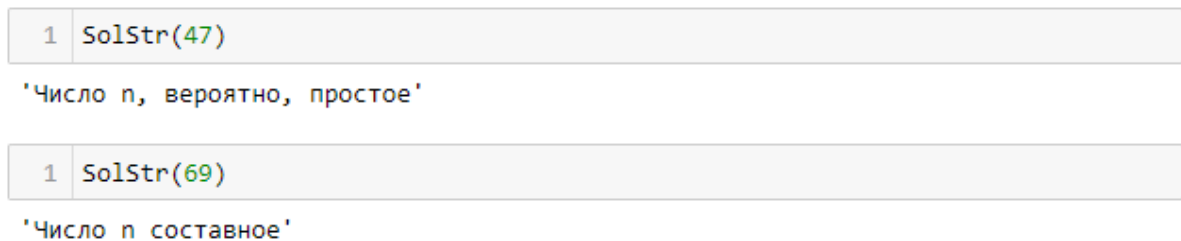
if r != 1 and r != n-1:
    return 'Число n составное'

s = Jacobi(a, n)

if r != s % n:
    return 'Число n составное'
else:
    return 'Число n, вероятно, простое'

```

Результатом запуска функции будет рис. 4.3.



```

1 SolStr(47)
'Число n, вероятно, простое'

1 SolStr(69)
'Число n составное'

```

Figure 4.3: Проверка функции теста Соловья-Штрассена

Наконец, реализуем последний алгоритм - тест Миллера-Рабина:

n - нечётное целое число ≥ 5

```

def MillRab(n):
    s = 0
    r = n - 1
    while r % 2 == 0:
        r = int(r / 2)
        s += 1

```

```

a = np.random.randint(2, n-2)

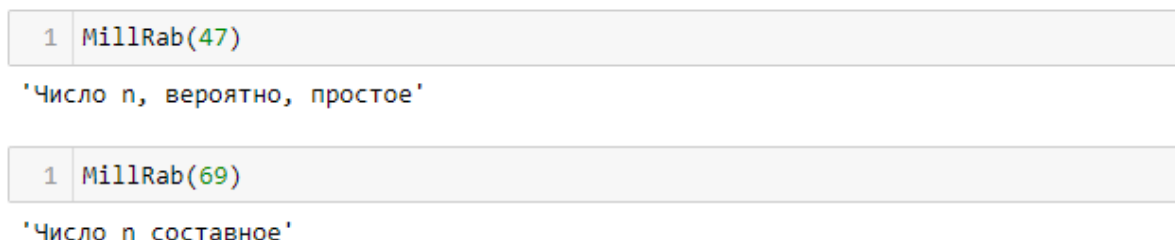
y = a**r % n

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 составное'
        j += 1
    if y != n-1:
        return 'Число n составное'

return 'Число n, вероятно, простое'

```

Результатом запуска функции будет рис. 4.4.



```

1 MillRab(47)
'Число n, вероятно, простое'

1 MillRab(69)
'Число n составное'

```

Figure 4.4: Проверка функции теста Миллера-Рабина

Можно видеть, что все функции тестов дали одинаковый верны результат, и функция вычисления символа Якоби также работает корректно.

5 Выводы

Таким образом, была достигнута цель, поставленная в начале лабораторной работы. Было осуществлено знакомство с некоторыми вероятностными алгоритмами проверки чисел на простоту: тест Ферма, тест Соловья-Штрассена и тест Миллера-Рабина. Также была получена реализация на языке Python рассмотренных алгоритмов, а также алгоритма вычисления символа Якоби.

Список литературы

1. Лекция 12: Простые числа [Электронный ресурс]. НОУ «ИНТУИТ», 2021. URL: <https://intuit.ru/studies/courses/552/408/lecture/9368>.
2. Алгоритмы, используемые при реализации асимметричных криптосхем [Электронный ресурс]. CryptoWiki: Энциклопедия теоретической и прикладной криптографии, 2013. URL: http://cryptowiki.net/index.php?title=Алгоритмы_используемые_при_реализации_асимметричных_криптосхем&oldid=8198.
3. Мухачев В.А., Хорошко В.А. Методы практической криптографии. ООО «Полиграф-Консалтинг», 2005.