Отчёт по лабораторной работе №5

Вероятностные алгоритмы проверки чисел на простоту

Бакундукизе Эжид Принц НФИмд-01-21

Содержание

List of Figures

1 Цель работы

Изучение алгоритмов Ферма, Соловэя-Штрассена, Миллера-Рабина.

2 Теоретические сведения

Для построения многих систем защиты информации требуются простые числа большой разрядности. В связи с этим актуальной является задача тестирования на простоту натуральных чисел.

Существует два типа критериев простоты: детерминированные и вероятностные. Детерминированные тесты позволяют доказать, что тестируемое число - простое. Практически применимые детерминированные тесты способны дать положительный ответ не для каждого простого числа, поскольку используют лишь достаточные условия простоты [1]. Детерминированные тесты более полезны, когда необходимо построить большое простое число, а не проверить простоту, скажем, некоторого единственного числа. В отличие от детерминированных, вероятностные тесты можно эффективно использовать для тестирования отдельных чисел, однако их результаты, с некоторой вероятностью, могут быть неверными. К счастью, ценой количества повторений теста с модифицированными исходными данными вероятность ошибки можно сделать как угодно малой. На сегодня известно достаточно много алгоритмов проверки чисел на простоту. Несмотря на то, что большинство из таких алгоритмов имеет субэкспоненциальную оценку сложности, на практике они показывают вполне приемлемую скорость работы. На практике рассмотренные алгоритмы чаще всего по отдельности не применяются. Для проверки числа на простоту используют либо их комбинации, либо детерминированные тесты на простоту. Детерминированный алгоритм всегда действует по одной и той же схеме и гарантированно решает поставленную задачу. Вероятностный алгоритм использует генератор случайных

чисел и дает не гарантированно точный ответ. Вероятностные алгоритмы в общем случае не менее эффективны, чем детерминированные (если используемый генератор случайных чисел всегда дает набор одних и тех же чисел, возможно, зависящих от входных данных, то вероятностный алгоритм становится детерминированным)[2].

2.1 Тест Ферма

- Вход. Нечетное целое число $n \geq 5$.
- Выход. «Число n, вероятно, простое» или «Число n составное».

1. Выбрать случайное целое число $a, 2 \leq a \leq n-2$.
2. Вычислить $r = a^{n-1}(modn)$
3. При $r = 1$ результат: «Число n, вероятно, простое». В противном случае результат: «Число n составное».

2.2 Тест Соловэя-Штрассена

- Вход. Нечетное целое число $n \geq 5$.
- Выход. «Число n, вероятно, простое» или «Число n составное».

1. Выбрать случайное целое число $a, 2 \leq a \leq n-2$.
2. Вычислить $r = a^{(\frac{n-1}{2})}(modn)$
3. При $r \neq 1$ и $r \neq n-1$ результат: «Число n составное».
4. Вычислить символ Якоби $s = (\frac{a}{n})$
5. При $r = s(modn)$ результат: «Число n, вероятно, простое». В противном случае результат: «Число n составное».

2.3 Тест Миллера-Рабина.

- Вход. Нечетное целое число $n \geq 5$.

- Выход. «Число n, вероятно, простое» или «Число n составное».

1. Представить $n-1$ в виде $n-1=2^s r$, где r - нечетное число
2. Выбрать случайное целое число $a, 2 \leq a \leq n-2$.
3. Вычислить $y = a^r (mod n)$
4. При $y \neq 1$ и $y \neq n-1$ выполнить действия
    - Положить $j = 1$
    - Если $j \leq s-1$ и $y \neq n-1$ то
        - Положить $y = y^2 (mod n)$
        - При $y = 1$ результат: «Число n составное».
        - Положить $j = j+1$
    - При $y \neq n-1$ результат: «Число n составное».
5. Результат: «Число n, вероятно, простое».

3 Выполнение работы

3.1 Реализация алгоритмов на языке Python

```
import random


# 1. Тест Ферма
# n - число, которое проверяется на простоту (больше или равно 5)
# test_count - количество экспериментов

def ferma(n, test_count):
    for i in range(test_count):
        a = random.randint(2, n - 2) # выбираем число от 2 до n - 2
        r = a ** (n - 1) % n
        if (r != 1):
            print("Число n составное")
            return False
    print("Число n вероятно простое")
    return True



# 2. Тест Соловэя-Штрассена

# основан на алгоритме нахождения числа Якоби
```

```
# алгоритм Якоби (символ якоби = a/n)
# n - целое число больше или равно 3
# a - число от 0 до n
# k - число прогонов
# a1 - нечетное число

def calculateJacobian(a, n, iterations):

    g = 1
    s = 0
    res = 0
    a1 = 3

    for k in range(iterations):

        if (a == 0):
            return 0

        if (a == 1):
            return g

        a = 2**k*a1

        if (k%2 == 0):

            s = 1

        else:
```

```
            if (n%8 == -1 or n%8 == 1):
                s = 1
            if (n%8 == -3 or n%8 == 3):
                s = -1
        if (a1 == 1):
            res = g*s
            return res
        if (n%4 == 3 and a1%4 == 3):
            s = -s
            a = n%a1
            n = a1
            g=g*s
            k+=1

# n - число, которое проверяется на простоту (больше или равно 5)
def solovoyStrassen(n, iterations):
    for i in range(iterations):
```

```
        a = random.randint(2, n - 2) # выбираем число от 2 до n - 2
        r = a ** ((n - 1)/2) % n
        if (r!=1 and  r!=n-1):
            print("Число n составное")
            return False
        else:
            s=calculateJacobian(a, n, 500)

            if (r%n == s):
                print ("Число n составное")
                return False

            else:
                print ("Число n вероятно простое")
                return True



# 3. Тест Миллера Рабина

# n - число, которое проверяется на простоту (больше или равно 5)

def miller_rabin(n):

    j = 0
    n1 = n-1
    s = 0
```

```
while (n1 % 2 == 0):
    n1 /= 2
    s += 1

n1=int(n1)


a = random.randint(2, n - 2) # выбираем число от 2 до n - 2

y = pow(a,n1,n)

if (y != 1 and y != n-1):

    for j in range(1,s):

        y = pow(y,2,n)

        if (y==1):

            print("Число n составное")
            return False

        j += 1

    if (y != n - 1):

        print("Число n составное")
        return False
```

```
    print ("Число n вероятно простое")
    return True


def main():
    n = int(input("Введите число "))
    print("Тест ферма для числа ", n )
    ferma(n, 500)
    print("Тест Соловэя-Штрассена для числа ", n )
    solovoyStrassen(n, 500)
    print("Тест Миллера Рабина для числа", n )
    miller_rabin(n)
```

3.2 Контрольный пример

```
main()

Введите число 17
Тест ферма для числа  17
Число n вероятно простое
Тест Соловэя-Штрассена для числа  17
Число n вероятно простое
Тест Миллера Рабина для числа 17
Число n вероятно простое
```

Figure 3.1: Работа алгоритмов

4 Выводы

В ходе выполнения данной лабораторной работы мы изучили вероятностные алгоритмы проверки чисел на простоту, в частности, были рассмотрены алгоритмы Ферма, Соловэя-Штрассена и Миллера-Рабина. Перечисленные алгоритмы были реализованы программно, представлены результаты работы алгоритмов.

Список литературы

1. Алгоритм проверки на простоту
2. Алгоритмы тестирования на простоту и факторизации