Защита лабораторной работы №5

Вероятностные алгоритмы проверки чисел на простоту

Бурдина К. П.

9 ноября 2023

Российский университет дружбы народов, Москва, Россия

Докладчик

* Бурдина Ксения Павловна
* студентка группы НФИмд-02-23
* студ. билет № 1132236896
* Российский университет дружбы народов
* 1132236896@rudn.ru

Вводная часть

Цель выполнения лабораторной работы

- Освоение вероятностных алгоритмов проверки чисел на простоту
- Программная реализация представленных алгоритмов проверки чисел на простоту

Пусть a - целое число. Числа $\pm 1, \pm a$ называются *тривиальными делителями* числа a.

Целое число $p \in Z/\{0\}$ называется *простым*, если оно не является делителем единицы и не имеет других делителей, кроме тривиальных. В противном случае число $p \in Z/\{-1, 0, 1\}$ называется *составным*.

Например, числа $\pm 2, \pm 3, \pm 5, \pm 7, \pm 11, \pm 13, \pm 17, \pm 19, \pm 23, \pm 29$ являются простыми.

Теоретические сведения

Алгоритмы проверки на простоту можно разделить на вероятностные и детерминированные.

Детерминированный алгоритм всегда действует по одной и той же схеме и гарантированно решает поставленную задачу (или не дает никакого ответа).

Вероятностный алгоритм использует генератор случайных чисел и дает не гарантированно точный ответ.

Схема вероятностного алгоритма проверки числа на простоту:

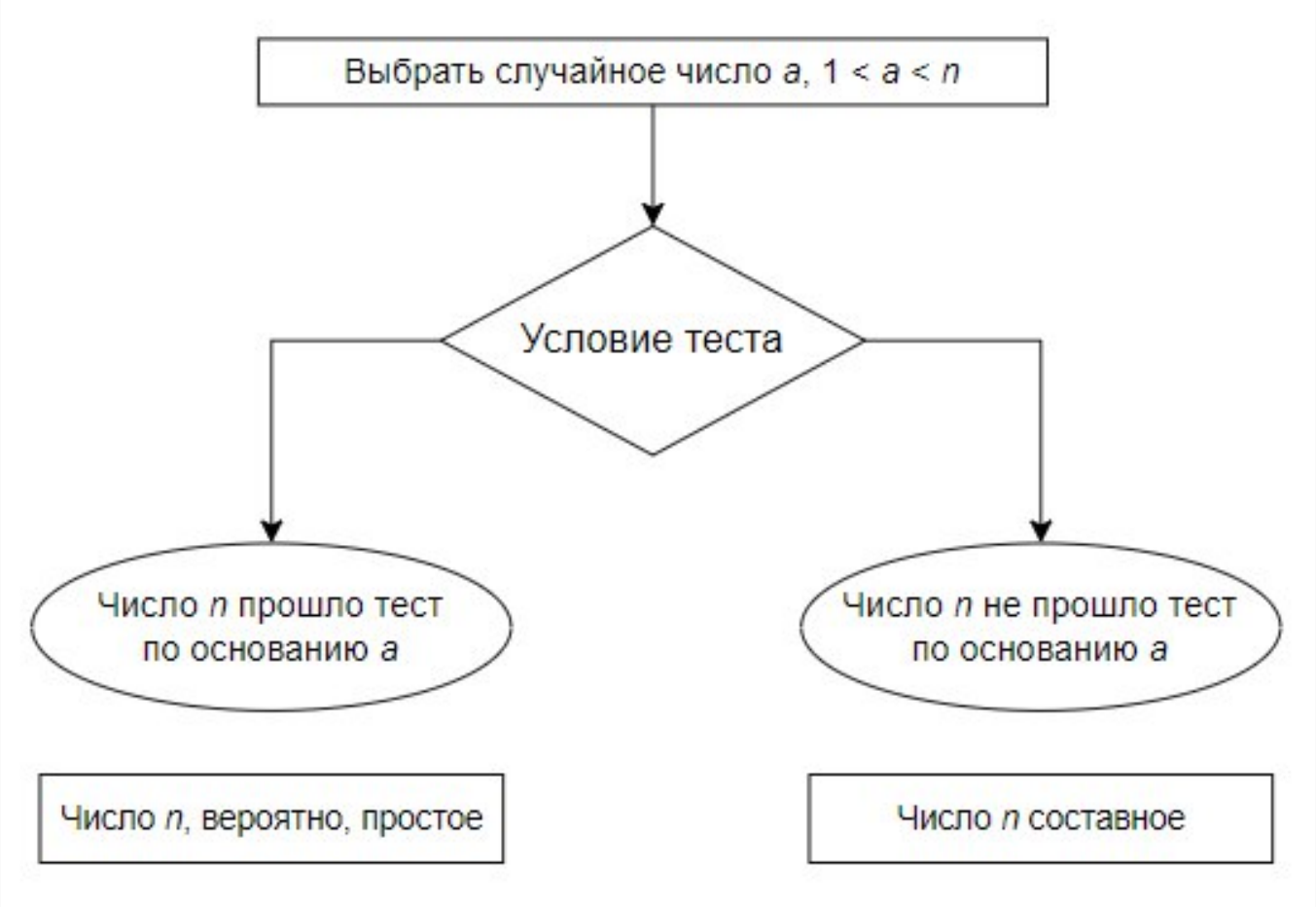

Рис. 1: Схема проверки числа

Результат выполнения лабораторной работы

Результат выполнения лабораторной работы

Постановка задачи:

Реализовать вероятностные алгоритмы проверки чисел на простоту, такие как: - Алгоритм, реализующий тест Ферма - Алгоритм вычисления символа Якоби - Алгоритм, реализующий тест Соловэя-Штрассена - Алгоритм, реализующий тест Миллера-Рабина

Результат выполнения лабораторной работы

Алгоритм, реализующий тест Ферма:

```
import random
def Ferma(n, count):
    for i in range(count):
        a = random.randint(2, n-1)
        if (a**(n-1) % n != 1):
            print("Число составное")
            return False
    print("Число, вероятно, простое")
    return True
```

Рис. 2: Тест Ферма

Результат выполнения лабораторной работы

Алгоритм вычисления символа Якоби:

```
def Jacob(a, n):
    g = 1
    if (a == 0):
        return 0
    if (a == 1):
        return g
    while (a):
        while(a % 2 == 0):
            a = a // 2
            if (n % 8 == 1 or n % 8 == 3):
                g = -g
        a, n = n, a
        if (a % 4 == 3 and n % 4 == 3):
            g = -g
        a = a % n
        if (a > n // 2):
            a -= n
    if (n == 1):
        return g
    return 0
```

Рис. 3: Символ Якоби

Результат выполнения лабораторной работы

Алгоритм, реализующий тест Соловэя-Штрассена:

```
def SolStras(r, count):
    if (r < 2):
        print("Число составное")
        return False
    if (r != 2 and r % 2 == 0):
        print("Число составное")
        return False
    for i in range(count):
        a = random.randrange(r - 1) + 1
    s = (r + Jacob(a, n)) % r
    mod = modul(a, (r - 1) / 2, r)
    if (s == 0 or mod != s):
        print("Число составное")
        return False
    else:
        print("Число, вероятно, простое")
    return True
```

Рис. 4: Тест Соловэя-Штрассена

Результат выполнения лабораторной работы

Алгоритм, реализующий тест Миллера-Рабина:

```
def MilRab(n):
    if n != int(n):
        print("Число составное")
        return False
    n = int(n)
    if n == 0 or n == 1 or n == 4 or n == 6 or n == 8 or n == 9:
        print("Число составное")
        return False
    if n == 2 or n == 3 or n == 5 or n == 7:
        print("Число, вероятно, простое")
        return True
    s = 0
    r = n - 1
    while r % 2 == 0:
        r >>= 1
        s += 1
    assert(2**s * r == n - 1)
```

Рис. 5: Тест Миллера-Рабина

Результат выполнения лабораторной работы

Проверка работы алгоритмов:

```
n = 1909
```

```
Ferma(n, 1000)
```

```
Число составное

False
```

```
SolStras(n, 1000)
```

```
Число составное

False
```

```
MilRab(n)
```

```
Число, вероятно, простое
Число составное

False
```

Рис. 6: Проверка работы алгоритмов 1

Результат выполнения лабораторной работы

Проверка работы алгоритмов:

```
n = 1901
```

```
Ferma(n, 1000)
```

```
Число, вероятно, простое

True
```

```
SolStras(n, 1000)
```

```
Число, вероятно, простое

True
```

```
MilRab(n)
```

```
Число составное
Число составное
Число составное
Число составное
Число составное
Число составное
Число составное
Число составное
Число, вероятно, простое

True
```

Рис. 7: Проверка работы алгоритмов 2

Выводы

Выводы

1. Изучили вероятностные алгоритмы проверки чисел на простоту
2. Программно реализовали представленные алгоритмы проверки чисел на простоту