Вероятностные алгоритмы проверки чисел на простоту

Дисциплина: Математические основы защиты информации и информационной безопасности

Студент: Гонсалес Ананина Луис Антонио, 1032175329
Группа: НФИмд-02-21
Преподаватель: д-р.ф.-м.н., проф. Кулябов Дмитрий Сергеевич

11 декабря, 2021, Москва

Цели и задачи

Цель лабораторной работы

Цель данной лабораторной работы- изучить теорию и реализовать все рассмотренные алгоритмы программно.

Выполнение лабораторной работы

Выполнение лабораторной работы

Вопрос определения того, является ли натуральное число N простым, известен как проблема простоты.

Тестом простоты (или проверкой простоты) называется алгоритм, который, приняв на входе число N, позволяет либо не подтвердить предположение о составности числа, либо точно утверждать его простоту. Во втором случае он называется истинным тестом простоты. Таким образом, тест простоты представляет собой только гипотезу о том, что если алгоритм не подтвердил предположение о составности числа N, то это число может являться простым с определённой вероятностью. Это определение подразумевает меньшую уверенность в соответствии результата проверки истинному положению вещей, нежели истинное испытание на простоту, которое даёт математически подтверждённый результат.

Выполнение лабораторной работы 2

Тест простоты Ферма в теории чисел — это тест простоты натурального числа n, основанный на малой теореме Ферма.

Если n — простое число, то оно удовлетворяет сравнению a^n-1 =1 (mod n) для любого a, которое не делится на n.

Выполнение сравнения a^n-1 =1 (mod n) является необходимым, но не достаточным признаком простоты числа. То есть, если найдётся хотя бы одно a, для которого a^n-1 !=1 (mod n), то число n — составное; в противном случае ничего сказать нельзя, хотя шансы на то, что число является простым, увеличиваются. Если для составного числа n выполняется сравнение a^n-1 =1 (mod n), то число n называют псевдопростым по основанию a . При проверке числа на простоту тестом Ферма выбирают несколько чисел a. Чем больше количество a, для которых a^n-1 =1 (mod n),

Выполнение лабораторной работы 3

Тест Соловея — Штрассена — тест всегда корректно определяет, что простое число является простым, но для составных чисел с некоторой вероятностью он может дать неверный ответ.

Алгоритм Соловея — Штрассена параметризуется количеством раундов k. В каждом раунде случайным образом выбирается число a < n. Если НОД(a,n) > 1, то выносится решение, что n составное. Иначе проверяется справедливость сравнения a^(n-1)/2=(a/n)(mod n). Если оно не выполняется, то выносится решение, что n — составное. Если это сравнение выполняется, то a является свидетелем простоты числа n. Далее выбирается другое случайное a и процедура повторяется. После нахождения k свидетелей простоты в k раундах выносится заключение, что n является простым числом с вероятностью 1-2^-k.

Тест Миллера — Рабина — вероятностный полиномиальный тест простоты. Тест Миллера — Рабина, наряду с тестом Ферма и тестом Соловея — Штрассена, позволяет эффективно определить, является ли данное число составным. Однако, с его помощью нельзя строго доказать простоту числа. Тем не менее тест Миллера — Рабина часто используется в криптографии для получения больших случайных простых чисел.

Как и тесты Ферма и Соловея — Штрассена, тест Миллера — Рабина опирается на проверку ряда равенств, которые выполняются для простых чисел. Если хотя бы одно такое равенство не выполняется, это доказывает что число составное.

Для теста Миллера — Рабина используется следующее

Результат выполнения работы 1

```
In [1]: import numpy as np

In [2]: #Тест Ферма
        def ferma(n):
            a=np.random.randint(2,n-2)
            r=(a**(n-1))%n
            if r==1:
                return f'Число n, вероятно, простое'
            else:
                return f'Число n составное'

In [3]: ferma(31)

Out[3]: 'Число n, вероятно, простое'

In [4]: #Символ Якобы
        def jacobi(n,a):
            assert n>=3
            assert 0<=a<n

            if a<0:
                return -a/n
            elif a%2==0:
                return (a/2)/n
            elif a==1:
                return 1
            elif a<n:
                return n/a
            else:
                return (a%n)/n
```

Figure 2: Тест Ферма

Результат выполнения работы 2

```
In [5]: #Тест Соловэя
        def solovey(n):
            assert n%2==1 and n>=5 #только можно вводить нечетные чилса
            a = np.random.randint(2, n-2)
            r=(a**((n-1)/2))%n
            if r!=1 and r!=n-1:
                return f'Число n составное'
            else:
                s=jacobi(n,a)
                print(s%n,r)
                if s%n==r:
                    return f'Число n составное'
                else:
                    return f'Число n, вероятно, простое'

In [6]: solovey(19)

        0.15789473684210525 1.0

Out[6]: 'Число n, вероятно, простое'
```

Figure 3: Тест Соловэя

Результат выполнения работы 3

```
In [7]: #Тест Миллера
def view(n):
    assert n%2==0
    init_n=n
    s=0
    t=0
    while 1:
        n/=2
        s+=1
        if n%2==1:
            return s,int(n)

def miller(n):
    assert n>=5 and n%2==1 ##только можно вводить нечетные чилса

    s,r=view(n-1)
    a=np.random.randint(2,n-2)
    y=(a**r)%n
    if y!=1 and y!=n:
        j=1
        if j<=s-1 and y!=n-1:
            y=(y**2)%n
            if y==1:
                return f'Число n составное'
            else:
                j+=1
        if y!=n-1:
            return f'Число n сосатвное'
    return f'Число n, вероятно, простое'
```

```
In [8]: miller(23)

Out[8]: 'Число n, вероятно, простое'

In [ ]:
```

Figure 4: Тест Миллера

Выводы

Выводы

В ходе данной лабораторной работы была изучена теория и реализованы все рассмотренные алгоритмы программно.