РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

Отчёт по лабораторной работе №5
Вероятностные алгоритмы проверки чисел на простоту

Дисциплина: *Математические основы защиты информации и информационной безопасности*

Студент: Агеева Анастасия Сергеевна, 1032212304
Группа: НФИмд-02-21
Преподаватель: Кулябов Дмитрий Сергеевич,
д-р.ф.-м.н., проф.

Москва 2021

Содержание

List of Figures

1 Цель работы

Цель данной лабораторной работы изучение алгоритмов проверки чисел на простоту.

2 Задание

1. Реализовать программно алгоритм, реализующий тест Ферма;
2. Реализовать алгоритм вычисления символа Якоби;
3. Реализовать программно алгоритм, реализующий тест Соловэя-Штрассена;
4. Реализовать программно алгоритм, реализующий тест Миллера-Рабина.

3 Теоретическое введение

3.0.1 Простые числа

Просто́е число́ — натуральное (целое положительное) число, имеющее ровно два различных натуральных делителя — единицу и самого себя. Другими словами, число x является простым, если оно больше 1 и при этом делится без остатка только на 1 и на x. К примеру, 5 — простое число, а 6 не является простым числом, так как, помимо 1 и 6, оно также делится на 2 и на 3 [1].

3.1 Алгоритмы поиска и распознавания простых чисел

Простые способы нахождения начального списка простых чисел вплоть до некоторого значения дают решето Эратосфена, решето Сундарама и решето Аткина. Однако, на практике вместо получения списка простых чисел зачастую требуется проверить, является ли данное число простым. Алгоритмы, решающие эту задачу, называются тестами простоты. Существует множество полиномиальных тестов простоты, но большинство их являются вероятностными (например, тест Миллера — Рабина) и используются для нужд криптографии. В 2002 году было доказано, что задача проверки на простоту в общем виде полиномиально разрешима, но предложенный детерминированный тест Агравала — Каяла — Саксены имеет довольно большую вычислительную сложность, что затрудняет его практическое применение. Для некоторых классов чисел существуют специализированные эффективные тесты простоты [1].

3.1.1 Тесты простоты

Тестом простоты (или проверкой простоты) называется алгоритм, который, приняв на входе число, позволяет либо не подтвердить предположение о составности числа, либо точно утверждать его простоту. Во втором случае он называется истинным тестом простоты. Задача теста простоты относится к классу сложности P, то есть время работы алгоритмов её решения зависит от размера входных данных полиномиально, что было доказано в 2002 году. Появление полиномиального алгоритма предсказывалось существованием полиномиальных сертификатов простоты и, как следствие, тем, что задача проверки числа на простоту принадлежала классам NP и co-NP одновременно.

Существующие алгоритмы проверки числа на простоту могут быть разделены на две категории: истинные тесты простоты и вероятностные тесты простоты. Результатом вычислений истинных тестов всегда является факт простоты либо составности числа. Вероятностный тест показывает, является ли число простым с некоторой вероятностью. Числа, удовлетворяющие вероятностному тесту простоты, но являющиеся составными, называются псевдопростыми. Одним из примеров таких чисел являются числа Кармайкла.

Одним из примеров истинных тестов простоты является тест Люка-Лемера для чисел Мерсенна. Очевидный недостаток этого теста заключается в его применимости только к числам определённого вида.

Среди других примеров можно привести основанные на малой теореме Ферма:

- Тест Пепина для чисел Ферма;
- Теорема Прота для чисел Прота;
- Тест Агравала — Каяла — Саксены (первый универсальный, полиномиальный, детерминированный и безусловный тест простоты);
- Тест Люка — Лемера — Ризеля.

А также:

- метод перебора делителей;
- Теорема Вильсона;
- Критерий Поклингтона;
- Тест Миллера;
- Тест Адлемана — Померанса — Румели, усовершенствованный Коэном и Ленстрой;
- Тест простоты с использованием эллиптических кривых.

К вероятностным тестам простоты относят:

- Тест Ферма [2];
- Тест Миллера — Рабина [3];
- Тест Соловея — Штрассена [4];
- Тест Бейли — Померанца — Селфриджа — Уогстаффа.

3.2 Тест Ферма

Если n — *простое* число, то оно удовлетворяет сравнению $a^{n-1} \equiv 1 \pmod{n}$ для любого a, которое не делится на n.

Выполнение сравнения $a^{n-1} \equiv 1 \pmod{n}$ является необходимым, но не достаточным признаком простоты числа. То есть, если найдётся хотя бы одно a, для которого $a^{n-1} \not\equiv 1 \pmod{n}$ 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 $a^{n-1} \equiv 1 \pmod{n}$, 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 $a^{n-1} \equiv 1 \pmod{n}$, тем больше шансы, что число n простое. Однако существуют составные числа, для которых сравнение $a^{n-1} \equiv 1 \pmod{n}$ выполняется для всех a, взаимно простых с n — это *числа Кармайкла*. Чисел Кармайкла — бесконечное множество, наименьшее число Кармайкла — 561. Тем не менее, тест Ферма довольно эффективен для обнаружения составных чисел.

3.3 Тест Соловея-Штрассена

Тест Соловея — Штрассена — вероятностный тест простоты, открытый в 1970-х годах Робертом Мартином Соловеем совместно с Фолькером Штрассеном. Тест всегда корректно определяет, что простое число является простым, но для составных чисел с некоторой вероятностью он может дать неверный ответ. Основное преимущество теста заключается в том, что он, в отличие от теста Ферма, распознает числа Кармайкла как составные.

3.3.1 Обоснование

Тест Соловея — Штрассена опирается на малую теорему Ферма и свойства символа Якоби [5]: Если n — нечетное составное число, то количество целых чисел a, взаимнопростых с n и меньших n, удовлетворяющих сравнению $a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \pmod{n}$, не превосходит $\frac{n}{2}$. Составные числа n удовлетворяющие этому сравнению называются псевдопростыми Эйлера-Якоби по основанию a.

3.3.2 Символ Якоби

Пусть P — нечётное, большее единицы число и $P = p_1 p_2 \dots p_n$ — его разложение на простые множители (среди $p_1, \ \dots, \ p_n$ могут быть равные). Тогда для произвольного целого числа a символ Якоби определяется равенством: $\left(\frac{a}{P}\right) = \left(\frac{a}{p_1}\right)\left(\frac{a}{p_2}\right)\cdots\left(\frac{a}{p_n}\right)$, где $\left(\frac{a}{p_i}\right)$— символы Лежандра. По определению считаем, что $\left(\frac{a}{1}\right) = 1$ для всех a.

Символ Якоби практически никогда не вычисляют по определению. Чаще всего для вычисления используют свойства символа Якоби, главным образом — квадратичный закон взаимности.

Символ Якоби нельзя напрямую использовать для проверки разрешимости квадратичного сравнения. То есть, если задано сравнение $x^2 \equiv a \mod n$, ((1)) то равенство единице символа Якоби $\left(\frac{a}{n}\right)$ вовсе не означает, что данное сравнение разрешимо. Например, $\left(\frac{2}{15}\right) = (-1)^{28} = 1$, но сравнение $x^2 \equiv 2 \mod 15$

не имеет решений (можно проверить перебором). Но если $\left(\frac{a}{n}\right) = -1$, то сравнение (1) не имеет решений.

3.4 Тест Миллера-Рабина

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 утверждение:

Пусть n — простое число и $n - 1 = 2^s d$, где d — нечётно. Тогда для любого a из Z_n выполняется хотя бы одно из условий:

1. $a^d \equiv 1 \pmod{n}$;
2. Существует целое число $r < s$, такое что $a^{2^r d} \equiv -1 \pmod{n}$.

4 Выполнение лабораторной работы

Для выполнения работы импортирую библиотеку random, поскольку понадобится генерация случайных чисел.

Figure 4.1: Импорт библиотеки random

1. **Реализация теста Ферма**

    1. Задам функцию *ferma()*, в которую буду передавать число *n*, проверяемое на простоту. По алгоритму, реализующему тест Ферма, осуществляется проверка простоты числа. В качестве результата возвращается строка, сообщающая простое/составное число или выдающая сообщение об ошибке, в случае ввода некорректных данных.

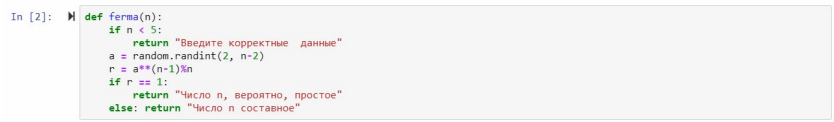

Figure 4.2: Реализация теста Ферма

    2. Вызову функцию для чисел 8 и 7. Алгоритм верно проверяет простоту чисел.

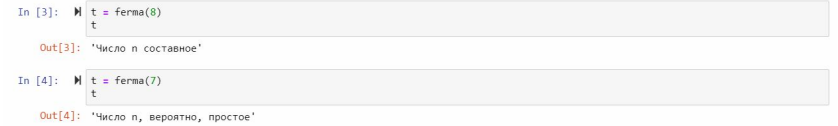

Figure 4.3: Результаты теста Ферма

2. **Реализация алгоритма вычисления символа Якоби**

1. Задам функцию *yakobi()*, в которую буду передавать два числа *a* и *n*. По алгоритму вычисления символа Якоби функция возвращает -1, 0, 1 или сообщение об ошибке, в случае ввода некорректных данных (работа программы прекращается).

```
In [5]:
def yakobi(a, n):
    if n < 3 or a >= n or a < 0:
        print ("Введите корректные  данные")
        return
    g = 1
    while True:
        if a == 0:
            return 0
        if a == 1:
            return g
        k = 0
        a1 = a
        while a1 % 2 == 0:
            k += 1
            a1 /= 2
        if k % 2 == 0:
            s = 1
        elif (n-1)%8 == 0 or (n+1)%8 == 0:
            s = 1
        elif (n-3)%8 == 0 or (n+3)%8 == 0:
            s = -1
        if a1 == 1:
            return s*g
        else:
            if (n - 3)%4 == 0 and (a1 - 3)%4 == 0:
                s = -s
        a = n%a1
        n = a1
        g = g*s
```

Figure 4.4: Реализация алгоритма вычисления символа Якоби

2. Вызову функцию для пар чисел (4, 13) и (4, 2). Алгоритм верно находит символ Якоби для первой пары чисел и выдает сообщение об ошибке для второй пары чисел.

```
In [6]:
y = yakobi(4, 13)
print("Символ Якоби:", y)

Символ Якоби: 1

In [7]:
y = yakobi(4, 2)
print("Символ Якоби:", y)

Введите корректные  данные
Символ Якоби: None
```

Figure 4.5: Результаты алгоритма вычисления символа Якоби

3. **Реализация теста Соловэя-Штрассена**

1. Задам функцию *solovey()*, в которую буду передавать число *n*, проверяемое на простоту. По алгоритму, реализующему тест Соловэя-Штрассена и использующему число Якоби, осуществляется проверка простоты числа. В качестве результата возвращается строка, сообщающая простое/составное число или выдающая сообщение об ошибке, в случае ввода некорректных данных.

```
In [8]:
def solovey(n):
    if n < 5 or n%2 == 0:
        return "Введите корректные  данные"
    a = random.randint(2, n-3) # включительно границы
    r = (a**int((n-1)/2))%n
    if r != 1 and r != n-1:
        return "Число n составное"
    s = yakobi(a, n)
    if (r-s)%n != 0:
        return "Число n составное"
    else:
        return "Число n, вероятно, простое"
```

Figure 4.6: Реализация теста Соловэя-Штрассена

2. Вызову функцию для чисел 11 и 9. Алгоритм верно проверяет простоту чисел.

```
In [9]:
t = solovey(11)
t

Out[9]: 'Число n, вероятно, простое'

In [10]:
t = solovey(9)
t

Out[10]: 'Число n составное'
```

Figure 4.7: Результаты теста Соловэя-Штрассена

4. **Реализация теста Миллера-Рабина**

1. Задам функцию *miller()*, в которую буду передавать число *n*, проверяемое на простоту. По алгоритму, реализующему тест Миллера-Рабина, осуществляется проверка простоты числа. В качестве результата возвращается строка, сообщающая простое/составное число или выдающая сообщение об ошибке, в случае ввода некорректных данных.

```
In [11]:
def miller(n):
    if n < 5 or n%2 == 0:
        return "Введите корректные  данные"
    tmp = n-1
    s = 0
    r = tmp
    while r % 2 == 0:
        s += 1
        r /= 2
    a = random.randint(2, n-3)
    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"
```

Figure 4.8: Реализация теста Миллера-Рабина

2. Вызову функцию для чисел 9 и 11. Алгоритм верно проверяет простоту чисел.

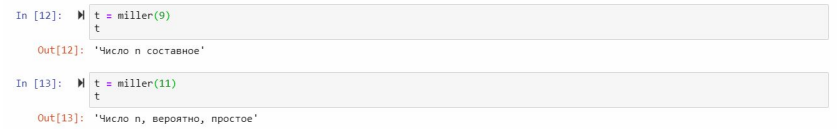

Figure 4.9: Результаты теста Миллера-Рабина

5 Выводы

В ходе данной лабораторной работы я реализовала три алгоритма проверки числа на простоту, а также алгоритм нахождения числа Якоби.

Список литературы

1. Простое число [Электронный ресурс]. Википедия, 2019. URL: https://ru.wikipedia.org/wiki/Простое_число#Алгоритмы_поиска_и_распознавания_простых_чисел.
2. Тест Ферма [Электронный ресурс]. Википедия, 2018. URL: https://ru.wikipedia.org/wiki/Тест_Ферма.
3. Теста Миллера-Рабина [Электронный ресурс]. Википедия, 2015. URL: https://ru.wikipedia.org/wiki/Тест_Миллера_—_Рабина.
4. Тест Соловея-Штрассена [Электронный ресурс]. Википедия, 2015. URL: https://ru.wikipedia.org/wiki/Тест_Соловея_—_Штрассена.
5. Символ Якоби [Электронный ресурс]. Википедия, 2017. URL: https://ru.wikipedia.org/wiki/Символ_Якоби.