РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

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

Дисциплина: *Математические основы защиты информации и информационной безопасности*

Студент: Банникова Екатерина Алексеевна
Группа: НФИмд-02-23

Москва 2023

Содержание

List of Figures

List of Tables

1 Цель работы

Целью данной лабораторной работы является ознакомление с вероятностными алгоритмами проверки чисел на простоту и программная реализация данных алгоритмов.

2 Задание

Реализовать все рассмотренные в инструкции к лабораторной работе алгоритмы проверки чисел на простоту программно.

3 Теоретическое введение

3.1 Тест Ферма

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

Figure 3.1: Основная информация по тесту Ферма

По Теореме Ферма, если `n` – простое число, тогда для любого а справедливо следующее равенство $a^{n-1}=1 \pmod{n}$. Отсюда мы можем вывести правило теста Ферма на проверку простоты числа: возьмем случайное $a \in \{1, ..., n-1\}$ и проверим будет ли соблюдаться равенство $a^{n-1}=1 \pmod{n}$. Если равенство не соблюдается, значит скорее всего `n` – составное.

Тем не менее, условие равенства может быть соблюдено, даже если `n` – не простое. Например, возьмем $n = 561 = 3 \times 11 \times 17$. Согласно Китайской теореме об остатках:

$Z_{561} = Z_3 \times Z_{11} \times Z_{17}$

Figure 3.2: Численный пример по тесту Ферма. Часть 1

, где каждое $a \in Z^*_{561}$ отвечает следующему:

$(x,y,z) \in Z^*_3 \times Z^*_{11} 11 \times Z^*_{17}$.

По теореме Ферма, $x^2=1$, $y^{10}=1$, *и* $z^{16}=1$. Поскольку 2, 10 и 16 все являются делителями 560, это значит, что $(x,y,z)^{560} = (1, 1, 1)$, другими словами $a^{560} = 1$ для любого $a \in Z^*_{561}$.

Figure 3.3: Численный пример по тесту Ферма. Часть 2

Не имеет значения какое a мы выберем, 561 всегда будет проходить тест Ферма несмотря на то, что оно составное, до тех пор, пока a является взаимно простым с n. Такие числа называются числами Кармайкла и оказывается, что их существует бесконечное множество.

Если a не взаимно простое с n, то оно тест Ферма не проходит, но в этом случае мы можем отказаться от тестов и продолжить искать делители n, вычисляя НОД(a,n).

3.2 Символ Якоби

Символ Якоби — теоретико-числовая функция двух аргументов, введённая К. Якоби в 1837 году. Является квадратичным характером в кольце вычетов.

Символ Якоби обобщает символ Лежандра на все нечётные числа, большие единицы. Символ Кронекера — Якоби, в свою очередь, обобщает символ Якоби на все целые числа, но в практических задачах символ Якоби играет гораздо более важную роль, чем символ Кронекера — 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.

Figure 3.4: Определение символа Якоби

Формальное описание [правитьǀ править код]

Входные данные: a — целое число, b — натуральное, нечётное, больше единицы.

Выходные данные: $\left(\frac{a}{b}\right)$ — символ Якоби

```
1 (проверка взаимной простоты). Если НОД (a, b)≠1, выход из алгоритма с ответом 0.

2 (инициализация). r:=1

3 (переход к положительным числам).
 Если a<0 то
  a:=-a
  Если b mod 4 = 3 то r:=-r
 Конец если

4 (избавление от чётности). t:=0
 Цикл ПОКА a — чётное
  t:=t+1
  a:=a/2
 Конец цикла
 Если t — нечётное, то
  Если b mod 8 = 3 или 5, то r:=-r.
 Конец если

5 (квадратичный закон взаимности). Если a mod 4 = b mod 4 = 3, то r:=-r.
  c:=a; a:=b mod c; b:=c.

6 (выход из алгоритма?). Если a≠0, то идти на шаг 4, иначе выйти из алгоритма с ответом r.
```

Figure 3.5: Алгоритм нахождения символа Якоби

3.3 Тест Соловэя-Штрассена

Роберт Соловей и Фолькер Штрассен разработали алгоритм вероятностного тестирования простоты числа, который использует символ Якоби. Определяет числа как составные или вероятно простые. Распознает числа Кармайкла как составные. Итак, для начала необходимо ввести нужные понятия.

Вход: $n > 2$, тестируемое нечётное натуральное число;
k, параметр, определяющий точность теста.
Выход: *составное*, означает, что n точно составное;
вероятно простое, означает, что n вероятно является простым.

for $i = 1, 2, \ldots, k$:
  a = случайное целое от 2 до $n-1$, включительно;
  если НОД(a, n) > 1, **тогда:**
    вывести, что n — составное, и **остановиться.**
  если $a^{(n-1)/2} \not\equiv \left(\frac{a}{n}\right) \pmod n$, **тогда:**
    вывести, что n — *составное*, и **остановиться.**

иначе вывести, что n — *простое с вероятностью* $1-2^{-k}$, и **остановиться.**

Figure 3.6: Алгоритм Соловэя-Штрассена

Вероятностные тесты применяются в системах основанных на проблеме факторизации, например RSA или схема Рабина. Однако на практике степень достоверности теста Соловея — Штрассена не является достаточной, вместо него используется тест Миллера — Рабина. Более того, используются объединенные алгоритмы, например пробное деление и тест Миллера — Рабина, при правильном выборе параметров можно получить результаты лучше, чем при применении каждого теста по отдельности.

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.

```
Ввод: n > 3, нечётное натуральное число, которое необходимо проверить на простоту;
      k — количество раундов.
Вывод: составное, означает, что n является составным числом;
       вероятно простое, означает, что n с высокой вероятностью является простым числом.
Представить n − 1 в виде 2^s·t, где t нечётно, можно сделать последовательным делением n - 1 на 2.
цикл А: повторить k раз:
   Выбрать случайное целое число a в отрезке [2, n − 2]
   x ← a^t mod n, вычисляется с помощью алгоритма возведения в степень по модулю
   если x = 1 или x = n − 1, то перейти на следующую итерацию цикла А
   цикл B: повторить s − 1 раз
      x ← x^2 mod n
      если x = 1, то вернуть составное
      если x = n − 1, то перейти на следующую итерацию цикла A
   вернуть составное
вернуть вероятно простое
```

Figure 3.7: Алгоритм Миллера-Рабина

4 Выполнение лабораторной работы

Примечание: комментарии по коду представлены на скриншотах к каждому из проделанных заданий.

В соответствии с заданием, были написаны программы реализации алгоритмов проверки чисел на простоту. Нами были рассмотрены следующие алгоритмы:

1. Тест Ферма;
2. Символ Якоби;
3. Тест Соловэя-Штрассена;
4. Тест Миллера-Рабина.

Программный код и результаты выполнения программ представлен ниже.

4.1 Тест Ферма

```
#ОБЩИЙ СЛУЧАЙ
#вообще говоря, число a задается случайно (в зависимости от алгоритма)
a=12
n=17#должно удовлетворять определенным условиям (в зависимости от алгоритма)
```

Figure 4.1: Входные данные для реализации алгоритмов проверки чисел на простоту

```
def test_ferma(a,n):
  '''
  Функция, реализующая тест Ферма по соотв. алгоритму
  '''
  r=(a**(n-1))%n
  if r==1:
    print('Число n=',n,', вероятно, ПРОСТОЕ')
  else:
    print('Число n=',n,', вероятно, СОСТАВНОЕ')
test_ferma(a,n)
```

Figure 4.2: Реализация алгоритма теста Ферма

Результаты выполнения программы представлены ниже.

```
Число n= 17 , вероятно, ПРОСТОЕ
```

Figure 4.3: Результат реализации алгоритма теста Ферма

4.2 Символ Якоби

```
def primefactors(n):
  '''
  Функция для расчета k,a1
  (предварительно разложить число на множители с использованием 2)
  '''
  list_2=[]
  m=n

  while m%2==0:
    list_2.append(2)
    m=m/2
  k=len(list_2)
  a1=int(n/(2**k))
  return k,a1
```

Figure 4.4: Реализация алгоритма вычисления символа Якоби 1 часть

```
def Jakobi_symbol(a,n):
  '''
  Функция, реализующая поиск символа Якоби по соотв. алгоритму
  '''
  g=1
  while True:
    if a==0:
      res=0
      break
    if a==1:
      res=g
      break
    else:
      k=primefactors(a)[0]
      a1=primefactors(a)[1]
      if k%2==0:
        s=1
      if k%2!=0:
        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:
      res=g*s
      break
```

Figure 4.5: Реализация алгоритма вычисления символа Якоби 2 часть

```
      if ((n-3)%4==0) and ((a1-3)%4==0):
        s=-s
      a=n%a1
      n=a1
      g=g*s
  return res

Jakobi_symbol(a,n)
```

Figure 4.6: Реализация алгоритма вычисления символа Якоби 3 часть

Результаты выполнения программы представлены ниже.

```
-1
```

Figure 4.7: Результат реализации алгоритма вычисления символа Якоби

4.3 Тест Соловэя-Штрассена

```
def solovey_strassen(a,n):
  '''
  Функция, реализующая тест Соловэя-Штрассена
  '''
  r=(a**((n-1)/2))%n
  if (r!=1) and (r!=n-1):
    print('Число n=',n,'СОСТАВНОЕ')
  s=Jakobi_symbol(a,n)
  if ((r-s)%n!=0):
    print('Число n=',n,'СОСТАВНОЕ')
  else:
    print('Число n=',n,', вероятно, ПРОСТОЕ')
solovey_strassen(a,n)
```

Figure 4.8: Реализация алгоритма теста Соловэя-Штрассена

Результаты выполнения программы представлены ниже.

```
Число n= 17 , вероятно, ПРОСТОЕ
```

Figure 4.9: Результат реализации алгоритма теста Соловэя-Штрассена

4.4 Тест Миллера-Рабина

```
def miller_rabin(a,n):
  '''
  Функция, реализующая тест Миллера-Рабина
  '''
  s=primefactors(n-1)[0]
  r=primefactors(n-1)[1]
  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=',n,'СОСТАВНОЕ'
      j=j+1
    if (y!=n-1):
      return'Число n=',n,'СОСТАВНОЕ'
  return 'Число n=',n,', вероятно, ПРОСТОЕ'
miller_rabin(a,n)
```

Figure 4.10: Реализация алгоритма теста Миллера-Рабина

Результаты выполнения программы представлены ниже.

```
('Число n=', 17, ', вероятно, ПРОСТОЕ')
```

Figure 4.11: Результат реализации алгоритма теста Миллера-Рабина

5 Выводы

Таким образом, была достигнута цель, поставленная в начале лабораторной работы: я ознакомилась с алгоритмами проверки чисел на простоту, – а так же реализовала данные алгоритмы на языке программирования Python 3.

Список литературы

1. Википедия. Тест Ферма [Электронный ресурс]. Википедия, свободная энциклопедия, 2023. URL: https://ru.wikipedia.org/wiki/Тест_Ферма.
2. MaxRokatansky. Тесты Ферма и Миллера-Рабина на простоту [Электронный ресурс]. Хабр, 2020. URL: https://habr.com/ru/company/otus/blog/486116/.
3. Википедия. Символ Якоби [Электронный ресурс]. Википедия, свободная энциклопедия, 2023. URL: https://ru.wikipedia.org/wiki/Символ_Якоби.
4. Википедия. Тест Соловея — Штрассена [Электронный ресурс]. Википедия, свободная энциклопедия, 2023. URL: https://en.wikipedia.org/wiki/Solovay–Strassen_primalit
5. Википедия. Тест Миллера — Рабина [Электронный ресурс]. Википедия, свободная энциклопедия, 2023. URL: https://en.wikipedia.org/wiki/Miller–Rabin_primality_test.