Отчет по лабораторной работе №5

Дисциплина: Математические основы защиты информации и информационной безопасности

Выполнила Дяченко Злата Константиновна, НПМмд-02-22

Содержание

List of Figures

1 Цель работы

Ознакомится и реализовать вероятностные алгоритмы проверки чисел на простоту.

2 Задание

Реализовать алгоритм, реализующий тест Ферма; алгоритм вычисления символа Якоби; алгоритм, реализующий тест Соловэя-Штрассена; алгоритм, реализующий тест Миллера-Рабина.

3 Теоретическое введение

Пусть a – целое число. Числа ±1, ±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. Проверка чисел на простоту является составной частью алгоритмов генерации простых чисел, применяемых в криптографии с открытым ключом. 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. Вероятностные алгоритмы в общем случае не менее эффективны, чем детерминированные (если используемый генератор случайных чисел всегда дает набор одних и тех же чисел, зависящих от входных данных, то вероятностный алгоритм становится детерминированным).

Тест Ферма основан на малой теореме Ферма: для простого числа p и произвольного числа a, $1 \leq a \leq$ p − 1, выполняется сравнение

$$a^{p-1} \equiv 1(modp)$$

Следовательно, если для нечетного n существует такое целое a, что $1 \leq a <$ n, НОД(a, n) = 1 и $a^{n-1} \neq 1(modn)$, то число n составное. Отсюда получаем следующий вероятностный алгоритм проверки числа на простоту.

1. Алгоритм, реализующий 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 $r \leftarrow a^{n-1}(modn)$.
3. При $r = 1$ результат: «Число n, вероятно, простое». В противном случае результат: «Число n составное».

Тест Соловэя-Штрассена. Основан на критерии Эйлера: нечетное число □является простым тогда и только тогда, когда для любого целого числа a, $1 \leq a \leq$ n $- 1$, взаимно простого с n, выполняется сравнение:

$$a^{\frac{n-1}{2}} \equiv (\frac{a}{n})(modn)$$

где $(\frac{a}{n})$ – символ Якоби. Пусть $m, n \in Z$, где $n = p_1 p_2 ... p_r$ и числа $p_i \neq 2$ простые (не обязательно различные). Символ Якоби $(\frac{m}{n})$ определяется равенством

$$(\frac{m}{n}) = (\frac{m}{p_1})(\frac{m}{p_2})...(\frac{m}{p_r})$$

2. Алгоритм вычисления символа Якоби.

   Вход. Нечетное целое число $n \geq 3$, целое число a, $0 \leq a <$ n.

   Выход. Символ Якоби $(\frac{a}{n})$.

   1. Положить $g \leftarrow 1$.
   2. При $a = 0$ результат: 0.
   3. При $a = 1$ результат: g.
   4. Представить a в виде $a = 2^k a_1$, где число a_1 нечетное.
   5. При четном k положить $s \leftarrow 1$, при нечетном k положить $s \leftarrow 1$, если $n \equiv \pm 1(mod8)$; положить $s \leftarrow -1$, если $n \equiv \pm 3(mod8)$.
   6. При $a_1 = 1$ результат: g*s.
   7. Если $n \equiv 3(mod4)$ и $a_1 \equiv 3(mod4)$, то $s \leftarrow -s$.
   8. Положить $a \leftarrow n(moda_1)$, $n \leftarrow a_1$, $g \leftarrow g * s$ и вернуться на шаг 2.

3. Алгоритм, реализующий 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 ≤ a < n − 2.
   2. Вычислить $r \leftarrow a^{\frac{n-1}{2}}(modn)$.
   3. При $r \neq 1$ и $r \neq n-1$ результат: «Число n составное».
   4. Вычислить символ Якоби $s \leftarrow (\frac{a}{n})$
   5. При $r \equiv s(modn)$ результат: «Число n составное». В противном случае результат: «Число n, вероятно, простое».

На сегодняшний день для проверки чисел на простоту чаще всего используется тест Миллера-Рабина, основанный на следующем наблюдении. Пусть число n нечетное и $n-1 = 2^s r$,где r – нечетное. Если n простое, то для любого $a \geq 2$, взаимно простого с n, выполняется условие $a_{p-1} \equiv 1(modp)$.

4. Алгоритм, реализующий 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 число r нечетное.
   2. Выбрать случайное целое число a, 2 ≤ a < n − 2.
   3. Вычислить $y \leftarrow a^r(modn)$.
   4. При $y \neq 1$ и $y \neq n-1$ выполнять следующие действия:
      1. Положить $j \leftarrow 1$.
      2. Если $j \leq s-1$ и $y \neq n-1$:
         1. Положить $y \leftarrow y^2(modn)$.
         2. При $y = 1$ результат: «Число n составное».
         3. Положить $j \leftarrow j+1$
      3. При $y \neq n-1$ результат: «Число n составное».
   5. Результат: «Число n, вероятно, простое».

4 Выполнение лабораторной работы

4.1 Шаг 1

Ознакомилась с предоставленными теоретическими данными. Для выполнения задания решила использовать язык Python. Написала функцию, выполняющую проверку числа на простоту с использованием теста Ферма. Код функции и результат ее использования представлен на Рисунке 1 (рис. - fig. 4.1). Функция принимает на вход число n. При условии, что $n \geq 5$ реализуется алгоритм, представленный в теоретическом введении и функция возвращает отчет. Если условие не выполняется, будет выведено соответствующее сообщение.

Figure 4.1: Реализация алгоритма, реализующего тест Фирма

4.2 Шаг 2

На Рисунке 2 (рис. - fig. 4.2) представлен код функции, реализующий алгоритм вычисления символа Якоби. Пример выполнения показан на Рисунке 3 (рис. - fig. 4.3).

```
In [90]: def yakobi(n, a):
             if (n>=3 and n%2!=0 and 0<=a<n):
                 g=1
                 while True:
                     if (a==0):
                         return 0
                     if (a==1):
                         return g
                     if (a%2!=0):
                         k=0
                         a1=a
                     else:
                         k=0
                         a1=a
                         while (a1%2==0):
                             a1=a1/2
                             k=k+1
                     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):
                         return g*s
                     if (n%4==3 and a1%4==3):
                         s=-s
                     a=n%a1
                     n=a1
                     g=g*s
             else:
                 return ("Введите нечетное число больше или равное 3 и проверьте, что a больше или равно 0 и меньше введенного числа")
```

Figure 4.2: Реализация алгоритма вычисления символа Якоби

```
In [91]: yakobi(21, 11)
Out[91]: -1

In [92]: yakobi(21, 7)
Out[92]: 0

In [75]: yakobi(21, 4)
Out[75]: 1
```

Figure 4.3: Пример работы алгоритма вычисления символа Якоби

4.3 Шаг 3

На Рисунке 3 (рис. - fig. 4.3) представлен код функции, реализующий тест Соловэя-Штрассена.

```
In [128]: def solshtr(n):
              if (n>=5 and n%2!=0):
                  a=random.randint(2, n-3)
                  r=math.pow(a, (n-1)/2)%n
                  if (r!=1 and r!=n-1):
                      return ("Число ", n, "составное")
                  else:
                      ss=yakobi(n, a)
                      if (r%n==ss):
                          return ("Число ", n, "составное")
                      else:
                          return ("Число ", n, ", вероятно, простое")
              else:
                  return ("Введите нечетное число больше или равное 5")

In [131]: solshtr(13)

Out[131]: ('Число ', 13, ', вероятно, простое')
```

Figure 4.4: Реализация теста Соловэя-Штрассена

4.4 Шаг 4

На Рисунке 4 (рис. - fig. 4.4) представлен код функции, реализующий тест Миллера-Рабина, и пример выполнения.

```
In [125]: def milrab(n):
              if (n>=5 and n%2!=0):
                  s=0
                  r=n-1
                  while (r%2==0):
                      r=r/2
                      s=s+1
                  a=random.randint(2, n-3)
                  y=math.pow(a, r)%n
                  if (y!=1 and y!=n-1):
                      j=1
                      if (j<=s-1 and y!=n-1):
                          y=math.pow(y,2)%n
                          if (y==1):
                              return ("Число ", n, "составное")
                          j=j+1
                      if (y!=n-1):
                          return ("Число ", n, "составное")
                  return ("Число ", n, ", вероятно, простое")
              else:
                  return ("Введите нечетное число больше или равное 5")

In [126]: milrab(7)

Out[126]: ('Число ', 7, ', вероятно, простое')

In [127]: milrab(9)

Out[127]: ('Число ', 9, 'составное')
```

Figure 4.5: Реализация теста Миллера-Рабина

5 Выводы

Я ознакомилась с алгоритмами проверки чисел на простоту и реализовала их. Результаты работы находятся в репозитории на GitHub, а также есть скринкаст выполнения лабораторной работы.