МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Проверка чисел на простоту

ОТЧЁТ

ПО ДИСЦИПЛИНЕ

«ТЕОРЕТИКО-ЧИСЛОВЫЕ МЕТОДЫ В КРИПТОГАФИИ»

студента 5 курса 531 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Серебрякова Алексея Владимировича

Преподаватель

профессор, д.ф.-м.н. ___________ В. А. Молчанов

подпись, дата

Саратов 2023

Оглавление

Введение

В данной лабораторной работе поставлена задача рассмотрения теста Ферма, Соловея-Штрассена, Миллера-Рабина, написание алгоритмов для изученных тем.

Цель работы и порядок её выполнения

Цель работы – изучение основных методов проверки простоты чисел и их программная реализация.

Порядок выполнения работы:

1. Рассмотреть тест Ферма проверки чисел на простоту и привести его программную реализацию.
2. Рассмотреть тест Соловея-Штрассена проверки чисел на простоту и привести его программную реализацию.
3. Рассмотреть тест Миллера-Рабина и привести его программную реализацию.

1 Теоретическая часть

1.1 Тест Ферма

Малая теорема Ферма. Если p – простое число, то для любого $a \in Z_p^*$ выполняется свойство $F_p(a) = a^{p-1} \equiv 1\ (mod\ p)$.

Алгоритм 1 – тест простоты на основе малой теоремы Ферма:

Вход. Нечетное число $n > 5$.

Выход. “Число n вероятно, простое” или “Число n составное”.

Шаг 1. Выбрать случайно $a \in \{1,2,\dots,n-1\}$ и вычислить $d =$ НОД(a,n). Если $d > 1$, то ответ “Число n составное”.

Шаг 2. Если $d = 1$, то проверить условие $F_n(a) = \left(a^{n-1} \equiv 1(modn)\right)$. Если оно не выполнено, то ответ “Число n составное”. В противном случае ответ “Число n, вероятно, простое”.

Трудоемкость алгоритма: $O(log^3 n)$.

Определение. Число n называется псевдопростым по основанию $a \in Z_n^*$, если выполняется $F_n(a)$. Здесь $F_n^+ = \{a \in Z_n^* \mid F_n(a)\}$.

Лемма 1. Для нечетного числа n справедливы утверждения:

F_n^+ – подгруппа Z_n^*;

если $F_n^+ \neq Z_n^*$, то по теореме Лагранжа $|Z_n^*| = |F_n^+| \cdot |Z_n^*/F_n^+| \geq 2 \cdot |Z_n^*|, |F_n^+| \leq \frac{|Z_n^*|}{2}$.

Вероятность успеха – вероятность получить “Число n составное” для составного числа n равна $P_0 = 1 - \frac{|F_n^+|}{n-1}$. Возможны три случая:

1. число n простое и тест всегда дает ответ “Число n, вероятно, простое”;
2. число n составное и $F_n^+ \neq Z_n^*$, тогда тест дает ответ “Число n составное” с вероятностью успеха $P_0 = 1 - \frac{|F_n^*|}{n-1} \geq 1 - \frac{|F_n^+|}{|Z_n^*|} \geq 1 - \frac{1}{2} = \frac{1}{2}$;

3. число n составное и $F_n^+ = Z_n^*$, тогда тест дает ответ “Число n составное” с вероятностью успеха $P_0 = 1 - \frac{\varphi(n)}{n-1}$.

В случае 2 при k повторах теста вероятность успеха $P_0^{(k)} = 1 - (1 - P_0)^k \geq 1 - \frac{1}{2^k} \approx 1$.

1.2 Тест Соловея-Штрассена

Критерий Эйлера. Нечетное число n является простым тогда и только тогда, когда для любого $a \in Z_n^*$ выполняется свойство:

$$E_n(a) = \left(a^{\frac{n-1}{2}} \equiv \left(\frac{a}{n}\right) \ (mod\ n)\right).$$

Причем n – простое число $\leftrightarrow E_n^* = Z_n^*$ где $E_n^* = \{a \in Z_n^* \mid E_n(a)\}$.

Алгоритм 2 – тест простоты Соловея-Штрассена

Вход. Нечетное число $n > 5$.

Выход. “Число n, вероятно, простое” или “Число n составное”.

Шаг 1. Выбрать $a \in \{1,2,\dots,n-1\}$ и вычислить $d = \text{НОД}(a,n)$. Если $d > 1$, то ответ “Число n составное”.

Шаг 2. Если $d = 1$, то проверить условие $E_n(a)$. Если оно не выполнено, то ответ “Число n составное”. В противном случае ответ “Число n, вероятно, простое”.

Трудоемкость алгоритма: $O(log^3 n)$.

Определение. Число n называется эйлеровым псевдопростым по основанию $a \in Z_n^*$, если выполняется $E_n(a)$.

Лемма 1. Для нечетного числа n справедливы утверждения:

- E_n^+ – подгруппа Z_n^*;
- Если n – составное число, то $|E_n^+| \leq \frac{|F_n^*|}{2}$.

Вероятность успеха Алгоритма 2 для составного числа n равна $P_0 = 1 - \frac{|E_n^+|}{n-1} \geq \frac{1}{2}$. Возможны два случая:

1. число n простое и тест всегда дает ответ “Число n, вероятно, простое”;
2. число n составное и тест дает ответ “Число n составное” с вероятностью успеха $P_0 \geq \frac{1}{2}$.

В случае 2 при k повторах теста вероятность успеха $P_0^{(k)} = 1 - (1 - P_0)^k \geq 1 - \frac{1}{2^k} \approx 1$.

1.3 Тест Миллера-Рабина

Теорема (Критерий Миллера). Пусть n – нечетное число и $n - 1 = 2^s t$ для нечетного t. Тогда n является простым в том и только том случае, если для любого $a \in Z_n^*$ выполняется свойство

$$M_n(a) = \left(a^t \equiv 1(modn) \vee (\exists 0 \leq k < s)\left(a^{2^k t} \equiv -1(modn)\right)\right).$$

Причем n – простое число $\leftrightarrow M_n^+ = Z_n^*$ где $M_n^+ = \{a \in Z_n^* \mid M_n(a)\}$.

Необходимость: для простого n выполняется:

$$a^{(n-1)} \equiv 1(modn), \qquad a^{(n-1)} - 1 \equiv 0(modn)$$

$$a^{2^s t} - 1 \equiv \left((a^t)^{(2^{s-1})}\right)^2 - 1 \equiv (a^t - 1)(a^t + 1) \dots \left((a^t)^{(2^{s-1})} + 1\right)$$
$$\equiv 0 \ (mod\ n\ )$$

Алгоритм 3 – тест простоты Миллера-Рабина

Вход. Нечетное число $n > 5$.

Выход. “Число n, вероятно, простое” или “Число n составное”.

Шаг 1. Выбрать $a \in \{1,2, \dots, n-1\}$ и вычислить $d = \text{НОД}(a, n)$. Если $d > 1$, то ответ “Число n составное”.

Шаг 2. Если $d = 1$, то вычислить $r_k = a^{2^k t}$ для значений $k \in \{0,1,2, \dots, s-1\}$. Если $r_0 \equiv 1\ (mod\ n)$ или $r_k \equiv -1\ (mod\ n)$ для некоторого

$0 \leq k < s$, то ответ “Число n, вероятно, простое”. В противном случае ответ “Число n составное”.

Трудоемкость алгоритма: $O(log^3 n)$.

Определение. Число n, псевдопростое по основанию $a \in Z_n^*$, называется сильно псевдопростым по этому основанию $a \in Z_n^*$, если выполняется $M_n(a)$, т.е. выполняется одно из условий:

$a^t \equiv 1 \ (mod\ n)$;

$a^{2^k t} \equiv -1 \ (mod\ n)$ для некоторого $0 \leq k < s$.

Для составного числа n выполняется $|M_n^+| \leq \frac{|Z_n^*|}{4}$ и, значит, вероятность успеха Алгоритма 3 для составного числа n равна $P_0 = 1 - \frac{|M_n^+|}{n-1} \geq \frac{3}{4}$. При k повторах теста вероятность успеха:

$$P_0^{(k)} = 1 - (1 - P_0)^k \geq 1 - \frac{1}{4^k} \approx 1.$$

2 Псевдокоды программ

2.1 Псевдокод алгоритма проверки на простоту по тесту Ферма

```
Процедура проверки_числа(n):
    Если n ≤ 5 || n % 2 == 0:
        Вернуть "Число n составное"
    Установить a как случайное значение в {1,2,…,n − 1}
    Вычислить d = НОД(a,n)
    Если d > 1:
        Вернуть "Число n составное"
    Если d = 1:
        Проверить условие F_n(a) − a^(n−1) % n == 1
        Если условие F_n(a) не выполнено:
            Вернуть "Число n составное"
        Иначе:
            Вернуть "Число n, вероятно, простое"
```

2.2 Псевдокод проверки на простоту по тесту Соловея-Штрассена

```
Процедура проверки_числа(n):
    допустим a = случайное число из {1,2,…,n − 1}
    допустим d = НОД(a,n)
    Если d > 1:
        Вернуть "Число n составное"
    Если d = 1:
        Если условие E_n(a) не выполнено:
            Вернуть "Число n составное"
        Иначе:
            Вернуть "Число n, вероятно, простое"
```

2.3 Псевдокод проверки на простоту по тесту Миллера-Рабина

```
Процедура проверки_числа(n):
    Убедиться, что n > 5 и нечетное.
    Выбрать a из {1,2,…,n − 1}
    Вычислить d = НОД(a,n)
    Если d > 1:
        Вернуть "Число n составное"
    Если d = 1:
        Вычислить t и s, такие что n − 1 = 2^s ∗ t и t нечетное.
    Для каждого k в {0,1,2,…,s − 1}:
        Вычислить r_k = a^(2^k t) % n
    Если r_0 ≡ 1 (mod n) или r_k ≡ 1 (mod n)
        Вернуть "Число n, вероятно, простое"
    Вернуть "Число n составное"
```

3 Тестирование программы

На рисунках 1-3 представлены результаты работы программы при проверке некоторых чисел на простоту. На рисунке 4 представлен случай, когда разные тесты выдают разный результат при проверке одного и того же числа. Это происходит потому, что тесты являются вероятностными и для достижения единого ответа потребуется большее количество проверок.

```
Выберите действие
0 - Выход из программы
1 - Проверка тестом Ферма
2 - Проверка тестом Соловея-Штрассена
3 - Проверка тестом Миллера-Рабина
1

Введите число для проверки на простоту
32131

Введите число проверок
100

Результат проверки тестом Ферма: Составное
```

Рисунок 1 - Проверка на простоту тестом Ферма

```
Результат проверки тестом Миллера-Рабина: Вероятно простое

Выберите действие
0 - Выход из программы
1 - Проверка тестом Ферма
2 - Проверка тестом Соловея-Штрассена
3 - Проверка тестом Миллера-Рабина
3

Введите число для проверки на простоту
321311

Результат проверки тестом Миллера-Рабина: Вероятно простое
```

Рисунок 2 - Проверка на простоту тестом Соловея-Штрассена

```
Выберите действие
0 - Выход из программы
1 - Проверка тестом Ферма
2 - Проверка тестом Соловея-Штрассена
3 - Проверка тестом Миллера-Рабина
3

Введите число для проверки на простоту
311

Результат проверки тестом Миллера-Рабина: Вероятно простое
```

Рисунок 3 - Проверка на простоту тестом Миллера-Рабина

```
Выберите действие
0 - Выход из программы
1 - Проверка тестом Ферма
2 - Проверка тестом Соловея-Штрассена
3 - Проверка тестом Миллера-Рабина
3

Введите число для проверки на простоту
9876541

Результат проверки тестом Миллера-Рабина: Вероятно простое

Выберите действие
0 - Выход из программы
1 - Проверка тестом Ферма
2 - Проверка тестом Соловея-Штрассена
3 - Проверка тестом Миллера-Рабина
1

Введите число для проверки на простоту
9876541

Введите число проверок
100

Результат проверки тестом Ферма: Составное

Выберите действие
0 - Выход из программы
1 - Проверка тестом Ферма
2 - Проверка тестом Соловея-Штрассена
3 - Проверка тестом Миллера-Рабина
2

Введите число для проверки на простоту
9876541

Введите число проверок
100

Результат проверки тестом Соловея-Штрассена: Составное
```

Рисунок 4 - Проверка одного числа на простоту тестами Ферма, Соловея-Штрассена и Миллера-Рабина

ПРИЛОЖЕНИЕ А

Листинг программы lab3.rb

```ruby
require "time"

def timer(f)
  lambda do |*args, **kwargs|
    start = Process.clock_gettime(Process::CLOCK_MONOTONIC)
    ans = f.call(*args, **kwargs)
    finish = Process.clock_gettime(Process::CLOCK_MONOTONIC)
    puts "Time: #{(finish - start) * 1000}ms"
    ans
  end
end

def hlp_ferma(a, n)
  ans = 1
  (1..n-1).each do |i|
    ans = (ans * a) % n
  end
  ans = (ans - 1) % n
  ans
end

def fermat(n, k)
  k.times do
    a = rand(n - 2) + 2
    return 0 if a.gcd(n) > 1

    tmp = hlp_ferma(a, n)
    return 0 if tmp != 0
  end
  1
end

def hlp(base, e, n)
  x, y = 1, base
  while e != 0
    if e % 2 != 0
      x = (x * y) % n
    end
    y = (y * y) % n
    e = e / 2
  end
  x % n
end

def jacobi(a, n)
  return 0 if a.gcd(n) != 1

  t = 1
  a %= n
  while a != 0
    while a % 2 == 0
      a /= 2
      if n % 8 == 3 || n % 8 == 5
        t = -t
```

```
      end
    end
    n, a = a, n
    if a % 4 == 3 && n % 4 == 3
      t = -t
    end
    a %= n
  end
  t
end

def solovay_strassen(n, k)
  return false if n < 2
  return false if n != 2 && n % 2 == 0

  k.times do
    a = rand(n - 1) + 1
    jacobian = (n + jacobi(a, n)) % n
    mod = hlp(a, (n - 1) / 2, n)

    if jacobian == 0 || mod != jacobian
      return false
    end
  end
  true
end

def miller_rabin(n)
  return 1 if n == 2 || n == 3
  return 0 if n.even? && n != 2

  s, t, x = 0, n - 1, 0
  r1, r2 = 2, n - 2
  a, r = Math.log2(n).to_i, Math.log2(n).to_i

  while t != 0 && t % 2 == 0
    s += 1
    t /= 2
  end

  r.times do
    a = r1 + rand(r2 - r1)
    x, j = 1, 1

    while j <= t
      x = (x * a) % n
      j += 1
    end

    next if x == 1 || x == n - 1

    (s - 1).times do
      x = (x * x) % n
      return 0 if x == 1
      break if x == n - 1
    end
```

```
    return 0 if x != n - 1
  end
  1
end

def main

  opt = 1

  while opt != 0
    puts "\nВыберите действие"
    puts "0 - Выход из программы"
    puts "1 - Проверка тестом Ферма"
    puts "2 - Проверка тестом Соловея-Штрассена"
    puts "3 - Проверка тестом Миллера-Рабина"
    opt = gets.strip.to_i

    break if opt == 0

    puts "\nВведите число для проверки на простоту"
    n = gets.strip.to_i

    if opt == 1 || opt == 2
      puts "\nВведите число проверок"
      k = gets.strip.to_i
    end

    case opt
    when 1
      r = fermat(n, k)
      if r == 0
        puts "\nРезультат проверки тестом Ферма: Составное"
      else
        puts "\nРезультат проверки тестом Ферма: Вероятно
простое"
      end
    when 2
      r = solovay_strassen(n, k)
      if !r
        puts "\nРезультат проверки тестом Соловея-Штрассена:
Составное"
      else
        puts "\nРезультат проверки тестом Соловея-Штрассена:
Вероятно простое"
      end
    when 3
      r = miller_rabin(n)
      if !r
        puts "\nРезультат проверки тестом Миллера-Рабина:
Составное"
      else
        puts "\nРезультат проверки тестом Миллера-Рабина:
Вероятно простое"
      end
    end
  end
end
```

```
if __FILE__ == $PROGRAM_NAME
  main
end
```