

[Введение 2](#_Toc149578503)

[Цель работы и порядок её выполнения 3](#_Toc149578504)

[1 Теоретическая часть 4](#_Toc149578505)

[1.1 Разложения чисел в цепную дробь 4](#_Toc149578506)

[1.2 Приложение цепных дробей 5](#_Toc149578507)

[1.3 Вычисление символов Лежандра и Якоби 7](#_Toc149578508)

[1.4 Извлечение квадратного корня в кольце вычетов 9](#_Toc149578509)

[2 Псевдокоды программ 10](#_Toc149578510)

[2.1 Псевдокод алгоритма разложения чисел в цепную дробь 10](#_Toc149578511)

[2.2 Псевдокод приложения цепных дробей 11](#_Toc149578512)

[2.3 Псевдокоды вычислений символов Лежандра и Якоби 12](#_Toc149578513)

[2.4 Псевдокод извлечения квадратного корня в кольце вычетов 13](#_Toc149578514)

[3 Тестирование программы 14](#_Toc149578515)

[ПРИЛОЖЕНИЕ А 17](#_Toc149578516)

Введение

В данной лабораторной работе поставлена задача рассмотрения алгоритмов разложения чисел в цепную дробь, приложений цепных дробей, вычисления символов Лежандра и Якоби, алгоритмы извлечения квадратного корня в кольце вычетов, написание алгоритмов для изученных тем.

Цель работы и порядок её выполнения

Цель работы – изучение основных свойств цепных дробей и квадратных сравнений.

Задачи работы:

* Разобрать алгоритм разложения чисел в цепную дробь и привести их программную
* Рассмотреть алгоритмы приложений цепных дробей и привести их программную реализацию.
* Разобрать алгоритмы вычисления символов Лежандра и Якоби и привести их программную реализацию.
* Рассмотреть алгоритмы извлечения квадратного корня в кольце вычетов.

1 Теоретическая часть

1.1 Разложения чисел в цепную дробь

С помощью алгоритма Евклида любое рациональное число можно представить в виде специального выражения, которое называется цепной дробью и которое играет важную роль в алгебре, теории чисел, криптографии и во многих других областях математики. Рассмотрим рациональное число , представленное в виде несократимой дроби . Так как , то результат вычисления этого наибольшего общего делителя по алгоритму Евклида имеет вид:

где .

Рациональное число r можно представить следующим образом:

где – целое число и – целые положительные числа.

Определение. Выражение вида

принято называть цепной (или непрерывной) дробью с неполными частными и обозначать символом .

Алгоритм разложения чисел в цепную дробь:

Вход. Целые сила .

Выход. Коэффициенты разложения в цепную дробь.

Шаг 1.

Шаг 2. Пока .

Шаг 2.1. Добавить целую часть от деления на в список .

Шаг 2.2. Обновить значение .

Шаг 2.3. Поменять местами значения и .

Шаг 3. К списку добавить целую часть от деления на .

Шаг 4. Вернуть список .

Сложность алгоритма: .

1.2 Приложение цепных дробей

* Решение линейных диофантовых уравнений ;
* Вычисление обратных элементов в кольце вычетов ;
* Решение линейных сравнений .

Приведенные выше свойства числителей и знаменателей подходящих дробей цепной дроби дают эффективный способ решения диофантовых уравнений.

Определение. Диофантовыми уравнениями называются алгебраические уравнения с целочисленными коэффициентами, решение которых отыскивается в целых числах.

Например, диофантовым уравнением является уравнение вида с целыми неотрицательными коэффициентами . Если коэффициенты удовлетворяют условию и – предпоследняя подходящая дробь представления числа в виде цепной дроби, то из равенств следует, что , т.е. значения являются целочисленными решениями уравнения . Легко видеть, что все целые решения исходного диофантова уравнения находятся по формулам:

где – произвольное целое число.

Нетрудно убедиться, что все решения диофантова уравнения с взаимно простыми коэффициентами находятся по формулам:

где – произвольное целое число.

Алгоритм решения диофантова уравнения:

Вход. Целые числа .

Выход. Решение уравнения в виде пары , удовлетворяющие уравнению .

Шаг 1. При помощи расширенного алгоритма Евклида находим .

Шаг 2. Проверяем делится ли на , если нет, то решения не существует, иначе переходим на шаг 3.

Шаг 3. Делим на .

Шаг 4. Создаем два массива для линейных коэффициентов в цепной дроби и заполняем их через цепную дробь числа .

Шаг 5. Представляем значения из и в качестве коэффициентов для и , т.к. и могут быть выражены через эти коэффициенты.

Шаг 6. Возвращаем получившиеся и как решение диофантова уравнения.

Сложность алгоритма: .

Алгоритм вычисления обратного элемента в кольце:

Вход. Целые числа .

Выход. Обратный элемент или ничего, если он не существует.

Шаг 1. При помощи расширенного алгоритма Евклида находим , а также коэффициенты , , такие что .

Шаг 2. Проверяем , если нет, то обратный элемент не существует, иначе переходим на шаг 3.

Шаг 3. Вычисляем обратный элемент по модулю .

Шаг 4. Возвращаем в качестве результата.

Сложность алгоритма: , где – максимальная битовая длина наибольшего из входных чисел.

Алгоритм решения линейного сравнения:

Вход. Целые числа .

Выход. Решение уравнения в виде пары , удовлетворяющие уравнению .

Шаг 1. Находим .

Шаг 2. Проверяем делится ли на , если нет, то решения не существует, иначе переходим на шаг 3.

Шаг 3. Делим на .

Шаг 4. Создаем два массива для линейных коэффициентов в цепной дроби и заполняем их через цепную дробь числа .

Шаг 5. Представляем значения из и в качестве коэффициентов для и , т.к. и могут быть выражены через эти коэффициенты.

Шаг 6. Возвращаем получившиеся и как решение диофантова уравнения.

Сложность алгоритма: , где – битовая длина наибольшего из входных чисел.

1.3 Вычисление символов Лежандра и Якоби

Определение. Для нечетного простого числа символом Лежандра числа называется выражение

Свойства символа Лежандра:

* для любого .
* Критерий Эйлера: для .

Алгоритм вычисления символа Лежандра:

Вход. Целое число и нечетное простое число .

Выход. Значение символа Лежандра.

Шаг 1. Если по свойству 4) выделяем множитель .

Шаг 2. Заменяем на остаток от деления на .

Шаг 3. Представляем и вычисляем по свойству 4) , при этом опускаем множители с четными степенями и вместо множителей с нечетными степенями оставляем .

Шаг 4. Если , то вычисляем по свойству 6) .

Шаг 5. К остальным символам применяется квадратичный закон взаимности Гаусса.

Шаг 6. При необходимости возвращаемся к шагу 2.

Сложность алгоритма: .

Определение. Пусть дано натуральное число .Символом Якоби числа называется выражение

Символ Якоби для простого числа совпадает с символом Лежандра и удовлетворяет почти всем свойствам символа Лежандра (хотя в общем случае символ Якоби не связан с квадратичными вычетами).

Символ Якоби позволяет упростить вычисление символа Лежандра (без разложения числа на множители).

Алгоритм вычисления символа Якоби для простого числа совпадает с алгоритмом вычисления символа Лежандра.

Простой алгоритм вычисления символа Якоби:

Вход. Целое число и нечетное простое число .

Выход. Значение символа Якоби.

Шаг 1. Заменяем на , что и .

Шаг 2. по свойству 4) выделяем множитель .

Шаг 3. Если и, если .

Шаг 4. К символу применяется квадратичный закон взаимодействия Гаусса.

Шаг 5. При необходимости возвращаемся к шагу 1.

Сложность алгоритма: .

1.4 Извлечение квадратного корня в кольце вычетов

1. Если , то и , , удовлетворяет .
2. Если , то и или , или .

В первом случае: , .

Во втором случае рассматриваем , так как в силу .

Тогда , и для получаем .

В общем случае применяются специальные полиномиальные вероятностные алгоритмы.

Алгоритм извлечения квадратного корня в кольце вычетов:

Вход. Нечетное простое число .

Выход. – решение сравнения .

Шаг 1. Случайным образом выбирается , такое, что .

Шаг 2. Положить .

Шаг 3. Вычислить – остаток от деления на . Тогда – искомое решение.

Шаг 4. Вернуть .

Сложность алгоритма: .

2 Псевдокоды программ

2.1 Псевдокод алгоритма разложения чисел в цепную дробь

Алгоритм разложения чисел в цепную дробь.

Процедура Разложить\_на\_цепную\_дробь(a,b):

a1 = a

a2 = b

q = []

Пока a1 mod a2 != 0:

частное = a1 div a2

Добавить к q частное

a1 = a1 mod a2

Поменять значениями a1 и a2

Конец пока

целая\_часть = a1 div a2

Добавить к q целую\_часть

Вернуть q

Конец процедуры

2.2 Псевдокод приложения цепных дробей

Решение диофантова уравнения

Процедура Решить\_диофантово\_Уравнение(a,b,c):

НОД = Евклид(a, b) # Находим НОД(a, b) с помощью алгоритма Евклида

Если c mod НОД != 0

Вернуть «Решений не существует»

Конец Если

a = a div НОД

b = b div НОД

c = c div НОД

p = []; q = []

Пока b != 0:

частное = a div b

остаток = a mod b

Добавить к p частное

Добавить к q остаток

Конец Пока

a = b

b = остаток

x = 0

y = 1

n = Длина(p)

Для всех i от n-1 до 0 с шагом -1:

новый\_x = y

новый\_y = x – p[i] \* y

Конец Для

Вернуть (x \* c, y \* c)

Конец процедуры

Вычисление обратного элемента в кольце.

Процедура Найти\_Обратный\_Элемент(a, m):

НОД, x, y = Расширенный\_Евклид(a, m) # Вычисляем НОД(a, m) и коэффициенты x, y

Если НОД != 1:

Вернуть «Обратный элемент не существует»

Конец Если

x = (x mod m + m) mod m

Вернуть x

Конец процедуры

2.3 Псевдокоды вычислений символов Лежандра и Якоби

Псевдокод вычисления символа Лежандра.

Процедура Вычислить\_Символ\_Лежандра(a, p):

Если a < 0:

a = a mod p

Конец Если

Если a == 0:

Вернуть 0

Конец Если

Символ\_Лежандра = 1

Пока a != 1:

Если a mod 2 == 0:

a = a div 2

Конец Если

Если p mod 8 == 3 ИЛИ p mod 8 == 5:

Символ\_Лежандра = - Символ\_Лежандра

Иначе:

Поменять значениями a и p

Конец Если

Если a mod 4 == 3 B p mod 4 == 3:

Символ\_Лежандра = - Символ\_Лежандра

Конец Если

Конец Пока

Вернуть Символ\_Лежандра

Конец Процедуры

Псевдокод вычисления символа Якоби.

Процедура Вычислить\_Символ\_Якоби(a, p):

Пока a != 0:

a = a mod p

Если a == 0:

Вернуть 0

Конец Если

Если a < 0:

a = a \* (-1)

Конец Если

Если a == 1:

Вернуть 1

Конец Если

Если a mod 2 == 0:

b = a

t = 0

Конец Если

Пока b mod 2 == 0:

b = b div 2

t = t + 1

Конец Пока

Если t mod 2 != 0 И (p mod 8 == 3 ИЛИ p mod == 5):

Вернуть -1

Конец Если

a = b

Если a == 2:

Если p mod 8 == 3 ИЛИ p mod 8 == 5:

Вернуть -1

Конец Если

Вернуть 1

Конец Если

Поменять значениями a и p

Если a mod 4 == 3 И p mod 4 == 3:

Вернуть (-1) \* Вычислить\_Символ\_Якоби(-1, p)

Конец Если

Конец Пока

Вернуть 1

Конец процедуры

2.4 Псевдокод извлечения квадратного корня в кольце вычетов

Извлечение квадратного корня в кольце вычетов

Процедура Извлечь\_Квадратный\_Корень(p, a):

Если a < 0:

a = a + p

Конец Если

Пока True:

b = Случайное\_Число(0, p - 1)

x = (b \*b – 4 \* a) mod p

Если Вычислить\_Символ\_Якоби(x, p) == 1:

y = 0

Конец Если

Конец Пока

Пока True:

z = (y \* y – b \* y + a) mod p

Если z == 0:

Вернуть y, y = (y+1) mod p

Конец Если

Конец Пока

3 Тестирование программы

На рисунках 1-6 можно увидеть работу, реализуемой программы, по рассмотренным алгоритмам.

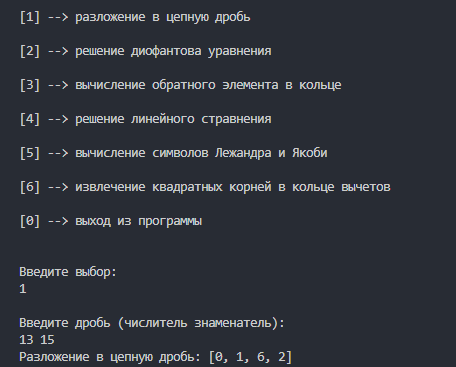


Рисунок 1 – Разложение в цепную дробь

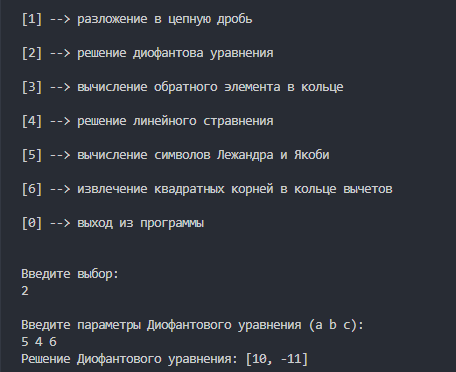


Рисунок 2 – Решение Диофантова уравнения

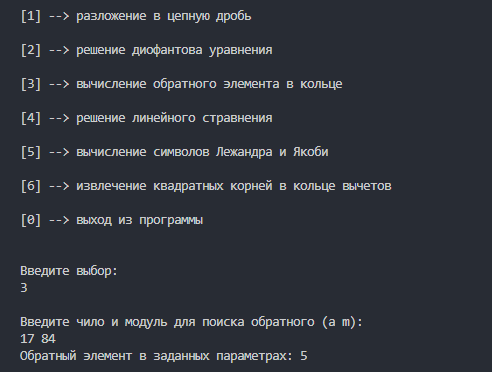


Рисунок 3 – Вычисление обратного элемента в кольце

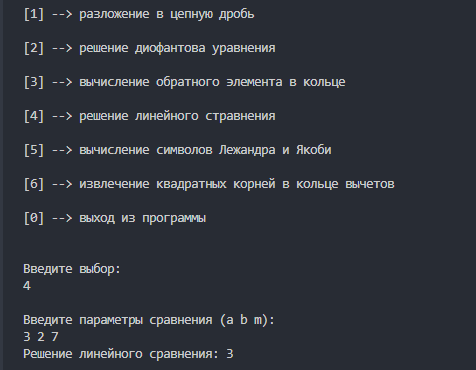


Рисунок 4 – Решение линейного сравнения

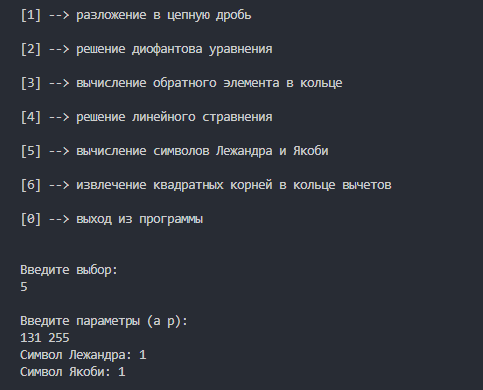


Рисунок 5 - Вычисление символов Лежандра и Якоби

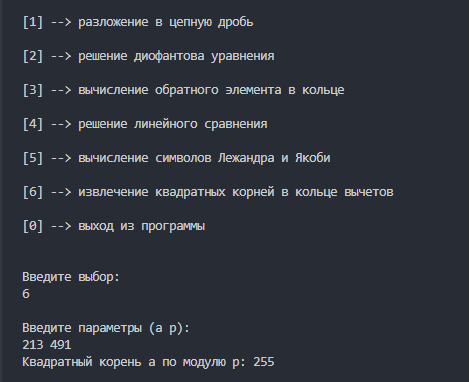


Рисунок 6 - Извлечение квадратного корня в кольце вычетов

ПРИЛОЖЕНИЕ А

**Код программы lab2.rb**

require 'prime'

class Methods

def initialize

end

def make\_chain(a,b)

q = []

while a % b != 0

q << (a / b).to\_i

a %= b

a, b = b, a

end

q << (a / b).to\_i

return q

end

def diophantus(a, b, c)

gcd, x, y = exea(a, b)

p, q = [0, 1], [1, 0]

q\_chain = make\_chain(a, b)

q\_chain.each do |q\_i|

p << q\_i \* p[-1] + p[-2]

q << q\_i \* q[-1] + q[-2]

end

p = p[2..-1]

q = q[2..-1]

if c % gcd == 0

k = p.length

a /= gcd

b /= gcd

c /= gcd

x = (-1) \*\* k \* q[-2] \* c + b

y = -1 \* ((-1) \*\* k \* p[-2] \* c + a)

return x, y

end

end

def inv\_mod(a, m)

g, x, \_ = exea(a, m)

if g != 1

return nil

else

return (x % m + m) % m

end

end

def jacobi(a, p)

return 0 if exea(a, p)[0] != 1

r = 0

t = 1

a = a % p

while a != 0

while a % 2 == 0

a /= 2

end

t = -t if p % 8 == 3 || p % 8 == 5

r = p

p = a

a = r

t = -t if a % 4 == 3 && p % 4 == 3

a = a % p

return t if p == 1

end

end

def sqrt(a, p)

q = p - 1

m = 0

while q % 2 == 0

q /= 2

m += 1

end

if jacobi(a, p) != 1

puts "Нет решения! a / p != 1"

return nil

end

if q.gcd(2) != 1

puts "Нет решения! НОД(2, q) != 1"

return nil

end

b = rand(1..p)

while lezhandr(b, p) != -1

b = rand(1..p)

end

\_as = [a]

ks = [min\_k(a, p, q)]

k = ks[0]

while k != 0

a = (a \* b \*\* (2 \*\* (m - k))) % p

k = min\_k(a, p, q)

\_as << a

ks << k

end

rs = []

r = (\_as[-1] \*\* ((q + 1) / 2)) % p

rs << r

for i in 0..\_as.length - 2

bc = b \*\* (2 \*\* (m - ks[-i - 2] - 1))

r = (rs[i] \* inv\_mod(bc, p)) % p

rs << r

end

return rs[0]

end

def lezhandr(a, p)

return 1 if a == 1

b = a % p

b -= p if b > p / 2

t = b > 0 ? 2 : 1

b = -b

k = 0

while b % 2 == 0

b /= 2

k += 1

end

c = b

t\_1 = 1

t\_1 = (-1) \*\* ((p - 1) / 2) if t % 2 == 1

k\_1 = 1

k\_1 = (-1) \*\* ((p \* p - 1) / 8) if k % 2 == 1

return t\_1 \* k\_1 if c == 1

return t\_1 \* k\_1 \* (-1) \*\* (((c - 1) / 2) \* ((p - 1) / 2)) \* lezhandr(p, c)

end

private

def exea(a, b)

if a == 0

return b, 0, 1

end

gcd, x1, y1 = exea(b % a, a)

x = y1 - (b / a) \* x1

y = x1

return gcd, x, y

end

def linear\_comparison(a, b, m)

x, \_ = diophantus(a, m, b)

x %= m

return x

end

def min\_k(a, p, q)

k = 0

while (a \*\* (2 \*\* k \* q)) % p != 1

k += 1

end

return k

end

end

require './methods.rb'

@methods = Methods.new

def s1

puts "\nВведите дробь (числитель знаменатель):"

input = gets.strip.split.map(&:to\_i)

if input[0].gcd(input[1]) != 1

puts "Задана некорректная дробь"

else

res = @methods.make\_chain(input[0], input[1])

puts "Разложение в цепную дробь: (#{res[0]}; #{res[1..-1].join(", ")})"

end

end

def s2

puts "\nВведите параметры Диофантового уравнения (a b c):"

input = gets.strip.split.map(&:to\_i)

if input[0].gcd(input[1]) != 1

puts "НОД(a,b) != 0 - решения нет!"

else

puts "Решение Диофантового уравнения: #{@methods.diophantus(input[0], input[1], input[2])}"

end

end

def s3

# сделать проверку 84 4

puts "\nВведите чило и модуль для поиска обратного (a m):"

input = gets.strip.split.map(&:to\_i)

if input[0].gcd(input[1]) != 1

puts "НОД(a,m) != 0 - решения нет!"

else

puts "Обратный элемент в заданных параметрах: #{@methods.inv\_mod(input[0], input[1])}"

end

end

def s4

puts "\nВведите параметры сравнения (a b m):"

input = gets.strip.split.map(&:to\_i)

if input[0].gcd(input[2]) != 1

puts "НОД(a,m) != 0 - решения нет!"

else

puts "Решение линейного сравнения: #{@methods.diophantus(input[0], input[2], input[1])[0] % input[2]}"

end

end

def s5

# проверка Лежандра на простоту p

puts "\nВведите параметры (a p):"

input = gets.strip.split.map(&:to\_i)

puts "Символ Якоби: #{@methods.jacobi(input[0], input[1])}"

if input[1].prime?

puts "Символ Лежандра: #{@methods.lezhandr(input[0], input[1])}"

else

puts "Число p не простое, символ Лежандра не может быть найден!"

end

end

def s6

puts "\nВведите параметры (a p):"

input = gets.strip.split.map(&:to\_i)

if !input[1].prime?

puts "Число p не простое, решения нет!"

else

res = @methods.sqrt(input[0], input[1])

puts "Квадратный корень a по модулю p: #{res}" if !res.nil?

end

end

def go

input = -1

while input != 0

puts "\n[1] --> разложение в цепную дробь"

puts "\n[2] --> решение диофантова уравнения"

puts "\n[3] --> вычисление обратного элемента в кольце"

puts "\n[4] --> решение линейного сравнения"

puts "\n[5] --> вычисление символов Лежандра и Якоби"

puts "\n[6] --> извлечение квадратных корней в кольце вычетов"

puts "\n[0] --> выход из программы"

puts "\n\nВведите выбор:"

input = gets.strip.to\_i

case input

when 1

s1

when 2

s2

when 3

s3

when 4

s4

when 5

s5

when 6

s6

else

puts "\n\nКонец работы программы"

return

end

end

end

go