ЛАБОРАТОРНАЯ РАБОТА №1

Основные теоретико-числовые алгоритмы

ЦЕЛЬ РАБОТЫ: изучение основных теоретико-числовых алгоритмов.

Введение

Для реализации криптографических задач очень важно использовать базовые теоретико-числовые алгоритмы, обладающие достаточно высокой вычислительной скоростью. В рамках данной лабораторной работы будут рассмотрены подобные алгоритмы.

**I. Обобщенный (расширенный) алгоритм Евклида**

Вход: Натуральные числа и ,

Выход: , целые числа и такие, что . НОД - наибольший общий делитель.

1.

2. Пока , выполнять

2.1. . - целая часть числа.

2.2.

3.

4. Результат:

Практическое применение данного алгоритма - поиск обратного элемента по модулю.

Пусть . Целые числа и называются сравнимыми по модулю (обозначается , если разность делится на .

Пусть Тогда для того, чтобы найти нужно положить в алгоритме Евклида . Тогда .

**II. Алгоритм быстрого возведения в степень**

**II.A. Алгоритм быстрого возведения в степень**

Пусть необходимо вычислить . Тогда алгоритм быстрого возведения в степень будет иметь следующий вид.

Алгоритм быстрого возведения в степень.

1. Представить в двоичном виде: .

2.

3. Результат: .

**II.B. Алгоритм быстрого возведения в степень по модулю**

Пусть необходимо вычислить . Тогда алгоритм быстрого возведения в степень будет иметь следующий вид.

Алгоритм быстрого возведения в степень по модулю.

1. Представить в двоичном виде: . Положим

2.

2.1

Если

3. Результат: .

**III. Вычисление символа Якоби**

Пусть дано сравнение вида , где - простое число , . Определим для таких и символ Лежандра

.

Пусть , где и числа - простые (не обязательно различные). Символ Якоби определяется равенством , т.е. является произведением соответствующих символов Лежандра. Отсюда, если число - простое, то символ Якоби является символом Лежандра. Стоит также отметить следующее - принимает значения 0, 1 или -1, причем тогда и только тогда, когда Полагают .

Алгоритм нахождения символа Якоби.

Вход. Нечетное целое n≥3, целое число a, 0≤a<n.

Выход. Символ Якоби 

1. g=1.

2. При a=0 результат: .

3. При a=1 результат: .

4. Представить a в виде , где - нечетное число.

5. При k - четном s=1.

При k - нечетном, если , положить s=1.

При k - нечетном, если , положить s=-1.

6. =1 результат: g\*s.

7. Если  и , то s=-s.

8. , , g=g\*s и вернуться на шаг 2.

**IV. Алгоритмы проверки чисел на простоту**

**IV.A. Тест Ферма**

Вход. Нечетное целое n≥5.

Выход. "Число n, вероятно, простое" или "Число n составное".

1. Выбрать случайное целое число a, 2≤a≤n-2.

2. .

3. При r=1 результат: "Число n, вероятно, простое". В противном случае результат "Число n составное".

**IV.B. Тест Соловэя-Штрассена**

Вход. Нечетное целое n≥5.

Выход. "Число n, вероятно, простое" или "Число n составное".

1. Выбрать случайное целое число a, 2≤a≤n-2.

2. .

3. При и  результат: "Число n составное".

4. Вычислить символ Якоби .

5. При  результат: "Число n, вероятно, простое". В противном случае результат: "Число n составное".

**V.C. Тест Миллера-Рабина**

Вход. Нечетное целое n≥5.

Выход. "Число n, вероятно, простое" или "Число n составное".

1. Представить n-1 в виде , где r - нечетное число.

2. Выбрать случайное целое число a, 2≤a≤n-2.

3. 

4. При и  выполнить следующие действия.

4.1. j=1.

4.2. Если j≤s-1 и , то

4.2.1. 

4.2.2. При y=1 результат: "Число n составное".

4.2.3. j=j+1.

4.3. При  результат: "Число n составное".

5. Результат: "Число n, вероятно, простое".

**VI. Генерация простого числа заданной размерности**

Вход: Разрядность искомого простого числа ; параметр .

Выход: Число , простое с вероятностью .

1. Сгенерировать случайное -битное число

2. Положить

3. Проверить, что не делится на простые числа 3,5,7,... (данный шаг можно опустить).

4. Для выполнить следующие действия

4.1. Выбрать случайное число

4.2. Проверить число (переведя его в десятичную систему) тестом Миллера-Рабина для основания . Если тест выдал, что число вероятно простое, то вернуться на шаг 4.1 и увеличить счетчик . В противном случае прекратить цикл и вернуться на шаг 1.

5. Результат: .

Целесообразно предполагать , хотя это замедлит работу программы.

**VII. Решение сравнения первой степени**

Пусть дано сравнение вида

Для того, чтобы данное сравнение имело хотя бы одно решение, необходимо и достаточно, чтобы число делилось на .

Если данное сравнение разрешимо и , тогда множество решений данного сравнения состоит из классов вычетов по модулю , а именно, если - одно из решений, то все другие решения - это , где .

Алгоритм решения сравнения.

Вход: .

Выход: Решение сравнения .

1. Проверяем разрешимость сравнения. Пусть .

Если сравнение не разрешимо (, то на выход "Решений нет", иначе алгоритм продолжает свою работу.

2. Если перейдем к новому сравнению , где , иначе найдем решение сравнения . С помощью расшренного алгоритма Евклида найдем значение и получим решение сравнения . Данное решение подается на выход алгоритма.

3. Аналогично шагу 2 найдем решение для сравнения Получим решение .

4. Найдем все оставшиеся решения сравнения . Данные решения подаются на выход совместно с решением, полученным на шаге 3.

**VIII. Решение сравнения второй степени**

Пусть дано сравнение вида , где число простое и целое число не делится на .

Вход: простое число ; такие целые числа и , что . Символ Лежандра можно вычислить, по алгоритму вычисления символа Якоби.

Выход: Решение сравнения

1. Представить в виде , где -нечетное.

2. Положить

3. Для

3.1 Положить .

3.2 Положить .

3.3 Вычислить При , положить при , положить

3.4 Положить

4. Результат:

**IX. Решение системы сравнений**

Пусть дана простейшая система сравнений первой степени:

,

где , тогда единственное решение имеет вид , где .

**X. Построение конечного поля и реализация операций над данным полем**

Для построения конечного поля .

1. Берем базовое поле , т.е. поле, в котором операции «+» и «⋅» выполняются по модулю .

2. Выбираем неприводимый многочлен степени над полем .

3. Факторизуем кольцо многочленов по модулю многочлена . Получим множество классов остатков при делении на многочлен с операциями «+» и «⋅».

Помимо того, что важно строить операции «+» и «⋅» над полем важной операцией является поиск обратных относительно операций «+» и «⋅». Для операции «+» обратный элемент определяется относительно элемента «0», а относительно операции «⋅» относительно элемента «1».

1. Рабочее задание

1.1. Реализовать описанные в лабораторной работе теоретико-числовые алгоритмы, согласно указанных требований.

2. Требования к реализации

2.1. Должен предоставляться выбор алгоритма в программе.

2.2. Пользователю должен предоставляться выбор метода ввода данных: ручным способом, либо чтением из файла.

2.3. Алгоритмы должны поддерживать большие числа ~2512 бит.

2.4. Для алгоритма X вычисления должны проводиться с помощью многочленов, а результат должен выводиться на экран в десятичном виде.