Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Институт информатики и вычислительной техники

09.03.01 "Информатика и вычислительная техника"

профиль "Программное обеспечение средств вычислительной техники и автоматизированных систем"

Кафедра прикладной информатики и кибернетики

**Практическая работа № 5  
 по дисциплине  
Теория информации**

Выполнил:

студент гр.ИП-911 Сентюров Святослав Алексеевич

ФИО студента

Новосибирск 2023 г.

Задание

1. Составить программу, определяющую характеристики линейного корректирующего кода.

2. Линейный код задан порождающей матрицей, которая записана в текстовом файле. Файл имеет формат: в первой строке через пробел записаны два натуральных числа n (количество строк матрицы) и m (количество столбцов), в следующих n строках записаны через пробел по m нулей и единиц.

Пример файла

3 5

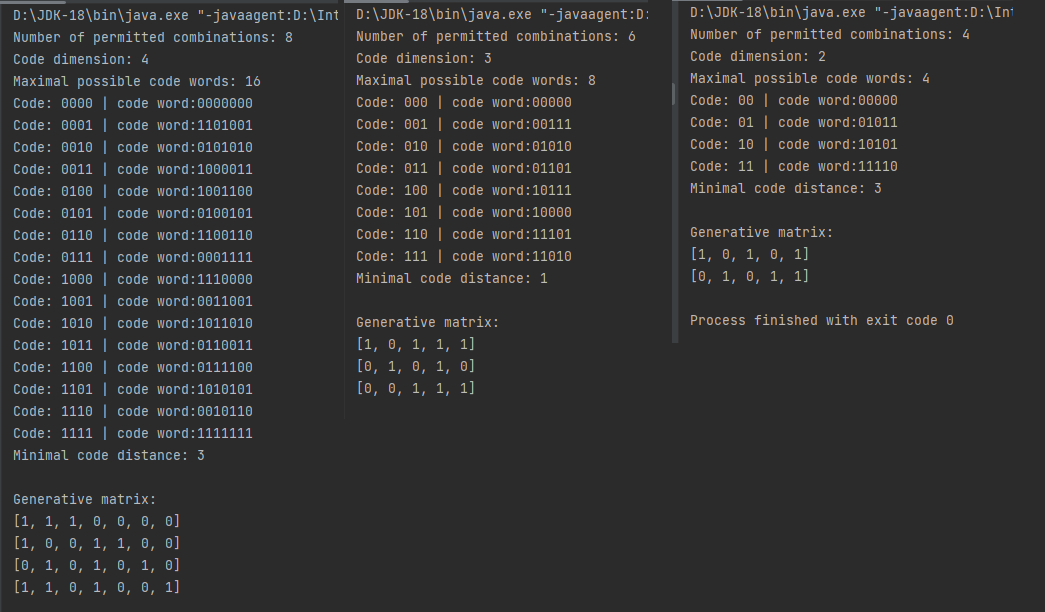
1 0 1 1 1

0 1 0 1 0

0 0 1 1 1

3. По заданной порождающей матрице определить характеристики линейного кода: размерность кода, количество кодовых слов, минимальное кодовое расстояние.

4. Проверить программу для случайно заполненных матриц с различными n и m, для порождающей матрицы кода Хэмминга.



Характеристики кодов (Хэмминг 7 4 слева)

Вывод: Исходя из полученных данных видно, что у Хэмминга минимальное кодовое расстояние равно 3, то позволяет отслеживать и исправлять 1 ошибку. В других матрицах минимальное кодового слово меньше необходимого количества, поэтому для них невозможно будет отследить и исправить ошибки.

Листинг кода:

import java.io.\*;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
public class Main {  
 public static void main(String[] args) throws IOException {  
 ArrayList<int[]> GMatrix;  
 ArrayList<String> codes;  
 int wordsNum, n, d;  
 String path = "D:\\Study\\4Rehc\\TI\\lab5\\TI\_Last\\lab5\\code.txt";  
 GMatrix = *fileReader*(path);  
 n = GMatrix.size();  
 wordsNum = (int)Math.*pow*(2, n);

int permitted = 2\*n;  
 System.*out*.println("Number of permitted combinations: " + permitted);  
  
 System.*out*.println("Code dimension: " + n);  
 System.*out*.println("Maximal possible code words: " + wordsNum);  
 codes = *codeWordGenerator*(GMatrix, n);  
 d = *minimalDistance*(codes);  
 System.*out*.println("Minimal code distance: " + d);  
 System.*out*.println("\nGenerative matrix:");  
 *matrixOutput*(GMatrix);  
 }  
  
 public static int minimalDistance(ArrayList<String> codes){  
 int toReturn = Integer.*MAX\_VALUE*, dist;  
 for(int i = 1; i < codes.size()-1; i++)  
 for(int j = i+1; j < codes.size(); j++){  
 dist = *distance*(codes.get(i), codes.get(j));  
 toReturn = dist < toReturn? dist: toReturn;  
 }  
 return toReturn;  
 }  
  
 private static int distance(String first, String second){  
 int toReturn = 0;  
 for(int i = 0; i < first.length(); i++){  
 if(first.charAt(i) != second.charAt(i))  
 toReturn++;  
 }  
 return toReturn;  
 }  
  
 public static ArrayList<String> codeWordGenerator(ArrayList<int[]>matrix, int n){  
 ArrayList<String> codeWords = new ArrayList<>(), codes;  
 ArrayList<String> permittedCodeWords = new ArrayList<>();  
 codes = *codeGenerator*(n);  
 int wordNum = (int)Math.*pow*(2, n);  
  
 for(int i = 0; i < wordNum; i++) {  
 codeWords.add(*codeMatrixMultiplier*(codes.get(i), matrix));  
 System.*out*.println("Code: " + codes.get(i) + " | code word:" + codeWords.get(i));  
 }  
  
 return codeWords;  
 }  
  
 private static ArrayList<String> codeGenerator(int n){  
 int wordsCount = (int)Math.*pow*(2, n);  
 ArrayList<String> codeWords = new ArrayList<>();  
 String code;  
 for(int i = 0; i < wordsCount; i++){  
 if(i == 0)  
 code = *setZero*(n);  
 else  
 code = *xorPlusOne*(codeWords.get(i-1));  
 codeWords.add(code);  
 }  
 return codeWords;  
 }  
  
 private static String setZero(int n){  
 String toReturn = "";  
 for(int i = 0; i < n; i++)  
 toReturn += "0";  
 return toReturn;  
 }  
  
 private static String xorPlusOne(String previous){  
 StringBuilder code = new StringBuilder(previous);  
 int memory = 0;  
  
 if(code.charAt(previous.length()-1) == '1') {  
 code.setCharAt(previous.length()-1, '0');  
 memory = 1;  
 }  
 else{  
 code.setCharAt(previous.length()-1, '1');  
 }  
  
 for(int pos = previous.length()-2; pos >= 0; pos--){  
 if(memory == 1){  
 if(code.charAt(pos) == '1')  
 code.setCharAt(pos, '0');  
 else{  
 code.setCharAt(pos, '1');  
 memory = 0;  
 }  
 }  
 }  
 return code.toString();  
 }  
  
 private static String codeMatrixMultiplier(String codeWord, ArrayList<int[]>matrix){  
 String toReturn = "";  
 int sum;  
 for(int i = 0; i < matrix.get(0).length; i++){  
 sum = 0;  
 for(int j = 0; j < codeWord.length(); j++){  
 if(codeWord.charAt(j) == '1' && matrix.get(j)[i] == 1)  
 sum++;  
 }

if(sum % 2 != 0)  
 toReturn += "1";  
 else  
 toReturn += "0";  
 }  
 return toReturn;  
 }  
  
 public static ArrayList<int[]> fileReader(String path) throws IOException {  
 BufferedReader file = new BufferedReader(new FileReader(path));  
 String params = file.readLine();  
 List<String> parameter = List.*of*(params.split(" "));  
 int height = Integer.*valueOf*(parameter.get(0)), width = Integer.*valueOf*(parameter.get(1));  
  
 String line;  
 ArrayList<int[]> toReturn = new ArrayList<>();  
 for(int i = 0; i < height; i++){  
 line = file.readLine();  
 int[] row = new int[width];  
 for(int j = 0; j < width; j++){  
 row[j] = Integer.*valueOf*(Character.*toString*(line.charAt(j)));  
 }  
 toReturn.add(row);  
 }  
  
 return toReturn;  
 }  
  
 public static void matrixOutput(ArrayList<int[]> matrix){  
 for (int[] row: matrix) {  
 System.*out*.println(Arrays.*toString*(row));  
 }  
 }  
}