Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ

УНИВЕРСИТЕТ»

«Защита информации и надёжность информационных систем»

Отчёт по лабораторной работе №4

ИЗБЫТОЧНОЕ КОДИРОВАНИЕ ДАННЫХ В ИНФОРМАЦИОННЫХ СИСТЕМАХ. ИТЕРАТИВНЫЕ КОДЫ

Выполнил: Лешук Д. И.

ФИТ 3 курс 7 группа

Преподаватель: Николайчук А.Н

Минск 2024

**Цель:** приобретение практических навыков кодирования/ декодирования двоичных данных при использовании итеративных кодов.

**Практическое задание:**

1) вписывать произвольное двоичное представление информационного слова Хk (кодируемой информации) длиной k битов в двумерную матрицу размерностью в соответствии с вариантом либо в трехмерную матрицу в соответствии с вариантом (указаны в табл. 5.2);

2) вычислять проверочные биты (биты паритетов): а) по двум; б) по трем; в) по четырем направлениям (группам паритетов);

3) формировать кодовое слово Xn присоединением избыточных символов к информационному слову;

4) генерировать ошибку произвольной кратности (i, i > 0), распределенную случайным образом среди символов слова Xn, в результате чего формируется кодовое слово Yn;

5) определять местоположение ошибочных символов итеративным кодом в слове Yn в соответствии с используемыми группами паритетов по пункту (2) и исправлять ошибочные символы (результат исправления – слово Yn’);

6) выполнять анализ корректирующей способности используемого кода (количественная оценка) путем сравнения соответствующих слов Xn и Yn’; результат анализа может быть представлен в виде отношения общего числа сгенерированных кодовых слов с ошибками определенной одинаковой кратности (с одной ошибкой, с двумя ошибками и т. д.) к числу кодовых слов, содержащих ошибки этой кратности, которые правильно обнаружены и которые правильно скорректированы.

**Выполнение работы:**

Дополним наше входное слово – строка «h6u9z». Теперь при преобразовании в ASCII её длина будет составлять 40 символов, как и требуется в 5 варианте (вообще вариант 10, но проходя по списку вариантов 2 раза, имеем 5 вариант). Вид входного сообщения в ASCII представлен на рисунке 1.



Рисунок 1 – Входное сообщение

Далее нам необходимо сформировать двумерную матрицу размера 5 на 8 на основе нашего информационного слова, функция для генерации матрицы – createInformationMatrix, представлена в приложении А, листинге 1. В итоге применения данной функции имеем матрицу 5 на 8 – рисунок 2.

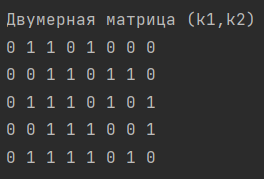


Рисунок 2 – Результат применения функции генерации матрицы

Сразу же после этого перейдём к вычислению паритетов (горизонтальных и вертикальных) – функция addParityBits, продемонстрирована в листинге 1 приложения А.

В итоге мы получили расширение нашей двумерной матрицы до размеров 6 на 9, где последние строка и столбец – значения паритетов для строки и столбца соответственно, а элемент на позиции [6;9] – является суммой паритетов – рисунок 3.

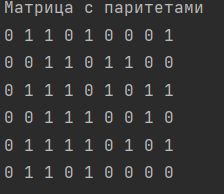


Рисунок 3 – Матрица с паритетами

Из матрицы с паритетами мы можем получить наше закодированное слово путём преобразования её в строку простым двойным циклом. Закодированное слово, которое мы можем видеть на рисунке 4, имеет длину 54.



Рисунок 4 – Закодированное слово

Теперь нам нужно сделать n-ое количество ошибок в нашем входном сообщении – пусть n = 3. Воспользуемся методом из прошлой лабораторной работы для инвертирования бит в случайном месте строки. Далее нам нужна функция, на основе которой мы будем исправлять ошибки в нашем сообщении Yn. Алгоритм следующий: мы вычисляем матрицу паритетов для нашего сообщения с ошибками, далее сравниваем эти матрицы, вследствие чего формируем вектор ошибок. Далее с помощью вектора ошибок инвертируем биты в некорректном сообщении. Имеем такой результат – рисунок 5.

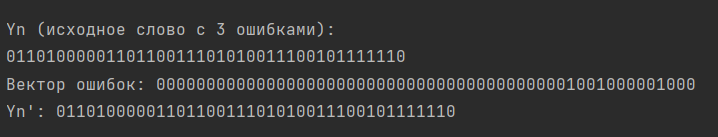


Рисунок 5 – Результат выполнения функции исправления ошибок

В результате первого выполнения ошибки были выявлены верно, однако ошибки исправлены не целиком. Повторим выполнение ещё 4 раза. В результате имеем: сгенерировано  вариантов кодового слова с 3 ошибками. Кратность ошибки определена верно раза, а полностью исправлена раза. Следовательно имеем соотношения , а также 4.

Далее попробуем поработать с трёхмерной матрицей размерностью 2x10x2 (в соответствии с варинатом №5). Введём почти аналогичную функцию, которая бует заполнять трёхмерный массив имеющейся бинарной последовательностью – createThreeDimensionalMatrix, представлена в приложении А. Так как в двумерную консоль будет невозможно вывести трёхмерную матрицу, выведем её «по слоям», показано на рисунке 6.

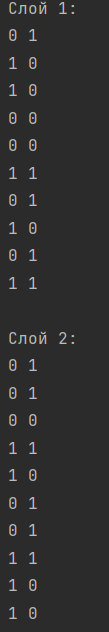


Рисунок 6 – «Слои» трехмерной матрицы

Далее определим на основе ранее используемой функции для вычисления битов паритета новую, которая будет высчитывать горизонтальные, вертикальные, диагональные и z-паритеты в трехмерной матрице – по 5-и направлениям. Её название – addParityBits3Dim, её подробное описание можно также найти в приложении А. В результате её использования получаем биты паритета в нашей матрице, а также записываем её построчно для формирования закодированного слова – рисунок 7.

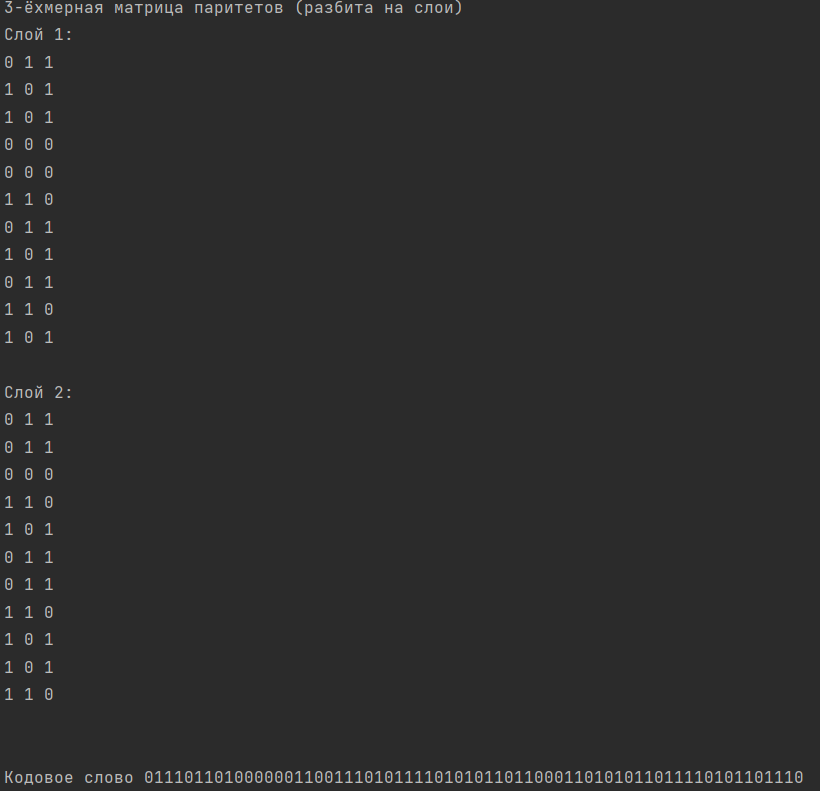


Рисунок 7 – Матрица с паритетами и закодированное сообщение

Далее возьмём уже имеющееся у нас входное слово с 3-мя ошибками, и трёхмерную матрицу паритетов, и на основе их вычислим вектор ошибки и попробуем их исправить с помощью функции correctErrors3D, которую также можно найти в листинге 1 приложения А. Первый результат выполнения представлен на рисунке 8.



Рисунок 8 – Вычисленный вектор ошибок и исправление слова

Как мы можем видеть, мы получили, мягко говоря, непредсказуемый результат. Вектор ошибок определён неверно, соответственно, и сообщение тоже не было скорректировано. После повторения выполнение программы ещё 4 раза получаем интересуемые нас соотношения для данной входной последовательности - , 2. Это может быть обусловлено некорректно применённым алгоритмом со стороны автора, либо же «неудобной» входной последовательностью или сложностью обнаружения ошибок в данной структуре. В Листинге 2 представлен код функции main, в которой и проходило выполнение основного задания.

**Вывод:** В результате лабораторной работы были приобретены теоретические знания о итеративных кодах, проверен на практике способ исправления ошибок в 40-битном сообщении с помощью итеративного кода и использования паритетов, а также выполнен анализ корректирующей способности используемого кода: вероятность обнаружения ошибки **,** а корректировки равно 0,4. Для многомерного же кода вероятности получились немного меньше 0,4 и 0,2 соответственно.

**Приложение А**

import java.util.Random;

public class InfoMetrics {

public static String binary(String text) {

StringBuilder binaryStr = new StringBuilder();

for (char c : text.toCharArray()) {

binaryStr.append(String.format("%8s", Integer.toBinaryString(c)).replaceAll(" ", "0"));

}

return binaryStr.toString();

}

public static int[][] createInformationMatrix(String binaryWord) {

// вариант 5, длина слова = 40 символам

if (binaryWord.length() != 40) {

throw new IllegalArgumentException("Длина двоичного слова должна быть 40 символов.");

}

int[][] infoMatrix = new int[5][8];

int index = 0;

for (int i = 0; i < 5; i++) {

for (int j = 0; j < 8; j++) {

if (index < binaryWord.length()) {

infoMatrix[i][j] = Character.getNumericValue(binaryWord.charAt(index));

index++;

} else {

infoMatrix[i][j] = 0;

}

}

}

return infoMatrix;

}

public static int[][] addParityBits(int[][] infoMatrix) {

int rows = infoMatrix.length;

int cols = infoMatrix[0].length;

int[][] parityMatrix = new int[rows + 1][cols + 1];

// Копируем информационную матрицу в матрицу с паритетами

for (int i = 0; i < rows; i++) {

System.arraycopy(infoMatrix[i], 0, parityMatrix[i], 0, cols);

}

// Вычисляем паритеты для строк

for (int i = 0; i < rows; i++) {

int parity = 0;

for (int j = 0; j < cols; j++) {

parity ^= parityMatrix[i][j];

}

parityMatrix[i][cols] = parity; // Добавляем паритет в конец строки

}

// Вычисляем паритеты для столбцов

for (int j = 0; j < cols; j++) {

int parity = 0;

for (int i = 0; i < rows; i++) {

parity ^= parityMatrix[i][j];

}

parityMatrix[rows][j] = parity; // Добавляем паритет в конец столбца

}

// Вычисляем паритет для паритетов (диагональный элемент)

int diagonalParity = 0;

for (int i = 0; i < rows; i++) {

diagonalParity ^= parityMatrix[i][cols];

}

for (int j = 0; j < cols; j++) {

diagonalParity ^= parityMatrix[rows][j];

}

parityMatrix[rows][cols] = diagonalParity;

return parityMatrix;

}

public static String matrixToCodeword(int[][] parityMatrix) {

StringBuilder codeword = new StringBuilder();

for (int i = 0; i < parityMatrix.length; i++) {

for (int j = 0; j < parityMatrix[i].length; j++) {

codeword.append(parityMatrix[i][j]);

}

}

return codeword.toString();

}

public static String introduceErrors(String word, int numErrors) {

Random rand = new Random();

char[] wordArray = word.toCharArray();

for (int i = 0; i < numErrors; i++) {

int errorPos = rand.nextInt(word.length());

wordArray[errorPos] = (wordArray[errorPos] == '0') ? '1' : '0';

}

return new String(wordArray);

}

public static String correctErrors(int[][] parityMatrix\_no\_errs, String encodedWord\_with\_errs) {

int[][] infoMatrix\_with\_errs = InfoMetrics.createInformationMatrix(encodedWord\_with\_errs);

int[][] parityMatrix\_with\_errs = InfoMetrics.addParityBits(infoMatrix\_with\_errs);

// Вычисляем вектор ошибок

StringBuilder errorVector = new StringBuilder();

for (int i = 0; i < parityMatrix\_with\_errs.length; i++) {

for (int j = 0; j < parityMatrix\_with\_errs[i].length; j++) {

errorVector.append(parityMatrix\_no\_errs[i][j] ^ parityMatrix\_with\_errs[i][j]);

}

}

System.out.println("Вектор ошибок: " + errorVector);

// Исправляем ошибки

StringBuilder correctedWord = new StringBuilder();

for (int i = 0; i < encodedWord\_with\_errs.length(); i++) {

correctedWord.append(encodedWord\_with\_errs.charAt(i) ^ errorVector.charAt(i));

}

return correctedWord.toString();

}

public static int[][][] createThreeDimensionalMatrix(String binaryWord) {

if (binaryWord.length() != 40) {

throw new IllegalArgumentException("Длина двоичного слова должна быть 40 символов.");

}

int[][][] matrix = new int[2][10][2];

int index = 0;

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 10; j++) {

for (int k = 0; k < 2; k++) {

if (index < binaryWord.length()) {

matrix[i][j][k] = Character.getNumericValue(binaryWord.charAt(index));

index++;

} else {

matrix[i][j][k] = 0;

}

}

}

}

return matrix;

}

public static int[][][] addParityBits3Dim(int[][][] matrix) {

int[][][] parityMatrix = new int[2][11][3];

// Копируем информационную матрицу в матрицу с паритетами

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 10; j++) {

System.arraycopy(matrix[i][j], 0, parityMatrix[i][j], 0, 2);

}

}

// Вычисляем паритеты для строк

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 10; j++) {

int parity = 0;

for (int k = 0; k < 2; k++) {

parity ^= parityMatrix[i][j][k];

}

parityMatrix[i][j][2] = parity; // Добавляем паритет в конец строки

}

}

// Вычисляем паритеты для столбцов

for (int i = 0; i < 2; i++) {

for (int k = 0; k < 2; k++) {

int parity = 0;

for (int j = 0; j < 10; j++) {

parity ^= parityMatrix[i][j][k];

}

parityMatrix[i][10][k] = parity; // Добавляем паритет в конец столбца

}

}

// Вычисляем паритеты по диагонали

for (int i = 0; i < 2; i++) {

int diagonalParity = 0;

for (int j = 0; j < 10; j++) {

diagonalParity ^= parityMatrix[i][j][j % 2];

}

parityMatrix[i][10][2] ^= diagonalParity; // Добавляем паритет в диагональный элемент

}

// Вычисляем z-паритеты

for (int j = 0; j < 10; j++) {

for (int k = 0; k < 2; k++) {

int zParity = parityMatrix[0][j][k] ^ parityMatrix[1][j][k];

parityMatrix[0][10][k] ^= zParity;

parityMatrix[1][10][k] ^= zParity;

}

}

return parityMatrix;

}

public static void printThreeDimensionalMatrix(int[][][] matrix) {

for (int i = 0; i < matrix.length; i++) {

System.out.println("Слой " + (i + 1) + ":");

for (int j = 0; j < matrix[i].length; j++) {

for (int k = 0; k < matrix[i][j].length; k++) {

System.out.print(matrix[i][j][k] + " ");

}

System.out.println();

}

System.out.println();

}

}

public static String matrixToCodeword3Dim(int[][][] parityMatrix) {

StringBuilder codeword = new StringBuilder();

for (int i = 0; i < parityMatrix.length; i++) {

for (int j = 0; j < parityMatrix[i].length; j++) {

for (int k = 0; k < parityMatrix[i][j].length; k++) {

codeword.append(parityMatrix[i][j][k]);

}

}

}

return codeword.toString();

}

public static String correctErrors3D(int[][][] parityMatrixNoErrs, String encodedWordWithErrs) {

int[][][] infoMatrixWithErrs = createThreeDimensionalMatrix(encodedWordWithErrs);

int[][][] parityMatrixWithErrs = addParityBits3Dim(infoMatrixWithErrs);

StringBuilder errorVector = new StringBuilder();

for (int i = 0; i < parityMatrixWithErrs.length; i++) {

for (int j = 0; j < parityMatrixWithErrs[i].length; j++) {

for (int k = 0; k < parityMatrixWithErrs[i][j].length; k++) {

errorVector.append(parityMatrixNoErrs[i][j][k] ^ parityMatrixWithErrs[i][j][k]);

}

}

}

System.out.println("Вектор ошибок: " + errorVector);

StringBuilder correctedWord = new StringBuilder();

for (int i = 0; i < encodedWordWithErrs.length(); i++) {

correctedWord.append(encodedWordWithErrs.charAt(i) ^ errorVector.charAt(i));

}

return correctedWord.toString();

}

}

Листинг 1 – Полный код вспомогательног о класса

import java.math.BigInteger;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.io.IOException;

public class Main {

public static void main(String[] args) {

String givenText = "";

try {

givenText = new String(Files.readAllBytes(Paths.get("src/source.txt")));

} catch (IOException e) {

e.printStackTrace();

}

givenText = InfoMetrics.binary(givenText);

System.out.println("Исходное сообщение в двоичной форме (Xk)");

System.out.println(givenText);

int[][] dDimentionalMatrix = InfoMetrics.createInformationMatrix(givenText);

System.out.println("\nДвумерная матрица (k1,k2)");

for (int i = 0; i < dDimentionalMatrix.length; i++) {

for (int j = 0; j < dDimentionalMatrix[i].length; j++) {

System.out.print(dDimentionalMatrix[i][j] + " ");

}

System.out.println();

}

int[][] matrixWParitets = InfoMetrics.addParityBits(dDimentionalMatrix);

System.out.println("\nМатрица с паритетами");

for (int i = 0; i < matrixWParitets.length; i++) {

for (int j = 0; j < matrixWParitets[i].length; j++) {

System.out.print(matrixWParitets[i][j] + " ");

}

System.out.println();

}

String codeWordXn = InfoMetrics.matrixToCodeword(matrixWParitets);

System.out.println("\nСобщение Xn: " + codeWordXn );

String codeWordYn = InfoMetrics.introduceErrors(givenText,3);

System.out.println("\nYn (исходное слово с 3 ошибками):\n" + codeWordYn);

String corrWYnStr = InfoMetrics.correctErrors(matrixWParitets, codeWordYn);

System.out.println("Yn': " + corrWYnStr);

System.out.println("\n\nРабота с трёхмерной матрицей (2,10,2):");

int[][][] threeDimMatrix = InfoMetrics.createThreeDimensionalMatrix(givenText);

InfoMetrics.printThreeDimensionalMatrix(threeDimMatrix);

int[][][] threeDimParitets = InfoMetrics.addParityBits3Dim(threeDimMatrix);

System.out.println("3-ёхмерная матрица паритетов (разбита на слои)");

InfoMetrics.printThreeDimensionalMatrix(threeDimParitets);

String codewordEncoded3Dim = InfoMetrics.matrixToCodeword3Dim(threeDimParitets);

System.out.println("\nКодовое слово " + codewordEncoded3Dim);

System.out.println("\nYn (исходное слово с 3 ошибками):\n" + codeWordYn);

String resultWord = InfoMetrics.correctErrors3D(threeDimParitets, InfoMetrics.introduceErrors(givenText,2));

System.out.println("Исправленное слово: " + resultWord);

}}

Листинг 2 – Код функции выполнения программы