Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ

УНИВЕРСИТЕТ»

«Защита информации и надёжность информационных систем»

Отчёт по лабораторной работе №6

ПЕРЕМЕЖЕНИЕ/ДЕПЕРЕМЕЖЕНИЕ ДАННЫХ ВИНФОРМАЦИОННО-ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

Выполнил: Лешук Д. И.

ФИТ 3 курс 7 группа

Преподаватель: Николайчук А.Н

Минск 2024

**Цель:** приобретение практических навыков использования методов перемежения/деперемежения двоичных данных в информационных система.

**Практическое задание:**

1. Необходимо разработать авторское приложение в соответствии с целью лабораторной работы. По умолчанию используется блочный перемежитель/деперемежитель. Задание выполняется по указанию преподавателя в соответствии с вариантом 10: используемый корректирующий код – Хемминга, длина пакета ошибок = 4,6,8, число столбцов в матрице = 7, длина сообщения = 14 байт, длина информационного слова k = 5.

2. Местоположение заданной группы ошибок выбирается (генерируется) случайным образом. Необходимо для группы ошибок каждой длины сгенерировать 30−40 случайных ситуаций. После деперемежения и исправления ошибок в сообщении сравнить передаваемую последовательность и полученную после исправления ошибок. Проанализировать эффективность перемежения/деперемежения.

3. Результаты оформить в виде отчета по установленным правилам

**Выполнение работы:**

Для начала сгенерируем входную последовательность, состоящую из 14 байт = 112 бит – рисунок 1.Используем параметр seed для генерации одинакового сообщения в разных запусках приложения. Случайно же будем задавать позицию ошибок.

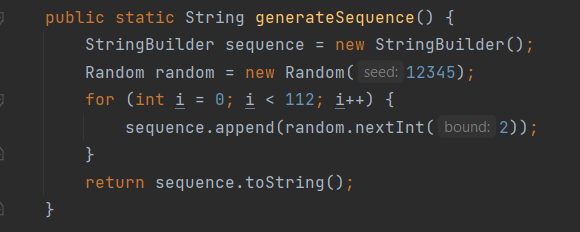


Рисунок 1 – Функция генерации входного сообщения

В итоге имеем последовательность: «0111100000010110011010011001001110010110011001011100001000000101001000001110000110110110110101111110100101110000».

Далее рассматриваем каждые 5 бит как отдельное сообщение и вычисляем для него избыточные биты по Хеммингу, формируем строку из Xn сообщений. Данная функция представлена на рисунке 2.

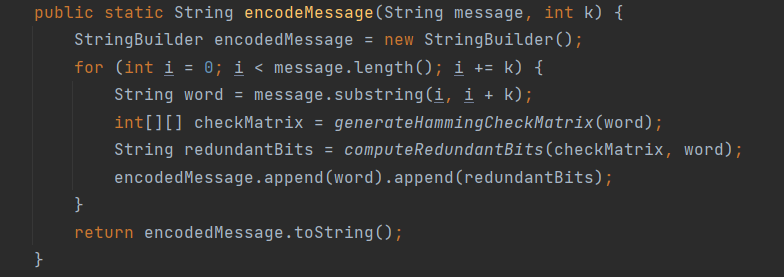


Рисунок 2 – Функция кодирования исходного сообщения

Далее наша последовательность принимает такой вид:  
«0111101100010000010010110100011010010011011001101001111110011010110100011010001010111100110001011000000000101011001011000000001110000000100110110110110100110111101111011110000100110101111010000000».

Определяем функцию для генерации матрицы перемежения – рисунок 3, по установленным правилам.

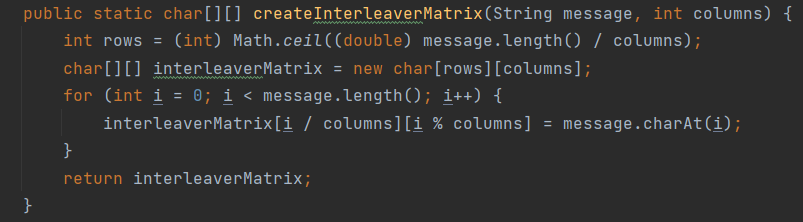


Рисунок 3 – Функция генерации матрицы перемежения

Матрица перемежения в данном варианте должна иметь 7 столбцов, функция генерации возвращает такую матрицу – рисунок 4.

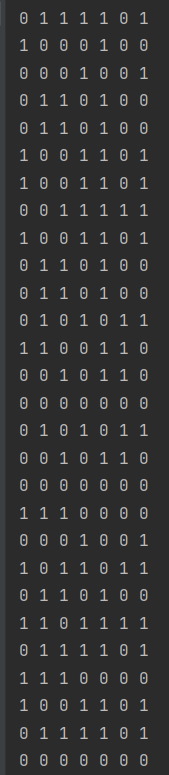


Рисунок 4 – Матрица перемежения

После перемежения сообщение соответствует последовательности: «0100011010001000001010101100100110000111100100100111101010011001011001001010110110101010011110010001000110110110110111111110110010000111011000000001000111011000101000001010011110010001000110110110».

Следующим шагом определим функцию генерации пакета ошибок – рисунок 5. Ошибки генерируются при каждом выполнении программы на разных позициях.

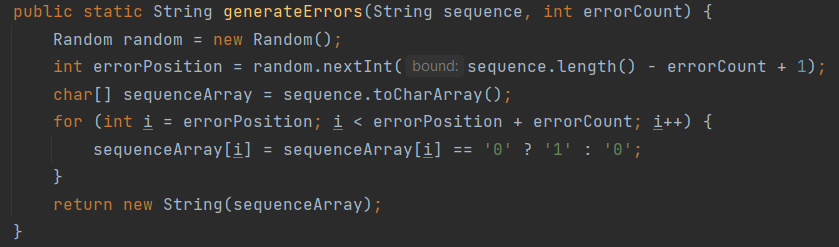


Рисунок 5 – Функция генерации пакета ошибок

Далее определяем функцию генерации матрицы деперемежения – рисунок 6, и выполняем её.

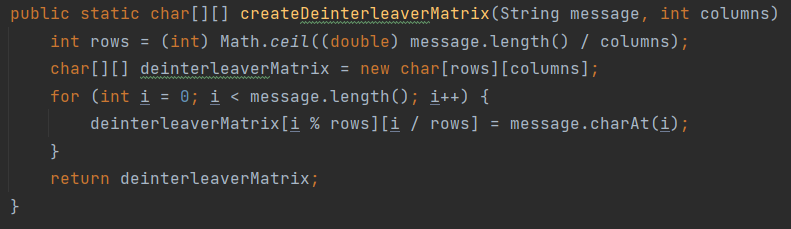


Рисунок 6 – Функция генерации матрицы деперемежения

Преобразуем её в строку, получим последовательность после деперемежения с ошибками, разнесёнными по разным частям сообщения. Далее применим к данному сообщению функцию correctMessage, которая будет проходить по каждому кодовому слову и исправлять ошибки – полный код функции представлен в приложении А. Сравниваем исходное сообщение с входным и выводим на консоль результат сравнения – рисунок 7.

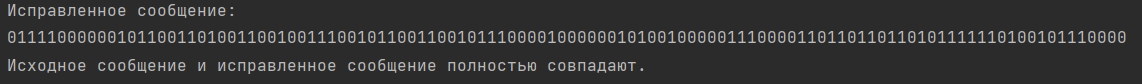


Рисунок 7 – Исправленное сообщение и результат сравнения

Как и было сказано в условии, повторим выполнение программы, допустим, 35 раз. Для группы из 4-ех ошибок в 22 из 35 случаев все ошибки были исправлены. Для группы ошибок длиной 6: в 12 случаях из 35 ошибки были исправлены. Для группы ошибок длиной 8: 17 из 35 вариаций сообщения с ошибками успешно исправлены.

**Вывод:** В результате лабораторной работы были приобретены теоретические знания о методе исправления групповых ошибок – перемежении, получены практические навыки применения методов перемежения и деперемежения к сообщению длиной более ста бит на основе кода Хемминга, проанализирована эффективность перемежения и деперемежения на группе ошибок длиной 4, 6 и 8.

**Приложение А**

import java.util.Random;

public class InfoMetrics {

// Функция для генерации последовательности из 14 байт

public static String generateSequence() {

StringBuilder sequence = new StringBuilder();

Random random = new Random(12345);

for (int i = 0; i < 112; i++) {

sequence.append(random.nextInt(2));

}

return sequence.toString();

}

// Функция для разбиения сообщения на слова длиной k и добавления избыточных бит

public static String encodeMessage(String message, int k) {

StringBuilder encodedMessage = new StringBuilder();

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

String word = message.substring(i, i + k);

int[][] checkMatrix = generateHammingCheckMatrix(word);

String redundantBits = computeRedundantBits(checkMatrix, word);

encodedMessage.append(word).append(redundantBits);

}

return encodedMessage.toString();

}

// Функция для создания матрицы перемежения

public static char[][] createInterleaverMatrix(String message, int columns) {

int rows = (int) Math.ceil((double) message.length() / columns);

char[][] interleaverMatrix = new char[rows][columns];

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

interleaverMatrix[i / columns][i % columns] = message.charAt(i);

}

return interleaverMatrix;

}

public static String matrixToString(char[][] matrix) {

StringBuilder sequence = new StringBuilder();

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

for (char[] row : matrix) {

sequence.append(row[j]);

}

}

return sequence.toString();

}

public static String matrixToStringAlternate(char[][] matrix) {

StringBuilder sequence = new StringBuilder();

for (char[] row : matrix) {

for (char element : row) {

sequence.append(element);

}

}

return sequence.toString();

}

// Функция для генерации группы ошибок в строке

public static String generateErrors(String sequence, int errorCount) {

Random random = new Random();

int errorPosition = random.nextInt(sequence.length() - errorCount + 1);

char[] sequenceArray = sequence.toCharArray();

for (int i = errorPosition; i < errorPosition + errorCount; i++) {

sequenceArray[i] = sequenceArray[i] == '0' ? '1' : '0';

}

return new String(sequenceArray);

}

// Функция для вывода матрицы перемежения

public static void printMatrix(char[][] matrix) {

for (char[] row : matrix) {

for (char element : row) {

System.out.print(element + " ");

}

System.out.println();

}

}

// Функция для создания матрицы деперемежения

public static char[][] createDeinterleaverMatrix(String message, int columns) {

int rows = (int) Math.ceil((double) message.length() / columns);

char[][] deinterleaverMatrix = new char[rows][columns];

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

deinterleaverMatrix[i % rows][i / rows] = message.charAt(i);

}

return deinterleaverMatrix;

}

// Функция для исправления ошибок в сообщении

public static String correctMessage(String message, int wordLength) {

StringBuilder correctedMessage = new StringBuilder();

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

String word = message.substring(i, i + wordLength);

int[][] checkMatrix = generateHammingCheckMatrix(word);

String redundantBits = computeRedundantBits(checkMatrix, word);

String correctedWord = word.substring(0, 4); // Исправляем ошибку

correctedMessage.append(correctedWord);

}

return correctedMessage.toString();

}

// Ваши функции для вычисления избыточных бит по коду Хемминга

public static int[][] generateHammingCheckMatrix(String infoWord) {

int k = infoWord.length();

int r = 0;

while (Math.pow(2, r) < k + r + 1) {

r++;

}

int n = k + r;

int[][] checkMatrix = new int[r][n];

int count = 1;

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

String binary = Integer.toBinaryString(count++);

while (binary.length() < r) {

binary = "0" + binary;

}

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

checkMatrix[j][i] = binary.charAt(r - j - 1) - '0';

}

}

for (int i = k; i < n; i++) {

checkMatrix[i - k][i] = 1;

}

return checkMatrix;

}

public static String computeRedundantBits(int[][] checkMatrix, String infoWord) {

StringBuilder redundantBits = new StringBuilder();

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

int sum = 0;

for (int j = 0; j < infoWord.length(); j++) {

sum += checkMatrix[i][j] \* Character.getNumericValue(infoWord.charAt(j));

}

redundantBits.append(sum % 2);

}

return redundantBits.toString();

}

}

Листинг 1 – Полный код вспомогательног о класса

public class Main {

public static void main(String[] args) {

String sequence = InfoMetrics.generateSequence();

System.out.println("Входное сообщение: \n" + sequence);

String encodedMessage = InfoMetrics.encodeMessage(sequence, 5);

System.out.println("Закодированное сообщение: \n" + encodedMessage);

char[][] interleaverMatrix = InfoMetrics.createInterleaverMatrix(encodedMessage, 7);

InfoMetrics.printMatrix(interleaverMatrix);

String interleavedSequence = InfoMetrics.matrixToString(interleaverMatrix);

System.out.println("Последовательность после перемежения: \n" + interleavedSequence);

String sequenceWithErrors = InfoMetrics.generateErrors(interleavedSequence, 8);

System.out.println("Последовательность с ошибками: \n" + sequenceWithErrors);

char[][] deinterleaverMatrix = InfoMetrics.createDeinterleaverMatrix(sequenceWithErrors, 7);

InfoMetrics.printMatrix(deinterleaverMatrix);

String deinterleavedSequence = InfoMetrics.matrixToStringAlternate(deinterleaverMatrix);

System.out.println("Последовательность после деперемежения: \n" + deinterleavedSequence);

String correctedMessage = InfoMetrics.correctMessage(deinterleavedSequence, 7);

System.out.println("Исправленное сообщение: \n" + correctedMessage);

if (sequence.equals(correctedMessage)) {

System.out.println("Исходное сообщение и исправленное сообщение полностью совпадают.");

} else {

System.out.println("Исходное сообщение и исправленное сообщение не совпадают.");

}

}

}

Листинг 2 – Код функции выполнения программы