Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ

УНИВЕРСИТЕТ»

«Защита информации и надёжность информационных систем»

Отчёт по лабораторной работе №8

СЖАТИЕ/РАСПАКОВКА ДАННЫХ НА ОСНОВЕ СТАТИСТИЧЕСКИХ МЕТОДОВ

Выполнил: Лешук Д. И.

ФИТ 3 курс 7 группа

Преподаватель: Николайчук А.Н

Минск 2024

**Цель:** приобретение практических навыков использования статистических методов Шеннона − Фано и Хаффмана (Shannon-Fano and Huffman coding) для сжатия/распаковки данных..

**Практическое задание:**

1. Разработать авторское приложение в соответствии с целью лабораторной работы.

2. С помощью приложения выполнить прямое и обратное преобразования сообщения, состоящего из собственных имени и фамилии. Можно использовать любой из известных методов сортировки символов массива. Метод кодировк – Шеннона−Фано. При этом таблица отсортированных символов строится:

а) на основе данных, полученных в лабораторной работе № 2;

б) динамически, на основе анализа сжимаемого сообщения.

3. Определить эффективность (в сравнении с кодами ASCII) сжатия сообщения.

4. Результаты оформить в виде отчета по установленным правилам.

**Выполнение работы:**

Для начала определим в нашем вспомогательном классе InfoMetrics две структуры данных: Map<Character, String> charPrefixHashMap и Map<Character, Double> frequency, которые будут выступать «словарём», где ключом у обоих будет – символ, а значением – закодированная бинарная последовательность и частота встречи символа в тексте соответственно. Для построения дерева нам понадобится рекурсивная функция, представленная на рисунке 1. Она будет пошагово добавлять бит значения для каждого нашего элемента.

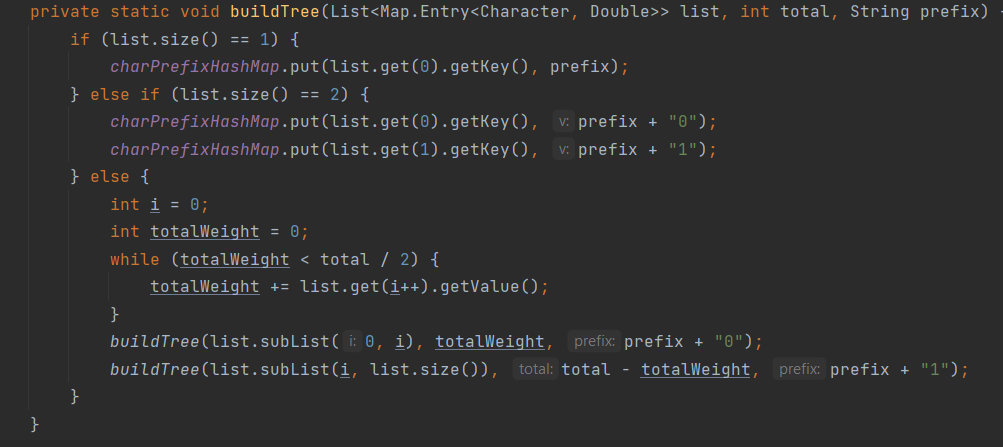


Рисунок 1 – Функция сжатия сообщения

Далее для получения частот и вывода вероятности определим функцию расчёта вероятности на основе данного входного сообщения и построения дерева с использованием функции buildTree, которое мы описали шагом ранее. Полный код данной функции можно найти в листинге 1, приложение А. Имеем следующий результат – рисунок 2.

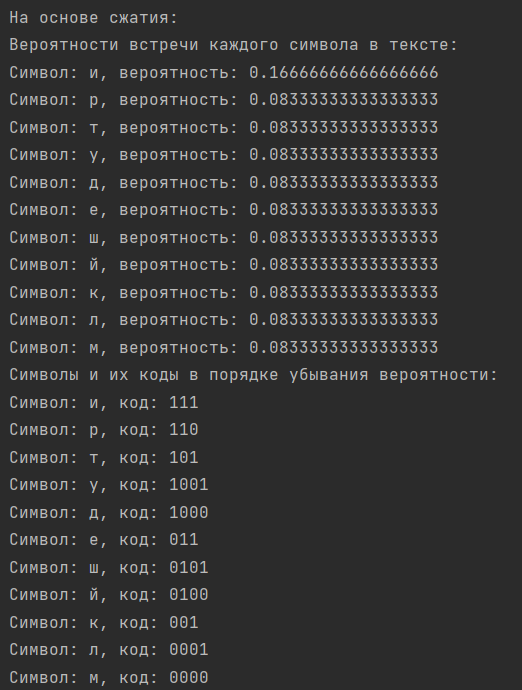


Рисунок 2 – Функция кодирования исходного сообщения

Можем сделать вывод, что для кодирования одного символа нам понадобится 4 бита.

Следующим шагом определим функции, которые на основе нашей структуры данных, содержащей соответствие бинарного кода символу, закодируют и раскодируют сообщение – рисунок 3.

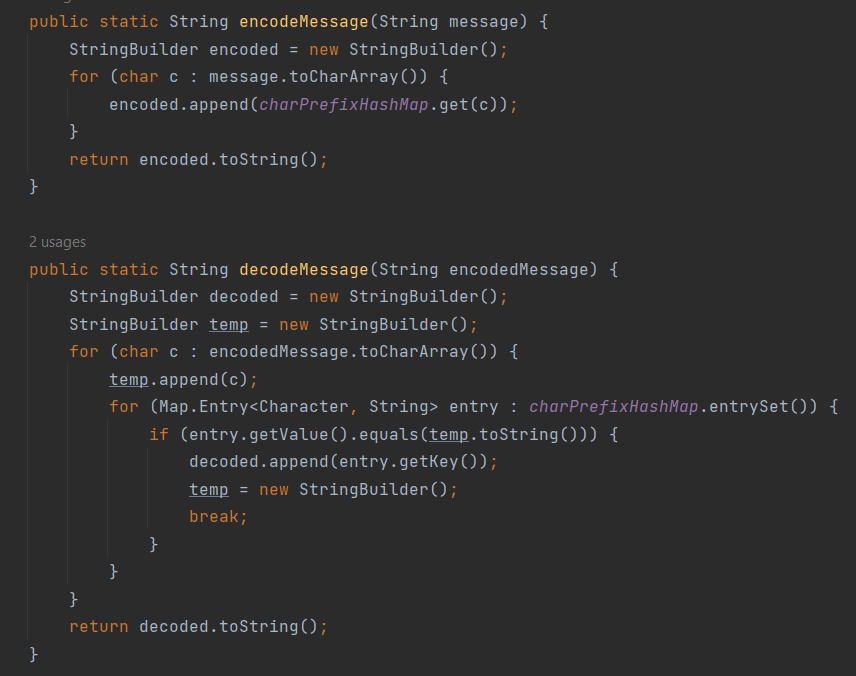


Рисунок 3 – Функция генерации матрицы перемежения

Результатом выполнения прямого и обратного преобразования будет следующий вывод – рисунок 4.



Рисунок 4 – Результат прямого и обратного преобразования

Как мы видим, с помощью данного дерева мы успешно провели преобразование в обе стороны с использованием методом Шеннона – Фано.

Теперь же необходимо использовать результат лабораторной работы №2. За основу текста будем использовать случайный отрывок длиной 86 символов из произведения «Война и Мир». Поместим количество совпадений с интересующими нас символами из строки «дмитрийлешук» – рисунок 5.

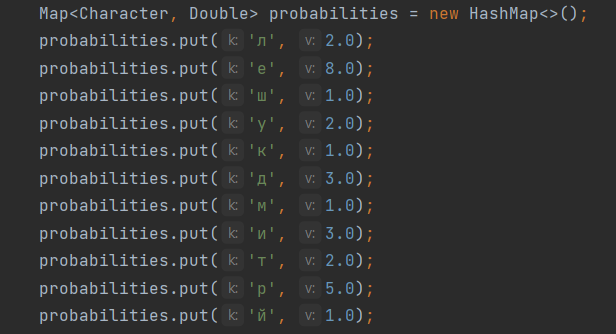


Рисунок 5 – Результат обратного преобразования

Определим функцию setFrequencyAndCompute, которая будет заменять в нашей структуре Map вероятности попадания каждого символа и скорректируем там длину текста и количество найденных символов. Данная функция расположена в приложении А, листинг 1. Получаем следующий результат – рисунок 6. Как мы видим, в данном случае у нас некоторые символы представляются уже пятью битами, что понижает степень сжатия. Результат кодирования и декодирования представлен на рисунке 7.

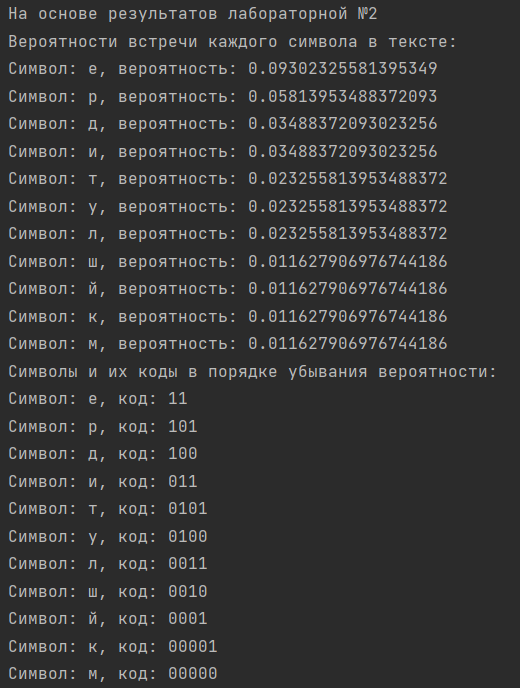


Рисунок 6 – Результат на основе вероятностей из лабораторной №2



Рисунок 7 – Результат прямого и обратного преобразования

Теперь сравним эффективность сжатия. В ASCII один символ – 8 бит информации. Следовательно, входное сообщение имеет объём 96 бит. В первом случае имеем: , во втором случае . В первом случае сообщение после сжатия занимает половину объёма сообщения до сжатия – в два раза эффективнее, чем ASCII.

**Вывод:** В процессе выполнения данной лабораторной работы были получены теоретические знания статистических методов сжатия, приобретение практических навыков использования статистических методов Шеннона−Фано и Хаффмана для сжатия/распаковки данных, исследована эффективность сжатия данного метода по сравнению со стандартным представлением символов – ASCII.

**Приложение А**

import java.util.\*;

public class InfoMetrics {

// Функция для прямого преобразования Барроуза-Уилера

public static String[] bwt(String input) {

int length = input.length();

ArrayList<String> table = new ArrayList<>();

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

table.add(input.substring(length - i) + input.substring(0, length - i));

}

System.out.println("Таблица циклических сдвигов:");

for (String row : table) {

System.out.println(row);

}

Collections.sort(table);

System.out.println("\nОтсортированная таблица:");

for (String row : table) {

System.out.println(row);

}

StringBuilder lastColumn = new StringBuilder(length);

for (String row : table) {

lastColumn.append(row.charAt(length - 1));

}

int originalRow = table.indexOf(input);

return new String[]{lastColumn.toString(), Integer.toString(originalRow)};

}

// Функция для обратного преобразования Барроуза-Уилера

public static String invBwt(String lastColumn, int originalRow) {

int length = lastColumn.length();

ArrayList<String> table = new ArrayList<>(Collections.nCopies(length, ""));

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

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

table.set(j, lastColumn.charAt(j) + table.get(j));

}

Collections.sort(table);

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

for (String row : table) {

System.out.println(row);

}

System.out.println();

}

return table.get(originalRow);

}

}

Листинг 1 – Полный код вспомогательног о класса

import java.util.Arrays;

public class Main {

public static void main(String[] args) {

String input = "достопримечательность";

//String input\_bin = "110001001100111011100001";

//System.out.println("Бинарное сообщение на входе = " + input\_bin);

// Измерение времени выполнения прямого преобразования

long startTime = System.nanoTime();

String[] bwtResult = InfoMetrics.bwt(input);

long endTime = System.nanoTime();

System.out.println("Время выполнения прямого преобразования: " + (endTime - startTime) / 1000000 + " миллисекунд");

System.out.println("Результат BWT: " + Arrays.toString(bwtResult));

// Измерение времени выполнения обратного преобразования

startTime = System.nanoTime();

String invBwtResult = InfoMetrics.invBwt(bwtResult[0], Integer.parseInt(bwtResult[1]));

endTime = System.nanoTime();

System.out.println("Время выполнения обратного преобразования: " + (endTime - startTime) / 1000000 + " миллисекунд");

System.out.println("Результат обратного преобразования: " + invBwtResult);

}

}

Листинг 2 – Код функции выполнения программы