Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ

УНИВЕРСИТЕТ»

«Защита информации и надёжность информационных систем»

Отчёт по лабораторной работе №10

СЖАТИЕ/РАСПАКОВКА ДАННЫХ АРИФМЕТИЧЕСКИМ МЕТОДОМ

Выполнил: Лешук Д. И.

ФИТ 3 курс 7 группа

Преподаватель: Николайчук А.Н

Минск 2024

**Цель:** приобретение практических навыков использования арифметических методов сжатия/распаковки данных.

**Практическое задание:**

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

2. С помощью приложения выполнить прямое и обратное преобразования сообщений в соответствии с таблицей.

Каждый студент выполняет задание, состоящее из двух частей. Первая часть предусматривает кодирование/декодирование сообщения «достопримечательность», вторая часть – составного сообщения, полученного конкатенацией последовательностей из слов «достопримечательность» и «сорокадневный»

3. Дать оценку возможности переполнения при выполнении вычислений.

4. Сравнить характеристики арифметического сжатия с вероятностными алгоритмами.

5. Результаты оформить в виде отчета по установленным правилам.

**Выполнение работы:**

Определим нужные нам функции для реализации арифметического сжатия – calculateInitialRanges, performArithmeticEncoding, performArithmeticDecoding – которые будут выполнять начальный расчёт вероятностей на основе входного слова, кодировать и декодировать соответственно. Первая функция была разработана на основе лабораторной работы №2, а те, которые непосредственно отвечают за сжатие и распаковку – на основе методического пособия. Полный код программы представлен в приложении А. Наше входное слово «достопримечательность» имеет довольно большую длину и символы в нём почти не повторяются. Предварительно, у нас должен быть 21 шаг – равен длине нашего слова. Для начала определим вероятности каждого символа и начальный диапазон значений – рисунок 1.

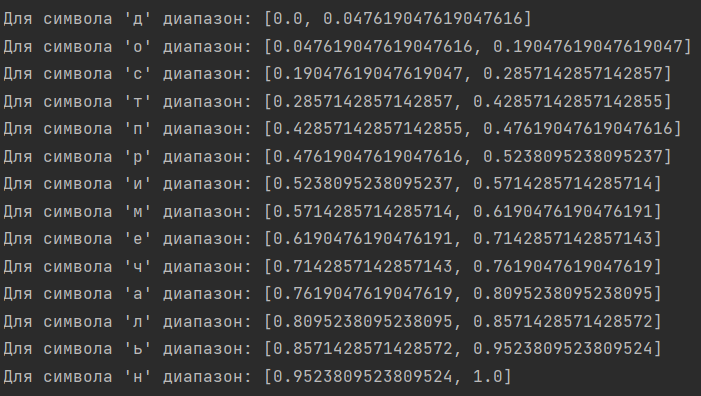


Рисунок 1 – Начальный диапазон значений

Как мы видим, начальный диапазон построен корректно – общая вероятность – от 0 до 1, повторяющихся символов в алфавите нету. Далее перейдём к выполнению функции кодирования. Во избежание вывода более двухсот строк, будем выводить только результат разбиение диапазона по конкретному символу – рисунок 2. Проблемы начинаются с 15-ого шага. Значения начинают повторяться – получилось переполнение числа Double, несмотря на то, что этот тип данных хранит 16 знаков после запятой.

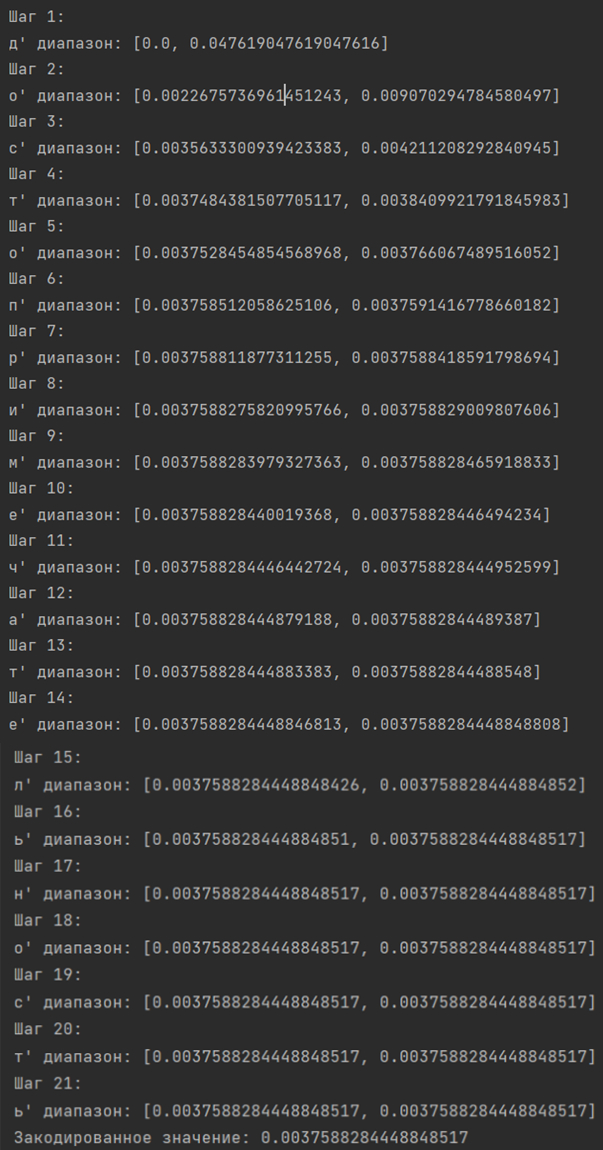


Рисунок 2 – Результат кодирования

Смотря на этот результат, уже можно сделать вывод, что следующее наше слово «достопримечательностьсорокадневный» тоже приведёт к переполнению

Однако, посмотрим на результат декомпрессии – рисунок 3.



Рисунок 3 – Функция генерации матрицы перемежения

Из-за переполнения мы не можем чётко определить несколько последних букв, из-за того, что они имеют одинаковое значение, их идентифицирует некорректно. Получается, данное слово невозможно сжать и распаковать без потерь. Имеем 16 правильных символов и 5 ошибочных – коэффициент ошибок 32%.

Теперь проверим наше второе входное слово, которое состоит из 34 символов. Значения здесь стали повторяться уже на 15 шаге применения алгоритма. Взглянем же на декодированный вариант – рисунок 4.



Рисунок 4 – Второе слово после декодирования

Если мы ещё можем догадаться, что первая часть слова – «достопримечательность», то на «сорокадневный» нету даже и намёка. Имеем 19 корректно распакованных символа и 15 ошибочных. Коэффициент ошибки равен приблизительно 75%. В данном примере переполнение ещё более выражено.

Однако стоит отметить, что данный алгоритм быстрее выполнил процесс кодирования и декодирования, чем вероятностные. Это можно объяснить тем, что здесь не используются операции взаимодействия с матрицами – транспонирование, умножение, сложение.

**Вывод:** В ходе лабораторной работы был протестирован метод арифметический метод сжатия и распаковки, который относительно быстро справляется с поставленной задачей, однако при его использовании необходимо следить за переполнением, особенно в неоднородных входных сообщениях большой длины.

**Приложение А**

import java.util.\*;

package org.example;

import java.util.LinkedHashMap;

import java.util.Map;

import java.util.Set;

import java.util.LinkedHashMap;

import java.util.Map;

public class InfoMetrics {

private static Map<Character, Double[]> ranges = new LinkedHashMap<>();

public static void main(String[] args) {

String word = "достопримечательность";

calculateInitialRanges(word);

double encodedValue = performArithmeticEncoding(word);

System.out.println("Закодированное значение: " + encodedValue);

String decodedWord = performArithmeticDecoding(encodedValue, word.length());

System.out.println("Декодированное слово: " + decodedWord);

}

private static void calculateInitialRanges(String word) {

Map<Character, Double> probabilities = new LinkedHashMap<>();

double totalLength = word.length();

// Вычисляем вероятности для каждого символа

for (char ch : word.toCharArray()) {

probabilities.put(ch, probabilities.getOrDefault(ch, 0.0) + 1);

}

probabilities.forEach((k, v) -> probabilities.put(k, v / totalLength));

// Инициализируем начальные границы

double lower = 0.0;

double upper;

// Вычисляем диапазоны для каждого символа

for (Map.Entry<Character, Double> entry : probabilities.entrySet()) {

upper = lower + entry.getValue();

ranges.put(entry.getKey(), new Double[]{lower, upper});

lower = upper;

}

System.out.println("Шаг 0");

ranges.forEach((k, v) -> System.out.println("Для символа '" + k + "' диапазон: [" + v[0] + ", " + v[1] + "]"));

}

private static double performArithmeticEncoding(String word) {

double lower = 0.0;

double upper = 1.0;

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

char ch = word.charAt(i);

Double[] currentRange = ranges.get(ch);

double range = upper - lower;

// Вычисляем новые границы для текущего символа

double newLower = lower + range \* currentRange[0];

double newUpper = lower + range \* currentRange[1];

// Обновляем рабочий диапазон

lower = newLower;

upper = newUpper;

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

System.out.println(ch + "' диапазон: [" + newLower + ", " + newUpper + "]");

}

// Возвращаем среднее значение между верхней и нижней границей как закодированное значение

return (lower + upper) / 2.0;

}

private static String performArithmeticDecoding(double code, int length) {

StringBuilder word = new StringBuilder();

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

// Находим символ, соответствующий текущему коду

double finalCode = code;

char ch = ranges.entrySet().stream()

.filter(entry -> entry.getValue()[0] <= finalCode && finalCode < entry.getValue()[1])

.findFirst()

.map(Map.Entry::getKey)

.orElseThrow(() -> new RuntimeException("Не удалось декодировать символ"));

word.append(ch);

// Обновляем код для следующего символа

Double[] currentRange = ranges.get(ch);

code = (code - currentRange[0]) / (currentRange[1] - currentRange[0]);

}

return word.toString();

}

}

Листинг 1 – Полный код приложения