Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ

УНИВЕРСИТЕТ»

«Защита информации и надёжность информационных систем»

Отчёт по лабораторной работе №7

СЖАТИЕ/РАСПАКОВКА ДАННЫХ МЕТОДОМ БАРРОУЗА − УИЛЕРА

Выполнил: Лешук Д. И.

ФИТ 3 курс 7 группа

Преподаватель: Николайчук А.Н

Минск 2024

**Цель:** приобретение практических навыков использования метода Барроуза − Уилера для сжатия/распаковки данных.

**Практическое задание:**

1. Разработать авторское приложение в соответствии с целью лабораторной работы. Входной блок данных может иметь произвольную длину.

2. С помощью приложения выполнить прямое и обратное преобразования 3 отдельных блоков данных, состоящих:

а) из собственного имени (можно краткий вариант записи);

б) собственной фамилии;

в) варианта 10 в соответствии с таблицей учебно-методического пособия – «достопримечательность».

Можно использовать любой из известных методов сортировки символов массива.

Выполнить качественный сравнительный анализ длительности процессов прямого и обратного преобразований в зависимости от длины блока данных

3. Перевести первые 3 символа из блока данных, указанного в варианте таблицы, в бинарную последовательность в соответствии с кодами ASCII. Выполнить прямое и обратное преобразование. Оценить время прямого и обратного преобразований.

4. Результаты оформить в виде отчета по установленным правилам.

**Выполнение работы:**

Определим функцию, которая будет выполнять сжатие входного сообщения на основании алгоритма BWT – рисунок 1. В соответствии с алгоритмом, имеем 4 шага: определение блока данных (передаём через параметр) составление матрицы W на основе входного сообщения, лексикографическая сортировка строк, формирование выходного сообщения, попутно выводим промежуточные результаты выполнения. Для сортировки строк используем готовое решение – метод Collections.sort().

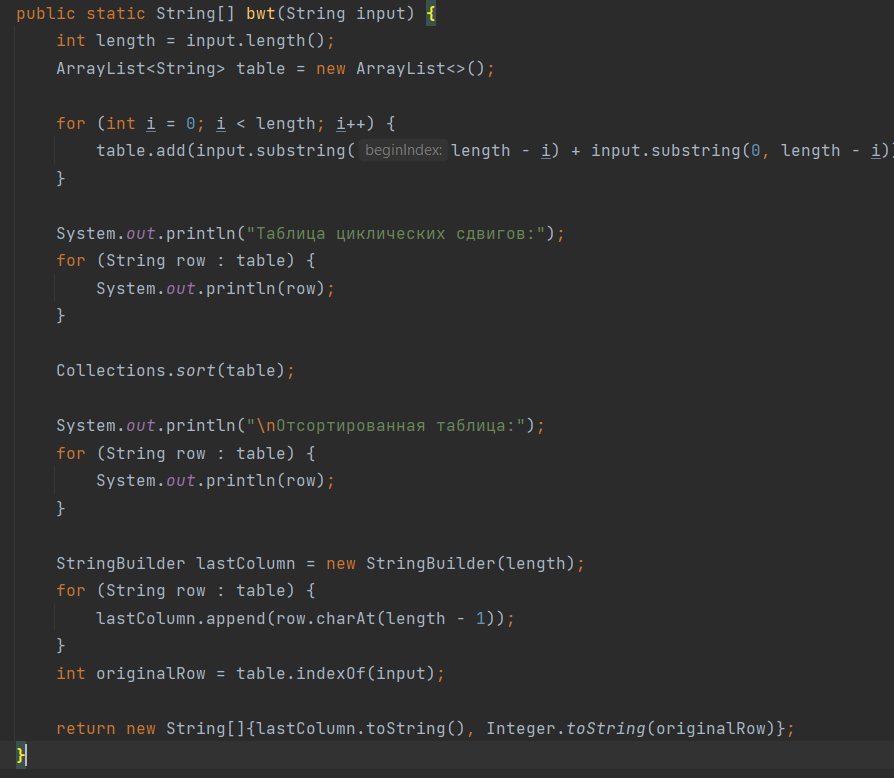


Рисунок 1 – Функция сжатия сообщения

Далее определяем функцию обратного преобразования – рисунок 2.

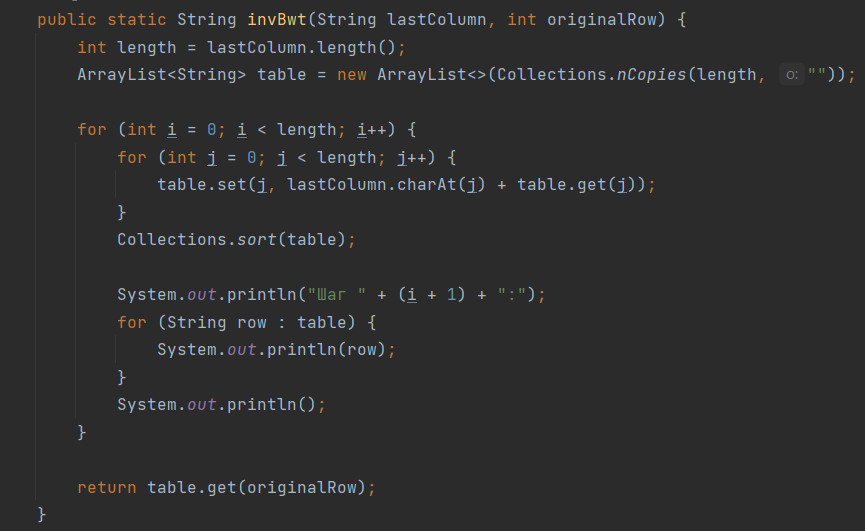


Рисунок 2 – Функция кодирования исходного сообщения

В этой функции мы производим восстановление таблицы циклических сдвигов по имеющемуся у нас сжатому сообщению, а затем и возвращаем наше входное слово.

Далее необходимо выполнить прямое и обратное преобразование для разных входных последовательностей. Первое сообщение – краткий вариант записи имени, «дима». На рисунке 3 представлен результат формирования матрицы W и сжатого сообщения.

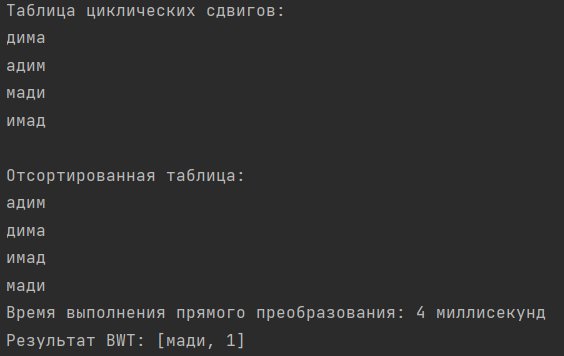


Рисунок 3 – Функция генерации матрицы перемежения

В дополнение, как мы можем видеть, потратили на данную операцию 4 миллисекунды. В результате имеем номер строки, совпадающий с входным сообщением равный 1, а не 2. Происходит это из-за того, что нумерация элементов массивов начинается с 0, а не с 1 – индекс. Результат обратного преобразования представлен на рисунке 4. Как мы видим, время выполнения составило у нас 17 миллисекунд – обратное преобразование длилось в 4 раза дольше, чем прямое.

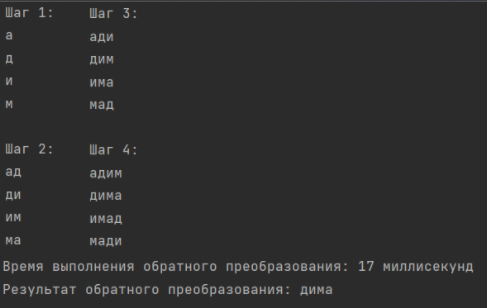


Рисунок 4 – Результат обратного преобразования

Следующее входное сообщение – фамилия, «лешук». Проверим, будет ли отличаться время выполнения при увеличении длины сообщения на 1 символ – рисунок 5, 6.

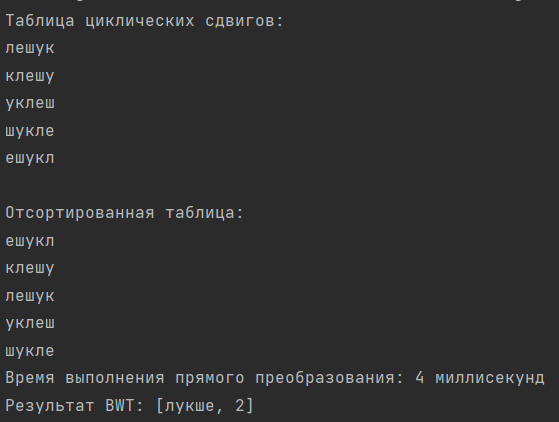


Рисунок 5 – Прямое BWT преобразование фамилии

Как мы видим, время прямого преобразования не поменялось, значит лдин символ не оказывает слиьного влияния на скорость выполнения. А вот при обратном преобразовании время выполнения уже слегка увеличилось.

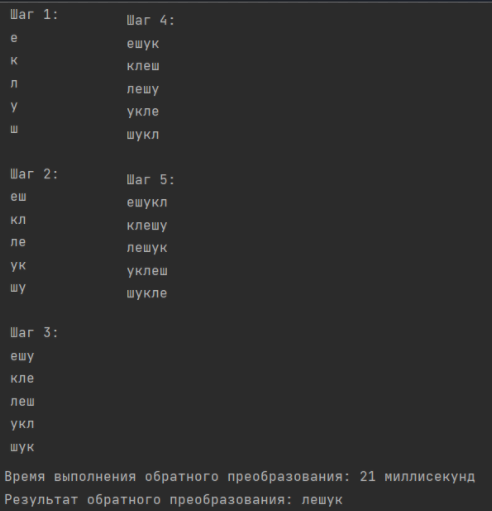


Рисунок 6 – Обратное преобразование фамилии из сжатого формата

Далее на входе – слово «достопримечательность». На рисунке 7 представлен результат прямого преобразования – сжатия.

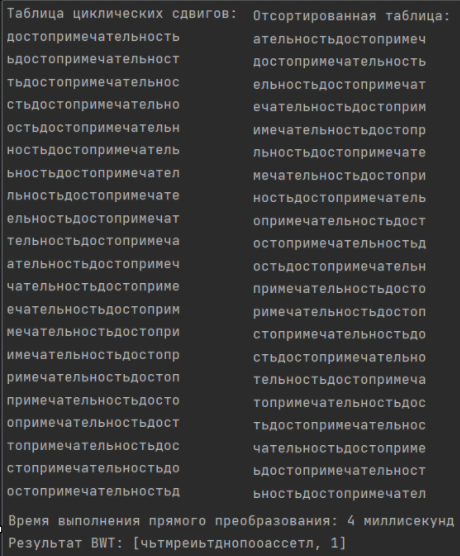


Рисунок 7 – Сжатие сообщения «достопримечательность»

Время прямого преобразования осталось неизменным. А вот количество шагов для построения матрицы значительно увеличилось – их стало 21 (так как длина входного сообщения = 21). Вот что имеем в результате обратного преобразования – рисунок 8.

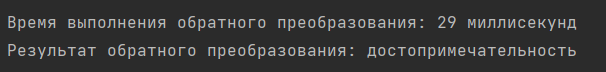


Рисунок 8 – Обратное преобразование в слово «достопримечательность»

Следующим шагом необходимо преобразовать «дос» – первые три буквы слова из варианта в бинарный вид по ASCII. Входное сообщение будет выглядеть так: «110001001100111011100001». Результат прямого преобразования – рисунок 9. Время незначительно увеличилось – на 1 мс.

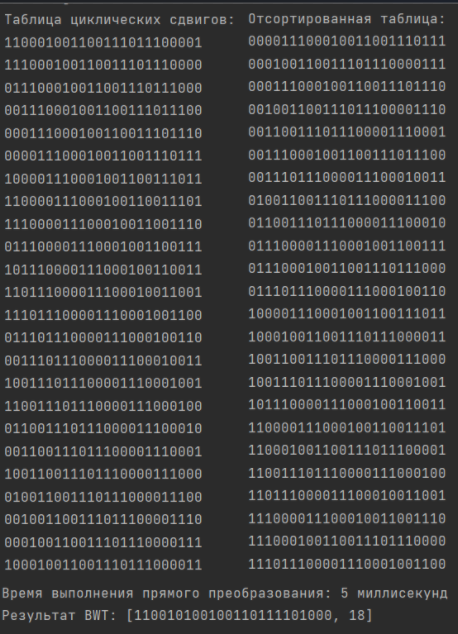


Рисунок 8 – Обратное преобразование в слово «достопримечательность»

Так как длина сообщения в данном случае равна 24, то для формирования матрицы циклических сдвигов по сжатому сообщению этот этап будет происходить в 24 шага. Результат обратного преобразования представлен на рисунке 9.

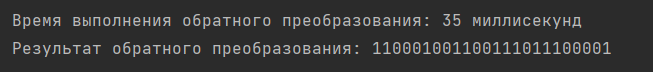


Рисунок 9 – Результат обратного преобразования двоичного сообщения

Исходя из наблюдений, затраты времени на обратное преобразование с возрастанием длины сообщения увеличивается стремительнее, чем время прямого преобразования.

**Вывод:** В процессе выполнения данной лабораторной работы были получены теоретические знания сжатия и распаковки данных методом Барроуза-Уилера, получены практические навыки написания функций прямого и обратного преобразования на языке Java, а также исследована зависимость времени выполнения преобразований от длины сообщения.

**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