**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

по дисциплине «Алгоритмы и структуры данных»

на тему: «Архивация данных»

Выполнил: студент гр. ИП-31

Казутин П. Н.

Принял: ст. преподаватель

Косинов Г. П.

Гомель 2021

**Цель работы:** изучить основные методы архивации, сравнить их эффективность.

**Практическая часть:**

Реализовать алгоритмы сжатия согласно варианту. Определить коэффициенты сжатия и время архивации.

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант + доп требования | | Алгоритм 1 | Алгоритм 2 |
| 9 | С паролем | Арифметическое сжатие | RLE+BWT |

***package by.ktsin.compress.BWTRLE;***

public final class **BWT** {

public static String encode(String rawString) {

String[] strs = new String[rawString.length()];

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

strs[i] = rawString.substring(i) + rawString.substring(0, i);

}

java.util.Arrays.sort(strs);

StringBuilder builder = new StringBuilder();

for (String str : strs) {

builder.append(str.charAt(str.length() - 1));

}

return builder.toString();

}

private static final class BWTComparator implements java.util.Comparator<Integer> {

private final String string;

BWTComparator(String string) {

this.string = string;

}

@Override

public int compare(Integer i, Integer j) {

return string.charAt(i) - string.charAt(j);

}

public boolean equals(Integer i, Integer j) {

return string.charAt(i) == string.charAt(j);

}

}

public static String decode(String encodedString) {

Integer[] indices = new Integer[encodedString.length()];

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

indices[i] = i;

}

java.util.Arrays.sort(indices, new BWTComparator(encodedString));

int startIndex = 0;

for (; encodedString.charAt(startIndex) != '$'; ++startIndex);

StringBuilder builder = new StringBuilder();

for (int i = 0; i < indices.length - 1; ++i) {

startIndex = indices[startIndex];

char c = encodedString.charAt(startIndex);

builder.append(c);

}

return builder.toString();

}

public static void main(String[] args) {

String encodedString = BWT.encode(args[0] + '$');

System.out.println(encodedString);

String decodedString = BWT.decode(encodedString);

System.out.println(decodedString);

}

}

***package by.ktsin.compress.BWTRLE;***

public final class **RLE** {

public static String encode(String chain) {

if (chain == null || chain.isEmpty())

return "";

if(chain.length()==1)return encodeUnit(chain);

StringBuilder prefix= new StringBuilder();

int i=0;

do {

prefix.append(chain.charAt(i));

i++;

} while (i<chain.length()&& chain.charAt(i-1)==chain.charAt(i));

if(i<chain.length()&&chain.charAt(i-1)!=chain.charAt(i)){

return encodeUnit(prefix.toString())+encode(chain.substring(prefix.length()));

}

return encodeUnit(chain);

}

private static String encodeUnit(String chain) {

return "" + chain.length() + chain.charAt(0);

}

public static String decode(String encodedString) {

String decodedString = null;

//aaabbbcccccdd

//3a3b5c2d

int n = encodedString.length();

StringBuilder sb= new StringBuilder();

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

if(i+1 <n && i%2 ==0)

sb.append(repeat(Integer.parseInt(String.valueOf(encodedString.charAt(i))),encodedString.charAt(i+1)));

}

return sb.toString();

}

private static String repeat(int length, char charAt) {

StringBuilder sb = new StringBuilder();

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

sb.append(charAt);

}

return sb.toString();

}

public static void main(String[] args){

System.out.println(args[0]);

String encoded = encode(args[0]);

System.out.println(encoded);

System.out.println(decode(encoded));

}

}

***package by.ktsin.compress;***

import by.ktsin.compress.BWTRLE.BWT;

import by.ktsin.compress.BWTRLE.RLE;

public class **BWTRLECompresser** implements Compresser{

@Override

public String compress(String input) {

String res = BWT.encode(input + "$");

res = RLE.encode(res);

return res;

}

@Override

public String decompress(String input) {

String res = RLE.decode(input);

res = BWT.decode(res);

return res;

}

public static void main(String[] args){

BWTRLECompresser c = new BWTRLECompresser();

System.out.print("Input: ");

System.out.print(args[0] + "\n");

System.out.print("Compressed: ");

String t = c.compress(args[0]);

System.out.print(t + "\n");

System.out.print("Decompress: ");

t = c.decompress(t);

System.out.print(t + "\n");

}

}

***package by.ktsin.compress.arithmetical;***

import java.nio.ByteBuffer;

import java.nio.charset.StandardCharsets;

import java.util.\*;

public class **ArithmeticCoding** {

private TreeMap<Character, Double> probabilities = new TreeMap<>();

private TreeMap<Character, Double> right\_interval = new TreeMap<>();

private TreeMap<Character, Double> left\_interval = new TreeMap<>();

public byte[] encode(String str) {

// строим модель

TreeMap<Character, Integer> count = new TreeMap<>();

int length = str.length();

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

if (count.containsKey(c)) {

count.put(c, count.get(c) + 1);

} else

count.put(c, 1);

}

for (Map.Entry<Character, Integer> c : count.entrySet()) {

probabilities.put(c.getKey(), (c.getValue() / (double) length));

}

Object[] sorted = probabilities.entrySet().stream().sorted(new EntrySetComparer()).toArray();

double max = 0;

double min = 0;

for (int i = sorted.length - 1; i >= 0; i--) {

Map.Entry<Character, Double> t = (Map.Entry<Character, Double>) sorted[i];

max += t.getValue();

right\_interval.put(t.getKey(), max);

left\_interval.put(t.getKey(), min);

min += t.getValue();

}

// разбиваем на блоки по 5 символов

Object[] blocks = Arrays.stream(str.split("(?<=\\G.{9})")).map(e -> String.format(e, "%-9s")).toArray();

ArrayList<Double> res = new ArrayList<>();

for(Object block : blocks){

double a = encodeBlock((String)block);

res.add(a);

}

res = res;

int block = 0;

byte[] bytes = new byte[res.size() \* Double.BYTES];

int num = 0;

for(Double r : res){

ByteBuffer byteBuffer = ByteBuffer.allocate(Double.BYTES);

byteBuffer.putDouble(r);

for(byte b : byteBuffer.array()){

num++;

}

block++;

}

return bytes;

}

private double encodeBlock(String block) {

double res = 0;

double p\_left = 0;

double p\_right = 0;

double low = 0;

double high = 0;

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

if (res == 0) {

res += right\_interval.get(c);

p\_left = left\_interval.get(c);

p\_right = right\_interval.get(c);

} else {

low = p\_left + (p\_right - p\_left) \* left\_interval.get(c);

high = p\_left + (p\_right - p\_left) \* right\_interval.get(c);

res = low;

p\_left = low;

p\_right = high;

}

res += 0.0;

}

return res;

}

private class EntrySetComparer implements Comparator<Map.Entry<Character, Double>> {

public int compare(Map.Entry<Character, Double> o1, Map.Entry<Character, Double> o2) {

return Double.compare(o1.getValue(), o2.getValue());

}

}

public static void main(String[] args) {

ArithmeticCoding a = new ArithmeticCoding();

String str = "Big fucking string for testing purposes";

System.out.print("Source: " + args[0]);

a.encode(args[0]);

}

}