МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ   
РОССИЙСКОЙ ФЕДЕРАЦИИ  
  
Федеральное государственное автономное образовательное учреждение   
высшего образования   
«Самарский национальный исследовательский университет   
имени академика С.П. Королёва» (Самарский университет)  
  
Факультет информатики  
Кафедра программных систем  
  
Дисциплина  
**Теория информации**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

**Сжатие информации**

Вариант №7

Студент: Гижевская В.Д.

Группа: 6413-020302D   
  
Преподаватель: Додонов М.В  
Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Самара 2021

**ЗАДАНИЕ**

1. Сжать сообщение методом Хаффмана с блокированием. Вычислить EX, ML(X), ML(Xбл).

Здесь:

* EX–энтропия алфавита из букв сообщения,
* ML(X) –среднее количество элементарных символов на букву при сжатии методом Хаффмана,
* ML(Xбл) –среднее количество элементарных символов на букву при сжатии методом Хаффмана с блокированием.

1. Сжать сообщение адаптивным методом Хаффмана.
2. Сжать сообщение методами LZ77, LZSS, LZ78. Размер словаря –8 символов, буфера –5 символов. В некоторых вариантах в скобках указаны иные размеры словаря и буфера, относящиеся только к этим вариантам!
3. Сжать сообщение из задания №2 (только 10 букв!) арифметическим методом.
4. Распаковать сообщения, сжатые адаптивным методом Хаффмана и методами LZ77, LZSS, LZ78 и арифметическим методом. Задания для раскодировки метода Хаффмана, LZ77 и LZSS приведены ниже в таблицах. Арифметическим методом раскодировать своё закодированное слово. Для декодирования LZ78 обменяться кодами с одногруппником.

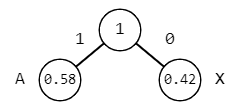
**ХОД РАБОТЫ**

**Задание 1**

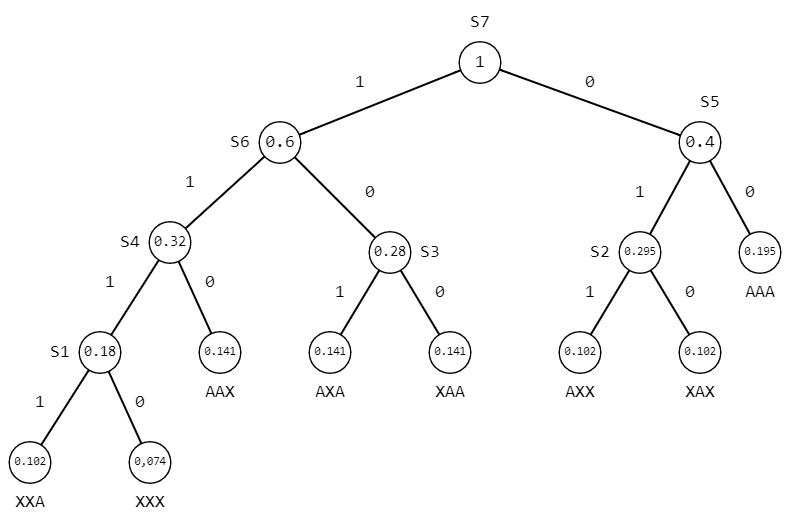
Сообщение: XAAAAXXAXAAX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество | Xi | Код | pi | ki |
| 7 | A | 1 | 0.58 | 1 |
| 5 | X | 0 | 0.42 | 1 |

Энтропия алфавита из букв сообщения:



Среднее количество элементарных символов на букву при сжатии методом Хаффмана:

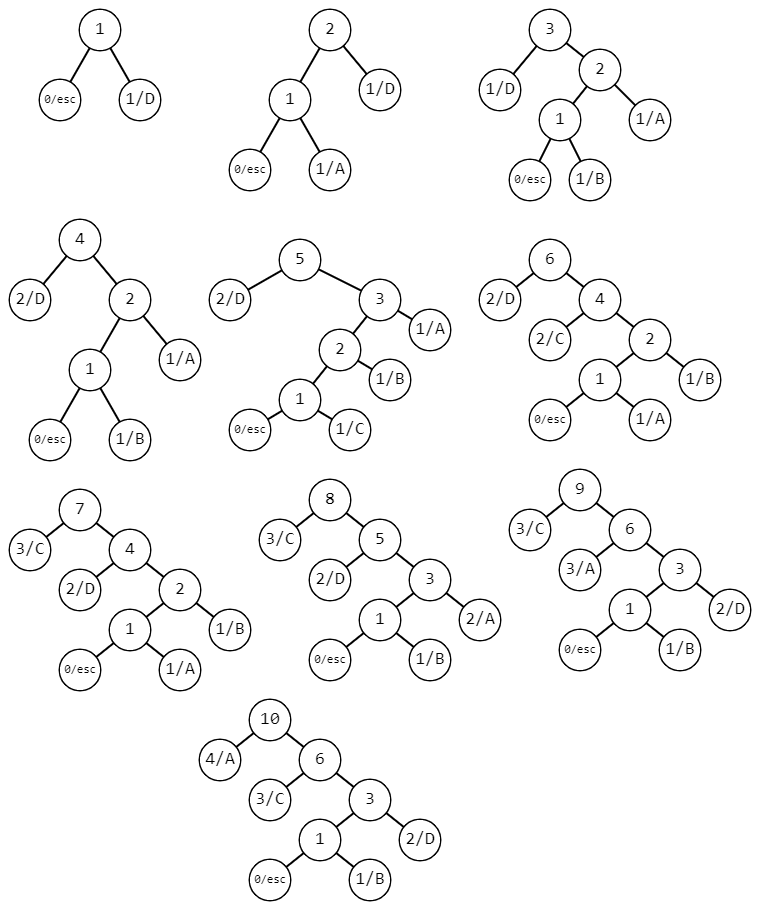
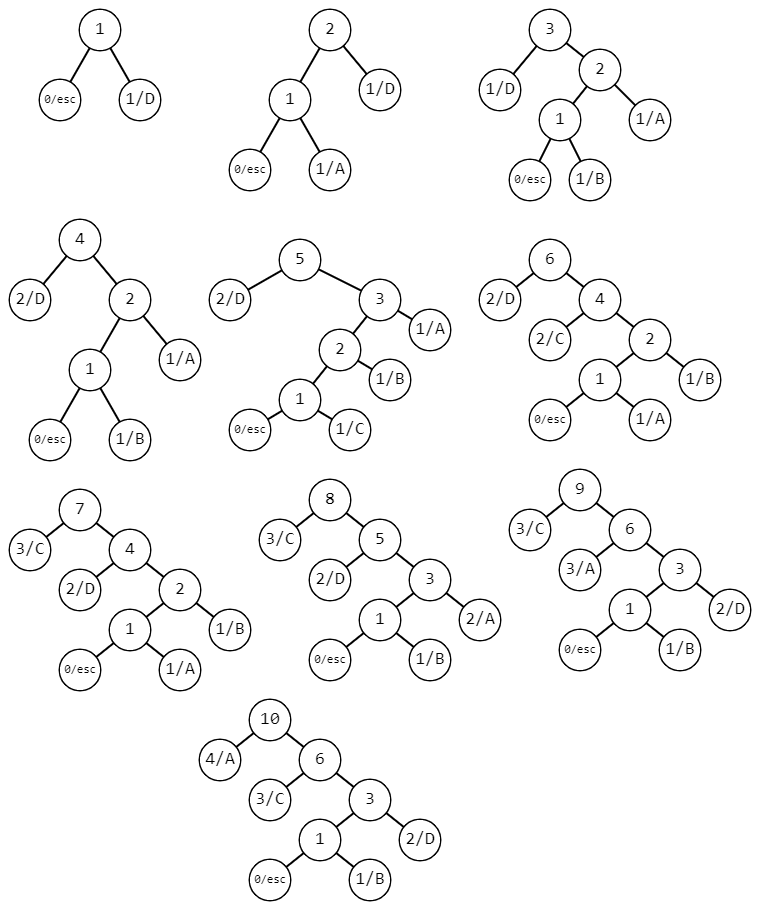


|  |  |  |  |
| --- | --- | --- | --- |
| Xi | pi | Код | ki |
| AAA | 0.195 | 00 | 2 |
| AAX | 0.141 | 110 | 3 |
| AXA | 0.141 | 101 | 3 |
| AXX | 0.102 | 011 | 3 |
| XAA | 0.141 | 100 | 3 |
| XAX | 0.102 | 010 | 3 |
| XXA | 0.102 | 1111 | 4 |
| XXX | 0.195 | 1110 | 4 |

Среднее количество элементарных символов на букву при сжатии методом Хаффмана с блокированием:

**Задание 2**

Сообщение: DABDCCCAAA



|  |  |  |  |
| --- | --- | --- | --- |
| Вх. данные | Вых. данные | № дерева | Длина кода, бит |
| D | ‘D’ | 1 | 8 |
| A | 0’A’ | 2 | 9 |
| B | 00’B | 3 | 10 |
| D | 0 | 4 | 1 |
| C | 100’C’ | 5 | 11 |
| C | 1101 | 6 | 4 |
| C | 10 | 7 | 2 |
| A | 1101 | 8 | 4 |
| A | 111 | 9 | 3 |
| A | 10 | 10 | 2 |

**Задание 3 – LZ77**

Сообщение: ЗАБОР ЗАБОРИСТЫЙ

Размер словаря - 8, размер буфера - 6

|  |  |  |
| --- | --- | --- |
| Словарь (8) | Буфер (6) | Код |
| «\_ \_ \_ \_ \_ \_ \_ \_» | «ЗАБОР » | <0, 0, ‘З’> |
| «\_ \_ \_ \_ \_ \_ \_ З» | «АБОР З» | <0, 0, ‘А’> |
| «\_ \_ \_ \_ \_ \_ ЗА» | «БОР ЗА» | <0, 0, ‘Б’> |
| «\_ \_ \_ \_ \_ ЗАБ» | «ОР ЗАБ » | <0, 0, ‘О’> |
| «\_ \_ \_ \_ ЗАБО» | «Р ЗАБО» | <0, 0, ‘Р’> |
| «\_ \_ \_ ЗАБОР» | « ЗАБОР» | <0, 0, ‘ ’> |
| «\_ \_ЗАБОР » | «ЗАБОРИ» | <2, 5, ‘И’> |
| «Р ЗАБОРИ» | «СТЫЙ\_ \_» | <0, 0, ‘С’> |
| «\_ ЗАБОРИС» | «ТЫЙ\_ \_ \_» | <0, 0, ‘Т’> |
| «ЗАБОРИСТ» | «ЫЙ\_ \_ \_ \_» | <0, 0, ‘Ы’> |
| «АБОРИСТЫ» | «Й\_ \_ \_ \_ \_» | <0, 0, ‘Й’> |

На рисунке 1 представлен результат работы программы.

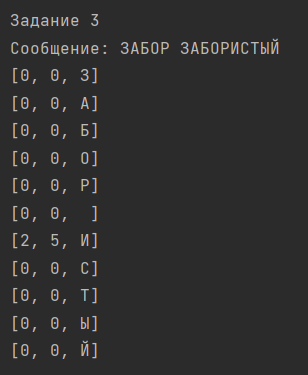


Рисунок 1 – Результат работы программы

**Задание 3 – LZSS**

Сообщение: ЗАБОР ЗАБОРИСТЫЙ

Размер словаря - 8, размер буфера - 6

|  |  |  |  |
| --- | --- | --- | --- |
| Словарь (8) | Буфер (6) | Код | Длина кода |
| «\_ \_ \_ \_ \_ \_ \_ \_» | «ЗАБОР » | 0, ‘З’ | 9 |
| «\_ \_ \_ \_ \_ \_ \_ З» | «АБОР З» | 0, ‘А’ | 9 |
| «\_ \_ \_ \_ \_ \_ ЗА» | «БОР ЗА» | 0, ‘Б’ | 9 |
| «\_ \_ \_ \_ \_ ЗАБ» | «ОР ЗАБ » | 0, ‘О’ | 9 |
| «\_ \_ \_ \_ ЗАБО» | «Р ЗАБО» | 0, ‘Р’ | 9 |
| «\_ \_ \_ ЗАБОР» | « ЗАБОР» | 0, ‘ ’ | 9 |
| «\_ \_ЗАБОР » | «ЗАБОРИ» | 1 <2, 5> | 7 |
| «Р ЗАБОРИ» | «СТЫЙ\_ \_» | 0, ‘С’ | 9 |
| «\_ ЗАБОРИС» | «ТЫЙ\_ \_ \_» | 0, ‘Т’ | 9 |
| «ЗАБОРИСТ» | «ЫЙ\_ \_ \_ \_» | 0, ‘Ы’ | 9 |
| «АБОРИСТЫ» | «Й\_ \_ \_ \_ \_» | 0, ‘Й’ | 9 |

**Задание 3 – LZ78**

Сообщение: ЗАБОР ЗАБОРИСТЫЙ

Размер словаря - 8, размер буфера - 6

|  |  |  |
| --- | --- | --- |
| Входная фраза | Код | Позиция словаря |
| «» |  | 0 |
| «З» | < 0, ‘З’ > | 1 |
| «А» | < 0, ‘А’ > | 2 |
| «Б» | < 0, ‘Б’ > | 3 |
| «О» | < 0, ‘О’ > | 4 |
| «Р» | < 0, ‘Р’ > | 5 |
| « » | < 0, ‘ ’ > | 6 |
| «ЗА» | < 1, ‘А’ > | 7 |
| «БО» | < 3, ‘О’ > | 8 |
| «РИ» | < 5, ‘И’ > | 9 |
| «С» | < 0, ‘С’ > | 10 |
| «Т» | < 0, ‘Т’ > | 11 |
| «Ы» | < 0, ‘Ы’ > | 12 |
| «Й» | < 0, ‘Й’ > | 13 |

**Задание 4**

Сообщение: DABDCCCAAA

A – 4 – 0.4

C – 3 – 0.3

D – 2 – 0.2

B – 1 – 0.1

Промежутки:

A [ 0 ; 0.4 ]

B [ 0.4 ; 0.7 ]

D [ 0.7 ; 0.9 ]

B [ 0.9 ; 1 ]

|  |  |  |  |
| --- | --- | --- | --- |
| Буква | Δ | min | max |
| D | 1 | 0,7 | 0,9 |
| A | 0,2 | 0,7 | 0,78 |
| B | 0,08 | 0,772 | 0,78 |
| D | 0,008 | 0,7776 | 0,7792 |
| C | 0,0016 | 0,77824 | 0,77872 |
| C | 0,00048 | 0,778432 | 0,778576 |
| C | 0,000144 | 0,7784896 | 0,7785328 |
| A | 0,0000432 | 0,7784896 | 0,77850688 |
| A | 0,00001728 | 0,7784896 | 0,778496512 |
| A | 0,000006912 | 0,7784896 | 0,7784923648 |

Закодированное сообщение: [ 0,7784896 ; 0,7784923648 ]

На рисунке 2 представлен результат работы программы.

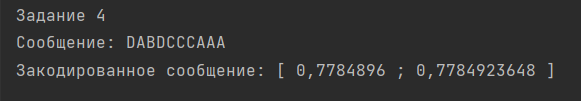


Рисунок 2 – Результат работы программы

**Задание 5 – адаптивный метод Хаффмана**

Закодированное сообщение: 'р'0'о'00'а'0100'п'1100'н'011111111101

|  |  |  |
| --- | --- | --- |
| Вх. данные | Вых. данные | № дерева |
| ‘Р’ | Р | 1 |
| 0’О’ | О | 2 |
| 00‘А’ | А | 3 |
| 01 | О | 4 |
| 00‘П’ | П | 5 |
| 1 | О | 6 |
| 1 | О | 7 |
| 00‘Н’ | Н | 8 |
| 01 | Р | 9 |
| 1 | О | 10 |
| 1 | О | 11 |
| 1 | О | 12 |
| 1 | О | 13 |
| 1 | О | 14 |
| 1 | О | 15 |
| 1 | О | 16 |
| 1 | О | 17 |
| 01 | Р | 18 |

Раскодированное сообщение: роаопоонроооооооор

**Задание 5 – LZ77**

Закодированное сообщение: <0,0,с> <0,0,о> <0,0,в> <0,0,а> <0,0, > <5,2,б> <5,1,к> <3,2,б> <5,2,е> <0,0,н> <4,3,к>

Размер словаря - 10

Раскодированное сообщение: сова собака бакен бак

**Задание 5 – LZSS**

Закодированное сообщение: [0'п'] [0'о'] [0'ж'] [0'а'] [0'р'] [0' '] [1<4,2>] [1<6,1>] [1<4,1>] [1<5,1>] [0'к'] [1<5,3>] [0'л'] [1<9,1>]

Размер словаря - 10

Раскодированное сообщение: пожар кора коралл

**Задание 5 – LZ78** (вариант 17)

Закодированное сообщение: <0, К><0, И><0, С><0, О><0, Н><0, Ь><0, К><0, И> <0, \_><1, И><0, С><0, -><1, И><0, С><1, И>

Размер словаря - 10

Раскодированное сообщение: КИСОНЬКИ КИС-КИСКИ

**Задание 5 – арифметический метод**

N: 0,7784896

Промежутки:

A [ 0 ; 0.4 ]

B [ 0.4 ; 0.7 ]

D [ 0.7 ; 0.9 ]

B [ 0.9 ; 1 ]

|  |  |  |
| --- | --- | --- |
| min | Буква | Δ |
| 0,7784896 | D | 0,2 |
| 0,392448 | A | 0,4 |
| 0,98112 | B | 0,1 |
| 0,8112 | D | 0,2 |
| 0,556 | C | 0,3 |
| 0,52 | C | 0,3 |
| 0,4 | C | 0,3 |
| 6,02111E-13 | A | 0,4 |
| 1,50528E-12 | A | 0,4 |
| 3,76319E-12 | A | 0,4 |

Раскодированное сообщение: DABDCCCAAA

**ЛИСТИНГ ПРОГРАММЫ**

package com.company;

import java.io.IOException;

import java.text.DecimalFormat;

import java.util.ArrayList;

import java.util.Collections;

public class Main {

public static void main(String[] args) throws IOException {

String hoffman = "XAAAAXXAXAAX";

String adapt = "DABDCCCAAA";

String methodLZ = "ЗАБОР ЗАБОРИСТЫЙ";

int sizeOfDictionary = 8;

int sizeOfBuffer = 6;

String decodAdapt = "'р'0'о'00'а'0100'п'1100'н'011111111101";

String decodLZ77 = "<0,0,с> <0,0,о> <0,0,в> <0,0,а> <0,0, > <5,2,б> <5,1,к> <3,2,б> <5,2,е> <0,0,н> <4,3,к>";

String decodLZSS = "[0'п'] [0'о'] [0'ж'] [0'а'] [0'р'] [0' '] [1<4,2>] [1<6,1>] [1<4,1>] [1<5,1>] [0'к'] [1<5,3>] [0'л'] [1<9,1>]";

//String decodLZ78 = "<0,0,с> <0,0,о> <0,0,в> <0,0,а> <0,0, > <5,2,б> <5,1,к> <3,2,б> <5,2,е> <0,0,н> <4,3,к>";

System.out.println("\nЗадание 1");

Hoffman.methodHoffman(hoffman);

System.out.println("\nЗадание 3");

methodLZ77(methodLZ,sizeOfDictionary,sizeOfBuffer);

System.out.println("\nЗадание 4");

methodArifm(adapt);

}

public static ArrayList<ArrayList<String>> searchChars(String text){

int numOfChars = text.length();

String chars = "";

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

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

int numOfCurrChar = 0;

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

char l = text.charAt(n);

String el = String.valueOf(l);

if (!chars.contains(el)){

chars = chars + el;

currChar.add(el);

for (int m = 0; m<numOfChars;m++){

if (text.charAt(n)== text.charAt(m)){

numOfCurrChar++;

}

}

currChar.add(Integer.toString(numOfCurrChar));

allChars.add(currChar);

}

}

ArrayList<String> a,b;

for (int i=0;i< allChars.size();i++){

for (int j=0;j<allChars.size();j++){

if (Integer.parseInt(allChars.get(i).get(1)) > Integer.parseInt(allChars.get(j).get(1))){

a = allChars.get(i);

b = allChars.get(j);

allChars.set(i,b);

allChars.set(j,a);

}

}

}

double startNum = 0;

double end = 1;

double dnn = end / numOfChars;

double freq;

allChars.get(0).add(Double.toString(startNum));

for (int i=1;i<allChars.size();i++){

int numOfCurrChar = Integer.parseInt(allChars.get(i-1).get(1));

freq = dnn\*numOfCurrChar;

startNum = startNum +freq;

allChars.get(i-1).add(Double.toString(startNum));

allChars.get(i).add(Double.toString(startNum));

}

allChars.get(allChars.size()-1).add(Double.toString(end));

return allChars;

}

public static void methodArifm(String text) {

System.out.println("Cообщение: " + text);

String chars = text;

int numOfChars = text.length();

ArrayList<ArrayList<String>> allChars = searchChars(chars);

double min = 0;

double max = 1;

double delta;

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

char l = text.charAt(n);

String el = String.valueOf(l);

for (int i=0;i<allChars.size();i++){

String currEl = allChars.get(i).get(0);

if (el.equals(currEl)){

delta = max - min;

double start = Double.parseDouble(allChars.get(i).get(2));

double end = Double.parseDouble(allChars.get(i).get(3));

max = min + delta\*end;

min = min + delta\*start;

}

}

}

DecimalFormat dfMin = new DecimalFormat("#.#######");

DecimalFormat dfMax = new DecimalFormat("#.##########");

System.out.println("Закодированное сообщение: [ " + dfMin.format(min) +" ; "+dfMax.format(max)+" ]");

}

public static void methodLZ77(String text, int nSlov, int nBuff){

System.out.println("Cообщение: " + text);

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

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

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

slovar.add("\_");

}

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

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

char l = text.charAt(i);

String el = String.valueOf(l);

buffer.add(el);

}

int currChar = nBuff;

int first = 0;

int second = 0;

do{

String elemFromSlov = null;

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

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

if (buffer.get(0).equals(slovar.get(i))){

elemFromSlov = buffer.get(0);

}

}

if (buffer.get(0).equals(elemFromSlov)){

second++;

if (second==1){

first = slovar.indexOf(elemFromSlov);

}

slovar.add(buffer.get(0));

buffer.remove(0);

slovar.remove(0);

if (currChar<text.length()) {

buffer.add(String.valueOf(text.charAt(currChar)));

}

currChar++;

}

else{

oneResult.add(Integer.toString(first));

oneResult.add(Integer.toString(second));

oneResult.add(buffer.get(0));

System.out.println(oneResult);

result.add(oneResult);

second = 0;

first = 0;

slovar.add(buffer.get(0));

slovar.remove(0);

if (currChar<text.length()) {

buffer.add(String.valueOf(text.charAt(currChar)));

}

buffer.remove(0);

currChar++;

}

}while (buffer.size()>0);

//System.out.println("Закодированное сообщение: " + result);

}

}

package com.company;

import java.util.\*;

public class Hoffman {

private static Map<Character, String> charPrefixHashMap = new HashMap<>();

static HuffmanNode root;

public static void methodHoffman(String text) {

System.out.println("Сообщение: " + text);

String encodeText = encode(text);

System.out.println("Закодированное сообщение: " +encodeText);

}

public static void encodeHoffman(String text) {

decode(text);

}

private static HuffmanNode buildTree(Map<Character, Integer> freq) {

PriorityQueue<HuffmanNode> priorityQueue = new PriorityQueue<>();

Set<Character> keySet = freq.keySet();

for (Character c : keySet) {

HuffmanNode huffmanNode = new HuffmanNode();

huffmanNode.data = c;

huffmanNode.frequency = freq.get(c);

huffmanNode.left = null;

huffmanNode.right = null;

priorityQueue.offer(huffmanNode);

}

assert priorityQueue.size() > 0;

while (priorityQueue.size() > 1) {

HuffmanNode x = priorityQueue.peek();

priorityQueue.poll();

HuffmanNode y = priorityQueue.peek();

priorityQueue.poll();

HuffmanNode sum = new HuffmanNode();

sum.frequency = x.frequency + y.frequency;

sum.data = '-';

sum.left = x;

sum.right = y;

root = sum;

priorityQueue.offer(sum);

}

return priorityQueue.poll();

}

private static void setPrefixCodes(HuffmanNode node, StringBuilder prefix) {

if (node != null) {

if (node.left == null && node.right == null) {

charPrefixHashMap.put(node.data, prefix.toString());

} else {

prefix.append('0');

setPrefixCodes(node.left, prefix);

prefix.deleteCharAt(prefix.length() - 1);

prefix.append('1');

setPrefixCodes(node.right, prefix);

prefix.deleteCharAt(prefix.length() - 1);

}

}

}

private static String encode(String test) {

Map<Character, Integer> freq = new HashMap<>();

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

if (!freq.containsKey(test.charAt(i))) {

freq.put(test.charAt(i), 0);

}

freq.put(test.charAt(i), freq.get(test.charAt(i)) + 1);

}

root = buildTree(freq);

setPrefixCodes(root, new StringBuilder());

StringBuilder s = new StringBuilder();

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

char c = test.charAt(i);

s.append(charPrefixHashMap.get(c));

}

return s.toString();

}

private static void decode(String s) {

StringBuilder stringBuilder = new StringBuilder();

HuffmanNode temp = root;

System.out.println("Закодированное сообщение: " + s);

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

int j = Integer.parseInt(String.valueOf(s.charAt(i)));

if (j == 0) {

temp = temp.left;

if (temp.left == null && temp.right == null) {

stringBuilder.append(temp.data);

temp = root;

}

}

if (j == 1) {

temp = temp.right;

if (temp.left == null && temp.right == null) {

stringBuilder.append(temp.data);

temp = root;

}

}

}

System.out.println("Раскодированное сообщение: " + stringBuilder.toString());

}

}

class HuffmanNode implements Comparable<HuffmanNode> {

int frequency;

char data;

HuffmanNode left, right;

public int compareTo(HuffmanNode node) {

return frequency - node.frequency;

}

}