BLG439E
Computer Project I
Project II

1) Huffman Encoding and Decoding

We used java.io and java.util libraries for java functions. In the android codes we used HuffmanCoding class for encoding and decoding the giving string.

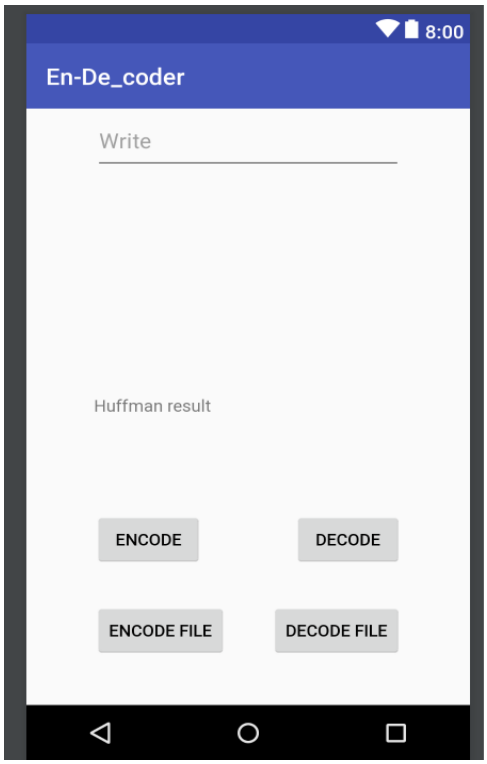
We used priority queue for building the tree. Priority queue's normal comparator is compare string values but we need to compare our nodes frequency therefore, we wrote our own comparator function in the our tree node class.

```
package com.two.project.computer.en_de_coder;

import java.io.*;
import java.util.*;

public class HuffmanCoding {

    //static Comparator<TreeNode> comparator = new TreeNodeComparator();
    //static PriorityQueue<TreeNode> nodes = new PriorityQueue<>(10, comparator);
    static PriorityQueue<TreeNode> nodes = new PriorityQueue<>();
    static TreeMap<Character, String> codeMap = new TreeMap<>();
    static String inputText;
    static String encoded;
    static  String decoded;
    static int freq[] = new int[128];
```

Pic1. Definitions

In the android application our main class is called MainActivity. It connected to the main_activity layout which is our programs main screen. In MainActivity class we called HCEncode and HCDecoder functions with input string. Input string can have 2 different values. One of them is the fileData which is the data inside the "input.txt" file inside the download folder. The other value for input string is users input text which is Pic 2. When user pushes the Encode button program sends the input with editText value and then encodes it. After encoding program shows the encoded result in the "Huffman result" area. When user pushes the Decode button program decodes it and shows the decoded result in the "Huffman result" area.

However when user pushes the Encode File or Decode File button program sends the inside of the file as a parameter of the HCEncode or HCDecode functions.



Pic2. Screen

```
100 @      public static String HCEncode(String text){
101
102            inputText = text;
103            if(inputText == null)
104                return null;
105
106            freq = new int[128];
107            nodes.clear();
108            codeMap.clear();
109            encoded = "";
110            decoded = "";
111
112            //Finding every characters frequency
113            for (int i = 0; i < inputText.length(); i++)
114                freq[inputText.charAt(i)]++;
115
116            //build tree and generate huffman codes
117            buildTree();
118            generateCodes(nodes.peek(), S: "");//Send root and the empty result
119
120            encodeText();
121            return encoded;
122        }
123
124        public static String HCDecode(){
125            decodeText();
126            return decoded;
127        }
```

Pic3. HCEncode & HCDecode functions which are called from MainActivity

In the HCEncode, funciton first checks inside of the inputText for null values. Then creates new new frequency array, clears priority queue (nodes) and map (codeMap) and assigns empty strings to encoded and decoded strings for every new input values.

After that, for loop calculates the frequency of every character in the inputText and store them insedi the freq array. Then function calls buildTree, generateCodes, encodeText functions and returns encoded string value.

```
17      private static void buildTree() {
18
19          //Putting letter in tree as nodes (char data and frequency
20          for(int i=0; i < 128; i++){
21              if(freq[i] != 0)
22                  nodes.add(new TreeNode(((char) i), freq[i], L: null, R: null));
23          }
24
25
26          //Merging two smallest trees and put data='\0' in the intersection node
27          while (nodes.size() > 1) {
28              TreeNode left  = nodes.poll();
29              TreeNode right = nodes.poll();
30              if(left.frequency < right.frequency)
31                  nodes.add(new TreeNode( Data: '\0', Frequency: left.frequency + right.frequency, left, right));
32              else
33                  nodes.add(new TreeNode( Data: '\0', Frequency: left.frequency + right.frequency, right, left));
34          }
35
36      }
```

Pic4. buildTree funciton

In the buildTree, function first puts every non-zero frequency charater inside a prioirty queue. After adding every character inside a priority queue it merges smallest trees inside a while loop. Inside that loop first polls the head of queue as a left node and then polls the queue again as a right node. Compares the left and the right nodes frequency and if left one is already bigger than the right nodes frequency it merges them in the order and puts their merged nodes data end of file character (‘\0’). If their order is reverse they merge them with reverse order.

```
38      //Traversing in the tree for getting every letter's huffman code and put them in the codeM
39      private static void generateCodes(TreeNode node, String s) {
40          if (node != null) {
41              //Right node -> 1
42              if (node.right != null)
43                  generateCodes(node.right, s + "1");
44              //Left node -> 0
45              if (node.left != null)
46                  generateCodes(node.left, s + "0");
47              //Encode is complited and we reached at one of the leafs
48              if (node.left == null && node.right == null)
49                  codeMap.put(node.data, s);//hashMap that holds char and the encode
50          }
51      }
```

Pic5. GenerateCodes function

In the generateCodes, function generates Huffman codes with using tree. The function is recursive and traverse all over the tree with inorder traverse method. When traversing on the tree it checks every leaf for character and their huffman codes. They saves every character and their code inside the codeMap hash map.

```
75      private static void encodeText() {
76
77          encoded = "";
78          BinaryOut bout = new BinaryOut();
79
80          //Our compressed file
81          String filename = "Out.huf";
82          new BinaryOut(filename);
83
84          for (int i = 0; i < inputText.length(); i++) {
85              String code = codeMap.get(inputText.charAt(i));
86              encoded += codeMap.get(inputText.charAt(i));
87              //Writing encoded bits to the file
88              for (int j = 0; j < code.length(); j++) {
89                  if (code.charAt(j) == '0')
90                      BinaryOut.write( x: false);
91
92                  else if (code.charAt(j) == '1')
93                      BinaryOut.write( x: true);
94
95                  else throw new IllegalStateException("Illegal state");
96              }
97          }
98      }
```

Pic6. EncodeText function

In the encodeText, function first clears the old encoded text then opens file with BinaryOut method which writes bits to the file. Later, it encodes file inside the for loop. It checks every character of inputText and finds every characters huffman code inside the codeMap. Finally, function write these bits to the file with BinaryOut.write function.

```
53      private static void decodeText() {
54          decoded = "";
55          //Get root
56          TreeNode root = nodes.peek();
57          //Decode text with tree
58          for (int i = 0; i < encoded.length(); ) {
59              TreeNode traverse = root;
60              //If traverse is not on leaf and whole encoded text is not complied then continue to loop
61              while (traverse.left != null && traverse.right != null && i < encoded.length()) {
62                  //Right node -> 1
63                  if (encoded.charAt(i) == '1')
64                      traverse = traverse.right;
65                  //Left node -> 0
66                  else traverse = traverse.left;
67                  i++;
68              }
69              if (traverse != null)
70                  if (traverse.data != '\0')
71                      decoded += traverse.data;
72          }
73      }
```

Pic7. DecodeText function

In the decodedText, function first clears the old decoded text then finds the root of the tree with peek function. After that, it traverses inside the encoded string at the same time it traverses inside the tree and finds the encoded character. When it reaches one of the leafs it stops and add that leafs character to the decoded strintg.

```
159     //Node class
160     class TreeNode implements Comparable<TreeNode> {
161         char data;
162         int frequency;
163         TreeNode left, right;
164
165         public TreeNode(char Data, int Frequency, TreeNode L, TreeNode R) {
166             this.left = L;
167             this.right = R;
168             this.data = Data;
169             this.frequency = Frequency;
170         }
171
172         public int compareTo(TreeNode b) {
173             if (frequency > b.frequency) {
174                 return 1;
175             } else if (frequency < b.frequency) {
176                 return -1;
177             } else {
178                 return 0;
179             }
180             //return this.frequency - b.frequency;
181         }
182     }
```

Pic8. TreeNode Class with compareTo funciton for priority queue

Tree Node class is the data structure for our tree. It contains char data, frequency of that data and 2 tree node pointers for leafs. Compare function is for priortiy queue. When priority queue wants to add new item to the queue it uses this function. We designed this function because standart compare function just compare string values while we need to compare our nodes frequency values.

We used BinaryOut and BinaryIn classes for writing and reading files with binary values. Therefore, we implement these functions inside our android application.

2) Connection

We used Bluetooth Chat Android Project for connection of android and pc. This project connect 2 devices via bluetooh sends strings. For our project we modified that Bluetooth Chat project and used it for our connection purpouses.

```
199     /**
200      * Sends a message.
201      * @param message  A string of text to send.
202      */
203     //We just changed this method because we just want to send our encoded text instead of message
204     private void sendMessage(String message) {
205         // Check that we're actually connected before trying anything
206         if (mChatService.getState() != BluetoothChatService.STATE_CONNECTED) {
207             Toast.makeText( context: this, R.string.not_connected, Toast.LENGTH_SHORT).show();
208             return;
209         }
210         //Boolean value for checking file condition
211         boolean FileOpened = false;
212         String encoded, fileData = null;
213
214         //Reading input.txt file in the download folder
215         try {
216             FileInputStream FIS = new FileInputStream( name: Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWN
217             int size = FIS.available();
218             byte[] buffer = new byte[size];
219             FIS.read(buffer);
220             FIS.close();
221             fileData = new String(buffer);
222             FileOpened = true;
223         }catch (Exception f){
224             f.printStackTrace();
225             Toast.makeText(getApplicationContext(),  text: "File could not open so input text sent", Toast.LENGTH_SHORT).show();
226         }
227
228         //if file could not be open use message text for encoding, else fileData for encoding
229         if(FileOpened == false)
230             encoded = HuffmanCoding.HCEncode(message);
231         else
232             encoded = HuffmanCoding.HCEncode(fileData);
233
234         // Check that there's actually something to send
235         if (encoded.length() > 0) {
235         if (encoded.length() > 0) {
236             // Get the message bytes and tell the BluetoothChatService to write
237             byte[] send = encoded.getBytes();
238             mChatService.write(send);
239
240             // Reset out string buffer to zero and clear the edit text field
241             mOutStringBuffer.setLength(0);
242             mOutEditText.setText(mOutStringBuffer);
243         }
244     }
```

Pic9. Bluetooth Chat sendMessage function

We created 2 different android apps. One of them (Bluetooth Chat) can send encoded data via bluetooth but it cannot send huffman tree so decoding will not work on this app. Our second app (En/decoder) can encode and decode the text in the app but this app cannot send data to anywhere because, it does not have any connection part.

In the sendMessage, function first try to read the file which inside the downloads directory. If file could not be read then instead of fileData function uses users chat input for encoding. After encoded one of them it will send encoded data to the server pc and pc server can see this encoeded data on screen. However we could not manage to send tree for decoding so decoding part will not work properly. But decoding part's code will correct and it will work on En/Decoder app.

3) Other Projects

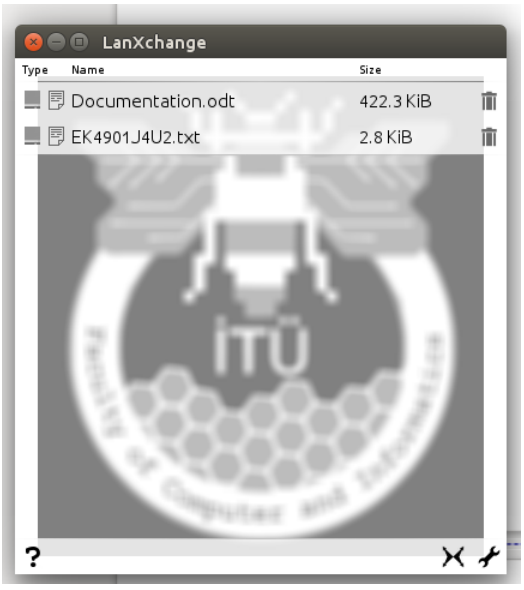
We investigated some other projects to understan implementation of huffman code, file operations like reading a file and writing to a file, bit operations, vs. in java code which is totally a mystery for us because of the fact that none of us knew or used java before. We could not run most of the packages we found because of being unfamiliar with the platform and finding required dependencies.

```
Priority.java  huffman.java
292         }
293     }    //WORKS !!!
294
295
296     // ********************************************************************
297     private void decode()
298     {
299         String str="";
300         String str2="";
301         StringBuffer str3=new StringBuffer();
302         try {
303             FileInputStream f= new FileInputStream( s: "encoded.bin");
304             int num=f.available();
305             for(int i=0;i<num;i++)
306             {
307                 int a=f.read();
308                 str=Integer.toBinaryString(a);
309                 while(str.length()!=8)
310                 {
311                     str='0'+str;
312                 }
313                 str3.append(str);
314
315             }
316             str2=str3.toString();
317
318             str2=str2.substring( i: 0, i1: str2.length()-counter);
319             //System.out.println("testing---"+str2);   //final decoded string
320
321             f.close();
322
323         } catch (FileNotFoundException e) {
324             // TODO Auto-generated catch block
325             e.printStackTrace();
326         } catch (IOException e) {
327             // TODO Auto-generated catch block
328             e.printStackTrace();
329         }
330         recursion(str2);
331     }
332
333
334     // ********************************************************************
335     void recursion(String s)
336     {
```

Pic10. Decoding function

An inspiring working project was Text-compression-using-Huffman-coding [4]. Project reads from an input file, creates an encoded file, than from this encoded file creates back a decoded file which is same with the original input file. Encoded files size is almost %50 of original. An example of these files can be found in our repository. Althoug this demo was capable of all required functionalaties required for this roject, all variables and functions were nested in each other so that we could not separete that to make a use of them.

For the sending files between devices we found a project for cross platform file transfer named lanXchange [10]. It gives the flexibility of sending and recieving any kind of file from the local area network. Drag and drop GUI simply shows up on other devices and shared files can be downloaded to local drives. We modified the GUI for visual refinement in case of using in our project but we did not implement it in our project becuse we cuold not manage to succesfully work on seperate files.



Pic11. lanXchange GUI

Authors:

150130032 – Baran Kaya
150130047 – Kadir Enes Karslıoğlı
150140804 – Mehmet Ali Osman Atik

References:

[1] **http://www.codemiles.com/java/standard-huffman-coding-t6.html**

[2] **https://www.codemiles.com/java/huffman-compression-decompression-t96.html**

[3] **https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/Huffman.java.html**

[4] **https://github.com/tarunsharma1/Text-compression-using-Huffman-coding**

[5] **https://gist.github.com/ahmedengu/aa8d85b12fccf0d08e895807edee7603**

[6] **http://sourcecodesforfree.blogspot.com.tr/2013/05/16-text-compression-using-huffman.html**

[7] **https://stackoverflow.com/questions/30253118/huffman-decompression**

[8] **https://stackoverflow.com/questions/43421273/huffman-code-writing-bits-to-a-file-for-compression**

[9] **https://stackoverflow.com/questions/25843949/sending-string-from-pc-to-android-app-using-bluetooth-dongle**

[10] **https://github.com/tfg13/LanXchange**