HACETTEPE UNIVERSITY

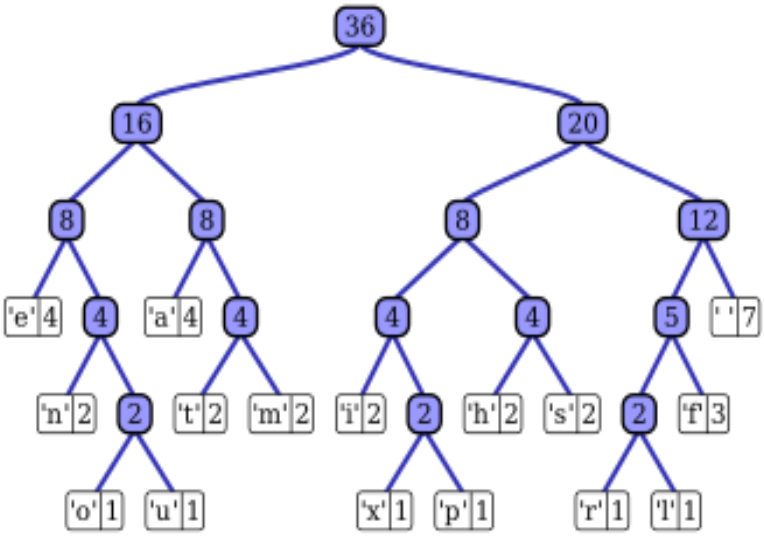
FACULTY OF ENGİNEERİNG - COMPUTER SCİENCE

BBM203

ASSIGNMENT 4

Student ID	Name & Surname	Due date
21827263	**Murat Çelik**	**02/01/2021**

Assignment 4 – Trees



Huffman coding

In computer science and information theory, a Huffman code is a particular type of optimal prefix code that is commonly used for lossless data compression. The process of finding or using such a code proceeds by means of Huffman coding, an algorithm developed by David A. Huffman while he was a Sc.D. student at MIT, and published in the 1952 paper "A Method for the Construction of Minimum-Redundancy Codes".[1]

Huffman tree generated from the exact frequencies of the text "this is an example of a huffman tree". The frequencies and codes of each character are on the right side. Encoding the sentence with this code requires 135 (or 147) bits, as opposed to 288 (or 180) bits if 36 characters of 8 (or 5) bits were used. (This assumes that the code tree structure is known to the decoder and thus does not need to be counted as part of the transmitted information.)

Char	Freq	Code
space	7	111
a	4	010
e	4	000
f	3	1101
h	2	1010
i	2	1000
m	2	0111
n	2	0010
s	2	1011
t	2	0110
l	1	11001
o	1	00110
p	1	10011
r	1	11000
u	1	00111
x	1	10010

[1] Wikipedia - Huffman coding => https://en.wikipedia.org/wiki/Huffman_coding

Subject

The goal are :

1) After take a input and encode it, print binary code on the terminal.

2) Print huffman tree which created according to input.

3) Print char's binary code according to algorithm.

4) After take a input and decode it, print text message on the terminal.

Samples

Sample inputs and outputs are;

• input file contains; BCCABBDDAECCBBAEDDCC, the output can be;

10111100110100101001000111110100010000101111 1 and vice-versa.

• input file contains; BCAADDDCCACACAC, the output can be;

1000111110110110100110110110 and vice-versa.

Compile and Run

- make
- ./Main -i <input>.txt -encode
- ./Main -l
- ./Main -s <character>
- ./Main -i <input>.txt -decode

Code – Algorithm

main.cpp

main() function is the entry point of any C++ program. It is the point at which execution of program is started. When a C++ program is executed, the execution control goes directly to the main() function. Every C++ program have a main() function.[2]

Firstly, I included ControlObject.h file. I called the methods of the admin object controlPermission by filtering the command received from the terminal.

*main.cpp

```
1  #include "ControlObject.h"
2  int main(int argc, char** argv){
3
4      ControlObject controlPermission;   // all command decisions are made here
5
6      if(argc == 4){
7          string third = argv[3];
8          if(third == "-encode")
9              controlPermission.encodeMethod(argc,argv); // ./Main -i encodeFile.txt -encode
10         else if(third == "-decode")
11             controlPermission.decodeMethod(argc,argv); // ./Main -i decodeFile.txt -decode
12     }
13     else if(argc == 2)
14         controlPermission.listMethod(argc,argv); // ./Main -l
15     else if(argc == 3)
16         controlPermission.findCharMethod(argc,argv);// ./Main -s <char>
17 }
```

ControlObject.h

This is an admin class. We have 4 commands and classes belonging to them. We added these classes inside. The methods of the class belonging to the called command run sequentially in body.

*ControlObject.h

```
1  #include <iostream>
2  #include "Encode.h"
3  #include "List.h"
4  #include "Decode.h"
5  #include "treeNode.h"
6  #include "GiveChar.h"
7
8  using namespace std;
9
10 // my Admin class
11
12 class ControlObject {
13 public:
14     void encodeMethod(int num, char** arr);
15     void decodeMethod(int num, char** arr);
16     void listMethod(int num, char** arr);
17     void findCharMethod(int num, char** arr);
18 };
```

[2] https://www.sitesbay.com/cpp/cpp-main()-function#:~:text=main()%20function%20is%20the,have%20a%20main()%20function.

treeNode.h

This class is the node class I use to create the tree. Each node has a letter, the letter's frequency number, a binary code, and left and right extensions. Right extension frequency is for large, left extension is for small ones.

*treeNode.h

```
1   #include <string>
2
3   class treeNode
4   {
5   public:
6       char letter;
7       int count;
8       std::string binaryCode = "";
9       treeNode *leftNode = nullptr;
10      treeNode *rightNode = nullptr;
11
12  };
13
```

Encode.h

We have 5 methods. The text read in the ControlObject's method named encodeMethod is sent to the countFreq method. Here the frequency of each letter is found. After that the createTree method is called. According to Huffmann algorithm, node is created for each letter here and a tree is created with them. The chechBinaryCodeLEafNode method records the binary code for each letter according to the tree. With the printPreorder method, the tree is saved in a file for the next commands. PreOrder traversal algorithm is used for this. I am displaying the binary text in the terminal with the encodeText method.

*Encode.h

```
1   #include <iostream>
2   #include <string>
3   #include "treeNode.h"
4   #include <vector>
5   #include <algorithm>
6   #include <fstream>
7
8   using namespace std;
9
10  void countFreq(string str) ;
11  void chechBinaryCodeLeafNode(struct treeNode* node);
12  void createTree(treeNode& rootNode);
13  void printPreorder(struct treeNode* node, ofstream &myFile);
14  void encodeText(string myText);
```

reCreateTree.h

This method allows 3 other commands to use the tree. Methods of this class are invoked before those commands.I am recreated the tree with the information in the saved file. I'm assigned the root node to the newRootNode property.

*reCreateTree.h

```
1  #include <fstream>
2  #include <vector>
3  #include "treeNode.h"
4  #include <iostream>
5  #include <sstream>
6  #include <string>
7  using namespace std;
8
9
10 // read file and reCreate tree map
11 class reCreateTree{
12 public:
13     vector<treeNode*> vDecodeTree;
14     vector<string> splitArray;
15     treeNode* newRootNode;
16
17     void CreateTree(ifstream &myTreeFile);
18     void splitWord(const string& str);
19     };
```

List.h

According to the PreOrder Traversal algorithm, I print the treemap to the terminal.

*List.h

```
1  #include "treeNode.h"
2  #include <fstream>
3  #include <iostream>
4  using namespace std;
5
6  void writeTree(treeNode* mainNode);
7
```

```
+-  20
    |  +-  9
    |      |  ---- d => 4 => 00
    |      |  ---- b => 5 => 01
    |  +-  11
    |      |  +-  5
    |      |      |  ---- e => 2 => 100
    |      |      |  ---- a => 3 => 101
    |      |  ---- c => 6 => 11
```

GiveChar.h

It finds and displays the binary code of the desired letter by browsing through the tree with the preOrder algorithm.

*GiveChar.h

```
1  #include <string>
2  #include <iostream>
3  #include "treeNode.h"
4  using namespace std;
5
6  void findNode(string findChar, treeNode* root, treeNode& initial);
7
```

Decode.h

According to the given binary text, it reaches the leaf node by navigating the tree. Saves the letter of the node reached. At the end, it prints the entire text on the screen.

*Decode.h

```
1  #include <fstream>
2  #include "treeNode.h"
3  #include <iostream>
4  #include <sstream>
5  #include <string>
6  #include "reCreateTree.h"
7  using namespace std;
8
9  void decode(ifstream &myDecodedFile, class reCreateTree &tree);
10
```

Additional

1) Like the example given in the homework file, I used the " character for whitespace.

2) My algorithm works for a single line file, according to the Piazza announcement.