Huffman Tree GUI

By: Mark Cledera

# Declaration of Authorship

* This work was done wholly or mainly while in candidature for a research degree at this University.
* Where any part of this proposal has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
* Where I have consulted the published work of others, this is always clearly attributed.
* Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project proposal is entirely my own work.
* I have acknowledged all main sources of help.
* Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Mark Cledera

Date: 1/12/2021

Contents

[Declaration of Authorship 2](#_Toc89285093)

[Introduction 3](#_Toc89285094)

[HuffmanTree Class 3](#_Toc89285095)

[HuffItem Class 8](#_Toc89285096)

[ListArrayBased Class 9](#_Toc89285097)

[GUI Class 10](#_Toc89285098)

[Testing 13](#_Toc89285099)

[Conclusion 23](#_Toc89285100)

# Introduction

In this report I will be going through my code for the Huffman tree, the frequency table that I used was the Letter Count Ascending text file as recommended by my lecturer, the specific classes I will be covering in this report are ListArrayBased which comes from the labs of our Data Algorithms Module, HuffItem which is a class that is used to initialise the data from the frequency table, TreeNode which is used to store the data and frequencies into nodes, HuffmanTree which contains code for reading the table, generating the tree, decoding and encoding the values. I will also briefly cover the GUI which contains a form of validation and how it outputs the results. I have also structured my directories appropriately to keep the directory as organised as possible by storing the required code for the classes in an algorithm package, my main class inside a main package and the GUI inside of a gui package. I also used IntelliJ to build this program, IntelliJ also generates class files and compiles the program automatically, these files are in the out folder.

# HuffmanTree Class

The HuffmanTree class starts with various global variables, each one is used in each method with respective comments to where the variables are used in. The variables can be seen here in this figure

Text

Description automatically generated

Figure 1 Variables to be used in the Huffman Tree class

After the variables the reading method comes for reading the frequency table the first part is variables for reading the files to be used for the tree, these variables are built into Java.

Text

Description automatically generated

Figure 2 Variables for the reading of the text file for the tree

Once the text file is initialised it is passed through a while loop which reads the file while it’s not empty and stores it into a string array, this string array contains 2 datasets which holds the by reading the strings on the array and making use of the split method to detect the space between the letter and the frequency on the text file. The letters are stored on index 0 of the array and the numbers are stored in index 1 so a variable is initialised to represent each of the stored values.

Text

Description automatically generated

Figure 3 variables which represent the stored values of the array

Once the variables are initialised to their respective indexes, they are then initialised to a HuffItem object which will be used for the TreeNode object which will then be used in adding to the List Array, with the data contained in the list array, the generate tree method can make use of the data to do the necessary calculations for the tree generation method.

Text

Description automatically generated

Figure 4 putting the letters and symbols into the Huffitem object to then initialise the tree node variable for the list array

The next method covers the generation of the tree. The while loop begins if there are items on the list array so it must be greater than 1 to continue. Variables are then initialised to represent the nodes that are located on the left and right. When the loop completes the summing of the latest frequency the items are removed from the list array to calculate the next 2 frequencies which is why there are 2 uses of the remove method in the loop. Following the variable initialisation, a new variable is made to represent the nodes and added into the tree based on the sum of the frequencies. The list array then adds the items into the tree then sorts the list to consider the completed frequencies which were removed above the code. So, a bubble sort is used to move the frequencies up from the list so the loop can continue summing the frequencies until there is no more items in the list. Once there are no more items on the list the tree is completed the root node is initialised to the final sum of the frequencies. By initialising the root node, the tree is completed.

Text

Description automatically generated

Figure 5 Showing the loop for summing the frequencies, sorting the list, and adding it to the tree

The decode method makes use of a String for the parameter, a treenode variable is initialised to the root node so the method can start from there. A variable which is a character is then used for the left and right sides which is 0 and 1. A for loop is then used to iterate through the tree and compare the character to 0 and 1 with an if statement which will go left and right respectively. This loop will then return a string which gets the letter that the traversal gets to. So, a variable is initialised to the get method of the node and is returned.

Text

Description automatically generated

Figure 6 The decode method

For the encode method a recursive call is used due to it needing to get the traversals from the tree when it encodes a letter into the binary digits. The method begins with a variable for the letter initialised to the get symbol method. The letter is then compared to the symbol and initialised to the code parameter which will contain the binary digits of the traversal. The if statements afterwards are the traversals for the 0’s and 1’s while the node is not empty. To get the actual encode results a get method is used to get the items on the global variables. A get method for the tree nodes is also made to pass the data into the method.

Text

Description automatically generated

Figure 7 The encode method and the get methods it uses

# HuffItem Class

In the HuffItem class there is 2 variables that are declared, one for the letters and the frequencies these are then used for the constructor in the class, a get method is also made for these two variables to be used in the Huffman Tree class.

Text

Description automatically generated

Figure 8 The HuffItem class

# ListArrayBased Class

In the ListArrayBased class the only new method that is created is the bubble sort, this is so it can detect the frequencies that were added to the list in the generate tree method. The bubble sort uses two for loops that go through the list, an if statement checks for when the summed frequency which will be greater than the next frequency to be added to the summed frequency. An object variable is used to store the data and sort the sum and the next frequencies to be calculated and repeats the calculation until only the final sum remains which will represent the root of the tree from the generate tree method

Text

Description automatically generated

Figure 9 The bubble sort method

# GUI Class

In the GUI class a JFrame is created and laid out appropriately as outlined in the assignment doc.

Website

Description automatically generated with low confidence

Figure 10 GUI of the Huffman Tree encode and decode

The buttons use an action listener to detect the input. There is a small implementation of input validation as well to ensure that the user does not enter invalid strings for encoding or decoding which will have more details in the testing section of this report.

When encode is pressed after typing a letter the result for the encode is set into the text area at the bottom.

A picture containing treemap chart

Description automatically generated

Figure 11 Of output for encoding

When the same 3 zeroes are typed in and the decode is pressed it outputs the letter located on the node.

A picture containing treemap chart

Description automatically generated

Figure 12 Output for decoding

I also made a tree beforehand to check all the letters and their digits, so it aligns perfectly.

# Testing

The validation for encode and decode uses a Boolean that allows the code to run only when the input is valid if it’s not the Boolean will be set to true, and a message will output to the text box.

For encoding the character is checked if it’s a letter, is lowercase or it contains special characters and if its empty.

Text

Description automatically generated

Figure 13 Encode validation method

Text

Description automatically generated

Figure 14 More validation catching the error when the text area is empty

The outputs of the error handling can be seen here in these figures

Chart, treemap chart

Description automatically generated

Figure 15 Error for empty text box

Chart, treemap chart

Description automatically generated

Figure 16 Error for detecting lower cases

Timeline

Description automatically generated with medium confidence

Figure 17 Error if anything that isn't capital letters is put in

For decoding the method checks for if it matches a letter, matches any number from 2 to 9, matches the regex of special characters and in the action performed if the digits land on a frequency node which only contains an asterisk a message appears saying there isn’t enough digits. There is also a catch for when the binary digits reach an empty or non-existent node which will just give an invalid input error.

Text

Description automatically generated

Figure 18 The decode validation method

Text

Description automatically generated

Figure 19 More handling in the decode part of the program

Treemap chart

Description automatically generated

Figure 20 Error when attempting to decode a letter

Treemap chart

Description automatically generated

Figure 21 Error when typing in any other number than 1 or 0

Chart, treemap chart

Description automatically generated

Figure 22 Error when text box is left empty

Chart, treemap chart

Description automatically generated

Figure 23 Error when attempting to reach a frequency instead of a letter

# Conclusion

In conclusion the tree is generated, and the output is as expected of what’s located on the tree, by comparing it to the tree I have created to check if it does what it is intended to do. The only other issues that are present in the program is the input validation. But aside from the small issues with validating the decode and encode the program is giving the expected outputs.