HACETTEPE UNIVERSITY DEPARTMENT OF

COMPUTER ENGINEERING

BBM 203 ASSIGNMENT 1

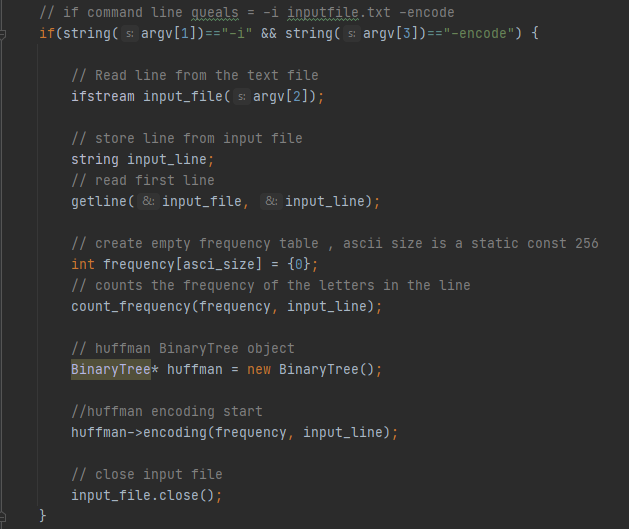


Name Surname – Number

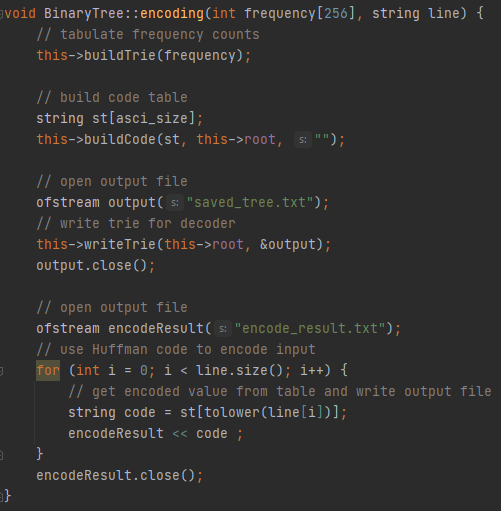
Subject : Trees

Programming Language : C++

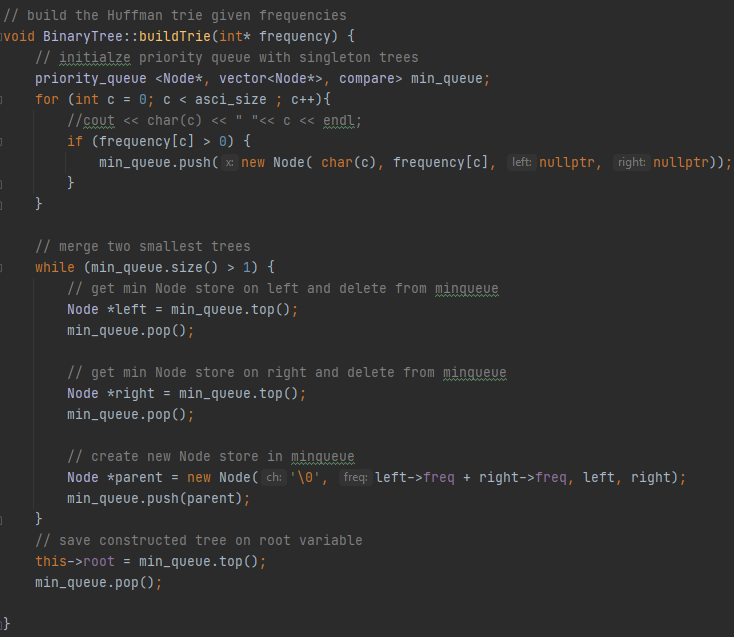
1. **Explain your encoding algorithm and code step by step in detail**



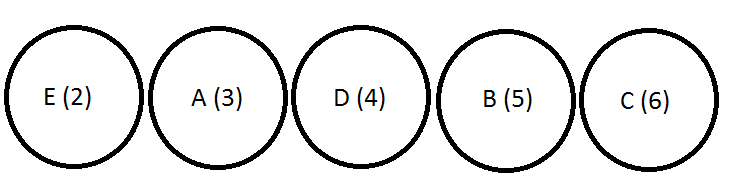
First, the first line of the file in the argument is read with the getline () function and saved in the input\_file variable. After the line is read, the frequency of the letters in the line is counted with the count\_frequency() function. BinaryTree object is created for Encoding. The process starts with the encoding function () in the BinaryTree Object.



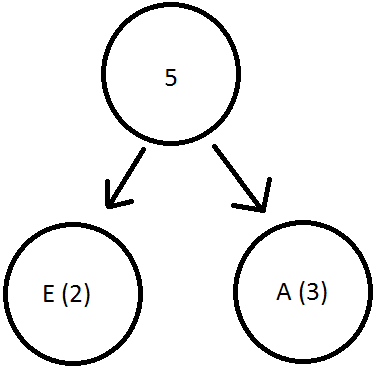
Binary tree is set up with the buildTrie () function during the encoding process.



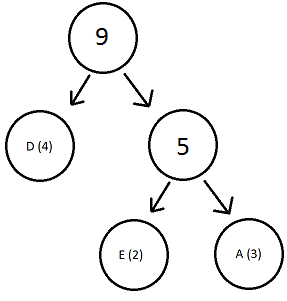
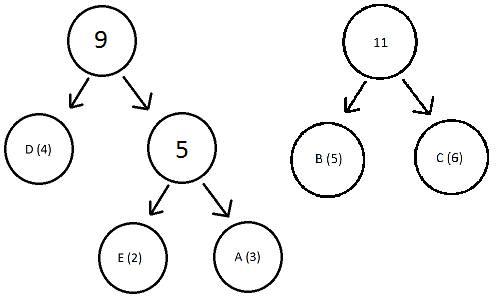
During the BuildTrie process, frequencies are converted to Node and added to min\_queue.After all nodes are added, nodes are taken from min\_queue and merged until the root node is created.



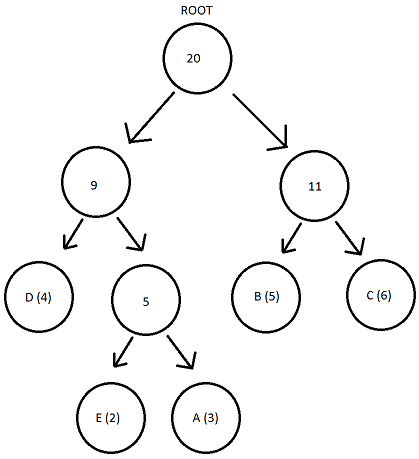
1)



2) 3)

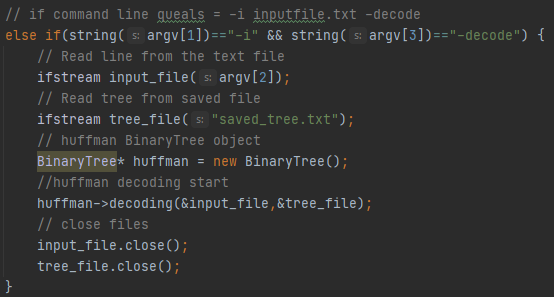
 

4)

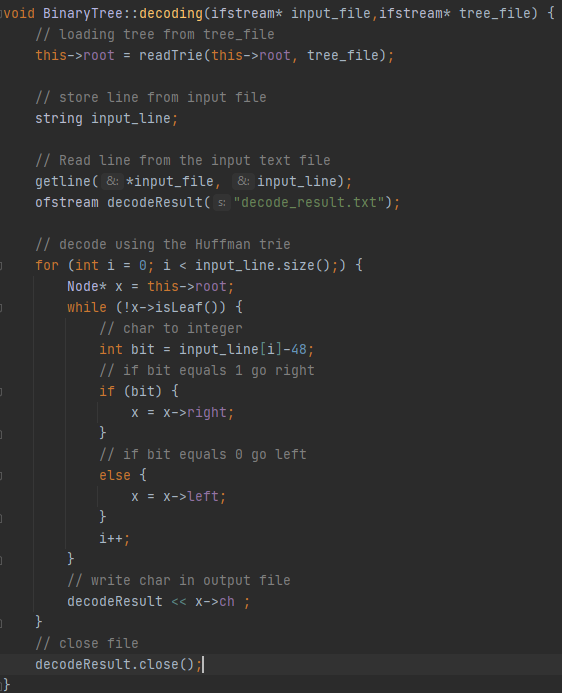


After the root node is created, it is saved in the root variable in BinaryTree. Then, the huffman value of each letter is saved to the string array with the buildcode () function. After the buildcode is finished, it is written to the file to use the BinaryTree for decode, –l, –s operations. The name of the created file is saved\_tree.txt (static). Finally, according to the results in BinaryTree, the compressed version of the given input is saved in the encode\_result.txt (static) file.

1. **Explain your decoding algorithm and code step by step in detail**

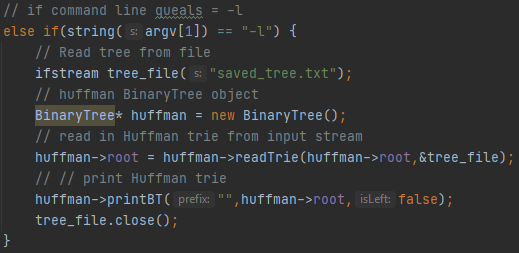


First, the file in the argument is read and saved in the input\_file variable. Then the BinaryTree file saved in the encode is read and saved to the tree\_file variable. The BinaryTree object is created. The process starts with the decoding () function on the object. The most recently read files are closed.

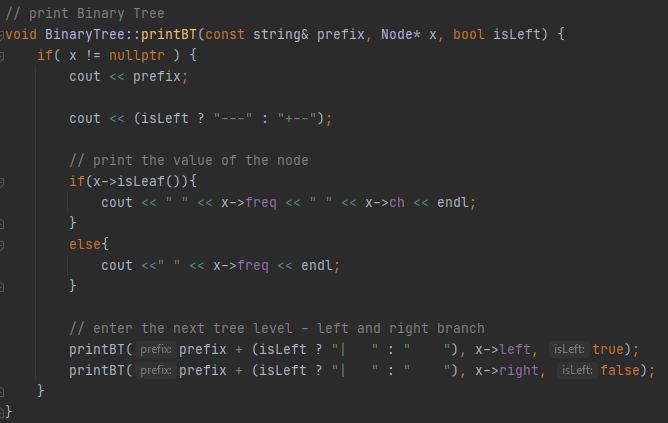


During decoding, first the tree is read from the file with the readTrie () function. Then the first line of the input\_file file in the argument is read with the getline () function and saved in the input\_line variable. A decode\_result.txt file is created to save the results. Finally, with the For loop, decoding is done according to the tree on the input\_line, the results are saved in decode\_result.txt. Finally, the output file is closed and the process is finished.

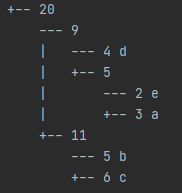
1. **Explain your list tree command**



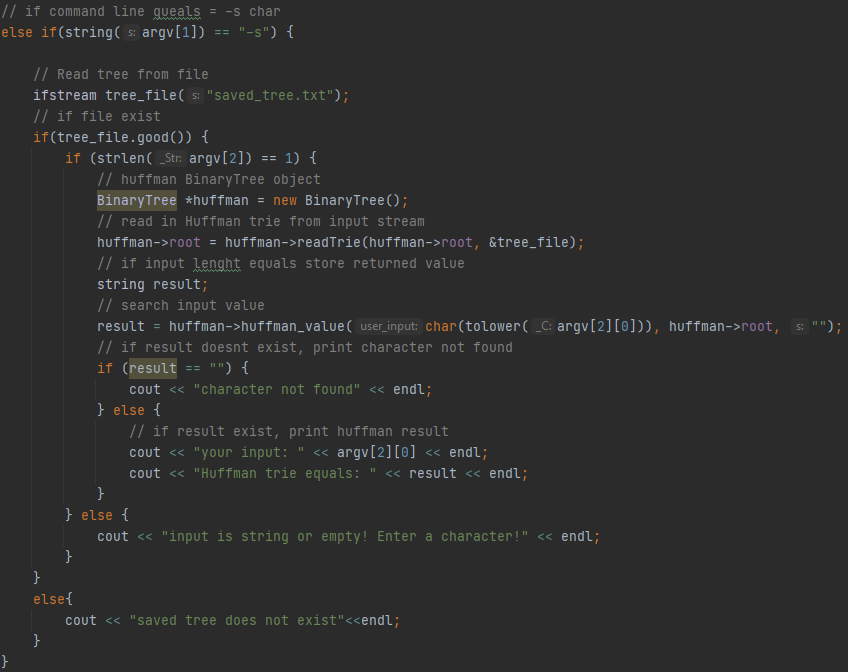
First, the BinaryTree file saved in the encode is read and saved to the tree\_file variable. The BinaryTree object is created. BinaryTree tree is created with the readTrie () function in the object. The BinaryTree tree is printed to the consol with the printBT () function on the object. Finally tree\_fil the file is closed.



The operation in the printBT function continues as long as the node is not equal to a null value. The prefix variable in the parameter sets the spaces between nodes. "---" or "+ -" is printed to the console according to the isLeft variable in the parameter. (First node is Root Node, --- means Left Node, + - means Right Node) If node is leaf, print frequency and character. If node is not leaf, print only frequency. Finally, the process continues by sending next tree level nodes to the printBT function and the result is printed to the console.



1. **–s command**



Saved\_tree.txt is read first. If the file does not exist, the process will not continue.

If the file exists, the length of the input is checked. Then the BinaryTree object is created. The tree is created with the readTrie () function in the BinaryTree object. After the BinaryTree is created, the character in the argument is searched in the BinaryTree with the huffman\_value () function in the object. If there is no character, “character not found” is printed on the console. If available, the result is printed on the console as in the image below.

