**PROCEDURE OF GENERATING PREFIX CODES USING HUFFMAN ALGORITHM**

The language that I used to implement Huffman algorithm is C++. The both question 1 and 2 was implemented using CPP. Huffman coding takes a string as an input and gives us an unique code for every character that is present in the string; and no code is a prefix of another code; this is also known as prefix code.

Now I’m going to elaborate the procedure of Huffman algorithm that I used to generate prefix code for each character.

1) First I generated the string on which I’ll run the algorithm. Then I declared a function named “PrefixCode” which takes String as an input.

2) I also created a structure named as node to store all the characters as a node and later it also can be used as a template for the future node that will be generated from the initial nodes by merging two to one.

3) As I imagined each character as a node and each node having to properties that is; the alphabet it is carrying and the frequency(the number of times it occurred in the string) of that alphabet. And later on a node will be a root for multiple nodes. So I declared two different constructors for the node structure one will take pointer to the nodes and one will create the initial nodes. The node that has the parameter of pointers will have a statement that will sum up the frequencies of two minimums and will assign it as a new freq of the new node.

4) I used priority\_queue to store all the nodes and used to push(), pop() and top() to execute different command. As priority queue sorts the data automatically for that I also defined a custom comparator to make priority-queue sort the element respect to the frequency of the characters. The Priority queue takes node type data and sort it according to the function of its comparator.

5) I used DFS algorithm to traverse through the tree that I created using node and print the final prefix code of each character. I assigned ‘0’ to the left node and ‘1’ to the right node. And leaf node will be NULL. So it will keep on assigning zero or one to the nodes and base case scenario is when it will get a NULL node; it will stop and print the prefix code of that character. And then backtrack to the nodes that haven’t been yet travelled. This procedure will go on until the last node.

The string that I worked on is “MYIDISTWOONETWOTWO”. The unique alphabets of the string and their frequency are given bellow:

D = 1

E = 1

M = 1

N = 1

S = 1

Y = 1

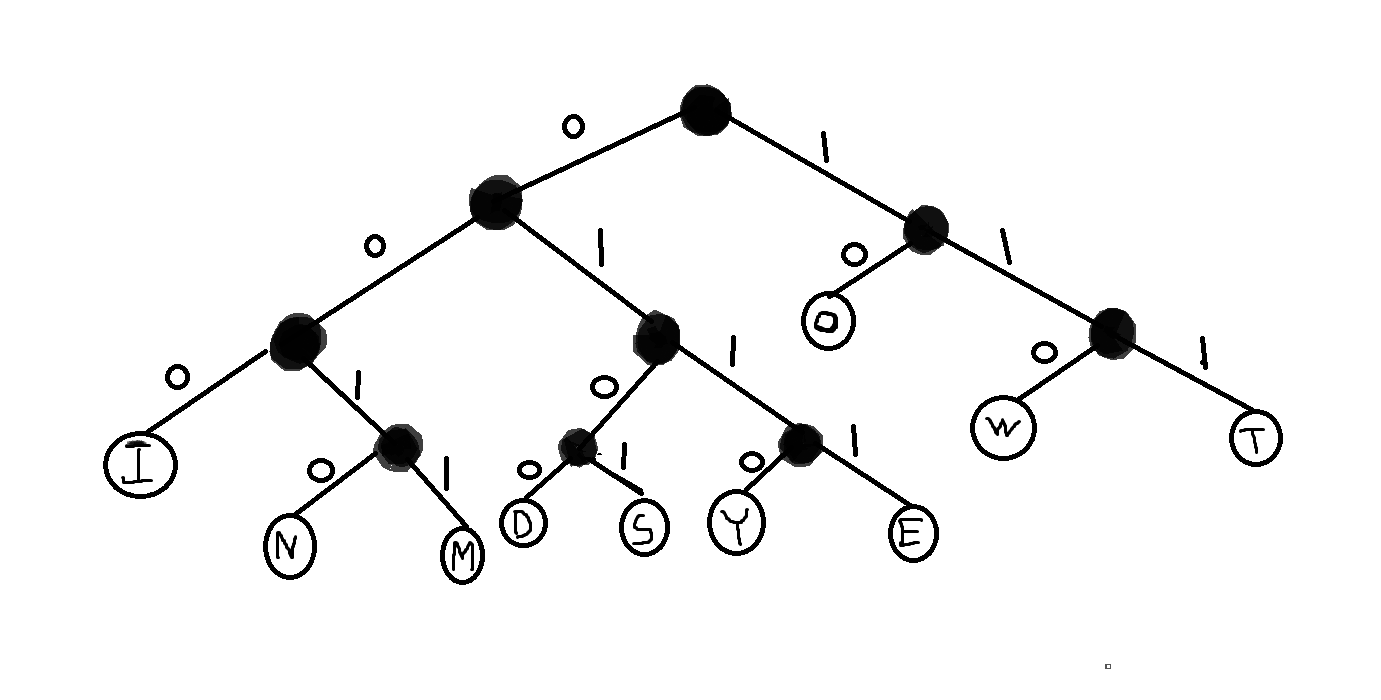
I = 2

T = 3

W = 3

O = 4

The following tree diagram represents the tree generated by the algorithm.



So from the tree we can also find the prefix code for the character of the string;

I = 000

N = 0010

M = 0011

D = 0100

S = 0101

Y = 0110

E = 0111

O = 10

W = 010

T = 011

So this is prefix code for every character of the string “MYIDISTWOONETWOTWO”.

==========================================================