Huffman code

prerequisite:
you have to have a solid understanding on priority queue and binary search tree(even though we won't work with the typical bst but having some knowledge on this topic is a plus).

Introduction:
Huffman is a lossless data compression algorithm. The idea behind the Huffman code is to assign a binary prefix based on the frequency of the element. The most frequent element get the smallest prefix and the less frequent element the smallest.

Implementation:
Initially we have to find the frequency of each character of the string and store them into a node array.

Before moving on to coding we have to understand how we should assign the binary prefix based of the frequency of the element. Huffman algorithm produce this binary prefix by creating a binary tree by taking two smallest frequency node and adding them up(this is a bottom up approach),i think it's best to show this in a figure.

Step 1: Frequency analysis
String : aabbbccccde

a = 2
b = 3
c = 4
d = 1
e = 1

Step 2: initializing the node array (optional)
Node array:

Character: a frequency: 2	Character: b frequency:3	Character: c frequency: 4	Character: d frequency:1	Character: e frequency:1

Step 3: setting up the priority queue
we have to insert this node array into a priority queue
This is how the node will be arranged after inserting them into a priority queue

Character: e frequency: 1	Character: d frequency:1	Character: a frequency: 2	Character: b frequency:3	Character: c frequency:4

Step 4: Making the huffman tree
Extract two smallest frequency character and add there frequency and character value and insert them into the priority queue. This process will continue until there is only one node into the priority queue.

Iteration:1

Character: a frequency: 2	Character: ed frequency:2	Character: b frequency: 3	Character: c frequency:4

Iteration:2

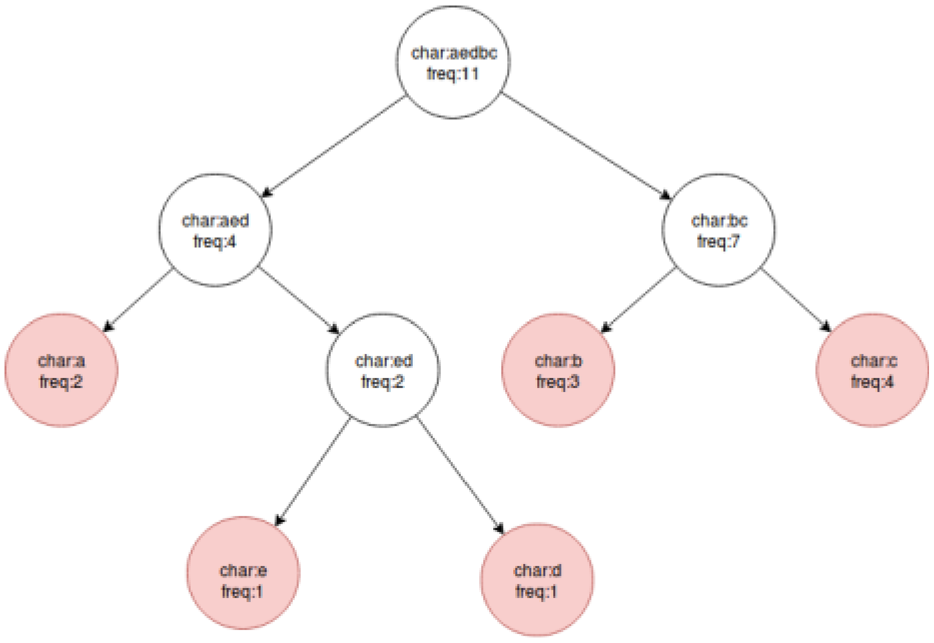
Character: b frequency: 3	Character: c frequency:4	Character: aed frequency: 4

Iteration:3

Character: aed frequency: 4	Character: bc frequency:7

Iteration 4:

Character: aedbc frequency: 11



This is how the tree will look like
Step 5: Traversing the tree