# Lab 3 Analysis-Haoyu Qi

## Description

In this project, I create my own program to realize the Huffman coding and decoding. I use Huffman tree to realize this function.

## Huffman Tree

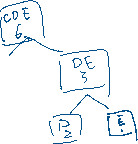
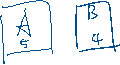
If we have N weights as N leaf nodes to construct a binary Tree is. If the weight path length of the Tree reaches the minimum, that kind of binary Tree is called an optimal binary Tree, also known as Huffman Tree. Huffman tree is a tree which has the shortest weight path length, and the node which has higher weight will be closer to the root.

## Justification of Utilizing Huffman Tree

If we code a sting like “I love you”, if we use ASCII Binary alphabet to code this string, each letter will have 8 digit to code, that will cost 8\*8=64 digits to code that, but if the sentence is very long with billions of letter, giving each of the letter 8 digits will cost much more space, plus, the most important is that some of the letter has very low frequency in the data stream, give each of them the same space to those has very high frequency’s letter to code seems uneconomic. So the Huffman tree shows its justification, each node of Huffman tree was weighted and the leaves stores the letters, and from the root of the tree, left child is 0, right is 1, till the pointer reach the leaves, and the 0-1 code is the Huffman code of the leaves letter, because Huffman tree is the tree which has the shortest weight path length, it will assign the high weight data a smaller length Huffman code, which will to spare the space when coding long string.

## Appropriateness of Application

To realize the Huffman coding, I first need to read the given frequency table, I designed a method to create the Huffman node based on the frequency table, then use insert sorting to sort the nodes generate a priority queue(from high weight to low weight). Then construct the tree by following principle:



Then I use recursive preorder traverse to get the Huffman code of each letter, then use Map<string, string> to store the Huffman code table. The input cleartext string can be compress to Huffman code with Map<string, string> get key. And the decompress of the Huffman coding is realized by traverse the string and use the tree, 0 is left child, 1 is right child, to get to the cleartext letter.

## Description and justification of your design decisions

1. I use reader/writer buffer to deal with each i/o files. And the frequency table is firstly formatting in order to assign the letter to the node’s data, and assign the frequency number to the weight.
2. Then use inset sorting to sort the nodes in order to create the Huffman tree. The sorting priority is weight-data length(i.e. node with abc will have more priority than ab), then alphabetical.
3. Create the Huffman tree with the upper method and preorder traverse the tree to gain Huffman code. Then used the method introduced upper to compress and decompress.
4. This Lab I realized many kinds of function, compress(cleartext to code), decompress(code to cleartext), print Huffman code table, preorder print the tree.
5. I try to apply some user friendly designed in the application, like file input guide, error warning text, as compact as possible output format. Also in this lab, it is very conscience for users to use different JAVA file to realize different function.

## Efficiency with time and space

I used the recursion to traverse the tree, so I will traverse each nodes once and that makes the traverse’s time complexity only influenced by nodes numbers, which is O(n). And the space complexity will be depends on the size of the recursion calls stack, which is the height of the tree, so the best space complexity is O(logN) and the worst is O(n).

I also use insert sorting, the best time complexity of insert sorting is O(n) but we all knows that the frequency sorting is far different with the alphabet. So the time complexity of this Lab is nearly O(n^2). But the space complexity is O(1).

## What I have learned

This lab really practiced my learning of Tree, recursion, and sorting. The recursion I used in this lab really spend me a lot of time, I tried to make the code more shorter. And as I did not good in last lab, I totally rewrote my code of the Tree, that helps me get more familiar with this kind of data structure. Why I adapt insert sorting is because it really sample and suit for small data, what we need to sort is just a 26 size of dataset.

## What I might do differently next time

Time is really limited for me. So I have lots of regret, like I can’t sperate the capital and small letter. Also, I think I will do better for the recursion use in the next time. Plus, I think this structure will be very useful for the protein data, because the amino acid frequency in organism is also disproportion, it seems that this kind of structure is very useful for this scenario.