Alexander Rodriguez

1. Discuss the computational complexity of the operations in the Huffman Algorithm, encoding and decoding, and traversing.
2. Say something about what you learned.

In my code, I used three main types of data structures. The first was hash table in terms of an array in the majority of my methods. This simplified counting the frequencies and storing string that were connected to specific characters. I used the ascii values of the characters as the index of the array and treated them as keys. In my getFrequencies method, I used this hash Table to connect the frequency of the characters to their specific character. The other two main data structures I used was a Huffman Tree and a Priority Queue. The Priority Queue caused me to create another class that I called the HuffmanComparator class that would compare values and decide where the nodes will go inside the Priority Queue. I used the Priority Queue to organize the nodes that I would be placing into the Huffman Tree beforehand. I would fist call my buildTree method and once the Priority Queue was filled with the leaf nodes, I would pull out the two smallest nodes from the heap, add up their sum and create an internal node. I would then add that internal node back into the Priority Queue and repeat this process until there remains only one node in the Priority Queue. This one node would be the root node to my Huffman Tree. I made a universal string array that I also treated as a hash table to save the binary values of each character while I traversed the Huffman Tree recursively in my traverseHuffmanTree method. This universal array was used again in my encodeFile method to save the binary codes in order to compare them to the characters in the file. Finally my decode method used the root of the Huffman Tree to find the leaf node that matches the order of the binary string. Below I added a table of the time complexity of my algorithms.

Methods	Time Complexity

getFrequencies	O(n)
	n = characters in file

buildTree	O(n + mlog(m))
	n = characters in file
	m = # of nodes

encodeFile	O(n + m)
	n = characters in file
	m = # of nodes

decodeFile	O(n + log(m))
	n = characters in file
	m = # of nodes

TraverseFile	O(m)
	m = # of nodes

I learned many interesting things on this assignment. I was able to learn how to use and apply Huffman Trees, Hash Tables, and Heaps. I also learned how to transverse a tree by using recursion in order to reach each node. In addition to this, I learned how an array can be treated as a Hash Table. I was also able to get a clearer idea of how to create the Huffman table by looking at the GeeksforGeeks[1] website that is referenced in certain areas of the code.

References

[1] https://www.geeksforgeeks.org/greedy-algorithms-set-3-huffman-coding/