*Discuss the data structures you used. Why were they appropriate? Discuss the computational complexity of the operations in the Huffman Algorithm, encoding and decoding, and traversing. Say something about what you learned.*

For this project I used two data structures, a heap, and a Huffman tree. The heap is appropriate for the project because I needed a way to order the leaf nodes from smallest to largest as the creation of the Huffman tree combines the two smallest nodes over and over from the pool of all nodes until it is all combined to a central node and a complete tree is created. I used a priorityqueue to implement this heap, as this was the easiest and most time efficient way to achieve the desired result. The Huffman tree data structure is obviously needed to encode the file as you traverse it to find the values of each character. It is also needed to decode the file as one would need to traverse the tree, going left or right depending on the sequence in the string one is trying to decode, until a leaf is reached in which case you found the character. One would then go back to the root and continue doing this until the end of the string.

The computational complexities of the operations in the Huffman algorithm is as follows in the table:

|  |  |
| --- | --- |
| **Method** | **Time complexity** |
| getFrequencies(File inputFile) | O(n) |
| buildTree(File inputFile) | O(n log(n)) |
| encodeFile(File inputFile, HuffTree huffTree) | O(n) |
| decodeFile(String code, HuffTree huffTree) | O(n) |
| traverseHuffmanTree(HuffTree huffTree) | O(n) |
| readFrequencies(File inputFile) | O(n) |
| createCodeIndex(String[] table, HuffNode node, String code) | O(log (n)) |

Over the course of completing this project, I was able to learn several things. First, I have never implemented a tree in any programming language before this, and thus this was a learning experience through trial and error. The second major thing I learned was comparing objects in java, for example, when I implemented comparable in the Huffman tree object. Before this, I did not know that this was possible and was thinking about using a sorting algorithm to figure out how to continuously get the lowest two weights to build the Huffman tree. The third major thing I learned is the creation of a heap using the built in priorityqueue of Java, this was also my first time implementing this type of data structure in any programming language as well.