Huffman Commentary

**Computational Complexity:**

* storingCodes – O(heightOfTree)
* huffman\_tree – O(n)
* get\_character\_code – O(1)
* encode – O(n)
* decode – O(n)

**Discussing the Data Structures:**

* storingCodes – This is a recursive function that places the binary path of the character into an array where the index represents the ascii code for the character. Initially I used a map with the key being the character and the value being the path, but because of the amount of accessing the encode function does I decided to use an array for quicker access.
* huffman\_tree – For creating the Huffman Tree I used a priority queue that uses a bool operator to sort by the frequency that the characters occur. I take the top two elements, create a new node and then pop those two elements from the queue and add the new node. This repeats until the queue has one element left which becomes the root node, and then is popped out of the queue.
* get\_character\_code – Because I went with an array where the index is representative of the ascii values; this function simple returns the binary path with the ascii value of the character being the index to access.
* encode – This function uses the get\_character\_code function to retrieve the binary path of the needed letter and concatenates it to a string. It returns that string at the end as long as all the letters needed where in the Huffman Tree, otherwise it would return an empty string.
* decode – This function traverses the Huffman Tree using the given numbers until it reaches a leaf, which it then concatenates that letter of the leaf to a string. It returns that string at the end.

**Would you choose a different implementation if you had to do the assignment again?**

If I had to do the assignment again, I would look for ways to reduce the space or time complexity of some of the functions. But overall, I believe that the approach I took was generally okay.

**What was the hardest part of this assignment?**

The hardest part of the assignment to me was receiving broad errors that encompasses multiple problems, ie. segmentation faults. These errors took a long time to debug and figure out where the underlying problem was occurring.