1 Encoding

1.1 Implementation Description

The first choice of data structure I made was how to implement what would become the elements of the Huffman tree. To do this I created a simple HuffNode class that contains four ata member. One char to store what character the node represents, an integer to represent the frequency of the character, and two HuffNode pointers to the children of the node in the tree. This seemed like the simplest way to create the tree that I would eventually need to generate in order to determine prefix codes.

The second choice I made about data structures in the implementation of the encoding part of the Huffman lab was how to store frequencies of letters. I chose to use an array of integers with size 256. I made this decision because it was mentioned in lecture that a simple way to count frequencies was to use the characters as indexes for an array. This structure allows us to check and update the frequency of any given letter in constant time and more complicated functions are not necessary.

The third choice was in regard to the heap that was used to construct the Huffman Tree. I chose to use a heap of HuffNode pointers. This decision was made partly because code for a heap was provided to us in the slides. Further, it was advised that we store pointers to objects rather than object themselves inside of the heap.

Next I needed a way to store the prefixes that were generated for each character. I chose an array of strings to do this because it's a very simple structure and by using the characters as indexes we can gte constant time lookup when converting characters to prefix codes.

Finally, I used an array of integers to calculate the total length of the encoded text. At the time this seemed like the simplest way to do it, but I realize now that using an array was not necessary here. The calculation could have been done without it and simply stored in an integer.

1.2 Efficiency

- Calulating Frequencies My implementation required one pass through the input file to determine the frequency of each character. For each character it increments a value in the frequency array, an operation that takes constant time. Therefore it has a running time of $\Theta(n)$ where n is the number of characters in the message to be encoded.