Ricardo Golac Barriga

Data Structures Project 2

The time complexity of Huffman’s algorithm is O(n log(n)). Inserting a new node with a heap in the tree requires O(log(n)) time and there are O(n) iterations. I created a map of all the characters and the string paths to the characters. Encoding and decoding took O(n) time.

In my program I used an int array of size 128 to keep track of all character frequencies. I then used the array to create a priority queue of node pointers containing character information and frequencies; this queue acted as a min heap. I then proceeded to build the Huffman tree by popping out the nodes from the queue, creating parents, and then pushing them back into the queue until I had one node left, the root of the tree. Using the priority queue made sense since it would always put the node with the smallest frequency at the top of the heap, allowing the Huffman tree to be built up. Using node pointers made traversing through the tree easy.

The hardest part of this assignment was figuring out the proper way to read in the information from the text files and creating the Huffman tree. I had to experiment with a few different methods to read in the text files correctly. Encoding and decoding was not difficult once the tree was built.