PA2_plan
Marvin Mui
ECE 368

Project Summary

This project is all about the reversal of one of the projects I worked on in ECE 26400. Where before I had to count the frequency of all ASCII characters in a file, make a priority queue of least to most frequent characters stored in binary search trees, and build the Huffman tree to encode the characters being written to a new file. Now, we have to read the file, check if the file is a Huffman tree, and rebuild the encoding tree and decode the file back to its original.

We'll be using binary search tree nodes and Huffman tree encoding. It should be able to take an encoded file and revert it into its original uncoded version. It write the original file, a file to hold the constructed Huffman tree, a file for the frequencies of all the characters, the topology information of the tree, and a file to hold the space needed by the encoding tree and a Huffman tree for the file.

Milestones

This project has a lot of different parts and arguments, so it is important to plan out the milestones of all the different parts accordingly. The preparation stage comes before the coding, to make all the c files necessary and to create a Makefile. The first stage would have to be the reading, writing, frequency counting, and binary search tree differentiation. Then we can start reconstruction the encoding tree and use it to decode the file in the second stage. Lastly, we need to be able to write all our other information into the other files, like the Huffman tree, the topology information, and counting how much space is used for each tree.

Timeline

The first stage should be completed by next Sunday, in 3 days. The second stage may use 5 days, as it is a large part of the assignment, and I have less working time during the weekdays, making the deadline on Friday. Finally, I should have all weekend to finish stage three, test, check for memory leaks and errors in 3 days before the assignment is due.

Assessment

Mainly, we will be checking using the diff command to compare the differences between the original file and the reverted file. They should be identical. We can also generate a sample expected file for argv outputs with a small test, or print out our expected outputs to standard error to visually confirm them. We will hopefully be able to use valgrind to check for memory leaks if I am able to install it into my Macbook.