Project 2 Milestone 1 – Ethan Glaser

This project primarily consists of 2 files – huff and unhuff. Each file essentially does the opposite of the other, huff takes a text file and compresses it using Huffman coding and unhuff takes a Huffman compressed file and decompresses it into a text file.

My Huffman coding process will be similar to the process outlined in the project document. My first step will be to scan the file and find the frequency of each character represented in the file, creating some sort of sorted array based on these frequencies. The next step will be to create a tree representing these characters, linking together the lowest frequency trees until a single tree has been created. Once this tree has been created the text can be converted into the binary representation using the tree. The last part of the huff file involves creating the header. My header will most likely be created using the pre-order tree traversal, with a 1 for each leaf node and a 0 for non-leaf nodes. Lastly, as pseudo EOF character will be generated, likely similar to the method outlined in the project document. The scanning and Huffman tree creation will likely be relatively straightforward but time consuming. The header generation will be a bit more complicated, but not take much time, and the end of file will likely be the most complex because it is something that is unfamiliar and seems to be a bit confusing.

The Huffman decoding file will cover the same things as the Huffman coder, just doing the opposite. The first step for the decoding will be to read and interpret the header, essentially generate the binary tree represented by the header file. As stated in the project document this can either involve reading in present characters and their frequencies or a pre order tree traversal with all of the characters represented as a node or a leaf node. Once the tree is made, the decoding of the file is possible, and each bit can be read and once a leaf node is hit, that character can be printed to the output file. This process continues for the entirety of the input file until an end of file character sequence is detected, or the end of the file is reached. The header interpretation should not be too complicated, especially if it involves a pre order tree traversal. Once the tree is created the decoding should be straightforward so that the file can be decompressed into text, although again the end of file part may be a bit more difficult.

I have some experience with Huffman and more experience working with zip files and accessing compressed files, but I never associated the two with each other. Obviously, Huffman coding is a preliminary compression method that primarily relates to text files, but I find it very interesting how the whole process works for compressing and decompressing a file. It is something I have thought about before but never really understood how it works, so I am looking forward to getting more involved with this project.