Augie Johnson

Huffman Compression

The code takes in a text file then reads in each char. Whilst reading it counts the chars density to be used to create a Huffman table, a sorted linked list of tree nodes that can be combined to create a binary tree. This tree acts a key to the file, both when encrypting and decrypting.

To encrypt the file, you re-read the file with the Huffman table searching through the tree and writing bits depending on whether you branch left or right to find it. In my case, right was 1 and left was zero.

To decrypt the file, you once again read a file, but this time the encrypted one. You take in a byte of data which was written by the encoder and search down the binary tree until you hit a leaf. If you hit a leaf then you print the char, reset your search to the root and continue to read through the bitstream.

To make this possible I added a .h file that includes a linked list of tree nodes and the tree data structure itself, as well as searches for the trees, ways to add nodes, and a method to merge to trees by adding a head then linking the two, and dereferencing the node in the linked list that was just merged.

You can run the code by just compiling then seeing the two files or adjust the names of the files in the main method if you want to encrypt something else and decrypt it to somewhere else.