Ethan Baird
ECB3AC
11/30/2019
Postlab 10

A description of your implementation. Describe the data structures used in your implementation and *why* you selected them.

For my implementation of decoding/encoding, I simply created nodes instead of an actual tree class, as I could link the nodes into a tree without the tree object created. So, my final implementation centered around creating the nodes themselves, and then having another process where these nodes are tied into a tree with the proper process. Then, while this occurs, the code or 'path' at which they are tied together is transcribed and planted into a map that can rip out the data and paste it appropriately.

When I decode, I do the same process with node creation, and then create the tree once more, however, I utilize the tree simply to act as a map to path the string inputs (001) and such to find the correct leaf node value.

For each of the steps of compression and decompression (see "Huffman Encoding and Decoding"), give the worst-case running time of your implementation.

I believe the time taken is log(n) as it is definitely not constant, and it is not linear. The average time of my encoder was 0.000163 seconds, and my decoder was 0.001102 seconds. I believe that my decoder was slightly longer as I was not able to HashMap the values in the end due to the tree implementation which may have slowed the process down.

2. In addition, give the worst-case *space complexity* (i.e. how many bytes of memory are used in each data structure) of your implementation.

Well, the compression ratio was 4.30769, and the cost of the Huffman tree is 1.85714 bits per character. Due to this, the special complexity of my structure set is linear, as it is not constant, growing as more inputs are given to the provided program.