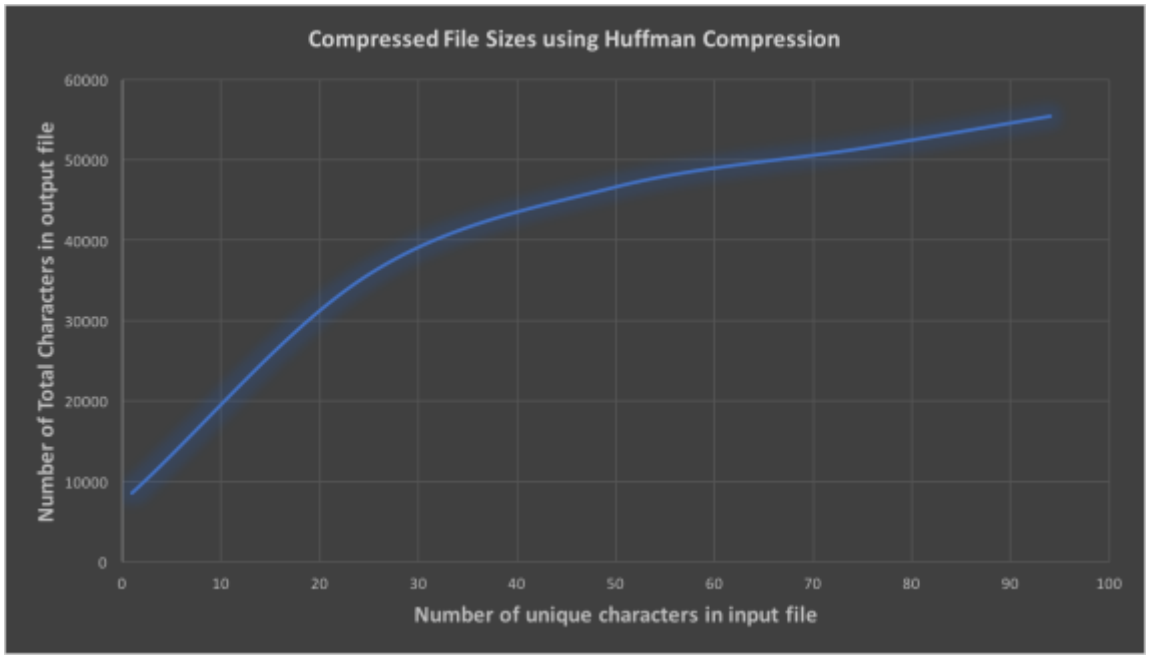
Kyle Price
U1018878
Assignment12
Analysis Document

1. Design and conduct an experiment to evaluate the effectiveness of Huffman's algorithm. How is the compression ratio (compressed size / uncompressed size) affected by the number of unique characters in the original file and the frequency of the characters? Carefully describe your experiment, so that anyone reading this document could replicate your results. Submit any code required to conduct your experiment with the rest of your program and make sure that the code is well-commented. Plot the results of your experiment. Since the organization of your plot(s) is not specified here, the labels and titles of your plots(s), as well as, your interpretation of the plots is critical. For this experiment, I made several files. Each containing 5_000 characters. The difference between them is the number of varying unique characters. They range anywhere from 1-94 unique characters. I then compressed each of them. I used the total number of resulting hex characters found in the output file as a unit of measurement. On the left side of the graph, you can see that that value, when 94 unique characters are used, shoots up to nearly 60_000! Which is nearly 12 times the number of characters contained in the input files. Clearly the compression rate increases as the number of unique characters increases.



2. For what input files will using Huffman's algorithm result in a significantly reduced number of bits in the compressed file? For what input files can you expect little or no savings? Files

that have a very low number of unique characters will benefit from Huffman Compression. As the number of unique characters increases, the size of the output file quickly follows suit.

3. Why does Huffman's algorithm repeatedly merge the two smallest-weight trees, rather than the two largest-weight trees? If the program were to merge the largest-weight trees, the bits with the highest frequency would go to the bottom and the bits with the lowest frequency would head to the top. The whole point of Huffman Compression is that the characters with the highest frequency take less bits to represent them, therefore making the resulting compressed file smaller. Merging the largest-weight trees would completely negate that.

4. Does Huffman's algorithm perform lossless or lossy data compression? Explain your answer. (A quick google search can define the difference between lossless and lossy compression). It performs lossless data compression. None of the characters are lost during the compression or decompression as there are never any approximations made about the data. All the original data is restored.

5. How many hours did you spend on this assignment? I spent around 7-8 hours on this assignment.