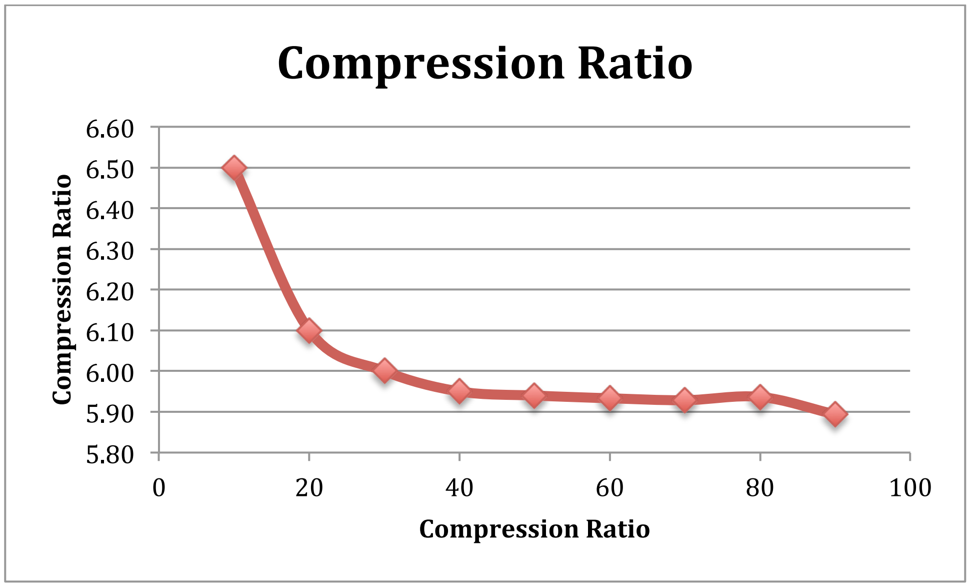
Jonathan Bown
U0696785

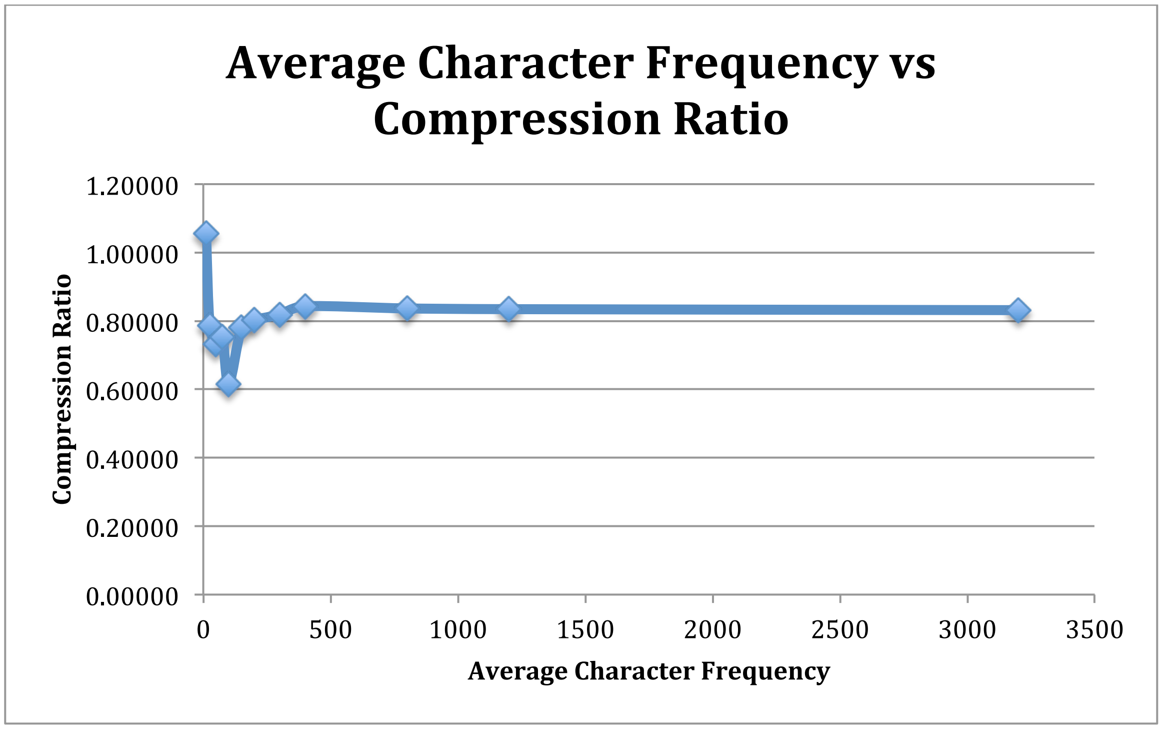
Analysis Document 12

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 chose to make 2 plots. For the first plot I created 10 files, they all have different counts of unique letters. I then plotted the number of unique characters against the compression ratio to show this behavior. As can be seen by the plot, the compression ratio does decrease as the number of unique characters increases, yet as the file gets more unique characters the asymptotic behavior levels out because the original file size and the compressed file size become more proportionally consistent.



-To show how the compression ratio behaves with frequency I used those same 10 files from the previous experiment but I repeated the lines so the average character count would be about the number of lines in the file. It didn't make sense at first but this behavior as the average character count increases is constant because Huffman compression is lossless compression. It has to preserve all the data. So even though it is reducing the amount of space that the file takes up it still will only remain at about 80% of what the uncompressed file size is because all the data is still contained within the compressed file.



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?

-Input files with not very many unique characters but higher frequencies of those few characters will result in lots of savings because the length of the binary sequence representing that file will be much shorter. Contrasting that with files with many different characters will end up resulting in little to no savings because Huffman compression performs lossless compression.

3. Why does Huffman's algorithm repeatedly merge the two smallest-weight trees, rather than the two largest-weight trees?

-This feature of Huffman's algorithm is essential to it's functionality. In order to maintain the proper weighting up the binary trie, it has to repeatedly merge

trees from the bottom up. This allows the proper Huffman code to be constructed by knowing if the items are to the left or right of each other. This also is efficient for algorithm complexity. By using the priority queue and selecting the minium-weighted items to merge into the trie, the algorithm only has to compare the weight at the current level of the tree or left most nodes instead of having to do other comparisons by building a trie from top down.

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).

-Huffman's algorithm performs lossless data compression because it reproduces the compressed data exactly as it was before. The binary trie stores all of the information that was contained in the file, so no data is lost when the compressed file is decompressed to original form whereas Lossy compression reproduces an approximation to the original data.

5. How many hours did you spend on this assignment?

7 Hours