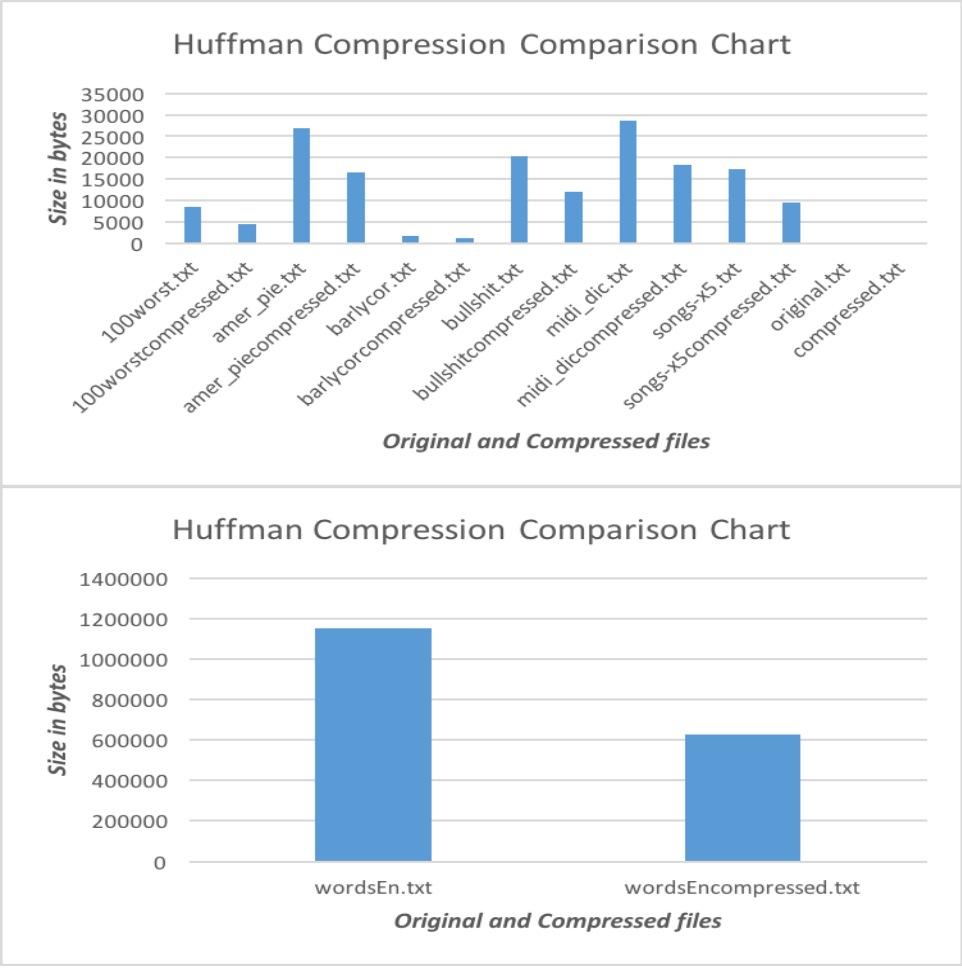
1. Design and conduct an experiment to evaluate the effectiveness of Huffman's algorithm. How is the compression ratio (compressed size / uncompressed size) affected by the number of unique characters in the original file and the frequency of the characters? Carefully describe your experiment, so that 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.ï¿½ï¿½ï¿½Plot 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.

After running the Huffman Compression Algorithm, I would get the info from the files and compare the size of the compressed file to the original file. After comparing the size I found that the compression to the file nearly reduced the file by half (give or take). For files that were short such as original.txt which just my name. This ended up being less efficient going from 13 bytes to 61 bytes.



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?

For an input file of 104 bytes, this seemed to be the threshold in terms of saving space. Anything over would result in a compressed file of lesser memory anything less would result in an inefficient increase in memory.

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

Huffman's Algorithm tries to return the most commonly used letters to the top of the trie based on priority. Giving the greatest weight (most common) near the root and the least common near the bottom. Doing this, in turn, will give you the most optimal solution in terms of assigning values to letters for the problem at hand.

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 algorithms perform's lossless data compression. Lossy is typically a data compression technique which amounts to some data being lost. Lossless refers to data compression techniques in which no data is lost. Graphics, audio, and video are some of the data that can tolerate lossy compression. For our .txt based compression data being lost would not be ideal.

5. How many hours did you spend on this assignment? Upload your solution (.pdf only) on the assignment page by 11:59pm on Nov 23.

I spent 15 hours on this assignment.