Cameron Pickle

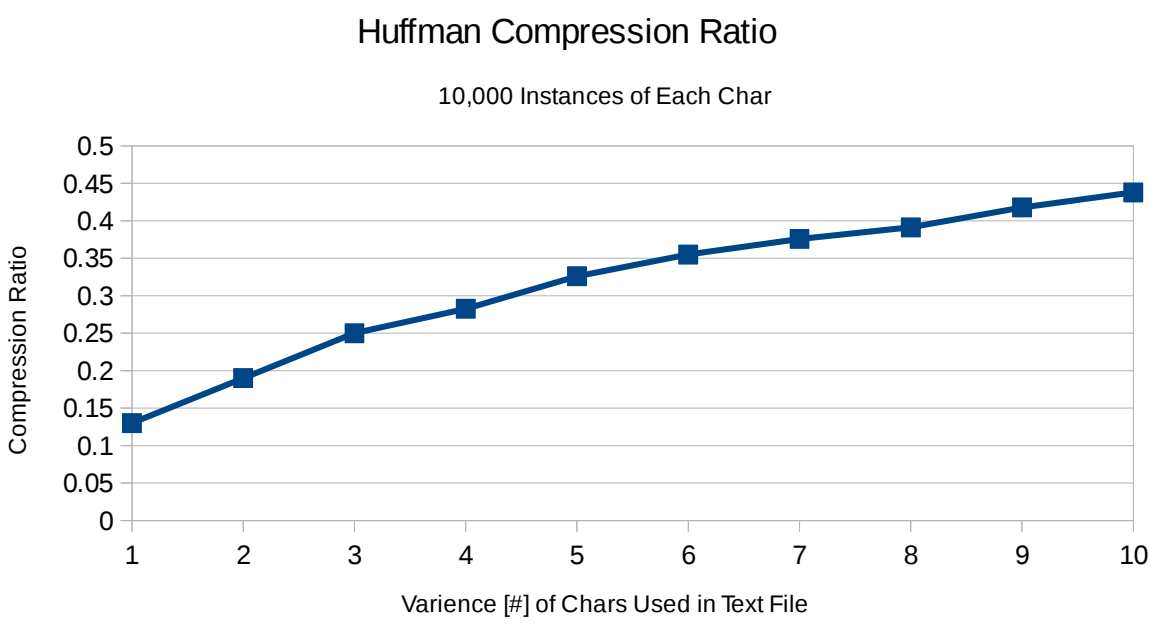
1. Who is your programming partner? Which of you submitted the source code of your program?

Daniel Avery. I will submit the code.

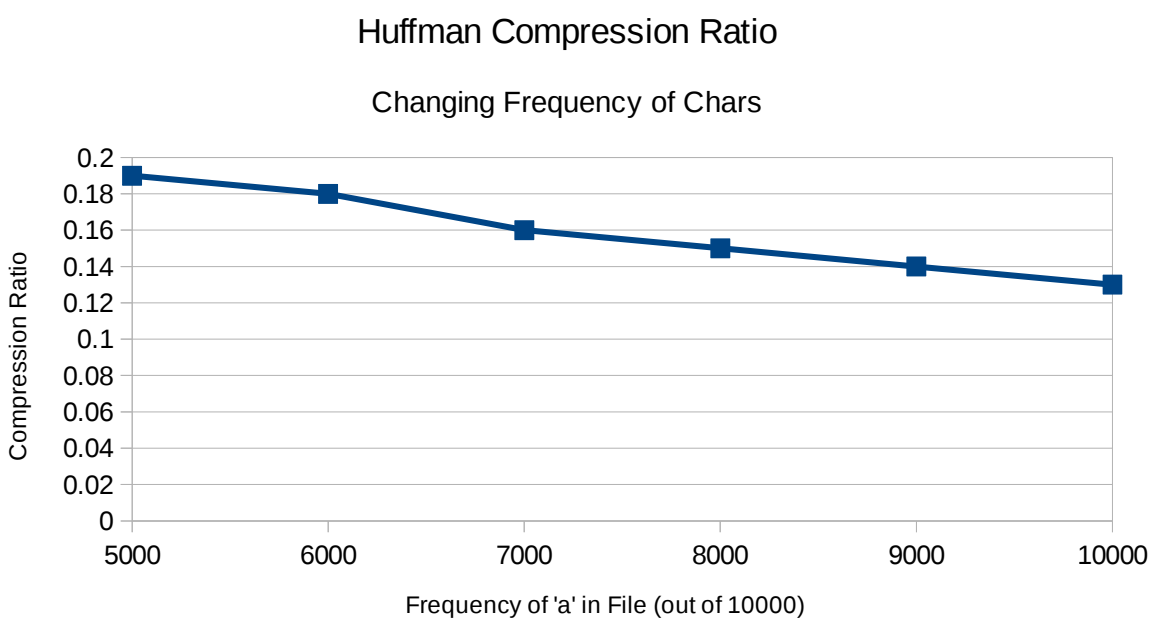
2. Evaluate your programming partner. Would you be willing to work with this person again?

He has been a good partner. He is good at taking time to analyze the code and make sure he understands what it is doing. I would be willing to work with him again.

3. 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.




We generated text documents and changing the number of unique variables with each 10,000 chars for each variable. Every text document we added 10,000 chars of a different unique variable so that in the end we ended up with ten unique chars each with a frequency of 10,000.




For our frequency experiment we had a total of 10,000 chars in each file. We started with an even number of 'a' and 'b' chars. Then in the following five files we took 1000 'b' chars out and added 1000 'a' chars until in the last one we ended up with all 'a' chars.

These charts show that the variance of the number of chars increases the size of the compressed file logarithmically which would be expected because the chars are represented in a binary tree making each new unique variable be logarithmically larger. The second chart shows us that the greater the frequency of a single specific char in relation to the rest the greater the compression rate possible.

4. 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?

The files that will result in the most significant reduction of bits in the compressed file is when there is little to no variety in the characters in the text file. This will allow the characters to be represented with the fewest bits possible. The files that would have little to no savings would be files that are very small or that have a large variety of characters all with a similar occurance rate.

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

If you were to start with the two largest-weight trees and merge to the smallest-weight then the final tree will be either right or left heavy and require more calculations to find the values of each character. By Merging the smallest tree at the top the first values would be either a subtree with only one item or a subtree with the rest of the items. If the smallest tree weights are merged first then the tree becomes more balanced and the routes to find the values are faster and therefore represented with less bits.

6. 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 allows us to perform lossless data compression. Huffman's nifty form of compression doesn't reduce the original data to compress it but rather stores in in a manner that requires less bits per character so that it represents the whole original file just in a smaller form. A lossy compression takes the original file and approximates chunks of data to make the reduction seem invisible or negligent to the user. The better the quality of the lossy function the less it will be noticeable that the compression has occurred.

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

We spent four hours on this project.