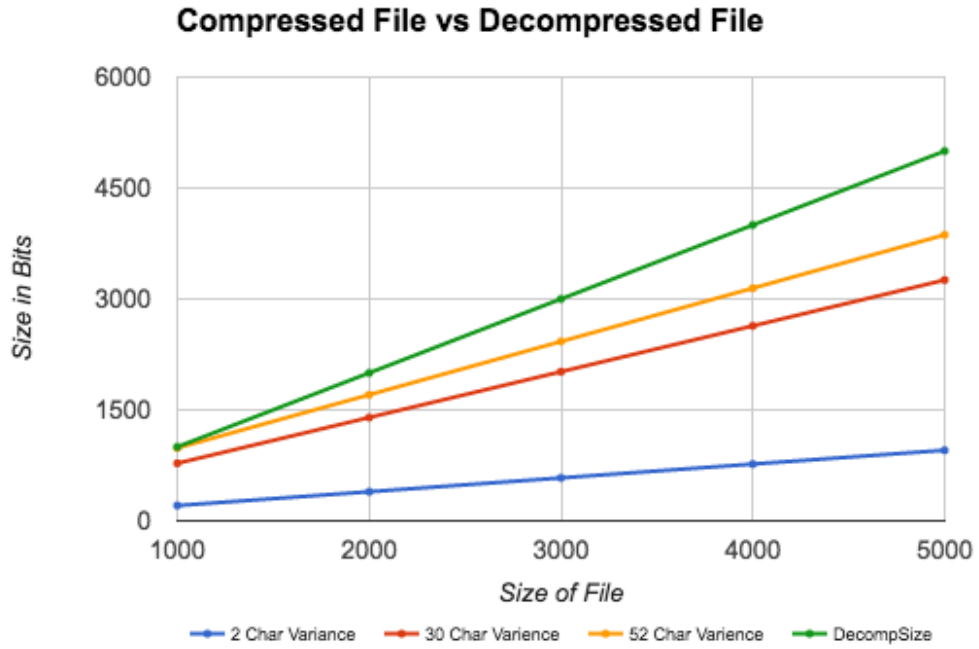
Chasen Chamberlain
u0583257
CS 2420
Assignment 12 Analysis

When you are satisfied that your program is correct, write a brief analysis document. The analysis document is 25% of your assignment grade. Ensure that your analysis document addresses the following.

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.

In my experiment I write to a file with random characters. The range of the unique characters vary from 2, 30 and 52. I then compress and decompress them to each other to determine the size differences. The size of the words grew from 1,000 to 5,000. I didn't do a huge growing file size because the data was clear enough and it took a long time to get each one.



To further clarify the green line is the decompressed file, so size without any Huffman compression, and then the 3 other lines are what the compression was like when allowing 2 different characters, 30 different characters, and 52 different characters.

This experiment turned out as expected. When my words had lower amount of unique characters it was more efficient to compress, while the more unique characters it had the bigger the compressed size. It is essentially good to have repeats a lot rather than have more unique characters in a 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?

File thats have a lot of repetition in their characters would do great such as anything that is mostly numbers. But for files that have a lot of unique characters there would be not much reduced number of bits because of them being very unique. I am not sure of a example of something that doesn't have a lot of repeats if its long because the longer a file goes the more chance of repetition.

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

If we didn't merge with the smallest I would imagine there would be almost

a reverse ordering issue. To me it logically makes sense to start the smallest and build to something large because the ordering would be building to the root, while if we started with the largest I would imagine the leaf nodes would end up displaced and just not good situation.

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).
As discusses in class Huffman's performs a lossless compression. You will lose no data so its good for things that need to keep everything like something that has important data or text. It would be terrible for audio or video because losing a little helps many different connections or devices be able to access/stream the content.

5. How many hours did you spend on this assignment?
8-10 hrs.

Upload your solution (.pdf only) on the assignment page by 11:59pm on Nov 23.