Scott Krstyen

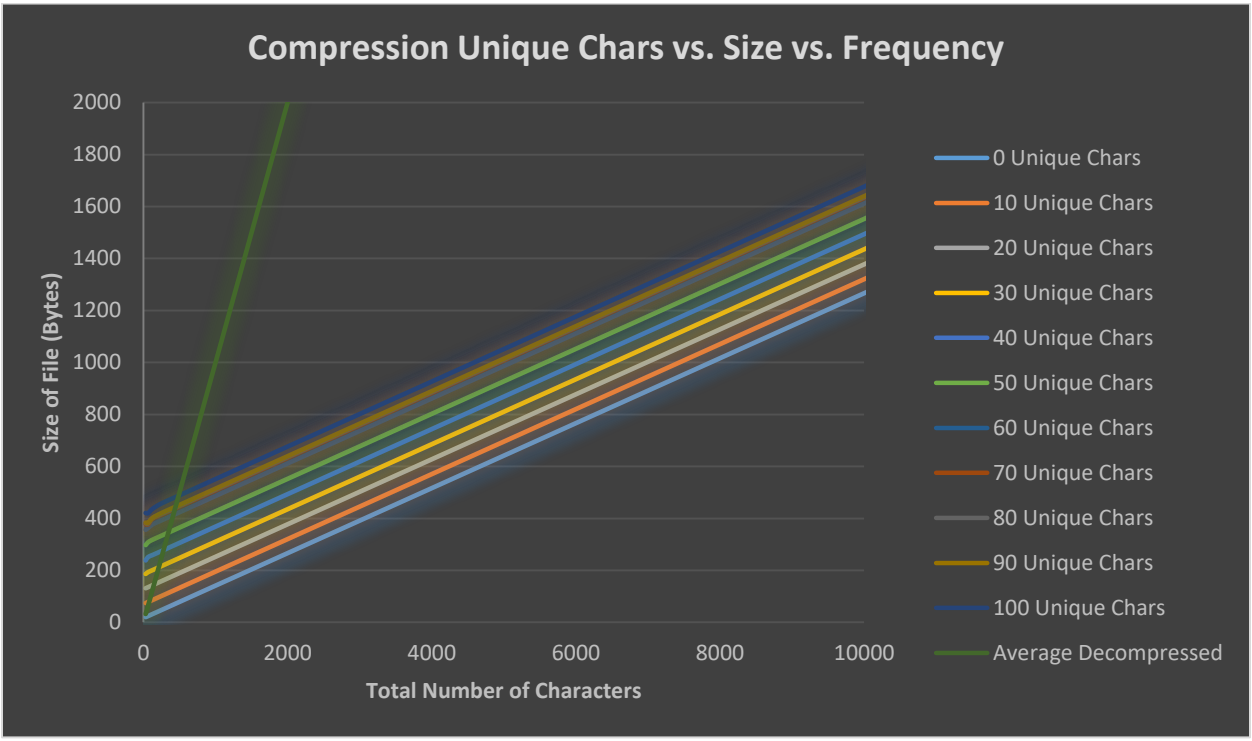
U0760822

CS 2420-001

November 23, 2016

Analysis Document: Assignment 12

1. **Note: I realized right before turn in time that my data is incorrect. I have thoughts on what went wrong below the graph.** Here is my graph of size comparisons. It is used to compare the total number of characters vs the number of unique characters vs the file size of the compressed file vs the average file size of the decompressed data. Basically the total number of characters was subtracted by unique characters and then used one character to represent that full sum of characters. The other few unique characters were unique. As you can see the fewer amounts of unique characters has the smaller the compressed file size. This makes sense because as it is compressed it can store more data using the repetitions. It also makes sense that at the very beginning decompressed data is slightly more efficient in size because there aren't enough repeated characters to make up for the size it takes to actually compress the file. I used an average decompressed file size depending on total number of characters because I noticed that regardless of the amount of unique characters the relative file sizes were very close so I decided to use an average instead of creating all of them.



Overall my graph shows that it is extremely more efficient to use Huffman's algorithm to compress the files to save on storage space.

Confusion:

Well thinking about it I just realized my graph is completely wrong because for 0 unique characters the file should be about constant or barely getting larger while the others should be much higher. I understand what went wrong in my code but unfortunately I can't create another one before the deadline. I guess this will be a lesson for me to start finishing my homework earlier. This is the problem. My tester would create a new file and then write to that file a certain amount of text depending on the amount of total characters and the amount of unique characters desired. Actually, I am not sure what went wrong but looking at the files it outputted, it definitely was not creating the right amount of unique characters. So my data is flawed but I definitely understand that the results should be different and I believe I understand what correct results should look like. The results shouldn't be as linear as mine are. Zero unique characters should be more extreme.

2. Input files that contain many of the same characters will result in a significantly reduced number of bits in the compressed file. Input files that contain many unique characters and don't repeat many of the same characters are likely to have little to no savings in the compressed file.
3. Because it makes the algorithm more efficient to make your tree that way instead of the other way around. For example, when you merge the two smallest-weight trees, less used characters in the file you are compressed are towards the bottom of the tree with more used characters at the top so it takes less space to make the steps to that point on the tree in the compressed file. If you did it the other way then most used characters would be on bottom and least used characters would be on top, which would be less efficient in the compressed file as explained above.
4. Huffman's algorithm performs lossless compression. Lossless compression means it can be compressed and then decompressed to the exact same size and file. Many lossless compression algorithms take advantage of repeated elements to minimize data size. Lossy data compression which is often used in videos, images, and audio can't be decompressed to the exact same file. Some data is lost.
5. I spent about 7 – 10 hours on this assignment.