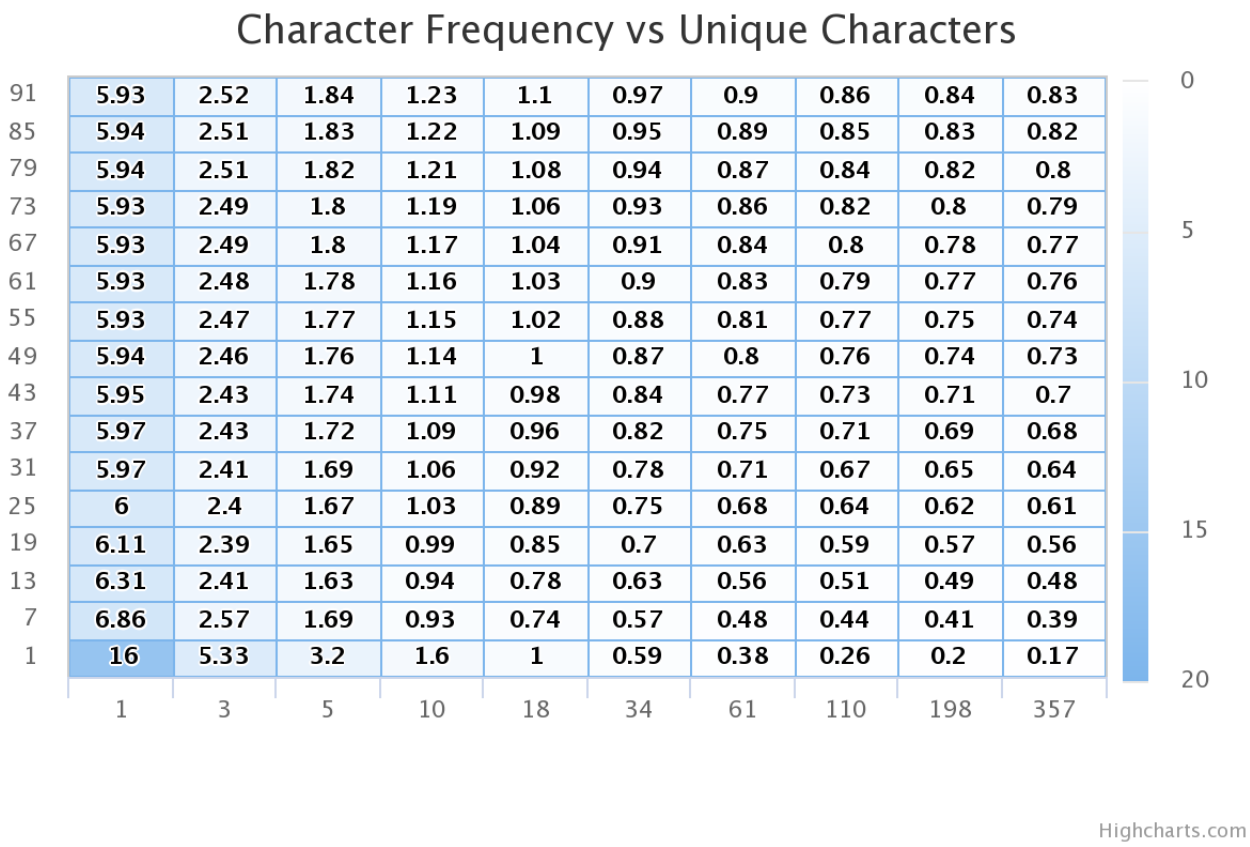
William Frank
11/23/16
Assignment 12 Analysis

1.



The above chart displays the ratio of the compressed file length compared to the uncompressed file length. The y-axis represents the number of unique characters in the uncompressed file and the x-axis represents the frequency of each character in the uncompressed file. To do this experiment I generated a grid of strings which had from 1 to 91 unique characters, and each unique character occurred between 1 and 357 times. For each amount of unique characters I then wrote these strings to a file, ran the Huffman compression algorithm on these files, then read back in the compressed data and compared the length of the uncompressed string to the compressed string. As we can see from this graph the file is more compressed the more times each character occurs, and the fewer unique characters there are. Towards the lower end of the character frequencies it appears that this trend is reversed, however this is likely because the effect of merely having more characters, unique or not, is outweighing the effect of having more unique characters.

2. Huffman's compression algorithm works best on very long files with very few unique characters. This allows the compression tree to be as small as possible and assign as few bits as possible to each unique character. Additionally the space taken up by the decoding tree will be increasingly negligible the larger the rest of the file is. This compression algorithm works the least well for a short file with a small number of very many unique characters. Lots of characters will create a larger Huffman tree, resulting in more bits being used for each character. Additionally, if the file is short enough, the information for the decoder tree alone could be longer than the uncompressed data.

3. To ensure that the most common characters use the fewest bits they must be the leaf nodes nearest the top of the tree. This is done by merging the lowest weight nodes first, allowing the higher weight characters to be on the same level as nodes of equal weight with lots of character sub-nodes.

4. Huffman's algorithm performs a lossless compression. Every character that is in the original file is encoded and can be recovered. No characters are lost or deleted, and the decompressed file should perfectly match the original.

5. 5–10 hours