Assignment 12 Analysis

Rahul Ramkumar

November 24, 2016

1. **Q:** 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.

   A: I designed an experiment that tested the effect on the number of unique characters by testing the compression ratio of a 1000 character text file from 1 character to all 93 visible ascii characters. To do this I used a PRNG to select the 1000 character string from the number of characters I was testing in each iteration of the test and then ran it through my Huffman's algorithm implementation and then used Java's File Library length() method to get the file-size in bytes of the original and compressed files to calculate the compression ratio as the size of the compressed file divided byt the size of the original text file. My results below show that when the number of unique Characters is increased, the Compression ratio grows as the algorithm performs worse when the number of unique characters grows.
   The next experiment dealt with testing the frequency of Characters in the file. Here I created a string from ascii value 32 to value 126 (all 93 visible ascii characters) then appended the string to itself N amount of times where N is the frequency of the characters I was testing. This way ensured the string in the initial test would have one of each character, then two of each the next test, then three, and so on. My results show that as the frequency of characters grows, the compression ratio lowers (i.e. performs better) and that for a frequency of 1 and 2 the compressed file size is around 5 times larger than the original file.

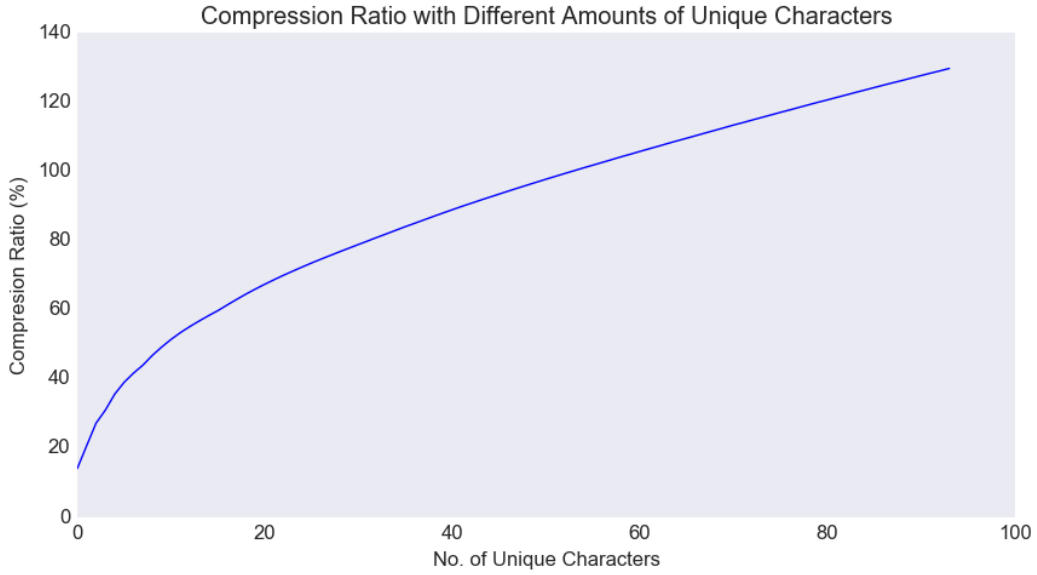

Figure 1: Plot of No. of Unique Characters against Compression Ratio

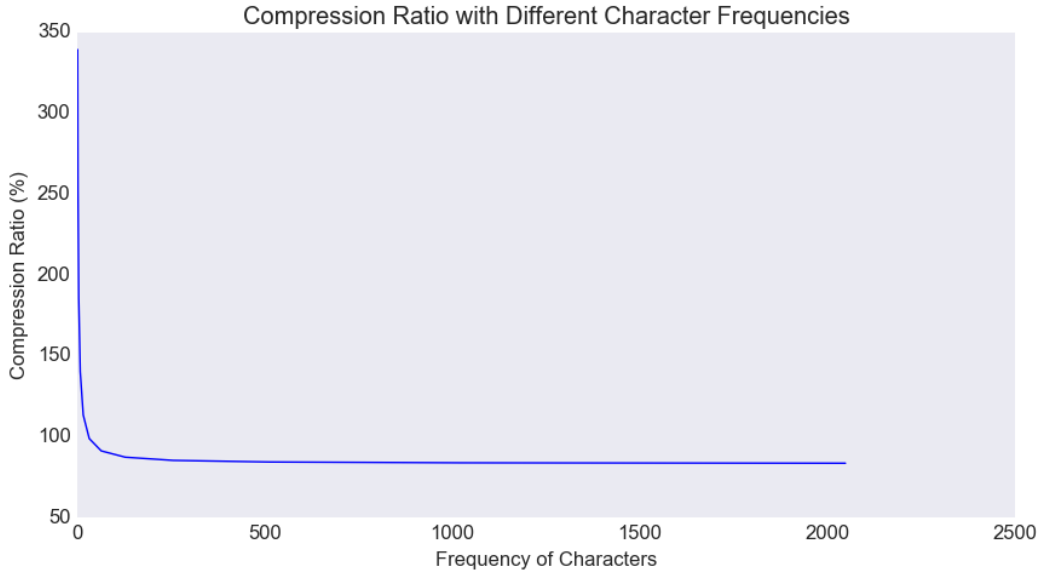

Figure 2: Plot of Character Frequency against Compression Ratio

2. **Q:** 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?

   A: For input files that involve low amounts of unique characters as well as high frequency of characters Huffman's algorithm will result in significant reductions in file size. For relatively small files with many unique characters this implementation of

Huffman's algorithm will result in larger filesizes due to the overhead introduced by the frequency table.

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

   A: Huffman's algorithm repeatedly merges the two smallest-weight trees because this results in a tree that has the lowest weight nodes closer to the bottom of the tree than the higher weight nodes. This is done because the more frequent a character appears in the original text the shorter we want its bitcode to be which means it needs to be closer to the root of the tree in Huffman's algorithm. The less the depth of any node in a Huffman tree, the shorter its bitcode.

4. **Q:** 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).

   A: Huffman's algorithm is an example of lossless data compression as the data output in the end is exactly the data that was input, no approximations are taken to compress the data like in lossy compresion. The decompressed file should be identical to the original file.

5. **Q:** How many hours did you spend on this assignment?

   A: I spent about 5-6 hours on this assignment.