Jess Terlato
CSC 202
Lab 7

Lab 7 Report

I. For the songs being compressed using a huffman tree, the length of the song (in words) is inversely correlated with the compression %. The compression % harshly falls within n<25, then slowly drops down until it reaches a relatively stable position at around n=75. The actual % value of the compression varies based on the song, but it's usually just under %50, as long as the n value is greater than 100. Letter frequency is also important, as greater frequency usually means an increase in compression %. Ideally, the string of text being compressed will have as few unique letters as possible, as having a small amount of unique letters will make the process much more efficient. This is because the fewer unique letters you have, the shorter your tree is, which makes compression much easier. Additionally, the frequency and number of unique letters affects how balanced a tree is. The ideal in this respect is a tree with an even number of unique letters and equal frequencies for each unique letter. Unique vocab words generally do not have an effect on the compression efficiency, unless they use uncommon letters like z or w, which may increase the number of unique letters and reduce the compression %.

II. The mantras' performance is generally not different from the performance of the song compression, barring a few notable exceptions I will be discussing here. If I do not mention something as an explicit exception, assume it is the same as described above. Firstly, one of my mantras had a compression % of 61, but this is largely due to its small n value of 16. Secondly, because mantras are generally shorter than songs, the amount of unique letters or characters will matter much more. For example, adding a ninth node to a tree with only eight nodes will have an adverse effect upon the compression time as the tree will have to be extended another level in order to fit the new node, making the process of navigating the tree that much longer.