Comp20290 - Assignment 1

Eanna Curran - 18311676

The goal of this assignment was to implement a functional utility for compressing and decompressing files. This was done using the lossless compression algorithm, Huffman's encoding. The implementation of Huffman's encoding can be found here.

As part of this assignment, we were also given a number of files to test the performance of our Huffman algorithm. The size of the compressed files, along with the time taken to compress and decompress the files can be found below.

File Name	Initial Size (kB)	Size Compressed (kb)	Compression Ratio	Time to Compress(Ms)	Time to Decompress (Ms)
genomeVirus.txt	6.3kB	1.6kB	4.2	59	4
medTale.txt	5.6kB	3kB	1.86	61	4
mobyDick.txt	1.2MB	667.7kB	1.79	202	63
q32x48.bin	192B	102B	1.8	44	2
beeMovie.txt	50.1kB	29kB	1.7	70	5

As shown by the results of compression, almost every file decreased in size by 35 - 45%. The only exception to this was genomeVirus.txt, where the file decreased in size by 74.6%. This was due to the fact that the file only contained 4 different characters which allowed the Huffman encoding to produce a very small bitstring of the file. The time that was taken to compress the files produced very linear results, with larger files taking longer to compress. The same can be said for the decompression of the files, except this procedure took substantially less time than compression. When traversing the Huffman trie, it takes less time to search for values using its value rather than its key, allowing for decompression to complete faster than compression. When attempting

to compress an already compressed file, it results in a file of the same size because the compressed file produces the same Huffman trie and thus the same resulting file. When comparing the compression sizes between Huffman encoding and Run-Length encoding the Run-Length encoding produced a compressed file of 143kB, larger than the Huffman compressed file. I believe this is due to the fact that a file in binary format is compressed using Run-Length encoding it only compresses efficiently for long instances of the same bit, resulting in larger compressed files.