COMP20290 Algorithms

Zach Dunne – 18416966

(Team members: Brian Byrne, Thomas Thornton)

Q1 and Q2.

File	**Original size (in KB)**	**Compressed size (in KB)**	**Time to compress (in ns)** average compression time	**Time to decompress (in ns)** average decompression time
Moby Dick.txt	1140	6	3,760,800	2,569,200
medTale.txt	5.5	3.01	980,200	867,500
q32x48.bin	0.192	0.102	19,000	18,300
genomeVirus.txt	6.1	2	98,700	101,400

File	**Compression ratio**
Moby Dick.txt	190:1
medTale.txt	550:301
q32x48.bin	32:17
genomeVirus.txt	61:20

The decompressed file sizes were identical to the original file sizes.

Q3.

When we compressed an already compressed file the file got smaller, but it became very lossy. We couldn't get it back to its original state. This makes sense because if files kept getting smaller and smaller without becoming lossy then a file that is a Terabyte in size could be compressed to 1 bit, which is obviously impossible.

Q4.

Run Length Encoding compressed q32x48.bin from 192 Bytes down to 159. This is a compression ratio of 64:53. We can clearly see that Huffman was a more effective algorithm than RLE. Huffman was 29% more effective than RLE.