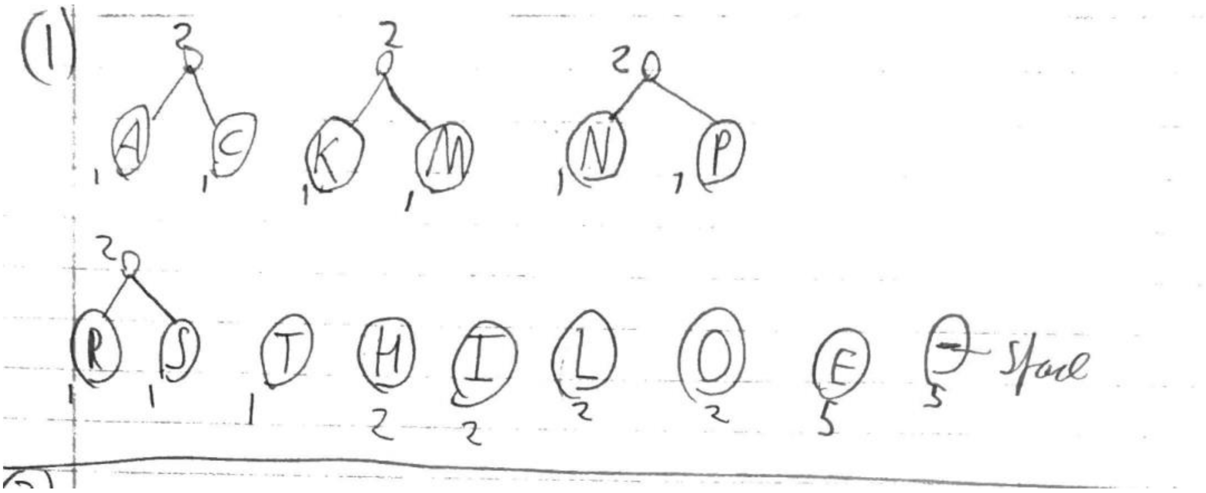
Brian Byrne

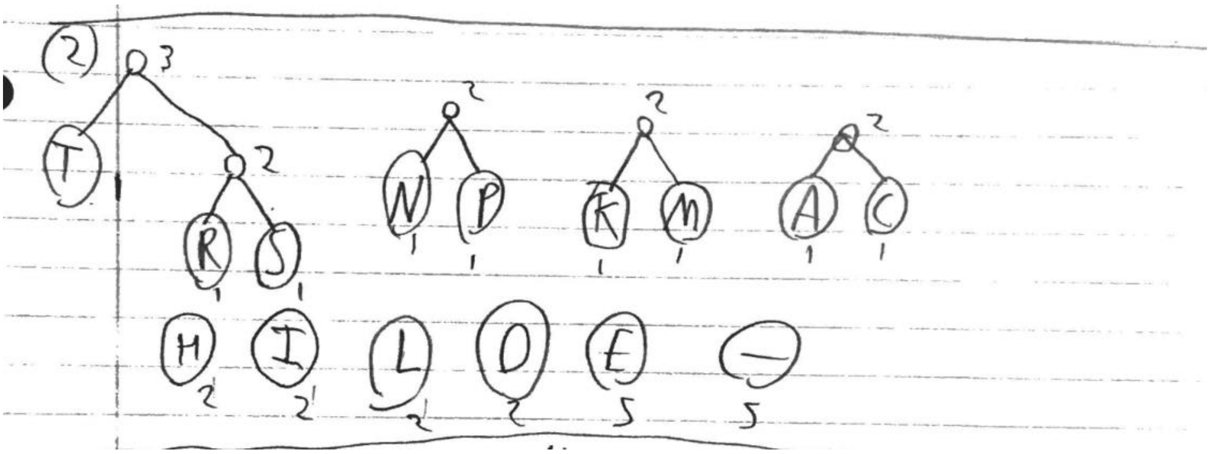
18391933

Algorithms Assignment

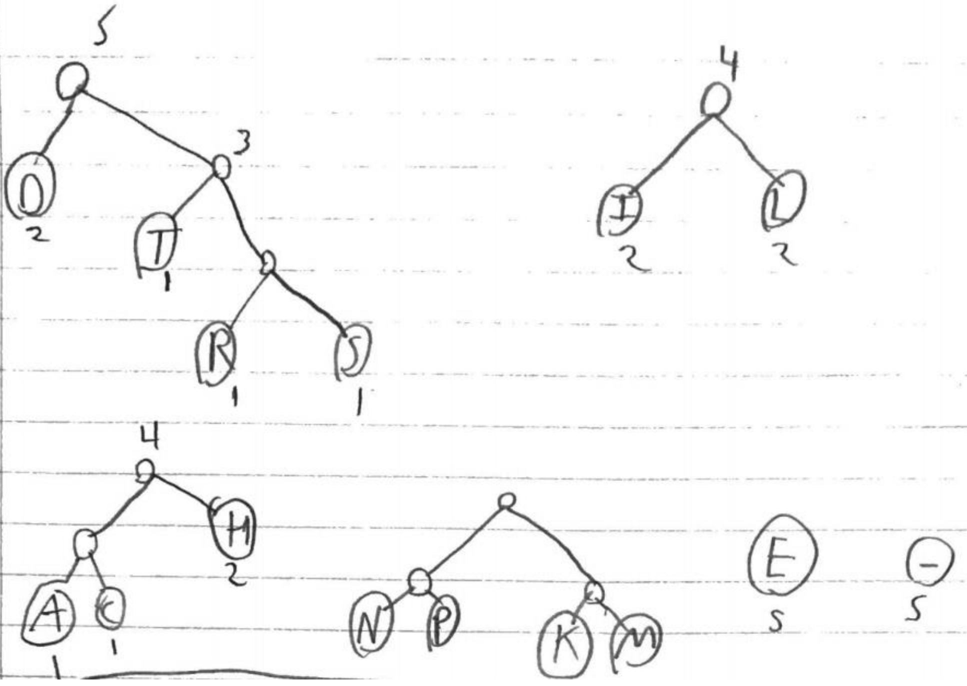
Team members: Thomas Thornton and Zach Dunne

Task 1)

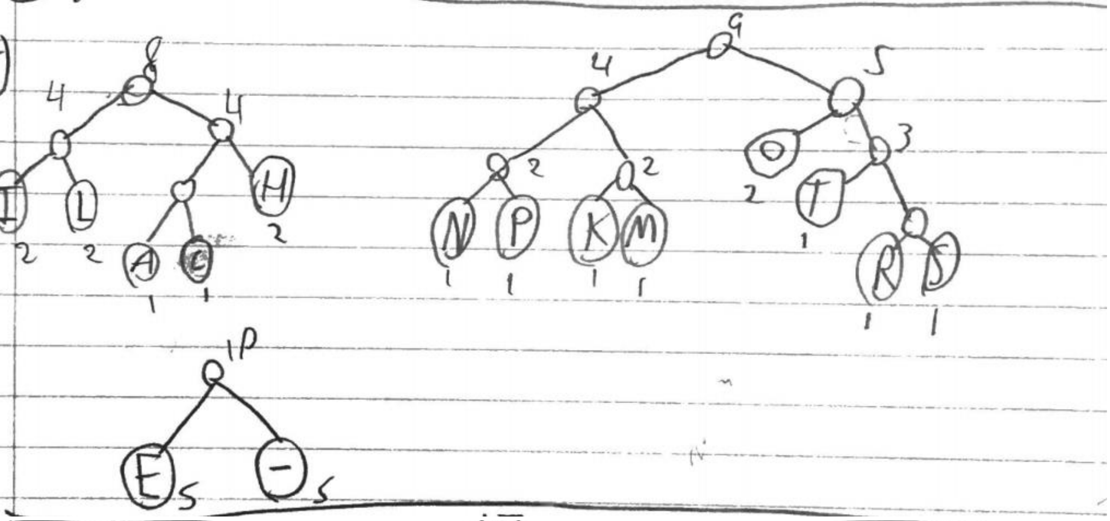




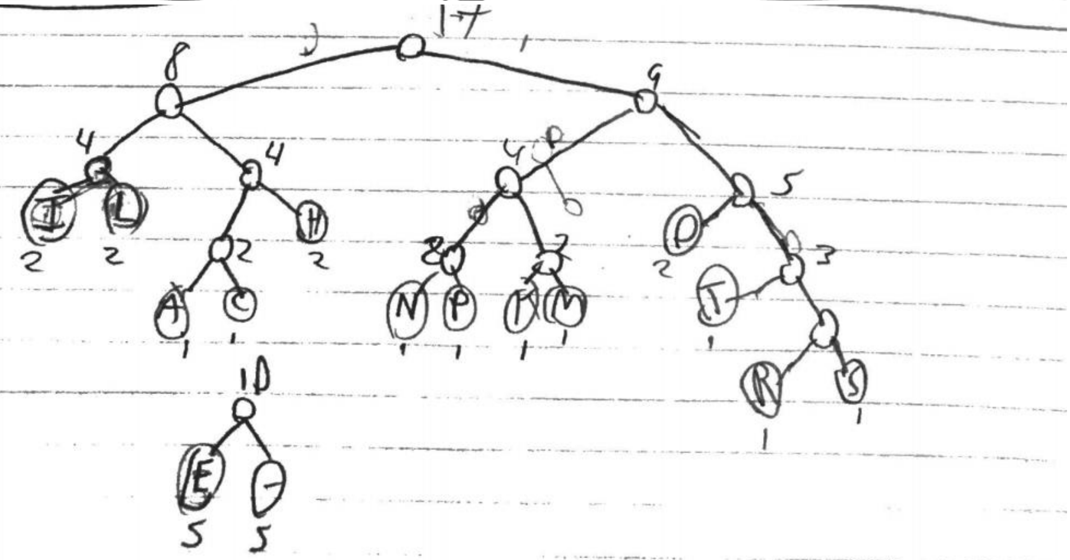
(4)



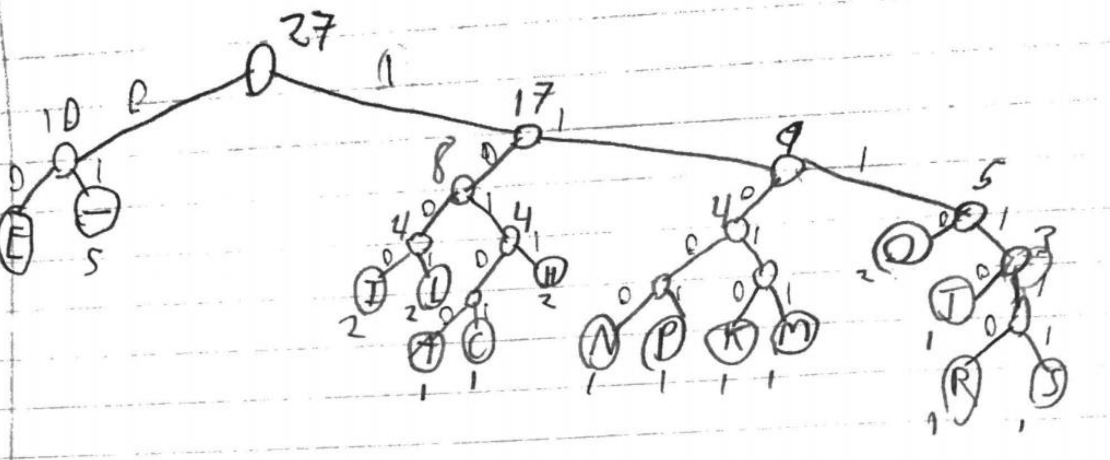
(5)



(6)



7)



Task 2) See GitHub classroom (brianbyrne99).

https://github.com/CompAlgorithms/algorithm-portfolio-20290-brianbyrne99

Task 3)

The following tables document a full analysis of our compression algorithms:

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	0.028	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	196:1
q32x48.bin	32:17
genomeVirus.txt	61:20

The decompressed files were the exact same sizes as the originals, resulting in accurate, quick, and lossless data compression.

3) Compressing an already compressed file resulted in losses of data. If lossless multiple compressions were indeed possible, then in theory any size od a file could be compressed down to bits, which in todays world has not yet been proven possible. To conclude, repeated compression increases losses of data.

4) Run Length Encoding compressed the q32x48 from 192 bytes into a 159 byte file. This is a compression ratio was 64:53. This is an example of how Run Length Encoding is slower than the Huffman compression algorithm, as it was 29% less effective than the Huffman, when compressing the exact same file.