Patrick O’Boyle – 19408716

Mohammed Eltayeb – 19349633

Tom Higgins - 19343176

COMP20290 – Algorithms

Compression Assignment

Algorithms Assignment:

Task 1:

Code Huffman Tree of Phrase:

"Home is where the heart is"

suppose the space character is represented as "_".

Frequency of characters:

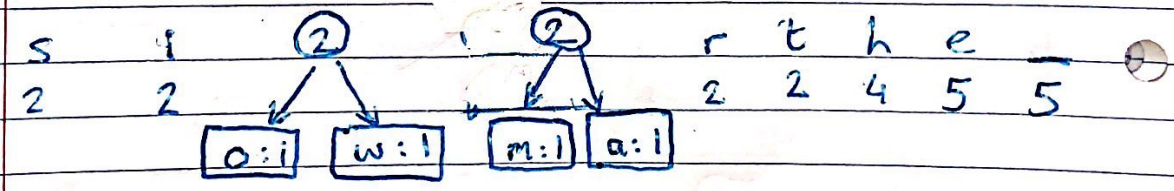
h	o	m	e	i	s	w	r	t	a	_
4	1	1	5	2	2	1	2	2	1	5

sorting characters in ascending order of their frequency:

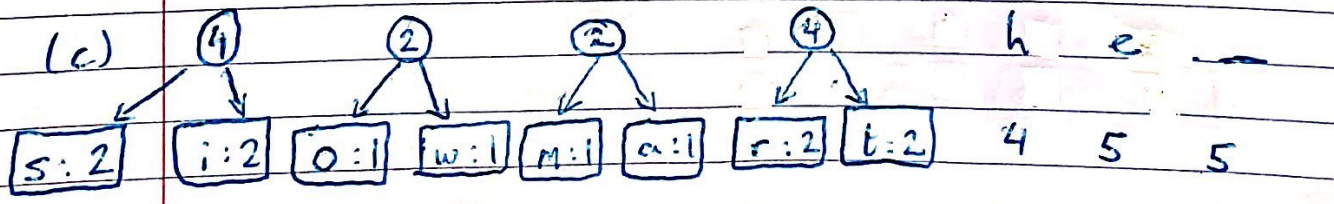
(a)

o	w	m	a	s	i	r	t	h	e	_
1	1	1	1	2	2	2	2	4	5	5

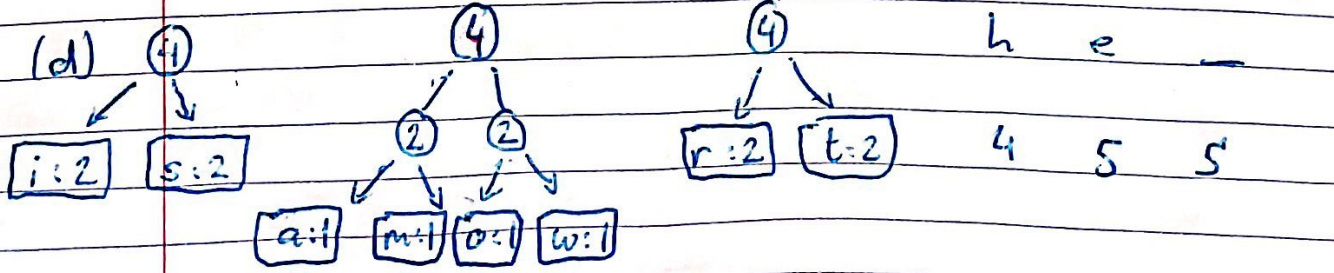
(b)

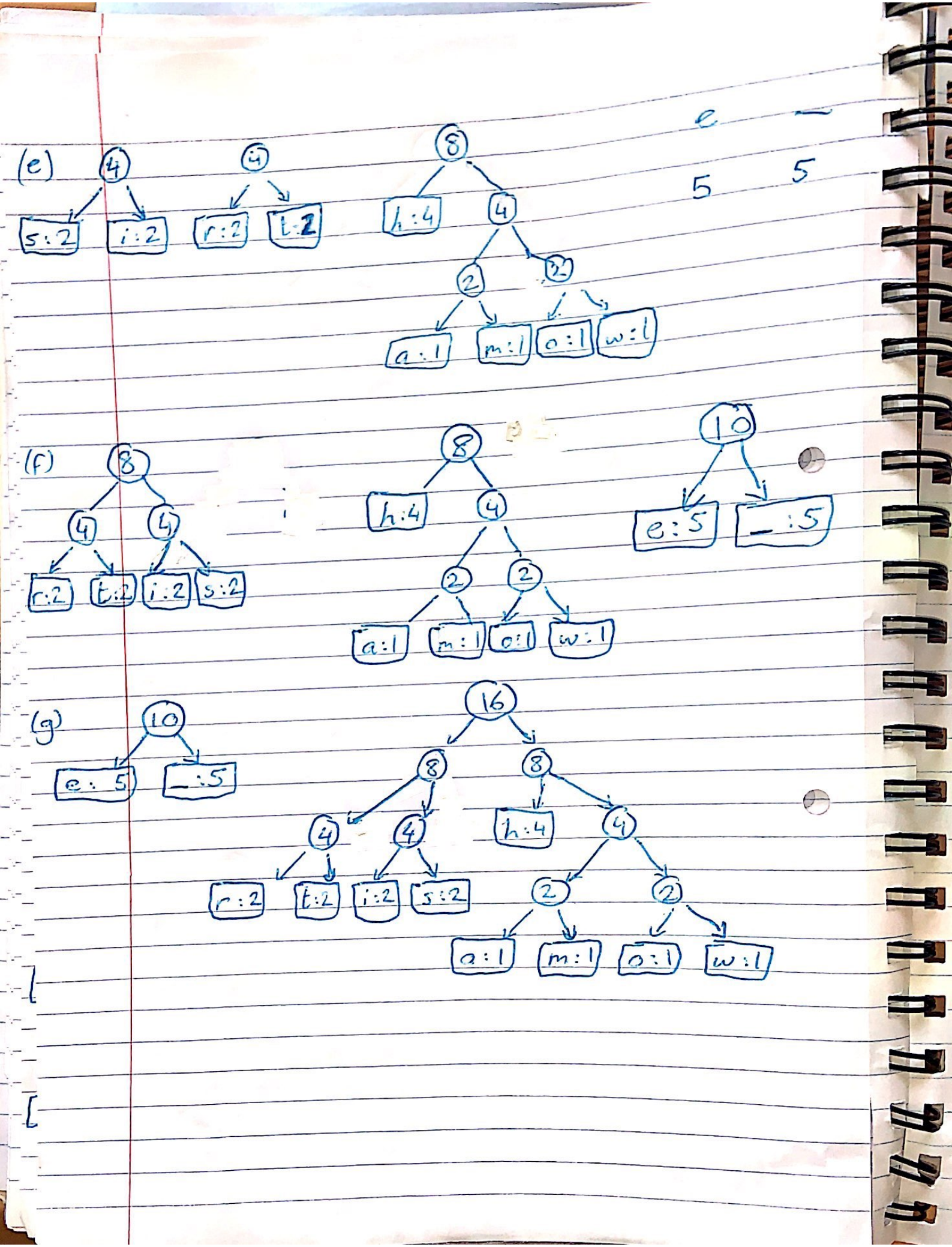


(c)

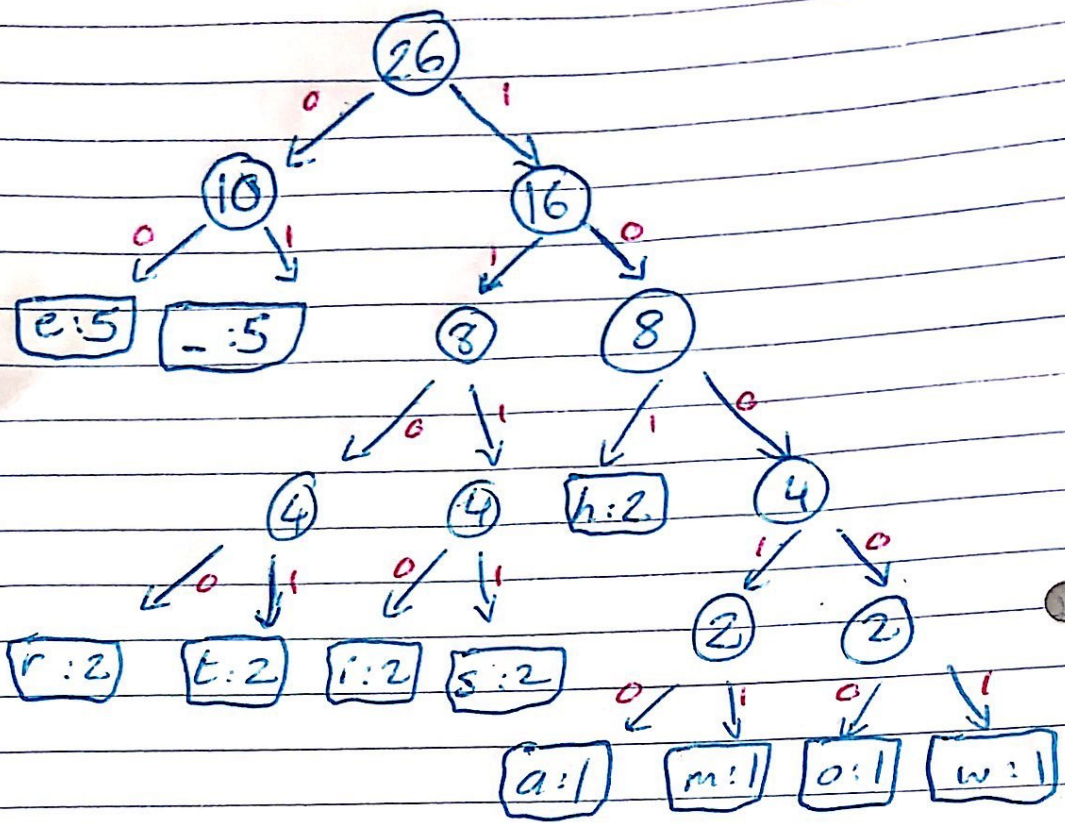


(d)





(h)



a → 1 0 0 1 0

e → 0 0

h → 1 0 1

i → 1 1 1 0

m → 1 0 0 1 1

o → 1 0 0 0 8

r → 1 1 0 0

s → 1 1 1 1

t → 1 1 0 1

w → 1 0 0 0 1

space (_) → 0 1

Task 3: Compression Analysis

Q1 + Q2:

File:	Time to compress:	Time To Decompress	Size before Compression:	Size after compression:	Compression Ratio
genomeVirus.txt	0.066s	0.0101s	50008 bits	12576 bits	74.85%
medTale.txt	0.0107	0.0106	45056 bits	23912 bits	46.92%
MobyDick.txt	0.234	0.1922	9531704 bits	5341208 bits	43.9%
q32x48.bin	0.00898	0.00664	1536 bits	816 bits	46.875%
AdditionalFile	0.0080	0.00747	4592 bits	2936 bits	36.06%

Q3:

Trying to run a compression algorithm on an already compressed file will run, however this can be dangerous for optimum compression. Most compression algorithms try to use the most optimized alphabet to achieve the smallest amount of entropy possible. Since Huffman algorithm is a lossless compressor sometimes compressing a second time actually increased file size. This is because of the decompression instructions stored to restore the file to its exact original. If the alphabet is optimised then the file cannot be compressed further since the entropy has been as reduced as it can be.

If the algorithm was a lossy algorithm multiple applications of the algorithm would degrade the quality of retained information while getting smaller since lossy algorithms exclude 'unimportant' data. However lossless algorithms will compress original data where possible and also include necessary transformation steps to return the compressed data to original.

Q4:

Original size for q32x48.bin : 1536bits

Compression via runlength: 1144bits

Compression via huffman: 816bitsbits

The runlength algorithm compression rate was 25.5% whereas the Huffman compression rate was 46.9%, a much higher compression rate from the Huffman algorithm. The runlength algorithm is not a competent algorithm when using files with many unique pieces of data. However Huffman encoding is quite a bit better at compressing data when there are numerous unique data bits in the file. An example of this, is a picture taken of something in real life. In a picture like this almost every

pixel would even slightly differ from every other pixel in colour. Runlength encoding is not an efficient algorithm for a file like this.