Assignment 1

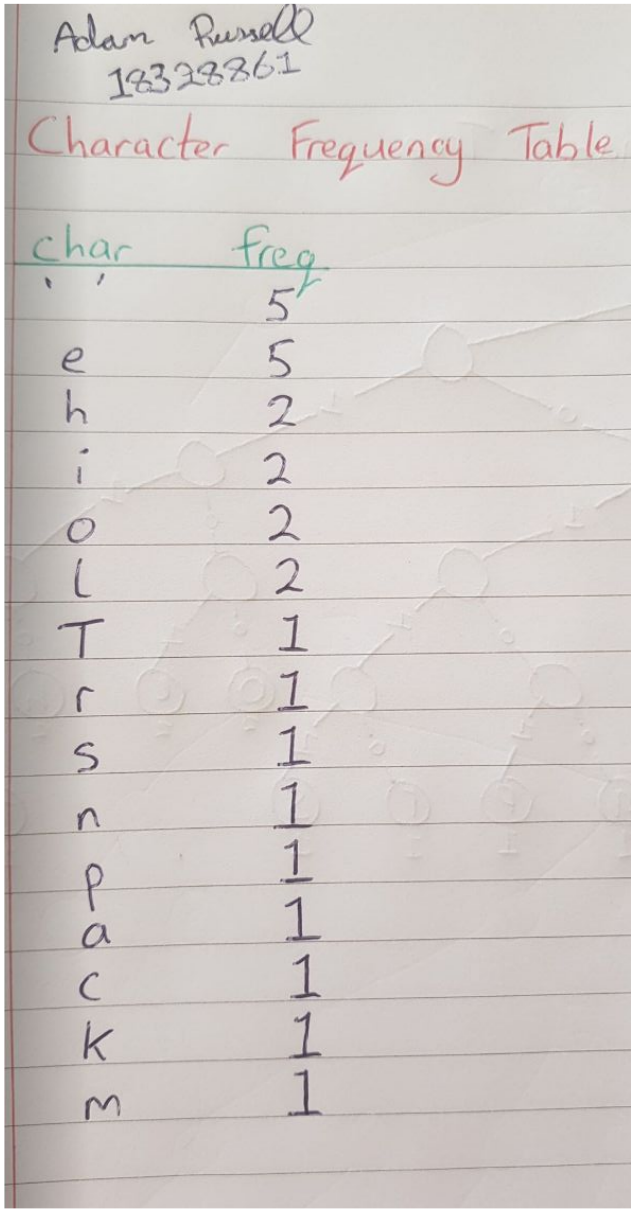
Huffman Compression

Adam Russell

18328861

Task 1 (Huffman Tree by Hand):

Character Frequency Table:

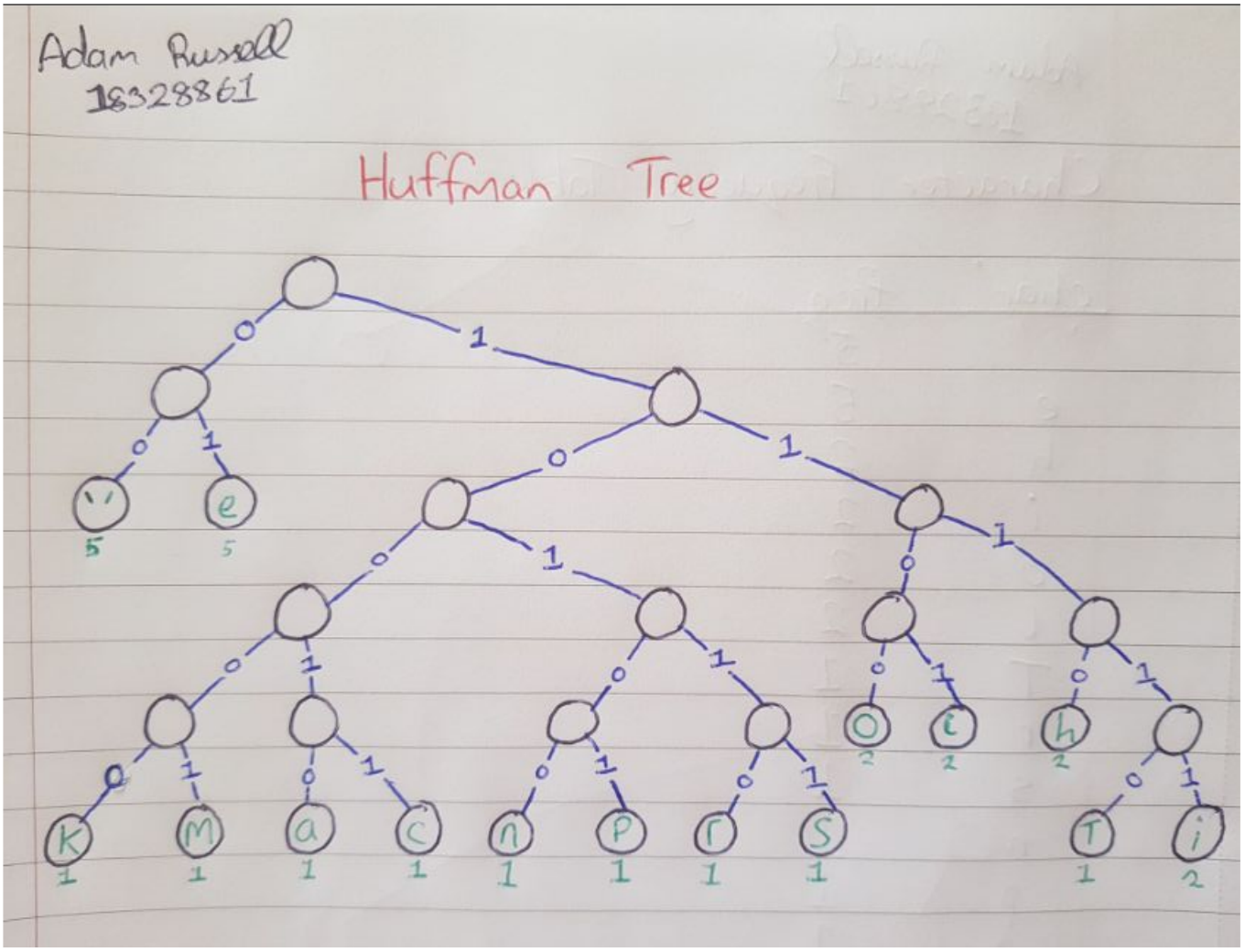


Adam Russell
18328861

Character Frequency Table

char	freq
' '	5
e	5
h	2
i	2
o	2
l	2
T	1
r	1
s	1
n	1
p	1
a	1
c	1
k	1
m	1

Huffman Tree:



Code Word Table:

Code Word Table

Adam Russell
18328861

char	code
' '	00
e	01
h	1110
i	11111
o	1100
l	1101
T	11110
r	10110
s	10111
n	10100
p	10101
a	10010
c	10011
k	10000
m	10001

There is no place like home

Original Message (BitString) = 216 bits:
0101010001101000011001010111001001100101001000000110100101110011001
0000001101110011011110010000001110000011011000110000101100011011001
0100100000011011000110100101101011011001010010000001101000011011110
110110101100101

Compressed Message (BitString) = 99 bits:
1111011000110110010011111101110010100110000101011101100101001101001
1011111110000010011101100100010 1

Compression Ratio:
99/216 = 0.46

<u>Task 3 (Huffman Algorithm Analysis):</u>

Compression:

	Time to Compress (Milliseconds)	# of bits (Original file)	# of bits (Compressed file)	Compression Ratio
genomeVirus.txt	31	50008	14008	14008/50008 = 0.28
medTale.txt	21	45808	24608	24608/45808 = 0.54
mobydick.txt	178	9531696	5505416	5505416/9531696 = 0.58
loremipsum.txt (5 paragraphs)	14	36776	19968	19968/36776 = 0.55

q32x48.bin	5	1536	816	816/1536 = 0.53
q64x96.bin	6	6144	2032	2032/6144 = 0.33

Decompression:

	Time to Decompress (Milliseconds)	# of bits (Original file)	# of bits (Decompressed file)	Original and Decompressed file identical? (Online file compare test)
genomeVirus.txt	10	50008	50008	Yes
medTale.txt	7	45808	45808	Yes
mobydick.txt	73	9531696	9531696	Yes
Loremipsum.txt (5 paragraphs)	7	36776	36776	Yes

q32x48.bin	4	1536	1536	Yes
q64x96.bin	5	6144	6144	Yes

Q. What happens if you try to compress one of the already compressed files? Why do you think this occurs?

I tried compressing two already compressed bin files again, and both times the number of bits in the doubly compressed file was slightly higher than in the original compressed file.

I think this may be because when encoding, the Huffman algorithm usually finds the most optimal (of many possible tries sometimes) trie to use to build codes for each character. When trying to recompress a compressed file, we are trying to compress text that has already been compressed as much as possible, so the only result can be slightly worse compression, which results in a slightly higher bit count.

Q. Use the provided RunLength function to compress the bitmap file q32x48.bin. Compare the results with your compression algorithm on this file. What reason can you give for the difference in compression rates?

Running RunLength Compression on q32x48.bin gives a compressed file containing 1144 bits. My algorithm gives a compressed bit count of 816. My algorithm treats each set of 8 bits as a character, which can allow really small encodings if we have a lot of similar characters, which can happen with long runs of 0's or 1's which would count as the same character over and over again. For example on the q64x96.bin file, my algorithm performed lossless compression from 6144 bits to 2032 bits. My algorithm successfully reproduced the original files from the compressed files.