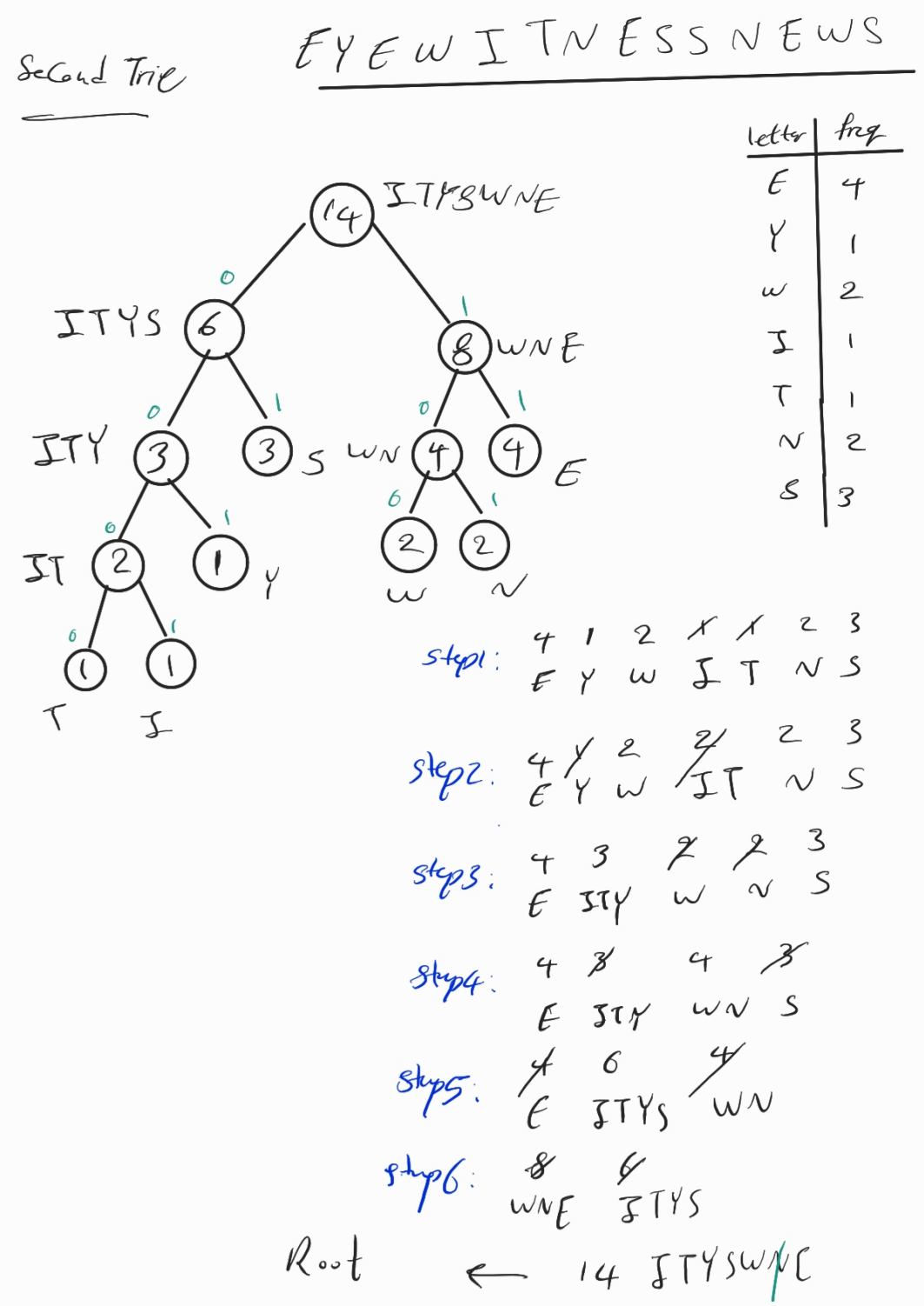
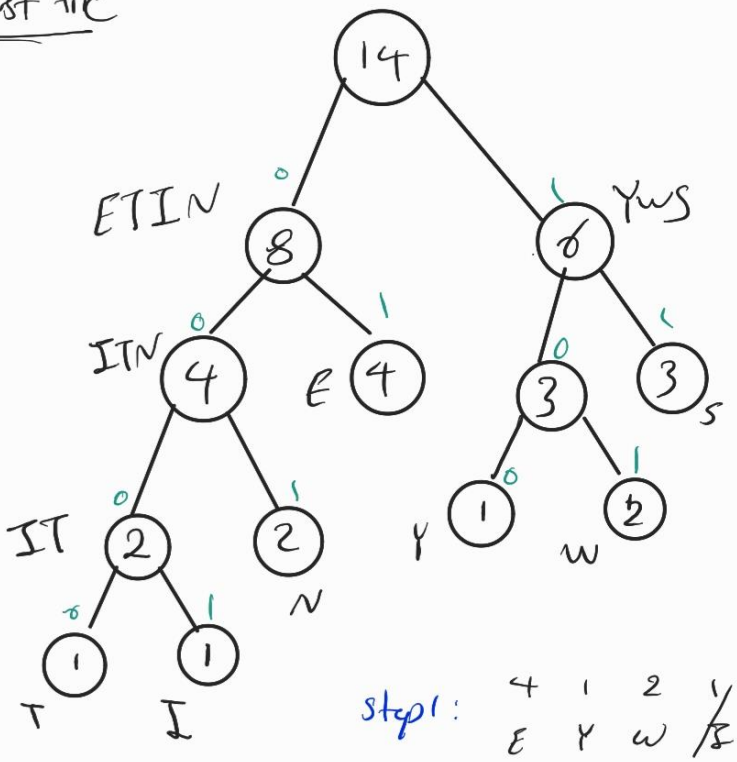
Exercise 1: part 1



EYEWITNESSNEWS

first trie



letter	freq
E	4
Y	1
W	2
I	1
T	1
N	2
S	3

step1: 4 1 2 1 1 2 3
E Y W ~~I~~ ~~T~~ N S

step2: 4 ~~1~~ ~~2~~ 2 2 3
E Y ~~W~~ IT N S

step3: 4 3 ~~2~~ ~~2~~ 3
E YW ~~IT~~ ~~N~~ S

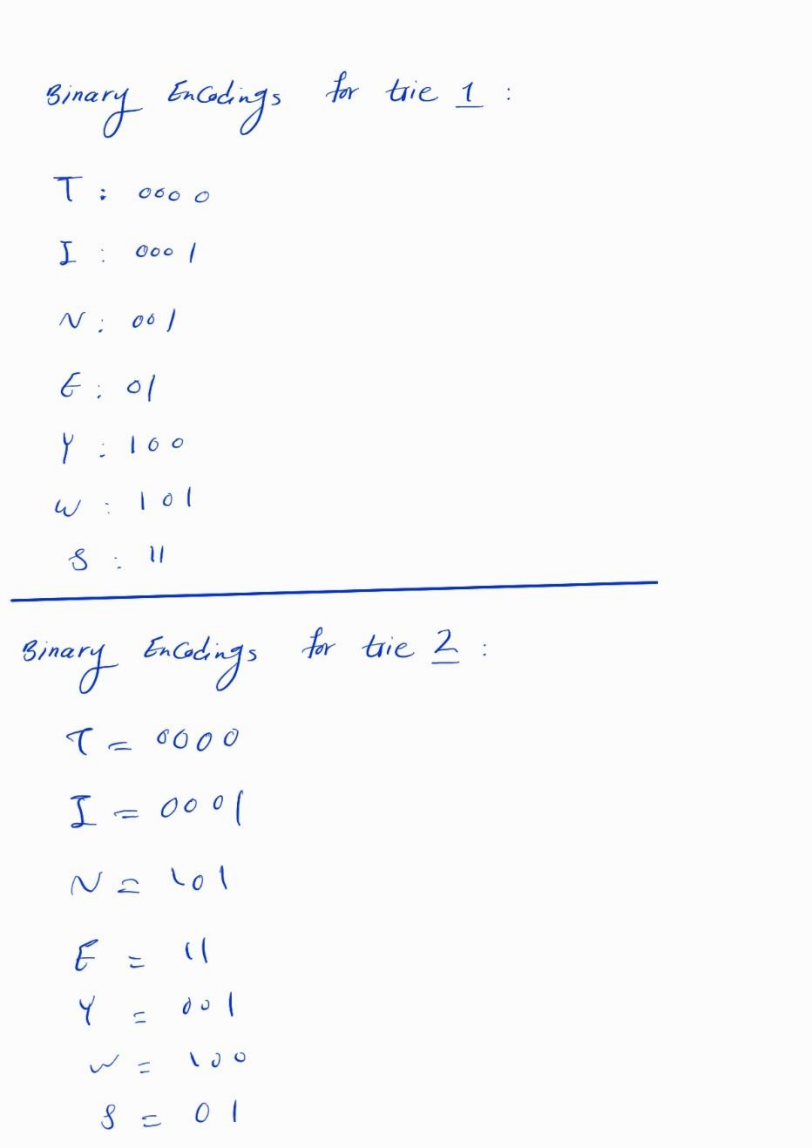
step4: 4 ~~3~~ 4 ~~3~~
E YW ITN S

step5: ~~4~~ 6 ~~4~~
~~E~~ YWS ITN

step6: ~~8~~ ~~6~~
EITN YWS

14 → Root
EITN YWS

Exercise1 : part 2



The number of bits of the two encoded messages is same, because we have same input data.

number of bits for trie 1 : 1*4+1*4+2*3+4*2+1*3+2*3+3*2=37

number of bits for trie 2 : 1*4+1*4+2*3+4*2+1*3+2*3+3*2=37

Exercise 1: part 3

As the number of bits of the two encoded messages is same, there is not best trie and it does not depend on our choice in building the trie, because no matter what how the shape of the Huffman trie is,

the result of encoding (the number of bits of encoded messages) will be same. Because in fact what does change number of bits of encoded messages is the frequency of inputs (letters here)

Exercise 2 -1: if we have these frequencies for the characters, we can have maximum height:

A:1

B:2

C:3

D:4

E:5

F:6

G:7

H:100

Exercise 2-2: if we have these frequencies for the characters, we can have minimum height:

A:16

B:16

C:16

D:16

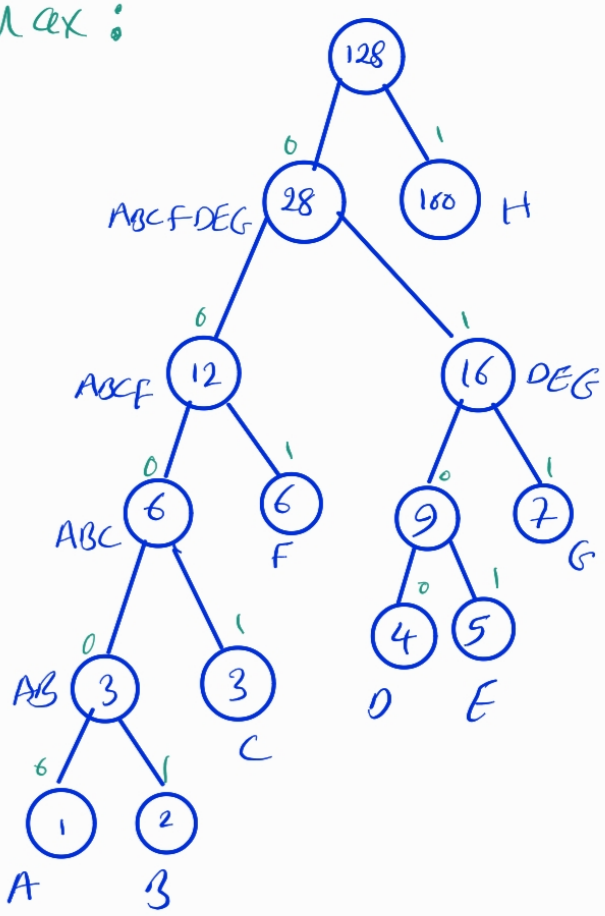
E:16

F:16

G:16

H:16

Max:



A : 00000

B : 00001

C : 0001

D : 0100

E : 0101

f : 001

G : 011

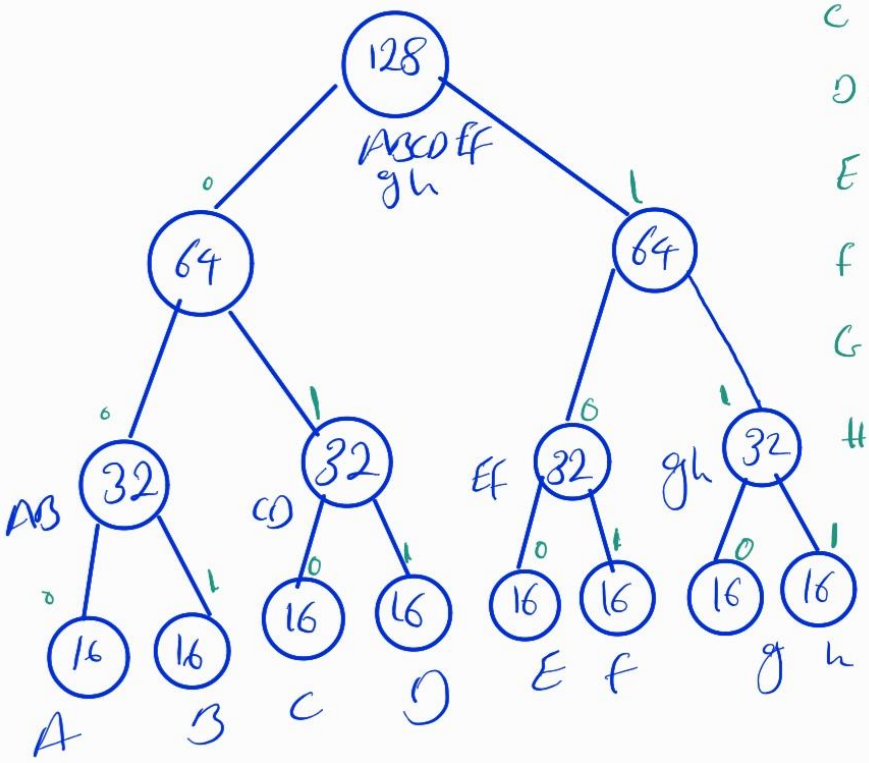
H : 1

number of bits = $1*5 + 2*5 + 3*4 + 4*4$

$5*4 + 6*3 + 7*3 + 100*1 = 202$

Min:

A : 000
B : 001
C : 010
D : 011
E : 100
f : 101
G : 110
H : 111



number of bits = 16*3 + 16*3 + 16*3 + 16*3

+16*3 + 16*3 + 16*3 + 16*3

= 384

Exercise 3 : I think theoretically it can be possible that an algorithm be able to do 10% compression, but we need to have more information about this, because when it says **any** file makes it harder because there is a very big variety of data type, size , characteristics to determine if the algorithm can work for any data (file). Also, beside the variety of data type and size, there many other factors such as nature of data, how much the algorithm is effective and some trade-offs beside which can affect the compression of data. For example, decompression speed and amount of loss of data in compression is important that need to be take cared of.

Exercise4-3: to find the time complexity of makeHuffmanTree we consider that :

Computing the frequency is O(n) n is length of string

Initially creating the priority queue is O (d logd) d is number of character

Creating the Huffman tree in worst case is O (d logd)

So in overall, approximately, it can be: O(n)+ O(d log d) = O(n + d logd)

Exercise 4-4: to find the time complexity of printTrie we consider that :

The time complexity of the 'printTrie' function depends on the number of distinct characters .

The time complexity is O(d), because by having recursive calls, function traverse all the nodes in the tree, the total number of nodes in the tree depends on the number of characters, which is 'd' so the time complexity of recursive calls is O(d). also, the function prints the leaves when it reaches the leaves, which is also O(d) (printing is constant, but because it prints all the leaves is O(d))