Huffman coding – Greedy Algorithm
:is a compression algo that uses greedy choice property to compress data so that it doesnt loose any information.
Huffman coding is data compression algo, that drives to compress data in a lossless form.

Based on lengths of assigned codes based on frequencies themselves.
Variable length codes are known as Prefix codes.

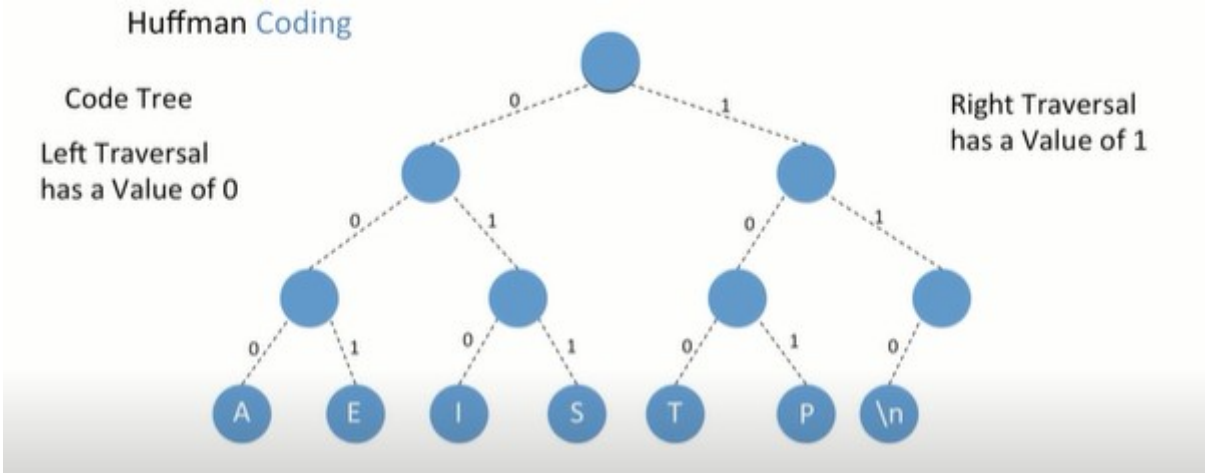
Huffman Coding

Character	Code	Frequency	Total Bits
A	000 Length = 3	10	30 Frequency x Bit Length
E	001	15	45
I	010	12	36
S	011	3	12
T	100	4	12
P	101	13	39
Newline	110	1	3

Total Bits Used: 174

Goal: Try to reduce the total number of bits used without losing any information.

Code Tree:



So Above, we want to find the char code for p, I traverse down the right branch adding 1 to the code, then I traverse down once from the left branch from that node, adding a 0, then I traverse down again from that node , adding a 1 to the code and getting to p.

P:101

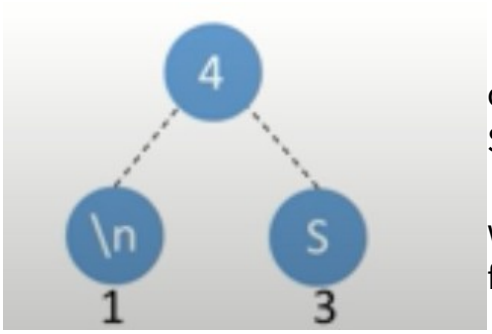
Goal: Reduce the code Tree.

So here we wanna reduce the traversal through the node tree, so that the nodes with highest frequency value.

Step1 :
Take the 2chars with the lowest frequency

Char	Freq
A	10
E	15
I	12
→ S	3
T	4
P	13
→ \n	1

Step 2: Make a two leaf node tree for them



The root node holds a value and the value is some of the frequency of the nodes beneath.
So 1+3=4;

we can treat this sub-tree as individual node with a frequency value of 4.

We are done with s and \n, so we scratch them off our list.

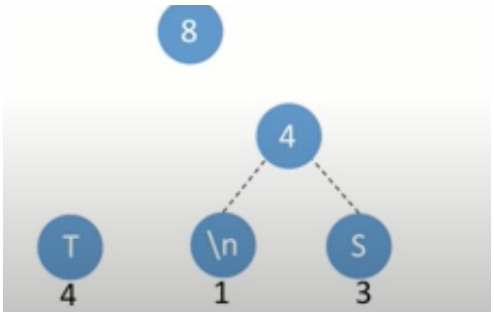
Step 3: Take the next lowest frequency char, and add it to the tree

Char	Freq
A	10
E	15
I	12
~~S~~	~~3~~
→ T	4
P	13
~~\n~~	~~1~~

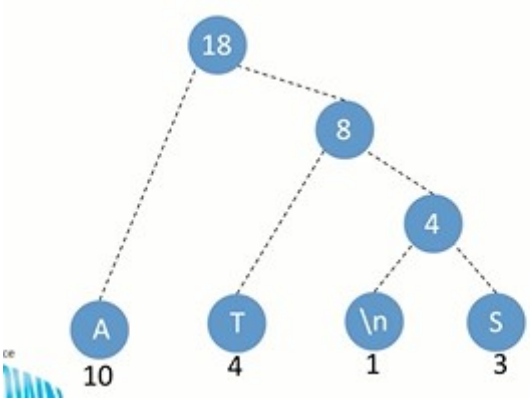
so, we create a node for T and put his frequency value beneath and create a parent node, which has a value of frequency of node T + value of frequency of the entire right sub-tree. So I.e 4+4=8;

we are done using T we can scratch them off our list

Char	Freq
A	10
E	15
I	12
~~S~~	~~3~~
~~T~~	~~4~~
P	13
~~\n~~	~~1~~



So, next char is A. so we create a node A and put his ……
So I.e 10+8=18

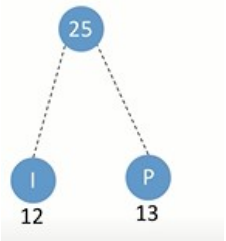


but hold it

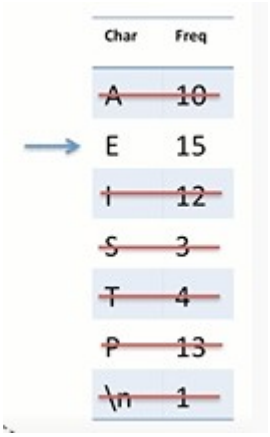
18 is really high, its higher than all the 3 remaining characters in the list.

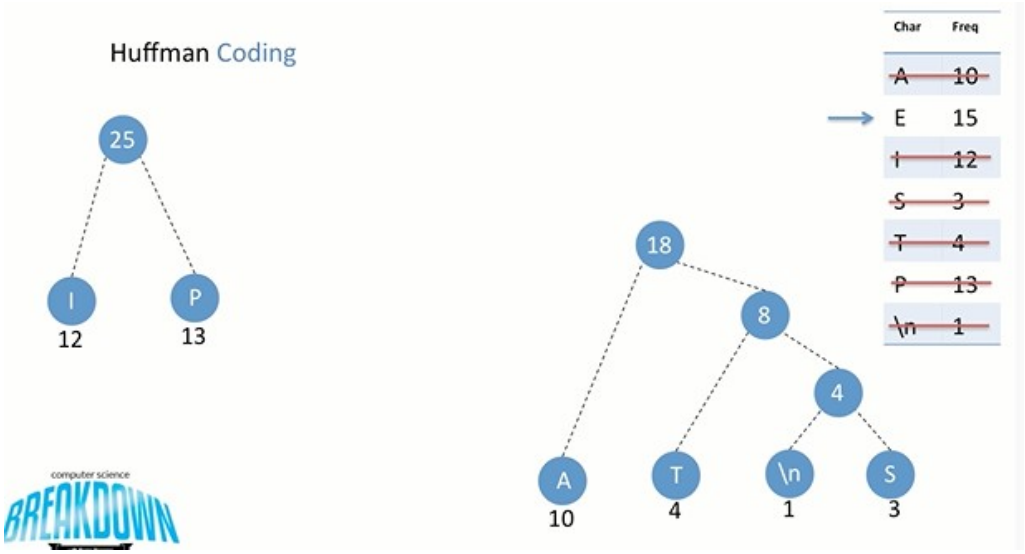
So we can't no-longer add to this sub-tree, so we have to create a new sub-tree using the next two lowest characters.

Char	Freq
~~A~~	~~10~~
E	15
→ I	12
~~S~~	~~3~~
~~T~~	~~4~~
→ P	13
~~\n~~	~~1~~

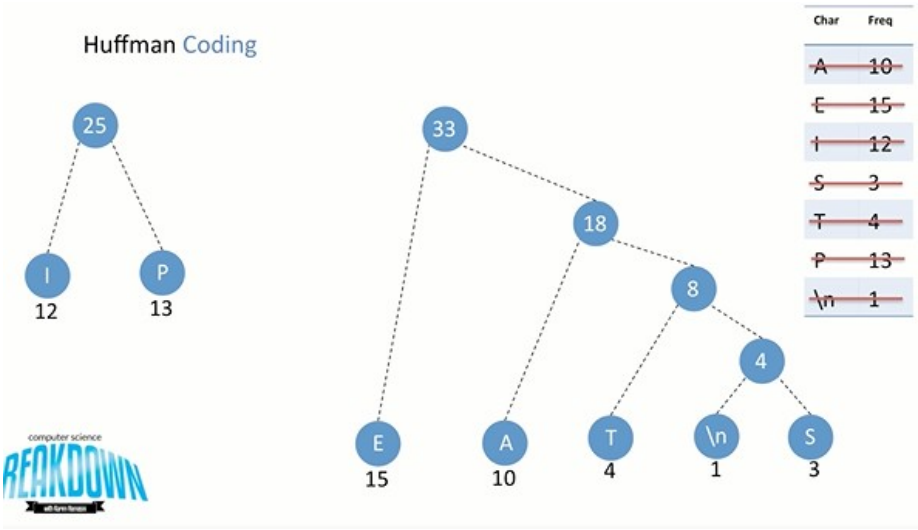


Now, we have 1 node is left



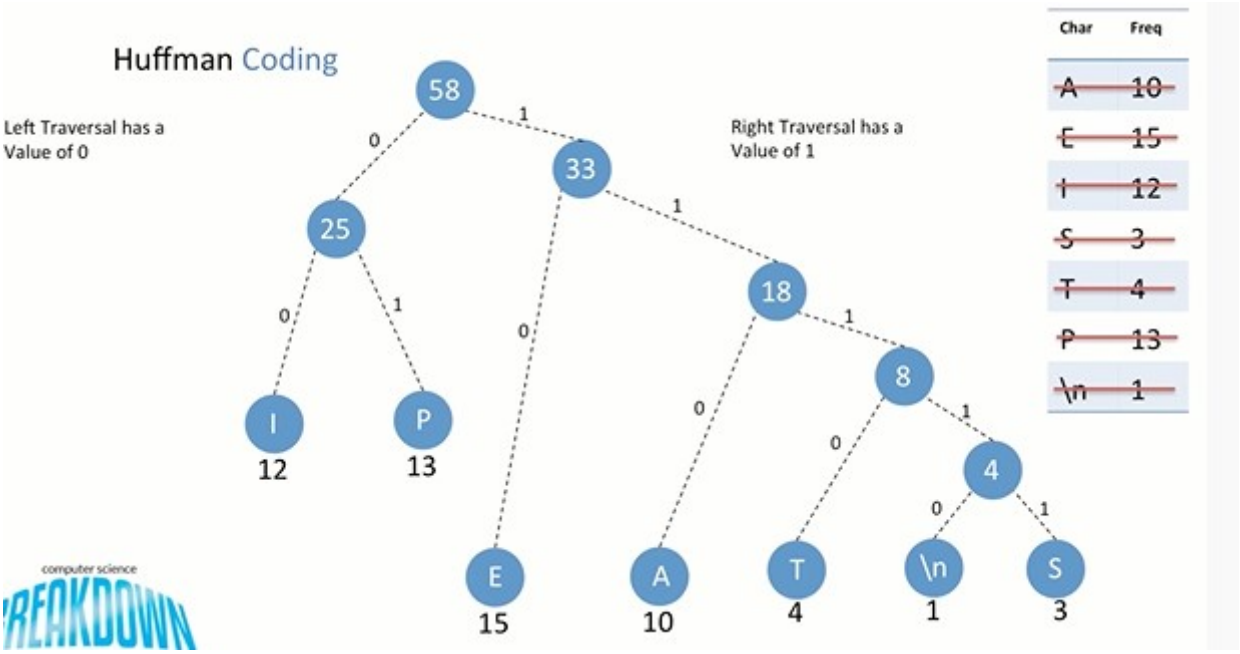


now I create a the node E, with its frequency value beneath, now I create the parent node with frequency value of 33.



so now we are done with E, we are scratching it off our list. So we are no-longer adding any leaf nodes characters in our tree.

But we are still left with two subtree, that we need to connect together to form one large tree.
So we connect the two sub-tree, now having a parent node with a frequency I.e 25+33=58;



Char	Code	Freq	Total Bits
A		10	
E		15	
I		12	
S		3	
T		4	
P		13	
\n		1	

--->

Char	Code	Freq	Total Bits
A	110	10	30
E	10	15	30
I	00	12	24
S	11111	3	15
T	1110	4	16
P	01	13	26
\n	11110	1	5

We have longer code for s and \n cuz they are seldom(rare,infrequent) used

Total Bits: 146

vs 174 without Huffman Coding