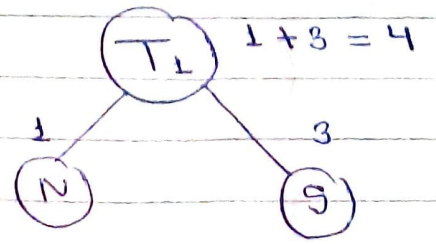
# Huffman Code

- Huffman code is a technique for compressing data, Huffman's greedy algorithm looks at the occurence of each character and it as a binary string in an optimal way.
- Initially we had the probability of occurence of characters. From the leaf level we start buildinging up the tree.
- Suppose we have two characters $T_1$ and $T_2$, after merging these characters we get $C_3$.
- The probability of character $C_3$ will be equal to the total sum of the probability of character $C_1$ and $C_2$.
- The whole process continues until we are left with single super character.

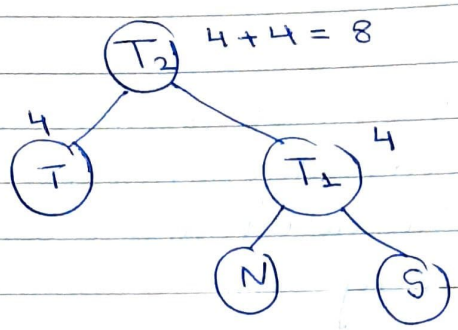
## Example

| Character | A | E | I | S | T | P | N |
|---|---|---|---|---|---|---|---|
| Frequency | 10 | 15 | 12 | 3 | 4 | 13 | 1 |

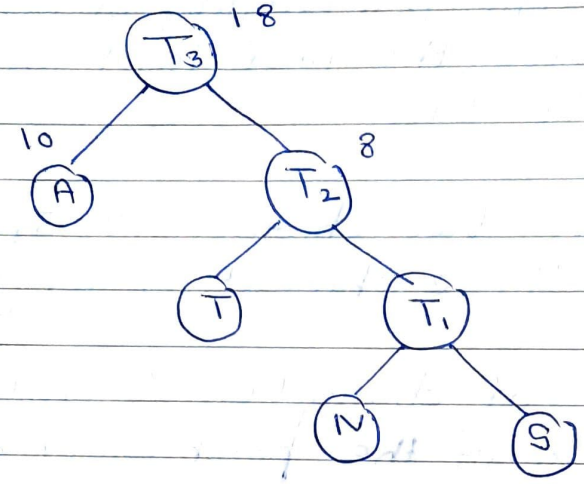
- Select two nodes having least frequency and construct a tree



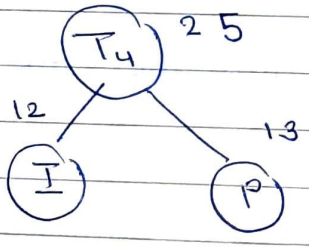
⇒


T2
4+4 = 8
4
T
T1
4
N
S

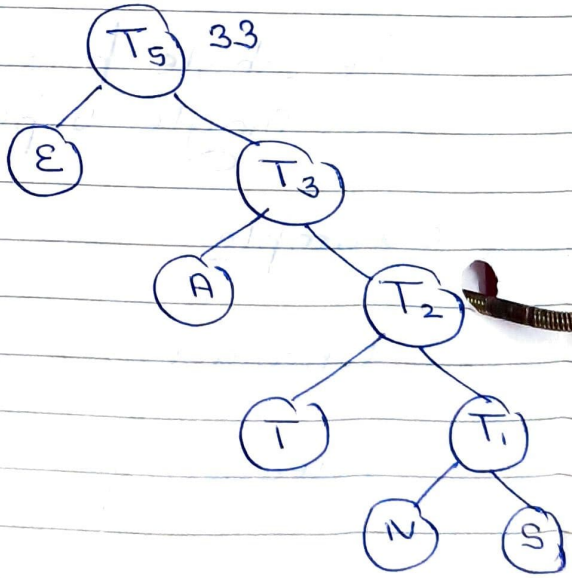
⇒


T3
18
10
A
T2
8
T
T1
N
S

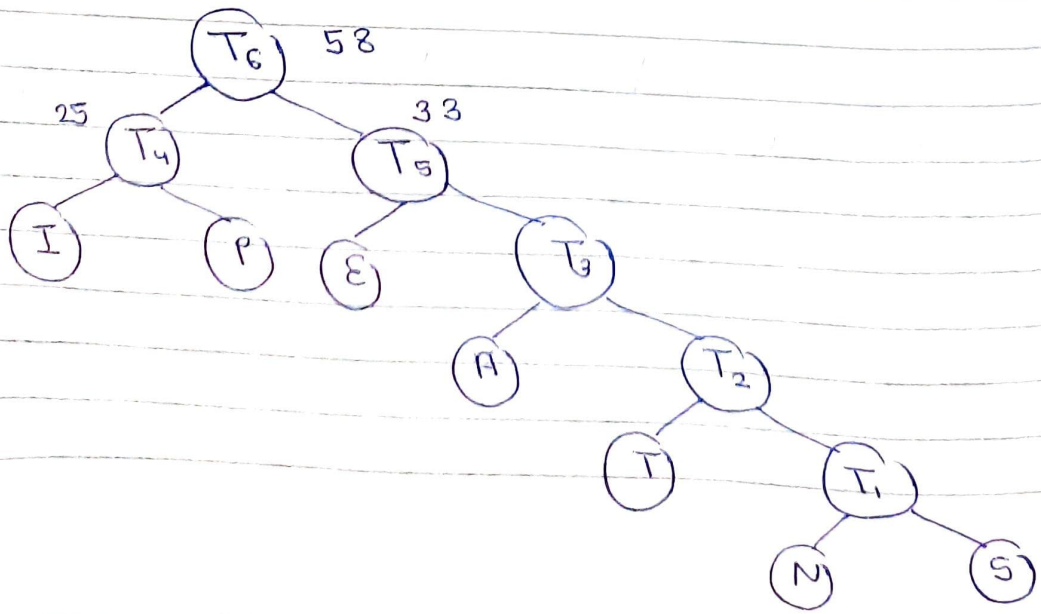
⇒


T4
25
12
I
13
P

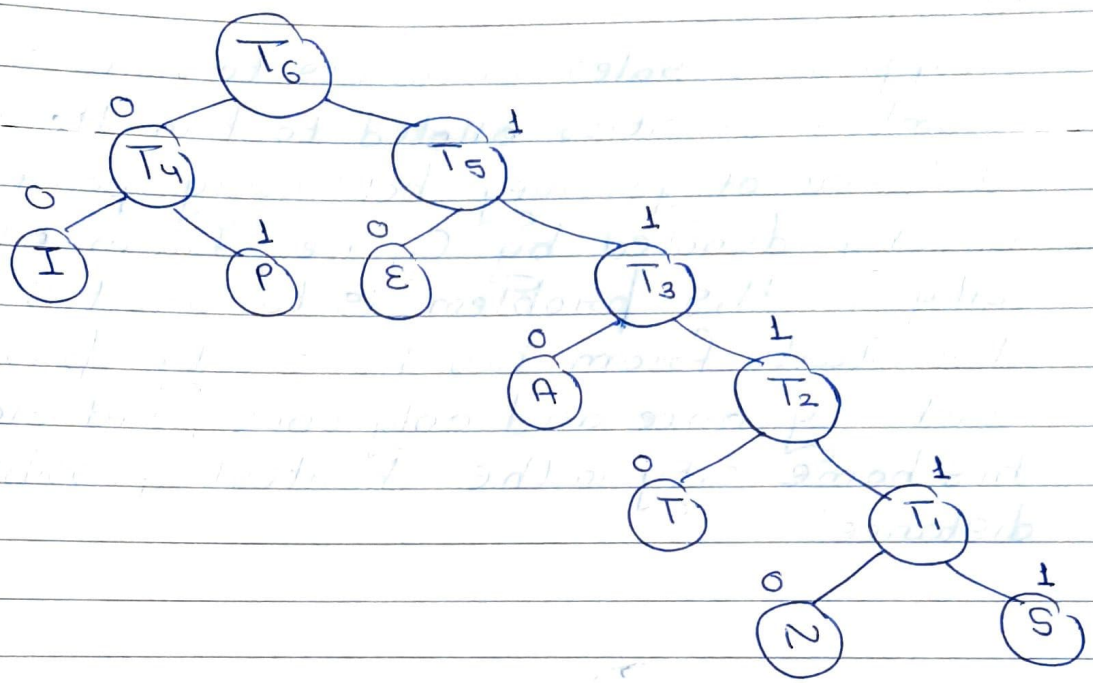
⇒


T5
33
E
T3
A
T2
T
T1
N
S

⇒


T6
58
25
T4
33
T5
I
P
E
T3
A
T2
T
T1
N
S

→ Assign bit 0 to all left sub tree and bit 1 to all right sub tree




- Generate bit patterns for all the alphabets

T → 1 1 1 0 (4)
I → 0 0 (12)
P → 0 1 (13)
E → 1 0 (15)
A → 1 1 0 (10)
N → 1 1 1 1 0 (1)
S → 1 1 1 1 1 (3)

Total Cost = 4×4 + 12×2 + 13×2 + 15×2 + 10×3 + 1×5 + 3×5

= 16 + 24 + 26 + 30 + 30 + 5 + 15

= 146