SNS COLLEGE OF ENGINEERING

Coimbatore-107

COURSE NAME: ANALYSIS OF ALGORITHM

II YEAR/ IV SEMESTER

UNIT – III

GREEDY TECHNIQUE

Topic

Greedy Technique: Huffman Coding /Trees

Greedy Technique : Huffman Coding Trees

Huffman Trees are constructed for encoding a given text of 'n' characters.

Codeword: Encoding a given text require each character to be associated with some bit sequence this is called code word.

Huffman Related to Greedy Technique:

*. At Each step, Huffman coding selects the two least frequent characters to combine [follows Greedy Choice].

*. It ensures that overall encoding will be as short as possible.

Types of Encoding:

(i) Fixed length Encoding:

It is an encoding technique in which each character is associated with a bit string of some fixed length.

(ii) Variable length Encoding:

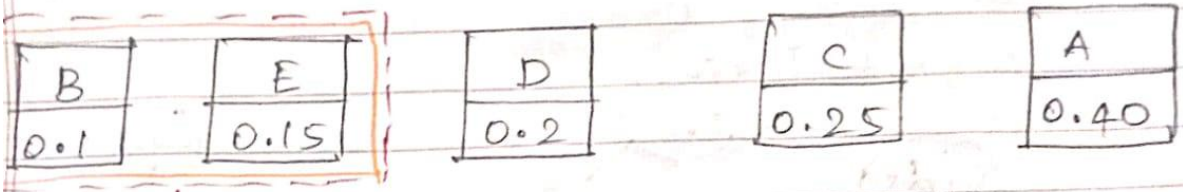
This technique includes each character associated with code word of different length.

Example:

Character	A	B	C	D	E
Probability	0.40	0.1	0.25	0.2	0.15

Step 1:

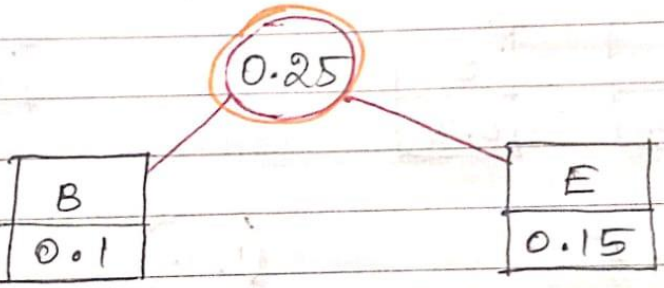
* First arrange characters in ascending order based on probabilities.



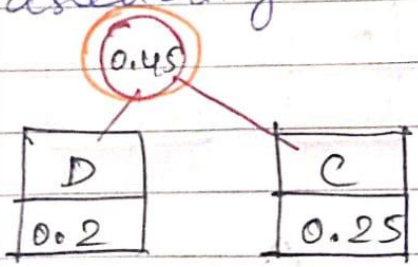
↓

mbine these node in to single node.

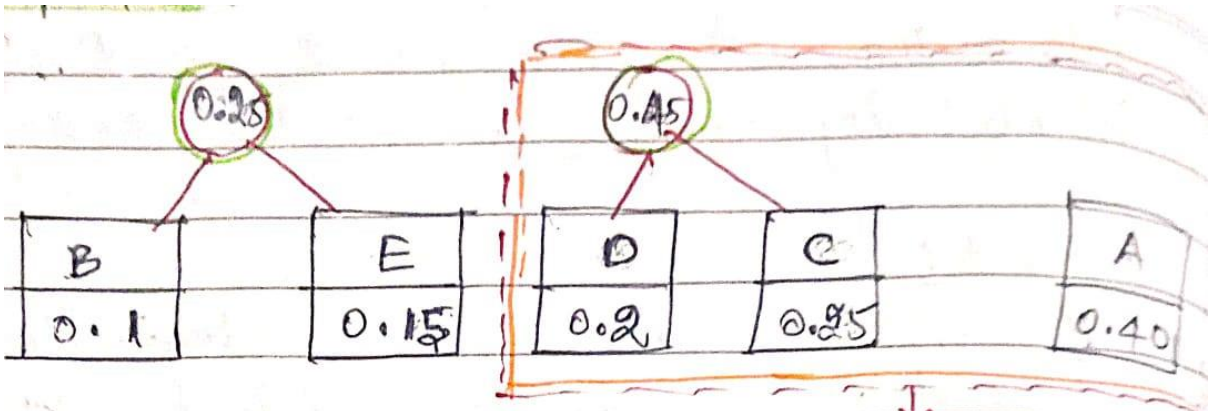
Step 2:



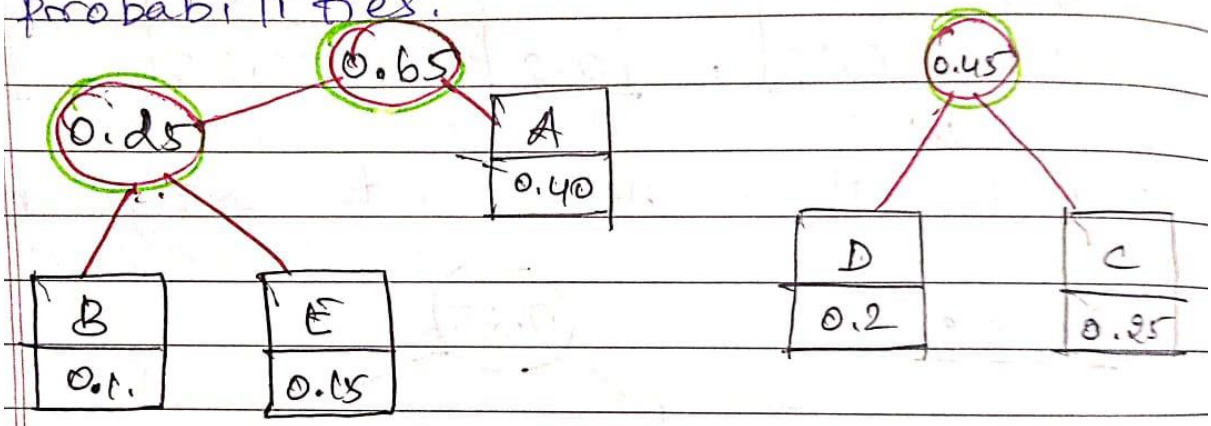
Again arrange the probabilities in ascending order, after combining C & D



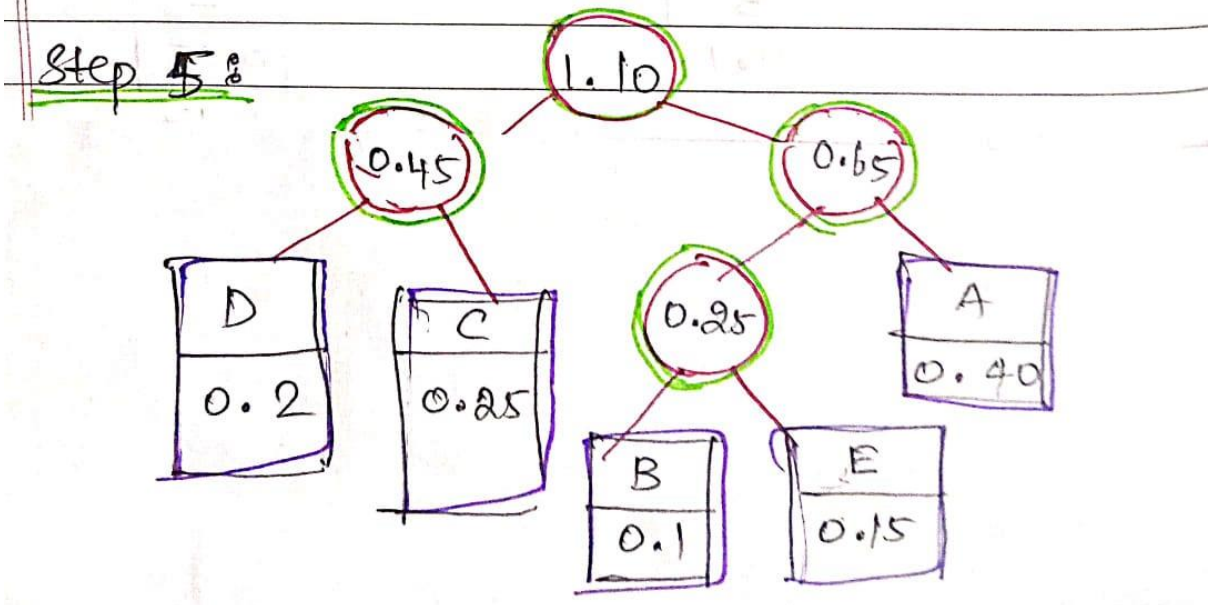
Arrange the nodes in ascending order based on probabili



Now combine (B, E) & A in Ascending order based on probabilities.



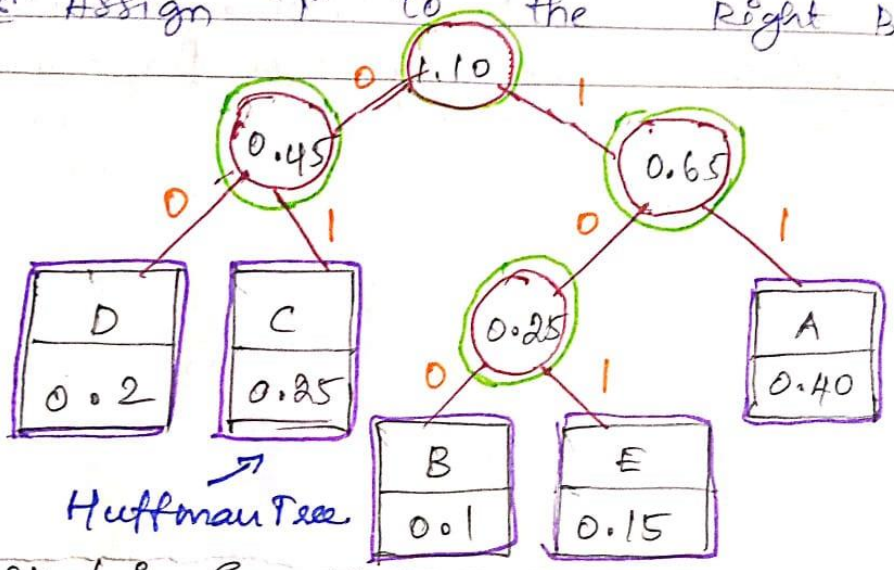
Step 5:



Step b : Encode the tree,

☆ Assign '0' to the Left Branch.

☆ Assign '1' to the Right Branch.



Huffman Tree

Step 6 : Encoding :

A	B	C	D	E
11	100	101	00	101

☆ Encode word DAD : D A D → 00 11 00

DAD can be encoded as 001100

Decode 11 01 100 → A C B : Decoded string ACB

Prefix code:

Code word associated with each character is called prefix code. Tree is called Huffman code.

Calculate Number of Bits Per character in given code:

Number of Bits per Character = $\sum_{\text{All Characters}}$ (Length of Code word × Probability of corresponding character)

= 2 × 0.4 + 3 × 0.1 + 2 × 0.25 + 2 × 0.2 + 3

= 2.45 ≈ 3 bits per character.

Algorithm:

```
Algorithm Huffman (data[], freq[], size)
{
while (size > 1) // find 2 min freq. node
min1 = 0, min2 = 1.
if (freq[min1] > freq[min2])
{ min1 = 1
  min2 = 0
}
for (i = 2, i < size; i++)
if (freq[min1] < freq[min1]
    min2 = min1
```

```
min1 = i
} else if (freq[i] < freq[min2])
  {
    min2 = i
  }
}
```

Complexity Analysis:

Time Complexity:

(i) Building Huffman Tree : $O(n^2)$
(Since we find the two smallest elements repeatedly)

(ii) Generating Huffman codes : $O(n)$

(iii) **Total complexity : $O(n^2)$** (simpler but slower than heap based $O(n \log n)$)

Space Complexity: → for Heap

(i) Huffman Tree Nodes : $O(n)$

(ii) Code Table : Huffman code Map : $O(n)$

(iii) Bit storage for Encoded Data : $O(m)$

m ⇒ Number of bits in encoded string.

Total space complexity : $O(n+m)$.