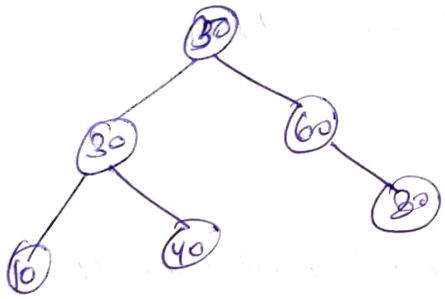
UNIT-3

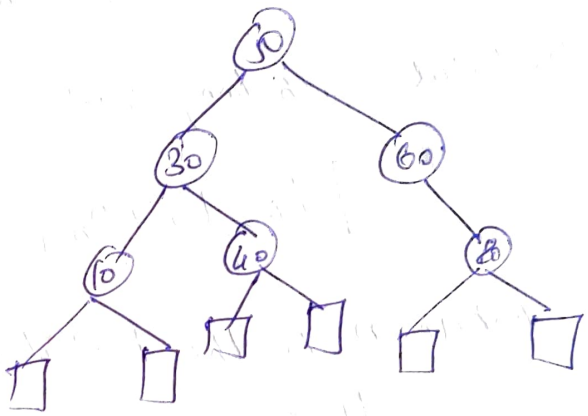
Extended Binary Tree:

A binary tree is called extended binary tree if every node of tree has zero or two children. The extended tree is also known as a 2-tree.

A binary tree can be converted to an extended binary tree by adding new nodes to its leaf nodes and to the nodes have only one child. The nodes of orginal tree are called internal nodes and new nodes that are added to binary tree to make it an extended binary tree are called external nodes



orginal Binary tree



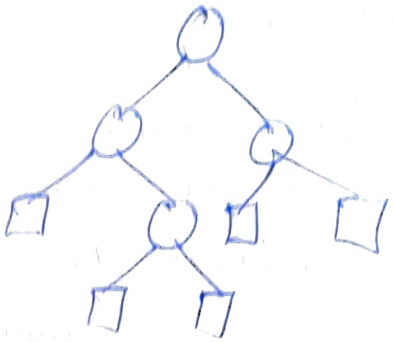
Extended binary tree.

Huffman Tree

In an Extended Binary tree is a binary tree in which every node has zero or two children. The nodes which have two children are called internal nodes and which are no children are called external node.

In any 2-tree the no. of External nodes is 1 more than no. of internal nodes.

ie $E = I + 1$



Internal node = 4
External node = 5

We know the path length of any node is the no. of minimum nodes traversed from root to that node.

$L_E = 2 + 3 + 3 + 2 + 2 = 12$

Total path length for internal nodes are

$L_I = 0 + 1 + 2 + 1 = 4$

The total path length of external node through

$$\boxed{L_E = L_I + 2n}$$

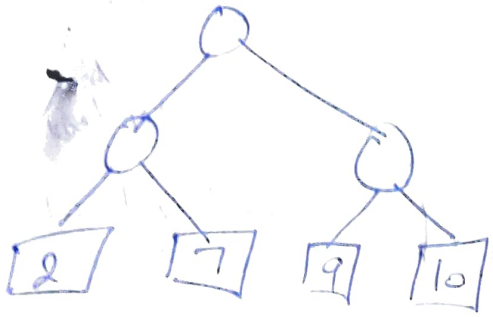
where n is the total number of nodes in the tree.

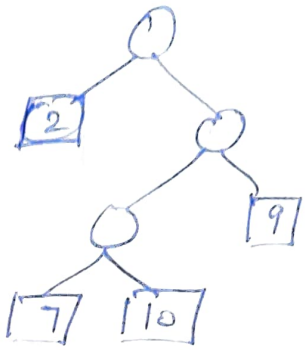
$L_E = 4 + 2 * 4$

$= 4 + 8 = 12$

The weighted path length for the external node will be

$P = w_1 L_1 + w_2 L_2 + w_3 L_3 + \ldots + w_n L_n$





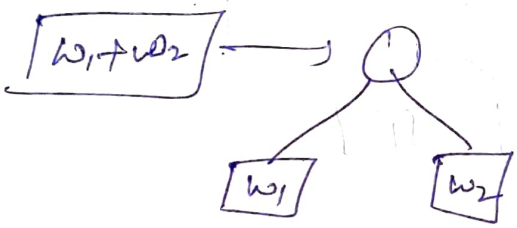
$P_1 = 2*2 + 7*2 + 9*2 + 10*2 = 56$

$P_2 = 2*1 + 7*3 + 10*3 + 9*2 = 71$

Huffman Algorithm

(1) Suppose there are n weights $w_1, w_2 \ldots w_n$.

(2) Take two minimum weights among the n given weights. Suppose w_1 and w_2 are first two minimum weights then subtree will be



(3) Now the remaining weights will be $w_1+w_2, w_3, w_4 \ldots w_n$.

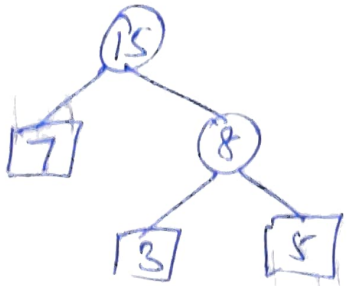
(4) Create all subtree at the last weight.

Q.

Items	A	B	C	D	E	F	G
Weights	15	10	5	3	7	12	25

minimum weights = 3, 5

Step 1

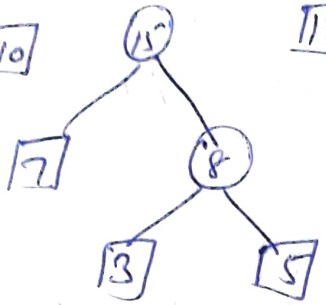


Step 2

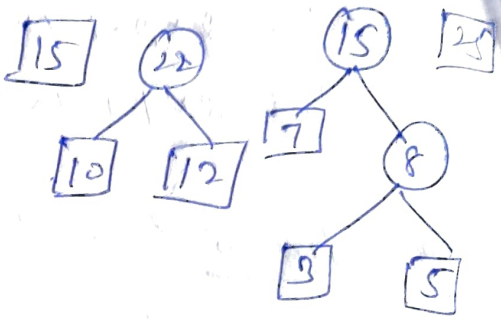
Now e... in the list ...

15, 10, 15, 12, ...

15 10



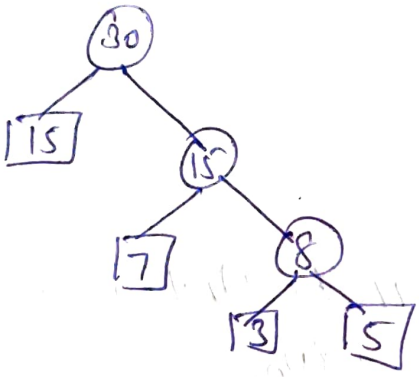
12 25



Step 3

Step 4.

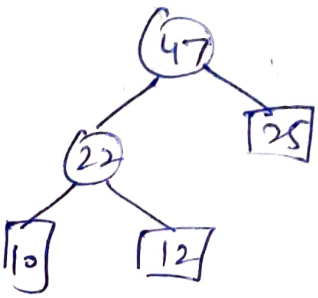
Taking two minimum weights nodes ie 15 and 15

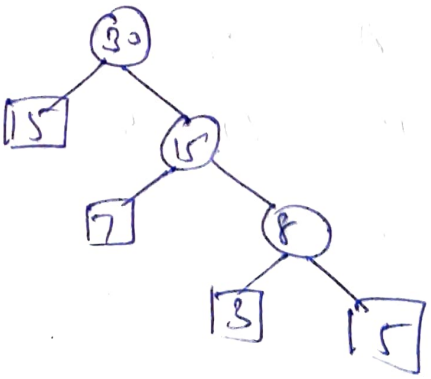


25

ie elements in the list are 30, 22, 25

Step 5: Taking two nodes with minimum weights which are 22 & 25.

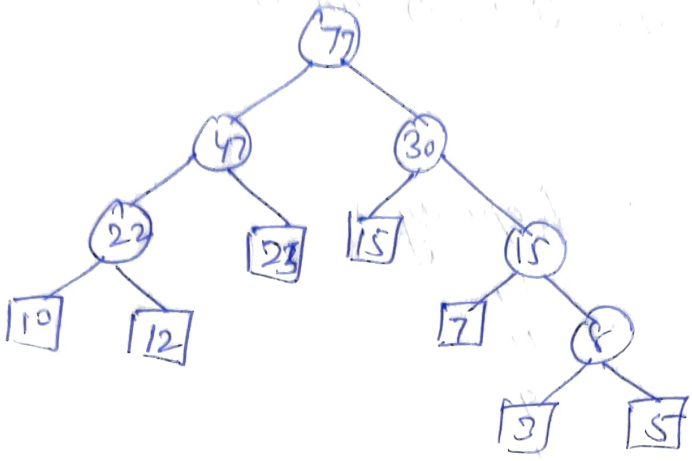




Now the elements in the list are 47 & 30.

4. Extended Binary Tree:

Taking two nodes with minimum weights which are 30 & 47.



Huffman Tree

Applications of Huffman Algorithm:-

The Huffman algorithm is used to perform the encoding & decoding of a long messages consisting of a set of symbols.

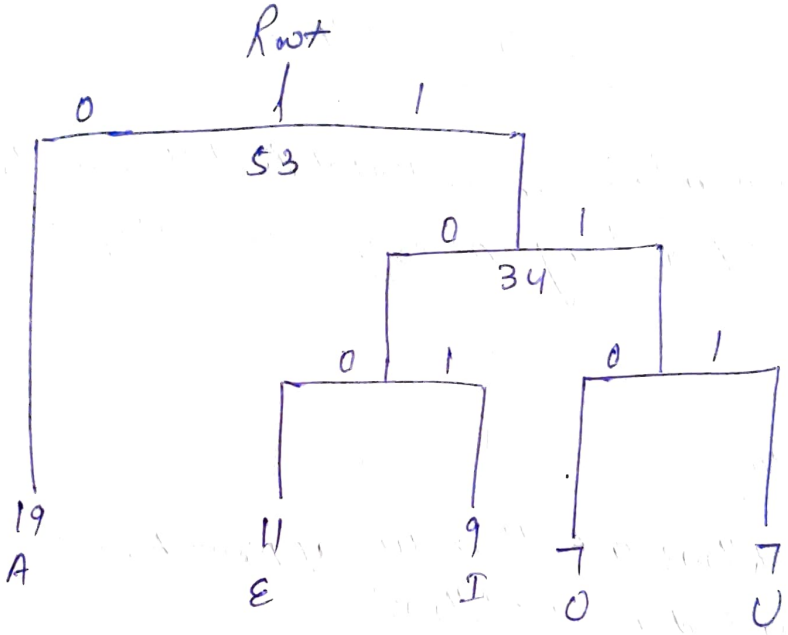
Huffman Coding:-

Huffman follows a bottom up approach. Procedure which we adopted is as follows:-

1. From the frequency list of all the symbols in the descending order.
2. Locate the two symbols in the list all the symbols in the descending order.
3. Create a parent node for these two nodes whose weight is equal to the sum of weights of two child nodes

4. Remove the two child from the list and
created parent node to the list alongwith the
5. Repeat through the step (2) until only one no
is left in the tree.

Symbol	Frequency
A	19
E	11
I	9
O	7
U	7



Huffman Decoding

Symbol	Codes
A	0
E	100
I	101
O	110
U	111

To perform decoding operation just traverse the nodes from the root node towards leaf nodes of the Huffman tree such that

given code bits are traversed.

Features of Huffman Tree

1) The Huffman tree is a binary tree.
2) In the Huffman tree, the most frequently occuring symbols will be the leaf nodes near the root node whereas the least frequently occuring symbols will be farther away from the root node.
3) The most frequently occuring symbols will have smaller code bits whereas the least frequently occuring symbols will have more code bits.
4) The root node of the Huffman tree is not assigned any code. The left child of the root node is assigned the code 0. The right child of the root node is assigned the code 1.
5) All the symbols of the given message will be leaf nodes of the Huffman tree.
6) The code for the child nodes of the same parent differs only in the last bit. The last bit is 0 for the left child node and is 1 for the right child node.

Applications

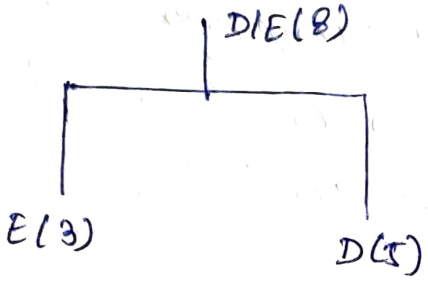
Wherever the messages have to be transmitted and received with less number of code bits.

Q

Symbol	Frequency
A	50
B	45
C	40
D	5
E	3

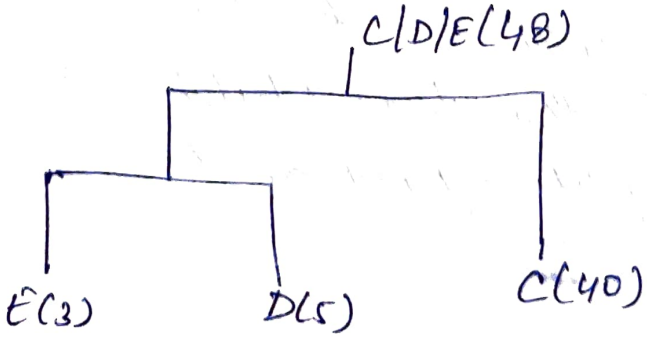
(d)

First,



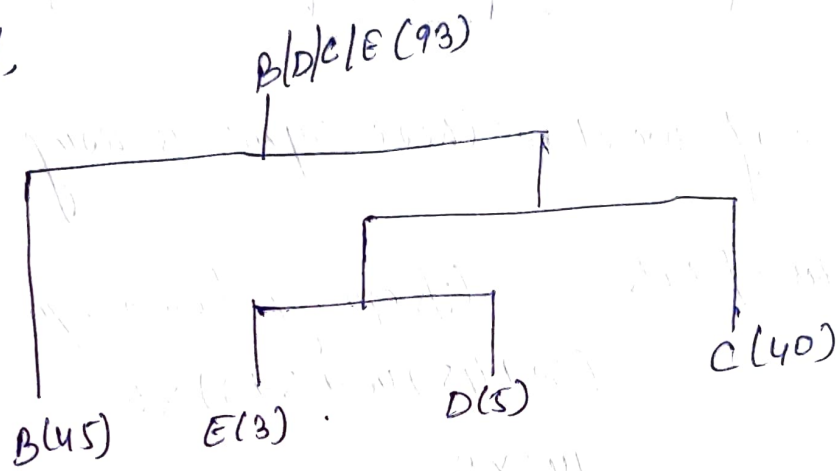
Symbol	Frequency
A	50
B	45
C	40
D/E	8

Second,

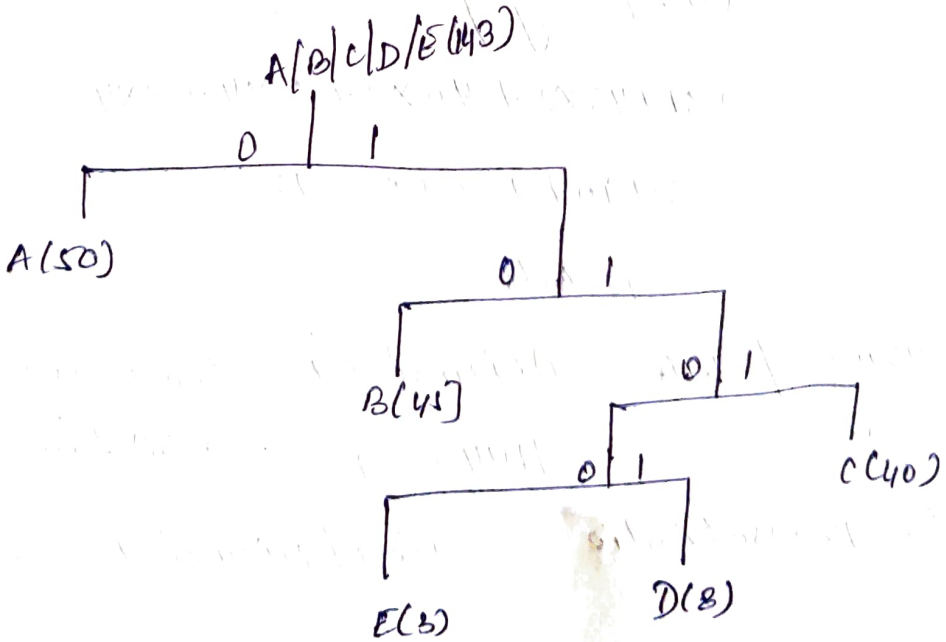


Symbol	Frequency
A	50
C/D/E	48
B	45

Third,



Symbol	Frequency
B/C/D/E	93
A	50



Symbol	Frequency	code	code length
A	50	0	1
B	45	10	2
C	40	111	3
D	5	1101	4
E	3	1100	4

Suppose if we store these symbols using ASCII then

Total number of bits = Σ(frequency of each symbol) $\times 8$

$= (50+45+40+5+3) \times 8$

$= 143 \times 8$

$= 1144$ bits

Bit requirement if we use Huffman Algorithm

$= \Sigma$(frequency of each symbol) $\times$ (code length)

$= 50 \times 1 + 45 \times 2 + 40 \times 3 + 5 \times 4 + 3 \times 4$

$= 50 + 90 + 120 + 20 + 12$

$= 292$ bits

Saving of bits = Original bit $-$ Compressed bit size.

$= 1144 - 292 = 852$ bits

Compressed ratio = $\frac{\text{Size of original data} - \text{size of compression data}}{\text{size of original data}}$