Trees

→ Definitions, properties and Examples.

→ Roted trees.
→ Preorder and Postorder Traversal.
→ Trees Sorting – Merge Sort.
→ Spanning trees.
→ weighted trees.

→ Prefix codes – optimal Prefix code.

TREES

Defn :- A graph G is said to be a Tree if it is connected and has no cycles [⇒ Simple graph ⇒ No loops & $\parallel^{el}$ edges]

Ex :-

* A pendant vertex of a tree is called a Leaf.

* Group of trees form a forest. ie each component of the disconnected graph is a tree. Such a graph is called a forest.

Theorem 1 :- A graph G is a tree iff there is one & only one path b/w every pair of vertices.

Proof :- Necessary:

Let T be a tree

⇒ T is a connected simple graph.

T is connected ⇒ there must be atleast one path b/w every pair of vertices, of T.

If suppose there are 2 paths b/w a pair of vertices of T, then the union of the paths will form a cycle & T cannot be a tree. Thus b/w every pair of vertices in a tree, there must be exactly one path.

Sufficiency:

Since there is a path b/w every pair of vertices in G, it is obvious that G is connected.

Since there is only one path b/w every pair of vertices, G cannot have a cycle; B'coz if there is a cycle, then ∃ 2 paths b/w 2 vertices on the cycle.

Thus G is a connected graph containing no cycles.

⇒ G is a tree.

Theorem 2 :- A tree with n vertices has $n-1$ edges.

[ie for a tree of n vertices & m edges, $m = n-1$.

(or) for a tree $T=(V,E)$, $|E| = |V|-1 \Rightarrow |V| = |E|+1$]

Proof :- proof is by induction on n.

The theorem holds good for $n=1$, $n=2$, $n=3$.

Assume that the theorem holds for all trees with n vertices where $n \leq k$.

Consider a tree T with $k+1$ vertices.

In T, let 'e' be an edge with end vertices u & v.

Since T is a tree, it has no cycles, and hence there is no other edge (or) path b/w u & v.

$\therefore$ Deletion of 'e' from T will disconnect the graph, and $T-e$ consists of exactly 2 components T_1 & T_2.

Since T does not contain any cycle, both T_1 & T_2 does not contain any cycles $\Rightarrow$ T_1 & T_2 are trees.

Both these trees have less than $k+1$ vertices each.

$\therefore$ By assumption made, theorem holds for T_1 & T_2 each.

ie each of T_1 & T_2 contains one less edge than the no of verticies in it.

Since the total no of vertices in T_1 & T_2 (taken together) is $k+1$, the total no of edges in T_1 & T_2 (taken together) is $(k+1) - 2 = k-1$.

But T_1 & T_2 taken together is $T-e$.

Thus $T-e$ has $k-1$ edges

$\Rightarrow$ T has exactly k edges.

Hence a tree T with $k+1$ verticies has k edges.

Hence the proof.

P.T.O.

Theorem 3 :- Any connected graph with n verticies & $n-1$ edges is a tree. [Ex cannot be given for this thm]

Proof :- Let G be a connected graph with n vertices & $n-1$ edges.

Assume that G is not a tree, then G contains a cycle say C. Let 'e' be an edge in C. Deleting the edge 'e' from G will not disconnect the graph. Thus $G-e$ is a connected graph. But $G-e$ has n verticies & $n-2$ edges, and hence it cannot be connected [by Thm, WKT a connected graph with n vertices has atleast $n-1$ edges]. This is a contradic-
-tion.

Hence G must not have a cycle.

$\Rightarrow$ G must be a tree.

Theorem 4 :- A connected graph G is a tree iff adding an edge b/w any 2 vertices in G creates exactly one cycle in G.

Proof :- Suppose G is a connected graph and is a tree. then G has no cycles & there is exactly one path b/w any 2 vertices u & v.

If we add an edge b/w u & v, then an additional path is created b/w u & v & the 2 paths constitute a cycle. Since G has no cycles earlier, this is the only cycle which G now possesses.

Conversely,

Suppose G is connected and adding an edge b/w any 2 vertices u & v in G creates exactly one cycle in G
$\Rightarrow$ before adding this edge, exactly one path was there b/w u & v $\Rightarrow$ G is a tree.

Hence the proof.

P.T.O.

Minimally connected graphs :-

A connected graph is said to be minimally connected if the removal of any one edge from it disconnects the graph.

Theorem 5 :- A connected graph is a tree if and only if it is minimally connected.

Proof :- Suppose G is a connected graph which is not a tree. then G contains a cycle C. The removal of any one edge 'e' from this cycle will not make the graph disconnected.

$\therefore$ G is not minimally connected.

This is equivalent to saying that if a connected graph is minimally connected, then it is a tree [contrapositive].

Conversely, suppose G is a connected graph which is not minimally connected. Then $\exists$ an edge 'e' in G such that $G-e$ is still connected.

$\therefore$ 'e' must be in some cycle in G. $\Rightarrow$ G is not a tree.

This is equivalent to saying that if a connected graph is a tree, then it is minimally connected (contrapositive).

Hence the proof:

Note :- Contrapositive:-

$$p \to q \quad \Rightarrow \quad \sim q \to \sim p.$$

$$\text{(or)} \quad q \to p \quad \Rightarrow \quad \sim p \to \sim q.$$

Problems :-

1) Show that (i) the complete graph K_n is not a tree when $n > 2$

(ii) the complete Bipartite graph $K_{r,s}$ is " ——— " $r \geq 2$

Soln :- (i) Let v_1, v_2, v_3 are any 3 vertices of K_n, where $n > 2$.

then the closed walk $v_1 v_2 v_3 v_1$ is a cycle in K_n.

Since K_n has a cycle, it cannot be a tree.

(ii) Let V_1 & V_2 be the bipartites.
Let v_1, v_2 be any 2 vertices in the first bipartite
& v_1', v_2' be " ______ " second " of $K_{r,s}$,
where $s \geq r > 1$.
Then the closed walk $v_1 v_1' v_2 v_2' v_1$ is a cycle in $K_{r,s}$.
Since $K_{r,s}$ has a cycle, it cannot be a tree.

2) Prove that a tree with 2 or more vertices contains atleast 2 leaves (pendant vertices).

Soln:- Consider a tree T with 'n' vertices, where $n \geq 2$, then it has $n-1$ edges.
∴ By HSP, sum of degrees of n vertices $= 2(n-1)$.
Thus if $d_1, d_2, \ldots, d_n$ are the degrees of vertices of T,
then $d_1 + d_2 + \ldots + d_n = 2n-2$.
If each of $d_1, d_2 \ldots d_n$ is ≥ 2, then $d_1 + d_2 + \ldots + d_n \geq 2n$.
Since this is not true, atleast one of the d's is less than 2.
ie there is a 'd' which is equal to 1 [Since T is connected, no 'd' can be zero]. Say d_1 is 1, then

$$d_2 + d_3 + \ldots + d_n = (2n-2) - 1 = 2n-3 \quad (*)$$

~~This is possible only if atleast one of $d_2, d_3 \ldots d_n$ is 1~~ which is = 1.
∴ there is atleast one more 'd' which is = 1. ie there are atleast 2 vertices with degree 1
Thus in T, ∃ atleast 2 pendant vertices (leaves).

(*) If each of $d_2, d_3 \ldots d_n \geq 2$, then $d_2 + d_3 + \ldots + d_n \geq 2(n-1)$ ie $2n-3 \geq 2n-2$ (not possible)

3) Prove that a graph with n vertices, $n-1$ edges & no cycles is connected.

Soln:- Let G be a graph with n vertices, $n-1$ edges & no cycles. Suppose G is not connected. Let the components of G be H_i, $i = 1, 2 \ldots k$. If H_i has n_i vertices, we have

$$n_1 + n_2 + \ldots + n_k = n.$$

Since G has no cycles, H_i's also do not have cycles, and they are all connected graphs.

$\Rightarrow$ they are trees.
(H_i's)

$\Rightarrow$ each H_i must have $n_i - 1$ edges.

$\therefore$ Total no of edges in these H_i's is

$(n_1-1)+(n_2-1)+\dots+(n_k-1) = (n_1+n_2+\dots+n_k) - \underbrace{(1+1+\dots+1)}_{k\ \text{times}}$

$= n-k.$

This must be equal to total no of edges in G ie $n-1$.

ie $n-k = n-1$.

$\Rightarrow k=1$. (k is the no of components)

This is not possible since $k>1$

$\therefore$ G must be connected.

4) Let F be a forest with k components (trees). If 'n' is the no of vertices & m is the no of edges in F,

P.T $n = m+k$.

Soln :- Let $H_1, H_2 \dots H_k$ be the components of F.

Since each of these is a tree, if n_i is the no of vertices in H_i & m_i is the no of edges in H_i, then

$m_i = n_i - 1$ for $i = 1, 2 \dots k$.

$\Rightarrow m_1+m_2+\dots+m_k = (n_1-1)+(n_2-1)+\dots+(n_k-1)$

$m = n-k.$

$\Rightarrow m = n-k$ (or) $n = m+k$.

5) Let $T_1 = (V_1, E_1)$ & $T_2 = (V_2, E_2)$ be 2 trees. If $|E_1| = 15$ & $|V_2| = 3|V_1|$, determine $|V_1|, |V_2|$ & $|E_2|$

Soln:- we have $|V_1| = |E_1| + 1 = 15+1 = 16$.

Since $|V_2| = 3|V_1| = 3\times16 = 48$

$\therefore |E_2| = |V_2| - 1 = 48-1 = 47$.

6) If a tree has 400 vertices, find the sum of the degree of the vertices.

— soln in page ④ —

8) In a tree, if the degree of every non-pendant vertex is 3, show that the no of vertices in the tree is an Even no.

Soln :- Let 'n' be the no of vertices in a tree T.

Let 'K' be the no of $\underbrace{\text{Pendant}}_{deg=1}$ vertices ⇒ there are $(n-k)$ non-pendant vertices.

If each non-pendant vertex is of degree 3, then the sum of deg of vertices $= (k\times1) + 3\times(n-k) = 2(n-1)$ (HSP)

$$\Rightarrow \quad k+3n-3k = 2n-2.$$

$$\Rightarrow \quad n = 2k-2$$

(or) $n = 2(K-1)$, which is an Even no.

9) Suppose that a tree T has N_1 vertices of deg 1, N_2 vertices of deg 2, N_3 vertices of deg 3, ... N_K vertices of deg K.

Prove that $N_1 = 2+N_3+2N_4+ ... + (K-2)N_K$.

Soln :- Total no of vertices $= N_1+N_2+ ... +N_K$.

& sum of deg of vertices $= (N_1\times1)+(N_2\times2)+(N_3\times3)+ ... (N_K\times K)$

$= N_1+2N_2+3N_3+ ... +KN_K$.

& Total no of edges $= N_1+N_2+ ... +N_K-1$.

By HSP, $N_1+2N_2+ ... +KN_K = 2(N_1+N_2+ ... +N_K-1)$

$(3N_3-2N_3)+(4N_4-2N_4)+ ... +(KN_K-2N_K) = 2N_1-N_1-2$

$$\Rightarrow \quad N_3+2N_4+ ... +(K-2)N_K = N_1-2.$$

(or) $N_1 = 2+N_3+2N_4+ ... +(K-2)N_K$.

10) If a tree has 4 vertices of deg 2, one vertex of deg 3, 2 vertices of deg 4 & one vertex of deg 5. Find the no of leaves in T.

Ans $N=10$

11) Suppose that a tree T has 2 vertices of deg 2, 4 vertices of deg 3 and 3 vertices of deg 4. Find the no of Pendant vertices in T.

Ans $N=12$

Given $|V| = 400$

$\therefore |E| = |V| - 1 = 399$.

$\Rightarrow$ By HSP, Sum of the degrees of the vertices $= 2|E|$
$= 2 \times 399$
$= 798$

DO (8) & (9) Here.

7) If a tree has 4 vertices of deg 3, 2 vertices of deg 4 & one vertex of deg 5. Find the no of pendant vertices in T.

Soln :- Let N be the no of pendant vertices in T. (degree 1)

Total no of vertices $= N + 4 + 2 + 1 = N + 7$

Sum of the degrees of the vertices $= (N \times 1) + (4 \times 3) + (2 \times 4) + (1 \times 5)$
$= N + 12 + 8 + 5$
$= N + 25$

Since T has N+7 vertices, it has $(N+7) - 1 = N+6$ edges.

$\therefore$ By HSP, $N + 25 = 2(N+6)$
$N + 25 = 2N + 12$
$N = 13$

$\Rightarrow$ given tree has 13 pendant vertices.

8) Prove that every tree is a planar graph.

Soln :- Since a tree has no cycle in it, no subgraph of a tree has any cycles.

Hence no subgraph of a tree can be isomorphic to K_5 (OR) $K_{3,3}$, which are non-planar.

$\therefore$ a tree cannot be non-planar

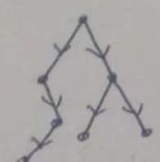
i.e Every tree is a planar graph.

Rooted Trees

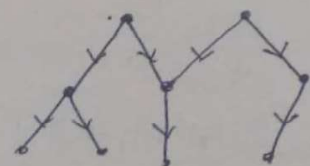
* **Directed Tree** :- A directed graph which is a tree is called a directed tree.

* **Rooted Tree** :- A directed Tree T is called a rooted tree if
  (i) T contains a unique vertex called the root whose in-degree is equal to zero.
  (ii) in-degrees of all other vertices of T are equal to one.

Ex :-



rooted tree



not a rooted tree.

* **Level number** :- A vertex 'v' of a rooted tree is said to be at the k^{th}-level (or) has level number 'k' if the path from r to v is of length k.

Note :-

1) If v_1 & v_2 are 2 vertices such that v_1 has a lower level no than v_2 & there is a path from v_1 to v_2, then v_1 is an ancestor of v_2 (or) v_2 is a descendant of v_1.

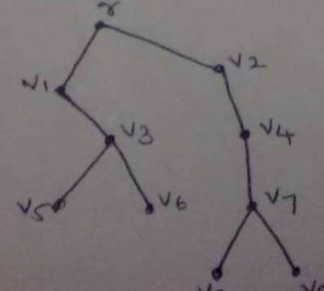
2) If v_1 & v_2 are such that v_1 has a lower level no than v_2 and there is an edge from v_1 to v_2, then v_1 is called the parent of v_2 (or) v_2 is called the child of v_1.

3) Two vertices with a common parent are called Siblings.

4) A vertex whose out-degree is zero is called a Leaf.

5) A vertex which is not a leaf is called Internal vertex.

Ex :-



v_1 is the ancestor of v_3, v_5, v_6.
(or) v_3, v_5, v_6 are the descendant of v_1.
v_1 is the parent of v_3
(or) v_3 is the child of v_1
v_5 & v_6 are siblings.

v_5, v_6, v_8, v_9 are leaves ; all other vertices are internal vertices.

Scanned by CamScanner

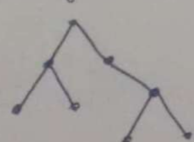
* **m-ary Tree** :- A rooted tree T is called an m-ary tree if every internal vertex of T is of out-degree $\leq m$; ie if " ______ " ______ " has atmost m children.

* **Complete m-ary Tree** :- A rooted tree T is called a complete m-ary tree if every internal vertex of T is of out-degree m ; ie if every internal vertex of T has exactly m children.

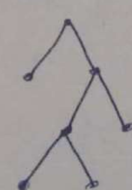
* **Binary Tree** :- An m-ary tree for which $m=2$ is called a binary Tree ie A rooted tree T is called a Binary tree if every vertex of T is of out-degree $\leq 2 \Rightarrow$ if every vertex has atmost two children.

* **Complete Binary Tree** :- A rooted tree T is called a complete Binary tree if every internal vertex of T is of outdegree 2 ; ie if every internal vertex has exactly 2 children.

Ex :-



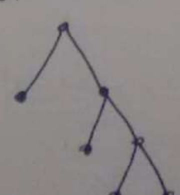
Binary tree



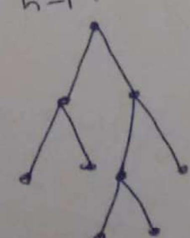
complete Binary tree.

* **Balanced Tree** :- If T is a rooted tree & h is the largest level no achieved by a leaf of T, ~~then~~ T is said to have height 'h'. A rooted tree of height 'h' is said to be balanced if the level no of every leaf is h (or) h-1.

Ex :-



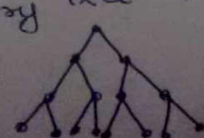
Not balanced



balanced.

* **full Binary Tree** :- Let T be a complete Binary tree of height h. Then T is called a full Binary tree if all the leaves in T are at level 'h'.

Ex :-



Problems :-

1) Let T be a complete m-ary tree of order n with p leaves & q internal vertices. Prove that

(i) $n = mq+1 = \frac{mp-1}{m-1}$.

(ii) $p = (m-1)q+1 = \frac{(m-1)n+1}{m}$

(iii) $q = \frac{n-1}{m} = \frac{p-1}{m-1}$.

Soln:- Given that T is a complete m-ary tree.

$\Rightarrow$ every internal vertex is of out-degree m.

Since every vertex in a rooted tree is either a leaf or an internal vertex, the total no of vertices in T is the sum of no of leaves in T & no of internal vertices in T.

ie $n = p+q \longrightarrow (1)$.

WKT out-degree of every leaf is zero and out-degree of every internal vertex is m

$\Rightarrow$ $\left.\begin{array}{l}\text{Sum of out-degrees of all vertices}\\ \text{in T}\end{array}\right\} = (p\times 0)+(q\times m) = qm \longrightarrow (2)$

WKT in a rooted tree of order n, the in-degree of the root is zero & the indegree of the remaining $(n-1)$ vertices are 1 each.

$\therefore$ $\left.\begin{array}{l}\text{Sum of indegrees of all vertices}\\ \text{of T}\end{array}\right\} = (1\times 0)+(n-1)1 = n-1 \longrightarrow (3)$

By 1st thm of digraph theory,

Sum of indegrees = Sum of out-degrees.

ie $n-1 = qm$

$\Rightarrow \boxed{n = qm+1} \rightarrow (4).$ $\Rightarrow \boxed{q = \frac{n-1}{m}}$

sub (4) in (1), $qm+1 = p+q$.

$\boxed{p = (m-1)q+1}$

$\Rightarrow \boxed{q = \frac{p-1}{m-1}}$

(4) $\Rightarrow$ $n = m\dfrac{(p-1)}{(m-1)} + 1 = \dfrac{m(p-1)+(m-1)}{m-1}$

$$\boxed{n = \frac{mp-1}{m-1}}$$

we have $q = \dfrac{p-1}{m-1}$

$\therefore$ (1) $\Rightarrow$ $n = p + \dfrac{p-1}{m-1} = \dfrac{pm - p + p - 1}{m-1}$

$$n = \frac{pm-1}{m-1}$$

$$\therefore \quad \boxed{p = \frac{(m-1)n+1}{m}}$$

Note :- for a complete binary tree, the results are :-

ie $m = 2$

(i) $n = 2q + 1 = 2p - 1$

(ii) $p = q + 1 = \frac{1}{2}(n+1)$

(iii) $q = \frac{1}{2}(n-1) = p - 1$.

2) Find the no of leaves in a complete Binary tree if it has 29 vertices.

Soln :- Complete Binary tree $\Rightarrow m = 2$.

Given $n = 29$

$p = ?$

we have $p = \frac{1}{2}(n+1) = \frac{1}{2}(29+1) = \frac{30}{2} = 15$

$\therefore$ No of leaves $= p = 15$.

3) Find the no of internal vertices in a complete 5-ary tree with 817 leaves. Hence find its order.

Soln :- Given :- $m = 5$; $p = 817$

$q = ?$

we have $q = \dfrac{p-1}{m-1} = \dfrac{816}{4} \Rightarrow \boxed{q = 204}$

ie given tree has 204 internal vertices.

order, $n = mq + 1$

$n = (5)(204) + 1$

$n = 1020 + 1$

$\boxed{n = 1021}$

4) A classroom contains 25 microcomputers that must be Connected to a wall socket that has 4 outlets. Connections are made by using extension cords that have 4 outlets each. Find the least no of cords needed to get this Computer set up for the class.

Soln:- Let us treat the wall Socket as the root of a complete 4-ary tree with the computers as its leaves & the internal vertices other than the root as extension cords.

$\therefore$ $m = 4$, $p = 25$

$\Rightarrow$ No of internal vertices is $q = \frac{p-1}{m-1} = \frac{24}{3} = 8$.

$\therefore$ no of extension cords needed [ie the no of internal vertices minus the root] is $q-1 = 8-1 = 7$.

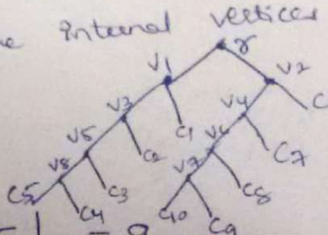
5) A complete Binary tree has 20 leaves. Find the no of vertices.

Soln:- $m = 2$; $p = 20$.

$$n = \frac{mp-1}{m-1} = \frac{2.20-1}{2-1} = \underline{\underline{39}}.$$

6) The computer laboratory of a school has 10 computers that are to be connected to a wall socket that has 2 outlets. connections are made by using extension chords that have 2 outlets each. Find the least no of cords needed to get these computers set up for use.

Soln:- Consider the complete Binary tree ($m=2$) having the wall socket as the root, the computers as the leaves & the internal vertices other than the root as extension cords.



$\therefore$ $m=2$, $p = 10$

$\Rightarrow$ No of internal vertices $= q = \frac{p-1}{m-1} = \frac{10-1}{2-1} = 9$.

$\therefore$ No of extension cords needed = no of internal vertices − root
$= 9-1 = 8$

Scanned by CamScanner

Pre-order and Post-order Traversals :-

Consider a rooted tree T. Let r be its root & let $T_1, T_2 \ldots T_K$ be subtrees of T as we move from left to right.

Suppose we list the vertices of T such that r is the first vertex & the subsequent vertices are selected in the foll order : vertices of T_1 read from left to right, vertices of T_2 read from left to right & so on. The list of vertices so prepared is called the pre-order Traversal of T.

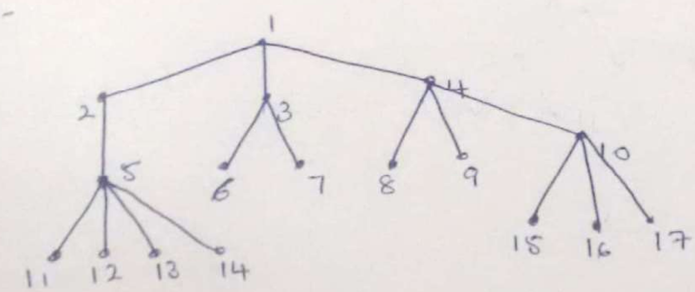
Suppose we list the vertices of T selected in the foll order : vertices of T_1 read from leftmost descendant, vertices of T_2 read from leftmost descendant & so on & finally the root r. The list of vertices so prepared is called the post-order Traversal of T.

Pre order :- node left right
NLR

Post order :- Left right node
LRN

EX :-

1)



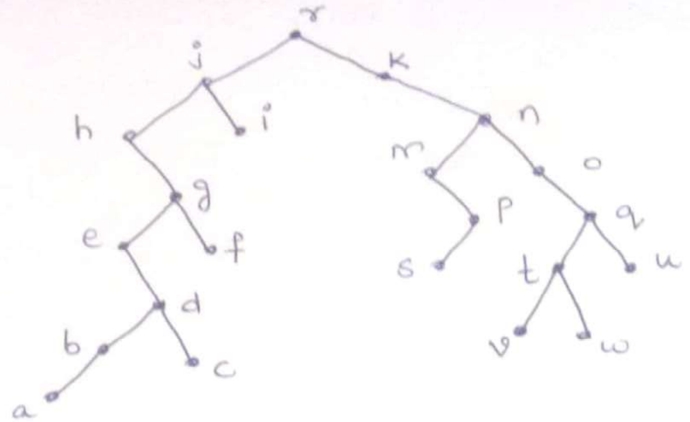
Preorder traversal :-

1 2 5 11 12 13 14 3 6 7 4 8 9 10 15 16 17

Postorder traversal :-

11 12 13 14 5 2 6 7 3 8 9 15 16 17 10 4 1

2)



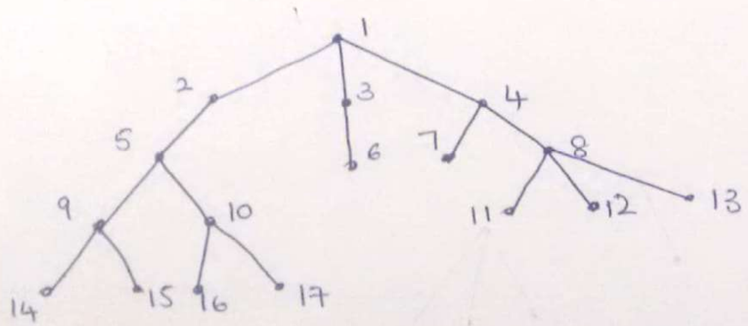
Pre order traversal :-

r j h g e d b a c f i k n m p s o q t v w u.

Postorder traversal :-

a b c d e f g h i j s p m v w t u q o n k r

3)



Pre order traversal :-

1 2 5 9 14 15 10 16 17 3 6 4 7 8 11 12 13.

Post order traversal :-

14 15 9 16 17 10 5 2 6 3 7 11 12 13 8 4 1

Sorting

The method of splitting and merging, done by the use of balanced complete trees is known as merge sort.

The process of sorting consists of two parts:

In the first part, we recursively split the given list & all subsequent list in half until each sublist contains a single element.

In the second part, we merge the sublists in non-decreasing order until the original 'n' integers have been sorted.

Ex 1 :- Apply Merge Sort to the lists -1, 7, 4, 11, 5, -8, 15, -3, -2, 6, 10, 3.

Q1

Soln :- {-1, 7, 4, 11, 5, -8, 15, -3, -2, 6, 10, 3}

{-1, 7, 4, 11, 5, -8} {15, -3, -2, 6, 10, 3}

{-1, 7, 4} {11, 5, -8} {15, -3, -2} {6, 10, 3}

{-1, 7} {4} {11, 5} {-8} {15, -3} {-2} {6, 10} {3}

{-1} {7} {4} {11} {5} {-8} {15} {-3} -2 {6} {10} 3

To merge,

-1 7 → {-1, 7} 4 → {-1, 4, 7}

11 5 → {5, 11} -8 → {-8, 5, 11}

{-1, 4, 7} {-8, 5, 11} → {-8, -1, 4, 5, 7, 11}

15 -3 → {-3, 15} -2 → {-3, -2, 15}

6 10 → {6, 10} 3 → {3, 6, 10}

{-3, -2, 15} {3, 6, 10} → {-3, -2, 3, 6, 10, 15}

{-8, -1, 4, 5, 7, 11} {-3, -2, 3, 6, 10, 15} → {-8, -3, -2, -1, 3, 4, 5, 6, 7, 10, 11, 15}.

∴ sorted out version of the given list is -8, -3, -2, -1, 3, 4, 5, 6, 7, 10, 11, 15.

Ex 2 :- -9, 6, 5, -3, 4, 2, -7, 6, -5, 10, -11, 0, 1.

Soln :- {-9, 6, 5, -3, 4, 2, -7, 6, -5, 10, -11, 0, 1}

{-9, 6, 5, -3, 4, 2, -7}   {6, -5, 10, -11, 0, 1}

{-9, 6, 5, -3}   {4, 2, -7}   {6, -5, 10}   {-11, 0, 1}

{-9, 6}   {5, -3}   {4, 2}   {-7}   {6, -5}   {10}   {-11, 0}   {1}

{-9} {6} {5} {-3} {4} {2} -7 {6} {-5} 10 {-11} {0} 1

To merge.

-9 6 5 -3 4 2 ~~-7~~ 6 -5 ~~10~~ -11 0 ~~1~~

{-9, 6}   {-3, 5}   {2, 4}   -7   {-5, 6}   10   {-11, 0}   1

{-9, -3, 5, 6}   {-7, 2, 4}   {-5, 6, 10}   {-11, 0, 1}

{-9, -7, -3, 2, 4, 5, 6}   {-11, -5, 0, 1, 6, 10}

{-11, -9, -7, -5, -3, 0, 1, 2, 4, 5, 6, 10}.

Ex 3 :- {7, 3, 8, 4, 5, 10, 6, 2, 9}.

Ex 4 :- -1, 0, 2, -2, 3, 6, -3, 5, 1, 4.

Spanning Trees:-

Defn:- Let G be a connected graph. A subgraph T of G is called a spanning tree of G if (i) T is a tree and (ii) T contains all vertices of G.

1) Spanning tree is also called a maximal tree since a spanning tree T of a graph G is a subgraph of G that contains all vertices of G.

2) The edges of a spanning tree are called its branches.

3) If G has n-vertices, a spanning tree of G must have n-vertices and hence $n-1$ edges.

4) If T is a spanning tree of a graph G, then the edges of G which are not in T are called the chords of G wrt T. The set of all chords of G is the complement of T in G. This set is called the chord set or cotree of T in G and is denoted by $\bar{T}$.

clearly $G = T \cup \bar{T}$. Ex:- * (P.T.O)

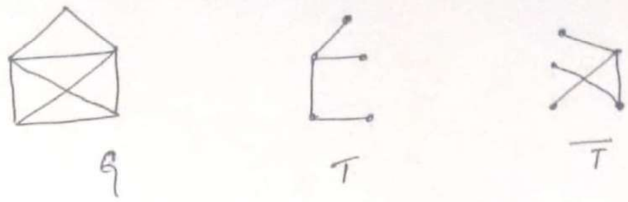
5) A spanning tree is defined only for a connected graph.

6). In a disconnected graph of n vertices each component have a spanning tree. These spanning trees together form a spanning forest.

7) Every tree is a spanning tree of itself.

8). A connected graph can have more than one spanning tree.

(*) Ex



$\therefore \; G = T \cup \bar{T}$

Theorem 1 :- A graph is connected if and only if it has a spanning tree.

Proof :- Let G be a connected graph.

If G has no cycles then G is a tree & G itself is a spanning tree.

If G has cycles, delete an edge from each cycle.

The resulting graph G' is cycle-free, connected & contains all vertices. ⇒ G' is a spanning tree of G.

Thus G has a spanning tree.

Conversely,

Suppose a graph G has a spanning tree T.

⇒ T is a tree. ⇒ ∃ a path between every pair of vertices of T.

Since T is a spanning tree ⇒ T contains all the vertices

∴ there is a path between every pair of vertices of in G.

⇒ G is connected.

Theorem 2 :- With respect to any of its spanning trees, a connected graph of n-vertices & m edges has n-1 branches and m-n+1 chords.

Proof:- Let G be a connected graph of n vertices & m edges, T be a spanning tree of G then T contains all vertices of G. $\Rightarrow$ T has n vertices.

$\Rightarrow$ T has $n-1$ edges.

$\Rightarrow$ T has $n-1$ branches.

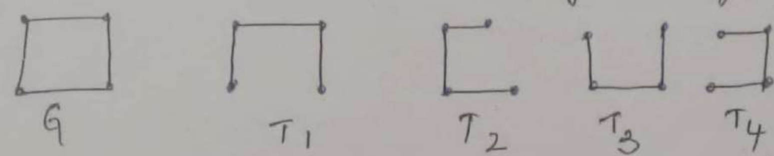
The no of edges in G that are not in T is $m-(n-1)$

$\Rightarrow$ G has $m-n+1$ chords.

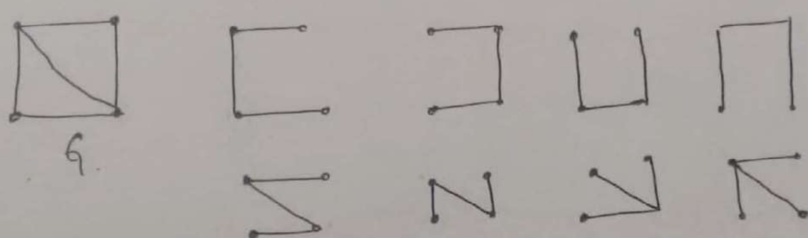
Hence the proof

Ex 1 Find all spanning trees of graph given.

(i) QP



(ii) QP



Ex 2 prove that a pendant edge in a connected graph G is contained in every spanning tree of G.

Soln: A pendant edge is an edge whose one end vertex is a pendant vertex.

let e be a pendant edge of a connected graph G & let v be the corresponding pendant vertex then e is the only edge that is incident on v.

Suppose there is a spanning tree T of G for which e is not a branch. Then T cannot contain the vertex v.

This is not possible because T must contain every vertex of G

(Hence there is no spanning tree of G for which e is not a branch.)

Hence a pendant edge in a connected graph is contained in every spanning tree of G.

Ex 3 Show that a Hamilton path is a spanning tree.

Soln:- Hamilton path P in a connected graph G. if such a path exists, then it contains every vertex of G. and that if G has n - vertices then P has n-1 edges. Thus P is a connected subgraph of G with n vertices & n-1 edges.

$\Rightarrow$ P is a tree & contains all vertices of G

$\Rightarrow$ P is a spanning tree of G.

Ex 4. How many edges are to be removed from the K_5 & $K_{4.4}$ in order to obtain their spanning trees.

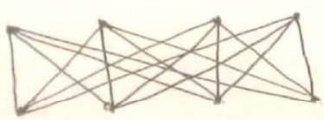
Soln:-

K_5

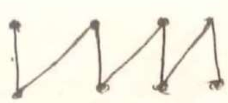
T

K_5 has 10 edges out of which 4 edges form a S. Tree

$\therefore$ 6 edges are removed.



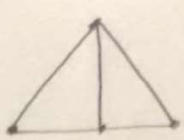
$K_{4,4}$



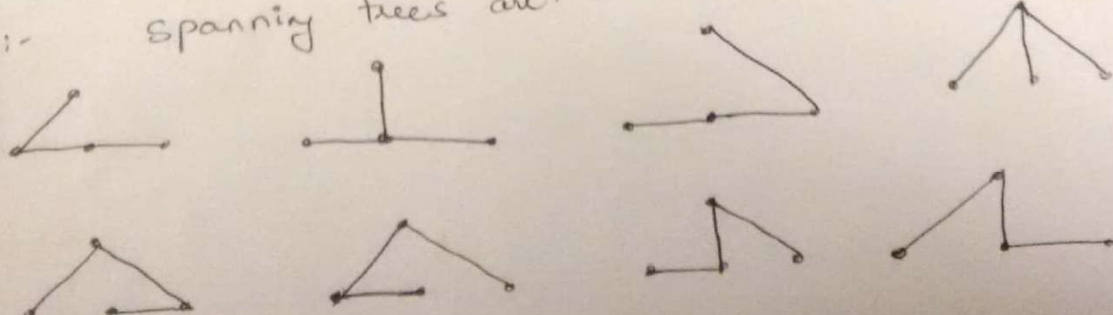
T.

$K_{4.4}$ has 16 edges.
7 are there in S.T
$\therefore$ 9 edges are removed.

Ex 5 Find all the spanning trees of the graph given below:



Soln:- Spanning trees are:



Algorithms for constructing Spanning trees:-

A Simple graph G has a Spanning tree iff G is Connected. Instead of constructing a Spanning tree by removing edges. Spanning tree can be built up by successively adding edges.

Two algorithms based on this principle for finding a Spanning tree are 1) Depth first Search (DFS) and
2) Breadth first search (BFS).

1) Depth first search (DFS):-

Step 1 :- Arbitrarily choose a vertex from the vertices of the graph and designate it as the root.

2:- Form a path starting at this vertex by successively adding edges as long as possible where each new edge is incident with the last vertex in the path without producing any cycle.

3: If the path goes through all vertices of the graph the tree consisting this path is a spanning tree.

otherwise move back to the next to last vertex in the path and if possible form a new path starting at this vertex passing through vertices that were not already visited.

4: If this cannot be done move back another vertex in the path that is two vertices back in the path & repeat.

5: Repeat this procedure. begining at the last vertex visited, moving back up the path one vertex at a time forming new paths that are as long as possible until no more edges can be added.

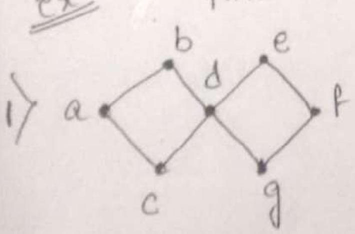
6) This process ends since the graph has a finite number of edges and is connected.

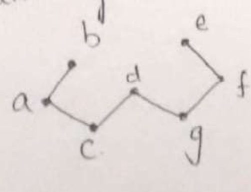
Thus a spanning tree is produced.

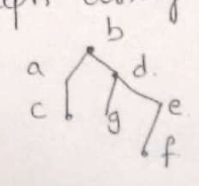
BFS. Algorithm:-

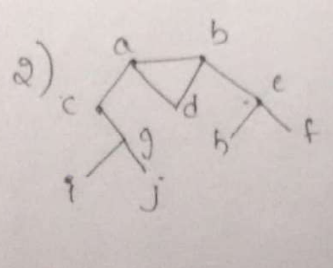
1) Arbitrarily choose a vertex and designate it as the root. then add all edges incident to this vertex such that the addition of edges does not produce any cycle.
2) The new vertices added at this stage become the vertices at level 1 in the spanning tree, arbitrarily order them.
3) For each vertex at level 1 visited in order add each edge incident to this vertex to the tree as long as it does not produce any cycle
4) Arbitrarily order the children of each vertex at level 1. This produces the vertices at level 2 in the tree.
5) Continue the same produce until all the vertices in the tree have been added. A spanning tree is produced since we have produced a tree without cycle containing every vertex of the graph.

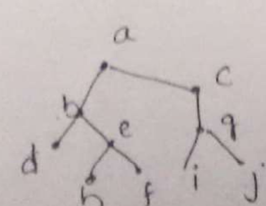
Ex Find the spanning tree of the graph using DFS, BFS.

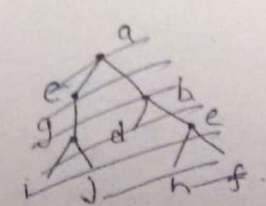
1) 

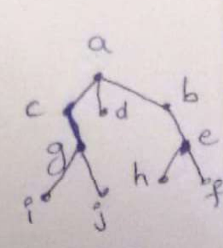

DFS


BFS

2) 


DFS


~~BFS~~.


BFS

Prefix codes and weighted trees :-

Sequence :- A sequence is a set whose elements are listed in order. The number of elements contained in a sequence is called its length.
A sequence consisting of only 0 and 1 is called a binary sequence or a binary string. Ex :- *

[NOTE :- Suppose a message consisting of the letters a, e, n, r, t is to be transmitted. & a = 1, e = 0, n = 10, r = 01, t = 101 be the coding scheme for these letters. Suppose a message 'eat' is to be transmitted then the coded form of the message is given by 01101.

This binary sequence may not be decoded properly since the sequence 01101 can be decoded as rt or rar or eaar. Hence this coding scheme cannot be used in transmissions.

Suppose a = 10, e = 0, n = 1101, r = 111, t = 1100.
The mes eat is transmitted as the binary sequence 0101100. clearly the decoding of this sequence yeilds eat & no other message. eat $\xrightarrow{\text{coding}}$ 0101100 $\xrightarrow{\text{decoding}}$ eat

∴ In the first of the coding schemes the code 1 is assigned to a, & the code 10 is to n & 101 to t
Thus the code assigned to t contains the codes assigned to a and n as prefixes.

In the second coding scheme, the code of any letter is not a prefix of the code of any other letter.]

* Ex :- 01, 001, 101, 1001, 11111 are binary strings of length 2, 3, 3, 4, 5 resp.

Binary sequences are used as codes for messages sent through transmitting Channel.

Prefix codes :-

Let P be a set of binary sequences that represent a set of symbols, then P is called a prefix code if no sequence in P is the prefix of any other sequence in P.

Ex

$P_1 = \{10, 0, 1101, 111, 1100\}$ is a prefix code

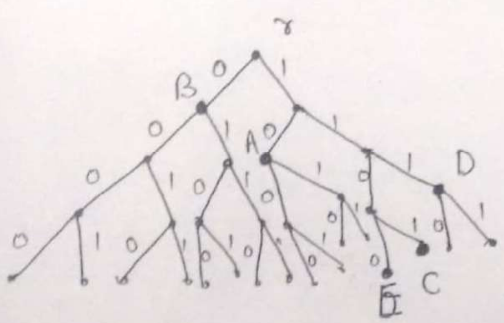
$P_2 = \{000, 001, 01, 10, 11\}$ — " —

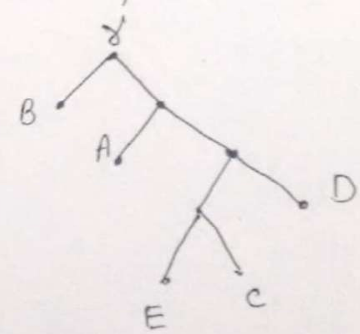
$A_1 = \{01. 0. 101. 10. 1\}$ is not a prefix code

$A_2 = \{1. 00. 01. 000. 0001\}$ is not a prefix code.

Prefix codes can be represented by binary trees as shown below:

Consider P_1, the longest sequence has length 4.
Now construct a full binary tree of height 4.
assign the symbol 0 to every left edge from its parent vertex & 1 to every right edge from its parent vertex





The subtree extracted from the ful binary tree ~~that~~ contains the root r and the vertices A . B . C . D . E
This subtree represents the prefix code given by P_1.

a = 10 e = 0 n = 1101 r = 111 t = 1100 ✓

a = 1, e = 0, n = 10, r = 01 t = 101

BEAD $\xrightarrow{\text{coding}}$ 0110010111 $\xrightarrow{\text{decoding}}$ BEAD

Ex. Consider the prefix code

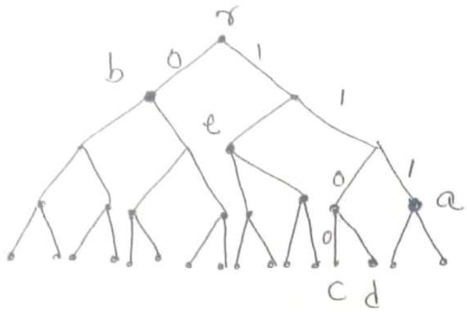
$a = 111 \quad b = 0 \quad c = 1100 \quad d = 1101 \quad e = 10$.

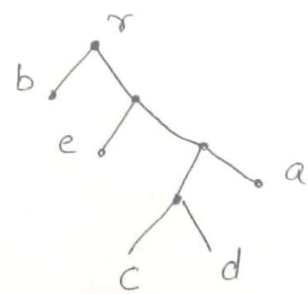
Using this code decode the following sequences and draw the binary tree that represents prefix code?

a) 100 11 11 101 — e b a d

(b) 10 111 100 11 0 0 0 1101 — e a e b c b d

(c) 1101 11 1 11 10010 — d a d e





Ex 2 given $a = 00$, $b = 01$. $c = 101$. $d = x10$. $e = yz1$.

where determine x y & z so that the given code is a prefix code. Hence draw the binary tree.

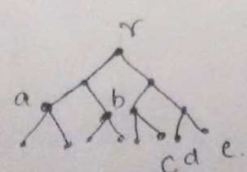
Soln:- If $x = 0$. then $d = 010$ contains the code 01 for b.

$\therefore$ $x = 1$ then $d = 110$

If $y = 0$	$z = 0$	$e = 001$	not 'a'
$y = 0$	$z = 1$	011	not 'b'
$y = 1$	$z = 0$	101	c
$y = 1$	$z = 1$.	111	is the code.

$x = 1$, $y = 1$. $z = 1$.

$a = 00$, $b = 01$. $c = 101$. $d = 110$ $e = 111$



3. Find the prefix code represented by

$\{00, 011, 0100, 0101, 10, 110, 111\}$

Weighted trees :-

Consider a set of n-positive integers $\omega_1, \omega_2 \cdots \omega_n$ where $\omega_1 \leq \omega_2 \leq \cdots \leq \omega_n$. Suppose we assign these integers to n-leaves of a complete binary tree $T = (V, E)$ in any 1-1 manner. The resulting tree is called a complete, weighted binary tree with $\omega_1, \omega_2 \cdots \omega_n$ as weights.

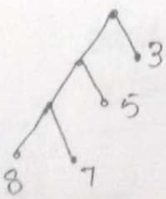
If $l(\omega_i)$ is the level number of the leaf of T to which the weight ω_i is assigned, then $\omega(T)$ defined by.

$\omega(T) = \sum_{i=1}^{n} \omega_i \cdot l(\omega_i)$. is called the weight of the tree ~~of~~ T

For a given set of weights, the value of $\omega(T)$ depends upon the tree T chosen.

Ex

1)



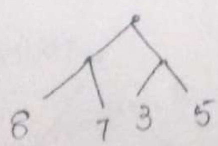
T_1

$\omega(T_1) = 1 \times 3 + 2 \times 5 + 7 \times 3 + 8 \times 3 = 58$

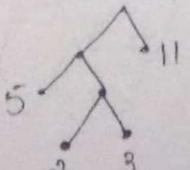
$\omega(T_2) = 8 \times 2 + 7 \times 2 + 3 \times 2 + 5 \times 2 = 46$

2)

T_2



3)



$11 \times 1 + 5 \times 2 + 2 \times 3 + 3 \times 3$

$11 + 10 + 6 + 6 = 33$

Scanned by CamScanner

Optimal Tree : Given a set of weights, if we consider the set of all complete binary trees to whose leaves these weights are assigned, then tree in this set which carries the minimum weight is called an optimal tree for the weights. For a given set of weights, there can be more than one optimal tree.

Huffman's Procedure :-

An optimal tree for a given set of weights can be constructed as follows :-

1) Arrange the weights in non-decreasing order and assign them to isolated vertices.
2) Select 2 vertices with minimum weights. Add these 2 vertices to get a new vertex. Draw a tree with new vertex as root and the selected vertices as children. Leave the remaining vertices undisturbed.
3) Repeat the procedure in step (2) until all the vertices and subtrees are connected to get a complete weighted binary tree.
4) The tree obtained is the optimal tree for the given set of weights and is also known as Huffman's tree. This tree is not unique.

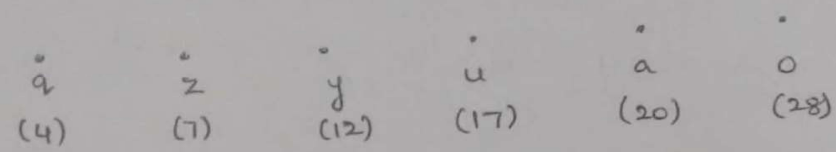
Optimal Prefix code :- Optimal tree can be used to obtain a prefix code for the symbols representing its leaves. For this purpose, we first label the symbols 0, 1 to its edges by labelling procedure indicated earlier. Then all the vertices & the leaves of tree can be identified by binary sequences. The binary sequences thru' which the leaves are identified, yield a prefix code for the symbols representing the leaves. This prefix code is known as optimal prefix code.

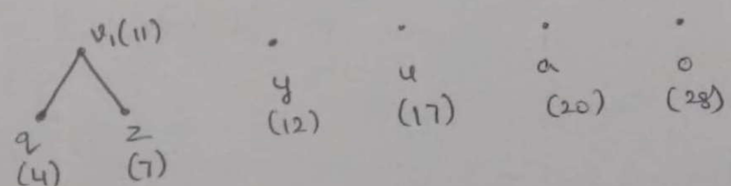
Note :- optimal prefix code is not unique since optimal tree is not unique.

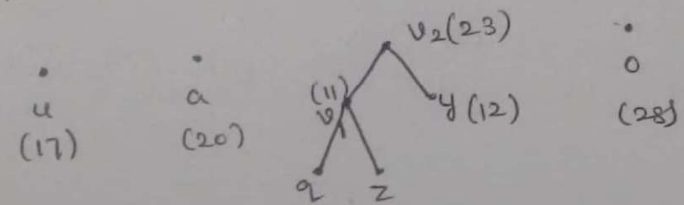
Problems :-

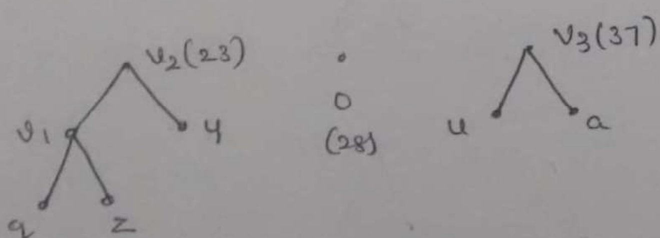
1) Construct an optimal prefix code for the symbols a, o, q, u, y, z that occur with frequencies 20, 28, 4, 17, 12, 7 respectively.

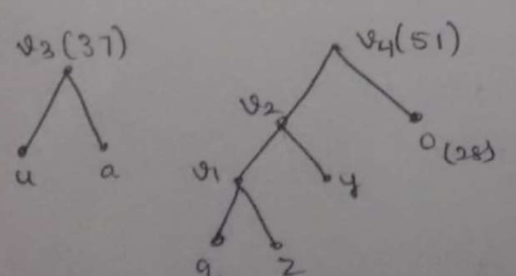
Soln:- Arranging these symbols along with their frequencies in non-decreasing order, treating frequencies as the weights and the corresponding symbols as the isolated vertices :

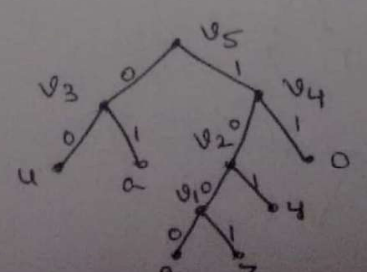












is the optimal tree (Huffman tree)

∴ optimal prefix code for the given symbols is :

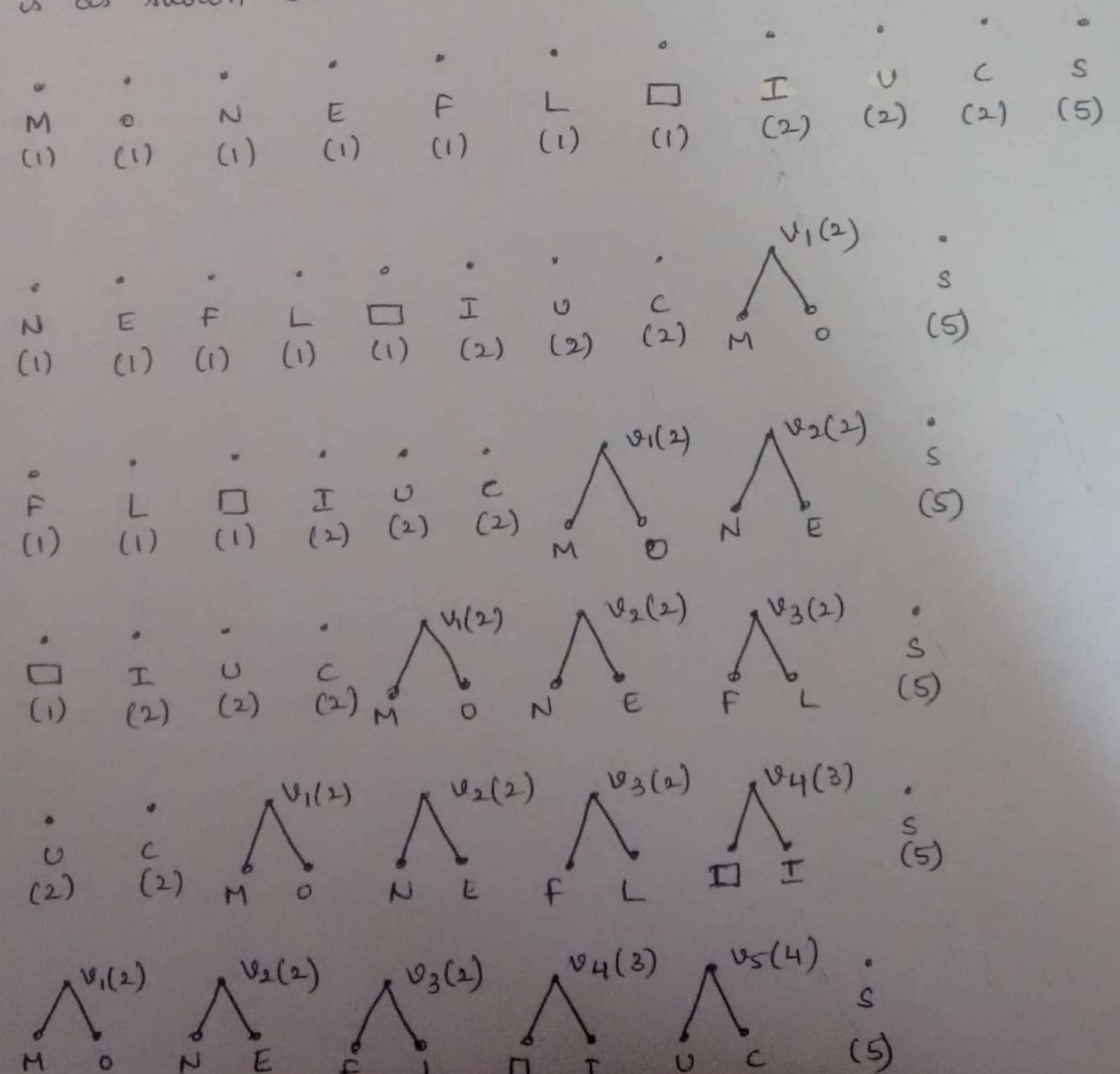
a : 01 u : 00 o : 11

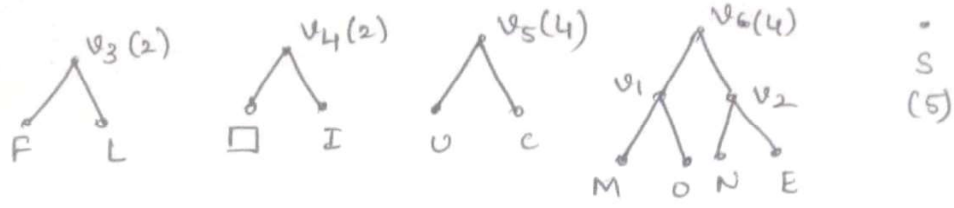
y : 101 q : 1000 z : 1001 .

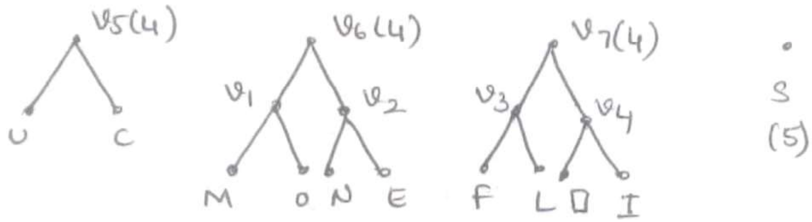
2) obtain an optimal Prefix code for the message MISSION SUCCESSFUL . INDICATE THE CODE -

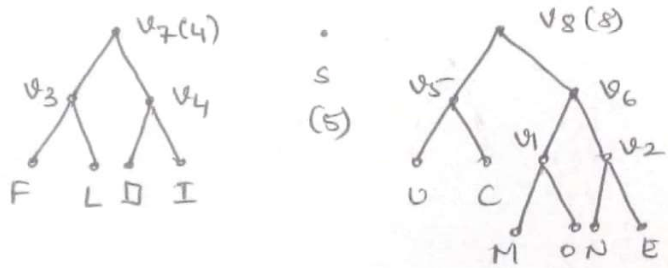
Soln :- The given message consists of the letters M , I , S , O , N , U , C , E , F , L with frequencies 1 , 2 , 5 , 1 , 1 , 2 , 2 , 1 , 1 , 1 resp. Also there is a blank space (□) b/w the 2 words of the message .

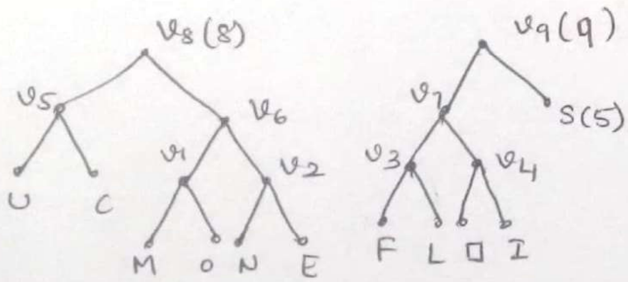
∴ Arranging these letters and □ in non-decreasing order of their weights (frequencies) in the form of isolated vertices is as shown below :

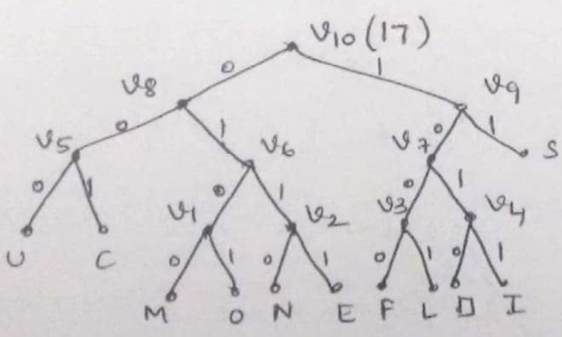












is the optimal tree.

M : 0100   I : 1011   S : 11   O : 0101   N : 0110

U : 000   C : 001   E : 0111   F : 1000   L : 1001   □ : 1010

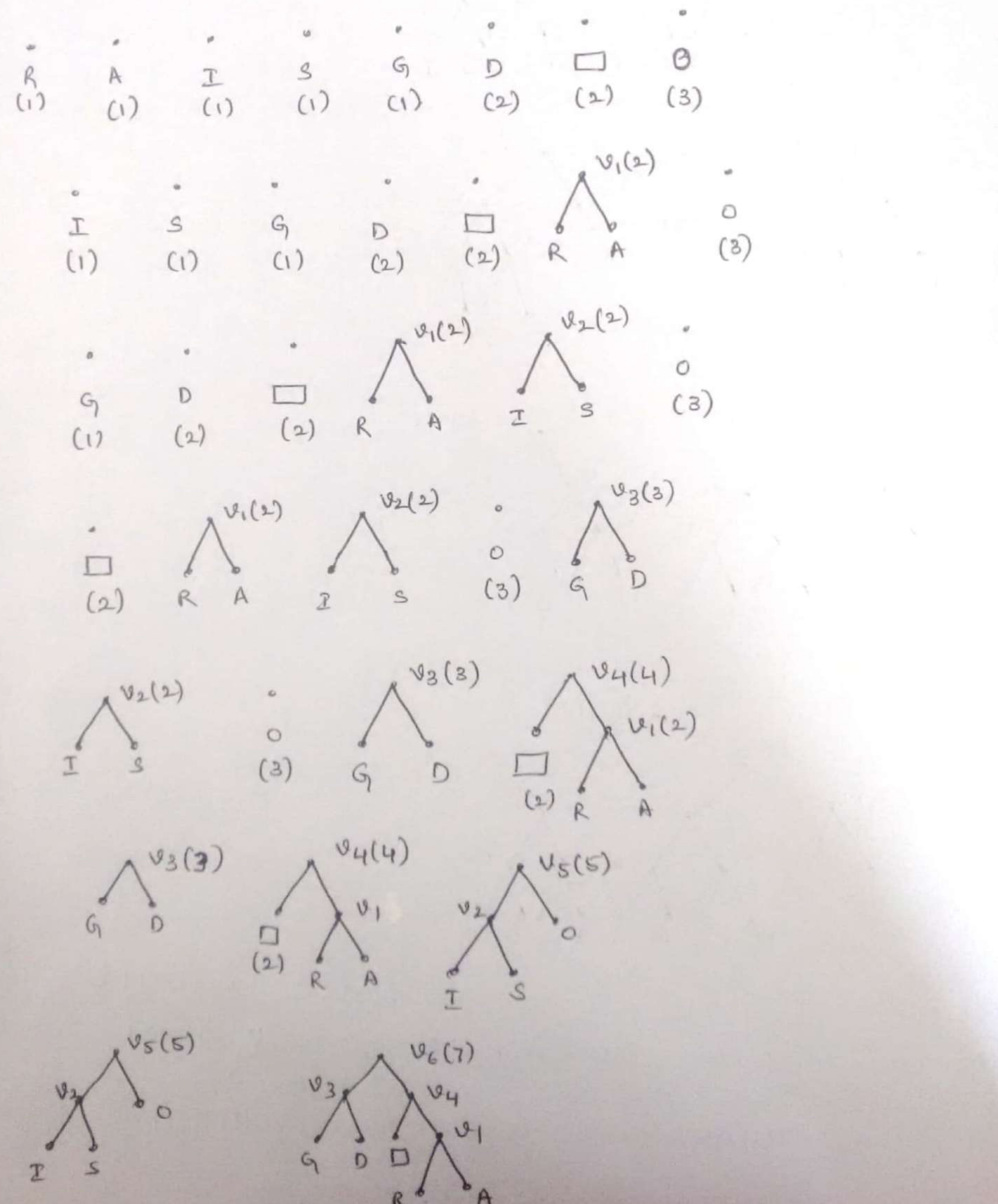
Code : 0100101111111011010101101010110000010010111111110000001001

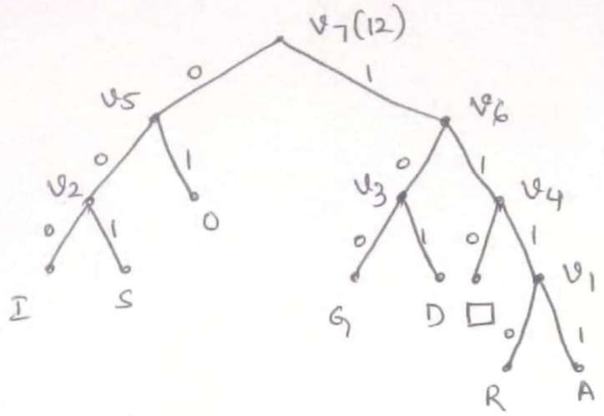
Q8

3) Obtain an optimal prefix code for the message ROAD IS GOOD. Indicate the code.

Sol^n:- The given message consist of the letters R, O, A, D, I, S, G with frequencies 1, 3, 1, 2, 1, 1, 1 resp. Also there are 2 blank spaces (□) b/w the words of the message.

∴ Arranging all these in non-decreasing order:





is the optimal tree.

R : 1110 O : 01 A : 1111 D : 101

□ : 110 I : 000 S : 001 G : 100

Code : 11100111111011100000011101000101101.

4) obtain an optimal prefix code for the symbols A, B, C, D, E, F, G, H, I, J that occur with frequencies : 78, 16, 30, 35, 125, 31, 20, 50, 80, 3.

QP

5) obtain an optimal prefix code for the letters of the following words and hence indicate the code :

QP i) ENGINEERING.

QP ii) FALL ON THE WALL.