UNIT-V TREES

Trees and Their Properties:

A tree is a connected acyclic graph i.e. a connected graph having no cycle. Its edges are called branches. Fig 1. are examples of trees with atmost five vertices. Fig 2.(a) and (b) are not trees, since they have cycles.

A tree with only one vertex is called a **trivial tree** otherwise T is a **non trivial tree**.

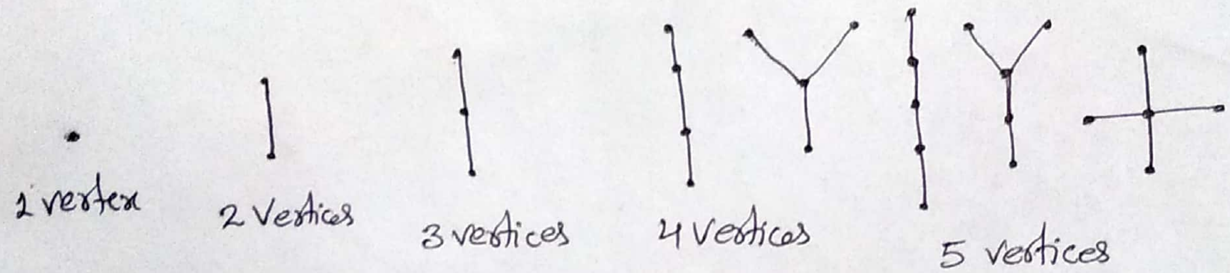


Fig 1.

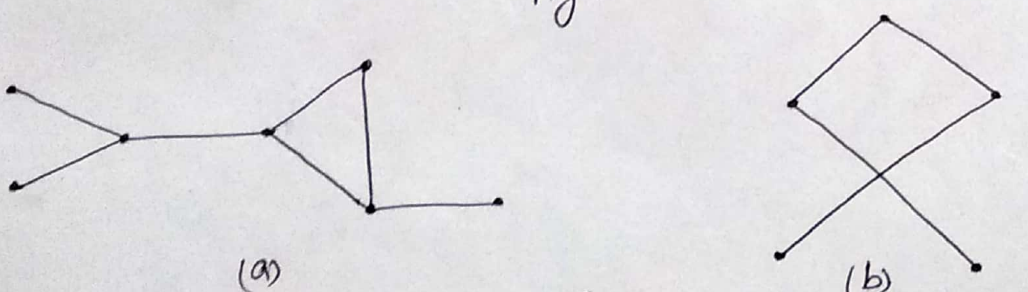


Fig 2.

Characterisations: Trees have many equivalent characterisations, any of which could be taken as the definition. A few simple and important theorems on the general properties of trees are given below.

Theorem 1. There is one and only one path between every pair of vertices in a tree, T.

Scanned by CamScanner

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

Theorem 3. For any positive integer n, if G is a connected graph with n vertices and $n-1$ edges, then G is a tree.

Ex: A tree has two vertices of degree 2, one vertex of degree 3 and three vertices of degree 4. How many vertices of degree 1 does it have?

Sol. Let x be the required no. Now, total no. of vertices

$$= 2+1+3+x = 6+x$$

Hence the no. of edges is [In a tree $|E| = |V-1|$]

$$6+x-1 = 5+x$$

The total degree of the tree

$$= 2\times2 + 1\times3 + 3\times4 + 1\times x = 19+x$$

So, the no. of edges are $\frac{19+x}{2}$ [$2e = \sum \deg(v_i)$]

Now,

$$\frac{19+x}{2} = 5+x$$

$$19+x = 10+2x \quad \text{or} \quad \boxed{x=9}$$

Thus, there are 9 vertices of degree one in the tree.

Spanning Tree:

A subgraph T of a connected graph $G(V,E)$ is called a spanning tree if

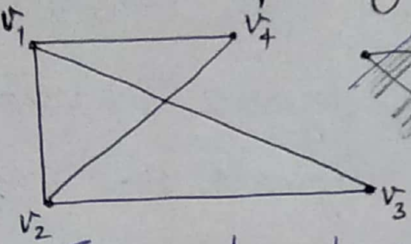
(i) T is a tree and

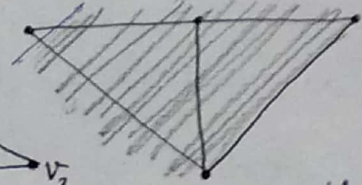
(ii) T includes every vertex of G i.e. $V(T) = V(G)$. If $|V| = n$ and $|E| = m$, then the spanning tree of G must have n vertices and hence $n-1$ edges. We must remove $m-(n-1)$ edges from G to obtain a spanning tree. In removing

Scanned by CamScanner

these edges one must ensure that the resulting graph remain connected and further there is no circuit in it.

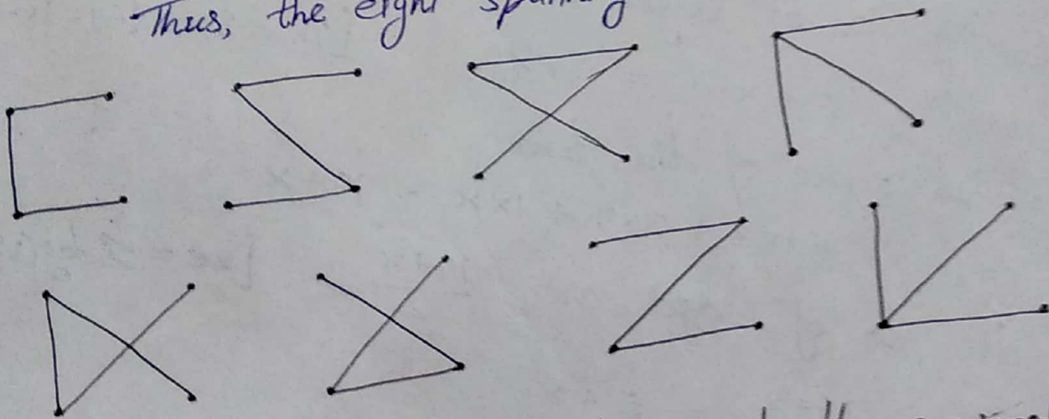
Ex: 1. Find all spanning trees of the graph G:



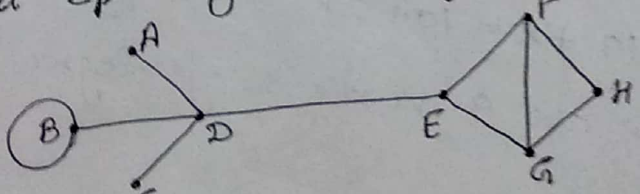


Sol The no. of vertices in the graph, $n=4$. The no. of edges, $m=5$. So, the no. of edges to be deleted to get the spanning trees $= m-n+1 = 5-4+1 = 2$.

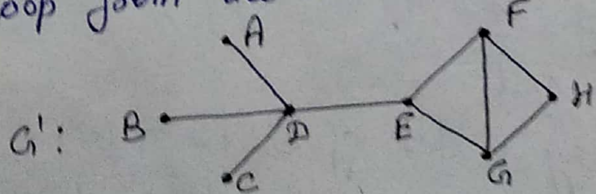
Thus, the eight spanning trees are



Ex: 2. Find all spanning trees of the connected graph G:



Sol. Since the vertex B contains self-loop, we remove the self-loop from the vertex B, and G becomes



The graph is connected and it has 9 edges and 8 vertices so $9-8+1 = 2$ edges has to be deleted from the graph to get a spanning tree which is connected and does not contain cycle.

Scanned by CamScanner

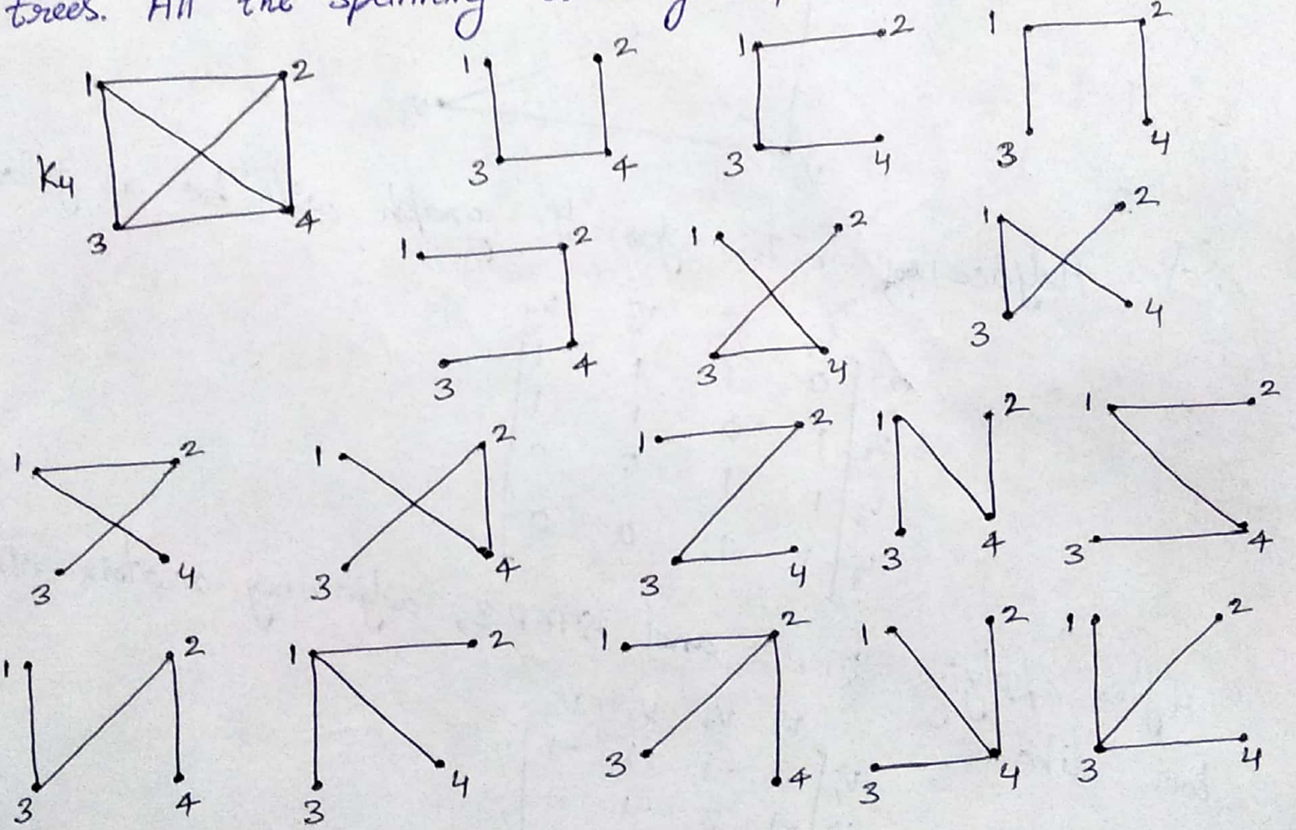
Total Number of Spanning Trees in a Graph

If a graph is a complete graph with n vertices, then total no. of spanning trees is n^{n-2} where n is the no. of vertices in the graph. It follows the Caylay's theorem.

Caylay's Theorem: The complete graph K_n has n^{n-2} different spanning trees.

Ex: Give all the spanning trees of K_4.

Sol. Here $n = 4$, so there will be $4^{4-2} = 16$ different spanning trees. All the spanning trees of K_4 are



What if graph is not complete?

Follow the given procedure:-

STEP 1: Create Adjacency Matrix for the given graph.

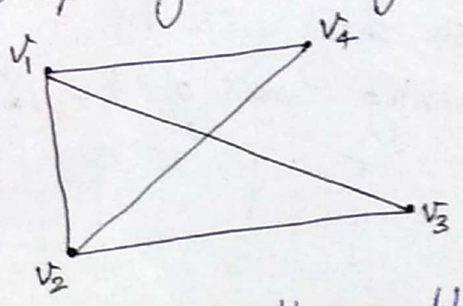
STEP 2: Replace all the diagonal elements with the degree of vertices. For eg. element at (1,1) position of adjacency matrix will be replaced by the degree of vertex 1, element at (2,2) position of adjacency matrix will be replaced by the degree of vertex 2, and so on.

STEP 3: Replace all non-diagonal 1's with -1.

STEP 4: Calculate co-factor for any element.

STEP 5: The cofactor that you get in the total no. of spanning trees for that graph.

Ex: 1. Find all the spanning trees of the graph shown:



Sol Adjacency matrix for the graph will be as follows:

$$\begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & 0 & 1 & 1 & 1 \\ v_2 & 1 & 0 & 1 & 1 \\ v_3 & 1 & 1 & 0 & 0 \\ v_4 & 1 & 1 & 0 & 0 \end{array}$$

After applying STEP 2 and STEP 3, adjacency matrix will look like

$$\begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ \hline v_1 & 3 & -1 & -1 & -1 \\ v_2 & -1 & 3 & -1 & -1 \\ v_3 & -1 & -1 & 2 & 0 \\ v_4 & -1 & -1 & 0 & 2 \end{array}$$

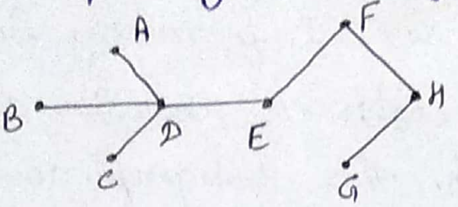
The co-factor for (1, 1) $= 3(4-0)+1(-2+0)-1(0+2)$

$= 12-2-2 = 8.$

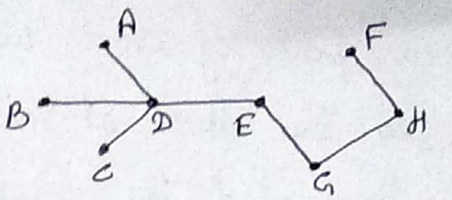
Hence total no. of spanning tree that can be formed is 8.

The spanning trees of the graph G' are:

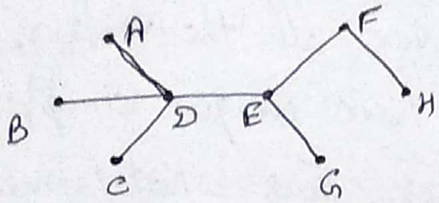
(i)



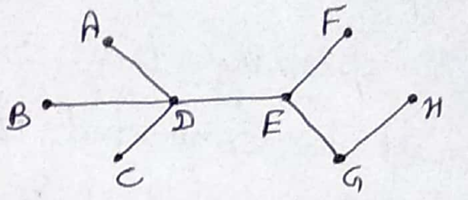
(ii)



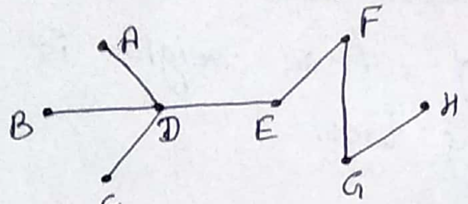
(iii)



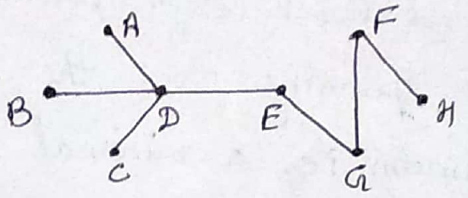
(iv)



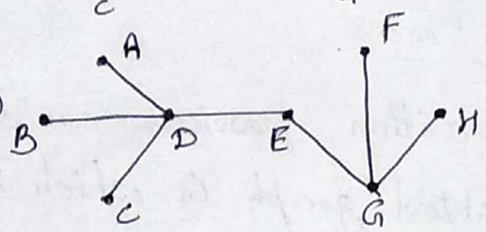
(v)



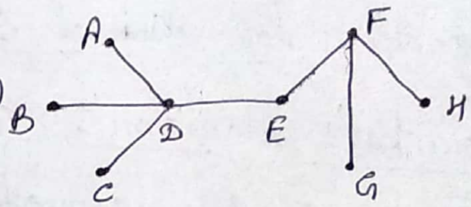
(vi)



(vii)



(viii)



Weighted Graph: A weighted graph is a graph G in which each edge e has been assigned a non-negative number w(e), called the weight (or length) of e. Fig 1. Shows a weighted graph. The weight (or length) of a path in such a weighted graph G is defined to be the sum of the weights of the edges in the path. Many optimisation problems amount to finding, in a suitable weighted graph, a certain type of subgraph with minimum (or maximum) weight.

Minimal Spanning Trees:

Let G be a connected weighted graph. The weight of a spanning tree of G is the sum of the weights of the edges. A minimal spanning tree of G is a spanning tree of G with minimum weight. The weighted graph G of Fig 1. shows

Scanned by CamScanner

six cities and the costs of laying railway links between certain pairs of cities. We want to set up railway links between the cities at minimum costs. The solution can be represented by a subgraph. This subgraph must be a spanning tree since it covers all the vertices, it must be connected, it must have unique simple path between each pair of vertices. Thus what is needed is a spanning tree the sum of whose weights is minimum i.e., a minimal spanning tree.

Algorithm for Minimal Spanning Trees

Kruskal's Algorithm: This algorithm provides an acyclic subgraph T of a connected weighted graph G which is a minimal spanning tree of G. The algorithm involves the following steps:

Input: A connected weighted graph G.

Output: A minimal spanning tree T.

Step 1. List all the edges (which do not form a loop) of G in non-decreasing order of their weights.

Step 2. Select an edge of minimum weight (If more than one edge of minimum weight, arbitrary choose one of them). This is the first edge of T.

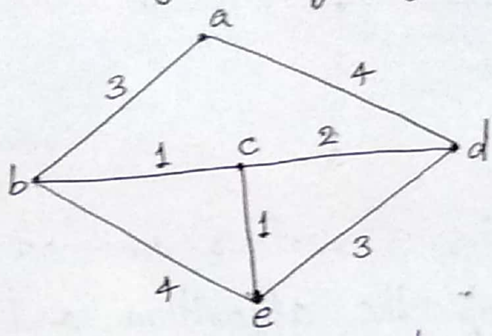
Step 3. At each stage, select an edge of minimum weight from all the remaining edges of G if it does not form a circuit with the previously selected edges in T. Include the edge in T.

Scanned by CamScanner

Step 4. Repeat step 3 until n-1 edges have been selected, when n is the no. of vertices in G.

Ex1. Show how Kruskal's algorithm find a minimal spanning tree for the graph of Fig2.

Sol.

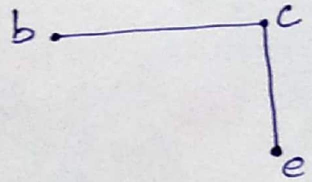


Step 1: List the edges in non-decreasing order of their weights, as in table

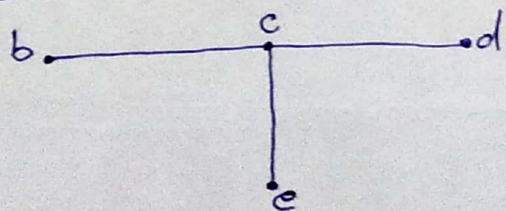
Edge:	(b,c)	(c,e)	(c,d)	(a,b)	(e,d)	(a,d)	(b,e)
Weight:	1	1	2	3	3	4	4

Step 2: Select the edge (b,c) since it has the smallest weight, inculde it in T.

Step 3: Select an edge with the next smallest weight (c,e) since it does not form circuit with the existing edges in T, so include it in T.



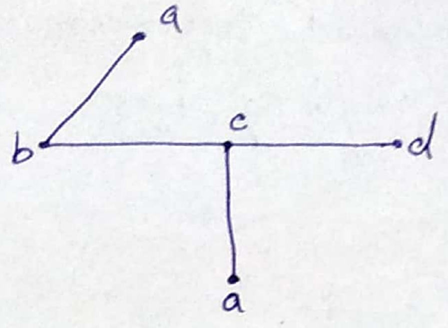
Step 4: Select an edge with the next smallest weight (c,d) since it does not form circuit with the existing edges in T, so include it in T.



Scanned by CamScanner

50

Step 5: Select an edge with the next smallest weight (a,b) since it does not form circuit with the existing edges in T, so include it in T.



Since G contains 5 vertices and we have chosen 4 edges, we stop the algorithm and the minimal spanning tree is produced.

Scanned by CamScanner

<u>Rooted Trees</u>: A rooted tree is a tree in which a particular vertex is distinguished from the others and is called the root. In contrast to natural trees, which have their roots at the bottom, in graph theory rooted trees are typically drawn with their roots at the top. First, we place the root at the top. Under the root and on the same level, we place the vertices that can be reached from the root on a simple path of length 1. Under each of these vertices and on the same level, we place vertices that can be reached from the root on a simple path of length 2. We continue in this way until the entire tree is drawn. We give definitions of some terms related to it.

1. The **level** of a vertex is the number of edges along the unique path between it and the root. The level of the root is defined as 0. The vertices immediately under the root are said to be in level 1 and so on.

2. The **height** of a rooted tree is the maximum level to any vertex of the tree. The **depth** of a vertex v in a tree is the length of the path from the root to v.

3. Given any internal vertex v of a rooted tree, the **children** of v are all those vertices that are adjacent to v and are one level further away from the root than v. If w is a child of v, the v is called the **parent** of w, and two vertices that are both children of the same parent are called **siblings**.

4. If the vertex u has no children, then u is called a **leaf** (pendant or a terminal vertex). A non-pendant

Scanned by CamScanner

vertex in a is called an **internal vertex.**

5. The **descendants** of the vertex u is the set consisting of all the children of u together with the descents of those children. Given vertices v and w, if v lies on the unique path between w and the root, then v is an **ancestor** of w and w is a descendant of v.

These terms are illustrated as Fig.

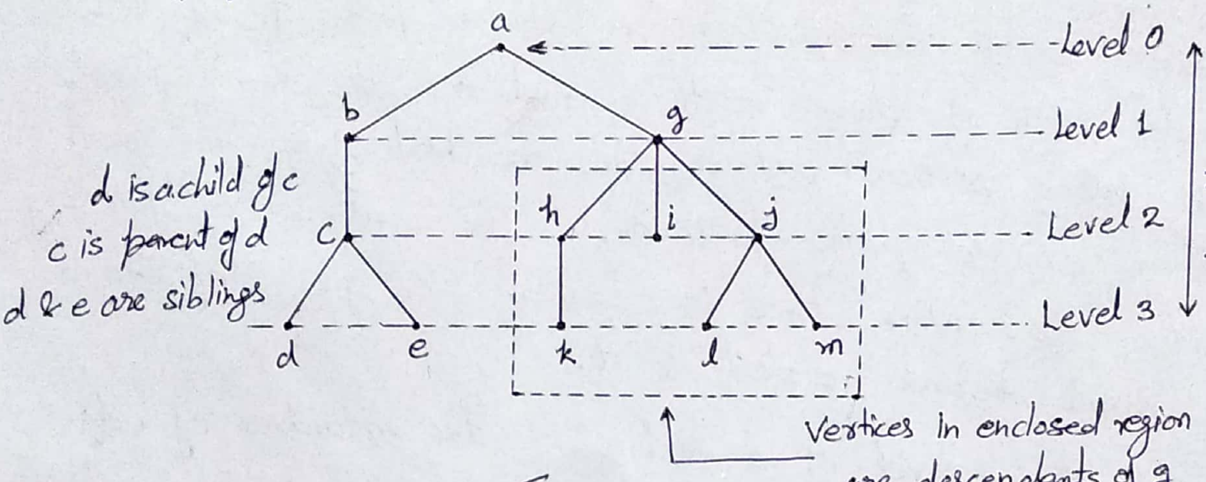
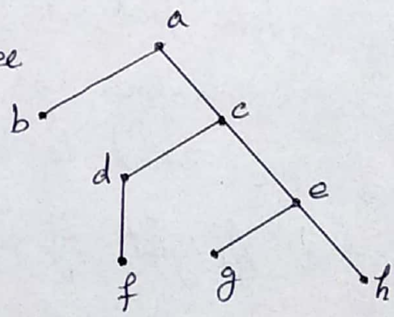


Fig.

Ex.1. Consider the rooted tree



(a) What is the root of T?

Sol. Vertex a is distinguished as the only vertex located at the top of the tree. Therefore, a is the root.

(b) Find the leaves and the internal vertices of T.

Sol The leaves are those vertices that have no children. These are b, f, g, and h. The internal vertices are c, d and e.

(c) What are the levels of c and e.

Sol The levels of c and e are 1 and 2 respectively.