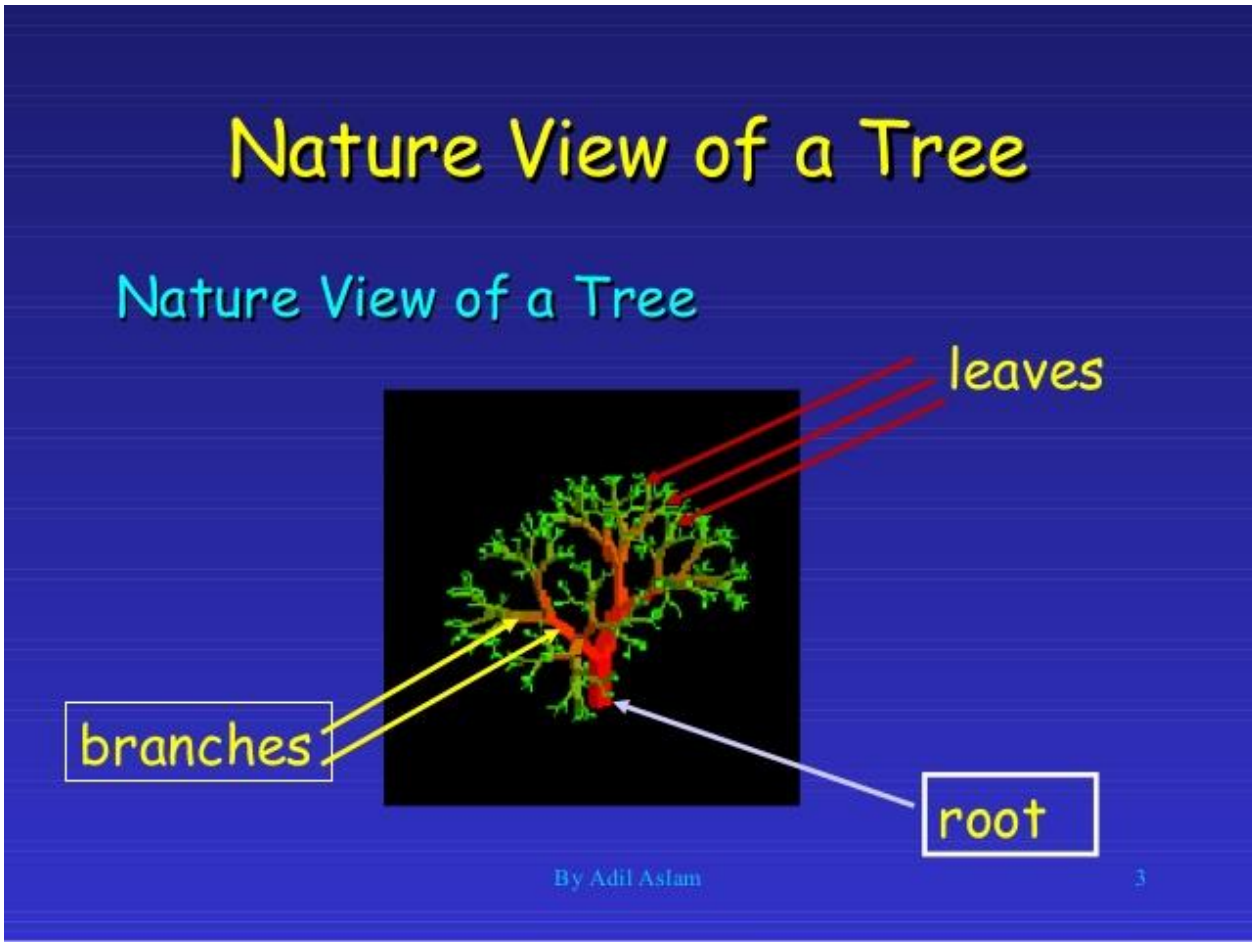
Trees

- A tree is a connected undirected graph with no simple circuits.
- Tree is a discrete structure that represents hierarchical relationships between individual elements or nodes.
- A graph is to be a circuit free if and only if, it has no circuits, a graph is called a tree if and only if, it is a circuit free and connected.



Computer Scientist's View

Computer Scientist's View

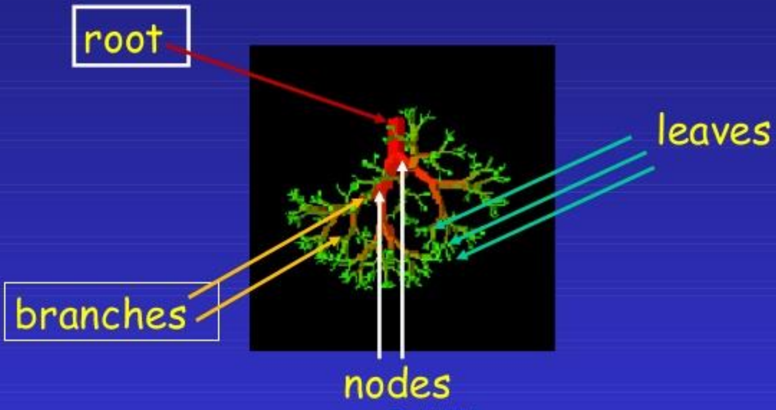
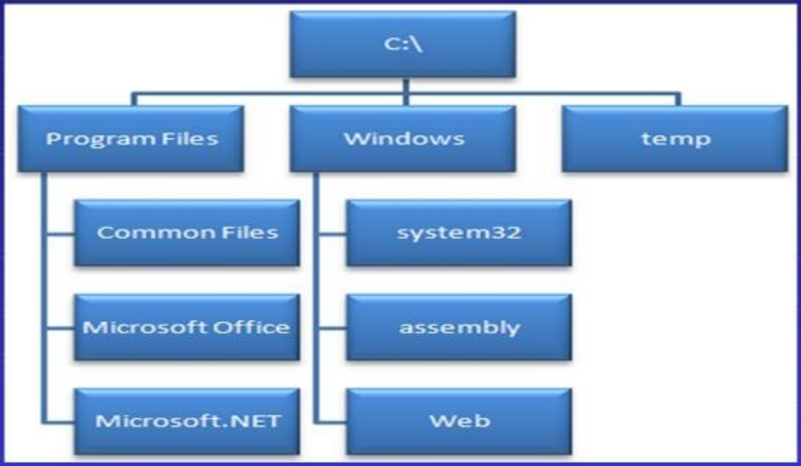
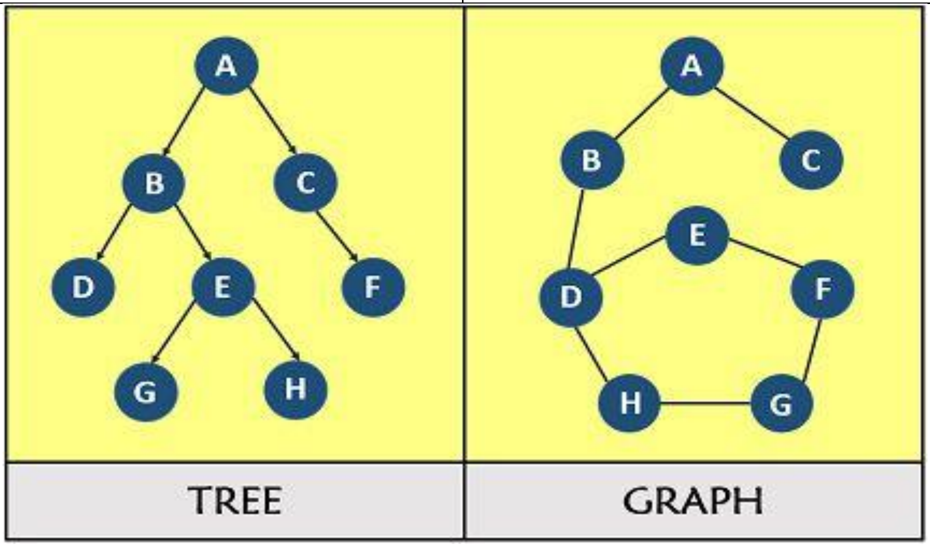
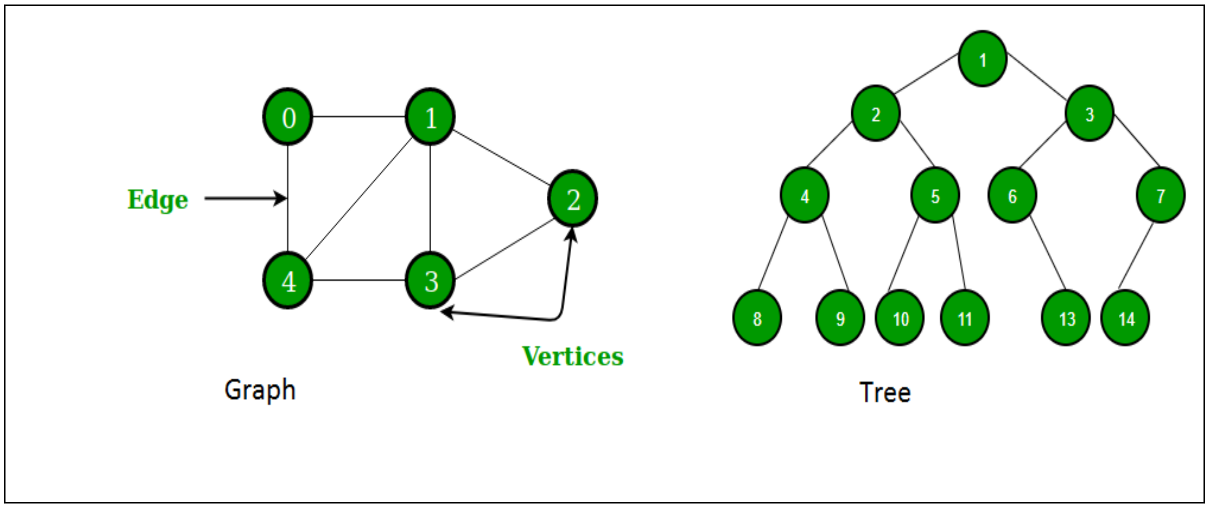


Figure : A Computer File System



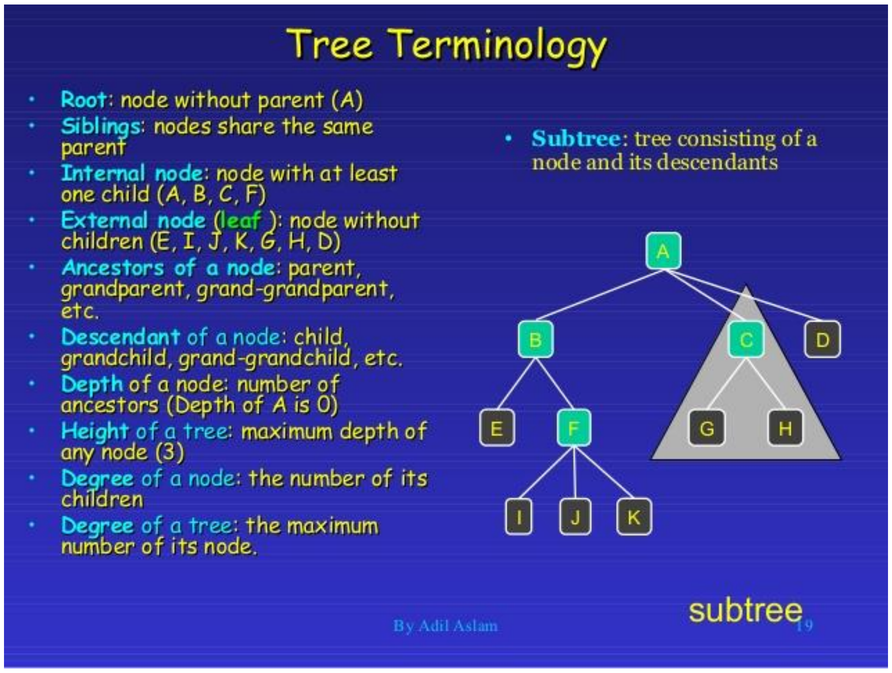
<u>Difference between Graph and Tree</u>

Tree	Graph
• All nodes must be connected • No loops and no circles • No multiple paths. Must be single path. • There is a relation between number of nodes and edges. For example if there are "n" nodes in a tree then number of edges must be "n – 1"	• It is not necessary that all nodes must have connections. • Loops and circles may exist • Multiple paths from one vertex to another may exists. • No relation between number of nodes and edges in a graph.
	



Properties of Trees:

- There is only one path between each pair of vertices of a tree.
- If a graph G there is one and only one path between each pair of vertices G is a tree.
- A tree T with n vertices has n-1 edges.
- A graph is a tree if and only if it a minimal connected.



Tree Terminology

- Root: node without parent (A)
- Siblings: nodes share the same parent
- Internal node: node with at least one child (A, B, C, F)
- External node (leaf): node without children (E, I, J, K, G, H, D)
- Ancestors of a node: parent, grandparent, grand-grandparent, etc.
- Descendant of a node: child, grandchild, grand-grandchild, etc.
- Depth of a node: number of ancestors (Depth of A is 0)
- Height of a tree: maximum depth of any node (3)
- Degree of a node: the number of its children
- Degree of a tree: the maximum number of its node.

- Subtree: tree consisting of a node and its descendants

By Adil Aslam 19

Properties of Rooted Trees

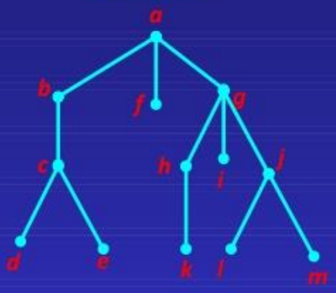
Parent – A vertex other than root is a parent if it has one or more children
- The parent of *c* is *b*

Children – If *A* is a vertex with successors *B* and *C*, then *B* and *C* are the children of *A*.
- The children of *a* is *b*, *f* and *g*

Siblings – Children with the same parent vertex.
- *h*, *i* and *j* are siblings

Level – the length of the unique path from the root to a vertex
- Vertex *a* is at level 0
- Vertices *d* and *e* is at level 3

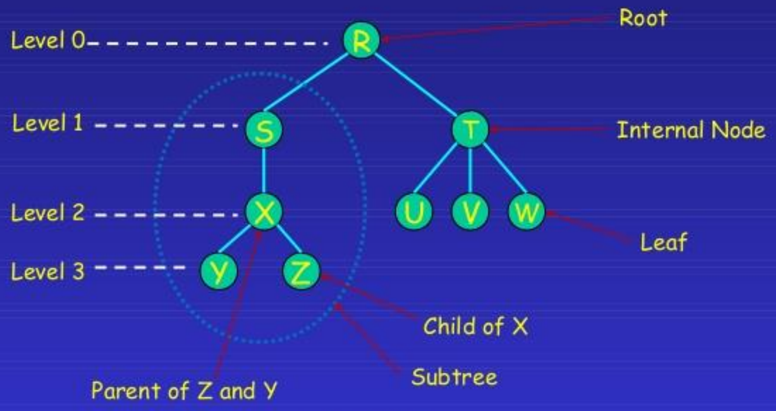
EXAMPLES



- **Height** – The maximum level of all the vertices
  - The height of this tree is 3.

Tree Anatomy

- The children of a node are, themselves, trees, called subtrees.



Types of Trees

General Trees: A graph which has no cycle is called an acyclic graph. A tree is an acyclic graph or graph having no cycles.

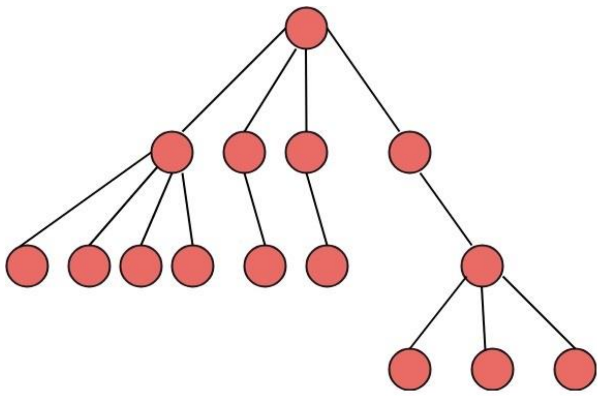


Fig:General Trees

Binary Trees: If the outdegree of every node is less than or equal to 2, in a directed tree than the tree is called a binary tree. A tree consisting of the nodes (empty tree) is also a binary tree.

Basic Terminology:

Root: A binary tree has a unique node called the root of the tree.

Left Child: The node to the left of the root is called its left child.

Right Child: The node to the right of the root is called its right child.

Parent: A node having a left child or right child or both are called the parent of the nodes.

Siblings: Two nodes having the same parent are called siblings.

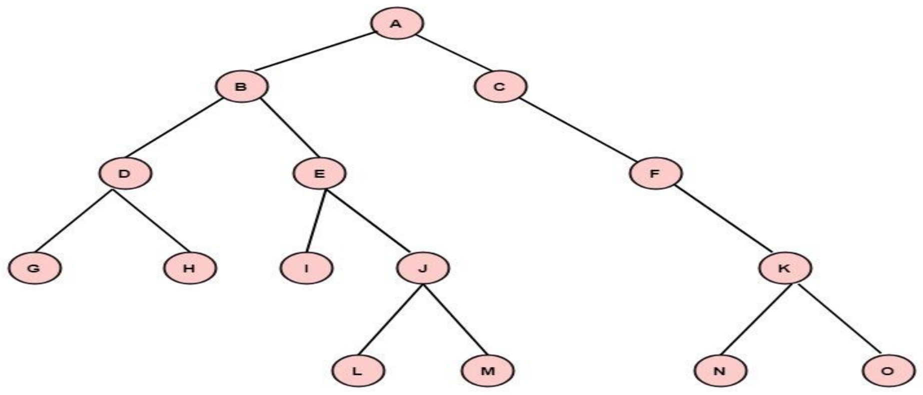
Leaf: A node with no children is called a leaf. The number of leaves in a binary tree can vary from one (minimum) to half the number of vertices (maximum) in a tree.

Descendant: A node is called descendant of another node if it is the child of the node or child of some other descendant of that node. All the nodes in the tree are descendants of the root.

Left Subtree: The subtree whose root is the left child of some node is called the left subtree of that node.

Example: For the tree as shown in fig:

- Which node is the root?
- Which nodes are leaves?
- Name the parent node of each node



Solution: (i) The node A is the root node.
(ii) The nodes G, H, I, L, M, N, O are leaves.
(iii)

Nodes	**Parent**
B, C	A
D, E	B
F	C
G, H	D
I, J	E
K	F
L, M	J
N, O	K

Right Subtree: The subtree whose root is the right child of some node is called the right subtree of that node.

Level of a Node: The level of a node is its distance from the root. The level of root is defined as zero. The level of all other nodes is one more than its parent node. The maximum number of nodes at any level N is 2^N.

Depth or Height of a tree: The depth or height of a tree is defined as the maximum number of nodes in a branch of a tree. This is more than the maximum level of the tree, i.e., the depth of root is one. The maximum number of nodes in a binary tree of depth d is 2^d-1, where d ≥1.

External Nodes: The nodes which have no children are called external nodes or terminal nodes.

Internal Nodes: The nodes which have one or more than one children are called internal nodes or non-terminal nodes.

Binary Expression Trees:

An algebraic expression can be conveniently expressed by its expression tree. An expression having binary operators can be decomposed into

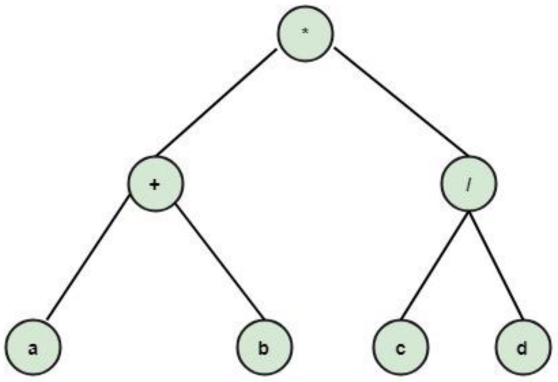
<left operand or expression> (operator) <right operand or expression>

Depending upon precedence of evaluation.

The expression tree is a binary tree whose root contains the operator and whose left subtree contains the left expression, and right subtree contains the right expression.

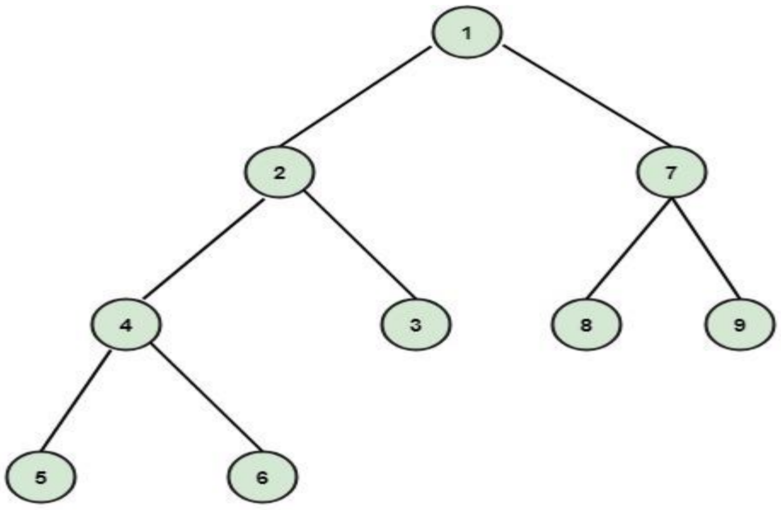
Example: Construct the binary expression tree for the expression (a+b)*(d/c)

Solution: The binary expression tree for the expression (a+b)*(d/c) is shown in fig:

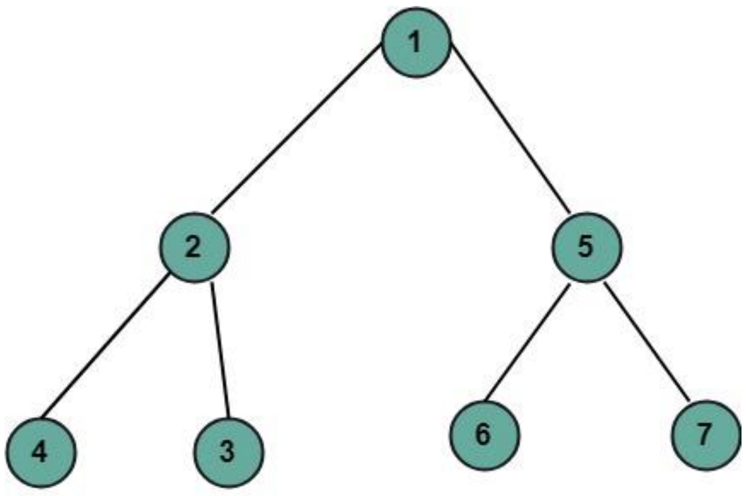


Complete Binary Tree: Complete binary tree is a binary tree if it is all levels, except possibly the last, have the maximum number of possible nodes as for left as possible. The depth of the complete binary tree having n nodes is $\log_2 n+1$.

Example: The tree shown in fig is a complete binary tree.



Full Binary Tree: Full binary tree is a binary tree in which all the leaves are on the same level and every non-leaf node has two children.



Full Binary Tree

Traversing Binary Trees

Traversing means to visit all the nodes of the tree. There are three standard methods to traverse the binary trees. These are as follows:

1. Preorder Traversal
2. Postorder Traversal
3. Inorder Traversal

1. Preorder Traversal: The preorder traversal of a binary tree is a recursive process. The preorder traversal of a tree is

- Visit the root of the tree.
- Traverse the left subtree in preorder.
- Traverse the right subtree in preorder.

2. Postorder Traversal: The postorder traversal of a binary tree is a recursive process. The postorder traversal of a tree is

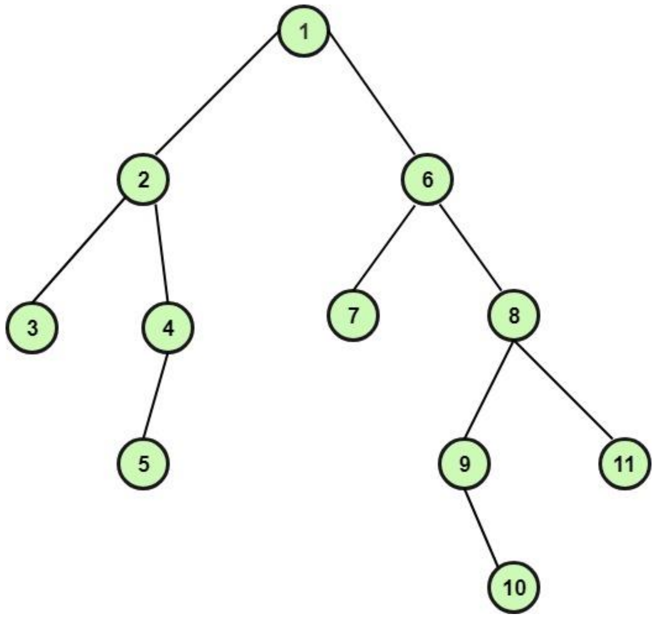
- Traverse the left subtree in postorder.
- Traverse the right subtree in postorder.
- Visit the root of the tree.

3. Inorder Traversal: The inorder traversal of a binary tree is a recursive process. The inorder traversal of a tree is

- Traverse in inorder the left subtree.
- Visit the root of the tree.

- Traverse in inorder the right subtree.

Example: Determine the preorder, postorder and inorder traversal of the binary tree as shown in fig:

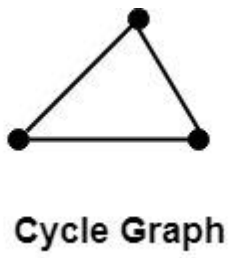


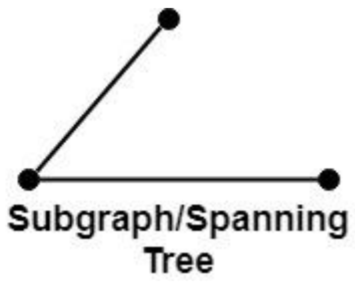
Solution: The preorder, postorder and inorder traversal of the tree is as follows:

Preorder	1	2	3	4	5	6	7	8	9	10	11
Postorder	3	5	4	2	7	10	9	11	8	6	1
Inorder	3	2	5	4	1	7	6	9	10	8	11

Spanning Tree

A subgraph T of a connected graph G is called spanning tree of G if T is a tree and T include all vertices of G.





Minimum Spanning Tree:

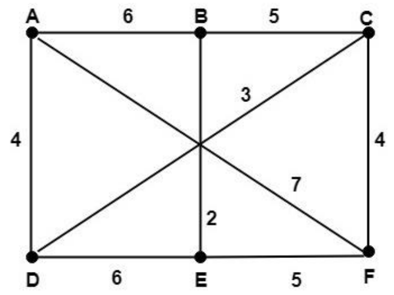
Suppose G is a connected weight graph i.e., each edge of G is assigned a non-negative number called the weight of edge, then any spanning tree T of G is assigned a total weight obtained by adding the weight of the edge in T.

A minimum spanning tree of G is a tree whose total weight is as small as possible.

Kruskal's Algorithm to find a minimum spanning tree: This algorithm finds the minimum spanning tree T of the given connected weighted graph G.

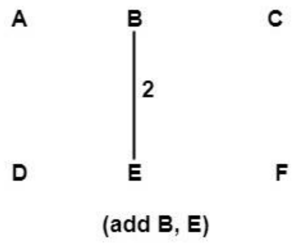
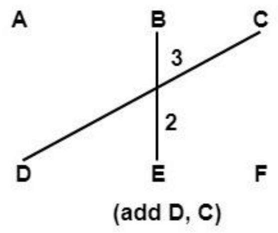
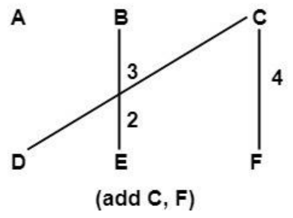
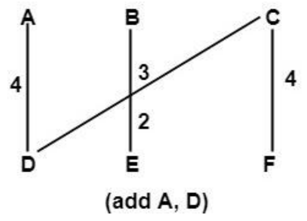
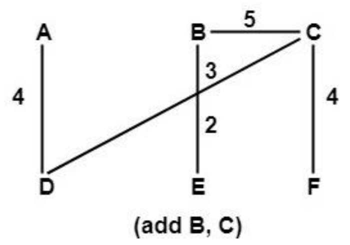
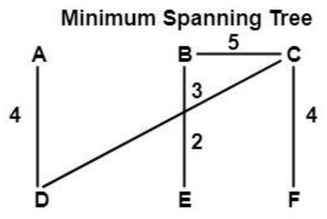
1. Input the given connected weighted graph G with n vertices whose minimum spanning tree T, we want to find.
2. Order all the edges of the graph G according to increasing weights.
3. Initialize T with all vertices but do include an edge.
4. Add each of the graphs G in T which does not form a cycle until n-1 edges are added.

Example1: Determine the minimum spanning tree of the weighted graph shown in fig:

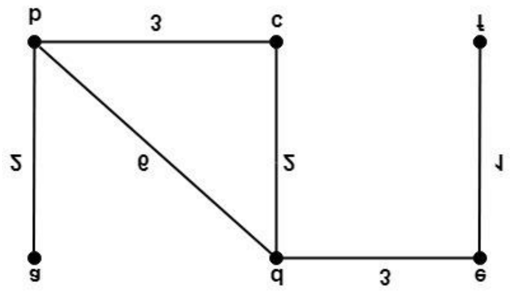


Solution: Using kruskal's algorithm arrange all the edges of the weighted graph in increasing order and initialize spanning tree T with all the six vertices of G. Now start adding the edges of G in T which do not form a cycle and having minimum weights until five edges are not added as there are six vertices.

Edges	**Weights**	**Added or Not**
(B, E)	2	Added
(C, D)	3	Added
(A, D)	4	Added
(C, F)	4	Added
(B, C)	5	Added
(E, F)	5	Not added
(A, B)	6	Not added
(D, E)	6	Not added
(A, F)	7	Not added

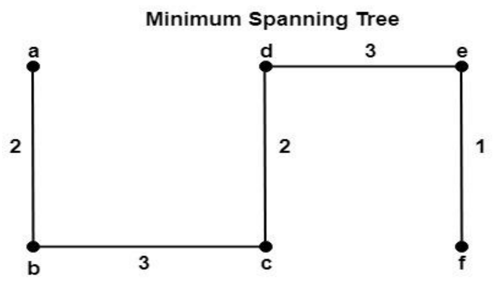
Step 1: 	Step 2: 
Step 3: 	Step 4:  A B C 4 3 4 2 D E F (add A, D)
Step 5:  A B 5 C 3 4 4 2 D E F (add B, C)	Step 6: **Step6:** Edge (A, B), (D, E) and (E, F) are discarded because they will form the cycle in a graph. So, the minimum spanning tree form in step 5 is output, and the total cost is 18.  Minimum Spanning Tree A B 5 C 3 4 4 2 D E F

Example2: Find all the spanning tree of graph G and find which is the minimal spanning tree of G shown in fig:



Solution: There are total three spanning trees of the graph G which are shown in fig:

Edges	**Weights**	**Added or Not**
(E, F)	1	Added
(A, B)	2	Added
(C, D)	2	Added
(B, C)	3	Added
(D, E)	3	Added
(B, D)	6	Not Added



The first one is the minimum spanning having the minimum weight = 11.