Trees

In graph theory, a tree is an undirected, connected and acyclic graph. ie. a connected graph having no cycle. is called a tree.

- A tree represents hierarchical structure in a graphical form.
- The elements of trees are called their nodes and the edges of the tree are called branches.
- A tree with n vertices has (n-1) edges.
- A leaf in a tree is a vertex of degree 1 or any vertex having no children is called a leaf.

Example -

Scanned with CamScanner

Properties of trees :—

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

proof — Since T is a connected graph, there must exist atleast one path between every pair of vertices.

Let there are two distinct paths between two vertices u and v of T, but union of two paths will contain a cycle and then T cannot be a tree.

Hence, the above statement is true. proved

2) If in a graph G there is one and only one path between every pair of vertices than graph G is a tree.

3) A tree with n vertices has (n-1) edges.

4) Any connected graph G with n vertices & (n-1) edges is a tree.

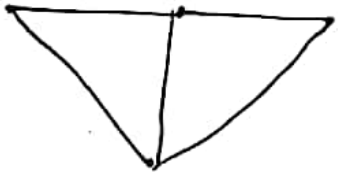
5) Any two vertices of graph T are connected by exactly one path.

Scanned with CamScanner

Spanning Trees :- Let G be a connected graph, then the sub-graph H of G is called Spanning tree of G if -

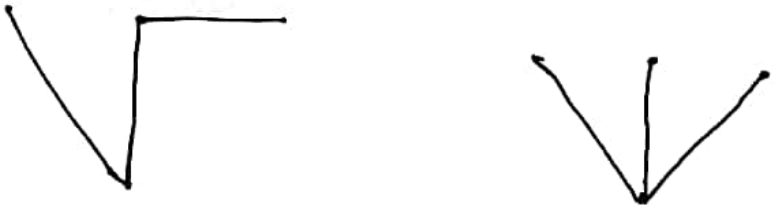
1) H is a tree.
2) H contains all the vertices of G.

Example - Consider graph G as-



Since the graph G has four vertices and hence each spanning tree must have 4-1 edges. Spanning trees of graph G as-

etc.



Scanned with CamScanner

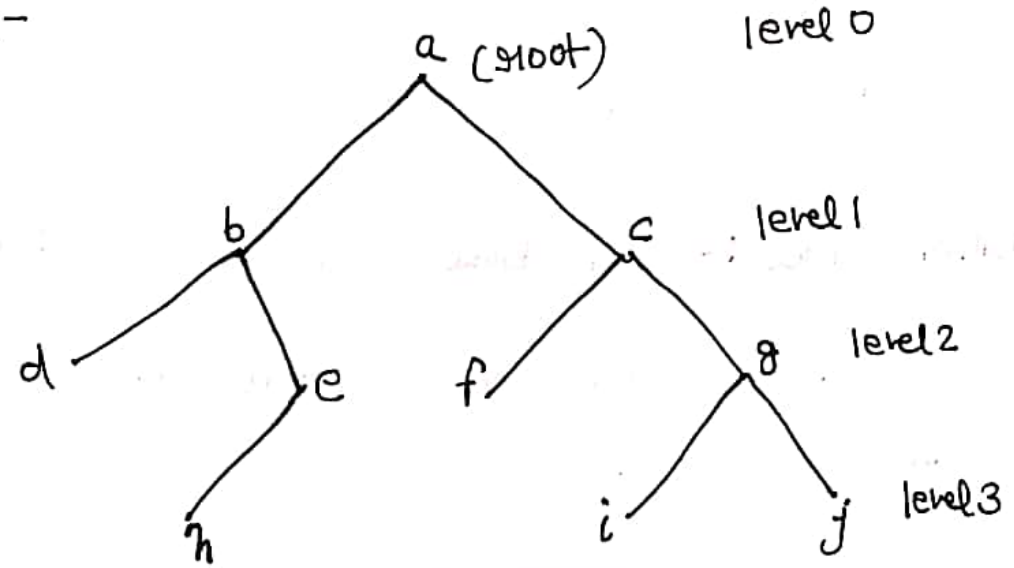
Note – Let $G = (V, E)$ be a connected graph such that $|V| = n$ and $|E| = m$

then the spanning tree of G must have n vertices & n-1 edges. So, we must remove $m - (n-1)$ edges from G to obtain a spanning tree.

Binary Tree :- A binary tree is a tree like structure that is rooted and in which each vertex has atmost two children and each child of a vertex is designed as its left or right child.

i.e. Binary tree is a rooted tree that contains – root, left child and right child

Example –



Nodes with no children is called leaves or external nodes.

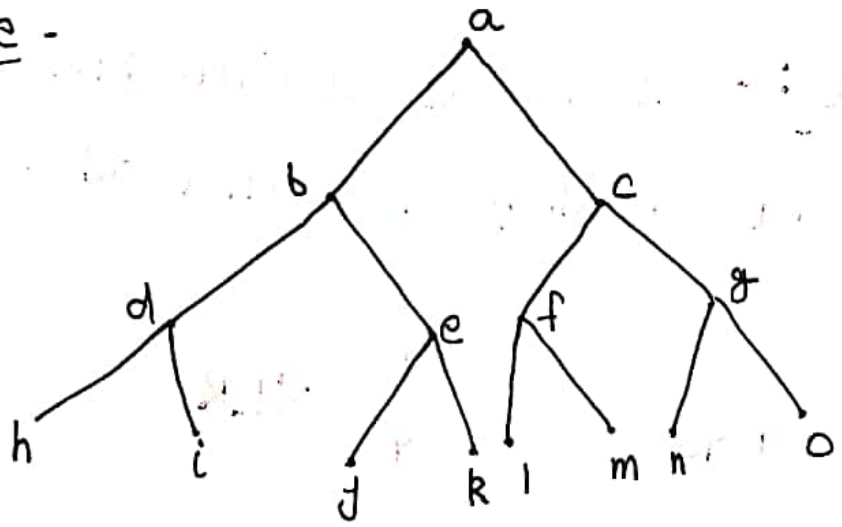
Scanned with CamScanner

Nodes which are not leaves are called internal nodes i.e. internal nodes have atleast one child.

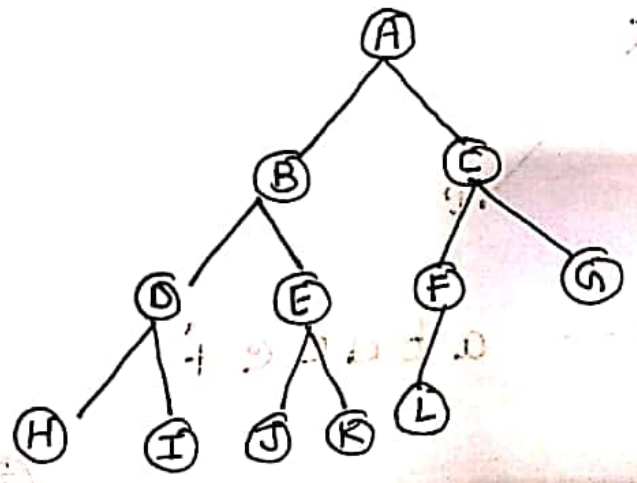
Nodes with the same parent are called siblings.

Full Binary Tree :- A full binary tree (sometimes proper binary tree or 2-tree) is a tree in which every node other than the leaves has two children.

Example -



Complete Binary Tree :- A complete binary tree is a binary tree in which every level, except possibly the last, is completly filled, and all nodes are as far left as possible.



Scanned with CamScanner

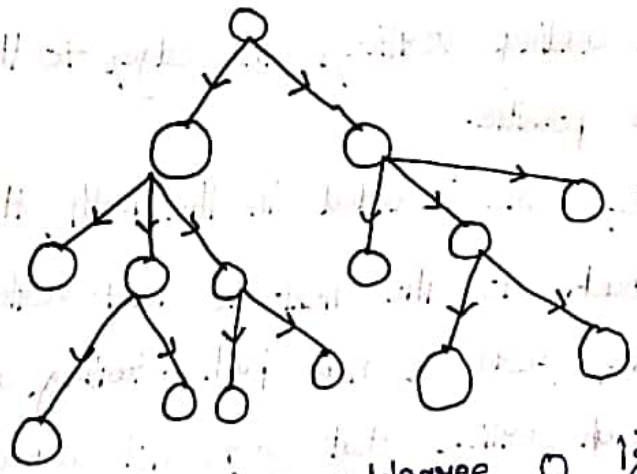
Rooted Trees :— A rooted tree is a tree in which a particular vertex is distinguished from the others and is called the root.

In graph theory rooted trees are typically drawn with their roots at the top. First, we place the root at the top and then other vertices are connected to the root vertices.

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 height of a rooted tree is the maximum level to any vertex of the tree.

Directed Tree :— A directed tree is an acyclic directed graph. All other node (excluding root) have indegree 1



The node which has outdegree 0 is called an external node or leaf. The nodes which have outdegree greater than or equal to one are called internal node.

Scanned with CamScanner

Extended binary Tree :- A binary tree T is said to be a 2-tree or extended binary tree if each node has either 0 or 2 children.

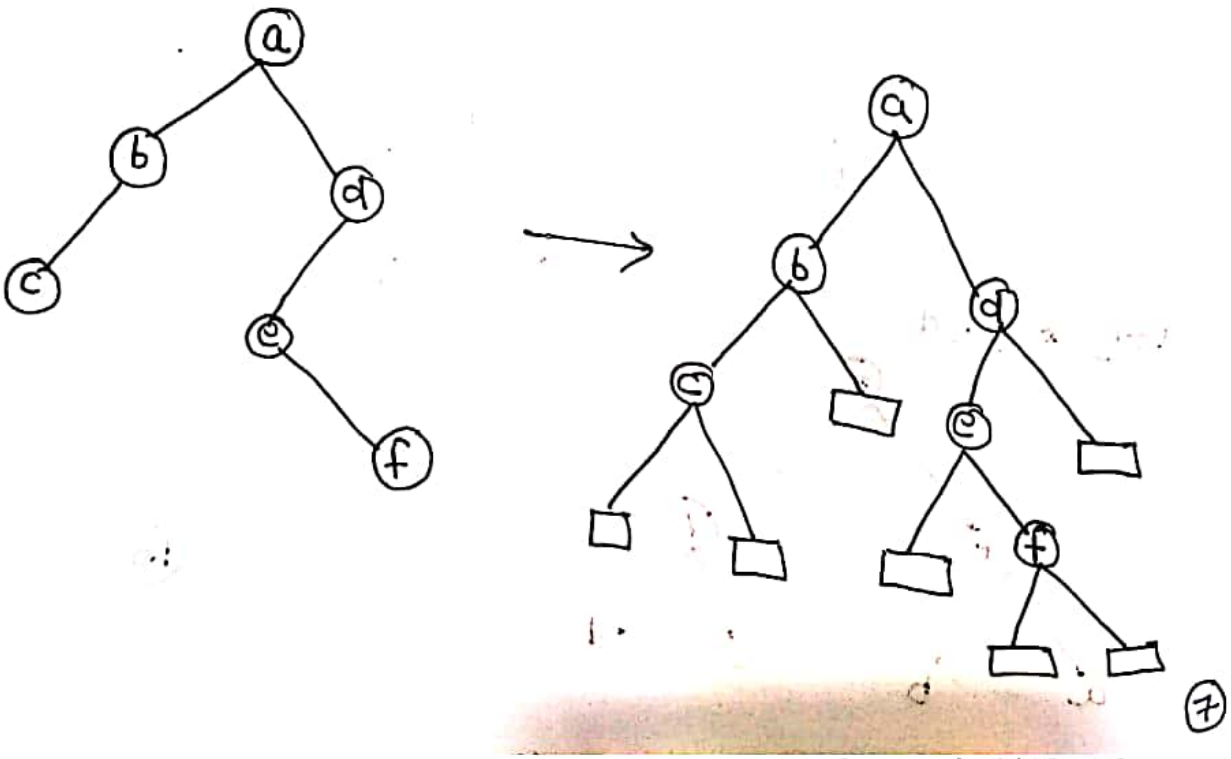
i.e. extended binary tree is a type of binary tree in which all the sub tree of the original trees are replaced with special ~~nodes~~ nodes called external nodes whereas other nodes are called internal nodes

○ — internal nodes

□ — external nodes

In extended binary tree, all external nodes are leaf nodes and the internal nodes are ~~leaf nodes~~. non leaf nodes

Example _



Scanned with CamScanner

Tree Traversal :- A traversal of a tree is a process of traverse (or visit) all nodes of tree in a systematic way so that each vertex is visited exactly once.

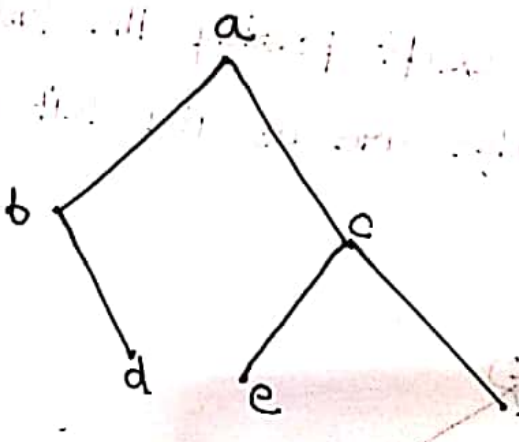
There are three ways which we use to traverse a tree —

1) **Preorder Traversal :-** To traverse a binary tree in preorder, the following operations are carried out -

(i) visit the root
(ii) Traverse the left subtree
iii) Traverse the right subtree.

} NLR

Example -



preorder traversal is — a b d c e f

Scanned with CamScanner

2) **Inorder Traversal** :- The inorder traversal of a binary tree is defined recursively as follows—

(i) Traverse the left subtree
(ii) visit the root
(iii) Traverse the right subtree.

} LNR

Example - Inorder traversal of the above tree is—

b d a e

3) **Postorder Traversal** :- The postorder traversal of a binary tree is as follows—

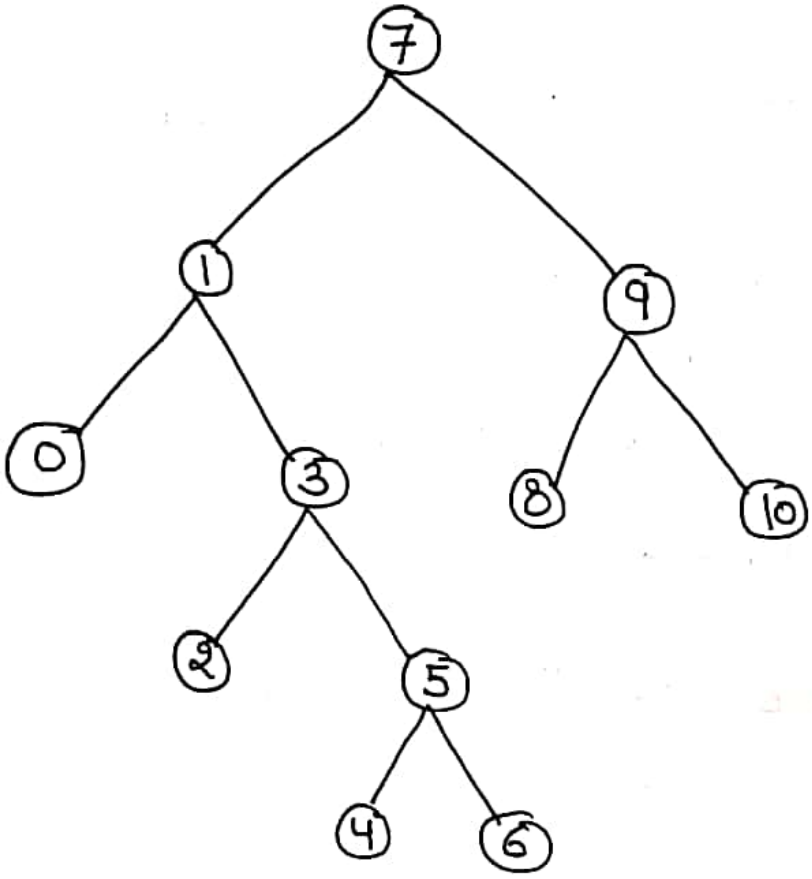
(i) Traverse the left subtree
(i) Traverse the right subtree
iii) Visit root.

Example - output of the previous tree by using postorder traversal is defined as—

d b e f c a

Scanned with CamScanner

Example — Let us consider the given binary tree-



The traversal of given tree in three different ways as—

Preorder — 7 1 0 3 2 5 4 6 9 8 10

Inorder — 0 1 2 3 4 5 6 7 8 9 10

Postorder — 0 2 4 6 5 3 1 8 10 9 7

Scanned with CamScanner

To Draw a unique binary tree when Inorder and preorder traversal of the tree is given —

1) The root of tree T is obtained by choosing the first vertex in its preorder.

2) The left child of the root vertex is obtained as follows—

Find the vertices in the left subtree of the binary tree, we use the inorder traversal ~~to~~ The left child of the root is obtained by selecting the first vertex in the preorder traversal of the left subtree Draw the left child.

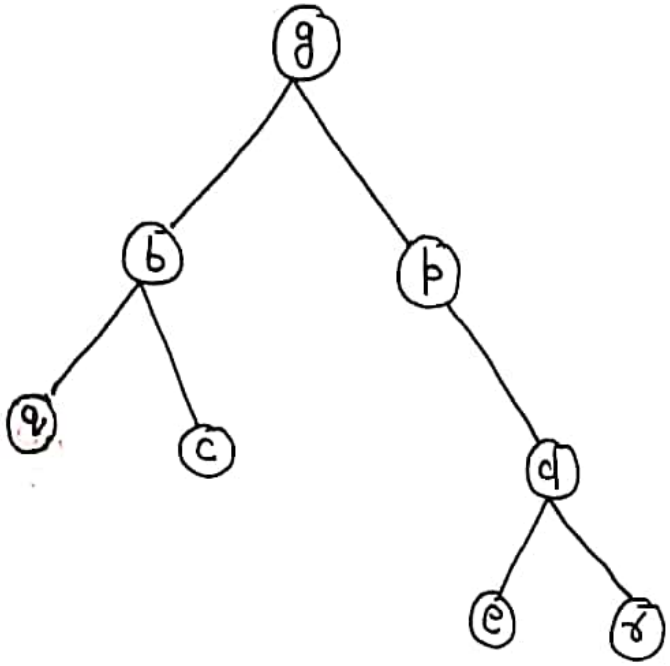
3) Use the inorder traversal to find the vertices in the right subtree of the binary tree. Then the right child is obtained by selecting the first vertex in the preorder traversal of the right subtree. Draw the right child.

4) The procedure is repeated recursively until every vertex is not visited in preorder.

Example - Given the preorder and inorder traversal of a binary tree, draw the unique tree —

preorder : g b q a c p d e r

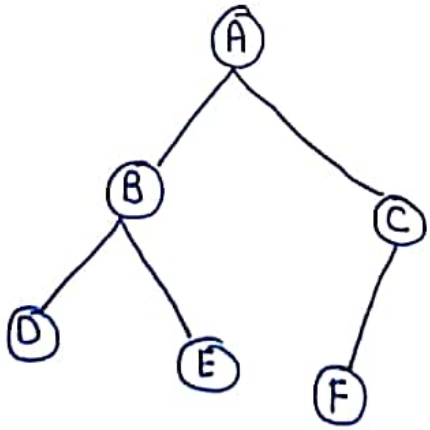
Inorder : q b c a g p e d r



Example - Draw the unique tree if -

preorder : A B D E C F

Inorder : D B E A F C



Ques Draw the unique tree if -

preorder : 6 1 5 11 3 4 8 7 2

Inorder : 5 1 3 11 6 8 2 4 7

Scanned with CamScanner

Ques Construct a binary tree whose traversal is given below -

preorder : a b d g e h i j c f

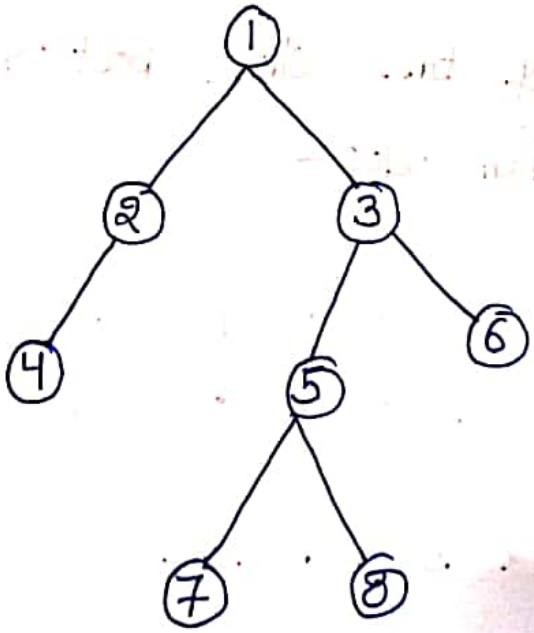
Inorder : d g b e i h j a c f

To draw a unique binary tree when Inorder and postorder traversal of the tree is given -

Example - Inorder : 4 2 1 7 5 8 3 6

Postorder : 4 2 7 8 5 6 3 1

Sol Binary tree is as -



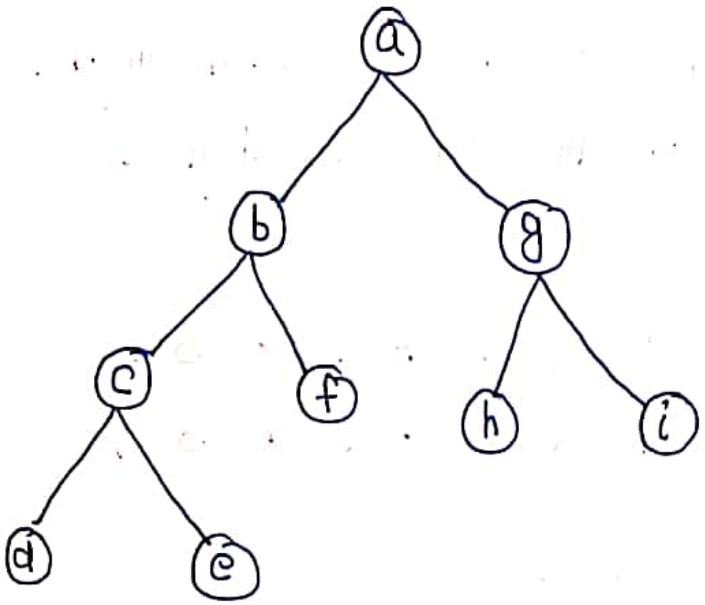
Scanned with CamScanner

Example - Construct a binary tree if —

Inorder : d c e b f a h g i

postorder : d e c f b h i g a

Sol



Ques Construct a binary tree whose post order and inorder is given below —

1) Inorder : 4 2 5 1 6 7 3 8

postorder : 4 5 2 6 7 8 3 1

2) Inorder : 4 8 2 5 1 6 3 7

postorder : 8 4 5 2 6 7 3 1

Scanned with CamScanner

Representation of Algebric structure by binary Trees :-

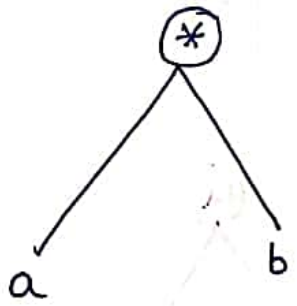
Binary trees are used to represent algebric expression, the vertices of the tree are labled with the numbers, variables or operations that make up the expression.

The leaves of the tree can be labled with numbers or variables. operations such as addition, subtraction, multiplication, division or exponentiation can only be assigned to internal vertices.

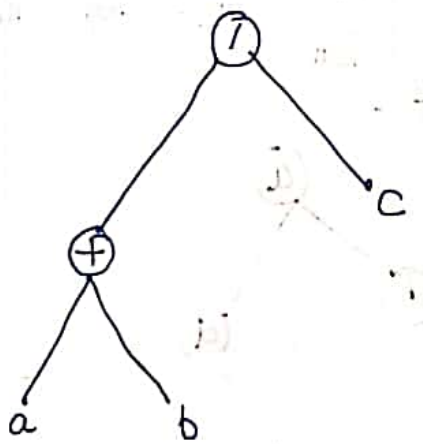
The operation at each vertex operates on its left and right subtree. from left to right.

Example - Use a binary tree to represent the expression

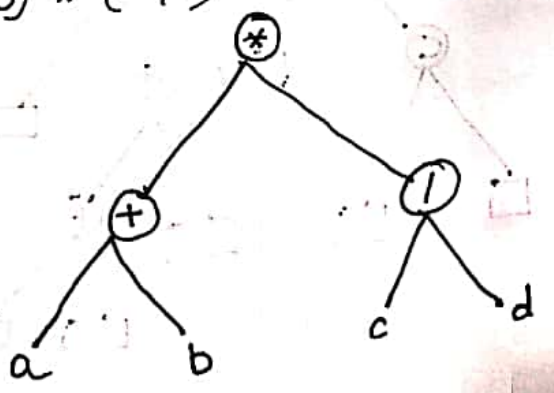
(i) $a * b$



(ii) $(a+b)/c$



(iii) $(a+b) * (c/d)$



Scanned with CamScanner

Depth – First Search :– In this algorithm a rooted tree will be constructed, and the underlying undirected graph of this rooted forms the spanning tree.

A spanning tree can be built by doing a depth first search of the graph.

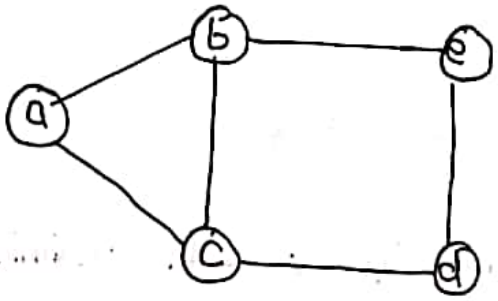
Procedure –

- Start with arbitrarily Chosen vertex of the graph as the root.
- Form a path from the root by successively adding vertices and edges where
  - → Each new edge is incident with the last vertex in the path.
  - → Vertices are not already in the path.
  - → Continue adding vertices and edges to this path as long as possible.
- If all the vertices are included in the path, then "Done".
- 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. If this cannot be done, move back another vertex and try again.

Scanned with CamScanner

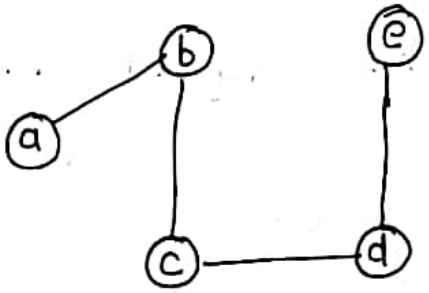
→ Repeat this procedure until no more edges can be added.

Example :- Find a spanning tree of the graph using DFS algorithm.

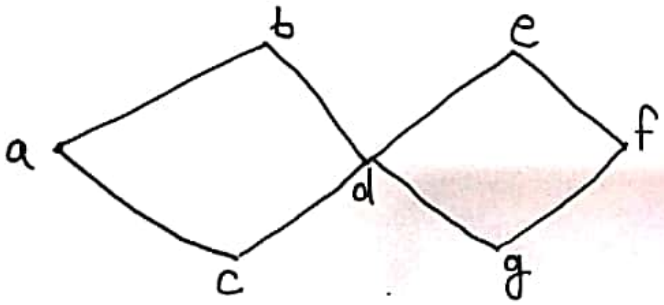


Sol DFS tree starting from a as the root

Spanning tree by DFS →



Example - Find a spanning tree of the graph



Scanned with CamScanner

Sol DFS, tree starting from vertex a as root.

Now Form path starting with root a by successively adding edges incident with vertices not already in the path as long as possible.

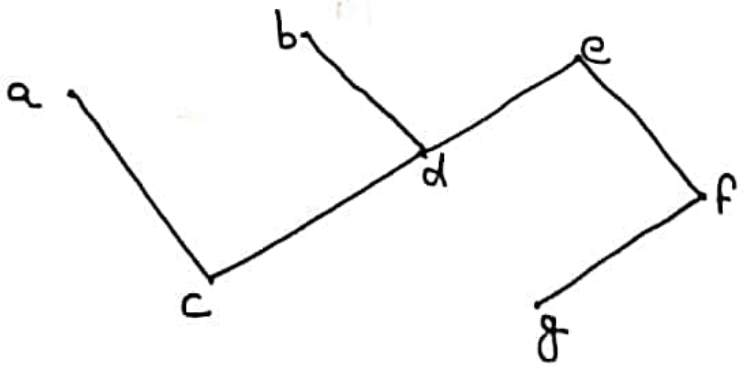
The path is a-c-d-e-f-g

Now, backtrack to f, there is no path beginning at f containing vertices not already visited.

Similarly after backtrack at e, there is also no path.

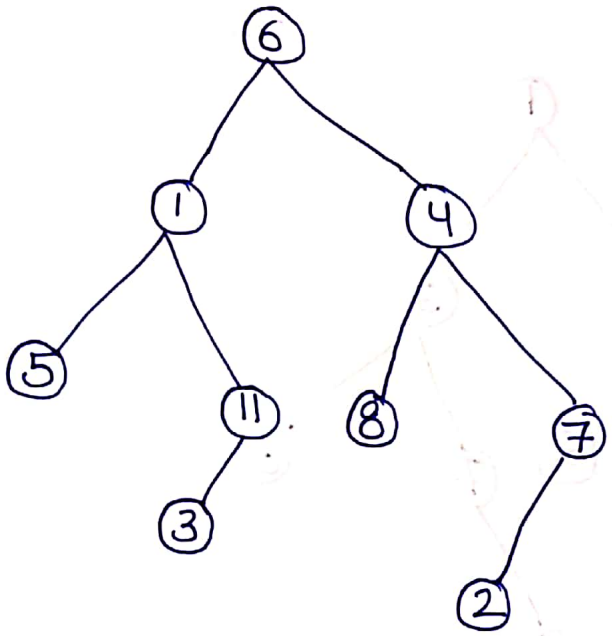
So, move backtrack at d and form the path d-b.

So, the required spanning tree is

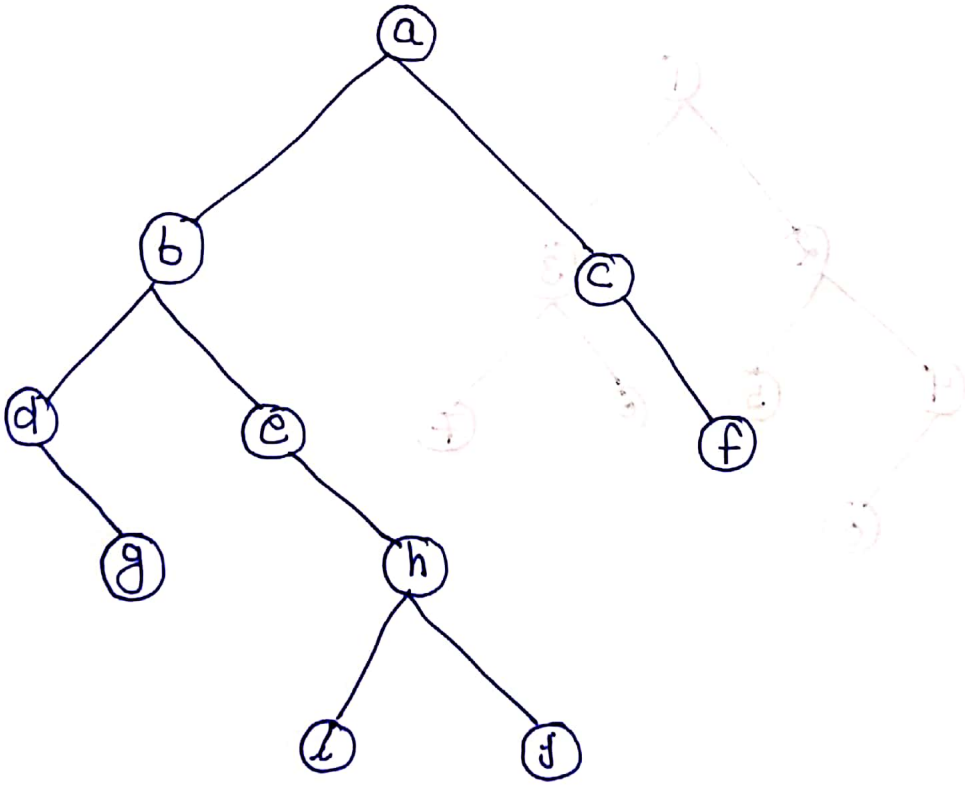


Scanned with CamScanner

Solution-1

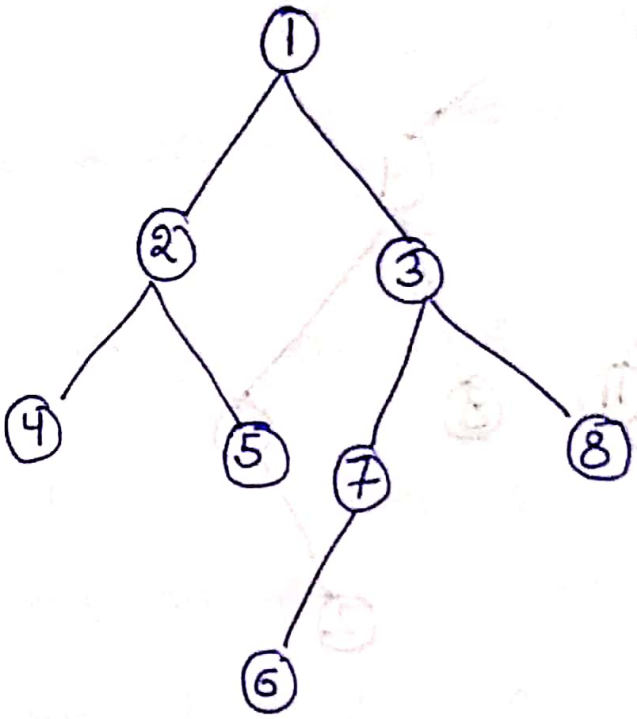


Solution-2

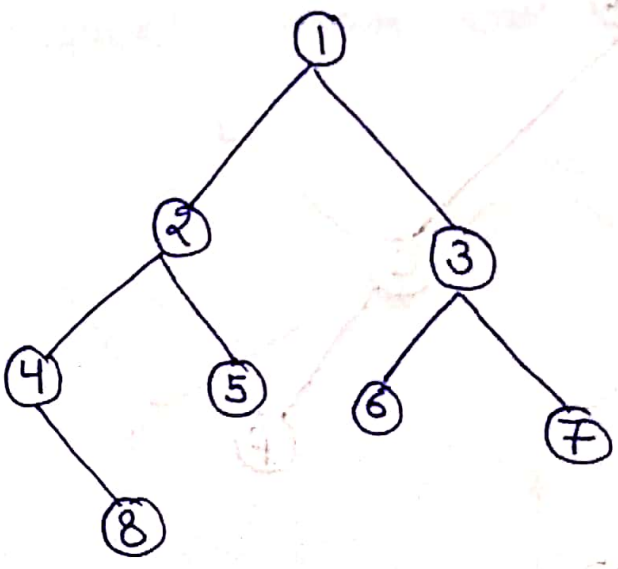


Solution - 3

(i)



(ii)



20