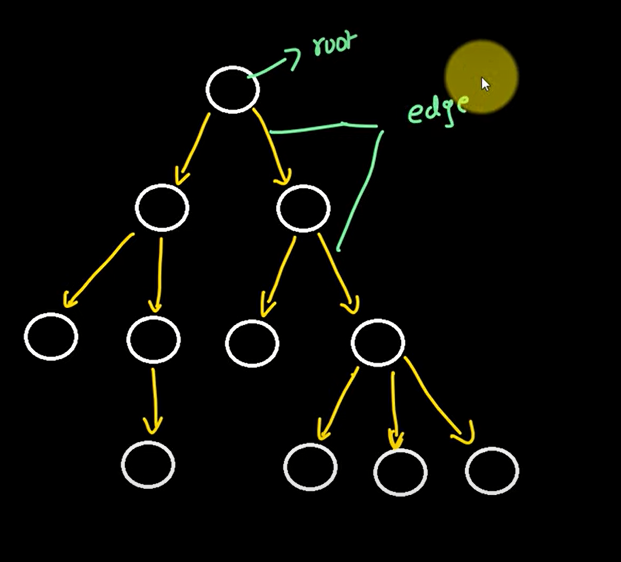
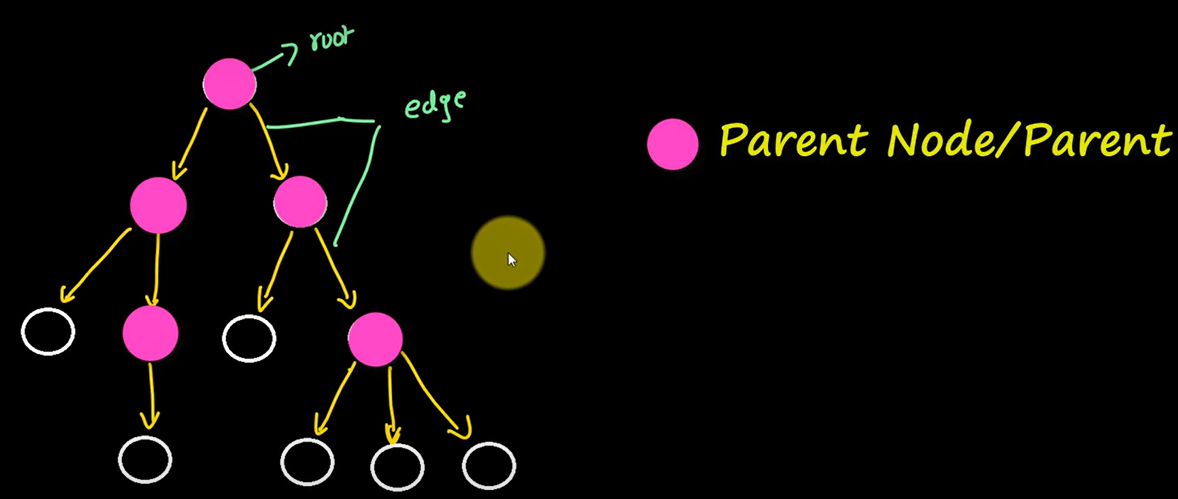
Tree-Tree Data Structure is a collection of entities called Nodes. Nodes are connected with edges. Each Node consists of data. A node may or may not have children

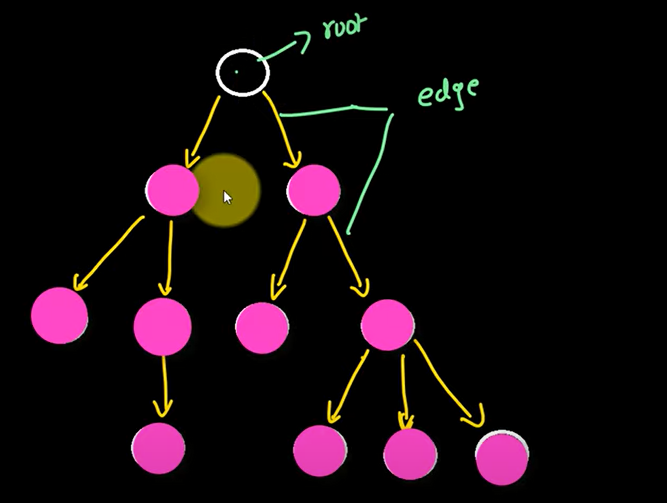
Node- The individual element of a tree are called nodes. A node contains data and a link to other node

Root Node-The top most node of tree is called Node. Root Node is the origin of tree. Every tree have only one root node.

Edge- It is a connection between two nodes

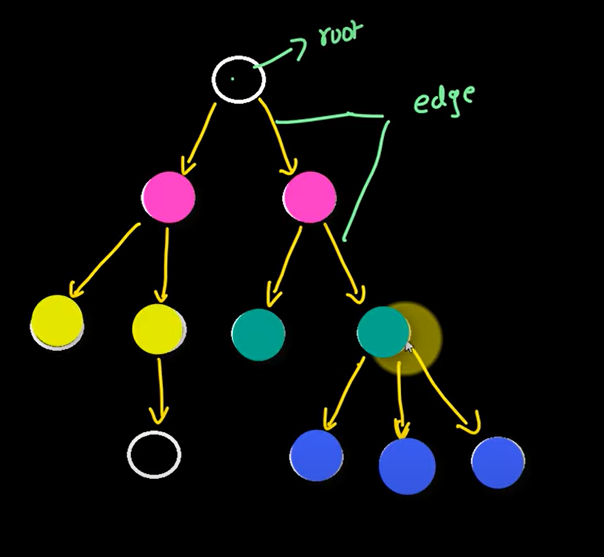
Parent Node- A node which have a branch from it two any other node is called parent node. A node which have children

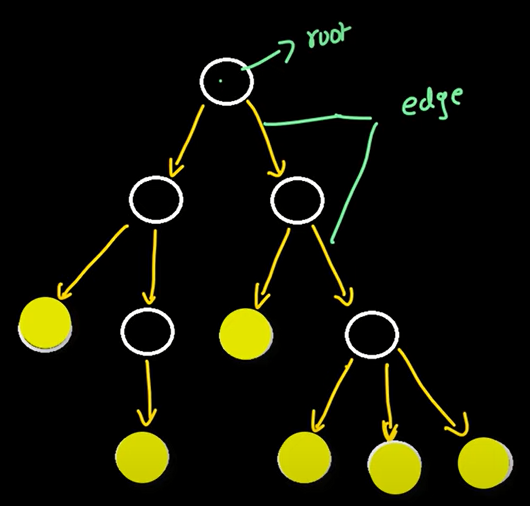
Child Node-A node which have a link from its parent is called child node. In a tree every node is a child node except the root node



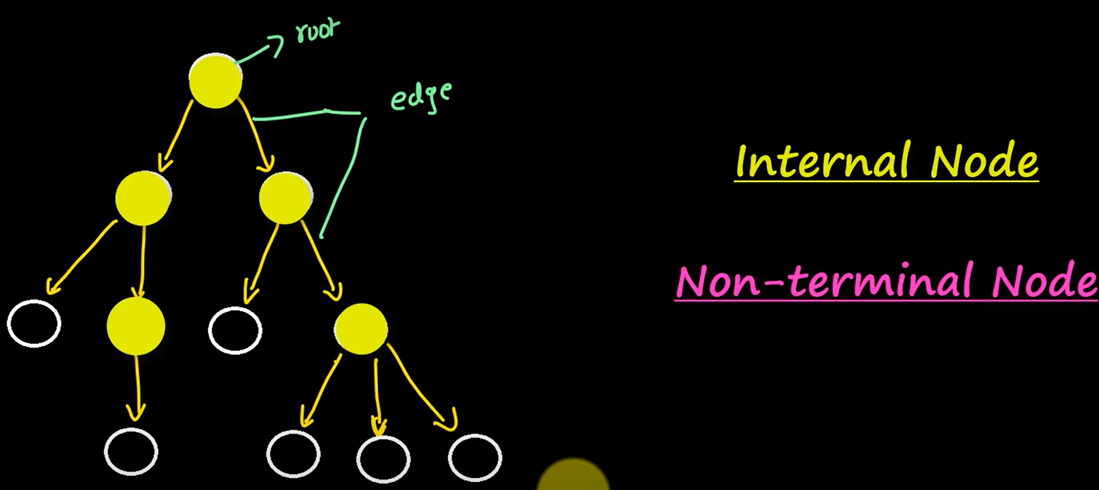


Siblings- The Nodes which have the same parent are called sibling

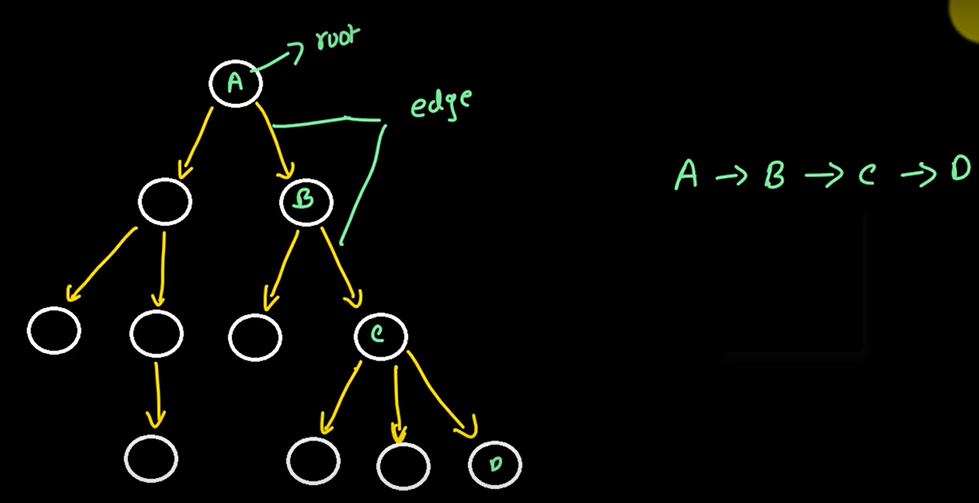


Leaf Nodes/External Nodes/Terminal Nodes-A node which doesn’t have a child

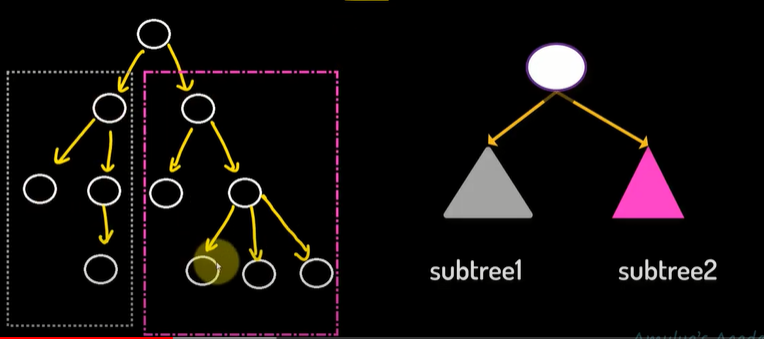
Internal Nodes/Non Terminal Nodes/Non Leaf Nodes -The node which have at least one child is called internal nodes



Path- The sequence of edges and nodes between two nodes is called a path between those nodes



Characteristics of Tree

1. Every Non-Empty tree contains a root node
2. In a tree if we have n nodes then we have (n-1) edges in that tree
   1. 11 nodes means 10 edges
3. Every child node have only one parent but one parent can have multiple children
4. A tree contains tree within them means tree is a recursive data structure
   1. 

Properties of tree

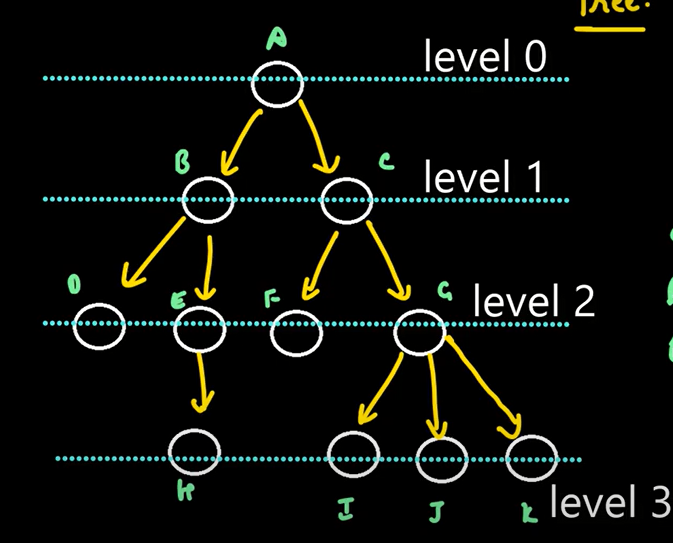
**Degree**

Node-Total Number of children of that node

Tree-Highest degree of a node among all the nodes in the tree

**Level**

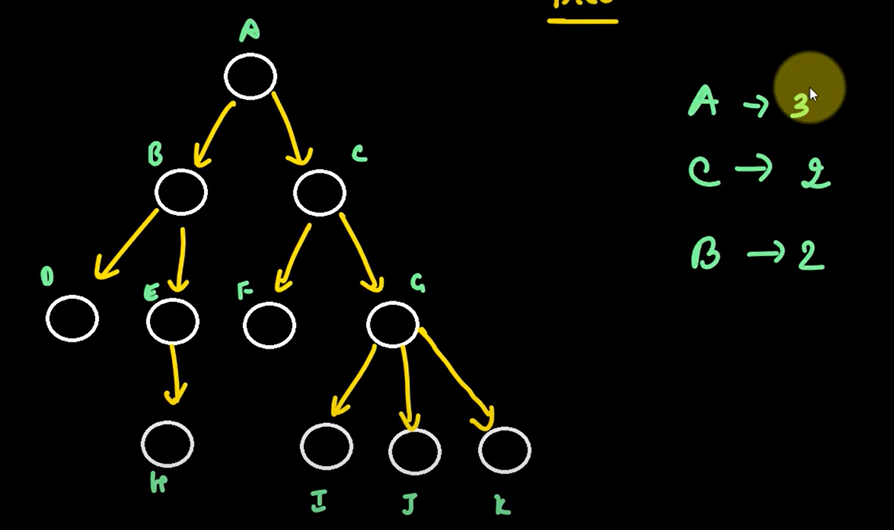
Each step from top to bottom of a tree is called level. level count starts from zero



**Height**

Node- Total number of edges lies on the longest path from any leaf node to particular node

Tree Height of root Node

Height of leaf node is always zero

**Depth**

Node- Total number of edges from root node to a particular node

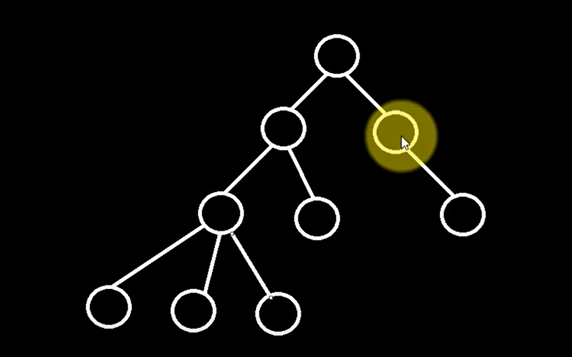
Depth of root node is always zero

Tree-Total number of edges from root node to a leaf node in the longest path

Types of Trees

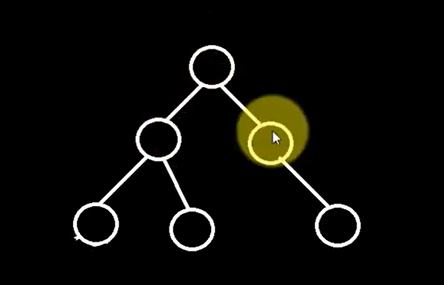
**General Tree**

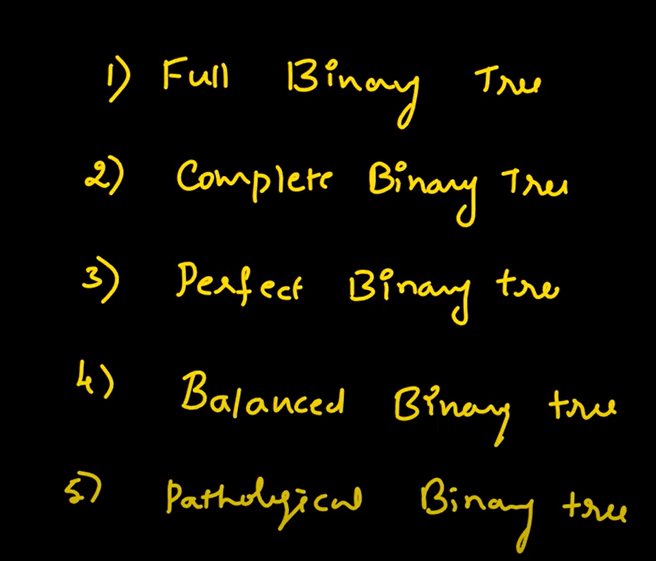
It is a type of tree data structure where each node can have any number of child nodes. No restriction on number of children a node can have.

****

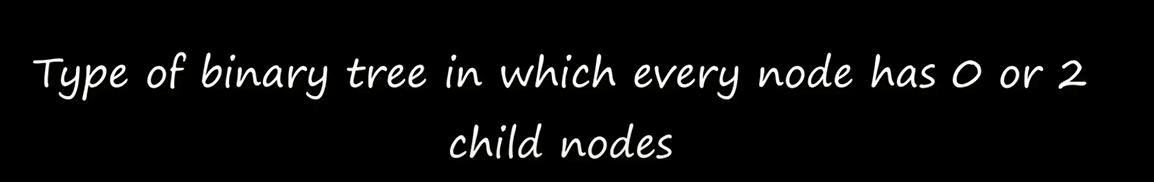
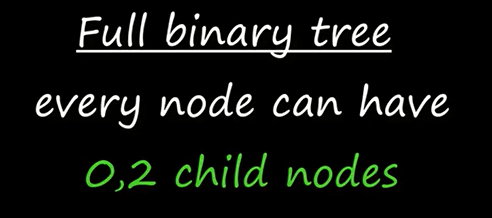
**Binary Tree**

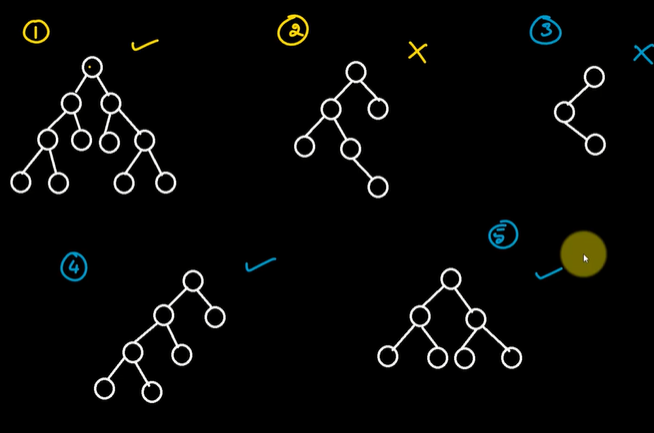
It is a type of binary tree which have a restriction on number of child nodes. A binary tree can have 0,1 or 2 child nodes.



Types of Binary Tree

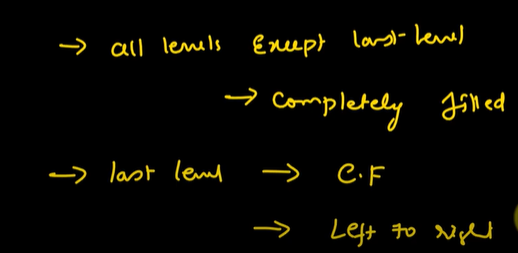
Full Binary/Strict/Strictly Binary Tree

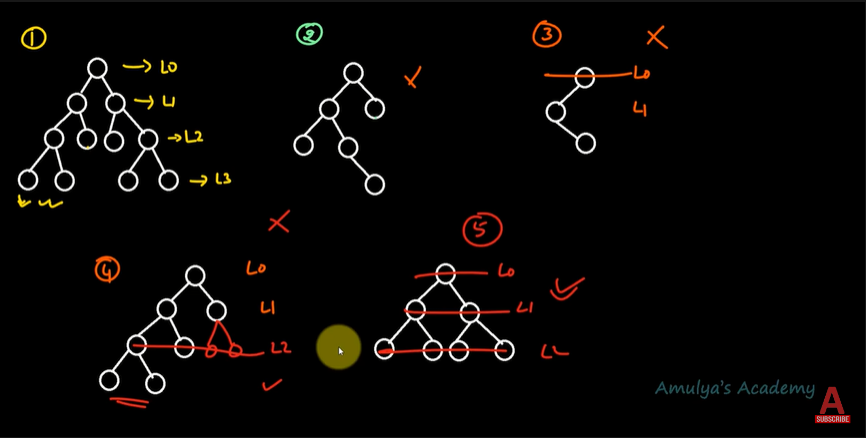




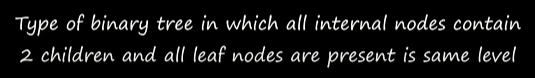
Complete Binary Tree

It is a type of binary tree in which all the levels except last level is completely filled with nodes and in the last level it can be filled with nodes completely or nodes need to be filled from left to right

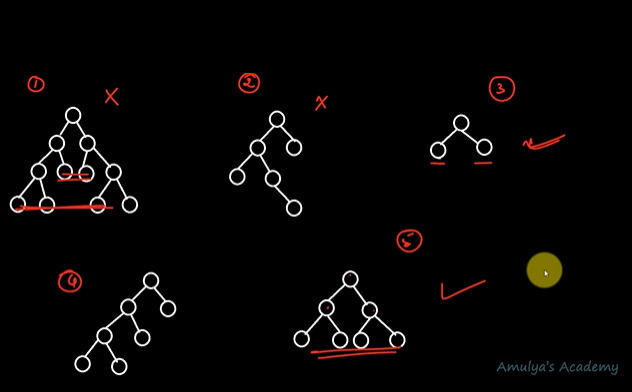




Perfect Binary Tree



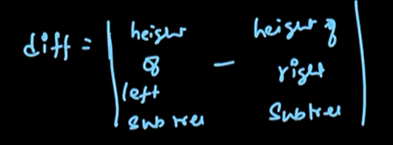


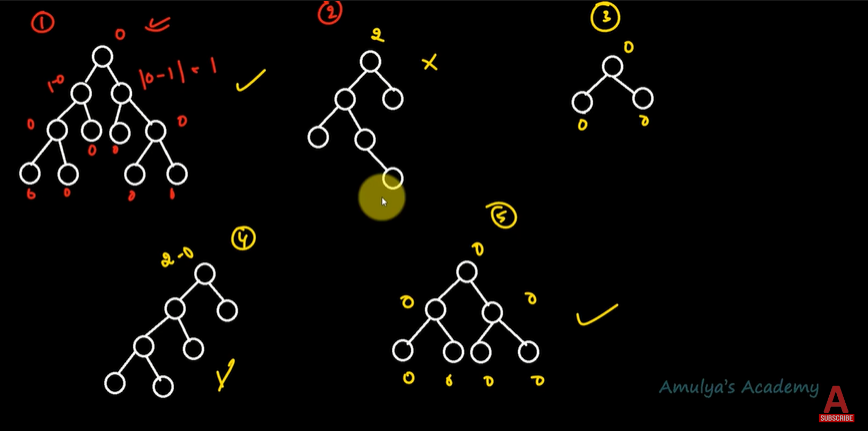


Balanced Binary Tree

A balanced binary tree, also referred to as a height-balanced binary tree, is defined as a binary tree in which the height of the left and right subtree of any node differ by not more than 1.

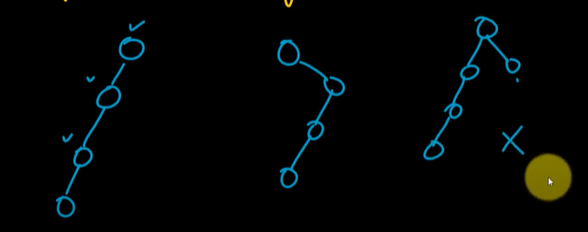
|height of left subtree - height of right subtree|<=1



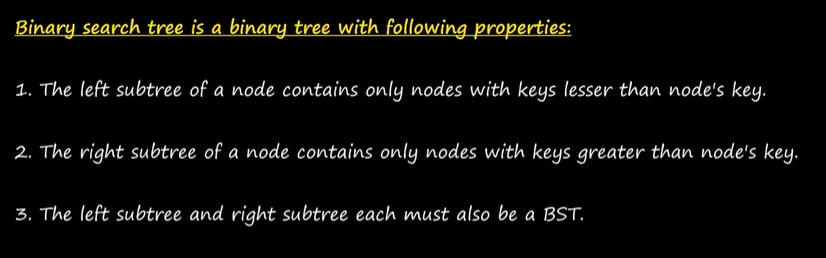


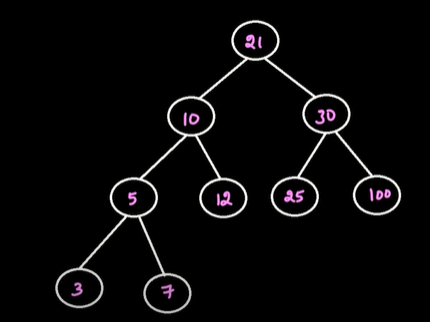
Pathological Binary Tree/Degenerate Binary Tree

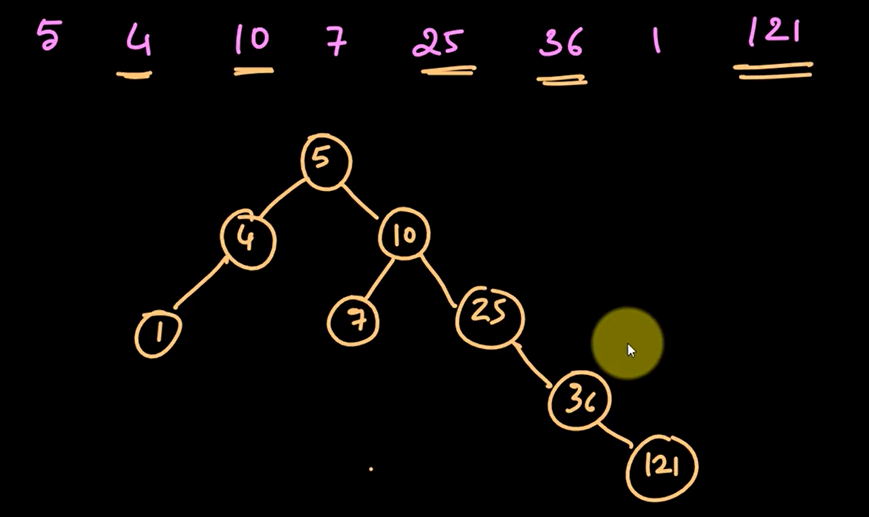
It is a type of binary tree where every parent node have only one child node



Binary Search Tree (BST)







Binary Search Tree with Duplicate Values

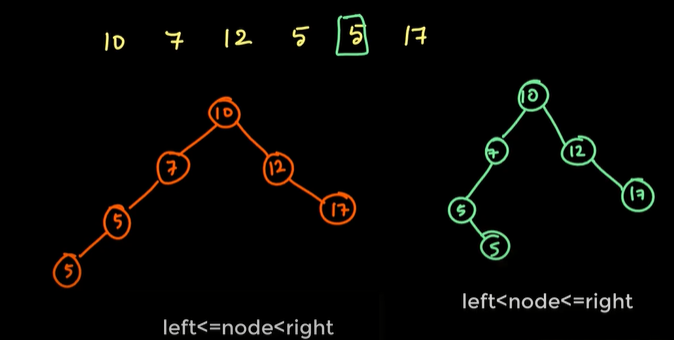
Approaches to deal with duplicate values

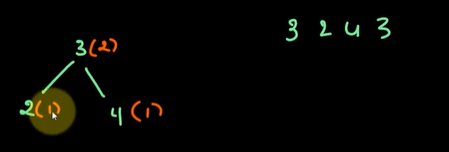
1 Do not allow duplicate values in BST

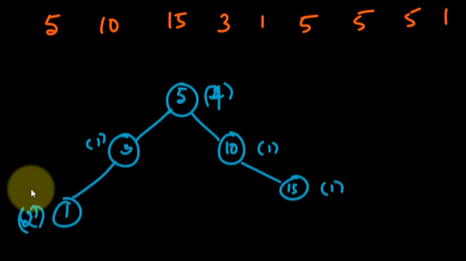
2 If duplicate value occurs put it on left side

3 If duplicate value occurs put it on right side

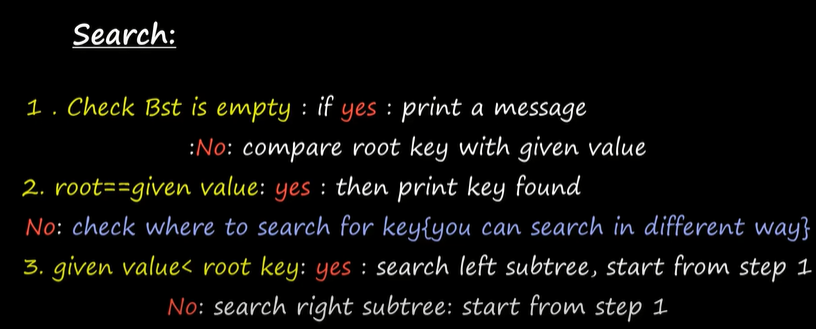
4 put a counter with each node which counts the number of occurrences of the value inside the node

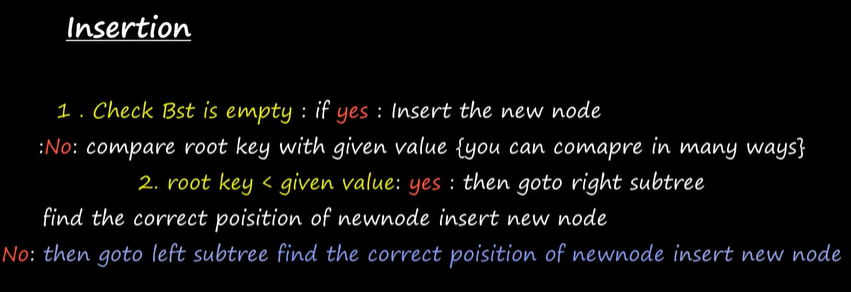


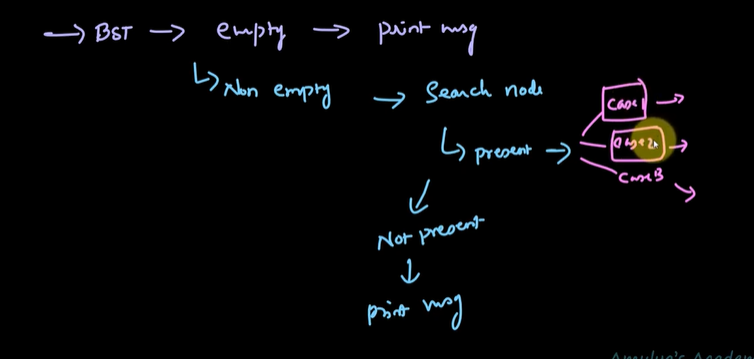


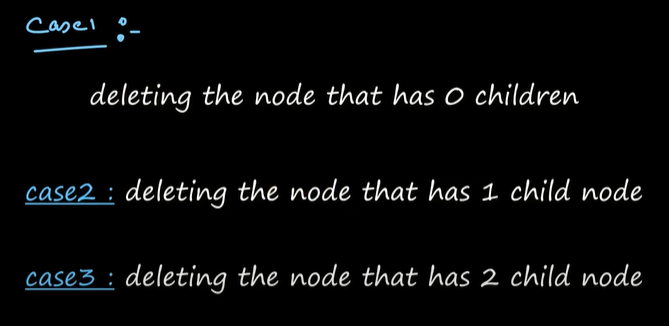


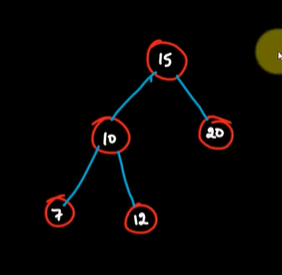
Searching in BST

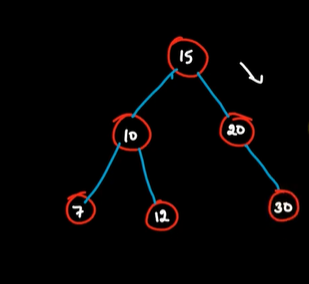


Insertion in BST

Deletion in BST

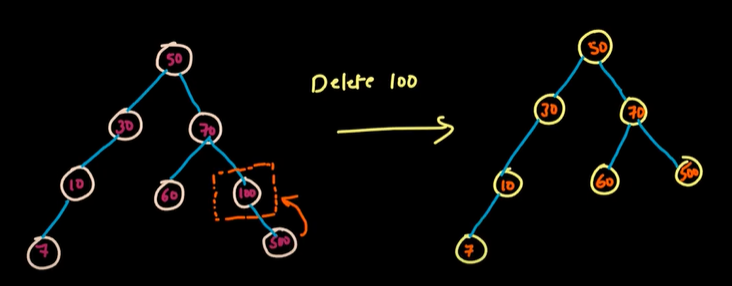


Case 1

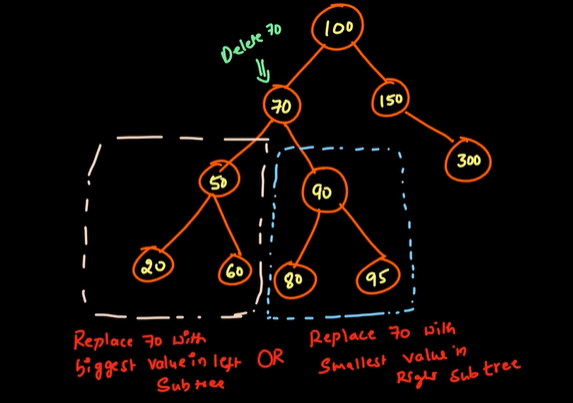


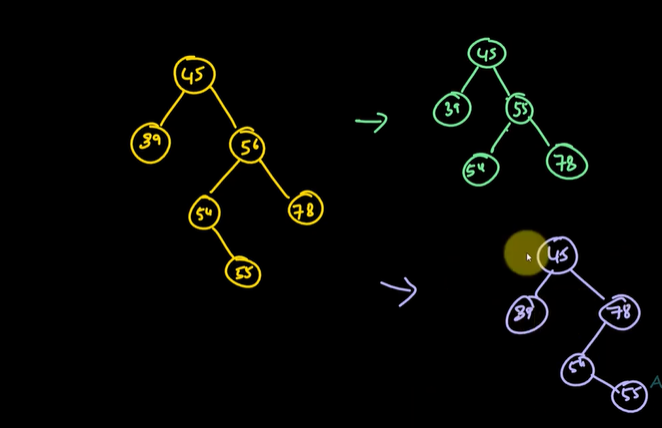
Case 2





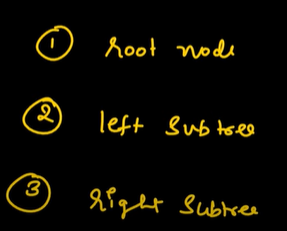
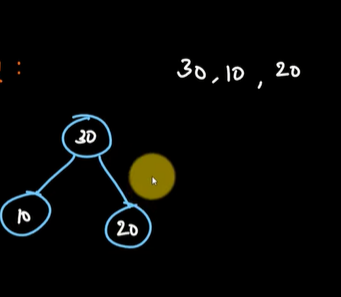
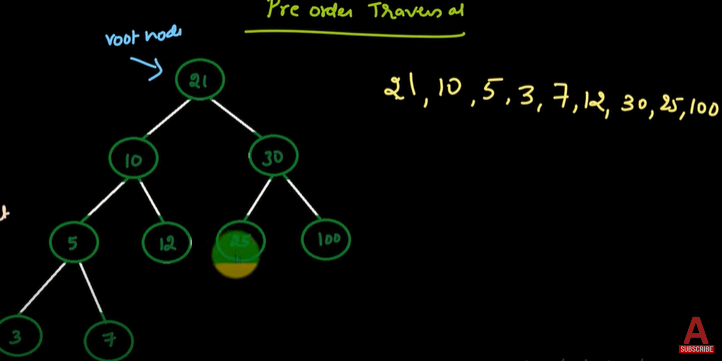
Case 3

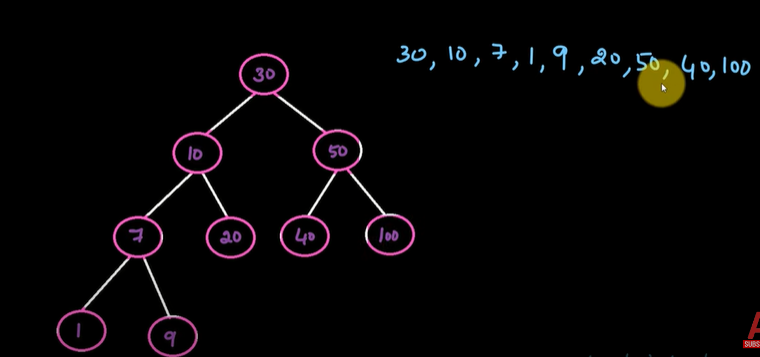


To delete is 56

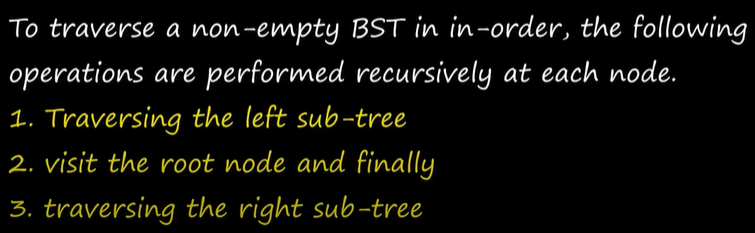
Traversal of a Tree

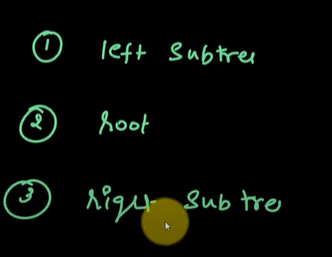
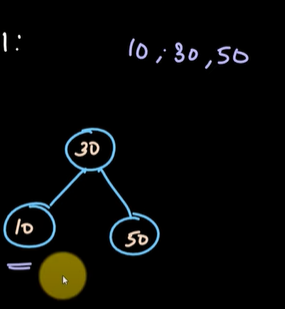
Pre-Order Traversal

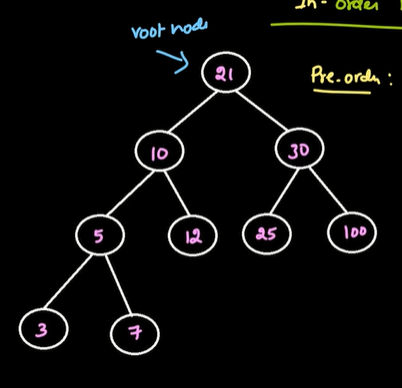


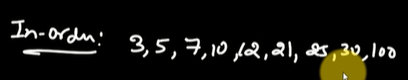


In Order Traversal

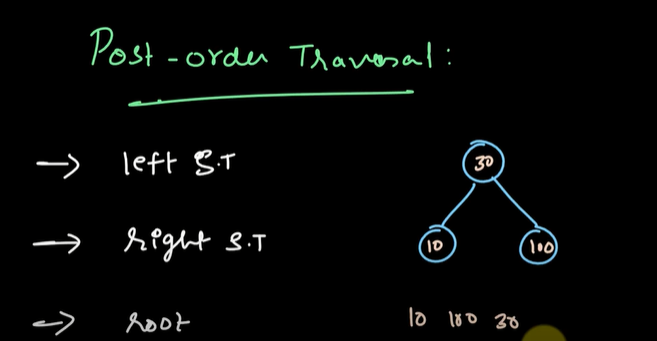


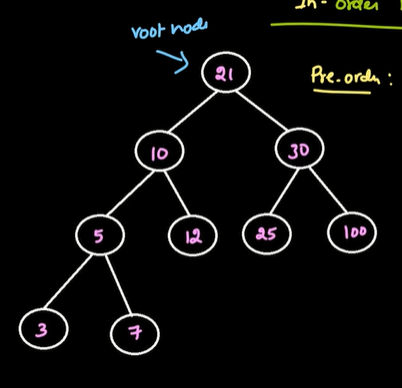


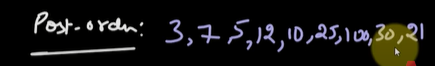




Post Order







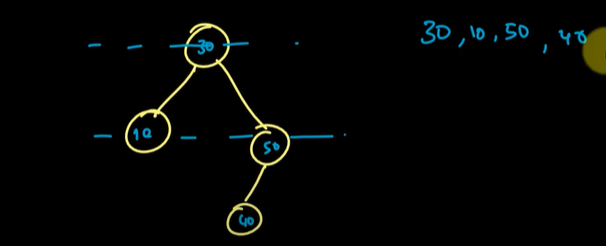
Level Order Traversal

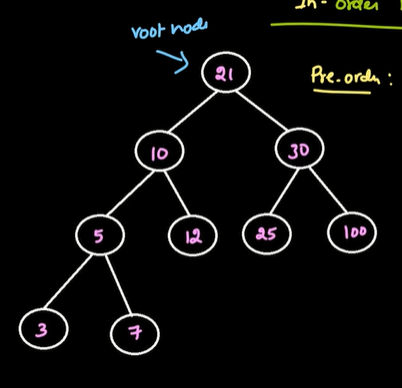
Traverse the nodes of a particular level at a time

Level 0

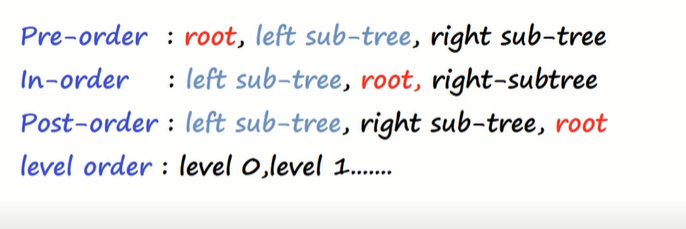
Level 1

Level 2 and so on.









Minimum value in a BST is on the left side recursively traverse the left nodes and if the left node is none

The current value is the required minimum value

Maximum value in a BST is on the right side recursively traverse the right nodes and if the right node is none

The current value is the required maximum value

