08/09/25

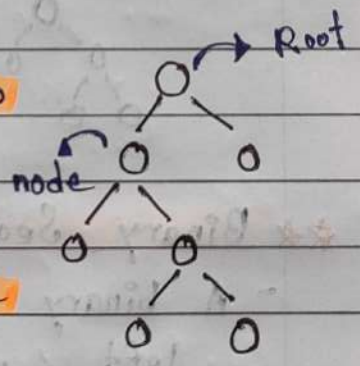
Binary Tree

Tree is a non-linear, hierarchial data structure.

→ Non-linear means elements are not arranged sequentially.
- A single element can connect to multiple elements.

→ Hierachical means elements are arranged in levels.

→ Each node can have zero or more children



→ Each node has only one parent (except root)

→ Tree can not have cycles.

→ Exactly one path between two nodes.

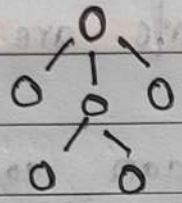
Examples:
1. File system
2. HTML Dom
3. Databases
4. Hierarchical data

08/09/25

Type of Trees.

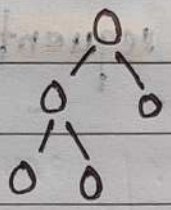
General Tree -

- A node can have any number of children



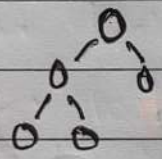
Complete Binary Tree

- All levels are completly filled except possibly the last level.



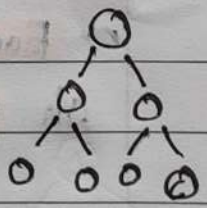
** Binary Tree -

- node can have at most 2 children (left and right)



Perfect Binary Tree

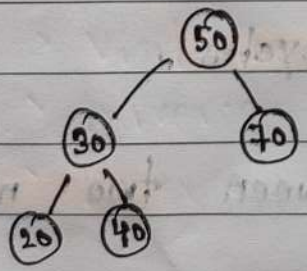
- All internal nodes exactly have 2 children
- Leaf nodes are at same level



** Binary Search Tree (BST)

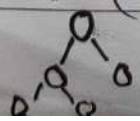
- A binary tree with ordering

left child < parent < right child



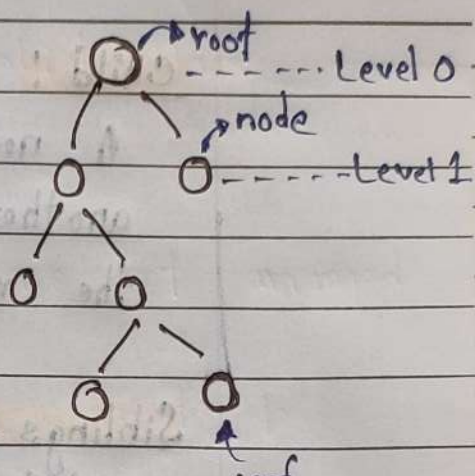
Full Binary Tree :

- Every node has 0 or 2 children (not one child)



08/09/25

Root:
- top most node of a tree
- It has no parent



Node:
- basic element of the tree containing data and links to children.

Parent:
- A node that has one or more child nodes.

Level
- The depth of a node in the tree (distance from root)

Height -
- The longest path from a node to a leaf
- Height of the tree = height of root node.

Leaf:
- A node with no children (end node)

Subtree -
- A smaller tree inside a tree.

08/01/25

Child :

- A node that is directly connected to another node when moving away from the root.

Siblings :

- Nodes that share the same parent.

Tree Traversals (Binary Tree)

Types :

1. Preorder traversal
2. Postorder traversal
3. Inorder traversal
4. Level order.

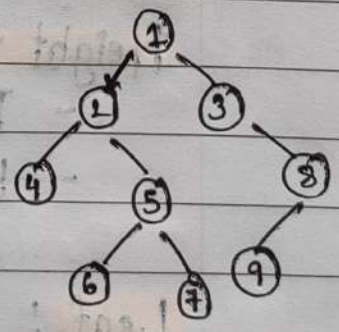
1. Preorder Traversal :

Root → Left → Right

→ Visit the root node

→ Traverse left subtree

→ Traverse right subtree

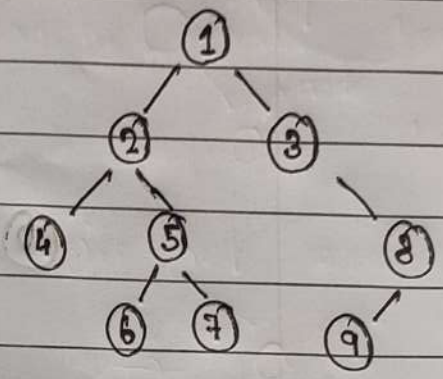


O/P : [1, 2, 4, 5, 6, 7, 3, 8, 9]

08/09/25

2. **Inorder Traversal :**

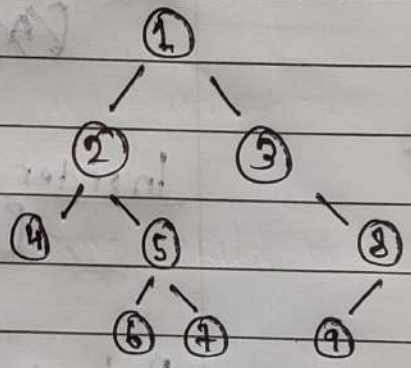
left → root → right



O/P : [4, 2, 6, 5, 7, 1, 3, 9, 8]

3. **Postorder Traversal :**

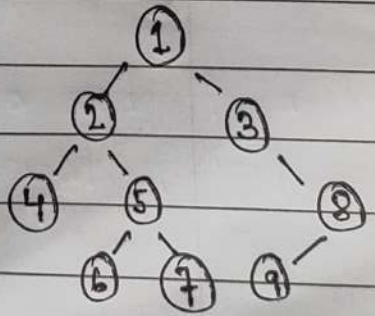
Left → right → root



O/P : [4, 6, 7, 5, 2, 9, 8, 3, 1]

4. **Level Order :**

Visit nodes level by level from top to bottom, left to right.



O/P : [1, 2, 3, 4, 5, 8, 6, 7, 9]