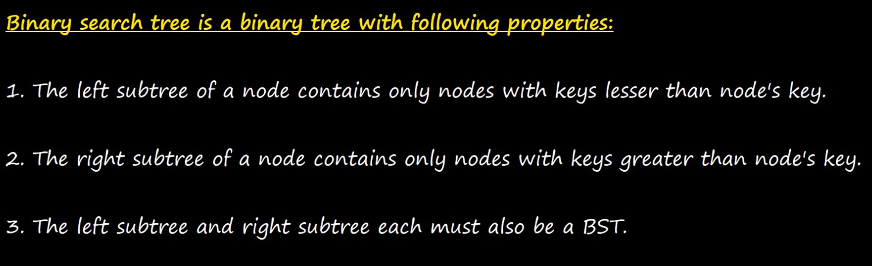
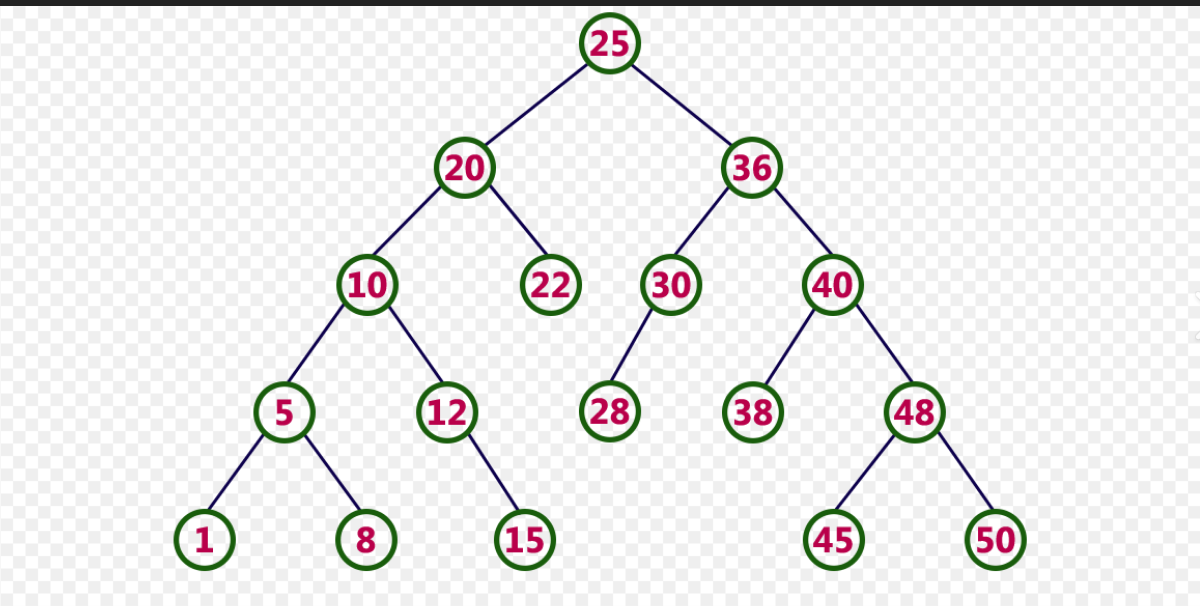
Binary Search Tree :

Properties :

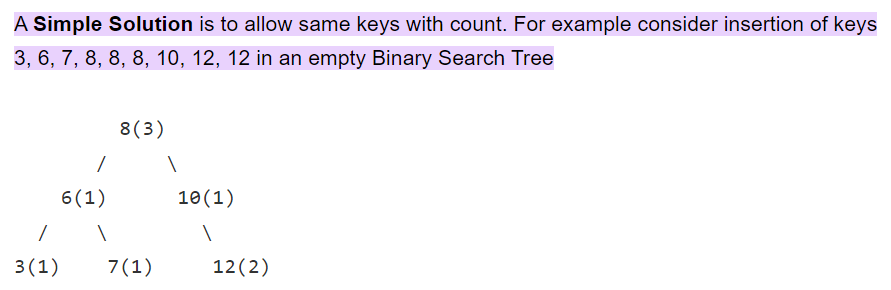




Binary Search Tree with duplicate values :

Aproches :

1. Duplicate values not allowed
2. Laft <= root/node < right
3. Left < root/node <= right



Operations of Binary Search Tree :

1.Searching

Algorithm :

1. If BST —-> empty —then---> given value not present .
2. If root == given value —-then-----> g.v present .
3. Elif —---> g.v < root node —---> search Left S.T
4. Elif —---> g.v > root node —----> search Right S.T .

2. Insertion

Algorithm :

1. If BST —---> empty —---> insert new node
2. Elif root < new\_node —--> insert RIGHT
3. Elif root > new\_node —---> insert LEFT

3.Deletion :

Algorithm :

1. If BST —-is--> empty —--> cant delete
2. Search mentioned key —-if--> found —---> delete node

Case 1 : deleting node that has 0 children

Case 2 : delete the node that has 1 child node

Case 3 : deleting the node that has 2 child node

4. Traversing :

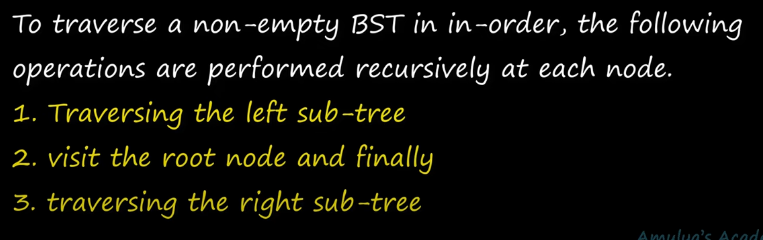
Algorithms :

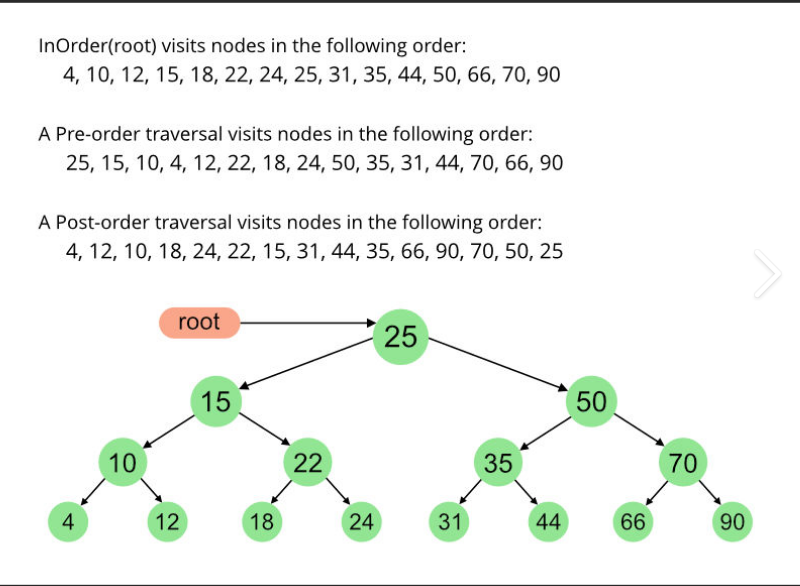
1. Pre-Ordered Traversal :

To traverse a non-empty BST in pre-order , the following operations are performed recursively at each node.

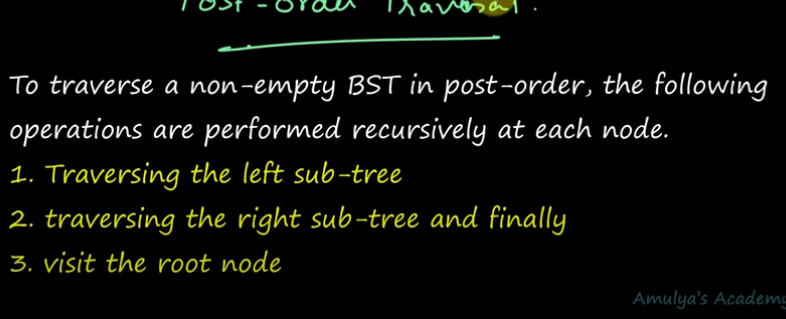
1. Visit the root node.
2. Traversing the left sub-tree .
3. Traversing the right sub-tree.

2. In-order traversal :

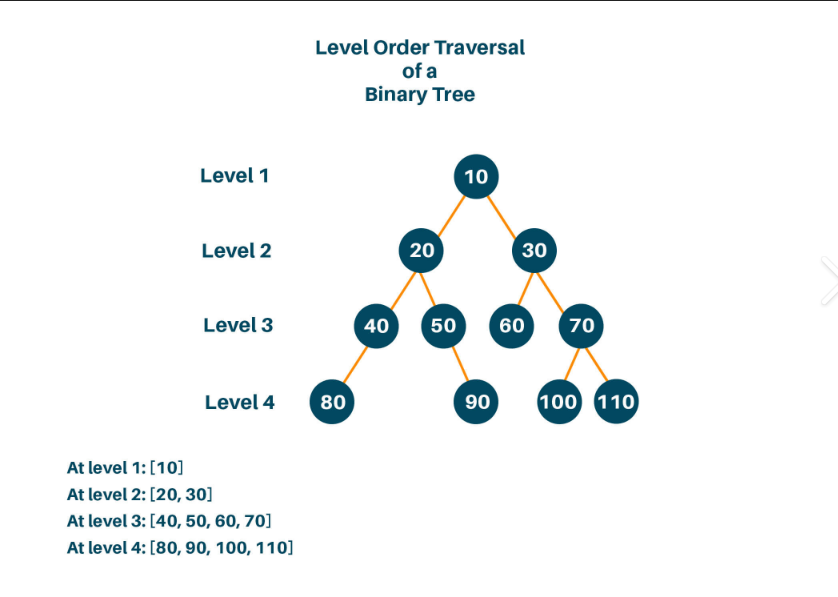




3. Post-Order Traversal ;



1. Level Order Traversal :

You need to visit node level wise .

Summary :

