UNIT-5

Binary Trees ...

Learn Vid Fun...

Tree Traversal:

Tree traversal refers to the process of visiting each node in a tree data structure exactly once.

Tree Traversal Techniques

- Depth First Traversal
  - i) Preorder Traversal
  - ii) Inorder Traversal
  - iii) Postorder Traversal
- Breadth First or Level Order Traversal

I) Preorder Traversal :

Algorithm :

i) Visit the root

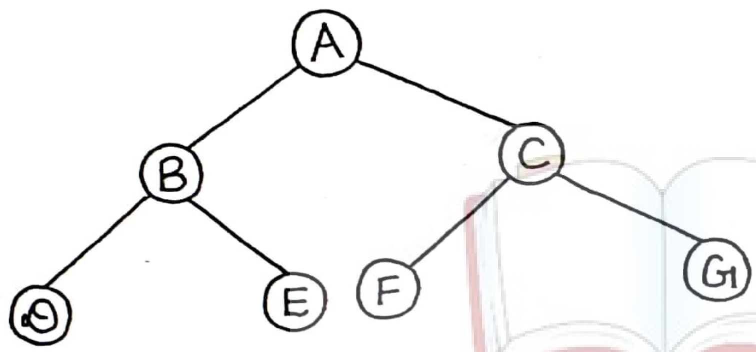
ii) Traverse the left subtree i.e. call Preorder (left subtree)

iii) Traverse the right subtree i.e. call Preorder (Right subtree)

Remember :

Root → Left → Right

Example:

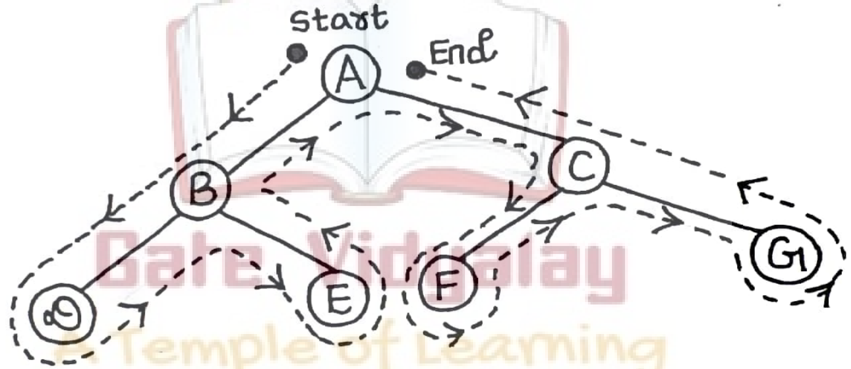


Binary Tree

Preorder Traversal : A B D E C F G

Shortcut for Preorder Traversal :

Just traverse the entire tree starting from the root node keeping yourself to the left.



∴ Preorder Traversal = A B D E C F G

II) In order Traversal :

Algorithm :

i) Traverse the left subtree i.e. call Inorder (left-subtree)

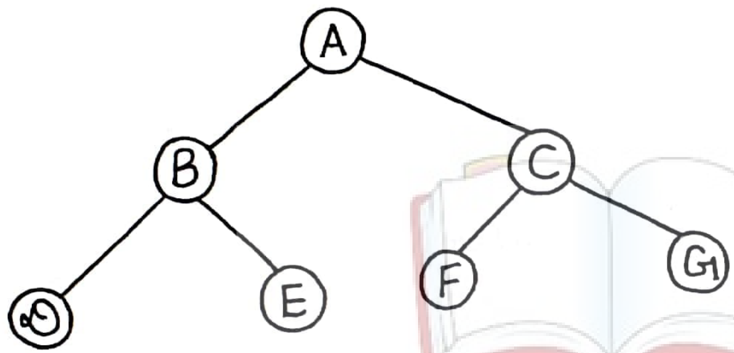
ii) Visit the root

iii) Traverse the right subtree i.e. call Inorder (right-subtree)

Remember :

Left → Root → Right

Example:

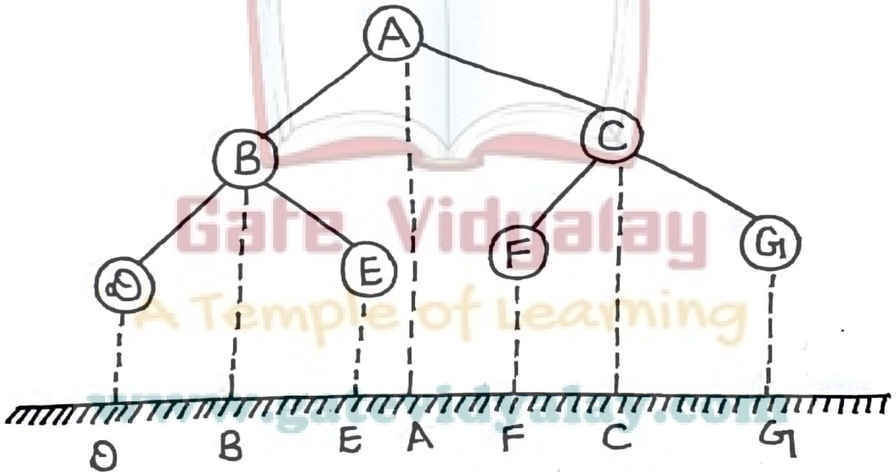


Binary Tree

Inorder Traversal : D B E A F C G

Shortcut for Inorder Traversal :

Just keep a plane mirror horizontally at the bottom of the tree and take the projection of all nodes.



∴ Inorder Traversal = D B E A F C G

iii) Postorder Traversal:

Algorithm:

i) Traverse the left subtree i.e. call Postorder (Left subtree)

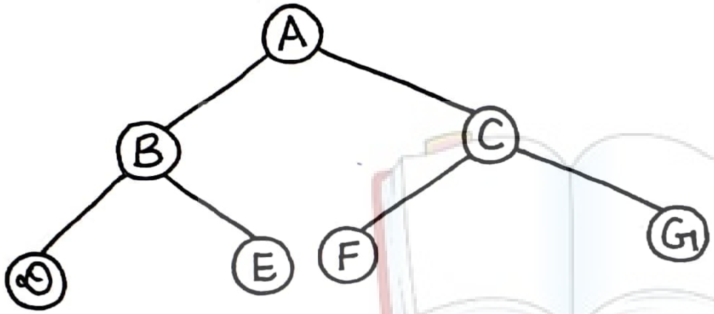
ii) Traverse the right subtree i.e. call Postorder (Right subtree)

iii) Visit the root

Remember:

Left → Right → Root

Example:

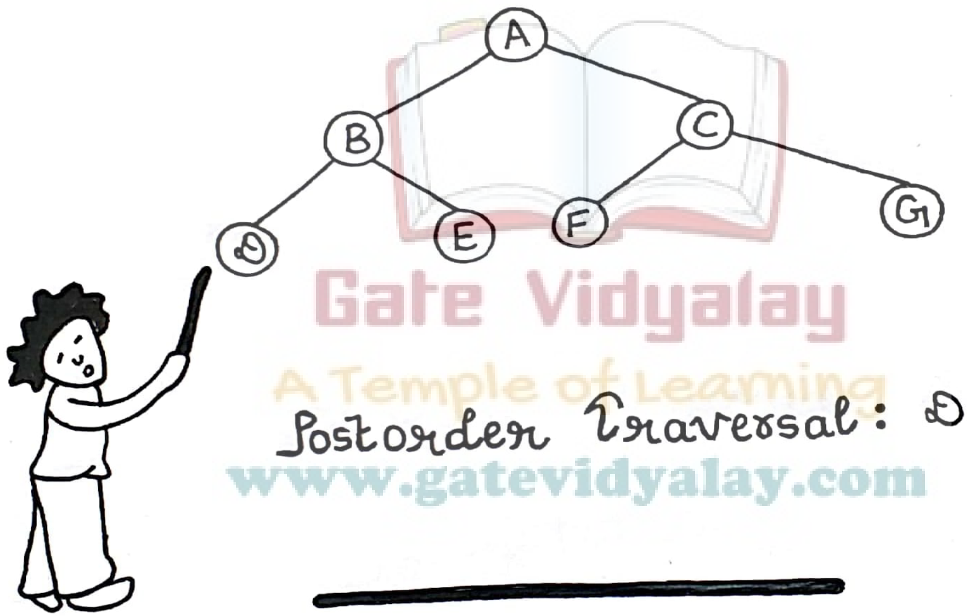


Binary Tree

Postorder Traversal : D E B F G C A

Shortcut for Postorder Traversal:

Just pluck the leftmost leaf nodes one by one.

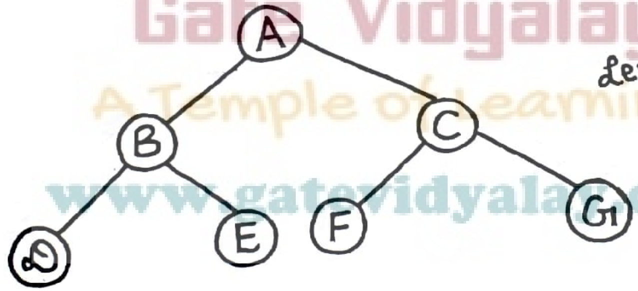


Postorder Traversal: D E B F G C A

iv) Level Order Traversal:

- Level Order Traversal of a tree is the breadth first traversal of a tree which prints all the nodes of a tree level by level.

- **Example:**



Level order Traversal:

A B C D E F G

Important Points for exam:

- Preorder traversal is used to get prefix expression of an expression tree.
- Inorder traversal is used to get infix expression of an expression tree.
- Postorder traversal is used to get postfix expression of an expression tree.
- Preorder traversal is used to create a copy of the tree.
- Postorder traversal is used to delete the tree.
- Level order traversal prints the data in the same order as it is stored in the array representation of complete binary tree.

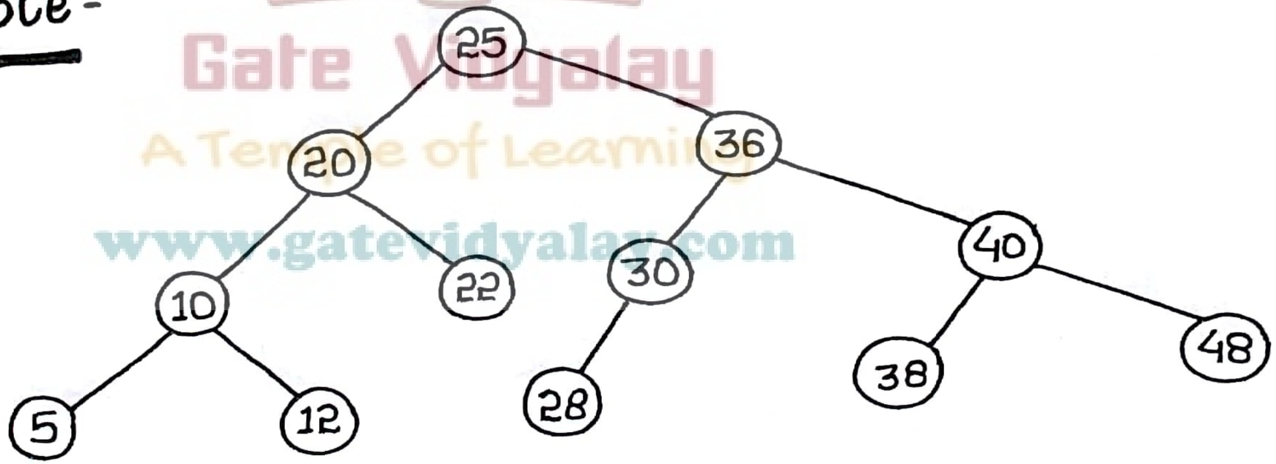
Binary Search Trees

Learn Vid Fun...

Definition-

Binary Search Tree (BST) is a special kind of Binary tree in which every node contains smaller values only in its left subtree and only larger values in its right subtree.

Example-



Number of distinct BSTs with 'n' distinct Keys-

$$\frac{{}^{2n}C_n}{n+1}$$

Example-

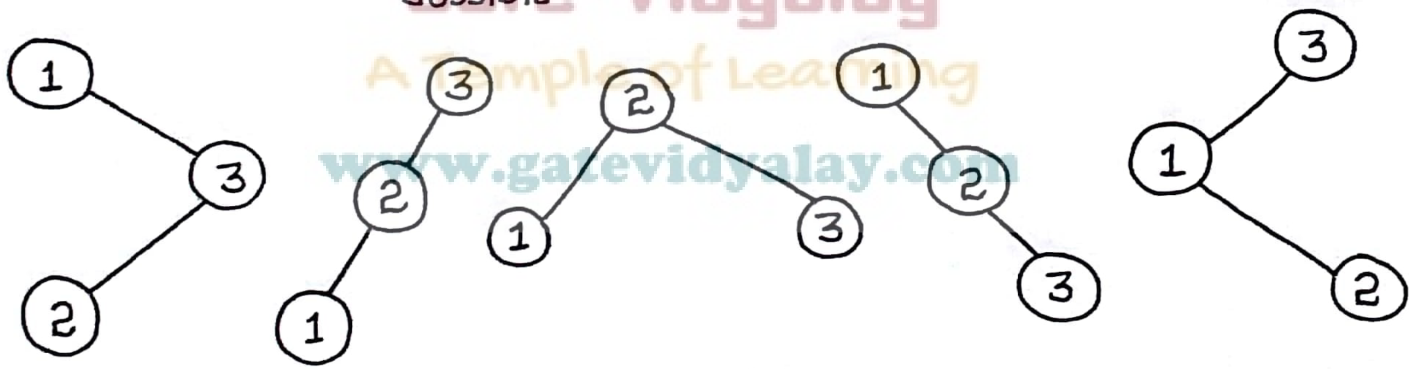
Number of distinct Binary search Trees with 3 distinct Keys

$= {}^{2\times3}C_3 / 3+1$

$= 5$

Consider three distinct Keys are - 1 , 2 , 3

Possible BSTs are -



Construction of BST-

Question-

Construct a Binary Search Tree (BST) for the following sequence of numbers-

50, 70, 60, 20, 90, 10, 40, 100

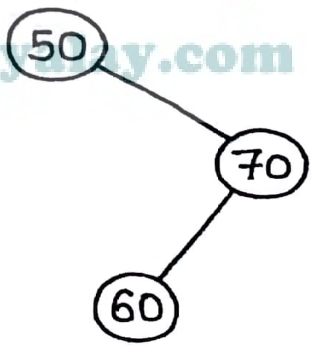
Solution-

When elements are given in a sequence, we consider the first element as the root node.

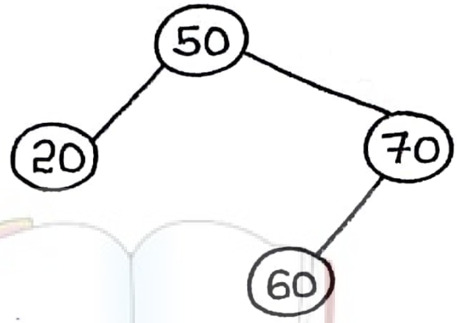
- **Insert 50 -**

- **Insert 70 -**

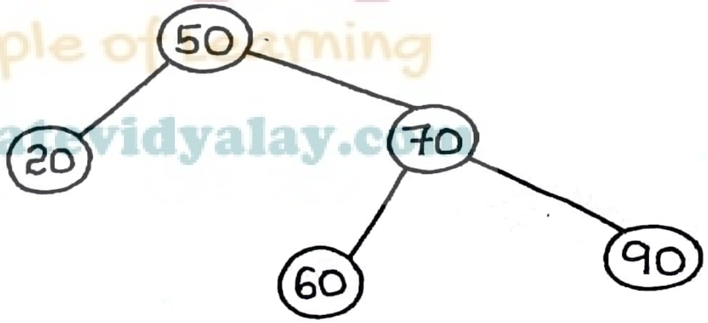
- **Insert 60 -**



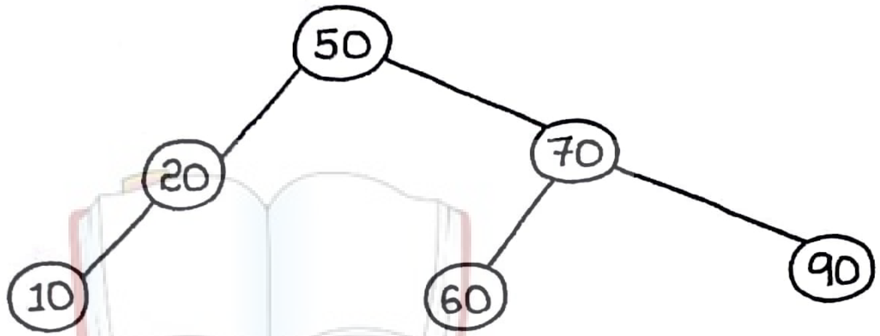
- **Insert 20 -**



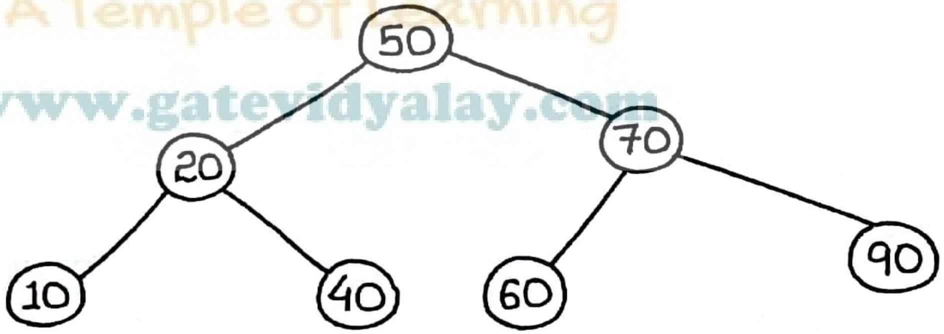
- **Insert 90 -**



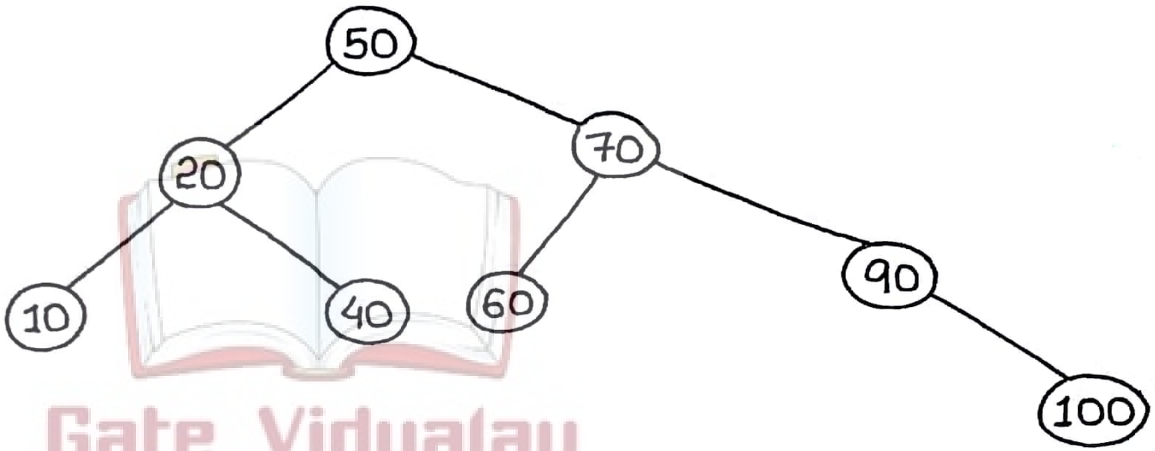
- **Insert 10 -**



- **Insert 40 -**



- **Insert 100 -**



This is the required Binary search Tree.

Binary Search Trees

Learn Vid Fun...

Operations on BST -

The following operations are performed on a Binary Search Tree -

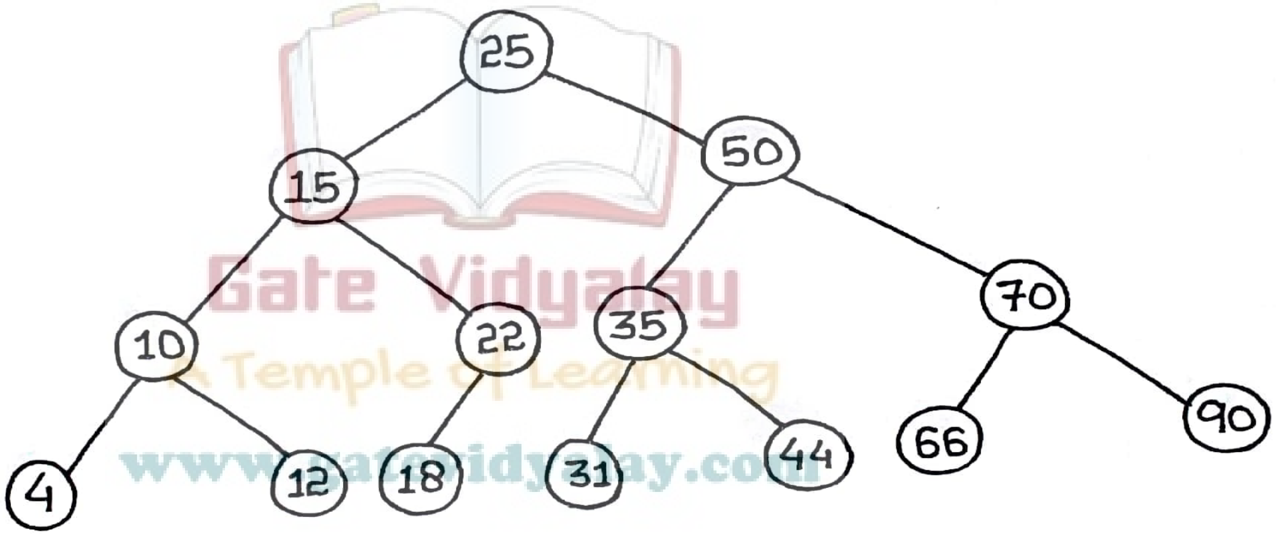
- Search Operation
- Insertion Operation
- Deletion Operation

Search Operation-

To search a given key in Binary Search Tree, we first compare it with root. If the key is present at root, we return root. If the key is greater than root's key, we recur for right subtree of root node otherwise we recur for left subtree.

Example-

Search for 45 in the BST-



Step-01: Start at the root. As 45 > 25, so search in right subtree.

Step-02: As 45 < 50, so search in 50's left subtree.

Step-03: As 45 > 35, so search in 35's right subtree.

Step-04: As 45 > 44, so search in 44's right subtree. But 44 has no subtrees. So, 44 is not present in the BST.

Insertion Operation-

A new key is always inserted at leaf. We start searching a key from root till we hit a leaf node. Once a leaf node is found, the new node is added as a child of the leaf node.

Example-

Step-01: Start at root node 100. As 40 < 100, so search in 100's right subtree.

Step-02: As 40 > 20, so search in 20's right subtree.

Step-03: As 40 > 30 (leaf node), so add 40 to 30's right subtree.

Deletion Operation-

Deleting a node from Binary search Tree gives rise to following 3 cases-

Case-I: Deleting a node with no child (leaf node)

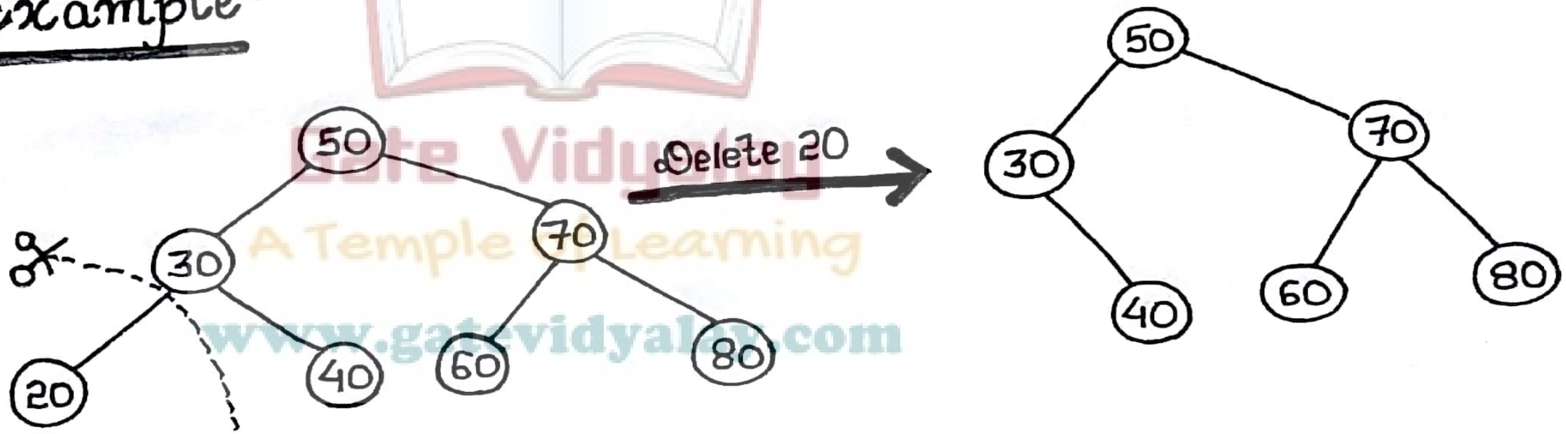
Case-II: Deleting a node with one child

Case-III: Deleting a node with two children

Case-I: Deleting a leaf node-

It is very simple. Just remove the leaf node from the tree.

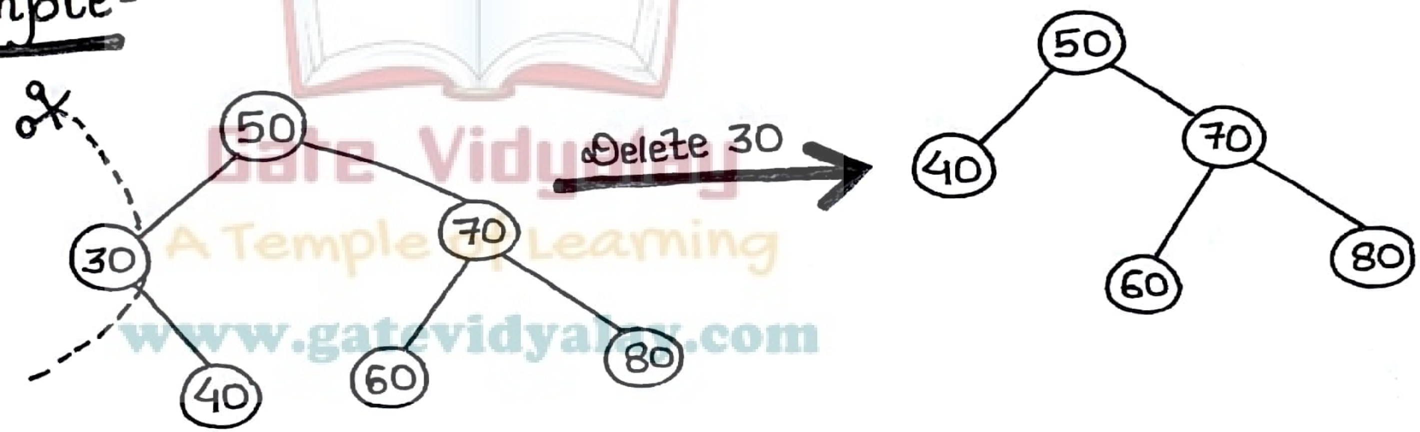
Example-



Case-II: Deleting a node with one child-

Just make the child of the deleting node, the child of the grandparent.

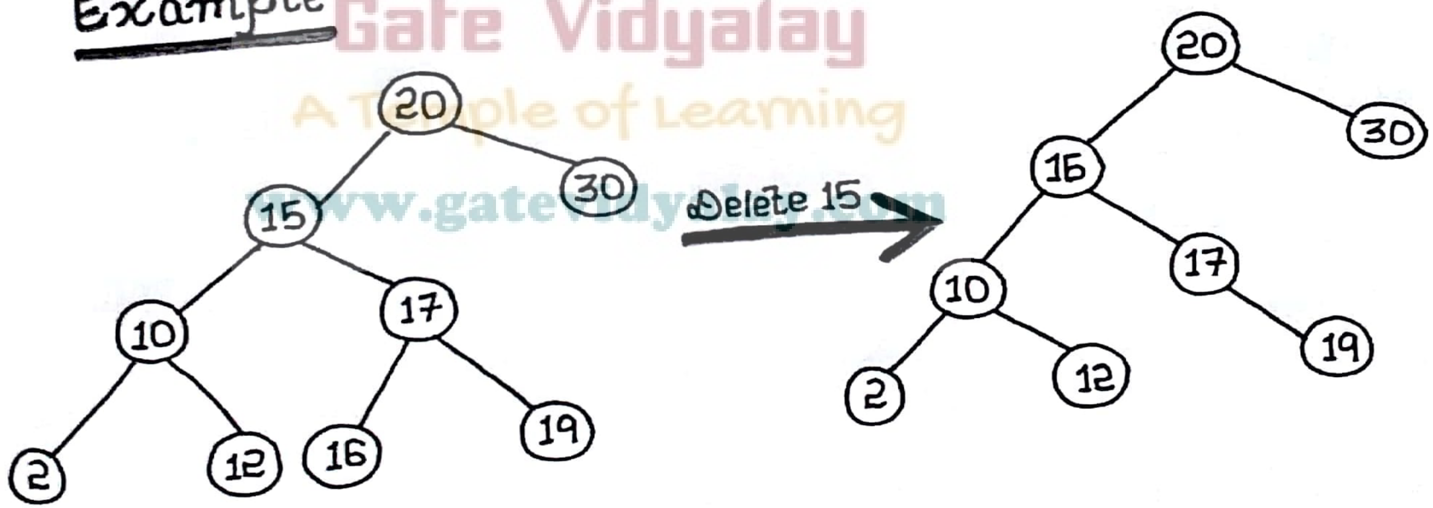
Example-



Case-III: Deleting a node with 2 children-

Method-I: Go to the right subtree of the deleting node, pluck the least element called inorder successor and replace with the deleting node.

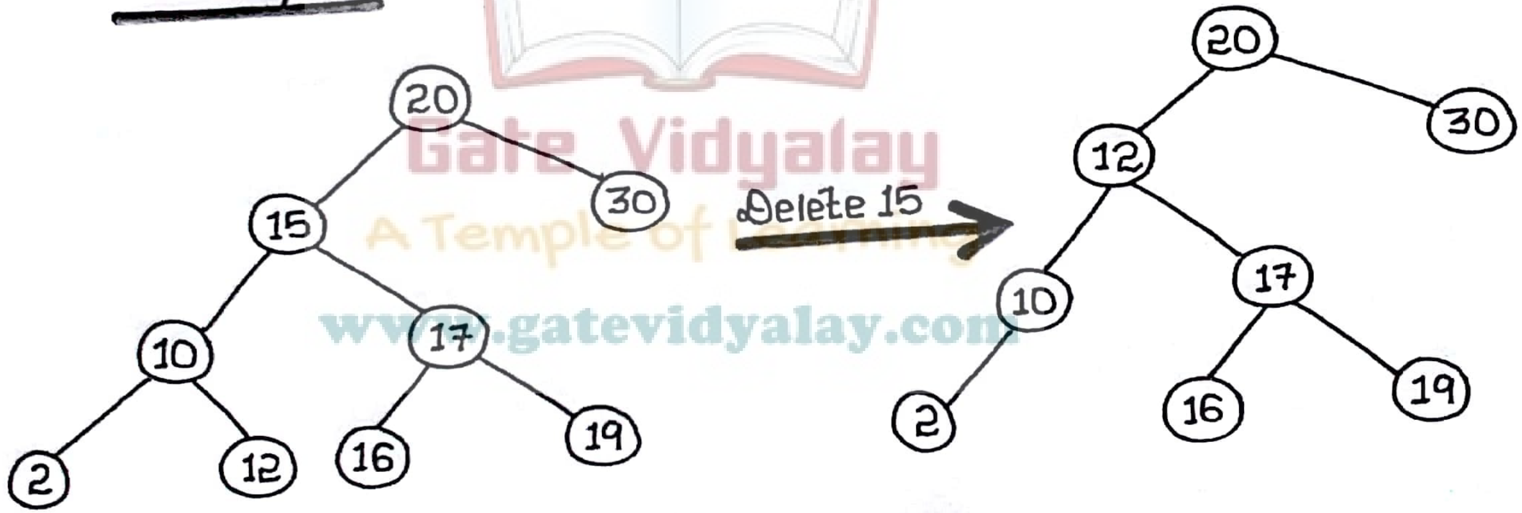
Example-



Method-II:

Go to the left subtree of the deleting node, pluck the greatest element called inorder predecessor and replace with the deleting node.

Example-



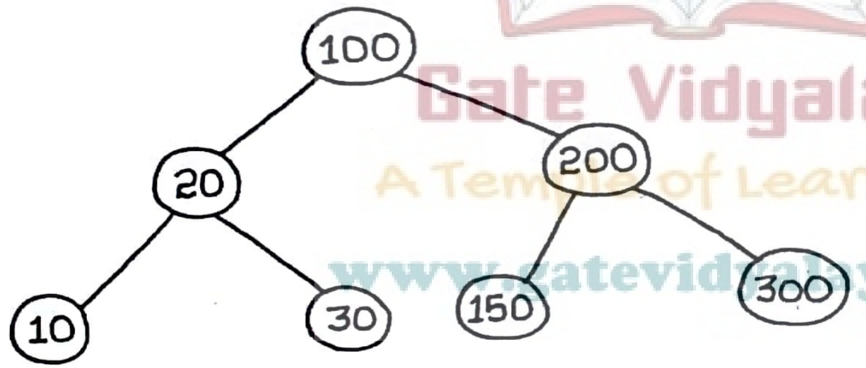
Binary Search Trees

Learn Vid Fun...

Traversal of a Binary Search Tree-

Traversal of a Binary search Tree is exactly same as that of a Binary Tree.

Example-



- **Preorder Traversal**

100, 20, 10, 30, 200, 150, 300

- **Inorder Traversal**

10, 20, 30, 100, 150, 200, 300

- **Postorder Traversal**

10, 30, 20, 150, 300, 200, 100

Important Points-

- Inorder Traversal of a Binary Search Tree (BST) always yields all the nodes in increasing order.

- We can construct a Binary Search Tree with only preorder or postorder traversal because you can always get the inorder traversal by sorting the given traversal result in increasing order.

Binary Search Trees

Learn Vid Fun...

Time Complexities of BST Operations

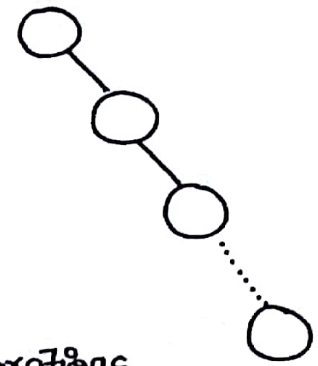
The time complexity for all Binary Search Tree operations be it search operation or Insert operation or Delete operation is $O(h)$ where h is the height of a Binary Search Tree.

Thus, In general-

Time complexity of BST Operations = O(height)

Worst Case-

In worst case, the binary search tree is a skewed binary search tree and we have to travel from root to the deepest leaf node.



In that case, the height of the binary search tree becomes 'n'.

Thus,

In worst case, Time complexity for BST operations

$= O(n)$

In this case, BST is as good as unordered list with no benefits.

Binary Search Tree in Worst Case-

Example-

Skewed Binary Search Tree

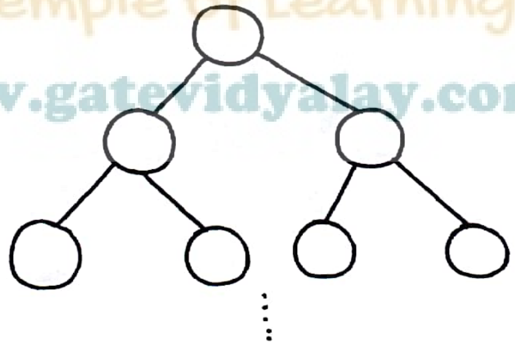
Best Case-

In best case, the binary search tree is a balanced binary search tree with height 'logn'

Thus,

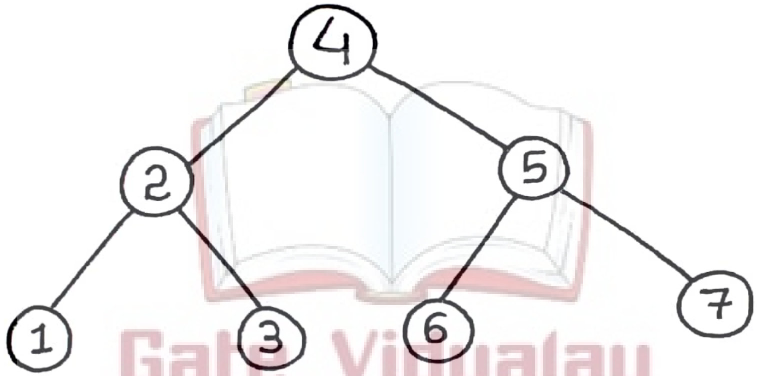
In Best Case, Time Complexity of BST Operations

$= O(\log n)$



Binary Search Tree in Best Case-

Example-



Balanced Binary Search Tree