DATA STRUCTURE - 02

1. Tree

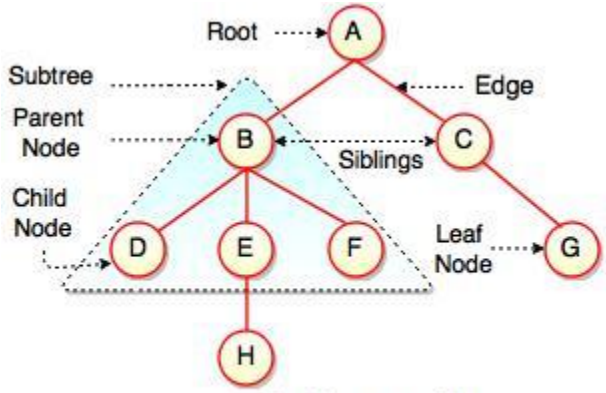
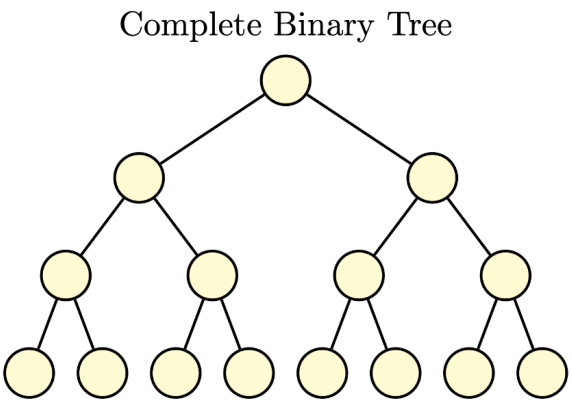


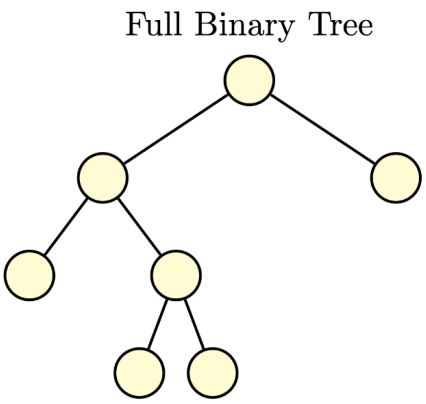
Fig. Structure of Tree

Important Terms

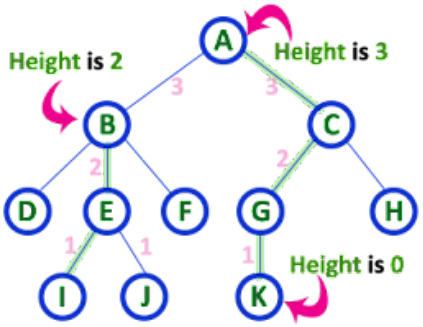
- **Path** – Path refers to the sequence of nodes along the edges of a tree.
- **Root** – The node at the top of the tree is called root. There is only one root per tree and one path from the root node to any node.
- **Parent** – Any node except the root node has one edge upward to a node called the parent.
- **Child** – The node below a given node connected by its edge downward is called its child node.
- **Leaf** – The node which does not have any child node is called the leaf node.
- **Subtree** – Subtree represents the descendants of a node.
- **Visiting** – Visiting refers to checking the value of a node when control is on the node.
- **Traversing** – Traversing means passing through nodes in a specific order.
- **Levels** – The level of a node represents the generation of a node. If the root node is at level 0, then its next child node is at level 1, its grandchild is at level 2, and so on.
- **Keys** – Key represents a value of a node based on which a search operation is to be carried out for a node.

- **Full Binary Tree** (0 or 2 Child)
- **Complete Binary Tree** (No Holes)





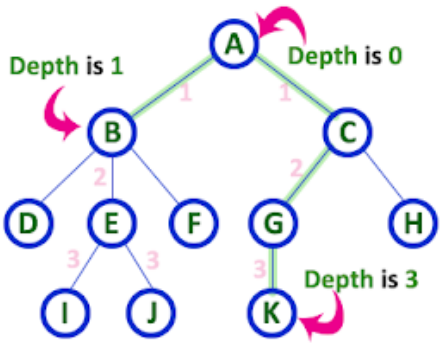
- **Height of Tree**



Here Height of tree is 3

- In any tree, 'Height of Node' is total number of Edges from leaf to that node in longest path.
- In any tree, 'Height of Tree' is the height of the root node.

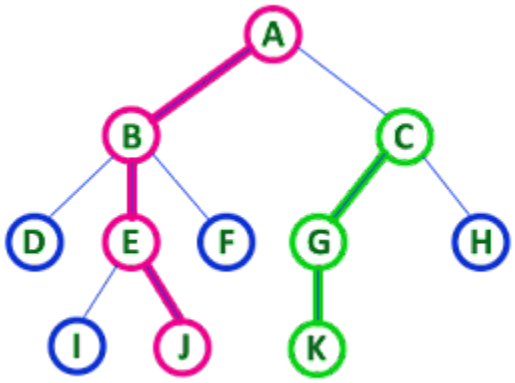
- **Depth of Tree**



Here Depth of tree is 3

- In any tree, 'Depth of Node' is total number of Edges from root to that node.
- In any tree, 'Depth of Tree' is total number of edges from root to leaf in the longest path.

- **Path in Tree**

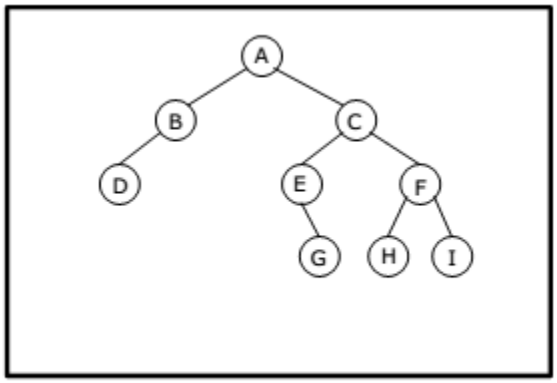


- In any tree, 'Path' is a sequence of nodes and edges between two nodes.

Here, 'Path' between A & J is
A - B - E - J

Here, 'Path' between C & K is
C - G - K

- **Traversal**
  - Pre-Order (Root, Left, Right) **[+5,2] = 7**
  - In-Order (Left, Root, Right) **[2+9] = 11**
  - Post-Order (Left, Right, Root) **[2,4+] = 6**



Binary Tree

- Preorder traversal yields:
  A, B, D, C, E, G, F, H, I
- Postorder traversal yields:
  D, B, G, E, H, I, F, C, A
- Inorder traversal yields:
  D, B, A, E, G, C, H, F, I
- Level order traversal yields:
  A, B, C, D, E, F, G, H, I

Pre, Post, Inorder and level order Traversing

Binary Search Tree

1. Binary Search tree can be defined as a class of binary trees, in which the nodes are arranged in a specific order. This is also called an ordered binary tree.

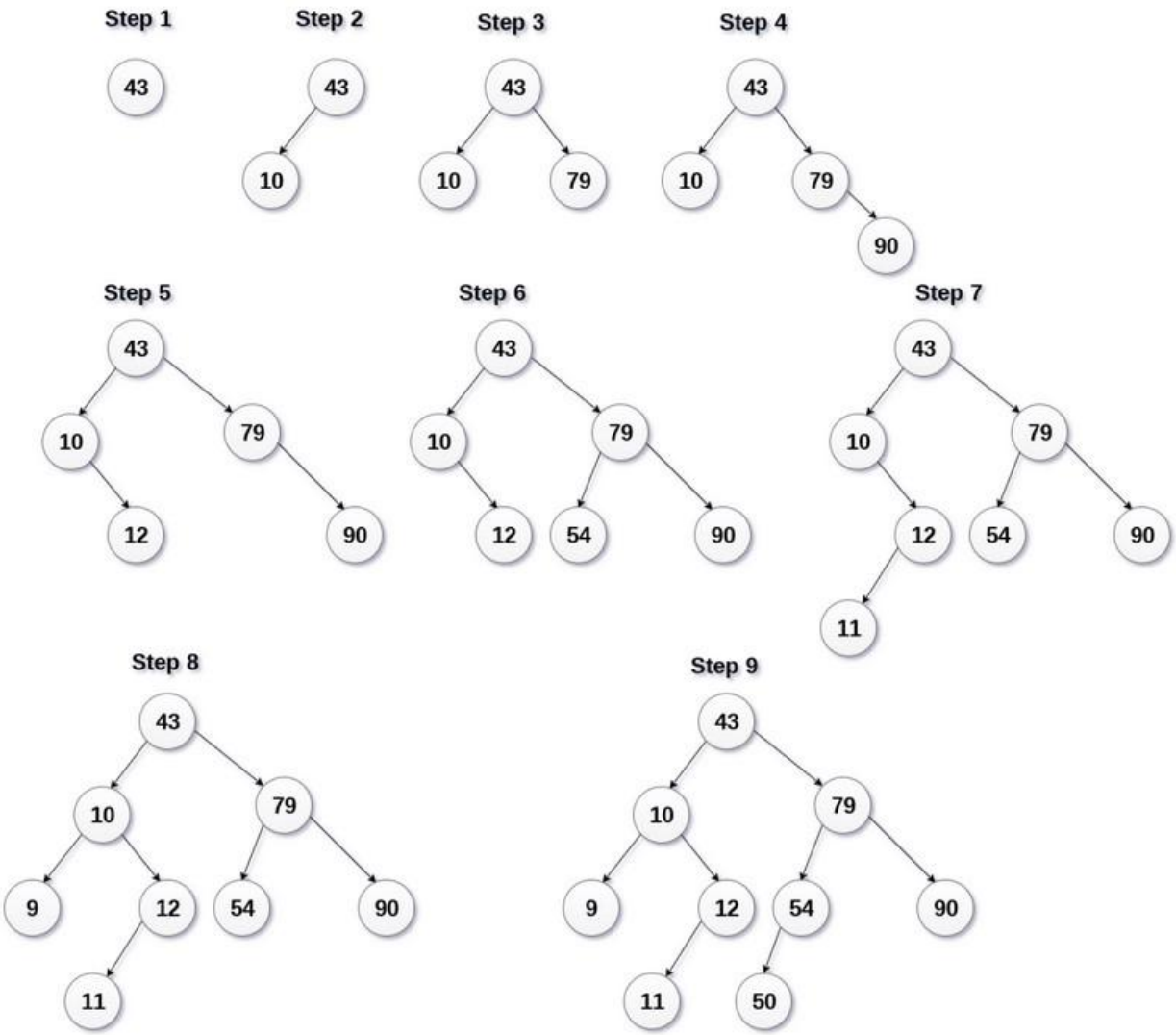
2. In a binary search tree, the value of all the nodes in the left sub-tree is less than the value of the root.

3. Similarly, the value of all the nodes in the right subtree is greater than or equal to the value of the root.

4. This rule will be recursively applied to all the left and right subtrees of the root.

Q. Create the binary search tree using the following data elements.

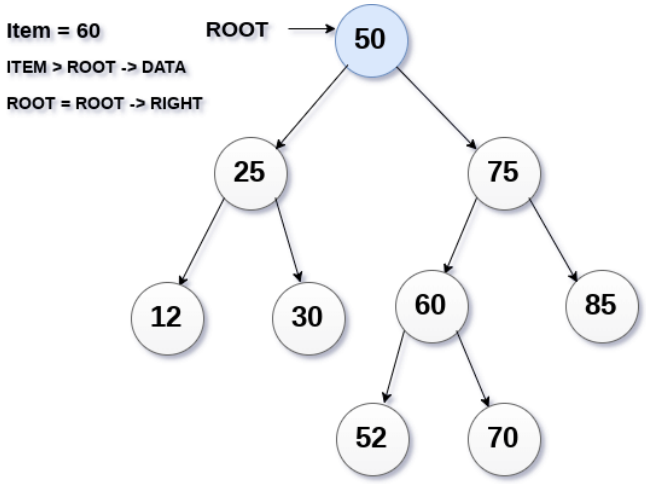
43, 10, 79, 90, 12, 54, 11, 9, 50



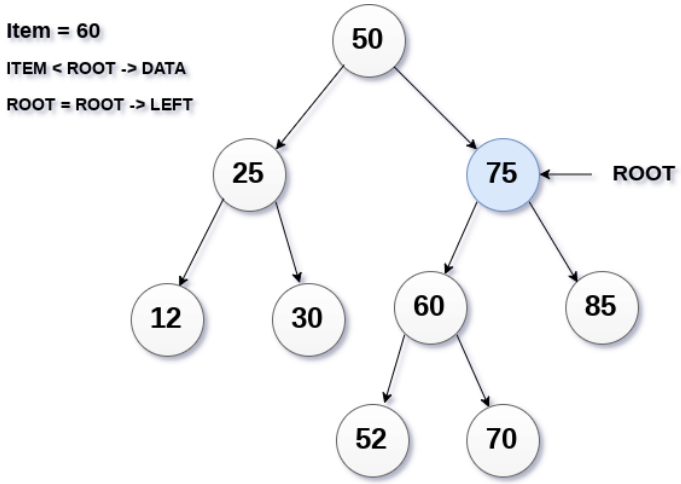
Operations on Binary Search Tree

- Searching
- Insertion
- Deletion

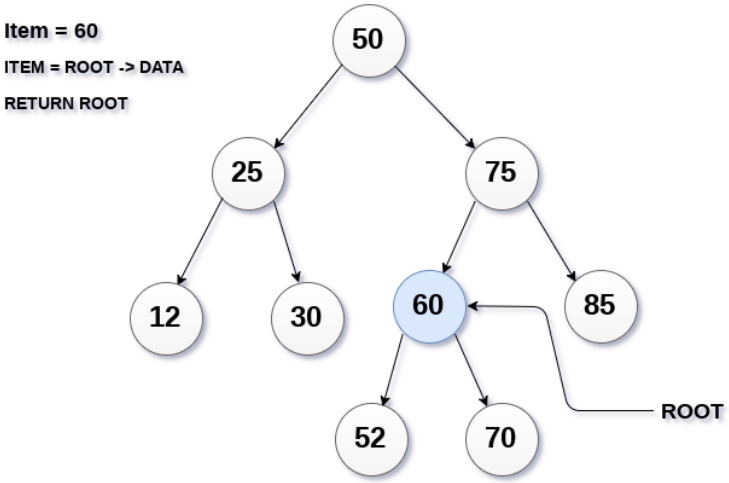
Searching in Binary Search Tree



STEP 1

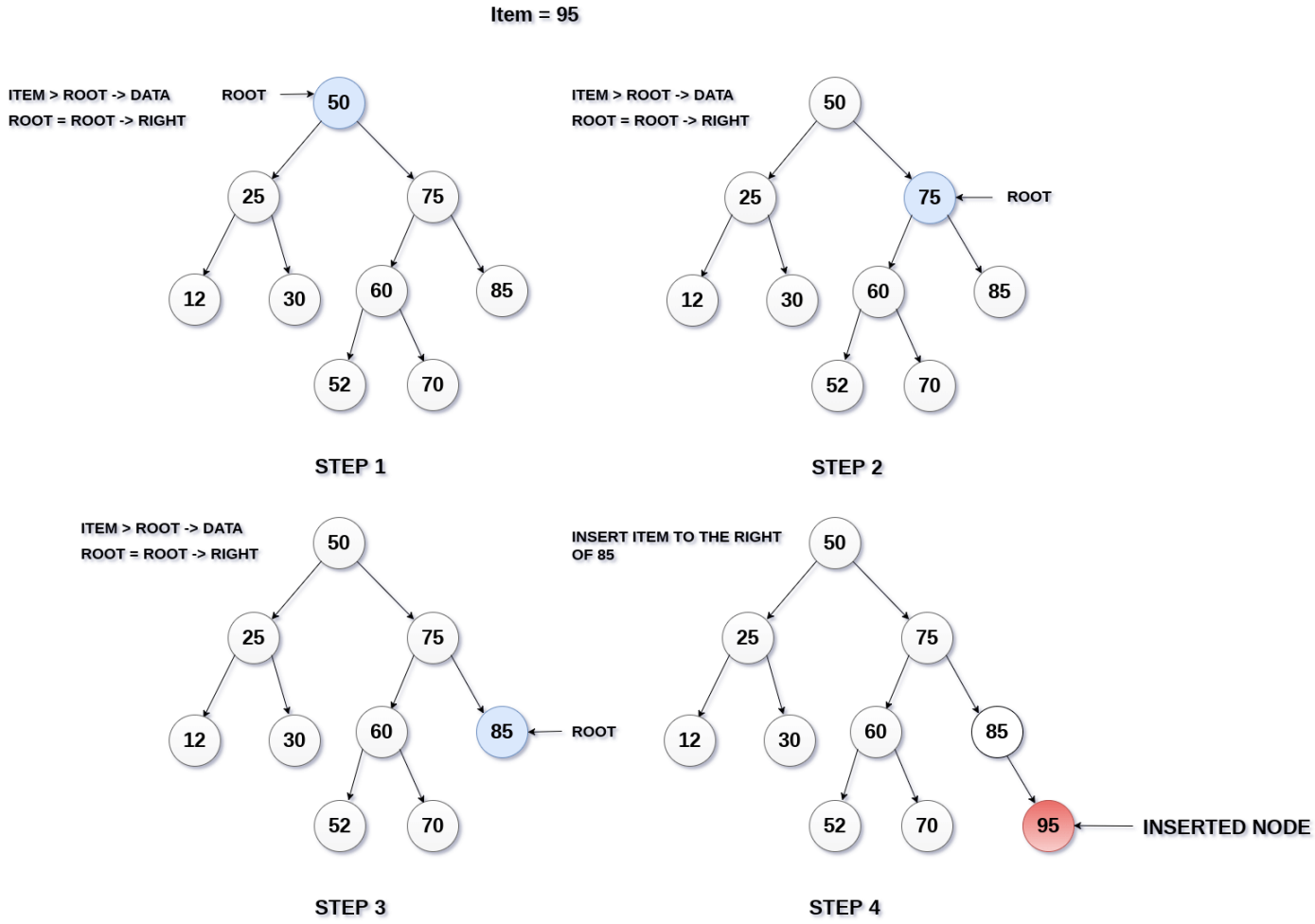


STEP 2



STEP 3

Insertion in Binary Search Tree

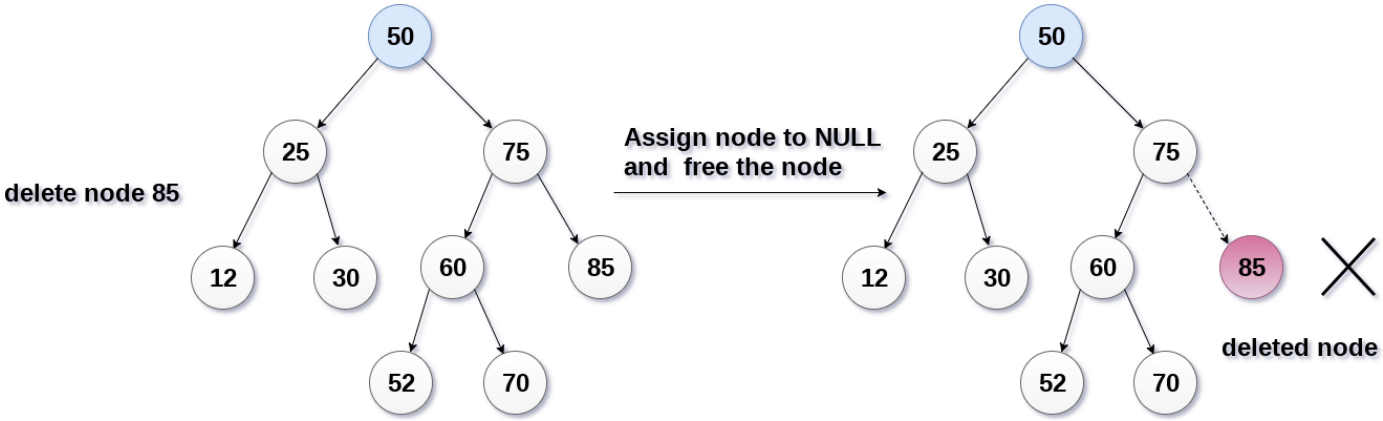


Deletion in Binary Search Tree

The node to be deleted is a leaf node

It is the simplest case, in this case, to replace the leaf node with the NULL and simply free the allocated space.

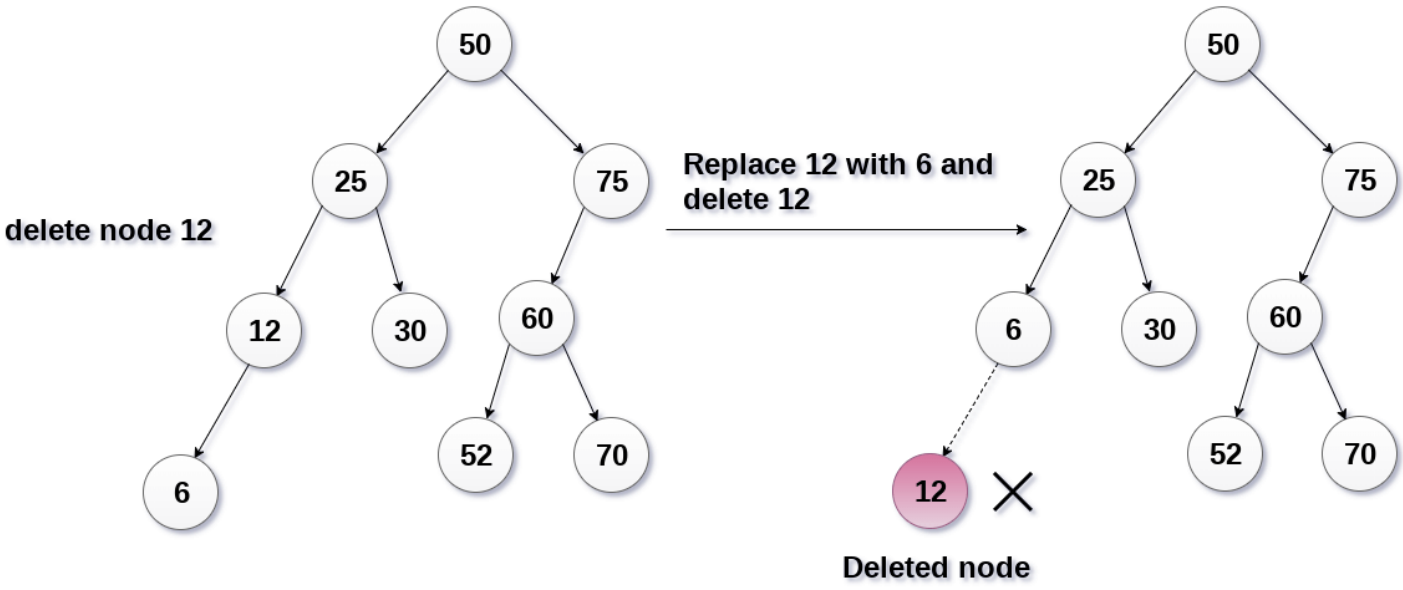
In the following image, we are deleting the node 85, since the node is a leaf node, therefore the node will be replaced with NULL and allocated space will be freed.



The node to be deleted has only one child.

In this case, replace the node with its child and delete the child node, which now contains the value which is to be deleted. Simply replace it with the NULL and free the allocated space.

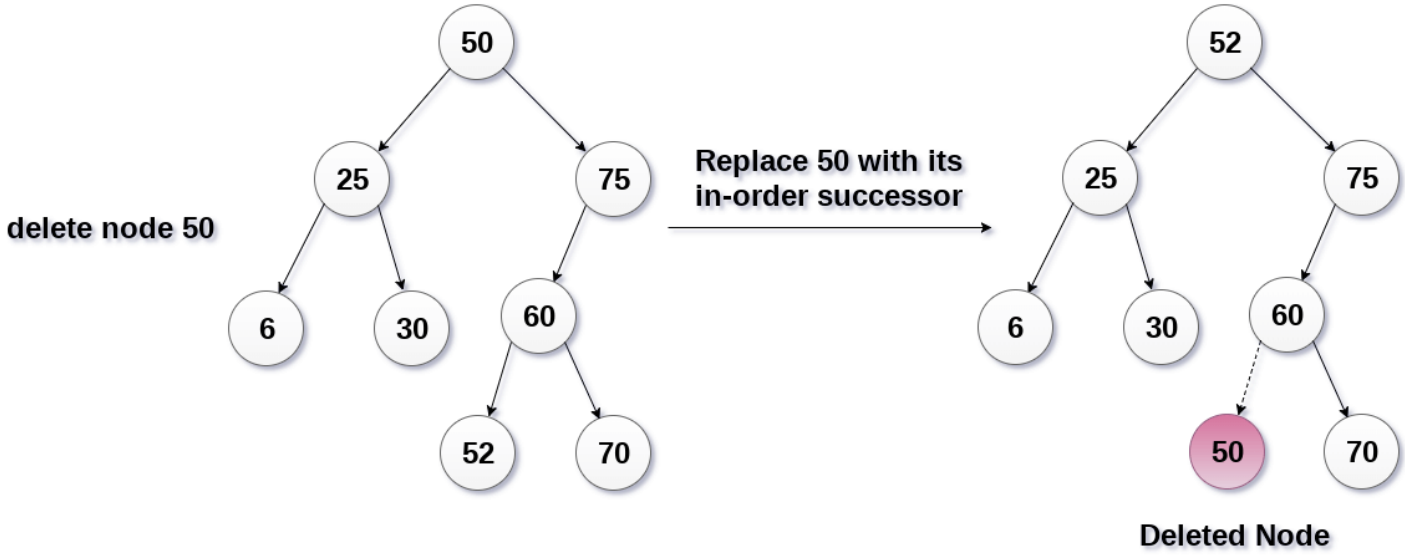
In the following image, the node 12 is to be deleted. It has only one child. The node will be replaced with its child node and the replaced node 12 (which is now leaf node) will simply be deleted.



The node to be deleted has two children.

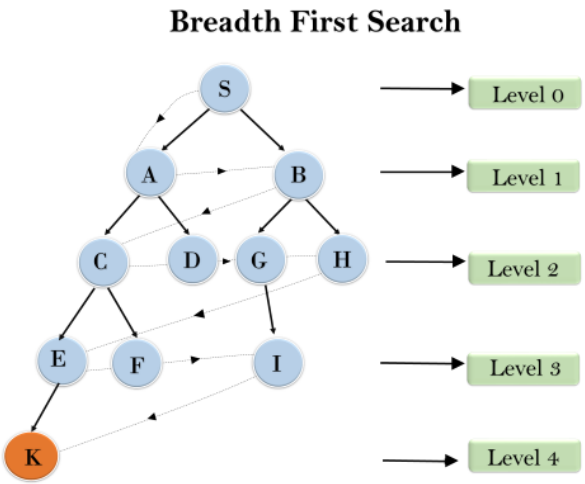
It is a bit complex compared to the other two cases. However, the node which is to be deleted, is replaced with its in-order successor or predecessor recursively until the node value (to be deleted) is placed on the leaf of the tree. After the procedure, replace the node with NULL and free the allocated space.

In the following image, the node 50 is to be deleted which is the root node of the tree. The in-order traversal of the tree given below.

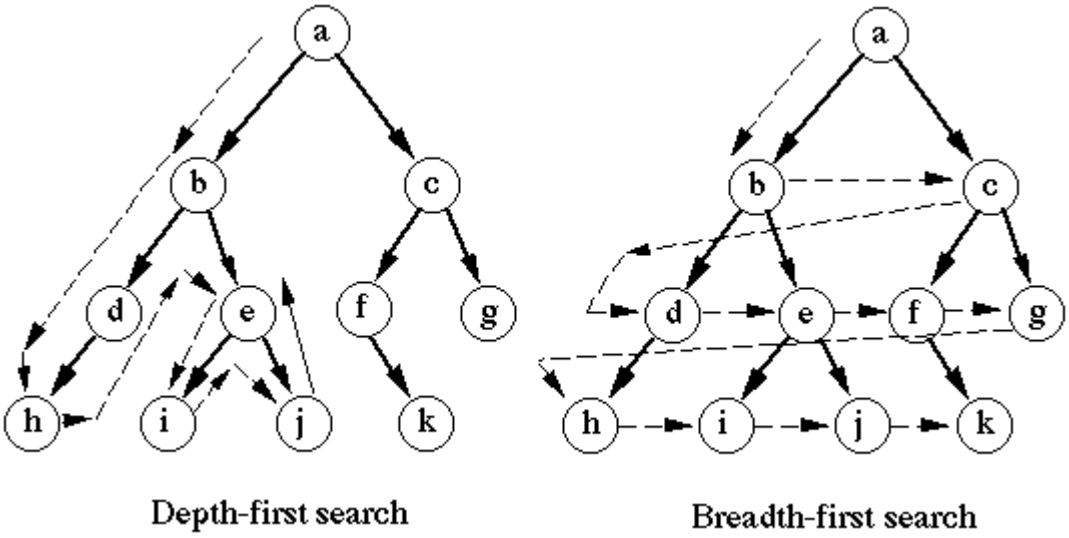


1. Breadth First Search
2. Depth First Search

Breadth First Search



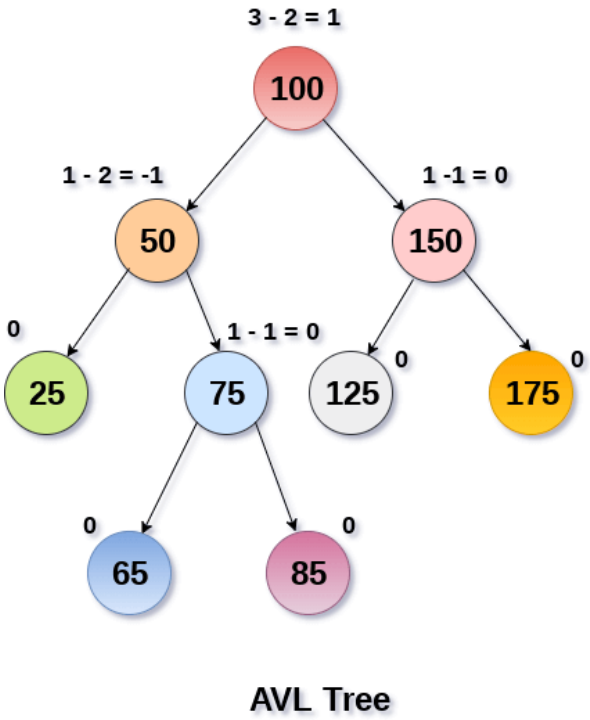
Depth First Search



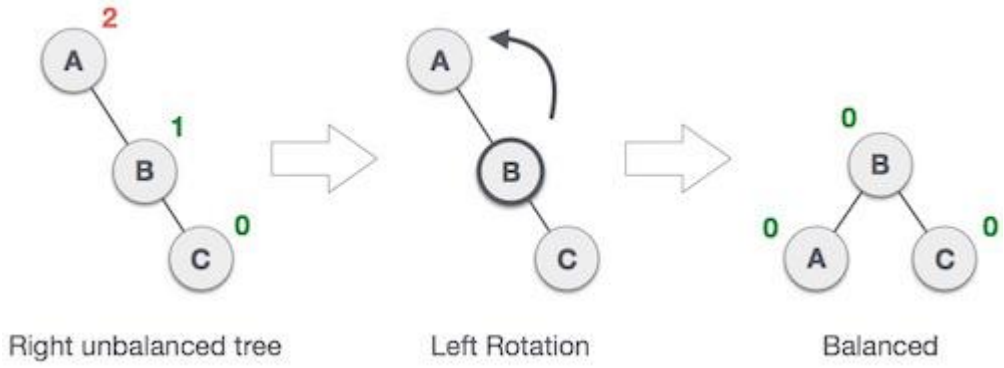
Depth First Search = {a,b,d,h,e,i,j,c,f,k,g}
Breadth First Search = {a,b,c,d,e,f,g,h,i,j,k}

AVL Tree (Balanced Binary Search Tree)

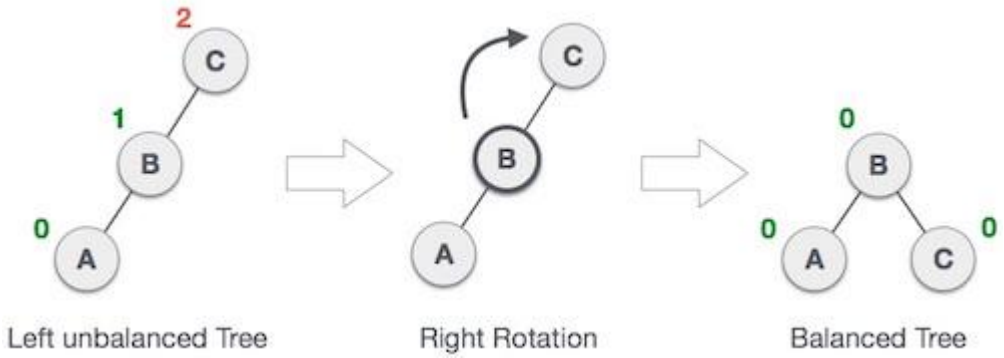
Balance factor = Left subtree - Right subtree (-1,0,+1)



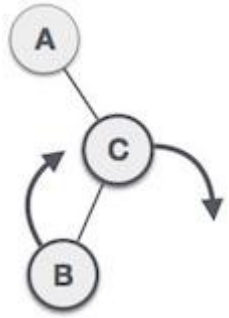
RR rotation (anti clockwise)



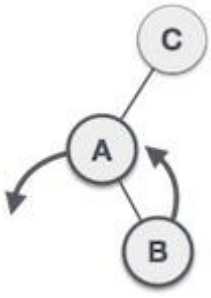
LL rotation (clockwise)



RL - RR rotation

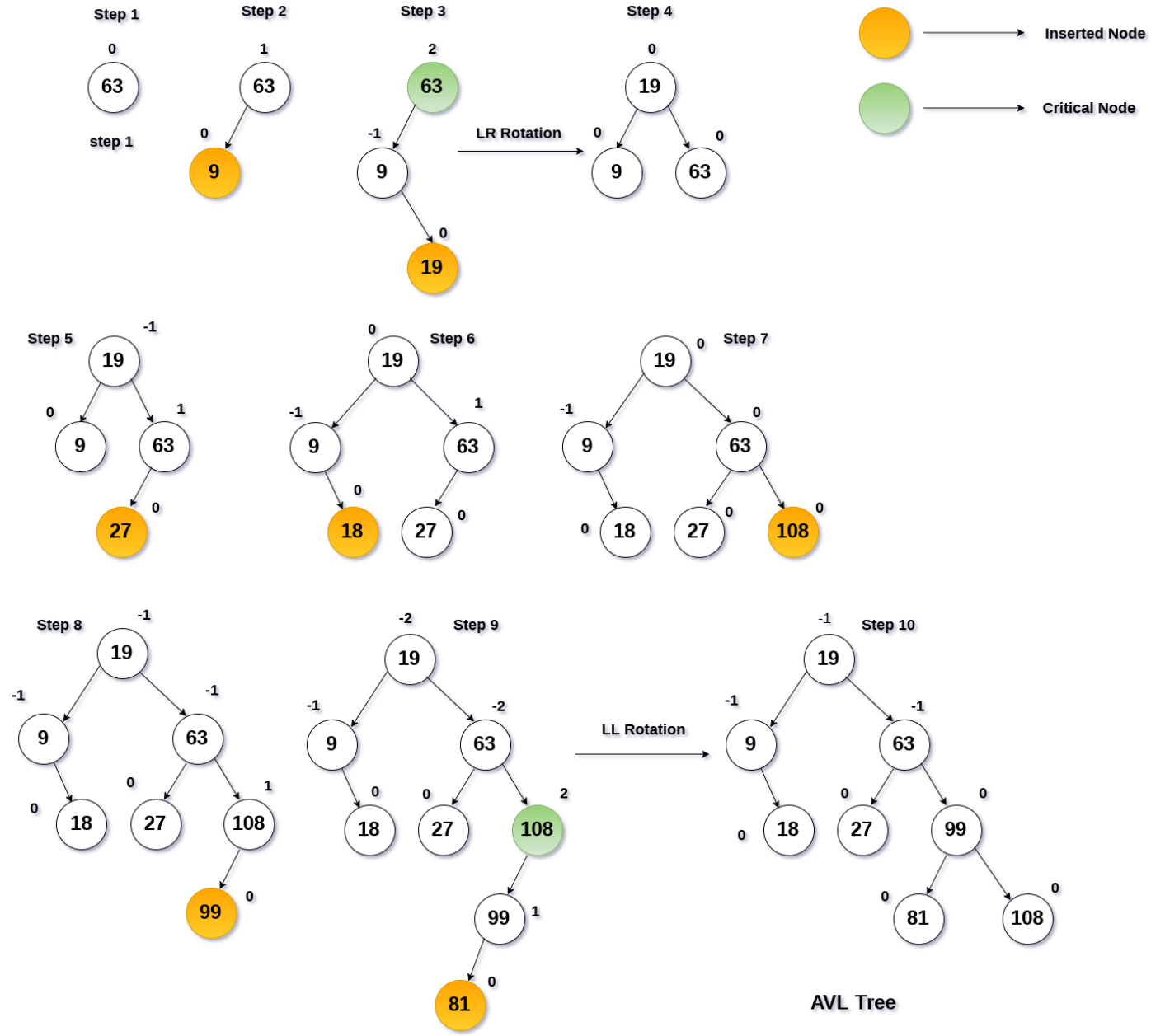


LR - LL rotation

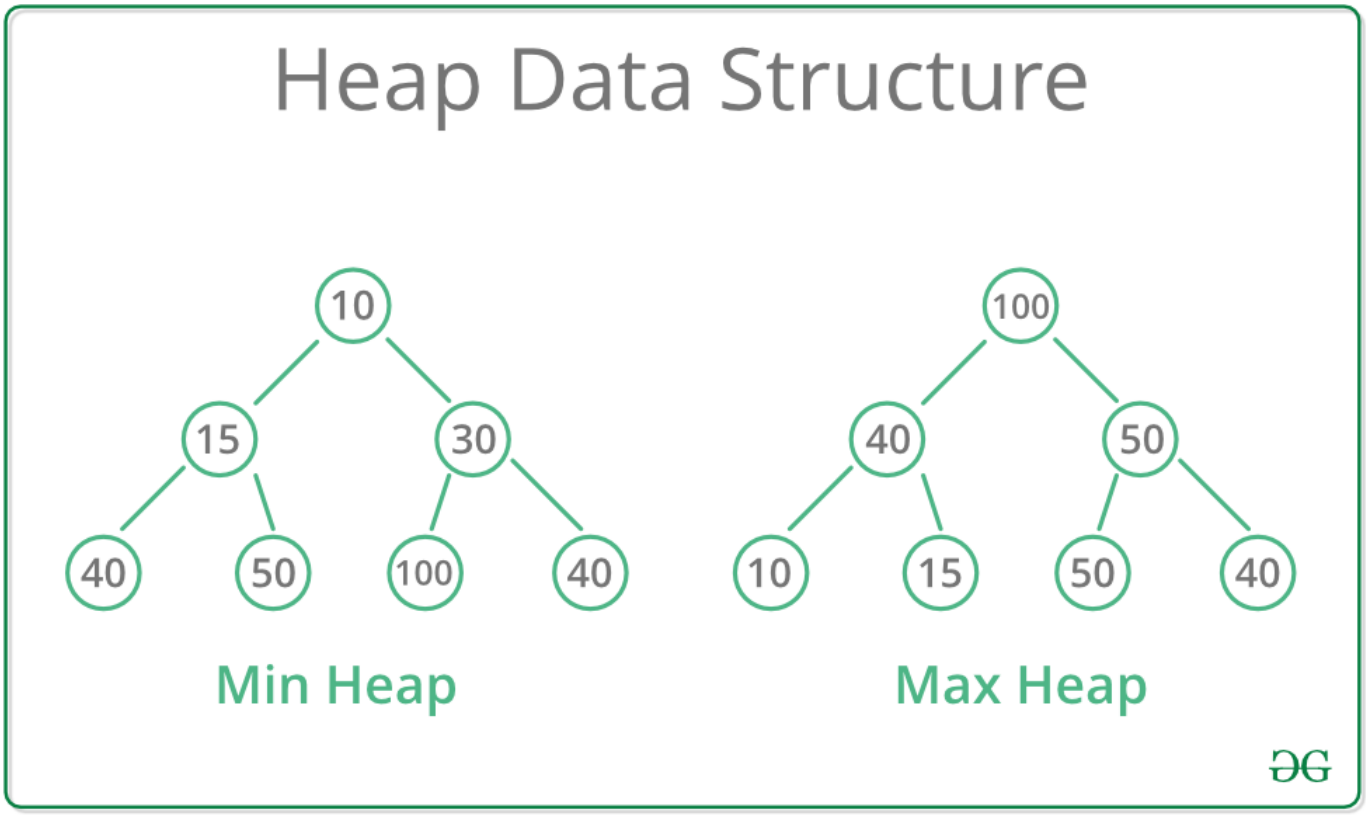


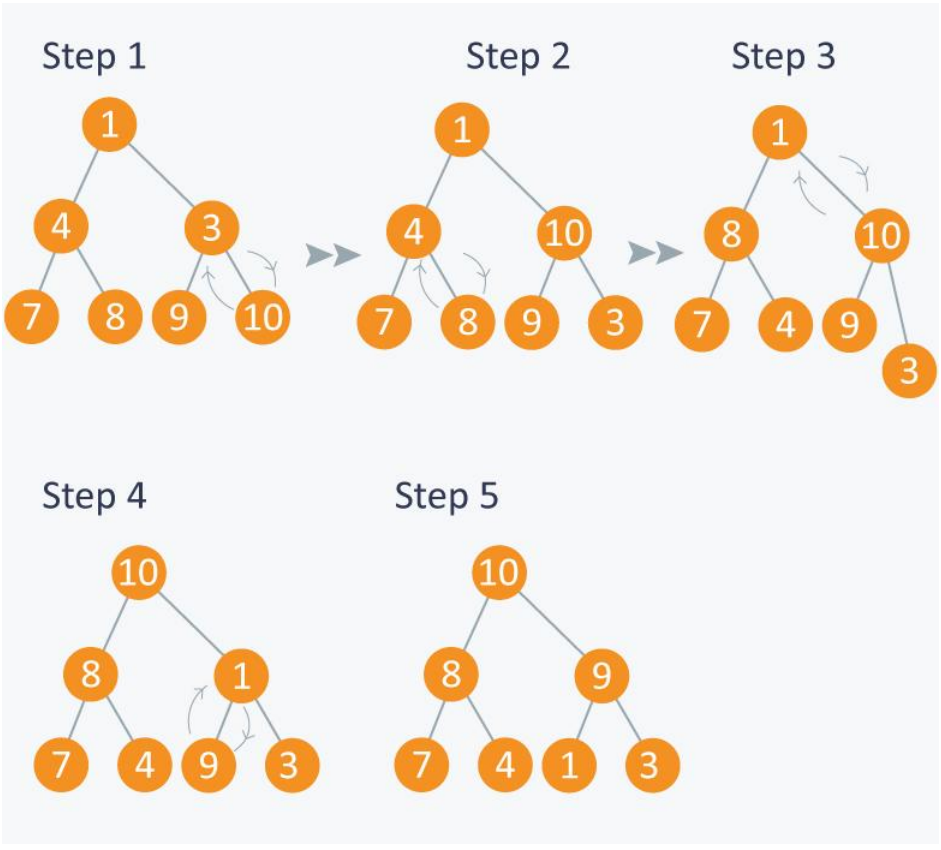
Q. Construct an AVL tree by inserting the following elements in the given order.

63, 9, 19, 27, 18, 108, 99, 81



Heap (must have to complete binary tree)





Huffman Coding