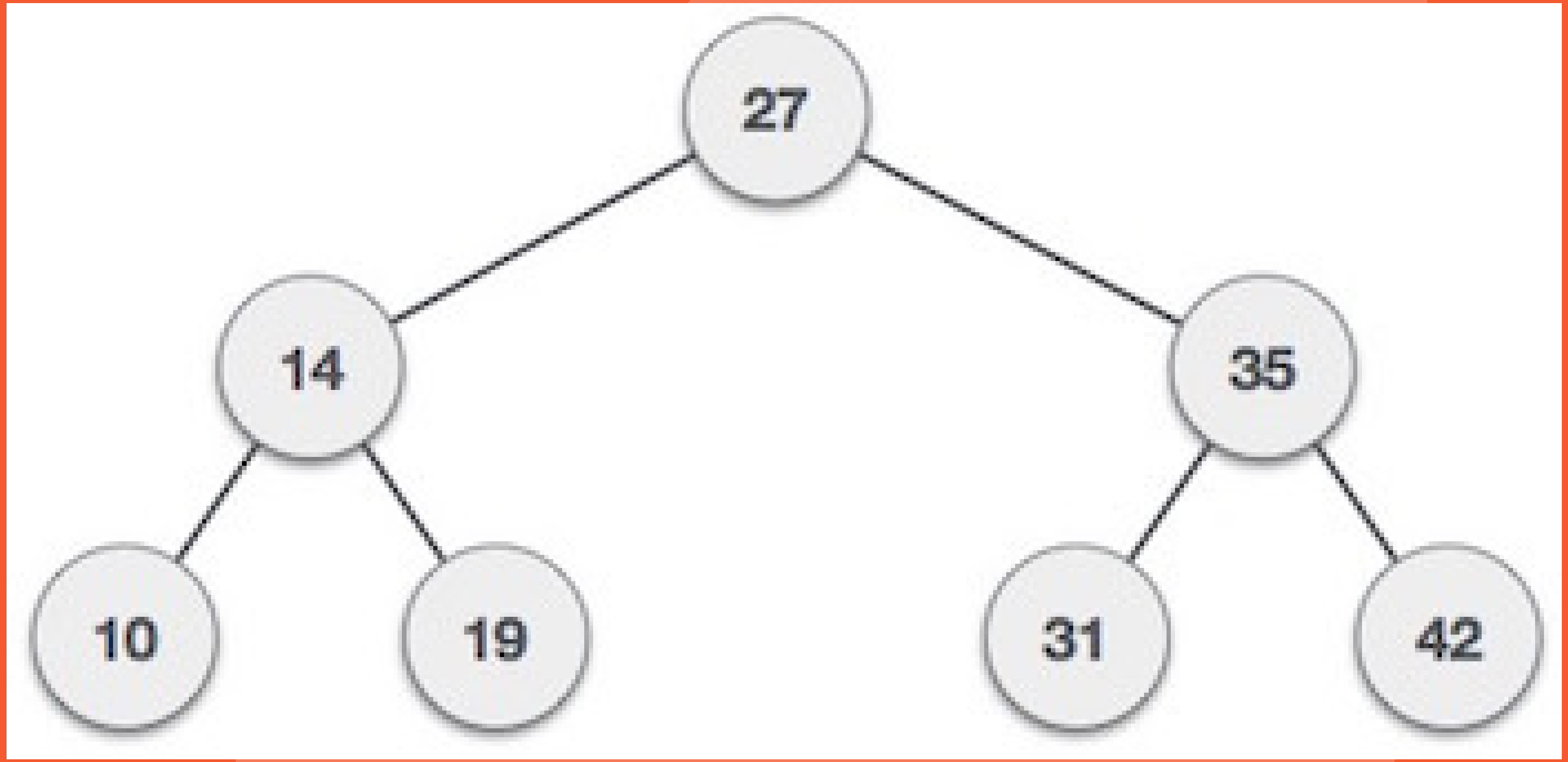
What is a Binary Search Tree?

A Binary Search Tree (BST) is a tree in which all the nodes follow the below-mentioned properties –

- The value of the key of the left sub-tree is less than the value of its parent (root) node's key.
- The value of the key of the right sub-tree is greater than or equal to the value of its parent (root) node's key.
- The left and right sub-tree each must be a binary search tree.
- There must be no duplicate nodes.

Pictorial representation of BST



Search in BST

Given a key we can search in BST as follows: -

- Compare the key with the root node.
- If they are equal return the root node as location of key.
- Else if key is greater than root, repeat the procedure on right child
- Else if key is lesser than root, repeat the procedure on left child

Insertion in BST

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

The worst case time complexity of search and insert operations is O(h) where h is height of Binary Search Tree.

In worst case, we may have to travel from root to the deepest leaf node. The height of a skewed tree may become n and the time complexity of search and insert operation may become O(n).

@AdishiSood

Deletion from Binary Search tree

If a node is to be deleted from BST, there are following situations: -

- **Node to be deleted is leaf node**: Simply remove the node from the tree.
- **Node to be deleted has only one child**: Remove the node and replace it with its child.
- **Node to be deleted has two children**: Find inorder successor of the node. Copy contents of the inorder successor to the node and delete the inorder successor. Note that inorder predecessor can also be used.

The important thing to note is, inorder successor is needed only when right child is not empty. In this particular case, inorder successor can be obtained by finding the minimum value in right child of the node.

The worst case time complexity of delete operation is O(h) where h is height of Binary Search Tree.