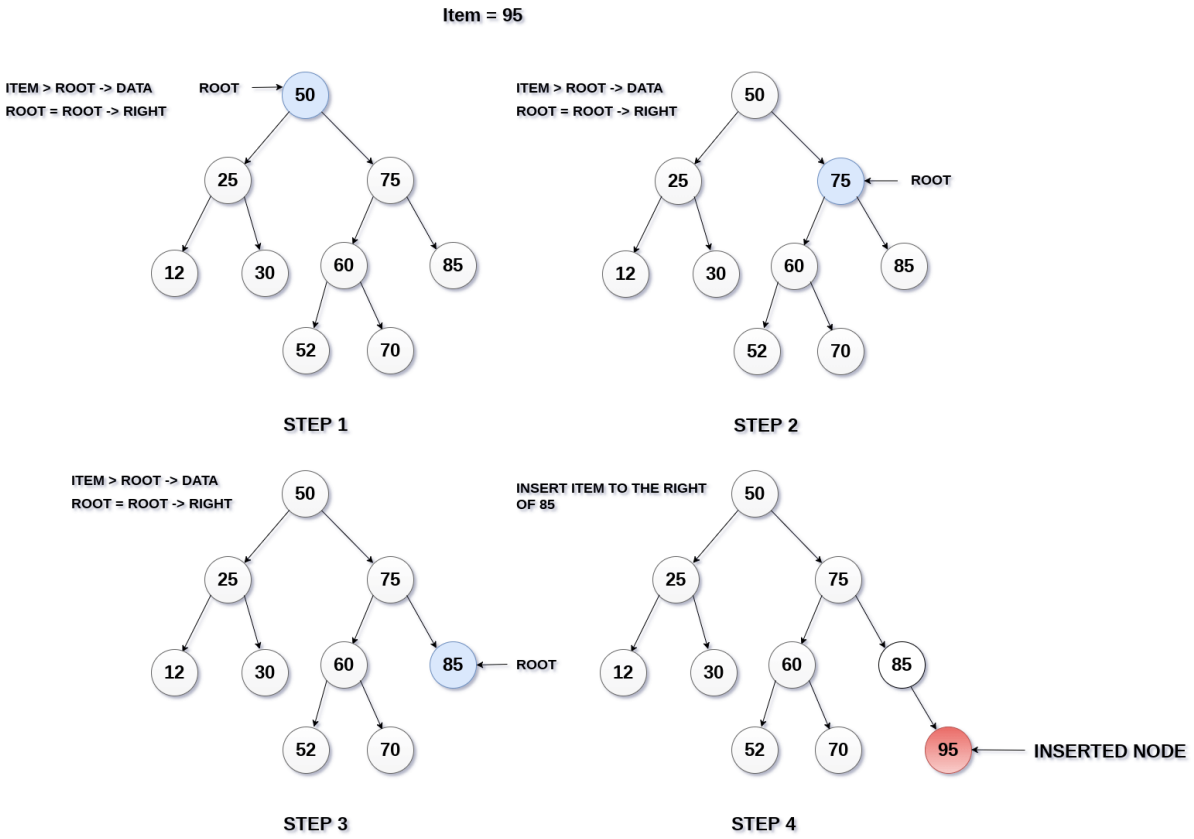
Algorithm for Insertion in a Binary Search Tree



Must node violate the property of binary search tree at each value.

1. Allocate the memory for tree.
2. Set the data part to the value and set the left and right pointer of tree, point to NULL.
3. If the item to be inserted, will be the first element of the tree, then the left and right of this node will point to NULL.
4. Else, check if the item is less than the root element of the tree, if this is true, then recursively perform this operation with the left of the root.
5. If this is false, then perform this operation recursively with the right sub-tree of the root.

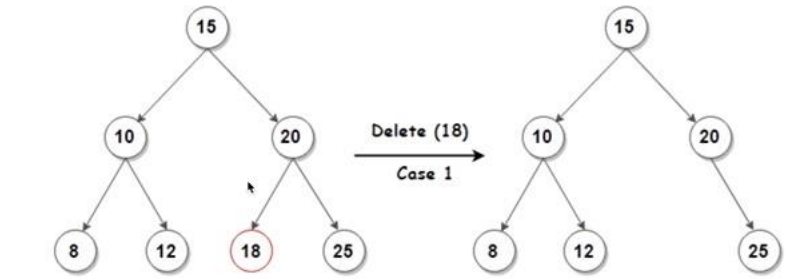
The pseudocode for insertion in a binary search tree:

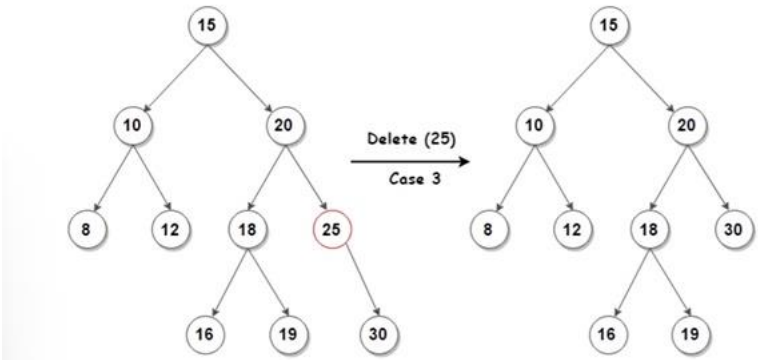
- **Step 1:** IF TREE = NULL
  Allocate memory for TREE
  SET TREE -> DATA = ITEM
  SET TREE -> LEFT = TREE -> RIGHT = NULL

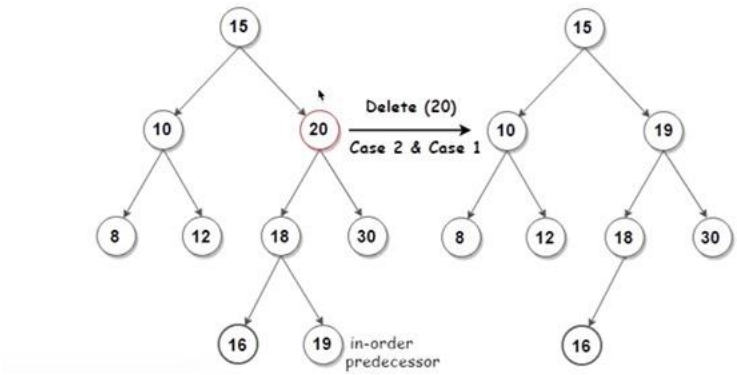
```
    ELSE
     IF ITEM < TREE -> DATA
      Insert(TREE -> LEFT, ITEM)
    ELSE
     Insert(TREE -> RIGHT, ITEM)
    [END OF IF]
    [END OF IF]
```

- **Step 2:** END

Algorithm for Deletion in a Binary Search Tree







1. Start at the root of the binary search tree.
2. If the root is NULL, return NULL.
3. If the value to be deleted is less than the root's data, recursively call the delete function/method on the left subtree.
4. If the value to be deleted is greater than the root's data, recursively call the delete function/method on the right subtree.

5. If the value to be deleted is equal to the root's data, check if the node to be deleted has one child or no child.
6. If the node has only one child or no child, free the memory allocated to the node and return the appropriate child (if it exists)
7. If the node to be deleted has two children, find the minimum value in the right subtree (i.e., the leftmost node in the right subtree). Copy the data in the minimum node to the node to be deleted, and recursively call the delete function/method on the right subtree to delete the minimum node.
8. Return the root of the updated binary search tree.

The pseudocode for deletion in a binary search tree:

```
function deleteNode(root, value):
    if root is NULL:
        return root

    if value < root->data:
        root->left = deleteNode(root->left, value)
    else if value > root->data:
        root->right = deleteNode(root->right, value)
    else:
        // Case 1: Node to be deleted has only one child or no child
        if root->left is NULL:
            temp = root->right
            free(root)
            return temp
        else if root->right is NULL:
            temp = root->left
            free(root)
            return temp

        // Case 2: Node to be deleted has two children
        temp = findMinNode(root->right)
        root->data = temp->data
        root->right = deleteNode(root->right, temp->data)

    return root
```