450. Delete Node in a BST

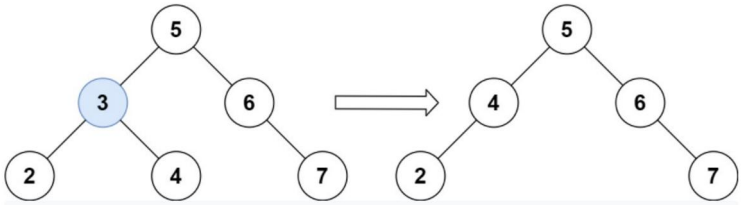
01 April 2022 08:56 PM

Given a root node reference of a BST and a key, delete the node with the given key in the BST. Return the root node reference (possibly updated) of the BST.

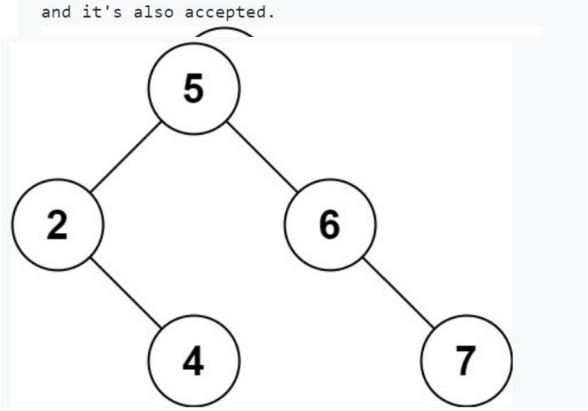
Basically, the deletion can be divided into two stages:

1. Search for a node to remove.
2. If the node is found, delete the node.

Example 1:




```
Input: root = [5,3,6,2,4,null,7], key = 3
Output: [5,4,6,2,null,null,7]
Explanation: Given key to delete is 3. So we find the node with
value 3 and delete it.
One valid answer is [5,4,6,2,null,null,7], shown in the above BST.
Please notice that another valid answer is [5,2,6,null,4,null,7]
and it's also accepted.
```

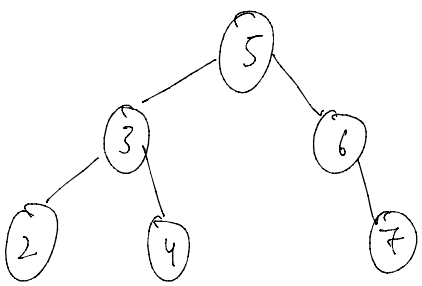


Example 2:

```
Input: root = [5,3,6,2,4,null,7], key = 0
Output: [5,3,6,2,4,null,7]
Explanation: The tree does not contain a node with value = 0.
```

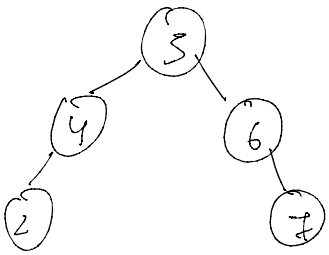
Example 3:

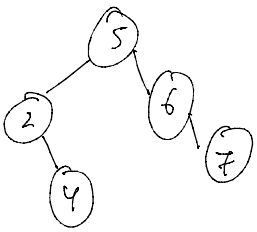
```
Input: root = [], key = 0
Output: []
```



node = 3

o/p

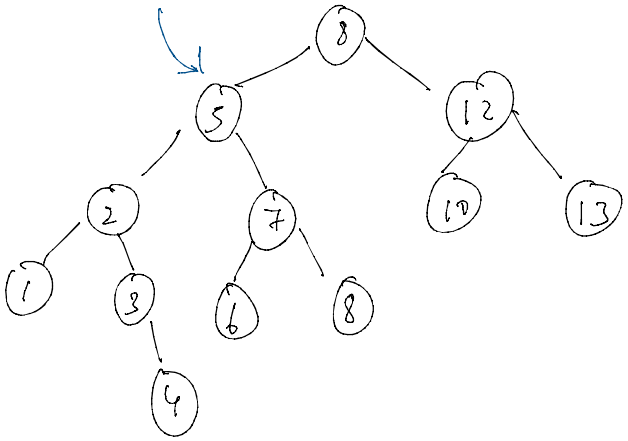


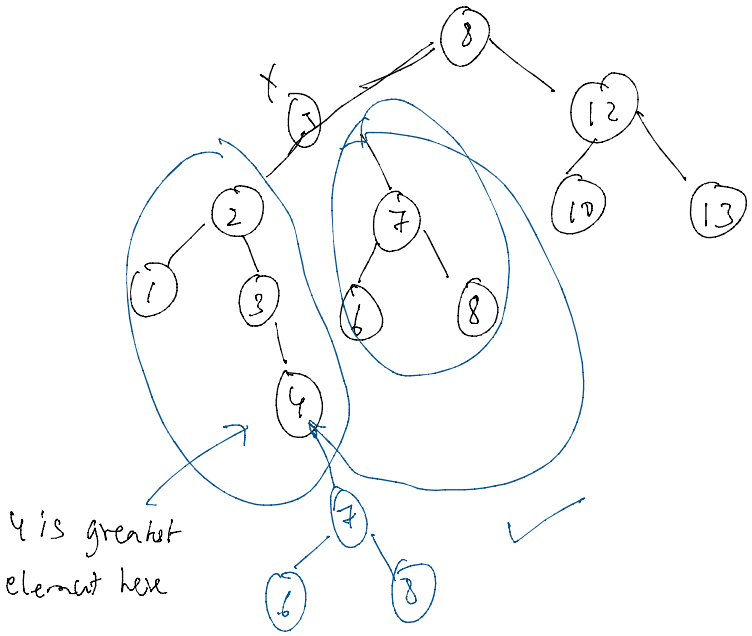


* Search node if it there or not

* Deleting

node = 5



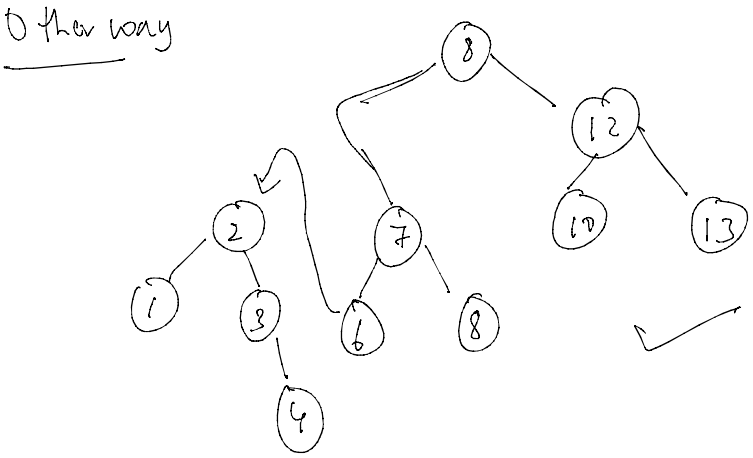


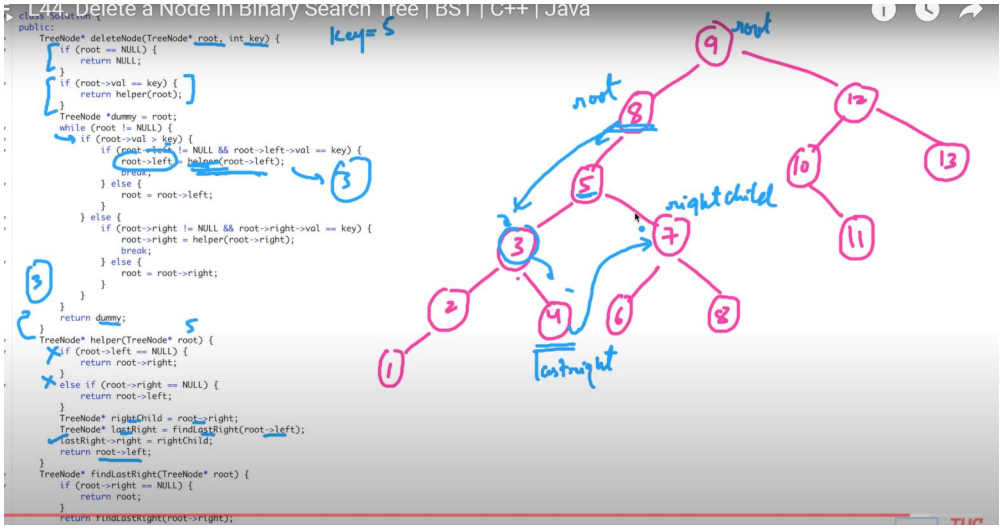
8 → left = 5 → left

link will attach

→ go to extr right of 2 which is 4 and link the 7.

Other way






```
class Solution {
    public TreeNode deleteNode(TreeNode root, int key) {
        if (root == null) {
            return null;
        }
        if (root.val == key) {
            return helper(root);
        }
        TreeNode dummy = root;
        while (root != null) {
            if (root.val > key) {
                if (root.left != null && root.left.val == key) {
                    root.left = helper(root.left);
                    break;
                } else {
                    root = root.left;
                }
```

```
            } else {
                if (root.right != null && root.right.val == key) {
                    root.right = helper(root.right);
                    break;
                } else {
                    root = root.right;
                }
            }
        }
        return dummy;
    }

    public TreeNode helper(TreeNode root) {
            if (root.left == null) {
                return root.right;
            } else if (root.right == null){
                return root.left;
            } else {
                TreeNode rightChild = root.right;
                TreeNode lastRight = findLastRight(root.left);
                lastRight.right = rightChild;
                return root.left;
            }
    }

    public TreeNode findLastRight(TreeNode root) {
        if (root.right == null) {
            return root;
        }
        return findLastRight(root.right);
    }
}
```