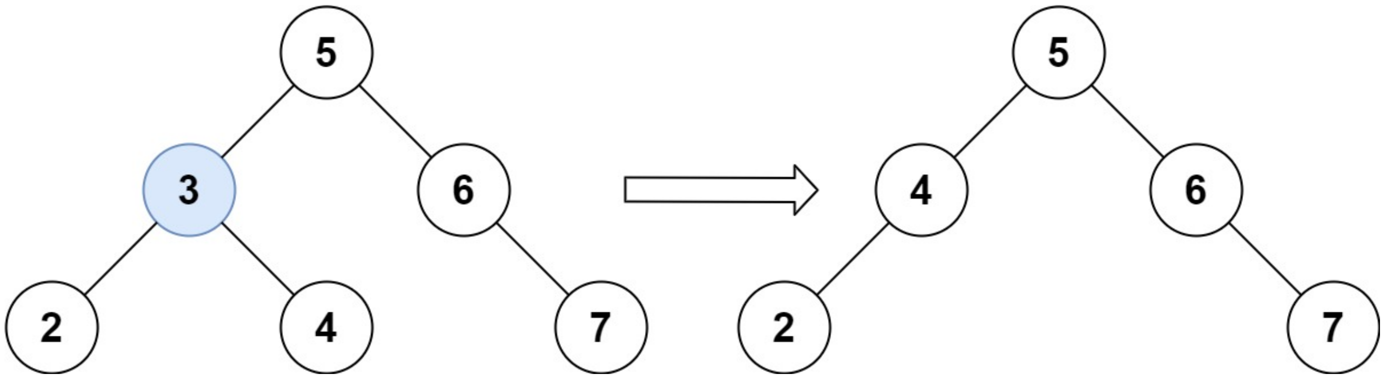
450. Delete Node in a BST

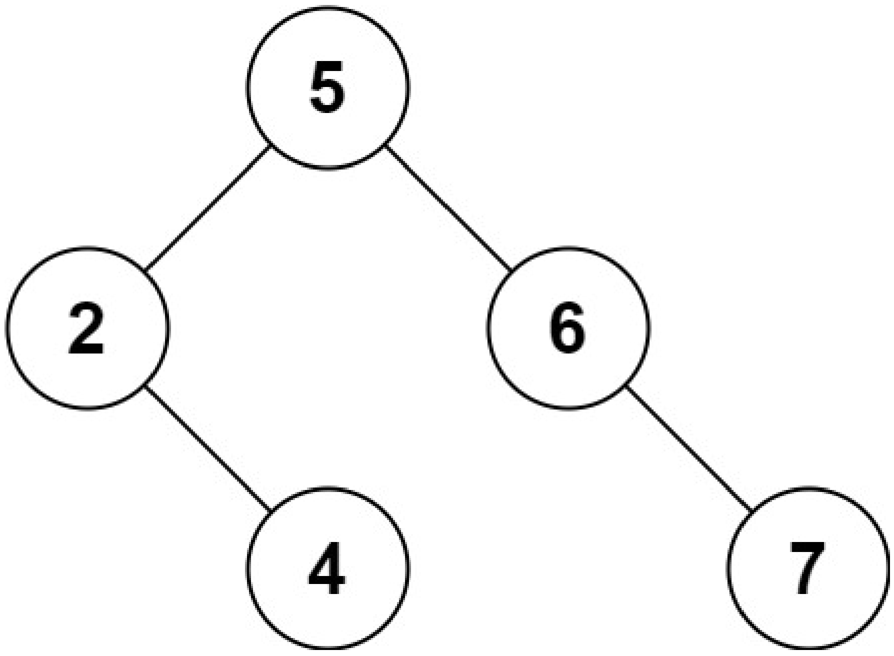
Given a root node reference of a BST and a key, delete the node with the given key in the BST. Return the root node reference (possibly updated) of the BST.

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.



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: []

```
def deleteNode(self, root: TreeNode, key: int) -> TreeNode:
        if root is None:
            return
        root = self.helper(root,key)
        return root



    def helper(self,root,key):
        if root is None:
            return
        if root.val>key:
            root.left = self.helper(root.left,key)
        elif root.val<key:
            root.right = self.helper(root.right,key)
        else:
            if root.left is not None and root.right is not None:
                temp = root.right
                while temp.left: temp = temp.left
                root.val = temp.val
                root.right = self.helper(root.right, root.val)
                return root
            elif root.left!=None:
                return root.left
            elif root.right!=None:
                return root.right
            else:
                return None
        return root





    def findMax(self,root):
        if root is None:
            return 0
        lt = self.findMax(root.left)
        rt = self.findMax(root.right)
        return max(lt,rt,root.val)
```