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.

Follow up: Can you solve it with time complexity `O(height of tree)`?

```
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
```

Three conditions are:

1. when node is leaf node==>return None

2. when node has either left or right child===>return left or right child accordingly

3. when node has both left or right child===> find the minimum from the right subtree and just replcae the node value with this minimum value and

just delete the node with min value in right sub-tree.