LEETCODE: 450

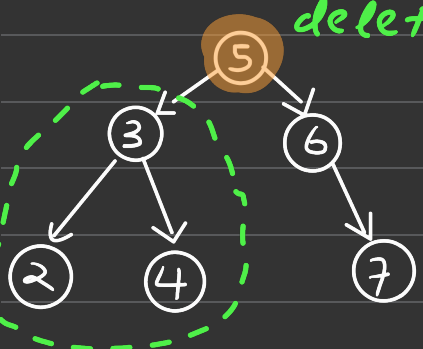
By

Priyansh

PROBLEM:

We are provided with A BST And a key value that is present in Any of Node of the Bst and we have to delete that node maintaing defination of BST...

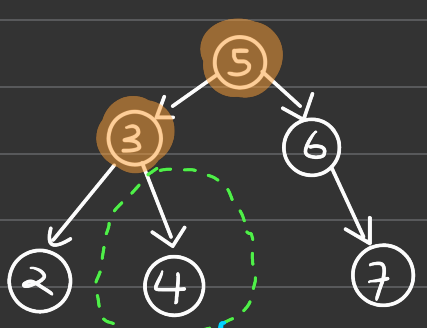
Example: Lets begin with the testcase of deleting a leafnode!!!



target = 4

4<5 move left And rearrange left Subtree:-

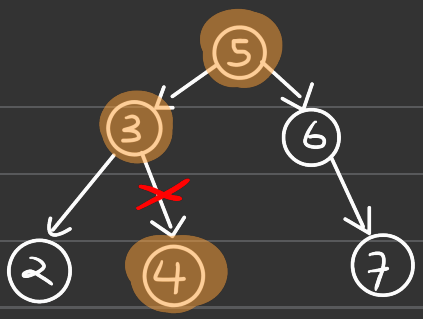
using recursion we would try to rearrange left subtree!!!



target = 4

4>3 move Right And rearrang Right Subtree:-

we would try rearranging Right subtree node of current root:: 3



target = 4

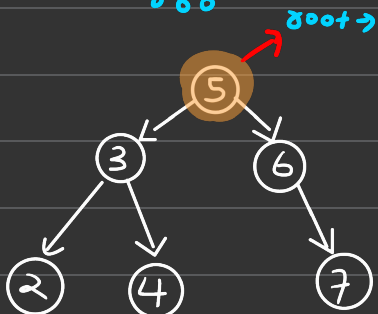
4 == 4

Now perform deletion

Since its a leaf node

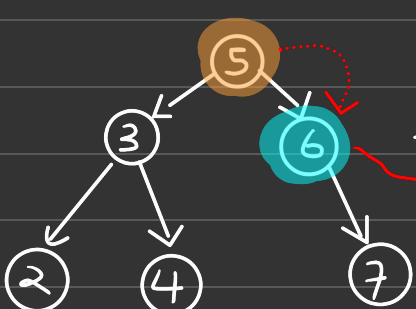
we would return NULL ;)

Case 2) Target Node is having a single child !!!



target == 6

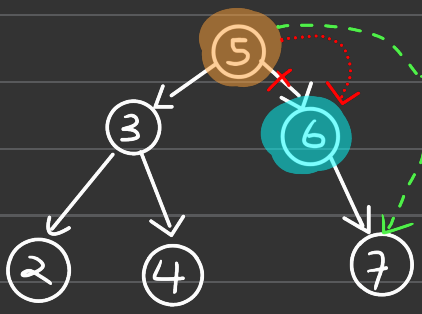
5 < 6 → 5 would Actually ask to rearrange its rightSubtree which could possibly have node with Val == 6



6 will say my val is equal to target Val

or root->Val == target->Val

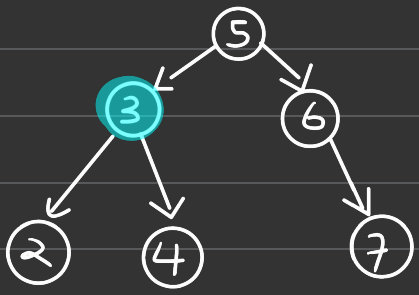
Now it would return its child Node that is not NULL to its calling function (It would be an updated connection)



New connection formed

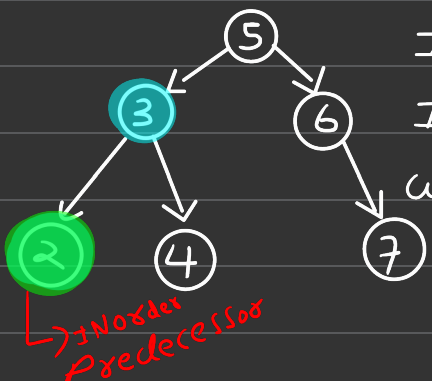
As 5 node was asking its right subtree to rearrange it self!!!

MOST IMPORTANT CASE:



when target node is parent of two child nodes!!!

LETS PROCEED with deletion of node 3...



Important Observation:

If we have to delete a parent node with two children we have to find its suitable replacement And suitable replacement would be either Inorder Sucesor or Inorder Predecessor

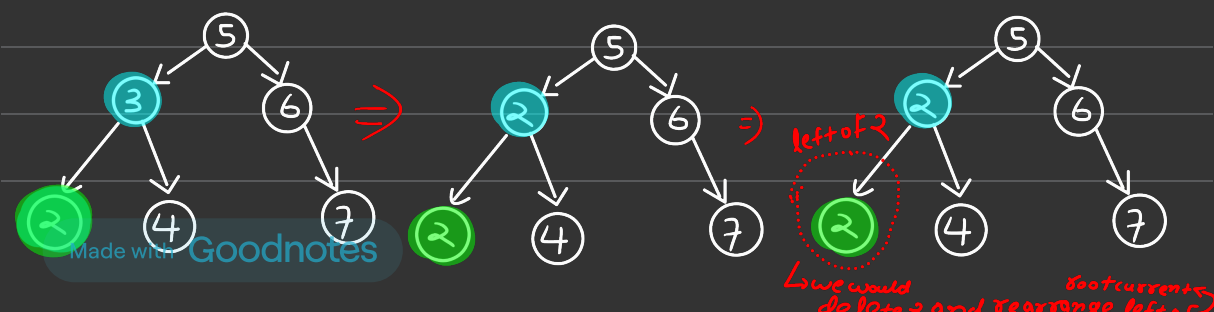
*We would use in_order predecessor so to find replacement of the node to be deleted we would firstly find out its in_order predecessor by a function:-

```
class Solution {
public:
    TreeNode* in_predecessor(TreeNode* root){
        TreeNode* pred = root->left;
        while(pred->right){
            pred = pred->right;
        }
        return pred;
    }
```

} function to find Inorder Predecessor

Now we would update current root's value to the value of its in_order predecessor now we would call the function delete_node to delete that in_order predecessor Node... (Method works Always As in_order predecessor node would Always be parent of Single child or will be a leaf Node)

Most Important case



Code Implementation:

```
TreeNode* deleteNode(TreeNode* root, int key) {
    if(root == NULL) return NULL;
    if(root->val == key){
        if(root->left == NULL && root->right == NULL)return NULL;
        if(root->left == NULL || root->right == NULL){
            if(root->left != NULL)return root->left;
            else return root->right;
        }
        if(root->left != NULL && root->right != NULL){
            TreeNode* pred = in_predecessor(root);
            root->val = pred->val;
            root->left = deleteNode(root->left,pred->val);
        }
    }
    else if(root->val > key){
        //move left
        root->left = deleteNode(root->left,key);
    }
    else{
        //move right
        root->right = deleteNode(root->right,key);
    }
    return root;
}
};
```

```
TreeNode* deleteNode(TreeNode* root, int key) {
    if(root == NULL) return NULL;
    if(root->val == key){
        if(root->left == NULL && root->right == NULL)return NULL;
        if(root->left == NULL || root->right == NULL){
            if(root->left != NULL)return root->left;
            else return root->right;
        }
        if(root->left != NULL && root->right != NULL){
            TreeNode* pred = in_predecessor(root);
            root->val = pred->val;
            root->left = deleteNode(root->left,pred->val);
        }
    }
    else if(root->val > key){
        //move left
        root->left = deleteNode(root->left,key);
    }
    else{
        //move right
        root->right = deleteNode(root->right,key);
    }
    return root;
}
};
```

Base

When root->val == key

root == leaf

root is parent of single child

when root is parent of two node

finding replacement

Editing value

After editing value of root

Now lets delete the predecessor

{Go to root->left, rearrange and Attach updated node}

{Go to root->right rearrange and Attach updated node}

finally return root;

LEETCODE:450