BST, July 31

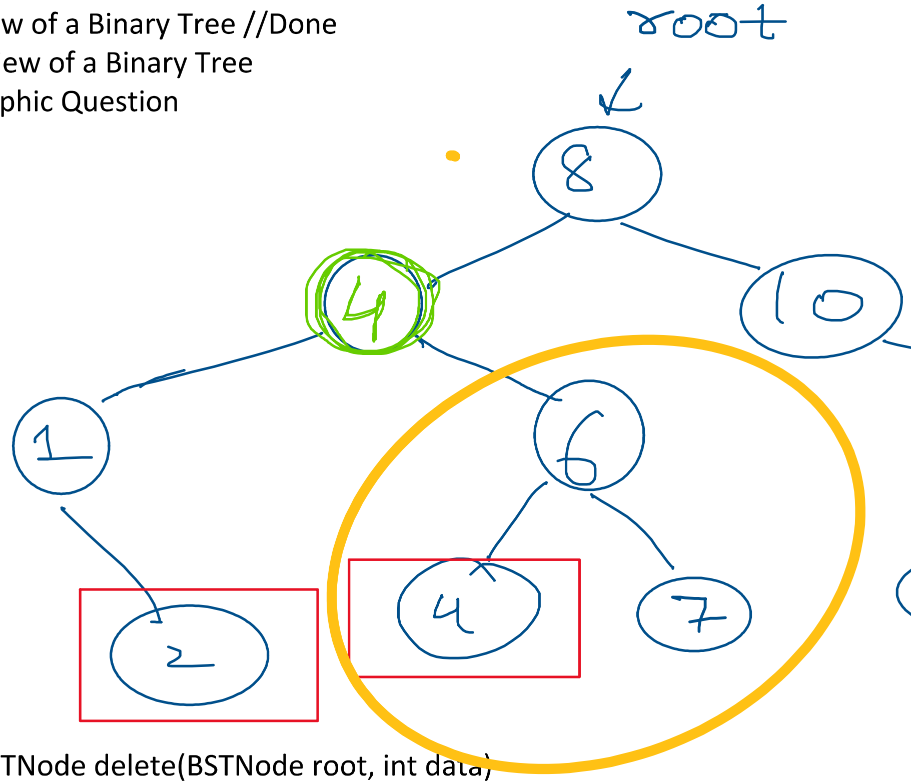
31 July 2020 18:52

Tasks:

1. Delete a Node from BST //Done
2. Lowest Common Ancestor in Binary Tree //Done
3. Lowest Common Ancestor in Binary Search Tree //Done
4. Left view of a Binary Tree //Done
5. Right view of a Binary Tree
6. IsoMorphic Question

Step 1 = Search the node to be deleted

Step 2 = Delete it according to the case.



Cases

1 -> Delete a leaf

Node with no children

2 -> Delete a Node with 1 child

3 -> Delete a Node with 2 children

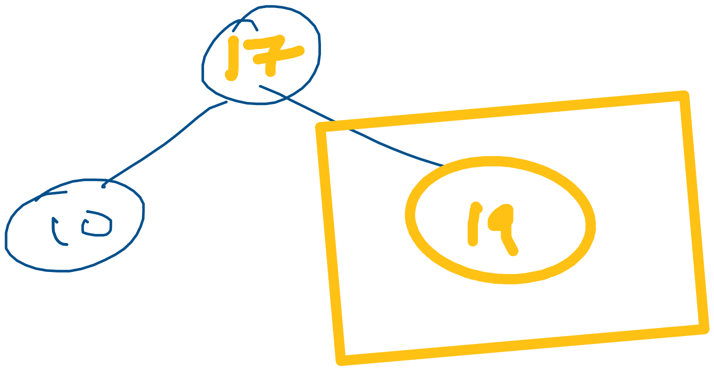
```
BSTNode delete(BSTNode root, int data)
{
    if(root==null) return null;
    if(data<root.data)
    {
        root.left = delete(root.left,data); //Left  side se delete kro aur left side ko update krdo
    }
    else if(data>root.data)
    {
            root.right = delete(root.right,data); //Right side se delete kro aur Right side ko update krdo
        }
        else{
            //We have found the node to be deleted
            if(root.left == null) //1Child Case && No Child Case
            {
                return root.right;
            }
            if(root.right == null)
            {
                return root.left;
            }
            //2 Child Case
            root.data = min(root.right);
            root.right = delete(root.right,root.data);
        }
        return root;
    }
```

// Leaf Node case also covered.

2 Cases for Node with 2children

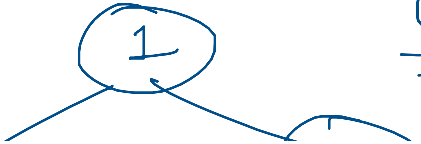
1 → Pick min from R.S.T

2 → Pick max from L.S.T



Lowest Common Ancestor in Binary Tree

(2,2)



(4, 7)

(4,5)

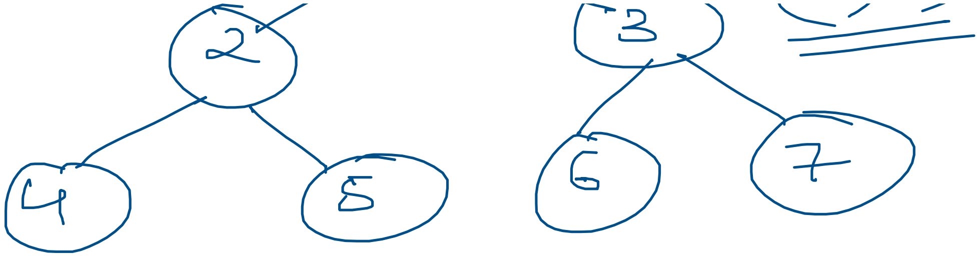
(n1, n2)

(4,5) → 2

(4,6) → 1

(1,7) → 1

(3,7) → 3



(3, 4) → 1

(2, 4) → 2

(4, 7) → 1

(2, 7) → 1

(2, 5) → 2

Condition for ancestor
↓

if(root.data == n1 ||
root.data == n2) return root;

Assuming both Nodes exist in the Binary Tree

```
Node findLCA(Node root, int n1,int n2)
{
    if(root==null) return null;
    if(root.data==n1 || root.data==n2) return root;

    //If one node lies on left side and the other lies on
    right side then current root is LCA

    Node leftLCA = findLCA(root.left,n1,n2);
    Node rightLCA = findLCA(root.right,n1,n2);
    if(leftLCA!=null && rightLCA!=null)
    {
        return root;
    }
```

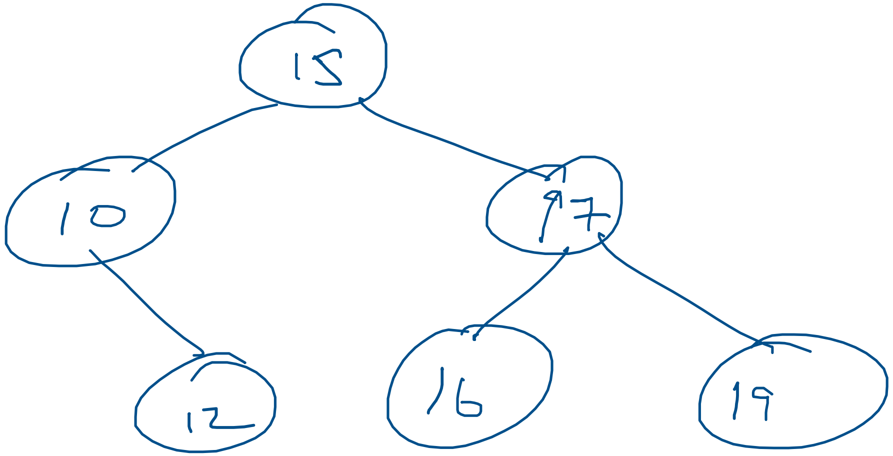
```
        if(leftLCA!=null) return leftLCA; //agar sirf left se aya
        to vahi answer hai

        return rightLCA; //agar sirf right se aya to vahi answer
        hai
    }
```

This will work for all Types of Binary Tree including BST.

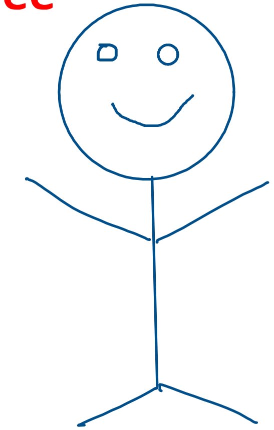
But we can do better for a BST.

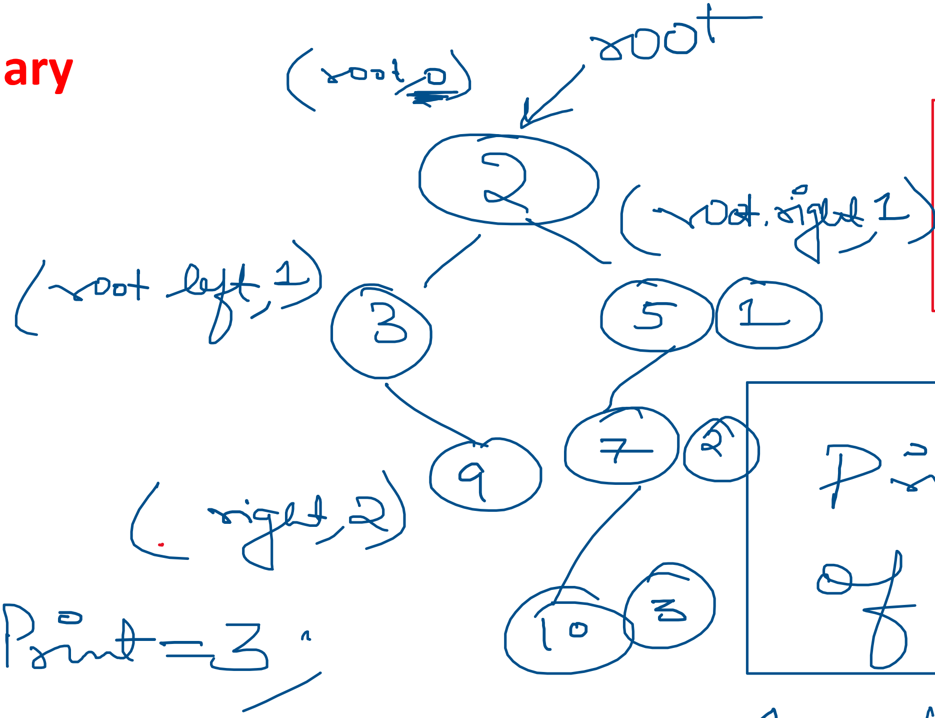
```
BSTNode lcaBST(BSTNode root, int n1,int n2)
{
    if(root==null) return null;
    if(n1<root.data && n2<root.data)
    {
        Return lcaBST(root.left,n1,n2);
    }
    if(n1>root.data && n2>root.data)
    {
        return lcaBST(root.right,n1,n2);
    }
    //left is the case of being equal to n1 or n2 or one
    being on left side and the other being on right side, in
    that case root will be LCA
```



```
        return root;
    }
```

Left view of a Binary Tree





int levelToPrint = 3;

Left View

2 3 9

Print 1st element of all levels

1 -> using BFS

2 -> using DFS

2 3 9 10