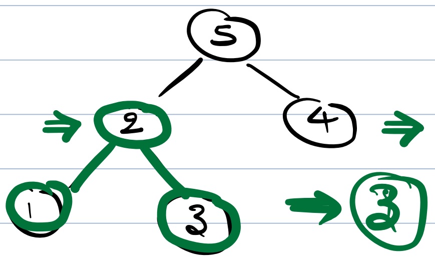
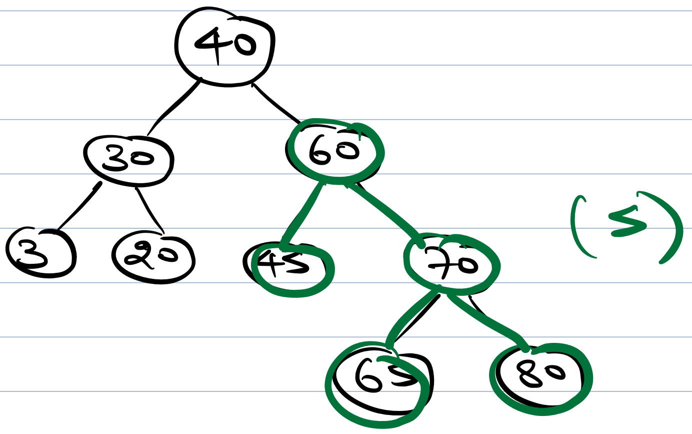
Q// Given a BT. Return the size (no. of nodes) of the max BST subtree in the BT.





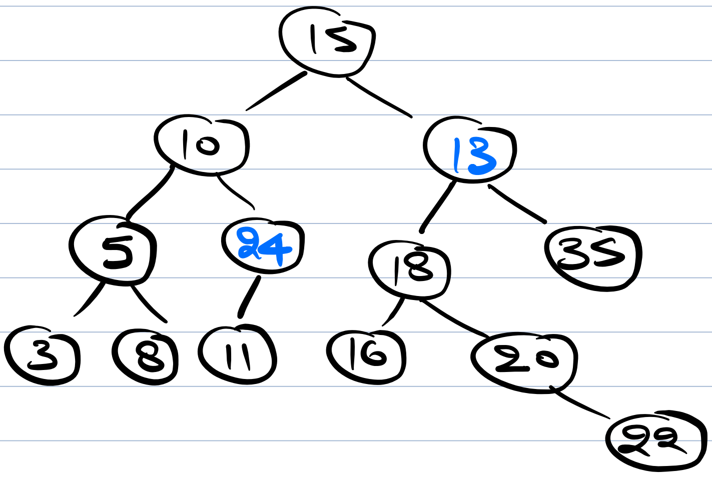
Code

```
class TreeInfo {
        int min;
        int max;
        boolean isBST;
        int count;
}

ans = 0;
```

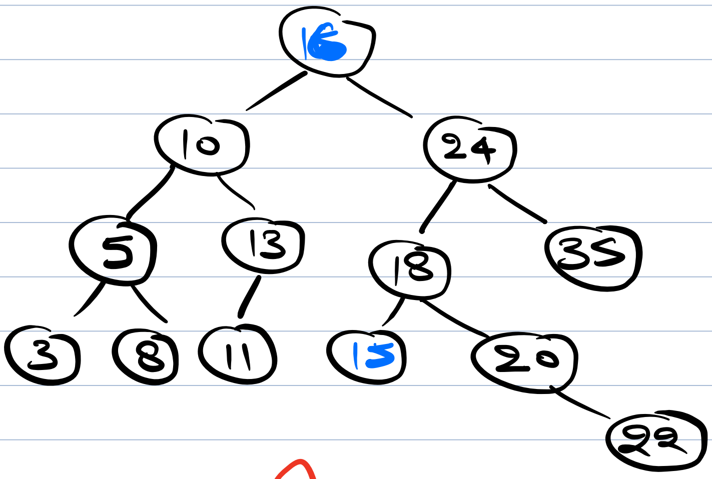
```
TreeInfo   checkBST ( root ) {
    if (root == NULL) {
            return  new  TreeInfo ( INT_MAX,
                          INT_MIN,  T ; 0);
    }

    TreeInfo  left  =  checkBST (root. left);
    Tree Info  right  =  check  BST  (root. right);

    if ((left. isBST == True) && (right. isBST == True)
      && (root.value >= left. max) &&
                  (root. value  < right. min)) {
        ans = max(ans,  left. count + right. count+1);
        return  new  TreeInfo (min (root. value,
          left. min),  max ( root. value, right. max,
             True , (left. count+ right. count +1);
    }

    return  new  TreeInfo ( ∞ , -∞
                              False , 0);
}
```

$T.C. = O(\underline{N})$

Q Given a BST.
2 nodes of this BST are swapped.
fix it ⇒ find those 2 nodes.



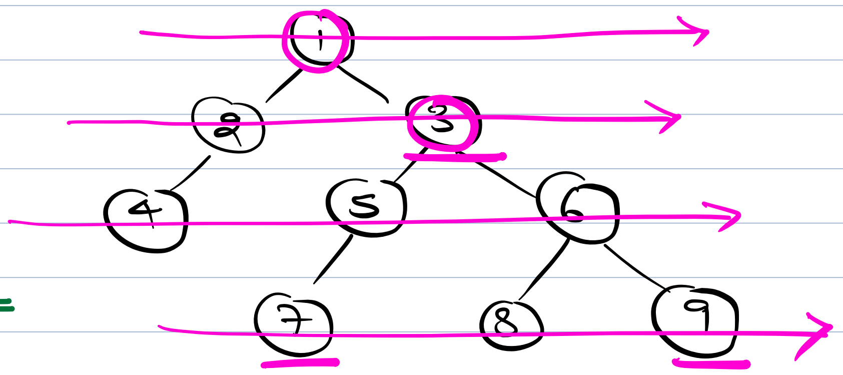
3, 5, 8, 10, 11, 24, 15, 16, 18, 20, 22, 13, 35



3, 5, 8, 10, 11, 13, 16, 15, 18, 20, 22, 24, 35

Ans

LCA (Lowest Common Ancestor)



(7, 3) = 3

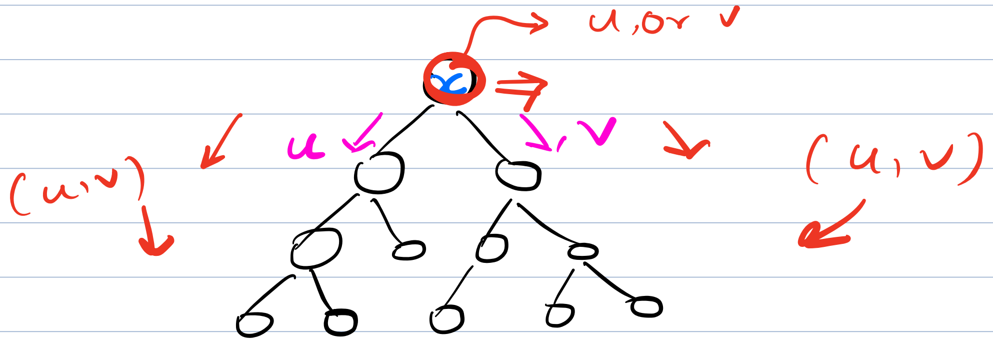
9 ⇒ 1, 3, 6, 9

7 ⇒ 1, 3, 5, 7

1, 3 — Ans

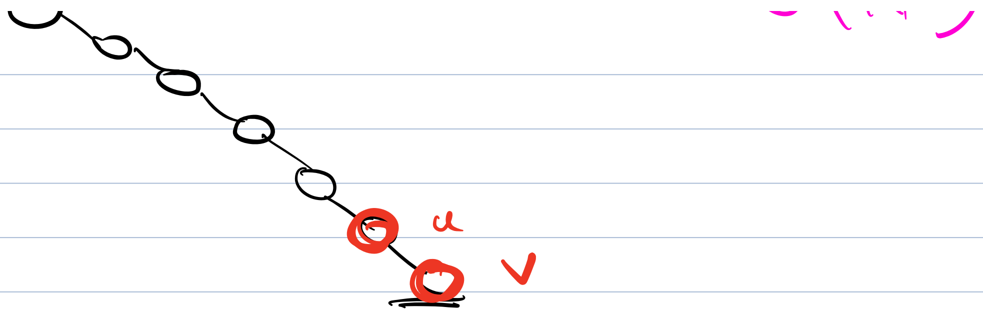
Q Given a BT & two nodes u & v.

Find the LCA (u, v)



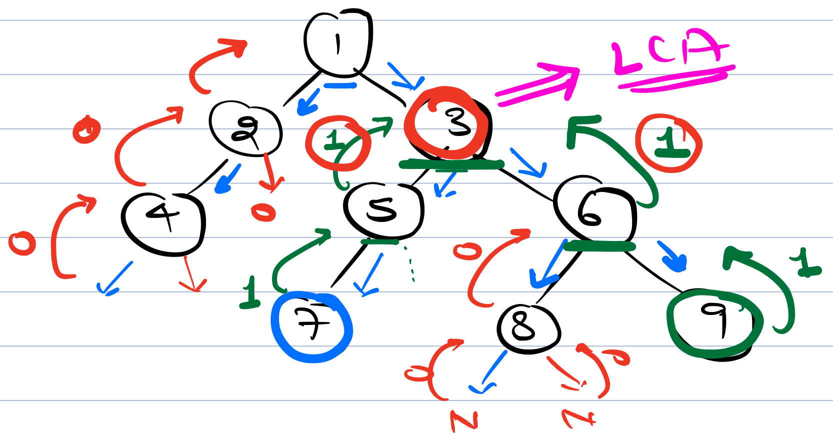
Pre Order

T.C. = O(N²)



Break (10:40)

7, 9



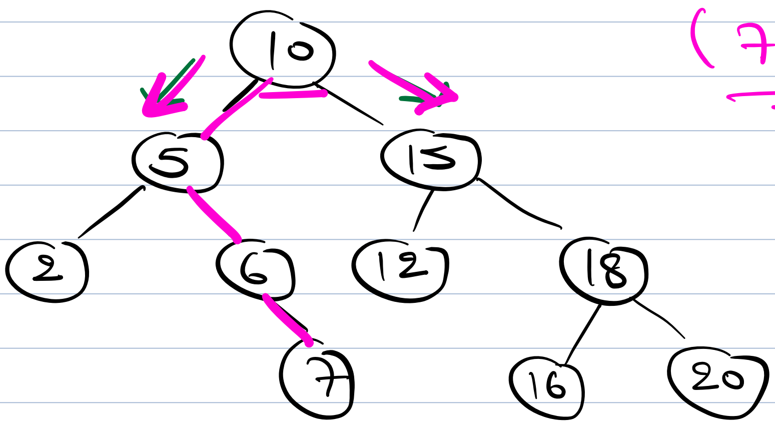
T.C. = O(N) ⇒ Post Order

1) 1 in LST, 1 in RST

2) 1 = node, 2nd in either left or right

Q. Given a BST.

LCA of 2 nodes in this BST.



(7, 16)

$$T.C. = O(H) \begin{cases} \log N \\ \boxed{N} \end{cases}$$

Doubt

lca = ∞

```
boolean findLCA (root) {
    if (root == NULL) { return
                false ;
    }

    boolean left = findLCA(root.left);
    boolean right = findLCA(root.right);

    if (left && right) {
        lca = root.value;
```

```
            return false;
    }

    if ((root.value == u) ||
                (root.value == v)){

        if ( left || right) {
            lca = root.value;
            return false;
        }
        else {
            return true;
        }
    }

    if (left || right) {
        return true;
    }

    return false;
}
```