Welcome ☺

Agenda : Morris Traversal
LCA
2-3 questions

Q Given a BST and a positive integer K, find the K^{th} smallest element.

App: Use inorder traversal and store the elements in an array. Return $(K-1)^{th}$ element

T.C = O(N)
S.C = O(N)

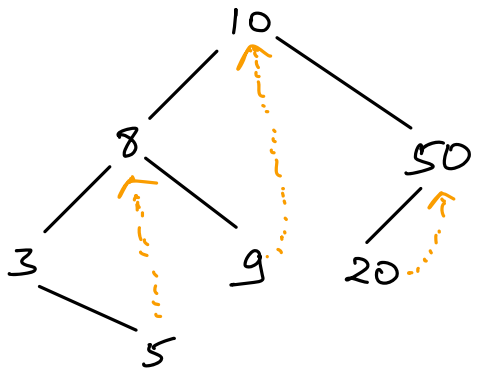
App 2 Keep track of visited nodes. T.C = O(N) S.C = O(1)/O(

Whenever count becomes K, return that value

Code

```
int count = 0
int res = INT_MIN
void inOrder ( root , K)
{
    if( !root) return ;
    inOrder ( root.left , K)
    count ++
    if( count == K)
    {
        res = root.data
        return ;
    }
    if(res == INT_MIN) inOrder ( root.right , K)
}
return res ;
```

Morris Traversal - Inorder LNR



C

in-order predecessor.

↳ rightmost element in Left subtree.

3 5 8 9 10 20 50

Code

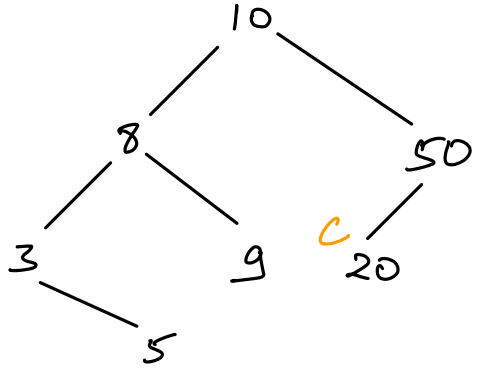
```
curr = root
while ( curr != NULL)
{  if ( ! curr.left )
   {   print( curr.data)
       curr = curr.right)
   }
   else
   {   // Find rightmost
       R = rightmostLST( curr )
       if( R.right == =NULL ) // visiting for first time
       {
           R.right = curr   // temp link
           curr = curr.left
       }
       else  // visiting for 2nd time
       {
           print ( curr.data)
           R.right = NULL   // delete temp link
           curr = curr.right
       }
   }
}
```

```
Node rightmostLST ( root )
{
    temp = root.left
    while ( temp.right != NULL && temp.right != root )
    {
        temp = temp.right
    }
    return temp
}
```

T.C $O(1 + 3N) = O(N)$

S.C $O(1)$

Q Given a value, find path from root to node of binary tree



9 ⇒ 10 → 8 → 9

50 ⇒ 10 → 50

10 ~~8~~ ~~3~~ ~~5~~ ~~9~~ 50 20

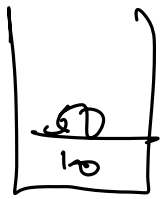
Code

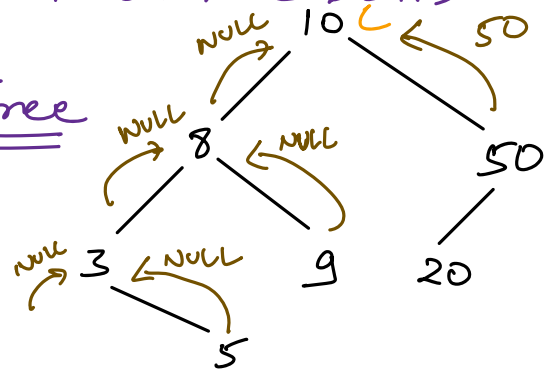
```
bool path ( root, target )
{
    if ( !root ) return false.
    arr.push ( root.data )
```

```
if ( root.data == target )
    return true;

if ( path ( root.left, target ) ||
     path ( root.right, target ))
        return True;

arr.remove() // remove last ele from dynamic array.
return false
}
return arr
```

Lowest Common Ancestor (LCA)

1) Binary Tree





5, 9 ⇒ 8

3, 5 ⇒ 3

50, 20 ⇒

Code

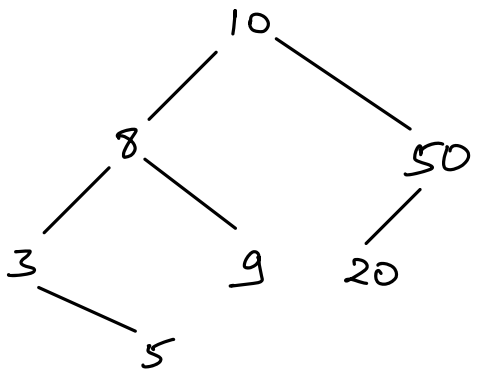
```
LCA ( root , node 1 , node 2)
{
    if ( !root || root ==node 1 || root == node 2)
        return root
    left_lca = LCA ( root.left , node 1, node 2)
    right_LCA = LCA ( root.right , node 1, node 2)
    if ( left_lca && right_LCA )
        return root
```

```
if ( left_lca )
    return left_lca
else
    return right_lca
}
```

T.C ⇒ O(N)

S.C ⇒ O(H)

2) BST



LCA (root , node 1, node 2)

Code

```
{
    if( !root )  return NULL.

    while ( root )
    {
        if ( root.data < node1.data  &&
                root.data. < node2.data )
            root = root.right

        if ( root.data > node1.data  &&
                root.data. > node2.data )
            root = root.left

        else
            return root
    }
    return NULL
}
```

T.C ⇒ O(H)

S.C ⇒ O(1)