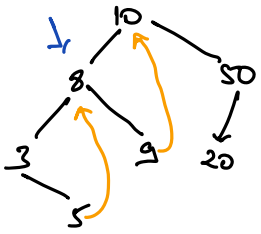
Welcome ☺

Agenda : Morris Inorder
LCA

Morris Inorder traversal



Inorder → LNR

3 5 8 9 10 20 50

T.C ⇒ O(N) S.C ⇒ O(H)

⇒ left subtree.
rightmost element of left subtree is 9.

1) Rightmost element.

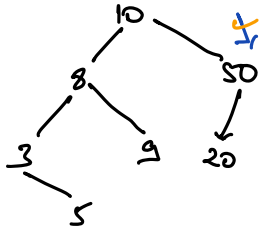
rightmost.right = root

Code

curr = = root

3 5 8 9 10

```
while ( curr != NULL)
{
   if ( curr.left = = NULL)
   {
      print ( curr.data)
      curr = curr.right
   }
   else
   {
      1. Find rightmost
      R = rightmostLST ( curr)
      if ( R.right = = NULL)
      {
         R.right = curr
         curr = curr.left
      }
```

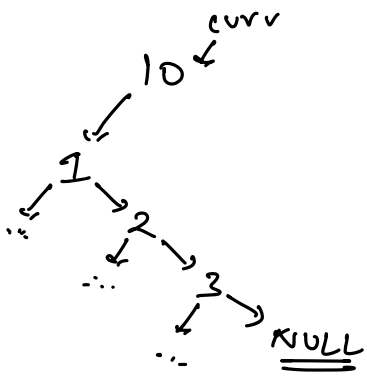


```
        else
        {
            print ( curr.data)
            curr. = curr.right
            R.right = NULL
        }
    }
}
```

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

S.C => O(1)

T.C = O(2N) ≃ O(N)

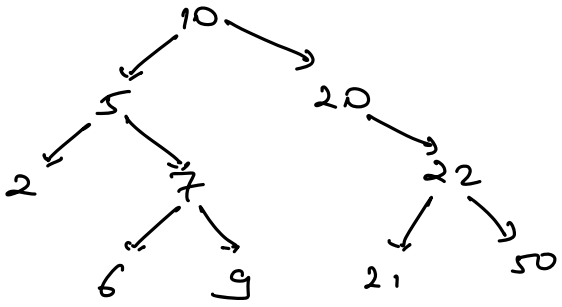


1 2 3

Q Find K^{th} smallest element in BST ?

K = 2 → 5

K = 3 → 21



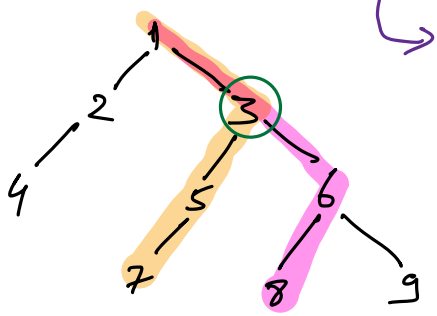
Soln ⇒ K^{th} node in inorder traversal of BST

T.C O(N / ~~K~~) SC → O(1)

Hw K^{th} largest element in BST

Lowest common Ancestor (LCA)

⇒ All the nodes in the path from root node to curr node including curr node.



ancestor (7) ⇒ 1 3 5 7

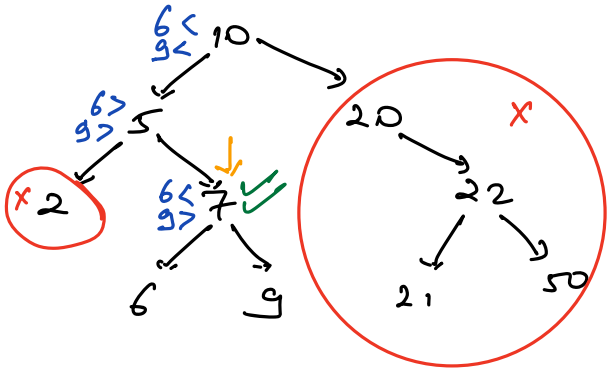
ancestor (8) ⇒ 1 3 6 8

LCA (4, 7) → 1

LCA (2, 4) → 2

LCA (5, 5) → 5

LCA in BST (when both x & y are present in tree)



LCA(6, 9)

Code

```
LCA(x, y)
temp = root
while (temp != NULL)
{
    if(temp.data > x && temp.data > y)
    {
        temp = temp.left
    }
    else if(temp.data < x && temp.data < y)
    {
        temp = temp.right
    }
    else
    {
        return temp
    }
}
```

T.C → O(H)

S.C → O(1)

In Time - Out Time Concept

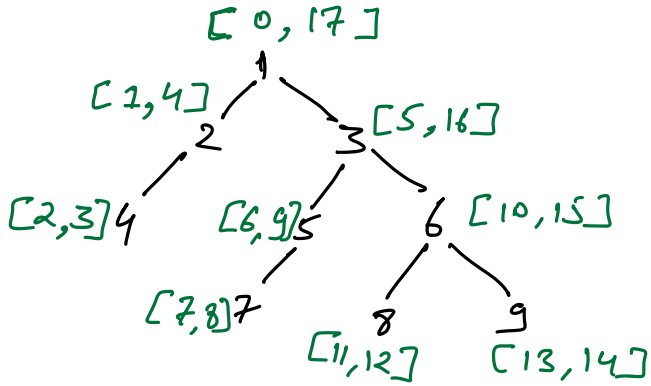
traversal of subtree starts	traversal of subtree ends.

pre → N L R → In Time

in → L N R

post → L R N → Out Time

start Time = 0 [start T, end T]



Code

```
t=0
void travel ( root)
{
   if ( root == NULL)  return ;
   in(root) = t ;        Node
   t++
   travel ( root. left)    left
   travel ( root. right)   Right
   out ( root) = t ;       Node
   t++
}
```

T.C → O(N)

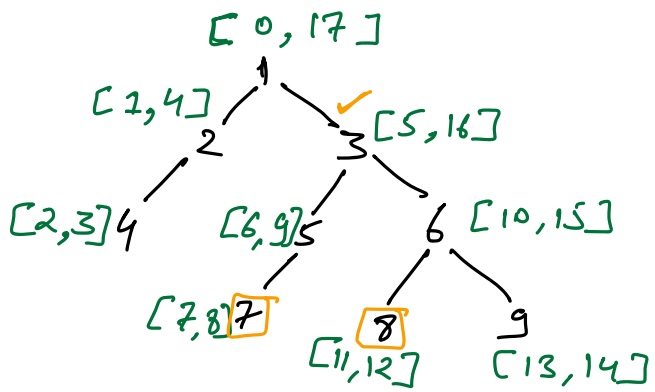
S.C → O(N + H) → map

Nodes x & y

in(x) ≤ in(y)

out(x) ≥ out(y)

⟹ x is ancestor of y



Find LCA(a, b) in B.T

Code

```
if ( root.left is ancestor of both a & b)
    root = root.left
else if ( root.right is ancestor of both a & b)
    root = root.right
else
    return root
```

to find in & out

T.C → (N + H)

S.C → O(N) → store in & out