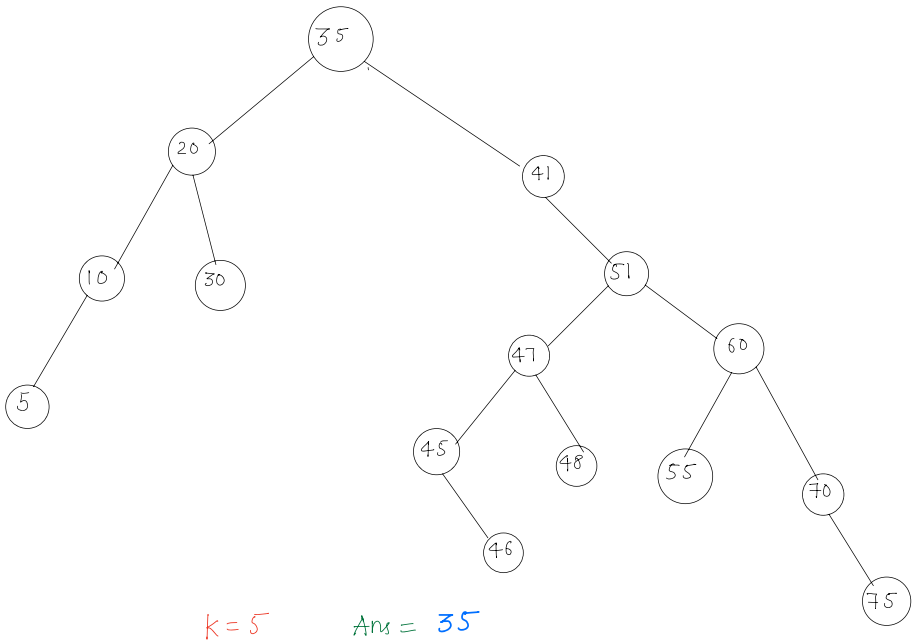
Lecture :- Trees - 4

Agenda

- Kth smallest element in BST
- Morris inorder traversal
- Root to node path
- LCA of BT and BST
- in and out time
- LCA of multiple queries.

Qu find kth smallest element in binary search tree.



K = 5 Ans = 35

Brute force:-

Inorder = sorted array.

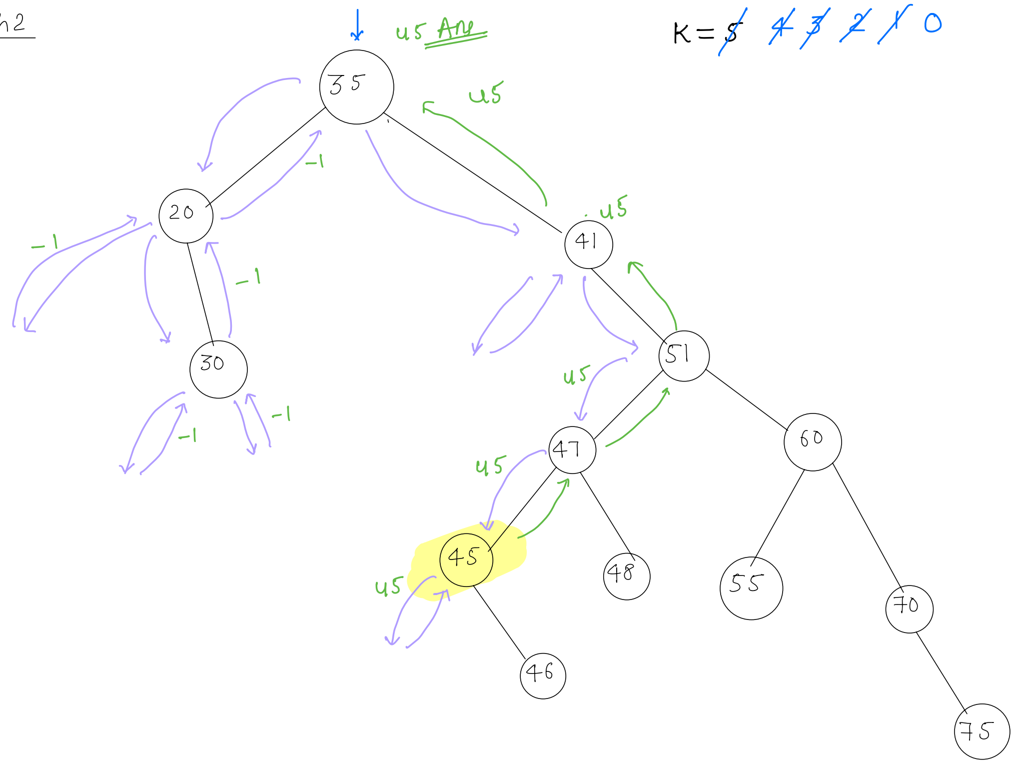
↓

kth element = ans.

TC: O(n)

SC: O(n)

Approach 2



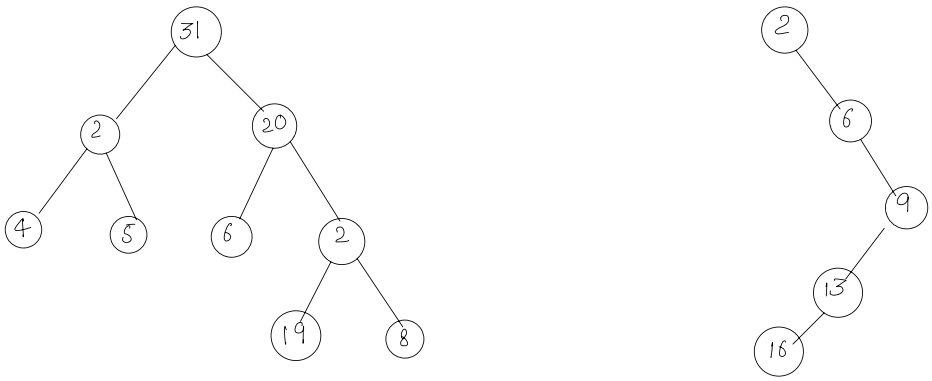
K = 5 4 3 2 1 0
45 Ans
35
45
20
-1
-1
-1
41
45
30
-1
-1
51
45
47
45
60
45
48
55
45
70
46
75

Pseudocode

```
int K = 5;
int KthSmallestInBST(TreeNode root) {
    if (root == null) {
        return -1;
    }

    int left = KthSmallestInBST(root.left);
    if (left != -1) {
        return left;
    }
    K--;
    if (K == 0) {
        return root.data;
    }
    return KthSmallestInBST(root.right);
}
```

[hard] Morris Inorder Traversal [Binary tree]



Last element of inorder traversal ⇒ 8 ——— 9

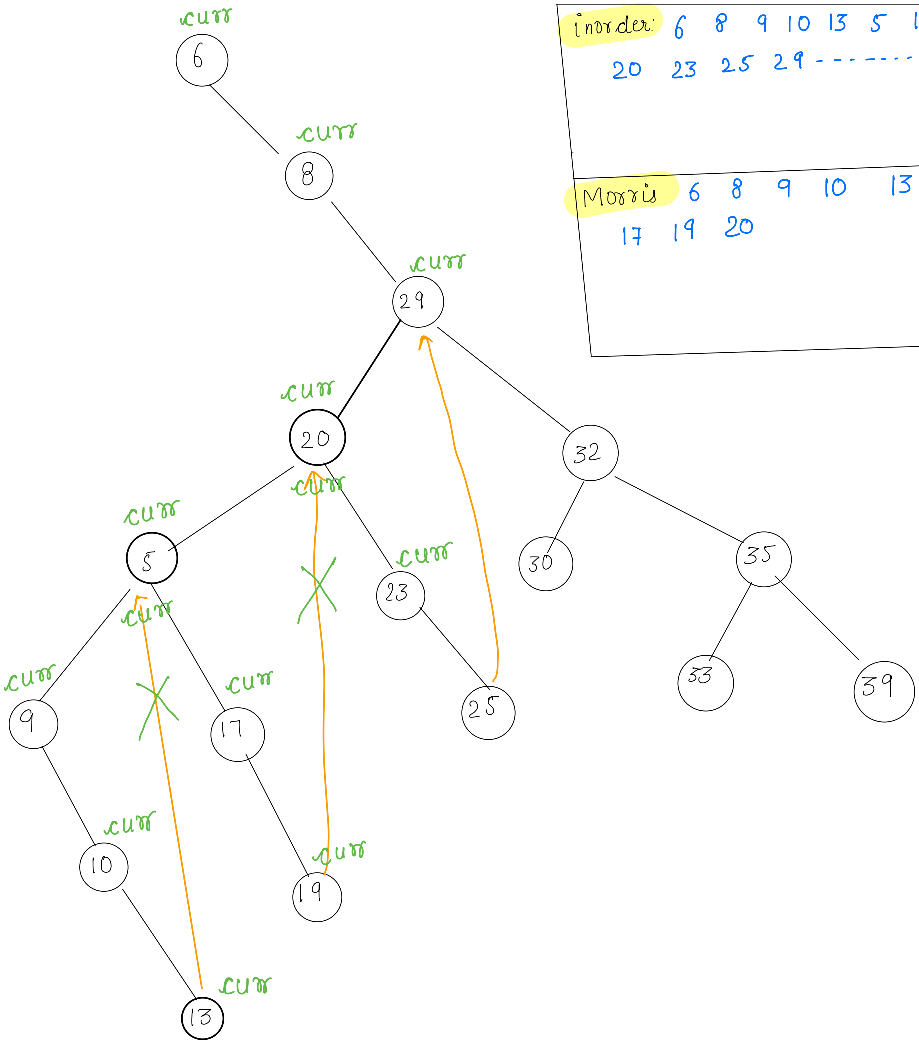
Claim

In a binary tree inorder traversal —

Last node = extreme right of right

Inorder

- Recursive — TC: O(n), SC: O(h)
- Iterative — TC: O(n), SC: O(h)
- Morris — TC: O(n), SC: O(1)



inorder: 6 8 9 10 13 5 17 19
20 23 25 29 - - - - - - - - - -

Morris 6 8 9 10 13 5
17 19 20

Pseudocode

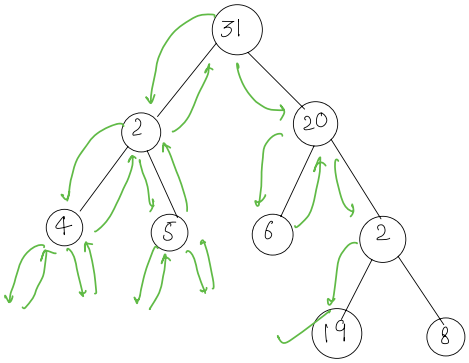
```
void morrisInorderTraversal(TreeNode root) {
    TreeNode curr = root;
    while (curr != null) {
        if (curr.left == null) {
            print(curr.data);
            curr = curr.right;
        } else {
            TreeNode temp = curr.left;
            while(temp.right != null &&
                  temp.right != curr)
                temp = temp.right;

            }
            if (temp.right == null) {
                temp.right = curr;
                curr = curr.left;
            } else {
                temp.right = null;
                print(curr.data);
                curr = curr.right;
            }
    }
```

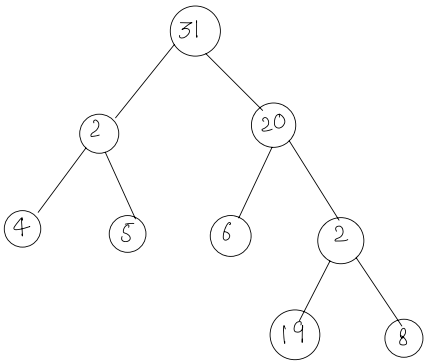
temp.right = null
!= null

TC: O(n)
SC: O(1)

Qu Search an element in binary tree. [Dfs]



K = 19



K = 16

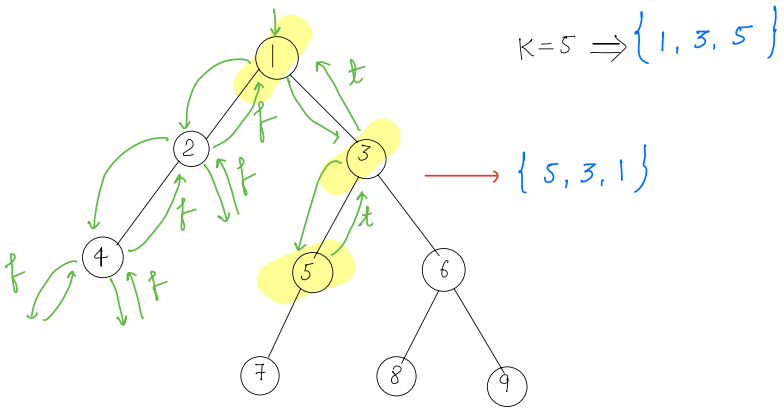
Pseudocode

```
boolean search(root, K) {
    if (root == null) {
        return false;
    }
    if (root.data == K) {
        return true;
    }
    left = search(root.left, K);
    if (left == true) {
        return true;
    }
    return search(root.right, K);
}
```

TC: O(n)
SC: O(height)

Break: 8:35 - 8:42

Qu Path from root to node in binary tree.



K = 5 ⇒ { 1, 3, 5 }

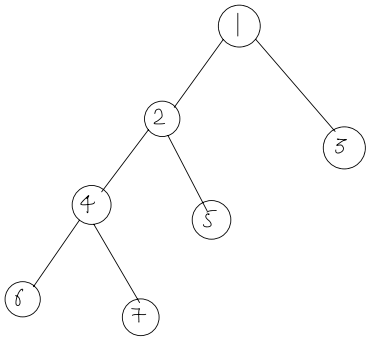
⟶ { 5, 3, 1 }

List<Integer> ans; ⟶ [5, 3, 1]

```
boolean rootToNodePath(root, k) {
        if(root == null) {
            return false;
        }
        if(root.data == k) {
            ans.add(root.data);
            return true;
        }
        left = rootToNodePath(root.left, k);
        if(left == true) {
            ans.add(root.data);
            return true;
        }
        right = rootToNodePath(root.right, k);
        if(right == true) {
            ans.add(root.data);
            return true;
        }
        return false;
}
```

LCA of binary tree

Lowest common Ancestor



lca(6,3) = 1

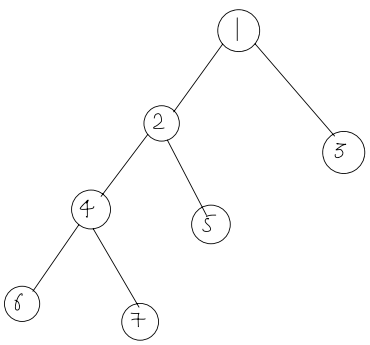
lca(2,5) = 2

lca(6,5) = 2

lca(7,3) = 1

lca(7,1) = 1

Approach:



lca(x,y)

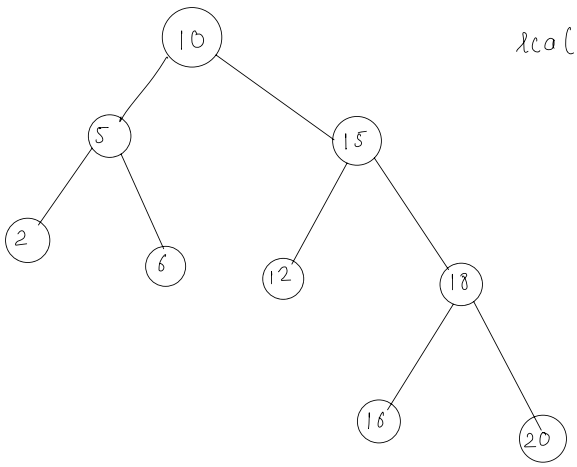
x	y	lca
		root
left	right	
left	left	left
right	right	right
root	any node	root
any node	root	root

Pseudocode

```
TreeNode findLCA(root, x, y) {
    if (root == null) {
        return null;
    }
    if (root.data == x || root.data == y) {
        return root;
    }
    left = findLCA(root.left, x, y);
    right = findLCA(root.right, x, y);
    if (left != null && right != null) {
        return root;
    }
    if (left == null) {
        return right;
    }
    return left;
}
```

TC: O(n)
SC: O(h)

LCA of BST



lca(12, 20) =

Logic x = 2 and y = 6 → lca(2, 6) = 5 ⟶ left subtree

lca (x = 16 and y = 20) ⟶ 18 ⟶ right subtree.

x = 2 and y = 16 ⟶ root

Pseudocode

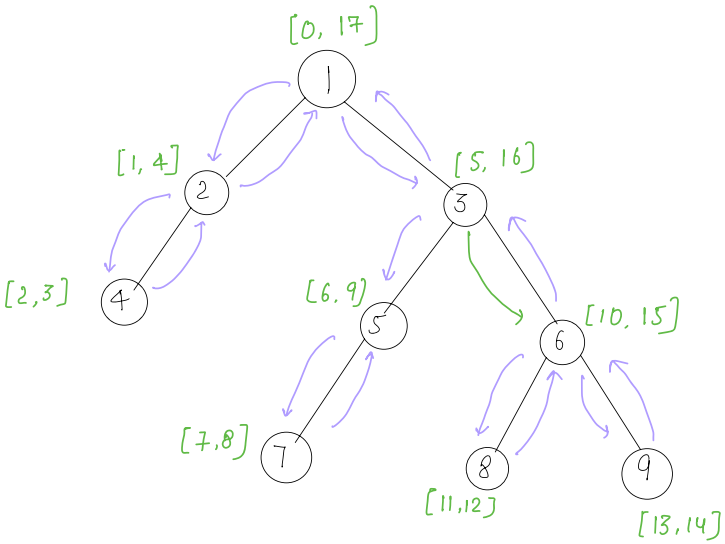
```
TreeNode findLCA(root, x, y) {
    curr = root;
    while(curr != null) {
        if(curr.data > x && curr.data > y) {
            curr = curr.left;
        } else if(curr.data < x && curr.data < y) {
            curr = curr.right;
        } else {
            return curr;
        }
    return null;
}
```

TC: O(n)
SC: O(1)

In time and out time

time at which you first encounter a node

time at which you leave / go away from a node



inmap < Integer, TreeNode >
time

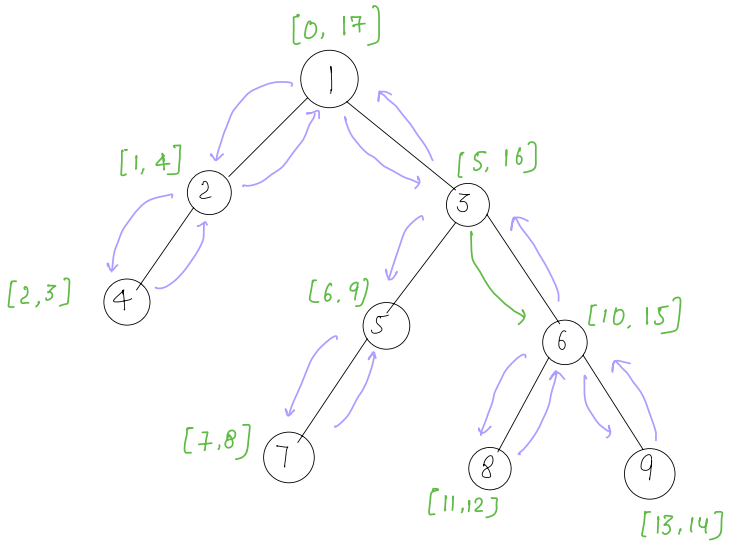
0 → 1
1 → 2
2 → 4
5 → 3
⋮

outmap < Integer, TreeNode >
time

17 : 0
4 : 2
⋮

Pseudocode

Qu Given Q queries, find LCA of all queries.



Approach $in(x) < in(y)$
$out(x) > out(y)$
} x is ancestor of y.

x = 5 and y = 7

in: 6 7

out: 9 8

x = 3 and y = 9

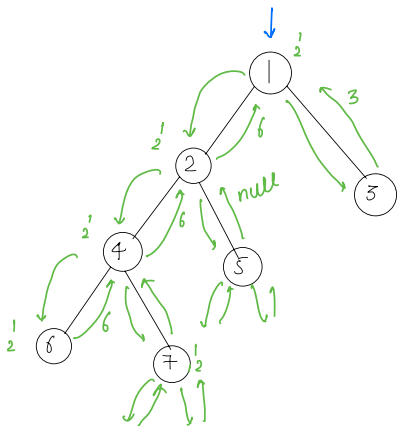
in: 5 13

out: 16 14

$in(y) < in(x)$
$out(y) > out(x)$
} y is ancestor of x.

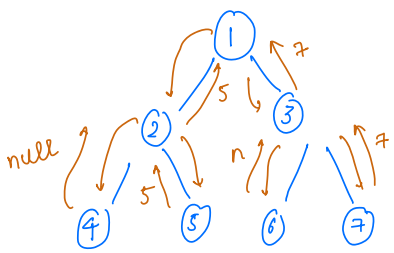
Thankyou

Doubts

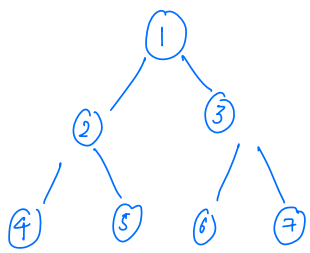


lca(6,3)

```
TreeNode findLCA(root, x, y) {
1   if (root == null) {
        return null;
    }
2   if (root.data == x || root.data == y) {
        return root;
    }
3   left = findLCA(root.left, x, y);
4   right = findLCA(root.right, x, y);
5   if(left != null && right != null) {
        return root;
    }
6   if (left == null) {
        return right;
    }
7   return left;
}
```



x = 5
y = 7



x = 4
y = 6