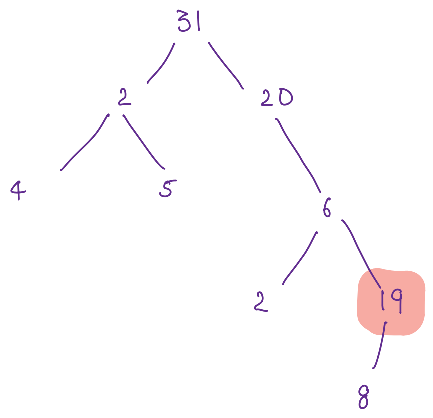
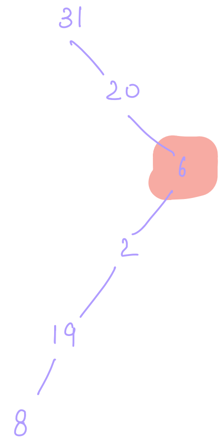
Lecture ÷ Trees -4

Agenda

- Morris inorder traversal
- Kth smallest element in BST
- LCA in BST
- LCA in BT.

Morris Inorder traversal [Hard]





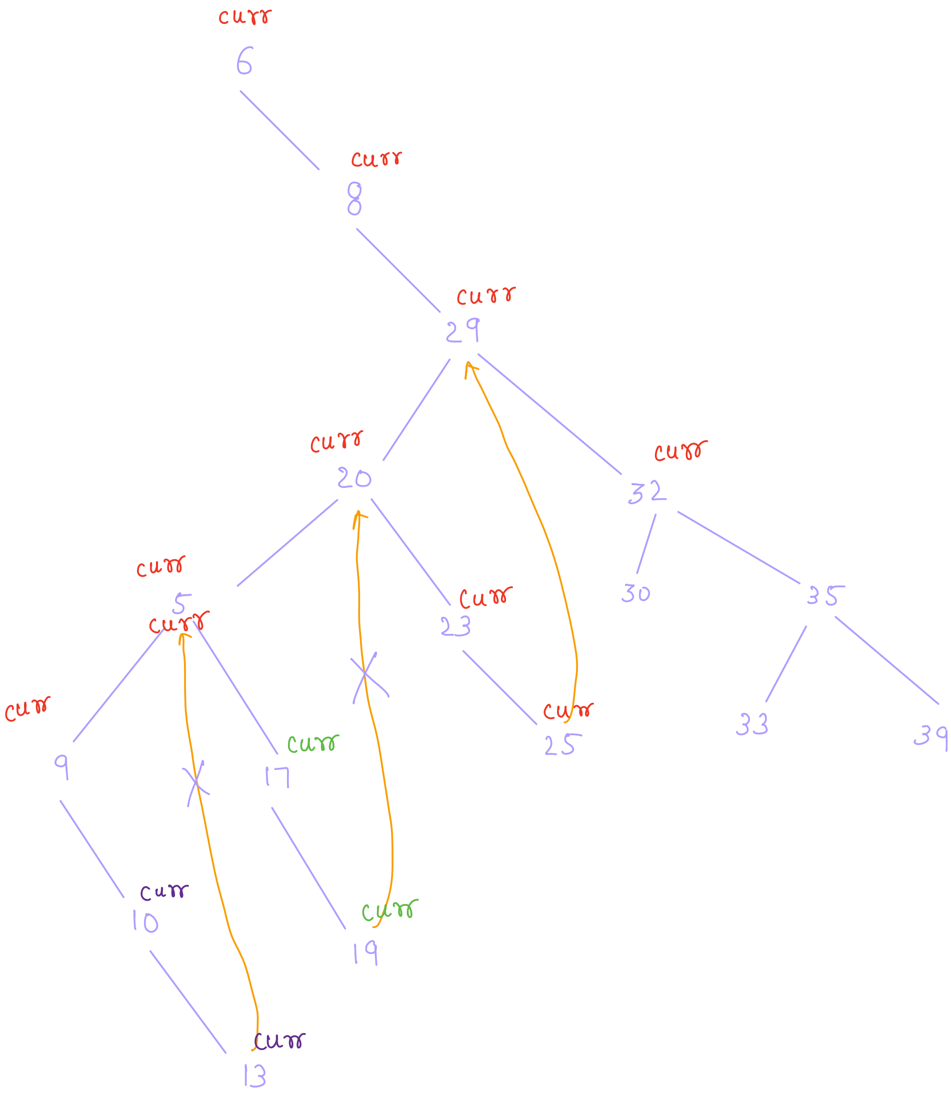
Last el of inorder ÷ 19

Claim: In a B.T inorder traversal —

Last node = Extreme right of root.

Inorder

- Recursively : TC: O(n)
SC: O(h)
- Iterative TC: O(n)
SC: O(h)
- Morris TC: O(n)
SC: O(1)



Print: 6 8 9 10 13 5 17 19 20

23 25 29

TC: O(n)

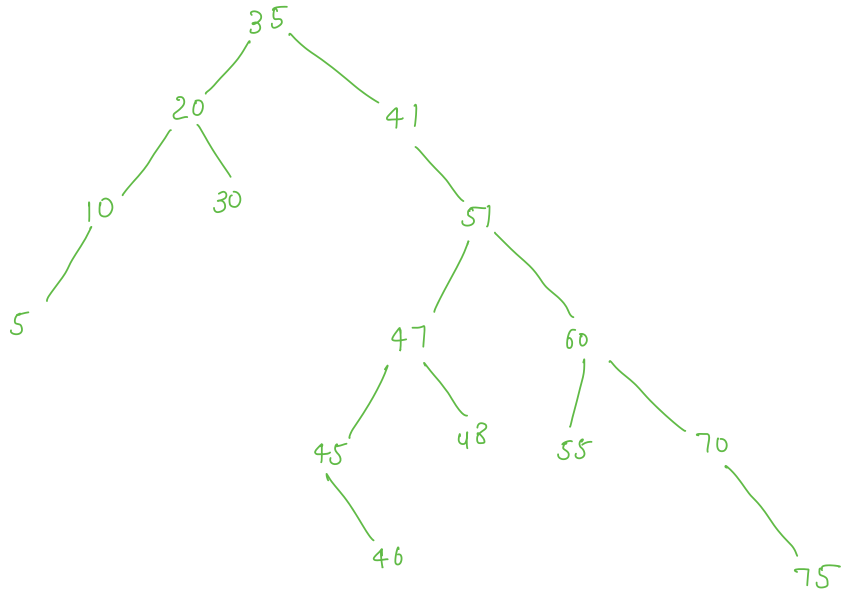
SC: O(1)

```
void  morrisInorder( Node  root) {
        Node  curr =  root;
        while( curr != null)  {
              if( curr.left == null) {
                    print(curr.data);
                    curr = curr.right;
              }   else  {
                  Node   last  =   curr.left;
                 while( last.right != null   &&
                             last.right != curr )   {
                        last = last.right;
                 }
                 if( last.right == null)  {
                       last.right  =  curr;
                       curr = curr.left;
                 }  else  {
                       last.right  = null;
                       print(curr.data);
                       curr = curr.right;
                 }
  }
```

last.right =null

last.right != null

Qu2 kth smalleot el in a BST.



k=1	→	5
k=2	→	10
k=3	→	20
k=4	→	30
k=5	→	35
k=6	→	41
k=7	→	45

Idea1: Inorder of BST = sorted array.

find kth el of sorted array.

TC: o(n)

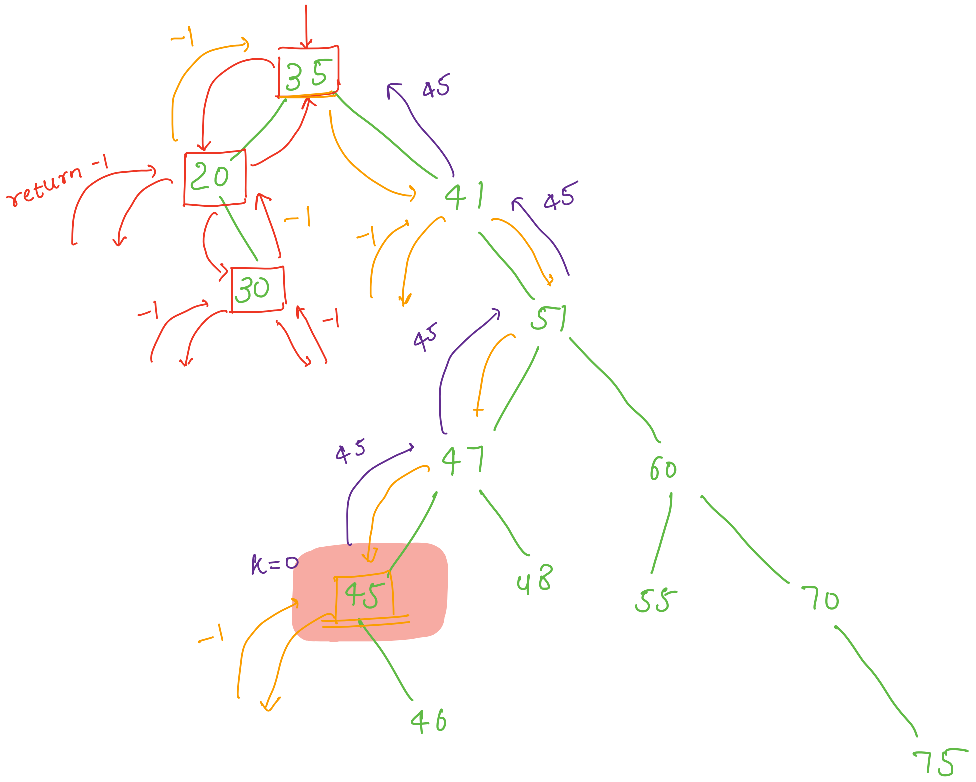
SC: o(n)

Idea2:

TC: O(n)

SC: O(h)

K = ~~5~~ ~~4~~ ~~3~~ ~~2~~ ~~1~~ 0

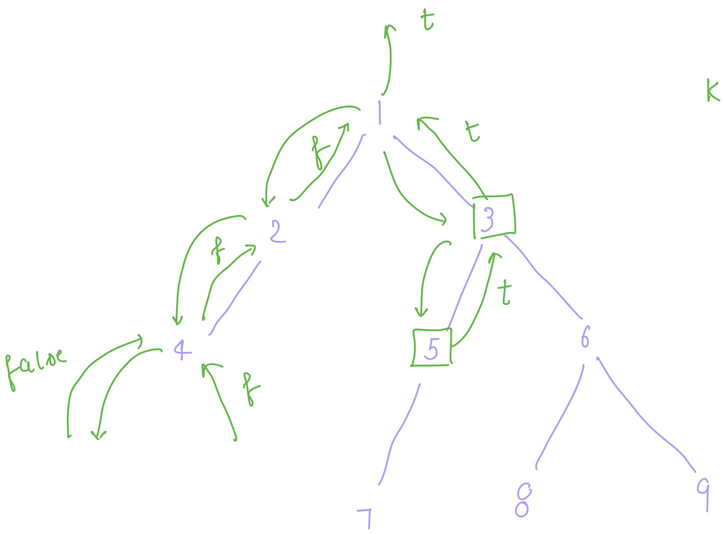


```
int k;

int kthsmallest ( Node root) {
        if(root == null) {
            return -1;
        }

        int left = kthsmallest(root.left);

        if( left != -1) {
            return left;
        }

        k--;
        if( k==0) {
            return root.data;
        }
        return kthsmallest( root.right);
}
```

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

Break: 8:18 - 8:30 AM

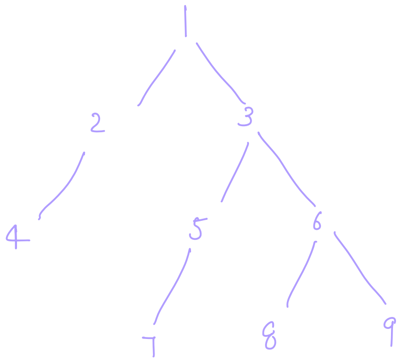
Qu **Root to Node Path**



k = 5 [1 3 5]

```
        List<Integer> list;          ——→  [ 5  3  1 ]
boolean search( Node root, int k) {

        if(root == null) {

            return false;
        }

        if(root.data == k) {
            list.add(root.data);
            return true;
        }

        boolean left = search(root.left, k);

        if( left == true) {
            list.add(root.data);
            return true;
        }
        boolean right = search(root.right, k);
        if( right == true) {
            list.add(root.data);
            return true;

        }
        return false;
}
```

Lowest common Ancestor



Root to node path:

7: [1 3 5 7] → ans = 3
8: [1 3 6 8]

5: [1 3 5]
9: [1 3 6 9] — ans = 3

6: [1 3 6] — ans = 6
8: [1 3 6 8]

lca(7, 9) = 3

lca(5, 8) = 3

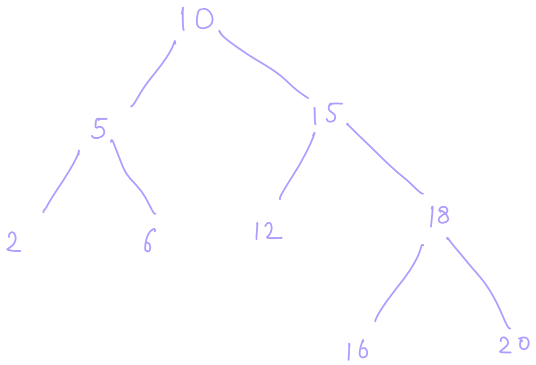
lca(4, 8) = 1

lca(3, 8) = 3

TC: O(n)

SC: O(h)

LCA of BST ★★★



lca(12, 20) = 15.

Logic:

x = 2 and y = 6 [Lca is in left subtree]

x = 16 and y = 20 [lca " " right "]

$\underbrace{x = 2}_{\text{Left subtree}}$ and $\underbrace{y = 16}_{\text{Right subtree}}$ [lca == root]

```
Node lcaBST( Node root, x, y) {
    curr = root;
    while( curr != null) {
        // x & y are in LST
        if( curr.data > x  && curr.data > y) {
            curr= curr.left;
        } else if( curr.data < x && curr.data < y) {
            curr = curr.right;
        } else {
           return curr;
        }
}
```

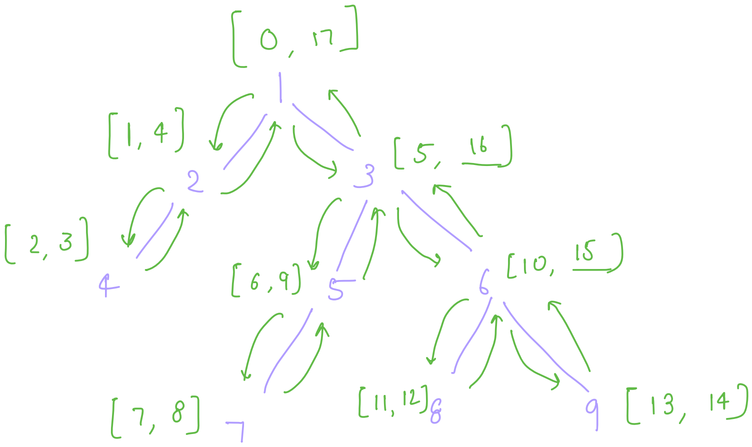
TC: O(height)
SC: O(1)

In time and out time

time at which you first come at a node

time at which you go from that node

t=0



Inmap < Integer, Node>

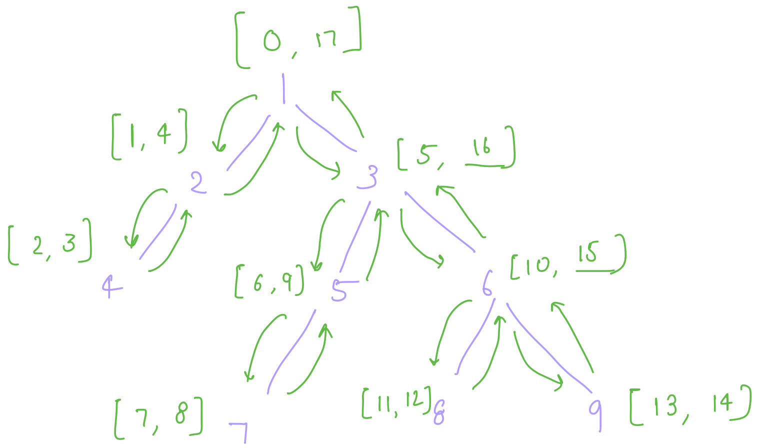
0 :	1
1 :	2
2 :	4
5 :	3
6 :	5
7 :	7
11 :	8
13 :	9
10 :	6

outmap

17 :	0
4 :	2

```
Map< Integer, Node> inMap;

Map <Integer, Node> outMap;

 t=0;

void  traversal (Node  root) {

      if( root == null) {

            return;
      }

      in Map. put( t , root);

      t ++;

      traversal (root. left);
      traversal (root. right);
      outmap  ( t , root);
       t++;
}
```

Qu Given q queries, find LCA of all queries.



1.
$$\left.\begin{array}{l} \text{in}(x) < \text{in}(y) \\ \text{out}(x) > \text{out}(y) \end{array}\right\} \longrightarrow$$ x is ancestor of y

$x = 5$ and $y = 7$ [anc = 5]

$\text{in}(5) = 6$ $\text{in}(7) = 7$ $\text{in}(x) < \text{in}(y)$

$\text{out}(5) = 9$ $\text{out}(7) = 8$ $\text{out}(x) > \text{out}(y)$

2.
$$\left.\begin{array}{l} \text{in}(y) < \text{in}(x) \\ \text{out}(y) > \text{out}(x) \end{array}\right\} \longrightarrow$$ y is ancestor of x

Algo:

```
if( x is ancestor of y) { return x; }      optional
              ↓
    in(x) < in(y)  &&  out(x) > out(y)

if( y is ancestor of x) { return y; }
    in(y) < in(x)  &&  out(y) > out(x)

curr = root;

while( curr != null ) {

        if( curr.left  is  ancestor of  x    &
             in(curr.left) < in(x)  &&
h/w          out(   "    ) > out(x)
           curr.left   "    "    "    y ) {

                curr = curr.left;

        } else if( curr.right is ancestor of x    &&
h/w
                curr.right  "   "   "   y) {

                curr = curr.right;
h/w     } else{
              return curr;
        }
}
```

TC: $O(\underline{Q * h}) + O(n)$

SC: $O(n)$