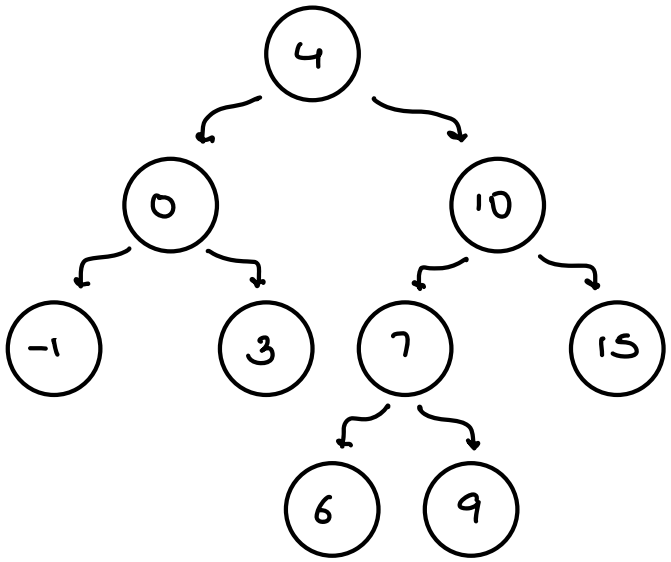
TREES 4 : LCA

Good Evening

Content

01. K^th Smallest Element in BST
02. Morris Inorder Traversal
03. LCA in BST
04. LCA in BT

01. Find K^{th} smallest element in BST



K = 3 → 3

K = 5 → 6

K = 8 → 10

Idea 1 → DO inorder traversal & store elements in arr & return arr[K-1]

TC: O(n)

SC: O(n)

Idea 2 → On the fly, try to maintain a count variable

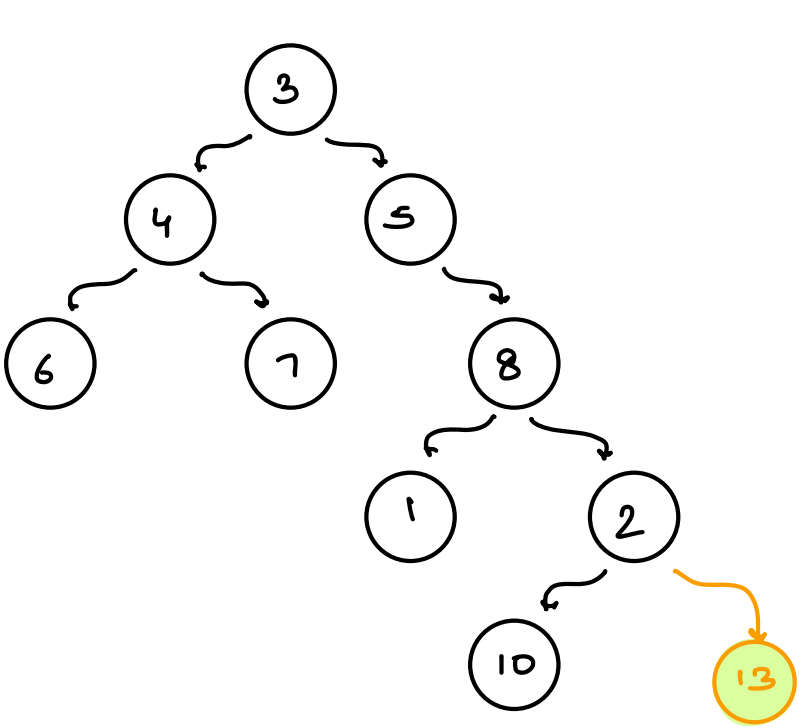
count = 0 , ans = 0 → Global variable

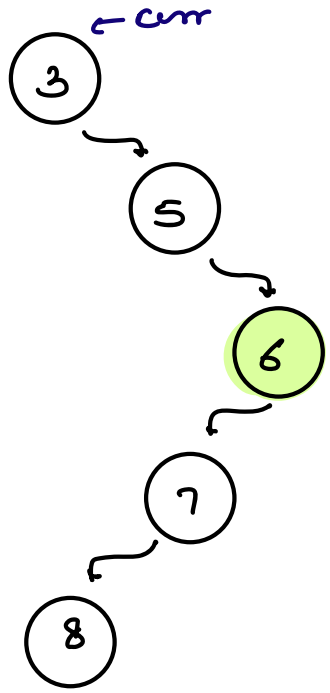
```
void traversal ( root, k )
    if (root == null) return;
     traversal (root.left , k )
      count = count + 1 ;
     if (count == k) { ans = root.data }
     traversal (root.right , k )
```

TC: O(n)

SC: O(n)

Q What is the last node we print when doing inorder traversal L D R

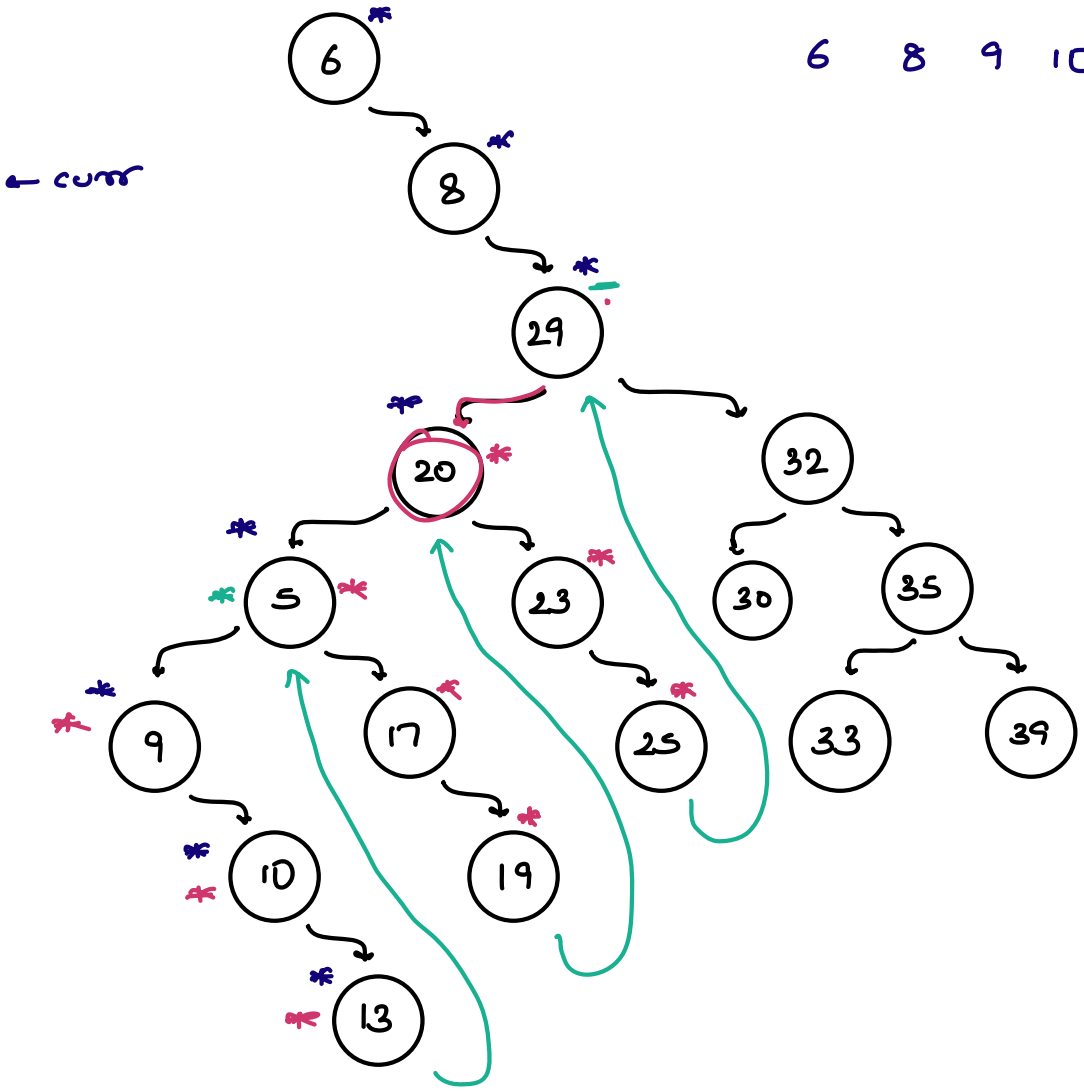




3 5 8 7 6

Claim :- In a BT inorder traversal, last node will always be extreme right node of root

Q Morris Inorder Traversal { No extra space }



6 8 9 10 13

```
Node temp = curr.left
while (temp.right != null)
    temp = temp.right;
```

- `temp.right = curr;`

`curr = curr.left;`

```
inorder ( Node root )

curr = root ;

while ( curr != null ) {

    if ( curr.left == null )
        print ( curr.data ) ;
        curr = curr.right ;

    else {
        Node 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 ;
    }
}
```

first time or second time

first time

second time

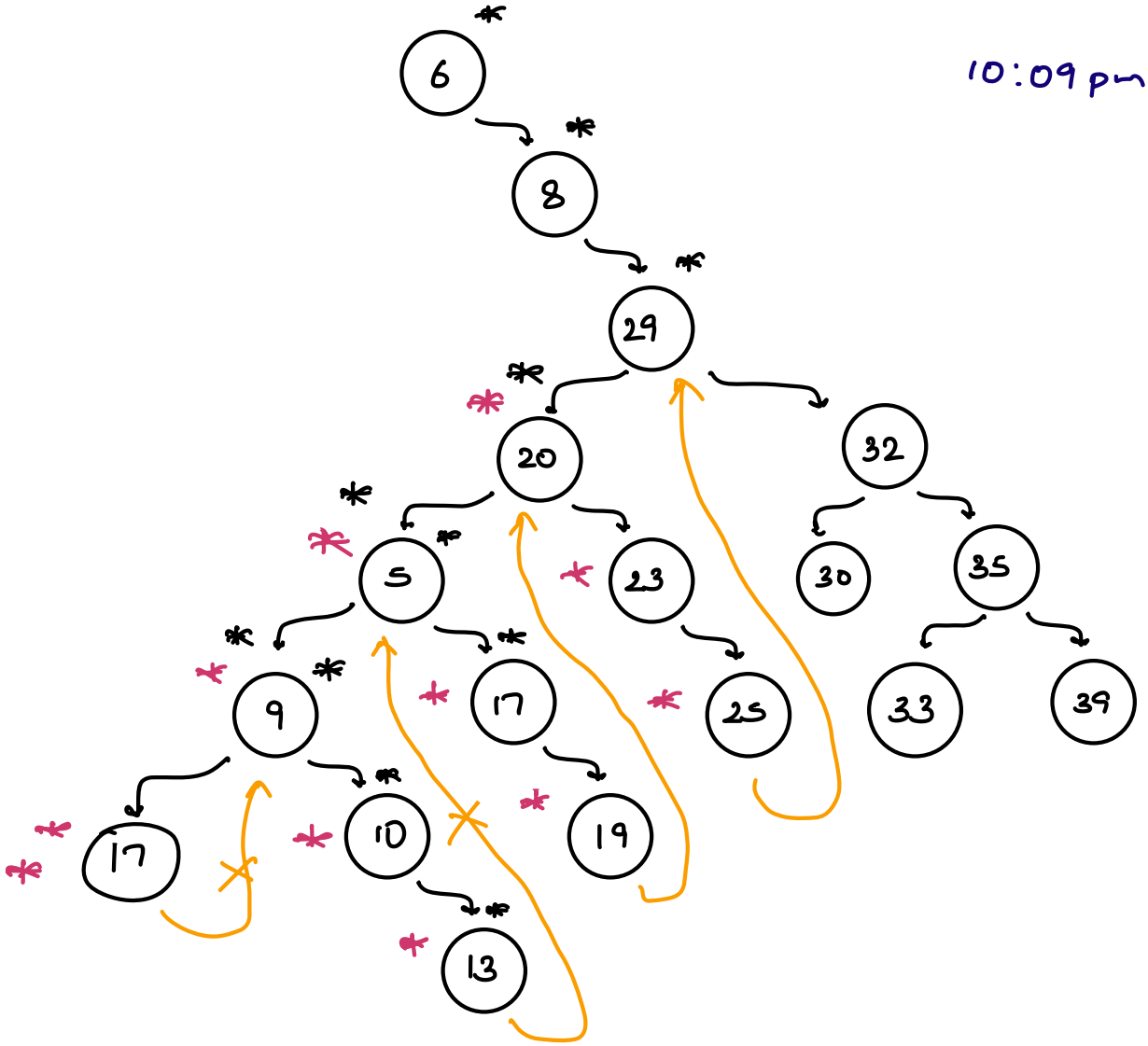
TC : O(n)

SC : O(1)

6 8 17 9 10 13 5

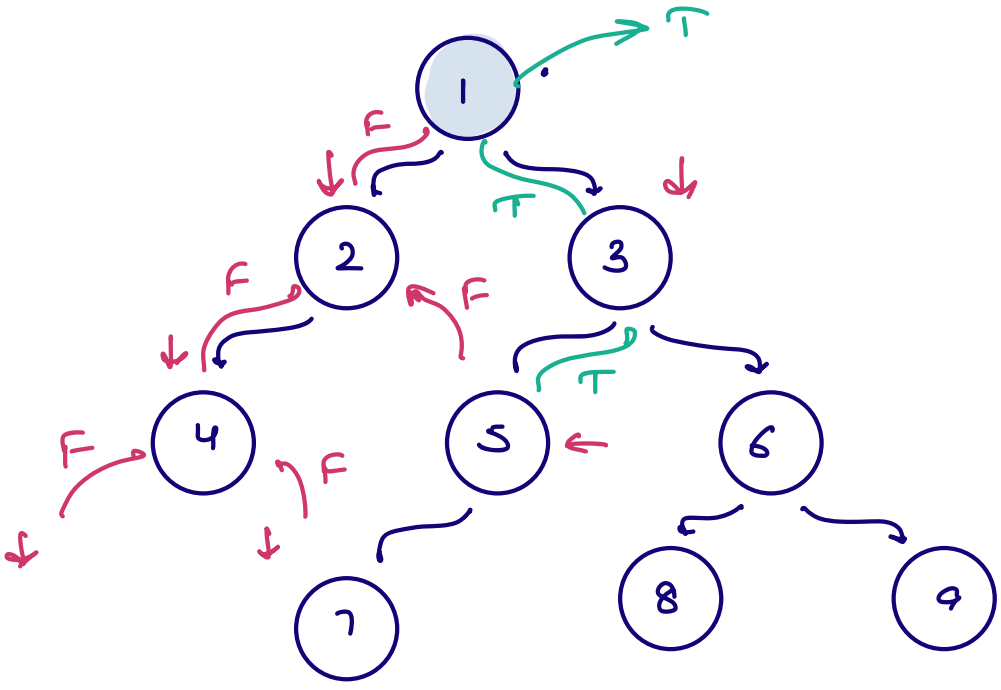
10:09 pm → 10:19 pm

10:23 pm



Node to Root path

Search if k is present in Binary Tree or not



K = 5

TC : O(n)

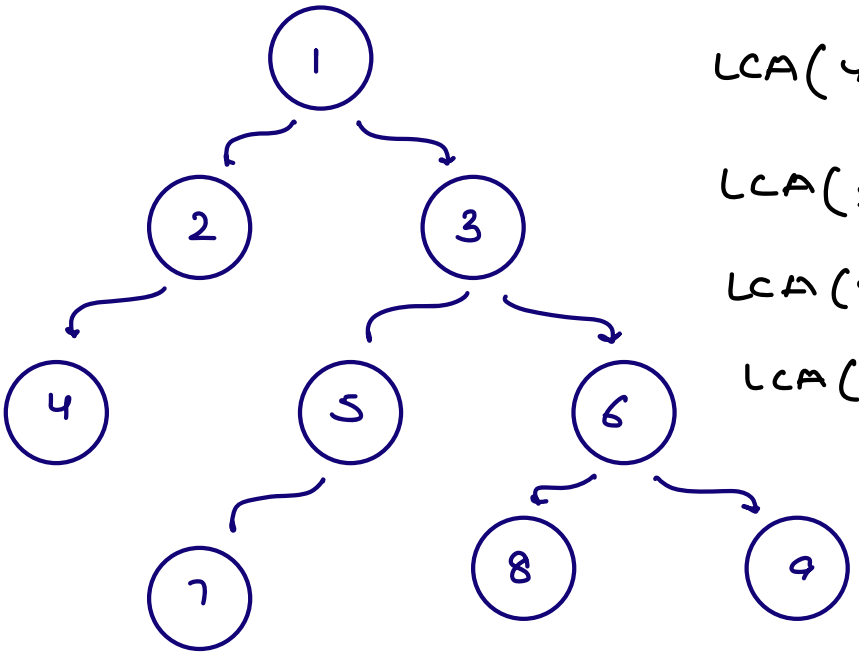
SC : O(h)

```
boolean search ( root , k , AL<I> al )
    if (root == null) false :
    if (root.data == k) {
        al.add (root.data);
        return true;
    }
    if ( search (root.left, k) == true ) {
        al.add (root.data);
        return true:
    }
```

```
        if (search(root.right, k) == true){
            al.add(root.data);
            return true;
        }
    }
    return false;
}
```

* LCA of BT

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

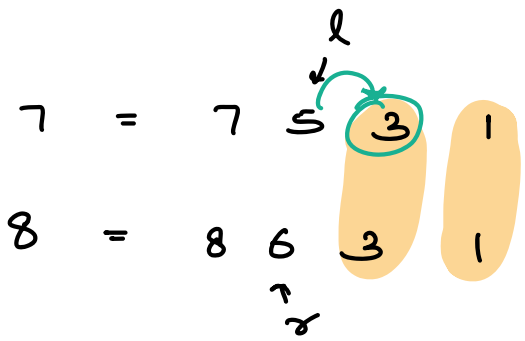


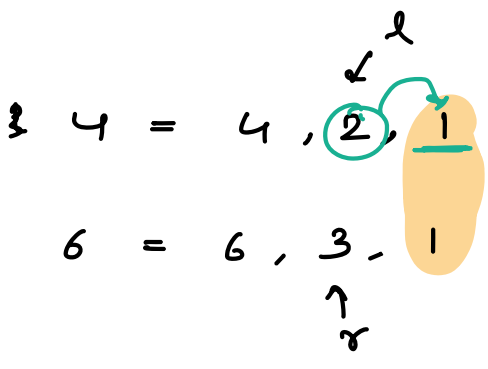
LCA(4,6) = 1

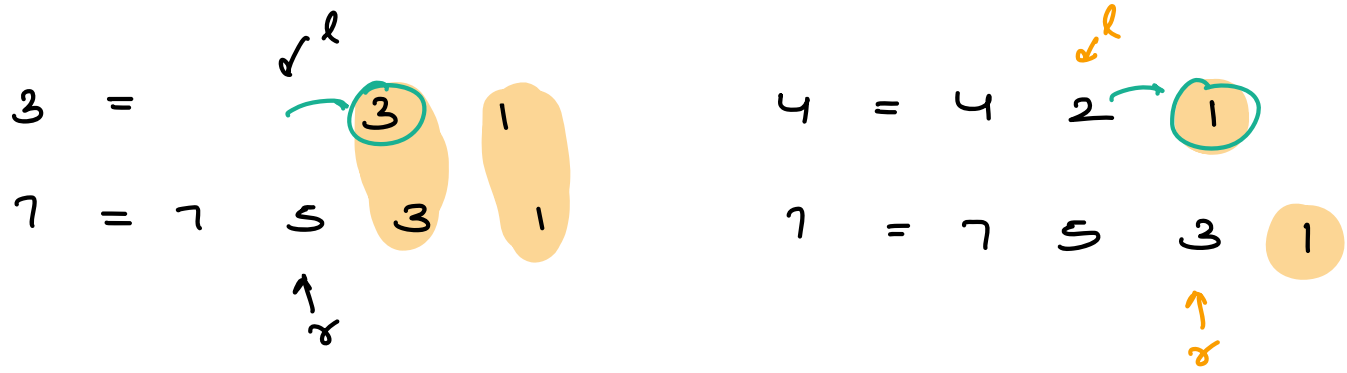
LCA(5,6) = 3

LCA(7,8) = 3

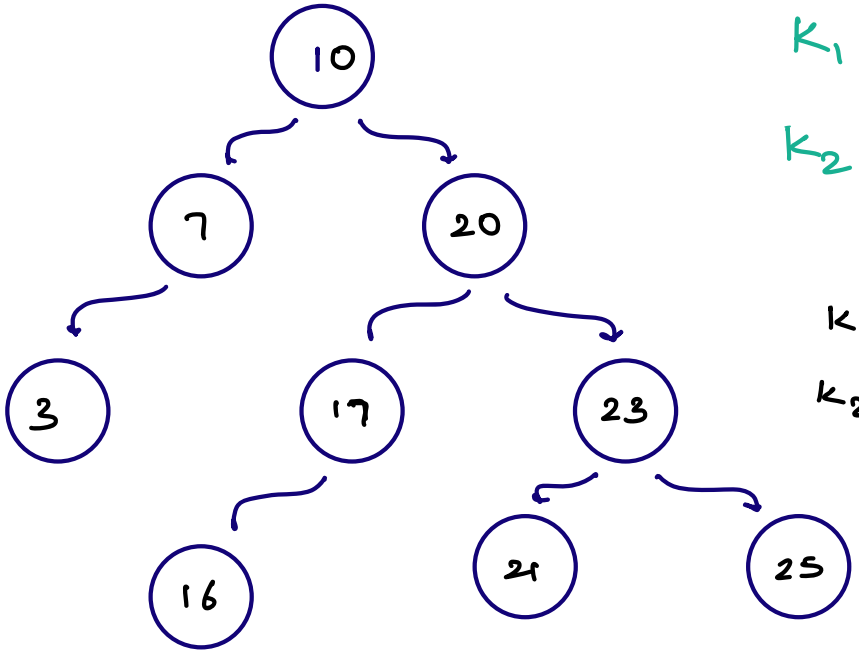
LCA(6,9) = 6







* LCA in BST



$k_1 = 25$

$k_2 = 21$

$k_1 = 20$

$k_2 = 23$

```
while (curr != null)
    if (root.data < k1 && root.data < k2){
        curr = curr.right;
    }
    else if (root.data > k1 && root.data > k2){
        curr = curr.left;
    }
```

```
        else {
            return curr;
        }
    }
}
```

In time – Out time Approach