Trees 4

Find Kth Smallest Element
Morris Traversal
Path from root to element's node
LCA in B tree
In Time and Out time in a B tree
LCA(a, b) for multiple Queries

Find Kth Smallest Element

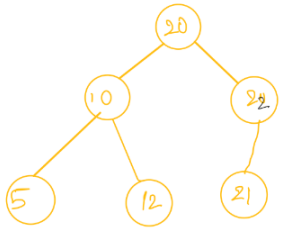
Binary Tree : Heirarchial data structure where each node can have at max 2 children.

Binary Search Tree : Type of B tree with an additional property.

for all nodes.

All ele in LST	<=	Node	<=	All ele in RST

Q Find K^{th} Smallest element in a BST.



K=1 : 5
K=3 : 12

→ Inorder traversal of BST is a sorted array.
{5, 10, 12, 20, 22, 21}

Approach 1 :
Do inorder traversal & store in array & return k^{th} index.
T : O(N)
S : O(N)

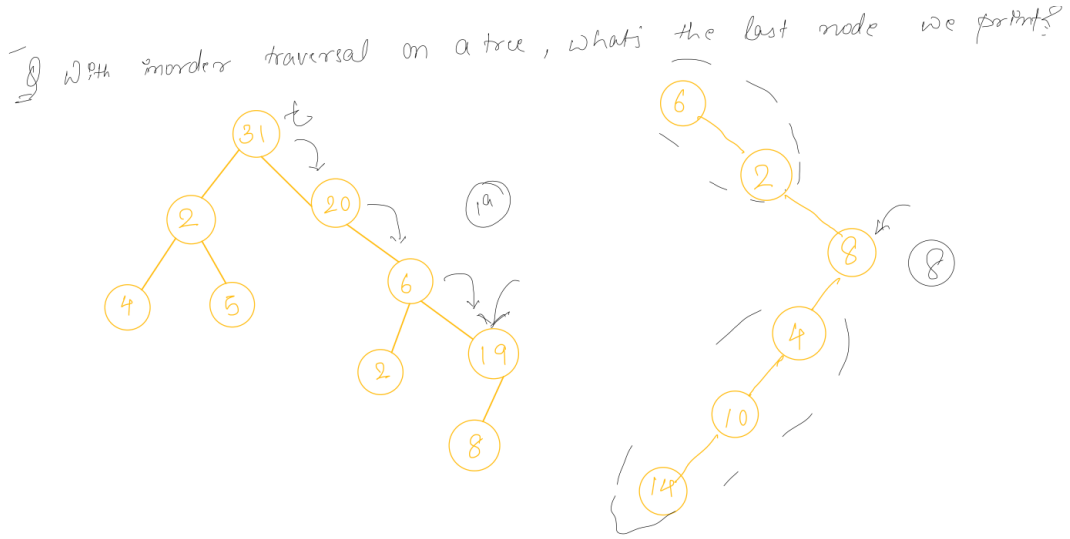
Approach 2 :
Keep a counter;

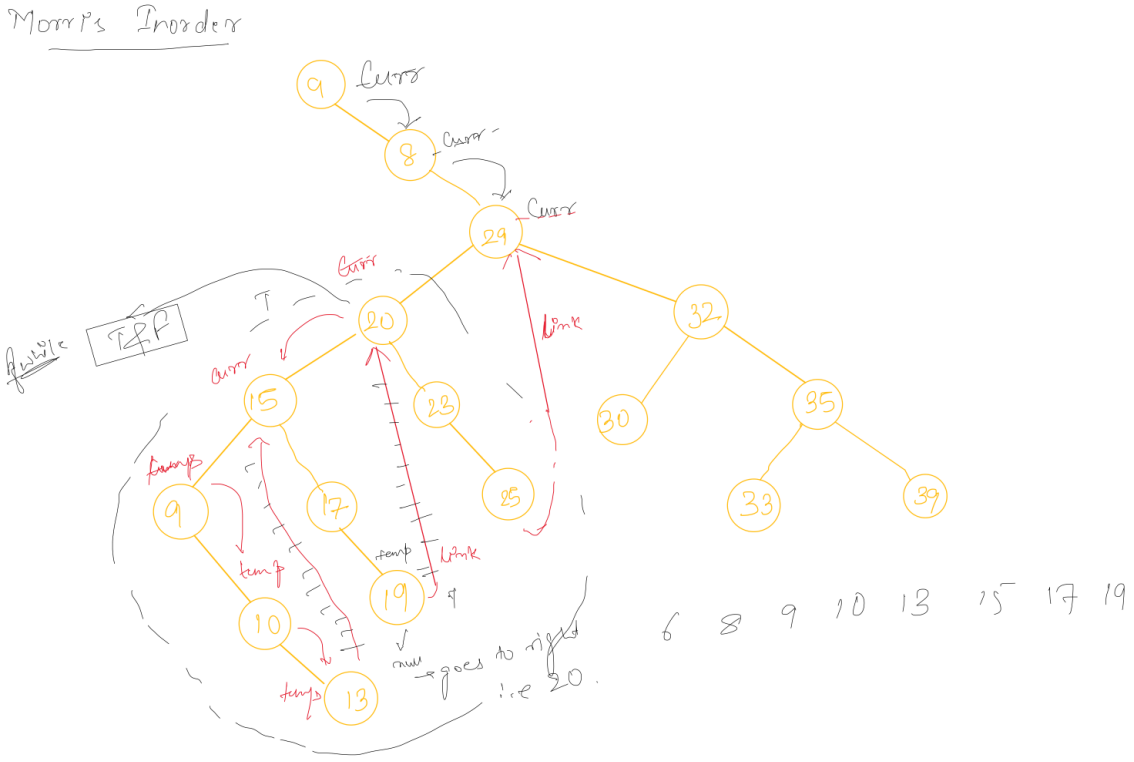
```
Void Inorder (Node root, Int K) {
    if (root == NULL)
        return;
    Inorder (root.left, K);
    cnt++;
    if (cnt == K) {
        ans = root.data;
        return;
    }
    Inorder (root.right, K);
}
```

T.C : O(N)
S.C : O(H)

Approach 3 : Inorder traversal using a Stack.
T.C : O(N)
S.C : O(H)

Morris Traversal





Pseudo Code :-

```
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    //visiting it for 2nd time;
    {   temp.right = NULL;
        print (Curr.data)
        Curr = Curr.right.
    }
}
```

{Scenario when Curr might be of prev node}

Visiting Curr for second time

Path from root to element's node

Q Search an ele in Binary tree; (All distinct) / Path from root to ele.

Search(25) : True

Pseudo Code :-

```
List<int> ans
bool Search(Node root, int k)
{
    if (root == NULL)
        return false;
    if (root.data == k)
    {
        ans.push(root.data)
        return true;
    }
    if (search(root.data, k) || search(root.right, k))
    {
        ans.push(root.data);
        return true;
    }
    else
        return false;
}
```

//reverse (ans) → path from root to node.

LCA in B tree

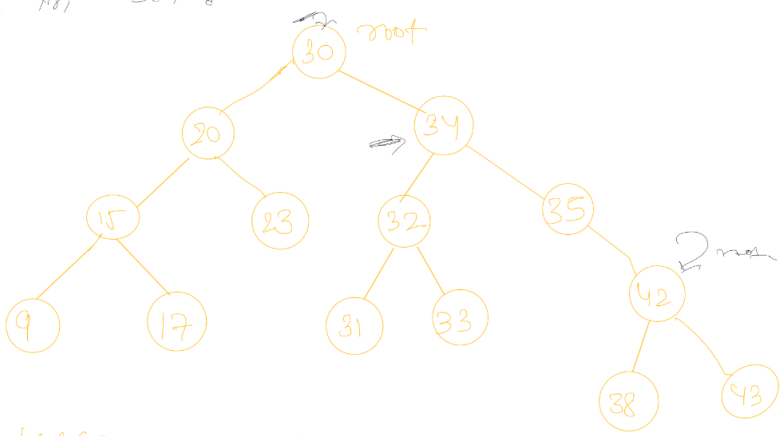
Q Find LCA (a, b) in a B-Tree.

LCA (16, 18) = 9
LCA (50, 19) = 16
LCA (11, 9) = 9
LCA (25, 9) = 3

LCA (a, b):
Path from Root to a: { 3 9 32 16 50 }
Path from Root to b: { 3 9 32 11 }
→ Traverse until you find first non common ancestor.

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

Q. LCA in BST :-



LCA(31, 43) = 34 LCA(34, 33) = 34

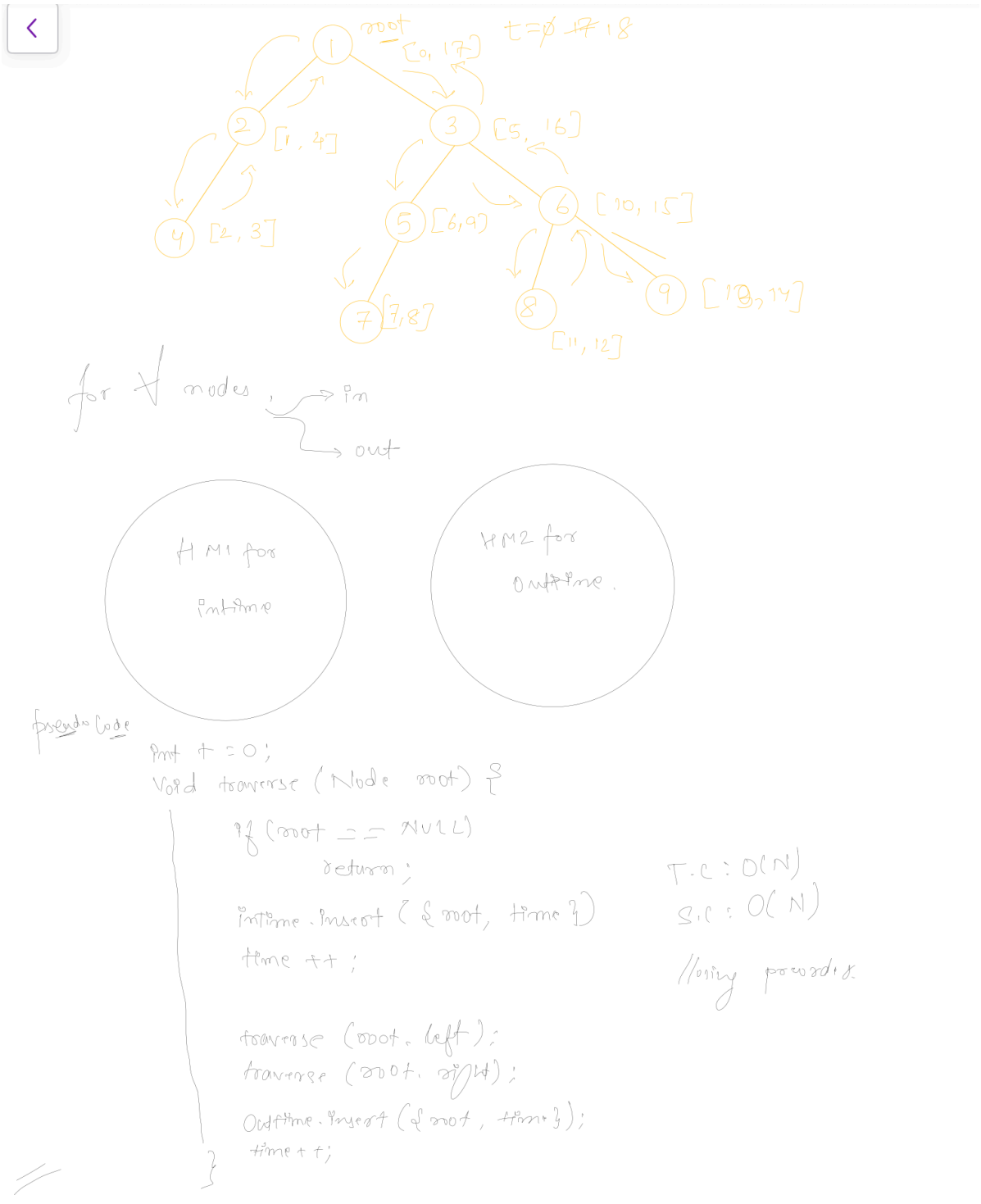
LCA(23, 35) = 30

LCA(38, 43) = 42

Pseudo code

```
Node curr = root;
while (true)
{
  if (a.data < curr.data && b.data < curr.data)
  {  curr = curr.left;
  }
  else if ( a.data > curr.data && b.data > curr.data)
  {  curr = curr.right;
  }
  else {
      break;
  }
}
return curr.data;
```

In Time and Out time in a B tree



LCA(a, b) for multiple Queries

LCA(5, 8) : 3

(6, 9) (11, 12)

a, b
in(a) < in(b)
out(a) > out(b)
} a is an ancestor.

Pseudo Code :-

```
Curr = root;
while (Curr != NULL)
{   if (Curr.left is an ancestor of a & b)
    {
        Curr = Curr.left;
    }
    else if (Curr.right is an ancestor of a & b) {
            Curr = Curr.right;
    } else {
        return Curr;
    }
}
```