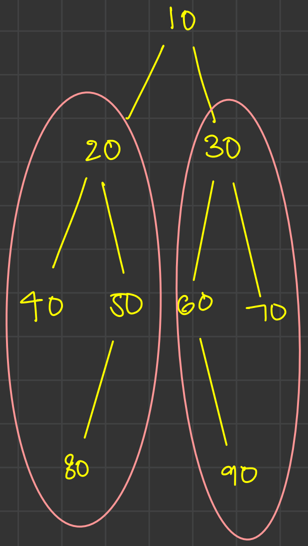
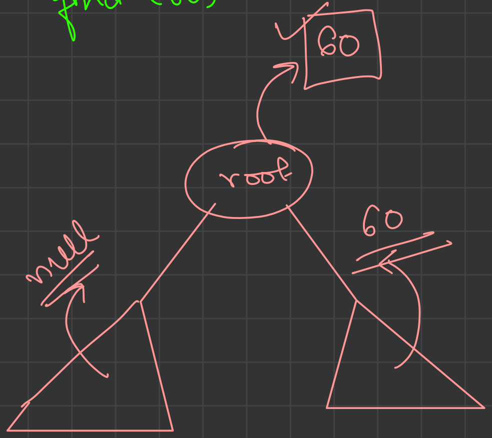
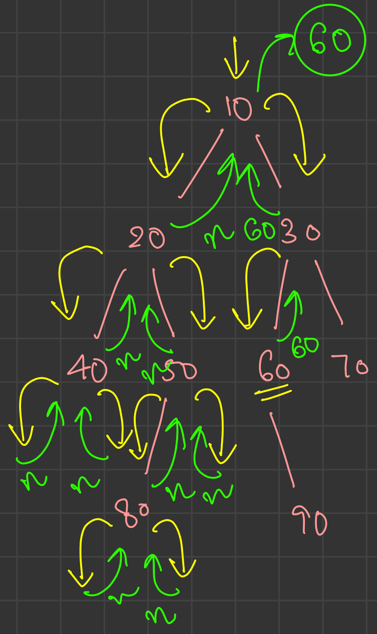
find a given node in a tree



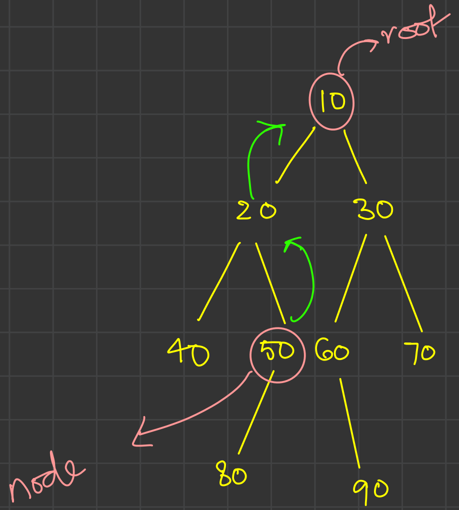
find (60)



```
Node find (TreeNode root, int target)
{
    if (root == null )
        return null;

    if (root .data == target)
        return root;

60  Node filc = find (root.left, target);
    if (filc != null)
        return filc;

60  Node firc = find (root.right, target);
    if (firc != null)
        return firc;

    return null;
}
```

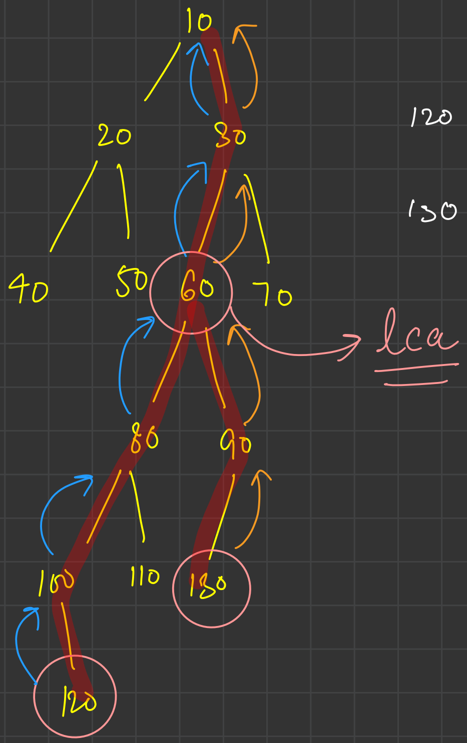


Node to Root Path



{50, 20, 10}

LCA (Lowest Common Ansecstor)



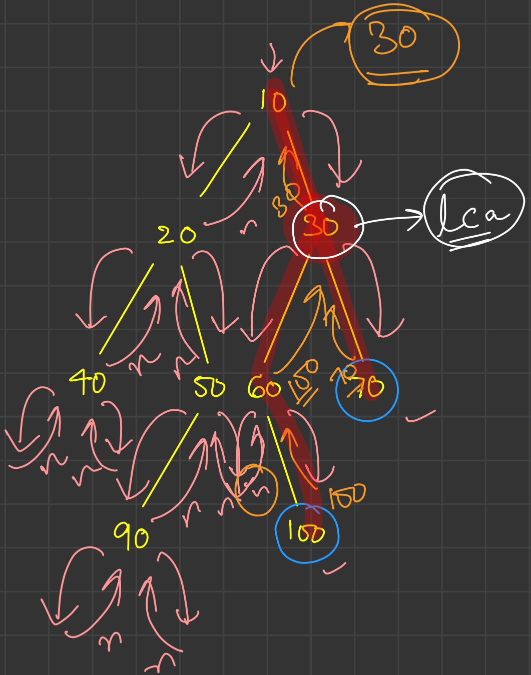
i

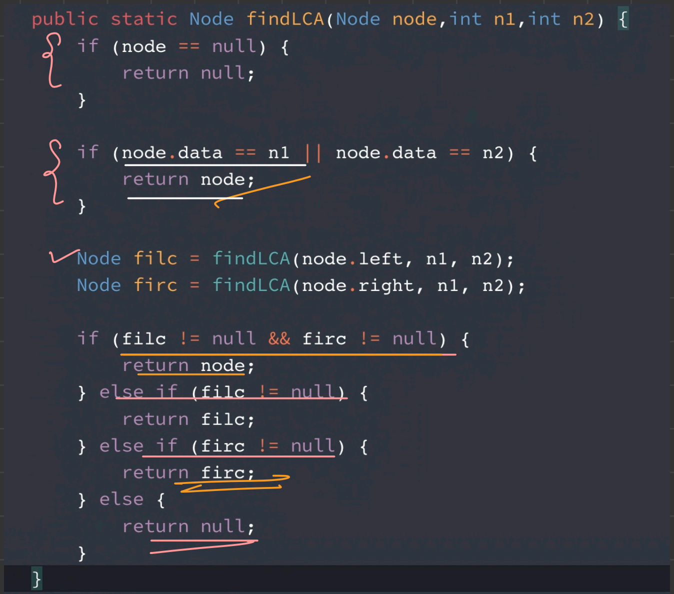
$120 \longrightarrow \{120, 100, 80, 60, 30, 10\}$

$130 \longrightarrow \{130, 90, 60, 30, 10\}$

j

$lca = \cancel{10}\ \cancel{30}\ 60$





```
public static Node findLCA(Node node,int n1,int n2) {
    if (node == null) {
        return null;
    }

    if (node.data == n1 || node.data == n2) {
        return node;
    }

    Node filc = findLCA(node.left, n1, n2);
    Node firc = findLCA(node.right, n1, n2);

    if (filc != null && firc != null) {
        return node;
    } else if (filc != null) {
        return filc;
    } else if (firc != null) {
        return firc;
    } else {
        return null;
    }
}
```

TC : O(N)

SC : O(H)

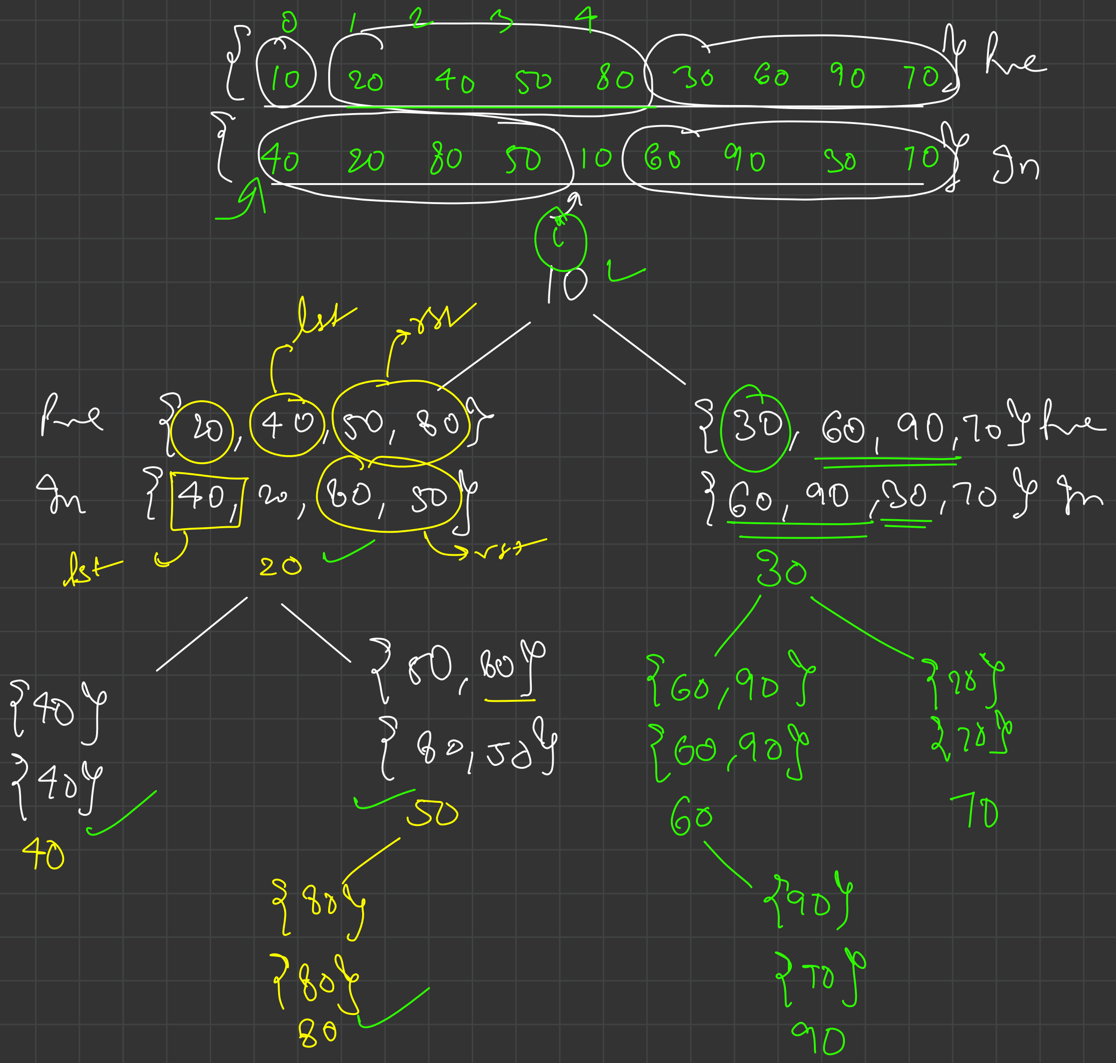
Construct Binary Tree from Pre Order In Order

root left right

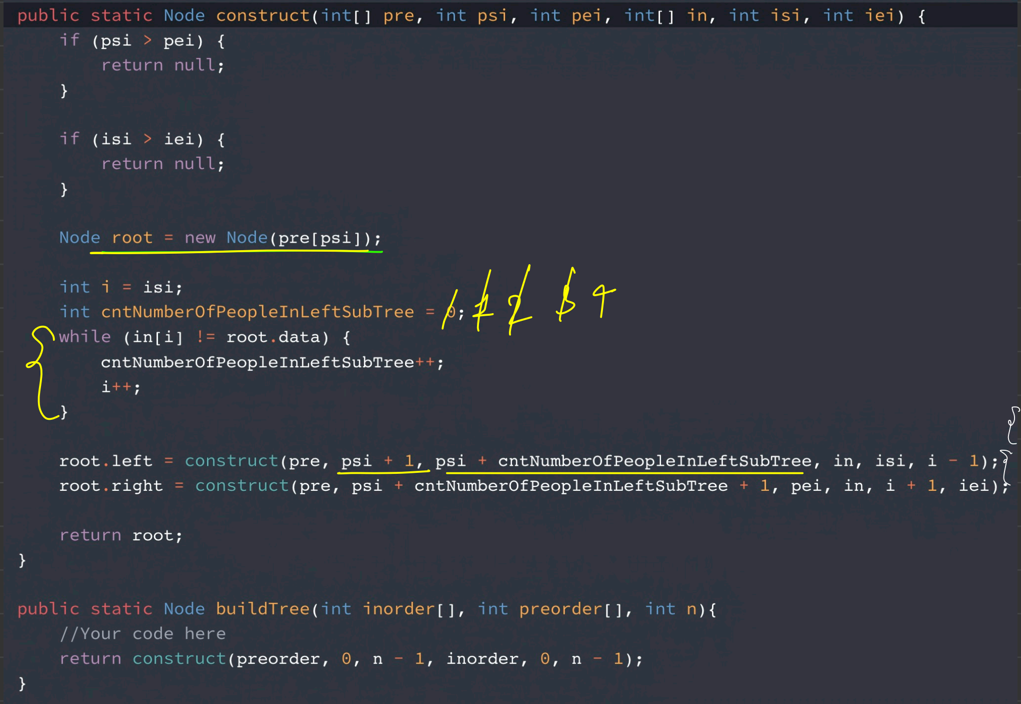
Pre : 10 20 40 50 80 30 60 90 70

In : 40 20 80 50 10 60 90 30 70

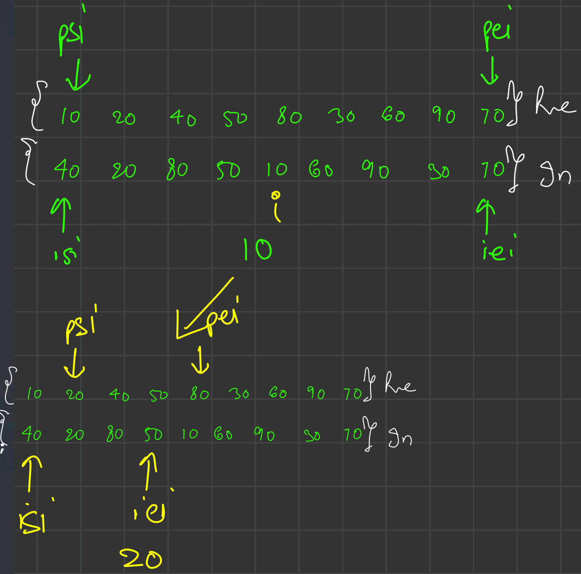
left root right



0 1 2 3 4
{10 20 40 50 80 30 60 90 70} Pre
{40 20 80 50 10 60 90 30 70} In
i
10
lst
rst
Pre {20, 40, 50, 80}
In {40, 20, 80, 50}
lst
rst
20
{30, 60, 90, 70} Pre
{60, 90, 30, 70} In
30
{40}
{40}
40
{50, 80}
{80, 50}
50
{80}
{80}
80
{60, 90}
{60, 90}
60
{90}
{90}
90
{70}
{70}
70

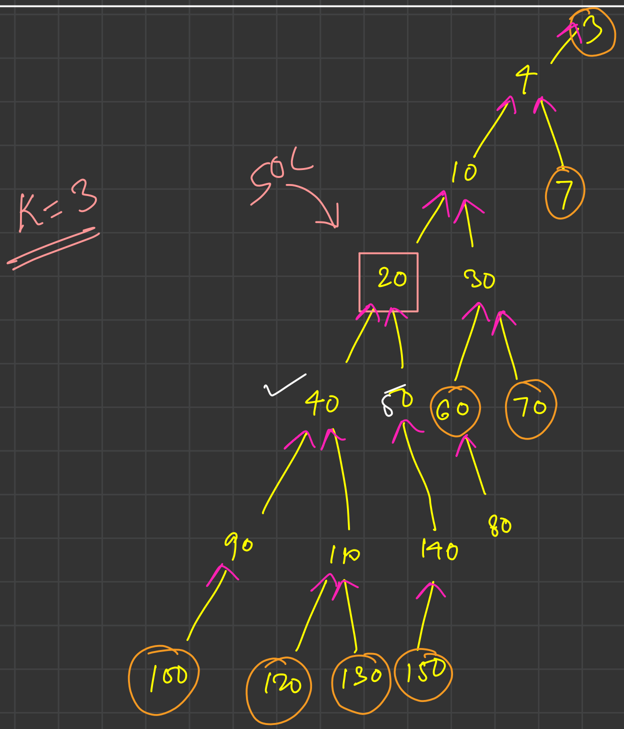


```java
public static Node construct(int[] pre, int psi, int pei, int[] in, int isi, int iei) {
    if (psi > pei) {
        return null;
    }

    if (isi > iei) {
        return null;
    }

    Node root = new Node(pre[psi]);

    int i = isi;
    int cntNumberOfPeopleInLeftSubTree = 0;     ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4
    while (in[i] != root.data) {
        cntNumberOfPeopleInLeftSubTree++;
        i++;
    }

    root.left = construct(pre, psi + 1, psi + cntNumberOfPeopleInLeftSubTree, in, isi, i - 1);
    root.right = construct(pre, psi + cntNumberOfPeopleInLeftSubTree + 1, pei, in, i + 1, iei);

    return root;
}

public static Node buildTree(int inorder[], int preorder[], int n){
    //Your code here
    return construct(preorder, 0, n - 1, inorder, 0, n - 1);
}
```



psi
pei
10 20 40 50 80 30 60 90 70 Pre
40 20 80 50 10 60 90 30 70 In
isi
i
10
iei
psi'
pei'
10 20 40 50 80 30 60 90 70 Pre
40 20 80 50 10 60 90 30 70 In
isi'
iei'
20

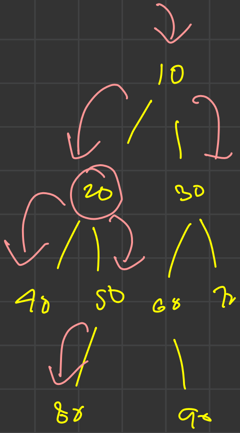
All Nodes Distance K in Binary Tree



{100, 120, 130, 150, 60, 70, 3, 7}

~~20~~ ~~40~~, ~~50~~, ~~10~~ ~~90~~, ~~110~~, ~~140~~, ~~30~~, ~~4~~

$\boxed{100, 120, 130, 150, 60, 70, 7, 3}$



HashMap

child	Parent
20	$\longrightarrow$ 10
30	$\longrightarrow$ 10
40	$\longrightarrow$ 20
50	$\longrightarrow$ 20
80	$\longrightarrow$ 50
60	$\longrightarrow$ 30
70	$\longrightarrow$ 30