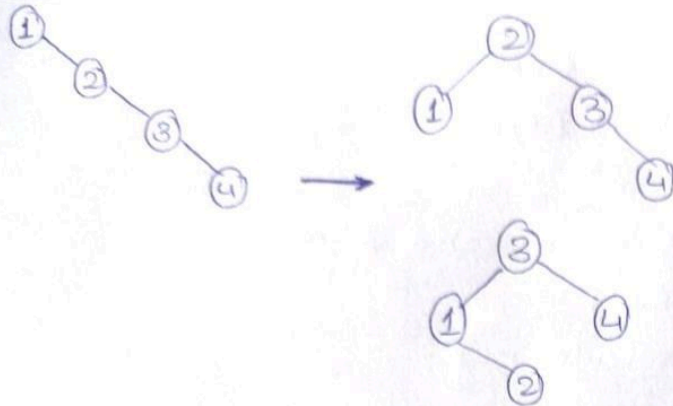
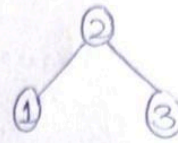


1382 Balance a Binary Search TreeMediumExample 1:

i/p = root = [1, null, 2, null, 3, null, 4, null, null]

o/p: [2, 1, 3, null, null, null, 4]

Example 2:

i/p: root = [2, 1, 3]

o/p: [2, 1, 3]

- (i) this is not just checking balance
- (ii) you must rebuild the BST structure

Approach: Brute

- (i) Traverse the BST and store all nodes values.
- (ii) Sort the values.
- (iii) Build a balanced BST from the sorted list.

Pseudocode:~~func~~

```

function balanceBST(root):
    values = empty list
    traverse (root, values)
    sort (values)
    return buildBST (values, 0, values.length-1)

```

```

function traverse (node, values):
    if node is null:
        return
    traverse (node.left, values)
    values.add (node.val)
    traverse (node.right, values)

```

```

function buildBST (values, start, end):

```

```

    if start > end:
        return null

```

```

    mid = (start+end) / 2

```

```

    node = new TreeNode (
        values [mid])

```

```

    node.left = buildBST (values,
        start, mid-1)

```

```

    node.right = buildBST (values,
        mid+1, end)

```

```

    return node

```

Time Complexity : (i) Traversal $\rightarrow O(N)$
(ii) Sorting $\rightarrow O(N \log N)$
(iii) Tree build $\rightarrow O(N)$

Total TC $\rightarrow O(N \log N)$

Space Complexity : (i) Array storage $\rightarrow O(N)$
(ii) Recursion stack $\rightarrow O(N)$

Approach : Optimal

- (i) Do in-order traversal of BST \rightarrow sorted sequence
- (ii) store nodes directly (not values)
- (iii) use divide-and-conquer to rebuild BST

Pseudocode :

function balance BST(root):

nodes = empty list

inorder(root, nodes)

return buildTree(nodes, 0, nodes.length - 1)

function inorder(node, nodes):

if node is null:

return

inorder(node.left, nodes)

nodes.add(node)

inorder(node.right, nodes)

function buildTree(nodes, start, end):

if start > end:

return null

mid = (start + end) / 2

root = nodes[mid]

root.left = buildTree(nodes, start, mid - 1)

root.right = buildTree(nodes, mid + 1, end)

return root

Time Complexity : (i) Inorder traversal $\rightarrow O(N)$

(ii) Rebuild tree $\rightarrow O(N)$

Total $\rightarrow O(N)$

Space Complexity : (i) Array list $\rightarrow O(N)$

(ii) Recursive stack $\rightarrow O(N)$