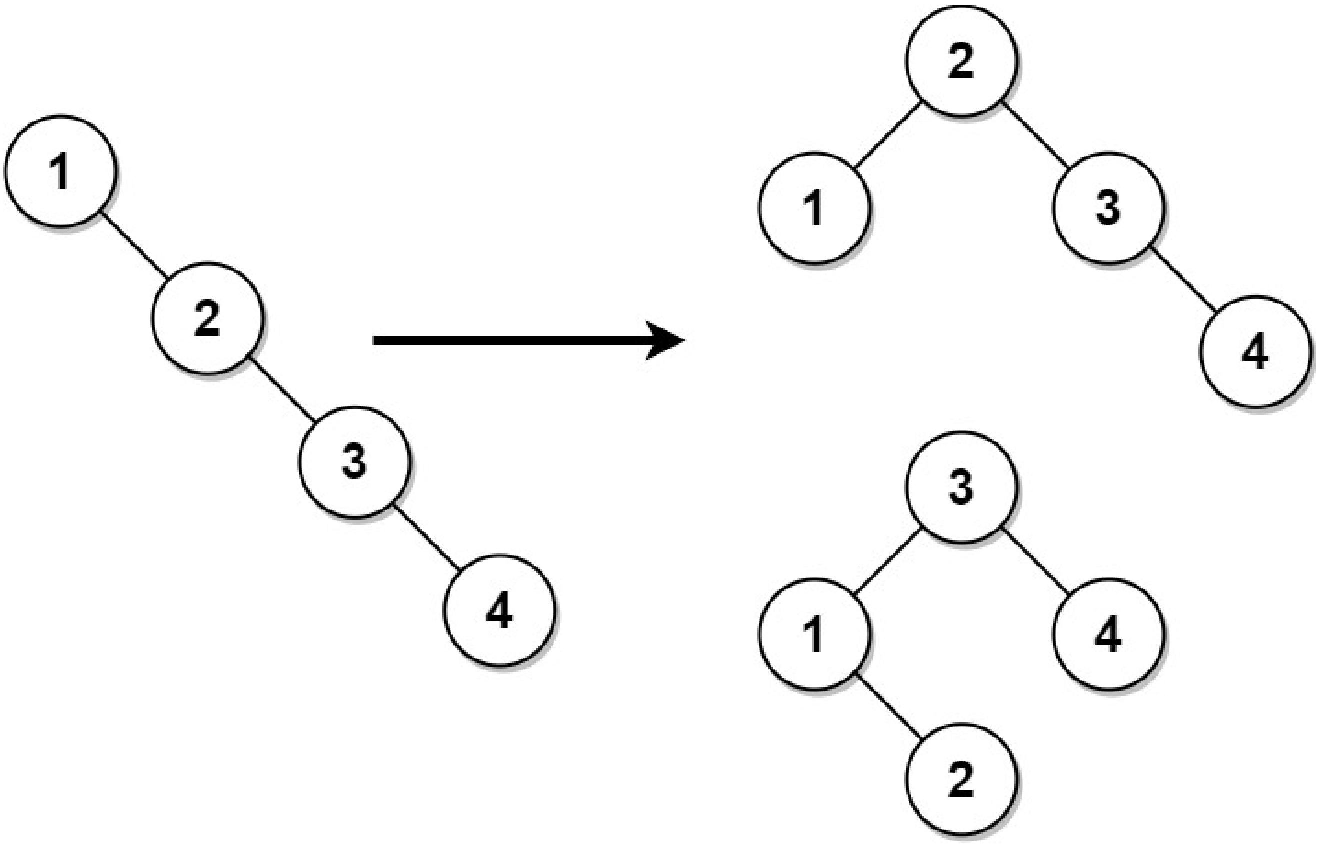
1328. Balance a Binary Search Tree

Given the `root` of a binary search tree, return *a **balanced** binary search tree with the same node values*. If there is more than one answer, return **any of them**.

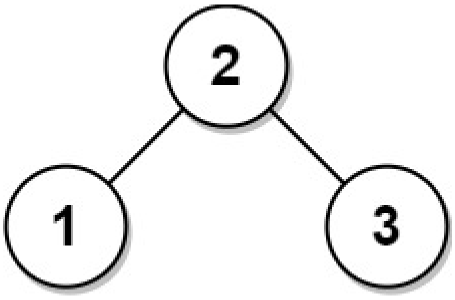
A binary search tree is **balanced** if the depth of the two subtrees of every node never differs by more than `1`.

Example 1:



Input: root = [1,null,2,null,3,null,4,null,null]
Output: [2,1,3,null,null,null,4]
Explanation: This is not the only correct answer, [3,1,4,null,2] is also correct.

Example 2:



Input: root = [2,1,3]
Output: [2,1,3]

Constraints:

- The number of nodes in the tree is in the range `[1, 104]`.
- `1 <= Node.val <= 105`

Solution:

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode() {}
 *     TreeNode(int val) { this.val = val; }
 *     TreeNode(int val, TreeNode left, TreeNode right) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * }
 */
class Solution {
    public TreeNode balanceBST(TreeNode root) {
        List<Integer> sortedElements = new ArrayList<>();
        inOrderTraversal(root, sortedElements);
        return buildBalancedBST(sortedElements, 0, sortedElements.size() -
1);

    }

    private void inOrderTraversal(TreeNode node, List<Integer>
sortedElements) {

        if (node == null) {

            return;

        }

        inOrderTraversal(node.left, sortedElements);

        sortedElements.add(node.val);

        inOrderTraversal(node.right, sortedElements);

    }

    private TreeNode buildBalancedBST(List<Integer> elements, int start, int
end) {

        if (start > end) {

            return null;

        }

        int mid = start + (end - start) / 2;

        TreeNode node = new TreeNode(elements.get(mid));

        node.left = buildBalancedBST(elements, start, mid - 1);

        node.right = buildBalancedBST(elements, mid + 1, end);

        return node;

    }

}



class TreeNode {

    int val;
```

```java
    TreeNode left;
    TreeNode right;
    TreeNode() {}
    TreeNode(int val) { this.val = val; }
    TreeNode(int val, TreeNode left, TreeNode right) {
        this.val = val;
        this.left = left;
        this.right = right;
    }
}
```