Assignment 6

Total 80 points
Each question carries 10 marks

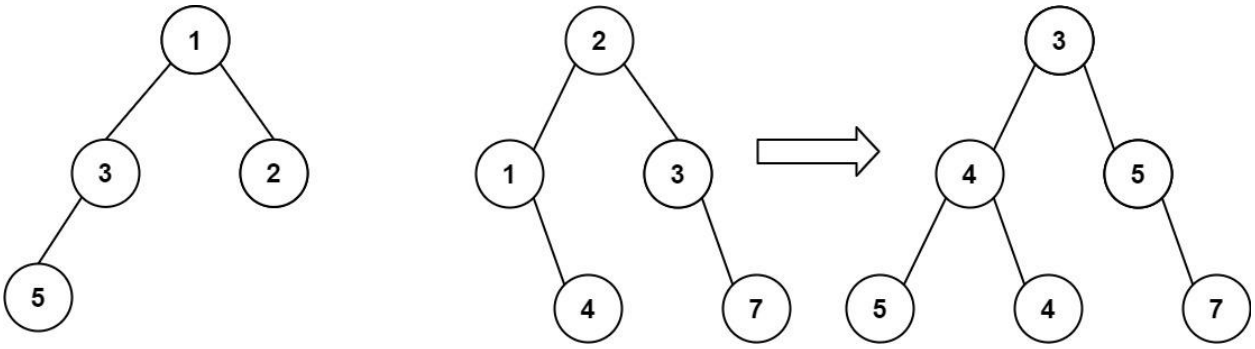
Question 1: You are given two binary trees root1 and root2.

Imagine that when you put one of them to cover the other, some nodes of the two trees are overlapped while the others are not. You need to merge the two trees into a new binary tree. The merge rule is that if two nodes overlap, then sum node values up as the new value of the merged node. Otherwise, the NOT null node will be used as the node of the new tree.

Return *the merged tree*.

Note: The merging process must start from the root nodes of both trees.

Example 1:



Input: root1 = [1,3,2,5], root2 = [2,1,3,null,4,null,7]

Output: [3,4,5,5,4,null,7]

Example 2:

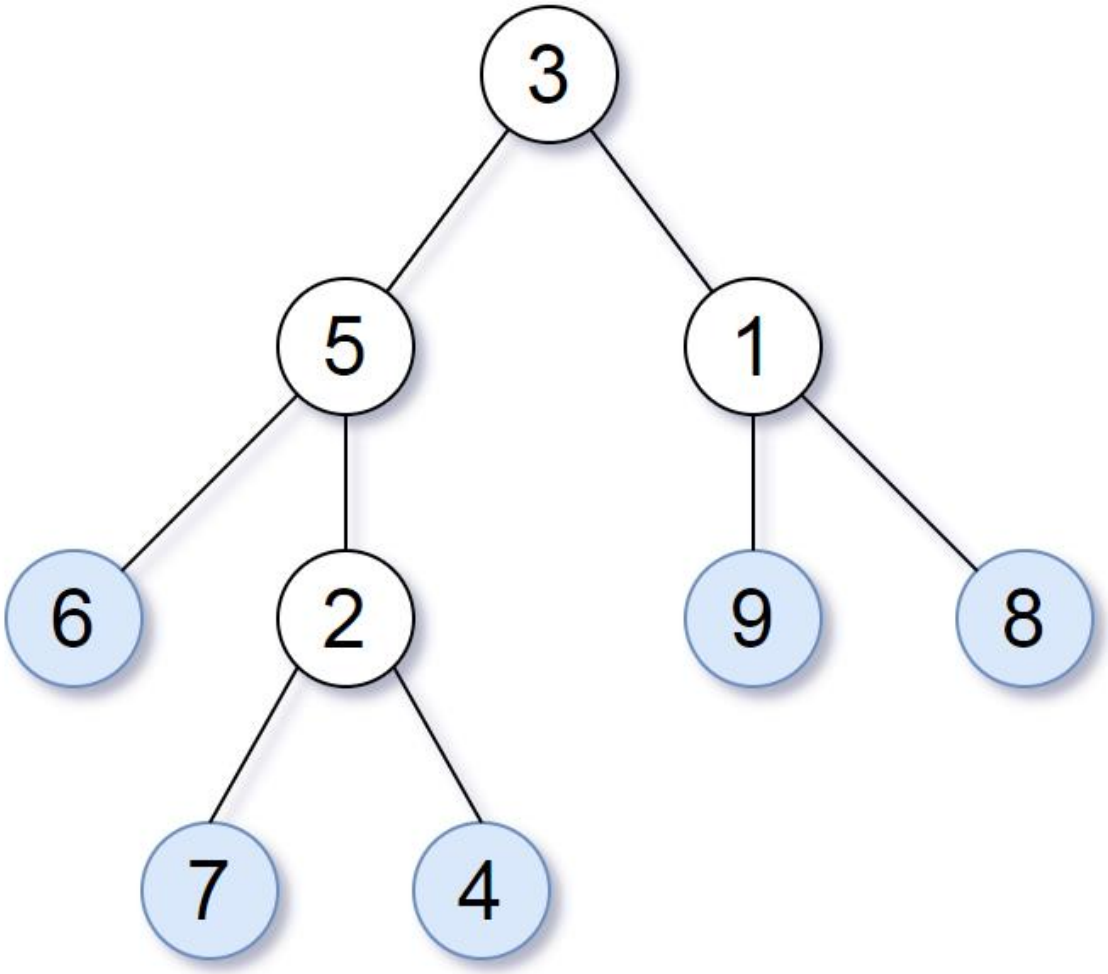
Input: root1 = [1], root2 = [1,2]

Output: [2,2]

Constraints:

- The number of nodes in both trees is in the range [0, 2000].
- -10^4 <= Node.val <= 10^4

Question 2: Consider all the leaves of a binary tree, from left to right order, the values of those leaves form a **leaf value sequence**.

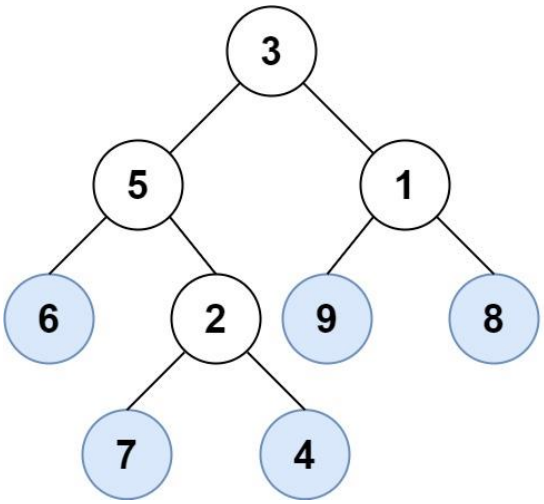


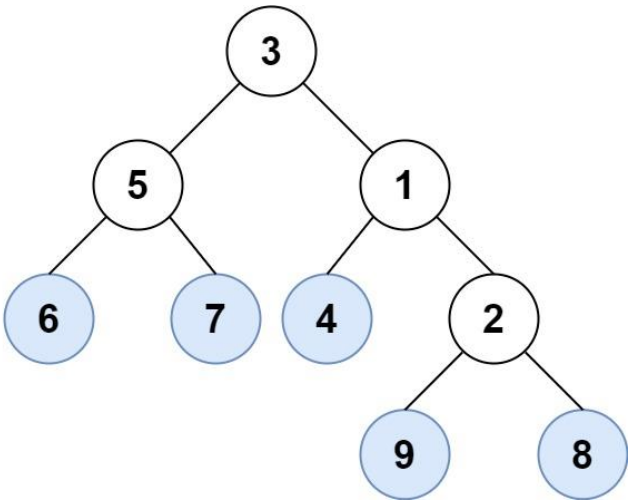
For example, in the given tree above, the leaf value sequence is `(6, 7, 4, 9, 8)`.

Two binary trees are considered *leaf-similar* if their leaf value sequence is the same.

Return `true` if and only if the two given trees with head nodes `root1` and `root2` are leaf-similar.

Example 1:





Input: root1 = [3,5,1,6,2,9,8,null,null,7,4], root2 = [3,5,1,6,7,4,2,null,null,null,null,null,null,9,8]

Output: true

Example 2:

Input: root1 = [1], root2 = [1]

Output: true

Example 3:

Input: root1 = [1], root2 = [2]

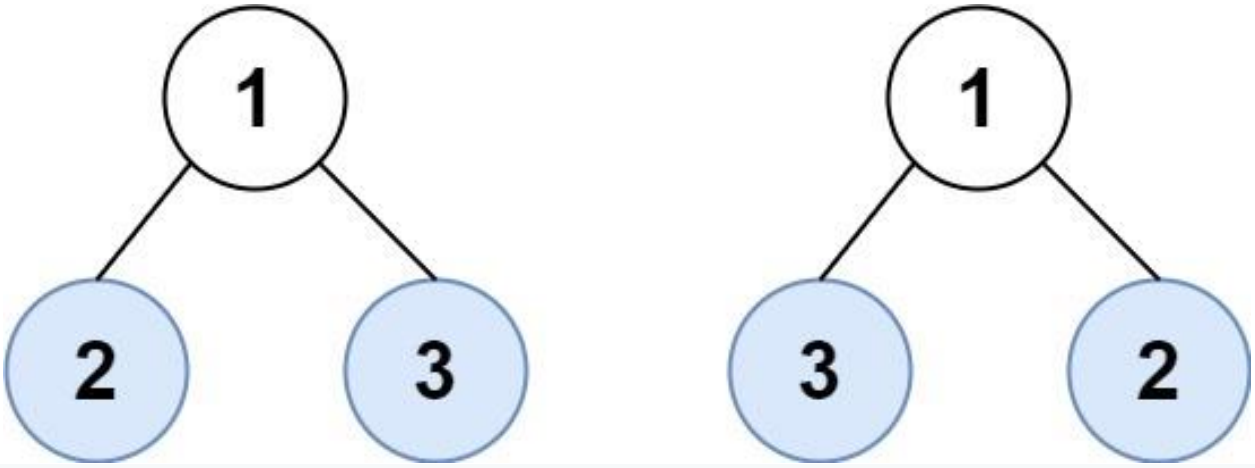
Output: false

Example 4:

Input: root1 = [1,2], root2 = [2,2]

Output: true

Example 5:



Input: root1 = [1,2,3], root2 = [1,3,2]

Output: false

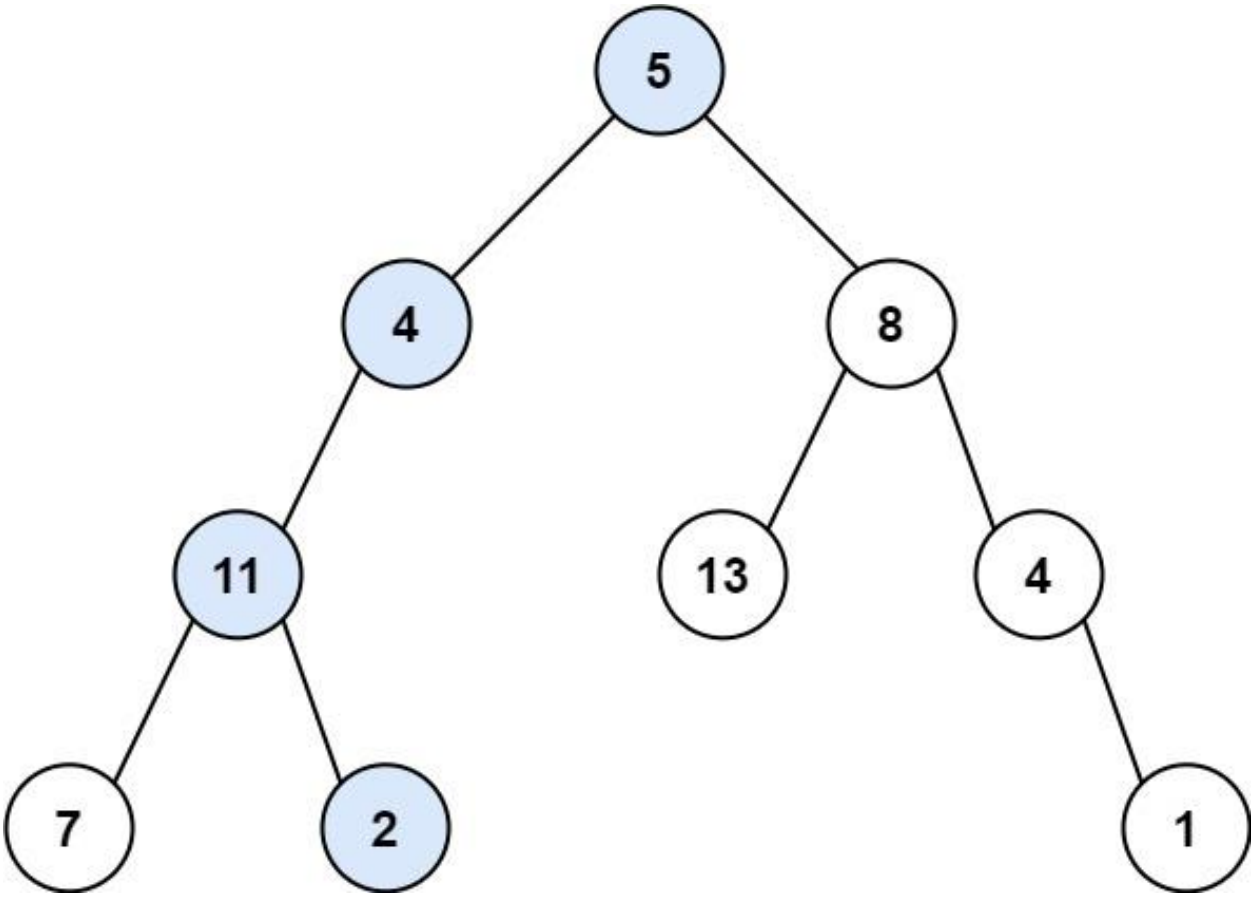
Constraints:

- The number of nodes in each tree will be in the range [1, 200].
- Both of the given trees will have values in the range [0, 200].

Question 3: Given the root of a binary tree and an integer targetSum, return true if the tree has a **root-to-leaf** path such that adding up all the values along the path equals targetSum.

A **leaf** is a node with no children.

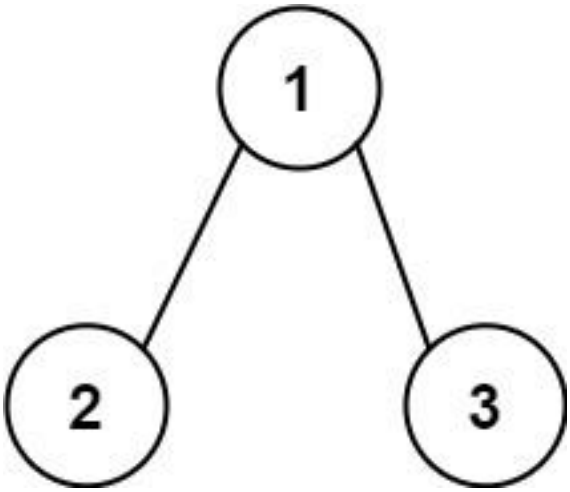
Example 1:



Input: root = [5,4,8,11,null,13,4,7,2,null,null,null,1], targetSum = 22

Output: true

Example 2:



Input: root = [1,2,3], targetSum = 5

Output: false

Example 3:

Input: root = [1,2], targetSum = 0

Output: false

Constraints:

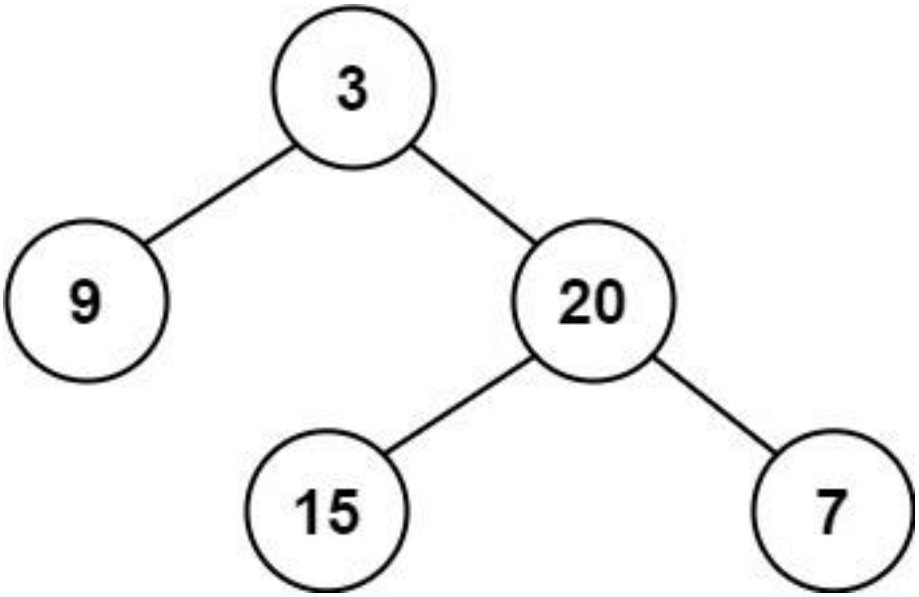
- The number of nodes in the tree is in the range [0, 5000].
- -1000 <= Node.val <= 1000
- -1000 <= targetSum <= 1000

Question 4: Given a binary tree, determine if it is height-balanced.

For this problem, a height-balanced binary tree is defined as:

a binary tree in which the left and right subtrees of *every* node differ in height by no more than 1.

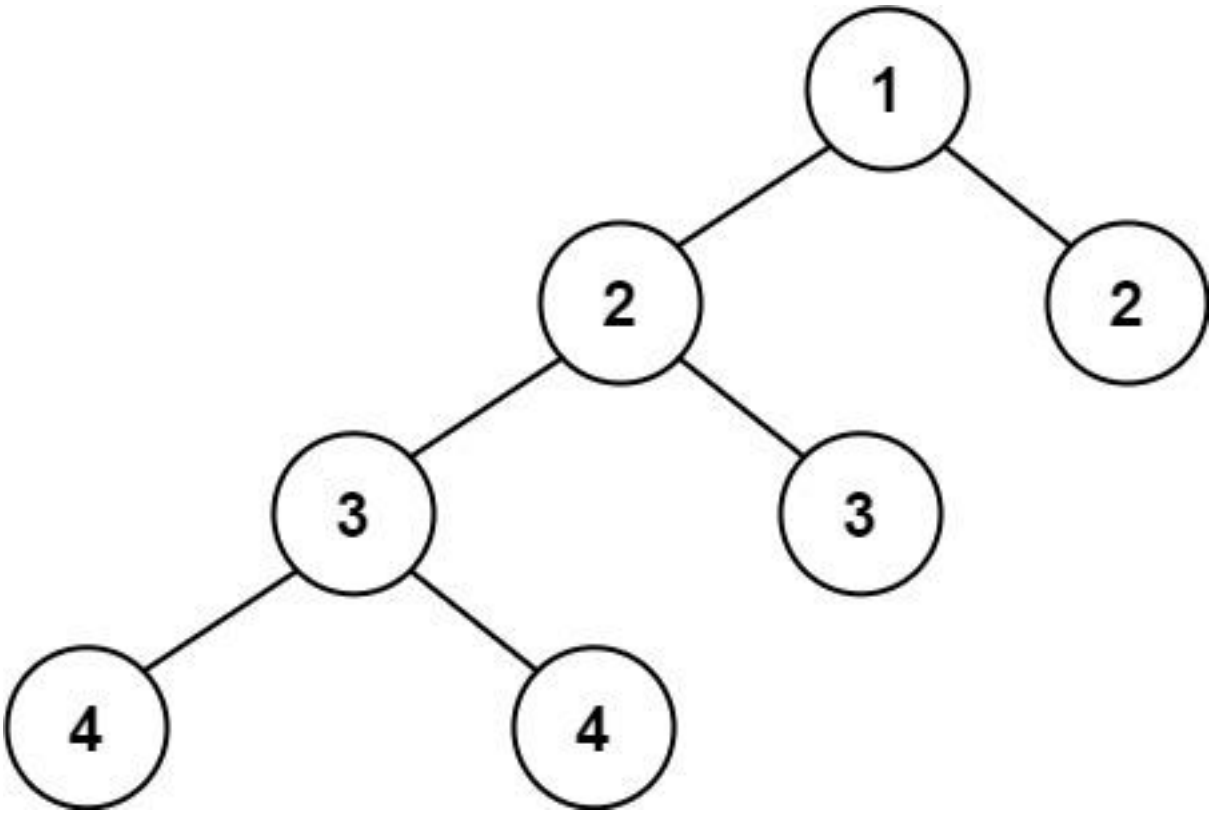
Example 1:



Input: root = [3,9,20,null,null,15,7]

Output: true

Example 2:



Input: root = [1,2,2,3,3,null,null,4,4]

Output: false

Example 3:

Input: root = []

Output: true

Constraints:

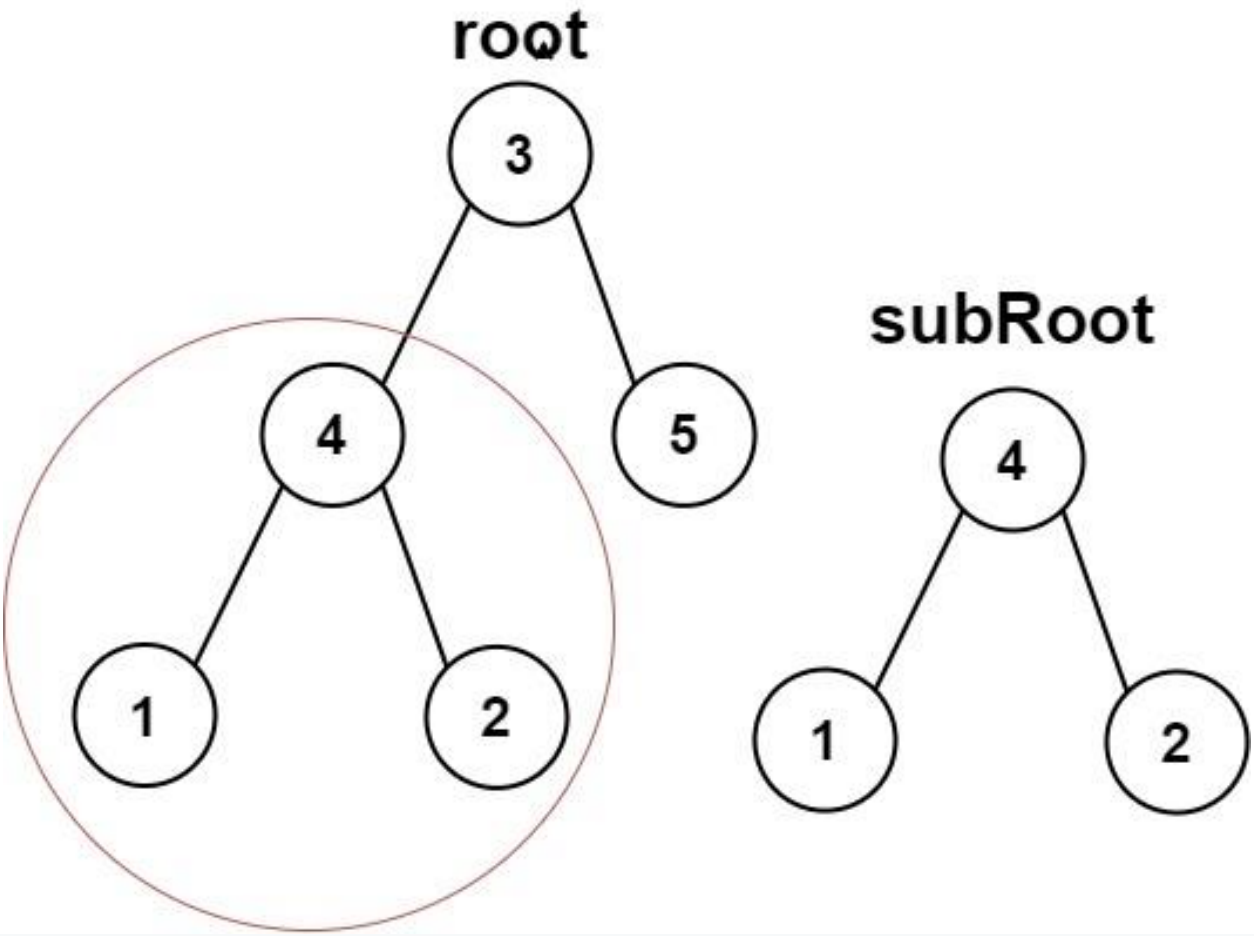
- The number of nodes in the tree is in the range [0, 5000].
- $-10^4 <= Node.val <= 10^4$

Question 5:

Given the roots of two binary trees root and subRoot, return true if there is a subtree of root with the same structure and node values of subRoot and false otherwise.

A subtree of a binary tree tree is a tree that consists of a node in tree and all of this node's descendants. The tree tree could also be considered as a subtree of itself.

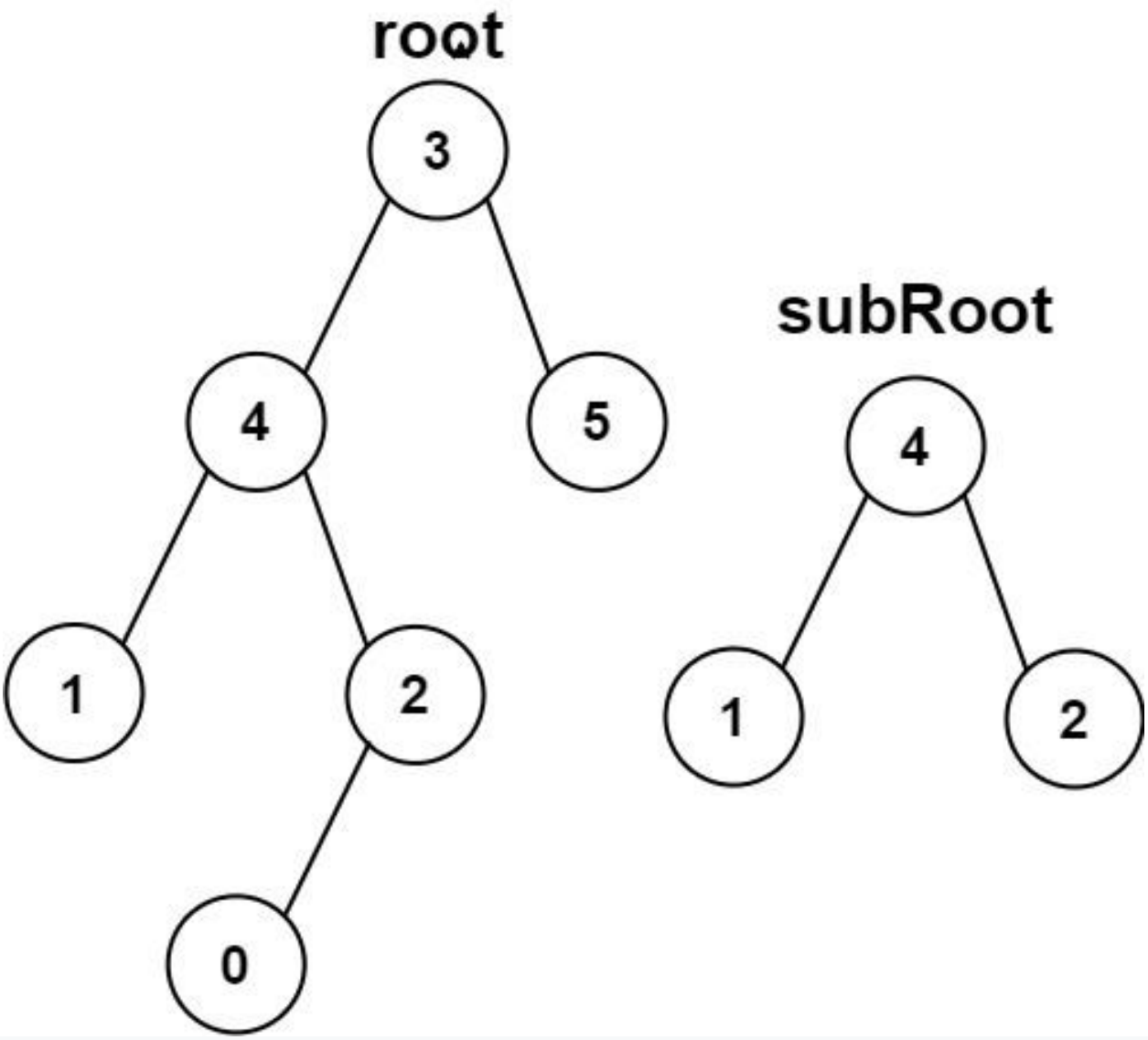
Example 1:



Input: root = [3,4,5,1,2], subRoot = [4,1,2]

Output: true

Example 2:



Input: root = [3,4,5,1,2,null,null,null,null,0], subRoot = [4,1,2]

Output: false

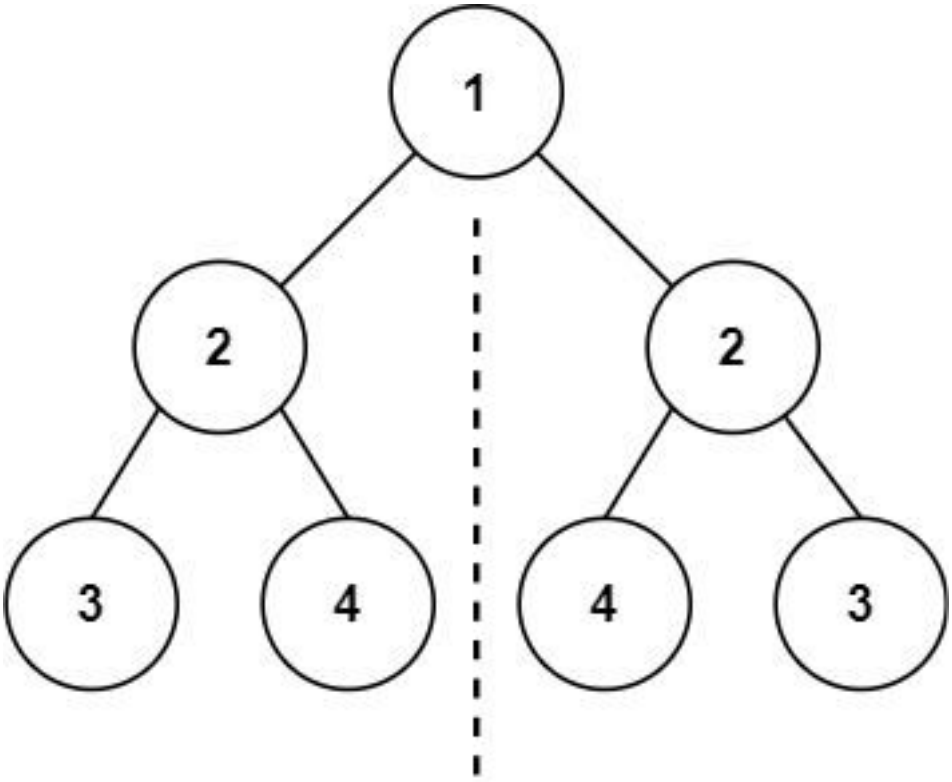
Constraints:

- The number of nodes in the `root` tree is in the range `[1, 2000]`.
- The number of nodes in the `subRoot` tree is in the range `[1, 1000]`.
- -10^4 <= root.val <= 10^4

- -10^4 <= subRoot.val <= 10^4

Question 6: Given the root of a binary tree, *check whether it is a mirror of itself* (i.e., symmetric around its center).

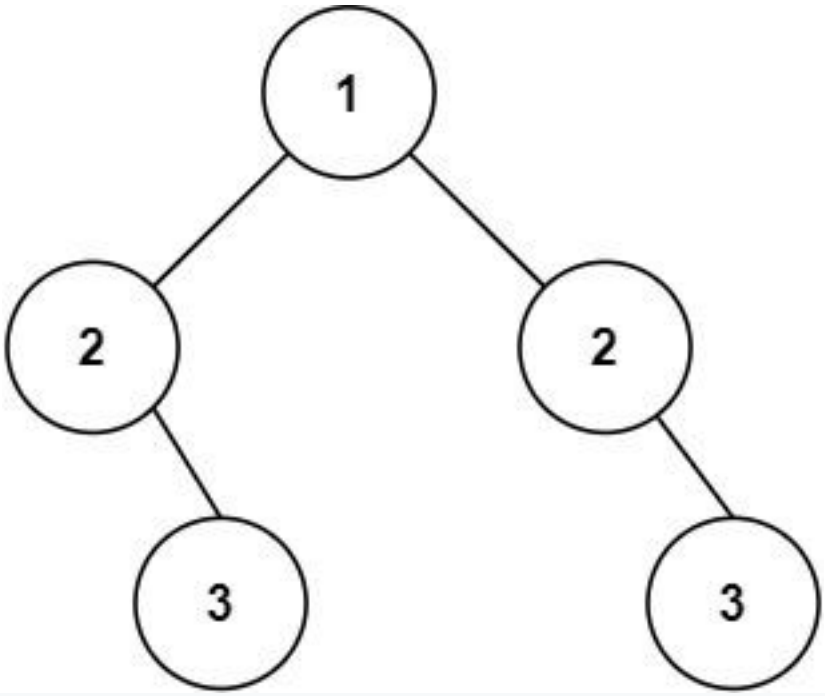
Example 1:



Input: root = [1,2,2,3,4,4,3]

Output: true

Example 2:



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

Output: false

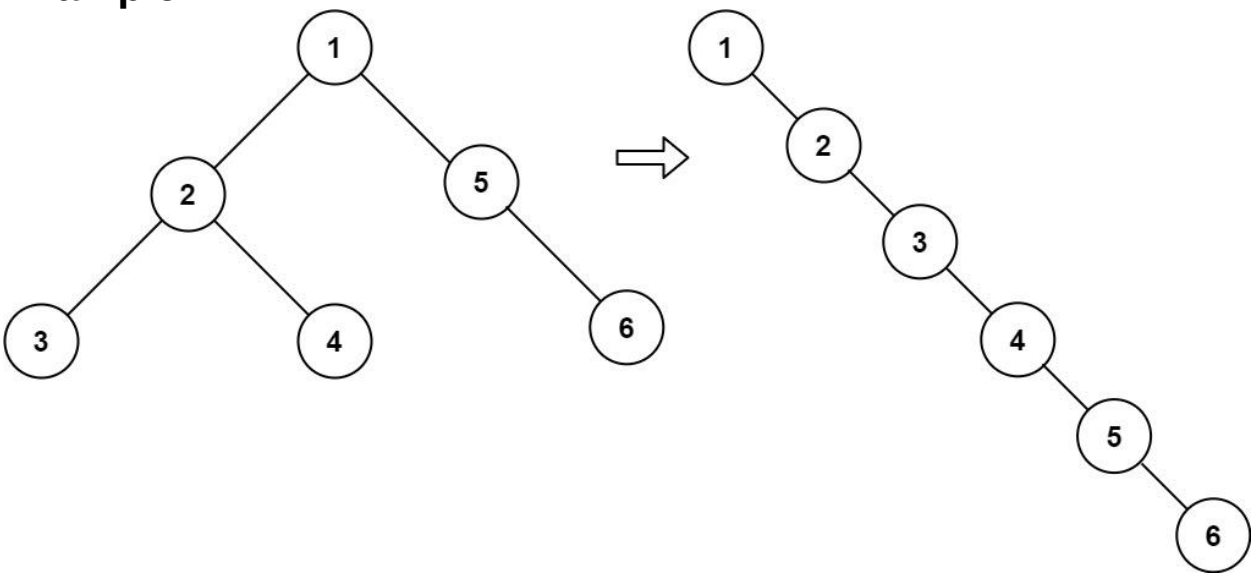
Constraints:

- The number of nodes in the tree is in the range [1, 1000].
- -100 <= Node.val <= 100

Question 7: Given the root of a binary tree, flatten the tree into a "linked list":

- The "linked list" should use the same TreeNode class where the right child pointer points to the next node in the list and the left child pointer is always null.
- The "linked list" should be in the same order as a **pre-order traversal** of the binary tree.

Example 1:



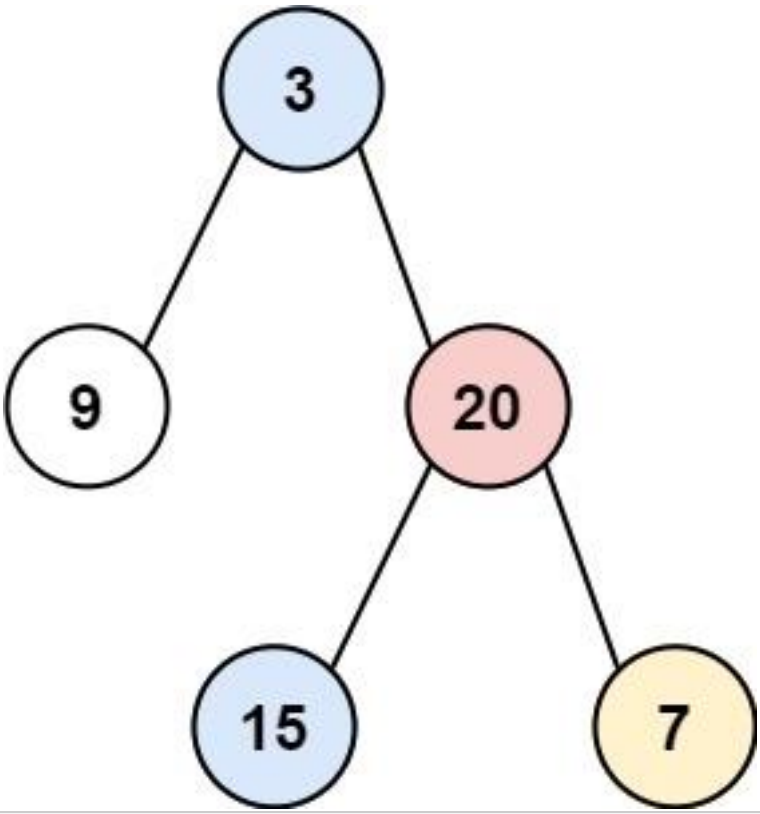
Input: root = [1,2,5,3,4,null,6]
Output: [1,null,2,null,3,null,4,null,5,null,6]
Example 2:
Input: root = []
Output: []
Example 3:
Input: root = [0]
Output: [0]

Constraints:

- The number of nodes in the tree is in the range [0, 2000].
- -100 <= Node.val <= 100

Question 8: Given the root of a binary tree, return ***the vertical order traversal*** *of its nodes' values*. (i.e., from top to bottom, column by column).
If two nodes are in the same row and column, the order should be from **left to right**.

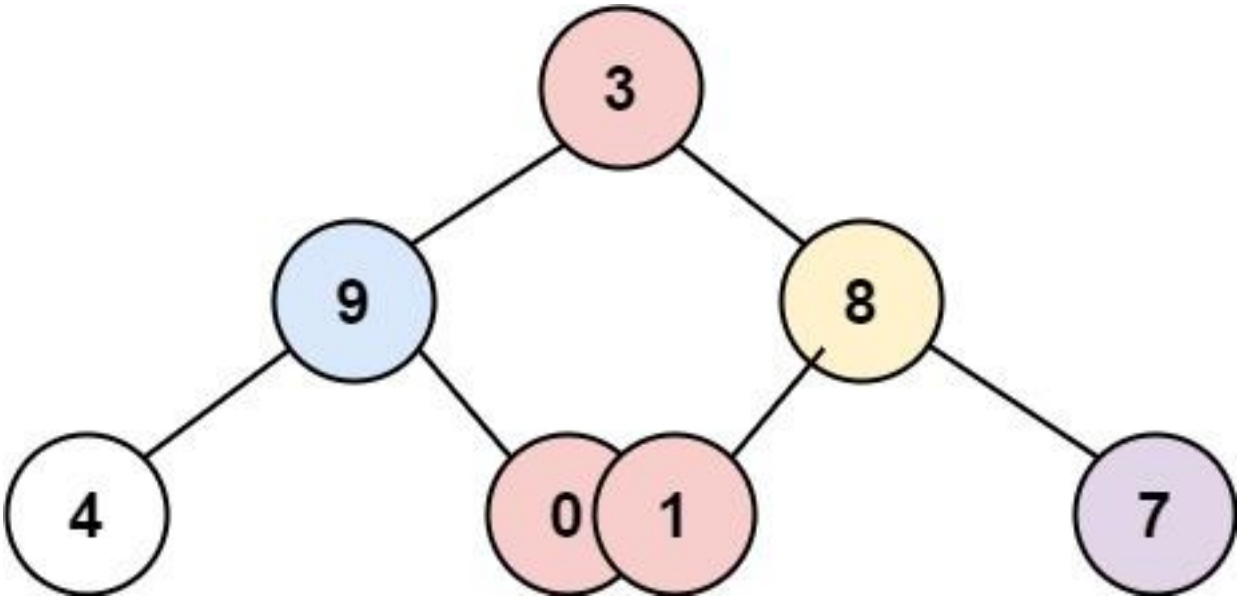
Example 1:



Input: root = [3,9,20,null,null,15,7]

Output: [[9],[3,15],[20],[7]]

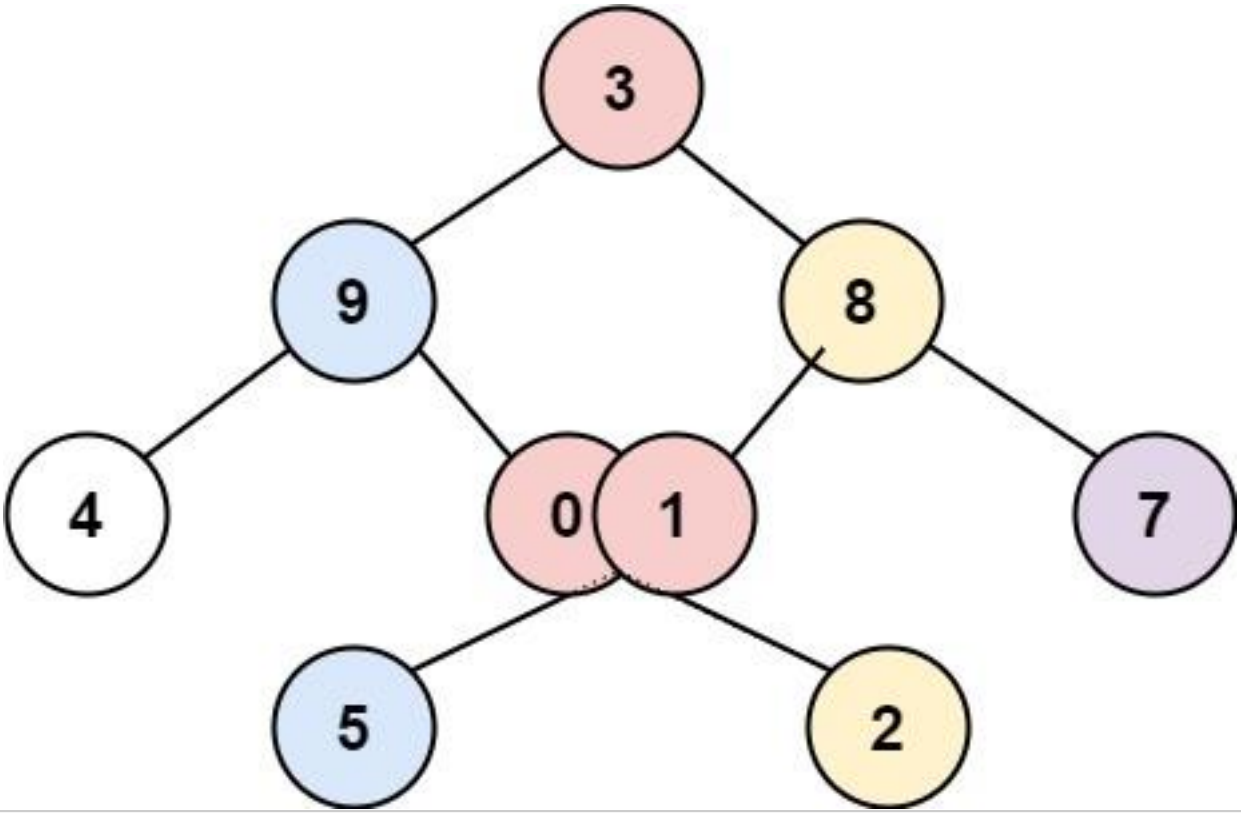
Example 2:



Input: root = [3,9,8,4,0,1,7]

Output: [[4],[9],[3,0,1],[8],[7]]

Example 3:



Input: root = [3,9,8,4,0,1,7,null,null,null,2,5]

Output: [[4],[9,5],[3,0,1],[8,2],[7]]

Example 4:

Input: root = []

Output: []

Constraints:

- The number of nodes in the tree is in the range `[0, 100]`.
- `-100 <= Node.val <= 100`