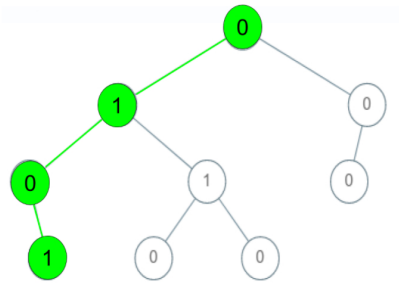
Check If a String Is a Valid Sequence from Root to Leaves Path in a Binary Tree

Given a binary tree where each path going from the root to any leaf form a **valid sequence**, check if a given string is a **valid sequence** in such binary tree.

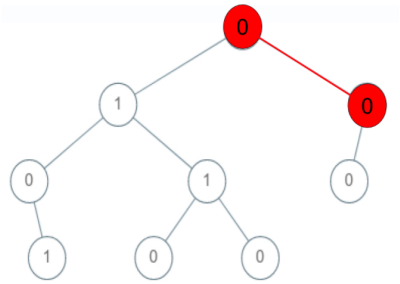
We get the given string from the concatenation of an array of integers `arr` and the concatenation of all values of the nodes along a path results in a **sequence** in the given binary tree.

Example 1:



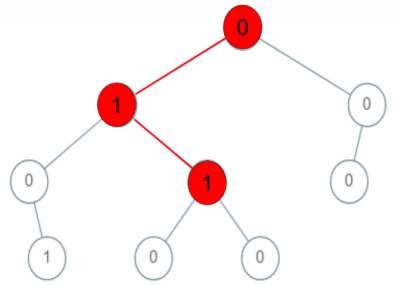

```
Input: root = [0,1,0,0,1,0,null,null,1,0,0], arr = [0,1,0,1]
Output: true
Explanation:
The path 0 -> 1 -> 0 -> 1 is a valid sequence (green color in the figure).
Other valid sequences are:
0 -> 1 -> 1 -> 0
0 -> 0 -> 0
```

Example 2:




```
Input: root = [0,1,0,0,1,0,null,null,1,0,0], arr = [0,0,1]
Output: false
Explanation: The path 0 -> 0 -> 1 does not exist, therefore it is not even a sequence.
```

Example 3:




```
Input: root = [0,1,0,0,1,0,null,null,1,0,0], arr = [0,1,1]
Output: false
Explanation: The path 0 -> 1 -> 1 is a sequence, but it is not a valid sequence.
```

Constraints:

- `1 <= arr.length <= 5000`
- `0 <= arr[i] <= 9`
- Each node's value is between [0 - 9].

Hide Hint #1

Depth-first search (DFS) with the parameters: current node in the binary tree and current position in the array of integers.

Hide Hint #2

When reaching at final position check if it is a leaf node.

Java

```java
/**
 * 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 boolean isValidSequence(TreeNode root, int[] arr) {
        if(root == null) return false;
        if(root.val != arr[0]) return false;
        return helper(root, arr, 0);
    }

    private boolean helper(TreeNode root, int[] arr, int index) {
        if(index == arr.length-1 && root.left == null && root.right == null) return true;
        if(index == arr.length-1) return false;
        boolean left = false;
        boolean right = false;
        if(root.left != null && root.left.val == arr[index+1]) {
            left = helper(root.left, arr, index+1);
        }
        if(root.right != null && root.right.val == arr[index+1]) {
            right = helper(root.right, arr, index+1);
        }

        return left || right;

    }
}
```

Custom Testcase (Contribute)

Run Code Submit

Submission Result: Accepted More Details >

Share your acceptance!

18