35. 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

PROGRAM:

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right


def isValidSequence(root, arr):
    def dfs(node, idx):
        # Base cases:
        if not node:
            return False
        if idx == len(arr) - 1:
            return node.val == arr[idx] and not node.left and not node.right
        if node.val == arr[idx]:


            return dfs(node.left, idx + 1) or dfs(node.right, idx + 1)
        else:
            return False


    return dfs(root, 0)


root = TreeNode(0)
root.left = TreeNode(1)
```

```
root.right = TreeNode(0)
root.left.left = TreeNode(0)
root.left.right = TreeNode(1)
root.right.left = TreeNode(0)
root.right.left.right = TreeNode(1)
root.right.right = TreeNode(0)

arr1 = [0, 1, 0, 1]
arr2 = [0, 0, 1]
arr3 = [0, 1, 1]

print(isValidSequence(root, arr1))
print(isValidSequence(root, arr2))
print(isValidSequence(root, arr3))
```

OUTPUT:

```
False
False
True
```

TIME COMPLEXITY: O(n)