DAY: 30 [Explained Solution will be available soon]

Follow me for more interesting programming questions:

https://www.github.com/shubhamthrills

https://www.instagaram.com/shubhamthrills

https://www.linkedin.com/in/shubhamsagar

DFS Approach

Python

Time: `O(N)`
Space: `O(H)`, where `H` is the height of the given tree

```
class Solution:

    def isValidSequence(self, root: TreeNode, arr: List[int]) -> bool:

        def is_valid(node: TreeNode, i: int) -> bool:

            if not node or node.val != arr[i]: return False

            if i == len(arr) - 1: return not node.left and not node.right

            return is_valid(node.left, i + 1) or is_valid(node.right, i + 1)

        return is_valid(root, 0)
```

C++

```
class Solution {

public:

    bool isValidSequence(TreeNode* root, vector<int> arr) {

        if(root==NULL) return false;

        if(arr.empty()) return false;

        if(root->val != arr[0]) return false;

        arr.erase(arr.begin());

        if(root->left == NULL && root->right == NULL  && arr.size()==0) return true;

        return (isValidSequence(root->left,arr) || isValidSequence(root->right,arr));

    }

};
```

Follow me for more interesting programming questions:

https://www.github.com/shubhamthrills

https://www.instagaram.com/shubhamthrills

https://www.linkedin.com/in/shubhamsagar

JAVA

```
class Solution {

    public boolean isValidSequence(TreeNode root, int[] arr) {

        return helper(root,arr,0);

    }


    private boolean helper(TreeNode root,int[] arr,int idx){

        if(root == null) return false;

        if(arr[idx] != root.val) return false;

        if(idx + 1 == arr.length) return root.left == null && root.right == null;

        return helper(root.left,arr,idx + 1) || helper(root.right,arr,idx + 1);

    }


}
```

Python[My Approch]

```
class Solution:
    def isValidSequence(self, root: TreeNode, arr: List[int],index=0,n=0) -> bool:
        n = len(arr)
        def visit(node,index):
            if index==n-1:
                if node is not None and node.val==arr[index]:
                    return node.left is None and node.right is None
                return False
            if node is None:
                return False
            if node.val==arr[index]:
                return visit(node.left,index+1) or visit(node.right,index+1)
            return False
        return visit(root,0)
```

Follow me for more interesting programming questions:

https://www.github.com/shubhamthrills

https://www.instagaram.com/shubhamthrills

https://www.linkedin.com/in/shubhamsagar