WangQiuc ★ 870 April 30, 2020 1:42 PM 558 VIEWS

4

If we find any one valid path, we return True. So dfs recursion should be:

```
# dfs(i, node)
if node.val == arr[i]:
    return dfs(i+1, node.left) or dfs(i+1, node.right)
```

Then we just need to figure out two base cases:

`return True`
We reach the end of arr and we reach a left node. And that `arr[-1] == node.val`, which gives us:

```
if i == n - 1 and not (node.left or node.right):
    return True
```

`return False`
When `arr[i] != node.val`, that path is invalid and there is no need to dfs on that path any more.
Or we finish iterating either the entire `arr`, or one of the tree path. But not **simultaneously**.
And we don't have to determine that **simultaneousness** because if it is, it will be caught in the previous dfs and return `True` or `False`.

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

And here is the code:

```
def isValidSequence(root, arr):
    n = len(arr)
    def dfs(i, node):
        if not node or i == n or arr[i] != node.val:
            return False
        if i == n - 1 and not (node.left or node.right):
            return True
        return dfs(i+1, node.left) or dfs(i+1, node.right)
    return dfs(0, root)
```

Comments: 3 Best | Most Votes | Newest to Oldest | Oldest to Newest

Type comment here... (Markdown is supported)

Post

DeliriumKlemens ★ 34 April 30, 2020 2:01 PM

Nice solution.
So worst case time complexity is O(size_of_tree) since we may need to traverse the whole tree and worst case space complexity is O(height_of_tree) for the call stack. Do you agree?

0 Show 3 replies Reply

poweric ★ 149 Last Edit: April 30, 2020 11:52 PM

Mine is similar as yours.

```
class Solution:
    def isValidSequence(self, root: TreeNode, arr: List[int]) -> bool:
        def dfs(node, i):
            if not node or i == len(arr) or node.val != arr[i]:
                return False
            # at leaf node
            if not node.left and not node.right:
                return i == len(arr) - 1
            return dfs(node.left, i+1) or dfs(node.right, i+1)

        return dfs(root, 0)
```

0 Reply

newRuanXY ★ 75 April 30, 2020 2:25 PM

nice code. and there is a non-recursive edition: https://leetcode.com/explore/challenge/card/30-day-leetcoding-challenge/532/week-5/3315/discuss/604499/summary-3-solutions-python-O(N)

0 Reply