

Step 1: Answer the [questions](#) about [Tree Traversal](#)

In-order Traversal: 1, 3, 4, 5, 6, 7, 8, 10, 13, 14


Pre-order Traversal: 8, 3, 1, 6, 4, 5, 7, 10, 14, 13

Post-order Traversal: 1, 5, 4, 7, 6, 3, 13, 14, 10, 8

Step 2: Study the answers from ChatGPT about "How to answer LeetCode question: '[100 Same Tree](#)' with the test case: Input: p =[1,2,3], q=[1,2,3]; Output: true"

To answer the LeetCode question "100 Same Tree" and test it with the given input case, you can use a recursive approach to check if two binary trees are the same. Here's the Python solution:

python

 Copy code

```
# Definition for a binary tree node
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def isSameTree(p: TreeNode, q: TreeNode) -> bool:
    # If both trees are empty, they are the same
    if not p and not q:
        return True

    # If one of the trees is empty or their values are different, they are not the same
    if not p or not q or p.val != q.val:
        return False

    # Recursively check if the left and right subtrees are the same
    return isSameTree(p.left, q.left) and isSameTree(p.right, q.right)

# Test the function with the given input case
p = TreeNode(1)
p.left = TreeNode(2)
p.right = TreeNode(3)

q = TreeNode(1)
q.left = TreeNode(2)
q.right = TreeNode(3)

output = isSameTree(p, q)
print(output)
```

The output will be `True`, indicating that the binary trees represented by `p` and `q` are the same.

In the solution, we define a recursive function `isSameTree` that takes two tree nodes `p` and `q` as input and returns a boolean value indicating if the trees are the same. We check if the current nodes have the same value, and if so, we recursively check if their left and right subtrees are also the same. The base cases handle the scenarios when one of the trees is empty or their values are different.

Step 3: Implement a Python solution for [100. Same Tree](#). - Easy – LC

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def isSameTree(self, p: Optional[TreeNode], q: Optional[TreeNode]) -> bool:
        if p and q:
            return (p.val == q.val and self.isSameTree(p.left, q.left) and self.isSameTree(p.right, q.right))
        return p is q
```

Console ^



Run

Submit



10:55 PM
6/5/2023

Step 4: Test all the test cases mentioned in '[100 Same Tree](#)'

Python3 | • Auto

```
1 # Definition for a binary tree node.
2 # class TreeNode:
3 #     def __init__(self, val=0, left=None, right=None):
4 #         self.val = val
5 #         self.left = left
6 #         self.right = right
7 class Solution:
8     def isSameTree(self, p: Optional[TreeNode], q: Optional[TreeNode]) -> bool:
9         if p and q:
10             return (p.val == q.val and self.isSameTree(p.left, q.left) and self.isSameTree(p.right, q.right))
11         return p is q
```

Testcase

Result

Accepted

Runtime: 73 ms

Case 1

Case 2

Case 3

Input

p =
[1,2,3]

q =
[1,2,3]

Output

true

Console

Run

Submit

10:57 PM
6/5/2023

i Python3 | Auto

{} ↺ ⚙ ↻ ↗

```
1 # Definition for a binary tree node.
2 # class TreeNode:
3 #     def __init__(self, val=0, left=None, right=None):
4 #         self.val = val
5 #         self.left = left
6 #         self.right = right
7 class Solution:
8     def isSameTree(self, p: Optional[TreeNode], q: Optional[TreeNode]) -> bool:
9         if p and q:
10             return (p.val == q.val and self.isSameTree(p.left, q.left) and self.isSameTree(p.right, q.right))
11         return p is q
```

Testcase

Result

Accepted Runtime: 73 ms

• Case 1 • Case 2 • Case 3

Input

p =
[1,2]

q =
[1,null,2]

Output

false

Expected

false

♥ Contribute a testcase

Console

⌕

Run

Submit



10:58 PM
6/5/2023

Python3 | * Auto

```
1 # Definition for a binary tree node.
2 # class TreeNode:
3 #     def __init__(self, val=0, left=None, right=None):
4 #         self.val = val
5 #         self.left = left
6 #         self.right = right
7 class Solution:
8     def isSameTree(self, p: Optional[TreeNode], q: Optional[TreeNode]) -> bool:
9         if p and q:
10             return (p.val == q.val and self.isSameTree(p.left, q.left) and self.isSameTree(p.right, q.right))
11         return p is q
```

Testcase

Result

Accepted

Runtime: 73 ms

Case 1

Case 2

Case 3

Input

p =
[1,2,1]

q =
[1,1,2]

Output

false

Expected

false

Contribute a testcase

Console

Run

Submit

10:59 PM
6/5/2023

Step 5: Optional homework: Practice more LeetCode questions about [Subject: Recursion](#)