Lists are Mutable

```
The problem wiht this code is:

The output should be [[1,2,5],[1,3]]
But i am getting [[1,2,5,3],[1,2,5,3]]

What is wrong?
```

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

class Solution:
    def __init__(self):
        self.op = []

    def binaryTreePaths(self, root: Optional[TreeNode]) -> List
        if not root:
            return

        cur_path = []
        self.dfs(root, cur_path)
        return self.op

    def dfs(self, root, cur_path):
        # Append the cur node to cur_path
        cur_path.append(root.val)

        # If leaf node
```

```
        if not root.left and not root.right:
            self.op.append(cur_path)
            return

        # If not leaf node
        if root.left:
            self.dfs(root.left, cur_path)

        if root.right:
            self.dfs(root.right, cur_path)

        return
```

What is wrong?

The issue lies in the way you handle the `cur_path` list when performing a depth-first traversal.
Since
lists are mutable in Python, when you append a value to `cur_path` and pass it recursively, all the recursive calls share the same list object.
This leads to unintended changes in the list during backtracking, causing your result to include incorrect paths.

```
"""
To fix this, you should create a new list when you pass cur_path
"""

class Solution:
    def __init__(self):
        self.op = []
```

```
    def binaryTreePaths(self, root: Optional[TreeNode]) -> List
        if not root:
            return []

        cur_path = []
        self.dfs(root, cur_path)
        return self.op

    def dfs(self, root, cur_path):
        # Create a new path list for the current recursion level
        cur_path = cur_path + [root.val] # Only change is here

        # If it's a leaf node, append the path to self.op
        if not root.left and not root.right:
            self.op.append(cur_path)
            return

        # Recur for the left and right children if they exist
        if root.left:
            self.dfs(root.left, cur_path)

        if root.right:
            self.dfs(root.right, cur_path)
```

Key Changes

- Instead of appending to `cur_path` directly, a new list, `cur_path + [root.val]` is created for each recursive call. This ensures that changes in one recursive call do not affect other calls.
- The logic for appending the current path to `self.op` remains the same.

Explanation

- By using `cur_path + [root.val]`, a new list is created for each recursion step, isolating the modifications in the current recursive path.

- This ensures that the `cur_path` used in one branch of the tree is not shared or corrupted by another branch.
