Problem 257: Binary Tree Paths

Problem Information

Difficulty: Easy
Acceptance Rate: 67.65%
Paid Only: No
Tags: String, Backtracking, Tree, Depth-First Search, Binary Tree

Problem Description

Given the `root` of a binary tree, return _all root-to-leaf paths in**any order**_.

A **leaf** is a node with no children.

Example 1:

Input: root = [1,2,3,null,5] **Output:** ["1->2->5","1->3"]

Example 2:

Input: root = [1] **Output:** ["1"]

Constraints:

* The number of nodes in the tree is in the range `[1, 100]`. * `-100 <= Node.val <= 100`

Code Snippets

C++:

```
/**
* Definition for a binary tree node.
* struct TreeNode {
```

```
* int val;
* TreeNode *left;
* TreeNode *right;
* TreeNode() : val(0), left(nullptr), right(nullptr) {}
* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
* TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left),
right(right) {}
* };
*/
class Solution {
public:
vector<string> binaryTreePaths(TreeNode* root) {

}
};
```

Java:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* int val;
* TreeNode left;
* TreeNode right;
* TreeNode() {}
* TreeNode(int val) { this.val = val; }
* TreeNode(int val, TreeNode left, TreeNode right) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
class Solution {
public List<String> binaryTreePaths(TreeNode root) {

}
}
```

Python3:

```
# 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 binaryTreePaths(self, root: Optional[TreeNode]) -> List[str]:
```