

// <https://leetcode.com/problems/binary-tree-paths/>

The screenshot shows the LeetCode interface for the problem '257. Binary Tree Paths'. On the left, the problem description states: 'Given a binary tree, return all root-to-leaf paths. Note: A leaf is a node with no children. Example: Input: A binary tree with root 1, left child 2, and right child 3. Node 2 has a left child 5. Output: ["1->2->5", "1->3"]'. The right side of the image shows a C# code editor with a solution. The code defines a `TreeNode` class, a `Solution` class with a `BinaryTreePaths` method, and a recursive `TreePaths` method. The `Height` method is also defined to calculate the height of the tree.

```
6 public class TreeNode {
7     public int val;
8     public TreeNode left;
9     public TreeNode right;
10    public TreeNode(int val=0, TreeNode left=null, TreeNode right=null) {
11        this.val = val;
12        this.left = left;
13        this.right = right;
14    }
15 }
16
17 public class Solution {
18     public IList<string> BinaryTreePaths(TreeNode root) {
19         int[] arr = new int[Height(root)];
20         List<string> result = new List<string>();
21         TreePaths(root, arr, 0, result);
22         return result;
23     }
24
25     public int Height(TreeNode root) => root==null? 0 : 1 + Math.Max(Height(root.left),Height(root.right));
26
27     public void TreePaths(TreeNode root, int[] arr,int index, List<string> result)
28     {
29         if(root==null) return;
30         arr[index]=root.val;
31
32         if (root.left == null && root.right == null)
33         {
34             string rootToLeaf="", arrow="";
35             for(int i=0;i<=index;i++)
36             {
37                 if(i>0)
38                     arrow="->";
39                 rootToLeaf=rootToLeaf+arrow+arr[i];
40             }
41             result.Add(rootToLeaf);
42
43             if(root.left!=null)
44                 TreePaths(root.left,arr,index+1,result);
45             if(root.right!=null)
46                 TreePaths(root.right,arr,index+1,result);
47         }
48     }
49 }
```

```
public class Solution {

    public IList<string> BinaryTreePaths(TreeNode root) {

        int[] arr = new int[Height(root)];

        List<string> result = new List<string>();

        TreePaths(root,arr,0,result);

        return result;

    }

    public int Height(TreeNode root) => root==null? 0 : 1 + Math.Max(Height(root.left),Height(root.right));

    public void TreePaths(TreeNode root, int[] arr,int index, List<string> result)

    {

        if(root==null) return;

        arr[index]=root.val;

        if (root.left == null && root.right == null)
```

```

{
    string rootToLeaf="", arrow="";
    for(int i=0;i<=index;i++)
    {
        if(i>0)
            arrow="->";
        rootToLeaf=rootToLeaf+arrow+arr[i];
    }
    result.Add(rootToLeaf);
}
if(root.left!=null)
    TreePaths(root.left,arr,index+1,result);
if(root.right!=null)
    TreePaths(root.right,arr,index+1,result);
}
}

```

Binary Tree Paths

Submission Detail

209 / 209 test cases passed.

Runtime: 248 ms

Memory Usage: 31.4 MB

Status: **Accepted**

Submitted: 0 minutes ago

Accepted Solutions Runtime Distribution

