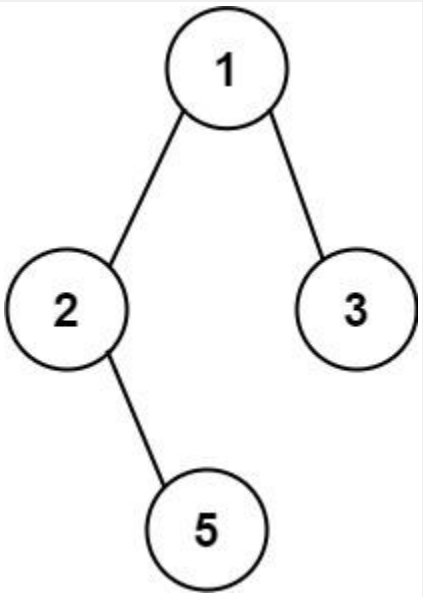
**Given the root of a binary tree, return *all root-to-leaf paths in any order*. A leaf is a node with no children.**

**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**

# Solution:

```
class Solution {

   public void treePaths(TreeNode root, String path, List<String> result) {

       if (root == null) {

           return;

       }
```

```
        // Append the current node's value to the path
        if (path.isEmpty()) {
            path = Integer.toString(root.val);
        } else {
            path += "->" + Integer.toString(root.val);
        }

        // If it's a leaf node, add the path to the result
        if (root.left == null && root.right == null) {
            result.add(path);
        } else {
            // Continue traversing the left and right subtrees
            treePaths(root.left, path, result);
            treePaths(root.right, path, result);
        }
    }

    public List<String> binaryTreePaths(TreeNode root) {
        List<String> result = new ArrayList<>();
        treePaths(root, "", result);
        return result;
    }
}
```