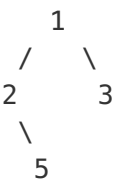
Binary Tree Paths

Given a binary tree, return all root-to-leaf paths.

For example, given the following binary tree:



All root-to-leaf paths are:

```
["1->2->5", "1->3"]
```

Credits:
Special thanks to @jianchao.li.fighter for adding this problem and creating all test cases.

Solution 1

```
public List<String> binaryTreePaths(TreeNode root) {
    List<String> answer = new ArrayList<String>();
    if (root != null) searchBT(root, "", answer);
    return answer;
}
private void searchBT(TreeNode root, String path, List<String> answer) {
    if (root.left == null && root.right == null) answer.add(path + root.val);
    if (root.left != null) searchBT(root.left, path + root.val + "->", answer);
    if (root.right != null) searchBT(root.right, path + root.val + "->", answer);
}
```

written by vimukthi original link here

Solution 2

```
void binaryTreePaths(vector<string>& result, TreeNode* root, string t) {
    if(!root->left && !root->right) {
        result.push_back(t);
        return;
    }

    if(root->left) binaryTreePaths(result, root->left, t + "->" + to_string(root->left->val));
    if(root->right) binaryTreePaths(result, root->right, t + "->" + to_string(root->right->val));
}

vector<string> binaryTreePaths(TreeNode* root) {
    vector<string> result;
    if(!root) return result;

    binaryTreePaths(result, root, to_string(root->val));
    return result;
}
```

written by jaewoo original link here

Solution 3

Lot of recursive solutions on this forum involves creating a helper recursive function with added parameters. The added parameter which usually is of the type List , carries the supplementary path information. However, the approach below doesn't use such a helper function.

```
public List<String> binaryTreePaths(TreeNode root) {

        List<String> paths = new LinkedList<>();

        if(root == null) return paths;

        if(root.left == null && root.right == null){
            paths.add(root.val+"");
            return paths;
        }

         for (String path : binaryTreePaths(root.left)) {
             paths.add(root.val + "->" + path);
         }

         for (String path : binaryTreePaths(root.right)) {
             paths.add(root.val + "->" + path);
         }

         return paths;

    }
```

written by sanket.purbey original link here