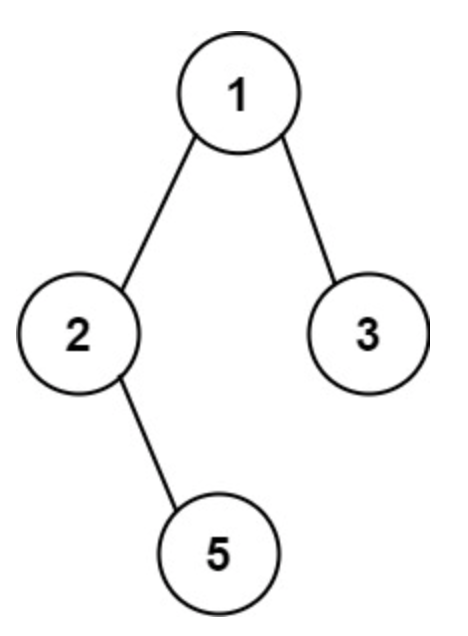
# 第四次算法设计作业

姓名： 刘宇骋 学号： 2021141440374

【说明】共两道题，至少完成一题，全完成可加分。解答应包括解题思路（方法、时间复杂度分析）、代码以及按结果验证中的输入得到的输出截图。作业提交时请只保留选做的题，删除空白题。

【题二】给你一个二叉树的根节点 root，按任意顺序返回所有从根节点到叶子节点的路径。

【示例1】



输入：root = [1,2,3,null,5]

输出：["1->2->5","1->3"]

【示例2】

输入：root = [1]

输出：["1"]

【解题思路】

最直观的方法是使用深度优先搜索。在深度优先搜索遍历二叉树时，我们需要考虑当前的节点以及它的孩子节点。

如果当前节点不是叶子节点，则在当前的路径末尾添加该节点，并继续递归遍历该节点的每一个孩子节点。

如果当前节点是叶子节点，则在当前路径末尾添加该节点后我们就得到了一条从根节点到叶子节点的路径，将该路径加入到答案即可。

如此，当遍历完整棵二叉树以后我们就得到了所有从根节点到叶子节点的路径。当然，深度优先搜索也可以使用非递归的方式实现，这里不再赘述。

【代码】

*import* my\_package.TreeNode;  
  
*import* java.util.ArrayList;  
*import* java.util.*List*;  
  
*public class* T257 {  
 *public static void* main(String[] args) {  
 TreeNode node5 = *new* TreeNode(5, *null*, *null*);  
 TreeNode node2 = *new* TreeNode(2, *null*, node5);  
 TreeNode node3 = *new* TreeNode(3, *null*, *null*);  
 TreeNode root = *new* TreeNode(1, node2, node3);  
 Solution solution = *new* Solution();  
 System.***out***.println(solution.binaryTreePaths(root));  
 }  
  
 *static class* Solution {  
 *private List*<String> ans = *new* ArrayList<>();  
  
 *private void* nextNode(TreeNode root, StringBuilder path) {  
 *if* (root.left == *null* && root.right == *null*) {  
 path.append(root.val);  
 ans.add(path.toString());  
 *return*;  
 }  
 path.append(root.val).append("->");  
 StringBuilder temp = *new* StringBuilder(path);  
 *if* (root.left != *null*) nextNode(root.left, path);  
 path = temp; *// 回溯  
 if* (root.right != *null*) nextNode(root.right, path);  
 }  
  
 *public List*<String> binaryTreePaths(TreeNode root) {  
 nextNode(root, *new* StringBuilder());  
 *return* ans;  
 }  
 }  
}

【结果验证】1、输入：root = [1,2,3,null,5]

2、输入：root = [1]

