二叉树的所有路径 · Binary Tree Paths

中文

脸书 (/problem/?tags=facebook) 苹果 (/problem/?tags=apple) 谷歌 (/problem/?tags=google) 二叉树遍历 (/problem/?tags=binary-tree-traversal) 二叉树 (/problem/?tags=binary-tree)

描述

给一棵二叉树，找出从根节点到叶子节点的所有路径。

样例

样例 1:

```
输入：{1,2,3,#,5}
输出：["1->2->5","1->3"]
解释：
   1
 /   \
2     3
 \
  5
```

样例 2:

```
输入：{1,2}
输出：["1->2"]
解释：
   1
 /
2
```

在线评测地址: https://www.lintcode.com/problem/binary-tree-paths/ (https://www.lintcode.com/problem/binary-tree-paths/)

收起题目描述 ^

语言类型 ALL (30) python (14) java (9) cpp (7)

上传题解

令狐冲
更新于 8/26/2020, 1:39:49 PM

使用分治算法(Divide Conquer)

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
// version 1: Divide Conquer
public class Solution {
    /**
     * @param root the root of the binary tree
     * @return all root-to-leaf paths
     */
    public List<String> binaryTreePaths(TreeNode root) {
        List<String> paths = new ArrayList<>();
        if (root == null) {
            return paths;
        }

        List<String> leftPaths = binaryTreePaths(root.left);
        List<String> rightPaths = binaryTreePaths(root.right);
        for (String path : leftPaths) {
            paths.add(root.val + "->" + path);
        }
        for (String path : rightPaths) {
            paths.add(root.val + "->" + path);
        }

        // root is a leaf
        if (paths.size() == 0) {
            paths.add("" + root.val);
        }

        return paths;
    }
}
```

获赞 23　　4 条评论

令狐冲
更新于 11/22/2020, 6:34:07 PM

另一种 Traversal

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
"""
Definition of TreeNode:
class TreeNode:
    def __init__(self, val):
        self.val = val
        self.left, self.right = None, None
"""

class Solution:
    """
    @param root: the root of the binary tree
    @return: all root-to-leaf paths
    """
    def binaryTreePaths(self, root):
        if root is None:
            return []

        result = []
        self.dfs(root, [str(root.val)], result)
        return result

    def dfs(self, node, path, result):
        if node.left is None and node.right is None:
            result.append('->'.join(path))
            return

        if node.left:
            path.append(str(node.left.val))
            self.dfs(node.left, path, result)
            path.pop()  # 回溯

        if node.right:
            path.append(str(node.right.val))
            self.dfs(node.right, path, result)
            path.pop()
```

获赞 20　　添加评论

令狐冲
更新于 10/30/2020, 3:46:26 AM

使用 Divider Conquer 版本的 DFS

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
"""
Definition of TreeNode:
class TreeNode:
    def __init__(self, val):
        self.val = val
        self.left, self.right = None, None
"""

class Solution:
    """
    @param root: the root of the binary tree
    @return: all root-to-leaf paths
    """
    def binaryTreePaths(self, root):
        if root is None:
            return []

        # 99% 的题，不用单独处理叶子节点
        # 这里需要单独处理的原因是 root 是 None 的结果，没有办法用于构造 root 是叶子的结果
        if root.left is None and root.right is None:
            return [str(root.val)]

        leftPaths = self.binaryTreePaths(root.left)
        rightPaths = self.binaryTreePaths(root.right)

        paths = []
        for path in leftPaths + rightPaths:
            paths.append(str(root.val) + '->' + path)

        return paths
```

获赞 20　　5 条评论

九章管理员
更新于 8/20/2020, 11:34:13 AM

使用遍历算法(Traverse)

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
public class Solution {
    /**
     * @param root the root of the binary tree
     * @return all root-to-leaf paths
     */
    public List<String> binaryTreePaths(TreeNode root) {
        List<String> result = new ArrayList<String>();
        if (root == null) {
            return result;
        }
        helper(root, String.valueOf(root.val), result);
        return result;
    }

    private void helper(TreeNode root, String path, List<String> result) {
        if (root == null) {
            return;
        }

        if (root.left == null && root.right == null) {
            result.add(path);
            return;
        }

        if (root.left != null) {
            helper(root.left, path + "->" + String.valueOf(root.left.val), result);
        }

        if (root.right != null) {
            helper(root.right, path + "->" + String.valueOf(root.right.val), result);
        }
    }
}
```

获赞 13　　23 条评论

令狐冲
更新于 6/12/2020, 3:37:16 PM

使用 Traverse 版本的 DFS 方法，(不推荐这种写法)

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
"""
Definition of TreeNode:
class TreeNode:
    def __init__(self, val):
        self.val = val
        self.left, self.right = None, None
"""

class Solution:
    """
    @param root: the root of the binary tree
    @return: all root-to-leaf paths
    """
    def binaryTreePaths(self, root):
        if root is None:
            return []

        result = []
        self.dfs(root, [], result)
        return result

    def dfs(self, node, path, result):
        path.append(str(node.val))

        if node.left is None and node.right is None:
            result.append('->'.join(path))
            path.pop()
            return

        if node.left:
            self.dfs(node.left, path, result);

        if node.right:
            self.dfs(node.right, path, result)

        path.pop()
```

获赞 5　　10 条评论

令狐冲
更新于 6/9/2020, 7:03:58 AM

遍历即可

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
/**
```

```
 * Definition of TreeNode:
 * class TreeNode {
 * public:
 *     int val;
 *     TreeNode *left, *right;
 *     TreeNode(int val) {
 *         this->val = val;
 *         this->left = this->right = NULL;
 *     }
 * }
 */
class Solution {
public:
    /**
     * @param root the root of the binary tree
     * @return all root-to-leaf paths
     */
    vector<string> binaryTreePaths(TreeNode* root) {
        // Write your code here
        vector<string> path;
        if(root == NULL)
            return path;
        vector<vector<int> > pathv;
        unordered_map<TreeNode*, bool> visited;
        stack<TreeNode*> stk;
        stk.push(root);
        visited[root] = true;
        if(root->left == NULL && root->right == NULL)
            save(pathv, stk);
        while(!stk.empty())
        {
            TreeNode* top = stk.top();
            if(top->left && visited[top->left] == false)
            {
                stk.push(top->left);
                visited[top->left] = true;
                if(top->left->left == NULL && top->left->right == NULL)
                    save(pathv, stk);
                continue;
            }
            if(top->right && visited[top->right] == false)
            {
                stk.push(top->right);
                visited[top->right] = true;
                if(top->right->left == NULL && top->right->right == NULL)
                    save(pathv, stk);
                continue;
            }
            stk.pop();
        }
        return convert(pathv);
    }

    void save(vector<vector<int> >& pathv, stack<TreeNode*> stk)
    {
        vector<int> cur;
        while(!stk.empty())
        {
            TreeNode* top = stk.top();
            cur.push_back(top->val);
            stk.pop();
        }
        reverse(cur.begin(), cur.end());
        pathv.push_back(cur);
    }

    vector<string> convert(vector<vector<int> >& pathv)
    {
```

```
        vector<string> path;
        for(int i = 0; i < pathv.size(); i ++)
        {
            string cur;
            cur += to_string(pathv[i][0]);
            for(int j = 1; j < pathv[i].size(); j ++)
            {
                cur += "->";
                cur += to_string(pathv[i][j]);
            }
            path.push_back(cur);
        }
        return path;
    }
};
```

获赞 1 添加评论

九章用户X8R5MA
更新于 6/9/2020, 7:03:46 AM

Divide & Conquer, CPP

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution {
public:
    /**
     * @param root the root of the binary tree
     * @return all root-to-leaf paths
     */
    vector<string> binaryTreePaths(TreeNode* root) {
        if (root == NULL) {
            return vector<string>();
        }
        if (root->left == NULL && root->right == NULL) {
            string x = to_string(root->val);
            return vector<string>{x};
        }

        vector<string> left = binaryTreePaths(root->left);
        vector<string> right = binaryTreePaths(root->right);

        vector<string> ret;
        for (auto& s : left) {
            ret.push_back(to_string(root->val) + "->" + s);
        }
        for (auto& s : right) {
            ret.push_back(to_string(root->val) + "->" + s);
        }
        return ret;
    }
};
```

获赞 5 添加评论

九章用户O8V2H5
更新于 6/9/2020, 7:03:46 AM

python 分治法

给一棵二叉树，找出从根节点到叶子节点的所有路径。

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
def binaryTreePaths(self, root):
        # write your code here
        result = []

        if root == None:
            return []

        left_path = self.binaryTreePaths(root.left)
        right_path = self.binaryTreePaths(root.right)

        for item in left_path:
            result.append(str(root.val) + '->' + str(item))

        for item in right_path:
            result.append(str(root.val) + '->' + str(item))

        if root.left is None and root.right is None:
            result.append(str(root.val))

        return result
```

获赞 5　　1 条评论

E同学
更新于 10/6/2020, 12:49:17 AM

遍历法 和标准答案唯一区别使用StringBuilder来记录path。好处是在做backtracking时，可以用append和delete的O(1)操作代替两string相加的O(n)操作

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
public class Solution {
    /**
     * @param root: the root of the binary tree
     * @return: all root-to-leaf paths
     */
    /**
     * Traversal
     * @param current path
     * @param current node
     * @result list<String>
     */
    public List<String> binaryTreePaths(TreeNode root) {
        // write your code here
        if (root == null)
            return new LinkedList<>();

        StringBuilder path = new StringBuilder();
        path.append(root.val);
        List<String> result = new LinkedList<>();
        traverse(root, path, result);
        return result;
    }

    public void traverse(TreeNode node, StringBuilder path, List<String> result) {
        int pathOrgLength = path.length();
        // This statement is redundant can be removed.
        if (node == null) {
            return;
        }

        if (node.left == null && node.right == null) {
            result.add(path.toString());
            return;
        }

        if (node.left != null) {
            path.append("->" + node.left.val);
            traverse(node.left, path, result);
            path.delete(pathOrgLength, path.length());
        }

        if (node.right != null) {
            path.append("->" + node.right.val);
            traverse(node.right, path, result);
            path.delete(pathOrgLength, path.length());
        }

        return;
    }
}
```

获赞 4　　添加评论

Tommy
更新于 6/9/2020, 7:03:50 AM

Use a vector to remember the path, save the path to the answer container when reach a leaf.

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
vector<string>
    findPaths(TreeNode* root,vector<int> path,vector<string>& stringPaths){
        if(NULL == root){
            return stringPaths;
        }
        path.push_back(root->val);
        if(root->left == NULL && root->right == NULL){
            stringstream solution;
            for(int x = 0; x < path.size();x++){
                if(0 == x){
                    solution<<path[x];
                }else{
                    solution<<"->"<<path[x];
                }
            }
            stringPaths.push_back(solution.str());
        }
        else{
            findPaths(root->left,path,stringPaths);
            findPaths(root->right,path,stringPaths);
        }
        return stringPaths;
    }
    vector<string> binaryTreePaths(TreeNode * root) {
        vector<int> path;
        vector<string> strPaths;
        return findPaths(root,path,strPaths);
    }
```

获赞 2　　添加评论

九章用户PJGZHC
更新于 6/9/2020, 7:03:50 AM

上传一版BFS，仅供参考，从上到下一个deque就可以，注意第一次传入的是空string

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
"""
Definition of TreeNode:
class TreeNode:
    def __init__(self, val):
        self.val = val
        self.left, self.right = None, None
"""

class Solution:
    """
    @param root: the root of the binary tree
    @return: all root-to-leaf paths
    """
    def binaryTreePaths(self, root):
        # write your code here
        if root is None:
            return []
        res = []
        deque = [(root,'')]

        while len(deque) > 0:
            current, path = deque.pop(0)
            if path:
                path += '->' + str(current.val)
            else:
                path += str(current.val)

            if not current.left and not current.right:#判断是否为叶节点，如果是，将路径加入 res
                res.append(path)
            if current.left:
                deque.append((current.left, path))
            if current.right:
                deque.append((current.right, path))

        return res
```

获赞 2　　添加评论

九章用户GVEB93
更新于 6/9/2020, 7:03:50 AM

非分治的traverse法。递归定义：求出当前节点的路径，如果当前节点是叶子节点，则将此路径添加到result里。 由于二叉树中每个叶子节点只会存在于一条路径，这个方法相当于是把每个叶子节点遍历一遍。思路参考了Maximum Depth of Binary Tree的方法，递归函数需要将root的父亲节点（唯一）的路径作为第二输入

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
# traverse
class Solution:
    """
    @param root: the root of the binary tree
    @return: all root-to-leaf paths
    """
    def binaryTreePaths(self, root):
        # write your code here
        self.result = []
        if not root:
            return []
        self.treePath(root, '')
        return self.result

    # 递归定义：得到包含root的路径，path是其父亲节点的路径
    def treePath(self, root, path):
        newPath = []
        if root.left is None and root.right is None:
            self.result.append(path + str(root.val))
            return
        newPath = path + str(root.val) + '->'

        if root.left:
            self.treePath(root.left, newPath)
        if root.right:
            self.treePath(root.right, newPath)
```

获赞 2　　添加评论

加载更多题解

进阶课程

视频+互动

九章算法班 2021 版

8周时间精通 57 个核心高频考点，9 招击破 FLAG、BATJ 算法面试。22....

直播+互动

系统架构设计 System Design 2021 版

成为百万架构师必上。30 课时带你快速掌握18大系统架构设计知识点与面...

直播+互动

九章算法面试高频题冲刺班

每期更新 15% 题目，考前押题，一举拿下FLAG & BATJ Offer

互动课

面向对象设计 OOD

应届生及亚马逊面试必考，IT求职必备基础

(/)

商务合作: fukesu@jiuzhang.com (mailto:fukesu@jiuzhang.com)