二叉树的最大深度 · Maximum Depth of Binary Tree

中文

领英 (/problem/?tags=linkedin) 雅虎 (/problem/?tags=yahoo) 苹果 (/problem/?tags=apple) 优步 (/problem/?tags=uber) 二叉树 (/problem/?tags=binary-tree) 分治法 (/problem/?tags=divide-and-conquer) 递归 (/problem/?tags=recursion)

描述

给定一个二叉树，找出其最大深度。

二叉树的深度为根节点到最远叶子节点的距离。

最终答案不会超过 `5000`

样例

样例 1:

```
输入: tree = {}
输出: 0
样例解释: 空树的深度是0。
```

样例 2:

```
输入: tree = {1,2,3,#,#,4,5}
输出: 3
样例解释: 树表示如下，深度是3
   1
  / \
 2   3
    / \
   4   5
它将被序列化为{1,2,3,#,#,4,5}
```

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

收起题目描述 ^

语言类型 ALL (17) java (6) python (6) cpp (5) 上传题解

九章-小原
更新于 10/23/2020, 8:19:31 PM

解题思路

- 思路
  - 这道题用DFS（深度优先搜索）来解答。我们知道，每个节点的深度与它左右子树的深度有关，且等于其左右子树最大深度值加上 1。
- 递归设计
  - 递归条件（recursive case）：对于当前节点 `root` ，我们求出左右子树的深度的最大值，再加1表示当前节点的深度，返回该数值，即为以 `root` 为根节点的树的最大深度。
  - 终止条件（base case）： 当前节点为空时，认为树深为0。

复杂度分析

- 时间复杂度： O(n)，其中 n 是节点的数量。我们每个节点只访问一次，因此时间复杂度为 O(n)。
- 空间复杂度：考虑到递归使用调用栈（call stack）的情况。
  - 最坏情况：树完全不平衡。例如每个节点都只有左节点，此时将递归 n 次，需要保持调用栈的存储为 O(n)
  - 最好情况：树完全平衡。即树的高度为 log(n)，此时空间复杂度为 O(log(n))

代码

python java c++

```
/**
* 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 binary tree.
    @return: An integer
    """
    def maxDepth(self, root):
        if root is None:
            return 0
        leftDepth = self.maxDepth(root.left)
        rightDepth = self.maxDepth(root.right)
        return max(leftDepth, rightDepth) + 1
```

获赞 6　　添加评论

令狐冲
更新于 9/8/2020, 1:13:32 AM

Given a binary tree, find its maximum depth.

The maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.
 详细解答请至九章微博：http://weibo.com/3948019741/BBY7gwi89

简单的树的遍历，点i为根的树高度，等于高度最大的子树的高度+1。

```
/**
* 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):
        this.val = val
        this.left, this.right = None, None
"""
class Solution:
    """
    @param root: The root of binary tree.
    @return: An integer
    """
    def maxDepth(self, root):
        if root is None:
            return 0
        return max(self.maxDepth(root.left), self.maxDepth(root.right)) + 1
```

获赞 5　　4 条评论

令狐冲
更新于 7/21/2020, 8:33:15 PM

Given a binary tree, find its maximum depth.

The maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.
 详细解答请至九章微博：http://weibo.com/3948019741/BBY7gwi89

简单的树的遍历，点i为根的树高度，等于高度最大的子树的高度+1。

```
/**
* 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 {
    public int maxDepth(TreeNode root) {
        if (root == null) {
            return 0;
        }

        int left = maxDepth(root.left);
        int right = maxDepth(root.right);
        return Math.max(left, right) + 1;
    }
}

// version 2: Traverse
/**
 * Definition of TreeNode:
 * public class TreeNode {
 *     public int val;
 *     public TreeNode left, right;
 *     public TreeNode(int val) {
 *         this.val = val;
 *         this.left = this.right = null;
 *     }
 * }
 */
public class Solution {
    /**
     * @param root: The root of binary tree.
     * @return: An integer.
     */
    private int depth;

    public int maxDepth(TreeNode root) {
        depth = 0;
        helper(root, 1);

        return depth;
    }

    private void helper(TreeNode node, int curtDepth) {
        if (node == null) {
            return;
        }

        if (curtDepth > depth) {
            depth = curtDepth;
        }

        helper(node.left, curtDepth + 1);
        helper(node.right, curtDepth + 1);
    }
}
```

获赞 1　　4 条评论

令狐冲
更新于 6/9/2020, 7:04:29 AM

Given a binary tree, find its maximum depth.

The maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.
 详细解答请至九章微博：http://weibo.com/3948019741/BBY7gwi89

简单的树的遍历，点i为根的树高度，等于高度最大的子树的高度+1。

```
/**
* 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 binary tree.
     * @return: An integer
     */
    int maxDepth(TreeNode *root) {
        if (root == NULL) {
            return 0;
        }
        int left = maxDepth(root->left);
        int right = maxDepth(root->right);
        return left > right ? left + 1 : right + 1;
    }
};
```

获赞 0 添加评论

TONG
更新于 12/15/2020, 6:57:02 PM

根据随课教程中的java版本改写的遍历法python版，直接套模板

```
/**
* 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 binary tree.
    @return: An integer
    """
    def maxDepth(self, root):
        # write your code here
        self.depth = 0
        self.traverse(root, 1)
        return self.depth

    def traverse(self, root, curdepth):
        if not root:
            return
        self.depth = max(self.depth, curdepth)
        self.traverse(root.left, curdepth + 1)
        self.traverse(root.right, curdepth + 1)
```

获赞 5　　1 条评论

Tin
更新于 8/27/2020, 3:13:23 PM

经典题，打基础题，一题多解题。

常见做法有： 1. 分治法，要用递归 2. 遍历法，要用全局变量，只见过用递归的 3. 宽搜法，属杀鸡用牛刀 4. 不用递归的遍历法，即所附原创代码，用后续遍历，这好像是所知后续遍历唯一有实际用处的地方。

```
/**
* 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:

    def maxDepth(self, root):

        running_depth, max_depth, stack = 0, 0, []

        while stack or root:
            if stack:
                running_depth = stack[-1][1]
            if root:
                stack.append((root, running_depth+1))
                root = root.left if root.left else root.right
            else:
                root, running_depth = stack.pop()
                max_depth = max(max_depth, running_depth)
                if stack and stack[-1][0].left is root:
                    root = stack[-1][0].right
                else:
                    root = None

        return max_depth
```

获赞 3　　添加评论

九章用户51HVT8
更新于 6/9/2020, 7:03:50 AM

我是不是偷懒了？用了BFS的思路，把树走一遍，用queue 记录每一层的节点数，记录层数。好像没有别的了。一次通过

```
/**
* 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:
    int maxDepth(TreeNode* root) {
        /*
        解法一： 递归
        if(root == NULL)
            return 0;
        int leftdep = maxDepth(root->left);
        int rightdep = maxDepth(root->right);
        int dep = max(leftdep, rightdep);
        return dep +1;
        */

        /*
        解法二： BFS
        */
        int deep = 0;
        if(root == NULL)
            return deep;
        queue<TreeNode *> Q;
        Q.push(root);

        while(!Q.empty())
        {
            int size = Q.size();
            for(int i = 0; i < size; i++)
            {
                TreeNode *temp = Q.front();
                Q.pop();
                if(temp->left != NULL)
                    Q.push(temp->left);
                if(temp->right != NULL)
                    Q.push(temp->right);
            }
            deep +=1;
        }
        return deep;

    }
};
```

获赞 2　　添加评论

Chichi
更新于 6/9/2020, 7:03:53 AM

传一个非递归的写法，同样采用前序遍历，只不过 stack 不仅记录了 node，还记录了当前 node 的深度

```
/**
* 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 binary tree.
    @return: An integer
    """
    def maxDepth(self, root):
        # write your code here
        if root is None:
            return 0
        stack = []
        result = []
        depth = 0
        node = root

        while node or len(stack):
            while node:
                result.append(depth + 1)
                depth += 1
                stack.append([node, depth])
                node = node.left

            popout = stack.pop()
            node = popout[0]
            depth = popout[1]
            node = node.right


        return max(result)
```

获赞 1　　添加评论

九章用户U6HP8S
更新于 6/9/2020, 7:03:53 AM

Java DFS solution in iterative way.

Notes: 1. only need to update max on the leaf node. No need to update max on other nodes 2. to my opinion this solution is more like pre-order traveral in its nature

```
/**
* 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 for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
class Solution {
    public int maxDepth(TreeNode root) {
        if (root == null) return 0;

        int max = 1;

        Stack<TreeNode> nodes = new Stack<>();
        Stack<Integer> depths = new Stack<>();

        nodes.push(root);
        depths.push(1);

        while (!nodes.empty()) {
            TreeNode curr = nodes.pop();
            int depth = depths.pop();

            if (curr.left == null && curr.right == null) {
                max = Math.max(max, depth);
            }

            if (curr.right != null) {
                nodes.push(curr.right);
                depths.push(depth + 1);
            }
            if (curr.left != null) {
                nodes.push(curr.left);
                depths.push(depth + 1);
            }
        }

        return max;

    }
}
```

获赞 1　　添加评论

Jerbear
更新于 6/9/2020, 7:03:52 AM

BFS 分层搜索 为了分层搜索所以每次要check queue的size 然后分别把左右儿子放到queue里 每次层搜索结束 然后更新max

```
/**
* 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:
 * public class TreeNode {
 *     public int val;
 *     public TreeNode left, right;
 *     public TreeNode(int val) {
 *         this.val = val;
 *         this.left = this.right = null;
 *     }
 * }
 */

public class Solution {
    /**
     * @param root: The root of binary tree.
     * @return: An integer
     */
    public int maxDepth(TreeNode root) {
        // write your code here
        // 1. Divide and conquer root = maxLeft + maxRight
        if (root == null){
            return 0;
        }

        Queue<TreeNode> queue = new LinkedList<>();
        queue.offer(root);
        int max = 0;

        while (!queue.isEmpty()){
            int levelSize = queue.size();
            for (int i = 0; i < levelSize; i++){
                TreeNode head = queue.poll();
                if (head.left != null){
                    queue.offer(head.left);
                }

                if (head.right != null){
                    queue.offer(head.right);
                }
            }
            max++;
        }
        return max;
    }
}
```

获赞 1　　添加评论

九章用户U6FLMC
更新于 10/21/2020, 10:48:22 PM

用DFS，不用分治的解法：

public class Solution { /** *@param root: The root of binary tree.* @return: An integer */

```
/**
* 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
*/
private int maxDepth;
public int maxDepth(TreeNode root) {
    // write your code here
    if (root != null){
        traverse(root, 1);
    }
    return maxDepth;

}

public void traverse(TreeNode root, int depth){
    if (root.left == null && root.right == null){
        maxDepth = Math.max(depth, maxDepth);
    }

    if (root.left != null){
        traverse(root.left, depth + 1);
    }
    if (root.right != null){
        traverse(root.right, depth + 1);
    }
}
```

}

获赞 0 添加评论

九章用户DE6NRS
更新于 6/9/2020, 7:04:27 AM

maximum-depth-of-binary-tree/#

```
/**
* 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 binary tree.
     * @return: An integer.
     */
    public int maxDepth_nonrecursion(TreeNode root) {
        if (root == null) {
            return 0;
        }
        Deque<TreeNode> stack = new ArrayDeque<>();
        Deque<Integer> depth = new ArrayDeque<>();
        stack.offerFirst(root);
        depth.offerFirst(1);
        int maxDepth = 0;

        while (stack.size() != 0) {
            TreeNode node = stack.pollFirst();
            int currDepth = depth.pollFirst();
            maxDepth = Math.max(maxDepth, currDepth);
            if (node.left != null) {
                stack.offerFirst(node.left);
                depth.offerFirst(currDepth + 1);
            }
            if (node.right != null) {
                stack.offerFirst(node.right);
                depth.offerFirst(currDepth + 1);
            }
        }
        return maxDepth;
    }
}
```


获赞 0 添加评论

jerron
更新于 6/9/2020, 7:04:11 AM

straight forward recursive 30 30 30 30 30 30 30 30

```
/**
* 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 binary tree.
     * @return: An integer
     */
    int maxDepth(TreeNode * root) {
        // write your code here
        if(root)
            return max(maxDepth(root->left),maxDepth(root->right))+1;
        return 0;
    }
};
```

获赞 0 添加评论

Sam
更新于 6/9/2020, 7:04:06 AM

BFS,binary tree level order traversal原码，速度还挺快的

```
/**
* 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
"""
from collections import deque
class Solution:
    """
    @param root: The root of binary tree.
    @return: An integer
    """
    def maxDepth(self, root):
        if root is None:
            return 0
        queue = deque([root])
        result = 0
        while queue:
            for _ in range(len(queue)):
                node = queue.popleft()
                if node.left:
                    queue.append(node.left)
                if node.right:
                    queue.append(node.right)
            result += 1
        return result
```

获赞 0 添加评论

Jet
更新于 6/9/2020, 7:04:04 AM

Maximum Depth of Binary Tree.Maximum Depth of Binary Tree.

```
/**
* 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 binary tree.
     * @return: An integer
     */
    int maxDepth(TreeNode* root){
        if(root==nullptr){
            return 0;
        }
        int l = maxDepth(root->left);
        int r = maxDepth(root->right);
        return max(l,r)+1;
    }

    int maxDepth2(TreeNode* root) {

        depth2=0;
        traverse(root,1);
        return depth2;
    }

    void traverse(TreeNode* root,int dep){
        if(root==nullptr){
            return;
        }
        int a = max(depth2,dep);
        depth2=a;
        traverse(root->left,dep+1);//a+1?
        traverse(root->right,dep+1);//a+1
    }

     int bfs(TreeNode * root) {
        // write your code here
        int depth =0;
        if(root ==nullptr){return depth;}
        std::queue<TreeNode*> queue;
        queue.push(root);

        while(!queue.empty()){
            size_t length=queue.size();
            while(length>0){
                TreeNode* node=queue.front();
                queue.pop();
                if(node->left != nullptr){
                    queue.push(node->left);
                }
                if(node->right!=nullptr){
                    queue.push(node->right);
                }
                length--;
            }
            depth++;
        }
        return depth;
    }
public:
    int depth2;
};
```

获赞 0 添加评论

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

(/)