二叉树的层次遍历 · Binary Tree Level Order Traversal

中文

Bloomberg (/problem/?tags=bloomberg) 微软 (/problem/?tags=microsoft) 脸书 (/problem/?tags=facebook) 亚马逊 (/problem/?tags=amazon)
领英 (/problem/?tags=linkedin) 苹果 (/problem/?tags=apple) 优步 (/problem/?tags=uber) 二叉树遍历 (/problem/?tags=binary-tree-traversal)
Breadth-first Search (/problem/?tags=breadth-first-search) 二叉树 (/problem/?tags=binary-tree) 队列 (/problem/?tags=queue)

描述

给出一棵二叉树，返回其节点值的层次遍历（逐层从左往右访问）

* 首个数据为根节点，后面接着是其左儿子和右儿子节点值，"#"表示不存在该子节点。 * 节点数量不超过20。

样例

样例 1:

```
输入：{1,2,3}
输出：[[1],[2,3]]
解释：
   1
  / \
 2   3
它将被序列化为{1,2,3}
层次遍历
```

样例 2:

```
输入：{1,#,2,3}
输出：[[1],[2],[3]]
解释：
1
 \
  2
 /
3
它将被序列化为{1,#,2,3}
层次遍历
```

挑战

挑战1：只使用一个队列去实现它

挑战2：用BFS算法来做

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

收起题目描述 ^

语言类型 ALL (21) python (14) java (3) cpp (3) golang (1) 上传题解

令狐冲
更新于 10/6/2020, 11:34:39 PM

用一个队列的方法。

```
/**
* 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
*/
from collections import deque

"""
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: Level order a list of lists of integer
    """
    def levelOrder(self, root):
        if root is None:
            return []

        queue = deque([root])
        result = []
        while queue:
            level = []
            for _ in range(len(queue)):
                node = queue.popleft()
                level.append(node.val)
                if node.left:
                    queue.append(node.left)
                if node.right:
                    queue.append(node.right)
            result.append(level)
        return result
```


获赞 9　　13 条评论

令狐冲
更新于 8/18/2020, 3:36:37 PM

考点：

- 二叉树的层次遍历

题解： 用两个队列轮换的方法。

```
/**
* 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 binary tree.
    @return: Level order in a list of lists of integers
    """
    def levelOrder(self, root):
        if not root:
            return []

        queue = [root]
        results = []
        while queue:
            next_queue = []
            results.append([node.val for node in queue])
            for node in queue:
                if node.left:
                    next_queue.append(node.left)
                if node.right:
                    next_queue.append(node.right)
            queue = next_queue
        return results
```

获赞 2　　7 条评论

令狐冲
更新于 6/9/2020, 7:03:51 AM

Given a binary tree, return the level order traversal of its nodes' values. (ie, from left to right, level by level). 考点：

- 二叉树的层次遍历

For example: Given binary tree {3,9,20,#,#,15,7}, 3 / \ 9 20 / \ 15 7 return its level order traversal as: [3 (), 9,20 (), 15,7 ()] 分析:二叉树的层次遍历通常实现方式为使用队列不断压入节点遍历,每次取出队列首个元素遍历左右子节点，继续压入子节点即可。

```
/**
* 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
*/
// version 1: BFS
public class Solution {
```

```
    public List<List<Integer>> levelOrder(TreeNode root) {
        List result = new ArrayList();

        if (root == null) {
            return result;
        }

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

        while (!queue.isEmpty()) {
            ArrayList<Integer> level = new ArrayList<Integer>();
            int size = queue.size();
            for (int i = 0; i < size; i++) {
                TreeNode head = queue.poll();
                level.add(head.val);
                if (head.left != null) {
                    queue.offer(head.left);
                }
                if (head.right != null) {
                    queue.offer(head.right);
                }
            }
            result.add(level);
        }

        return result;
    }
}


// version 2:  DFS
public class Solution {
    /**
     * @param root: The root of binary tree.
     * @return: Level order a list of lists of integer
     */
    public List<List<Integer>> levelOrder(TreeNode root) {
        List<List<Integer>> results = new ArrayList<List<Integer>>();

        if (root == null) {
            return results;
        }

        int maxLevel = 0;
        while (true) {
            List<Integer> level = new ArrayList<Integer>();
            dfs(root, level, 0, maxLevel);
            if (level.size() == 0) {
                break;
            }

            results.add(level);
            maxLevel++;
        }

        return results;
    }

    private void dfs(TreeNode root,
                     List<Integer> level,
                     int curtLevel,
                     int maxLevel) {
        if (root == null || curtLevel > maxLevel) {
            return;
        }

        if (curtLevel == maxLevel) {
```

```
            level.add(root.val);
            return;
        }

        dfs(root.left, level, curtLevel + 1, maxLevel);
        dfs(root.right, level, curtLevel + 1, maxLevel);
    }
}


// version 3: BFS. two queues
public class Solution {
    /**
     * @param root: The root of binary tree.
     * @return: Level order a list of lists of integer
     */
    public List<List<Integer>> levelOrder(TreeNode root) {
        List<List<Integer>> result = new ArrayList<List<Integer>>();
        if (root == null) {
            return result;
        }

        List<TreeNode> Q1 = new ArrayList<TreeNode>();
        List<TreeNode> Q2 = new ArrayList<TreeNode>();

        Q1.add(root);
        while (Q1.size() != 0) {
            List<Integer> level = new ArrayList<Integer>();
            Q2.clear();
            for (int i = 0; i < Q1.size(); i++) {
                TreeNode node = Q1.get(i);
                level.add(node.val);
                if (node.left != null) {
                    Q2.add(node.left);
                }
                if (node.right != null) {
                    Q2.add(node.right);
                }
            }

            // swap q1 and q2
            List<TreeNode> temp = Q1;
            Q1 = Q2;
            Q2 = temp;

            // add to result
            result.add(level);
        }

        return result;
    }
}

// version 4: BFS, queue with dummy node
public class Solution {
    /**
     * @param root: The root of binary tree.
     * @return: Level order a list of lists of integer
     */
    public List<List<Integer>> levelOrder(TreeNode root) {
        List<List<Integer>> result = new ArrayList<List<Integer>>();
        if (root == null) {
            return result;
        }

        Queue<TreeNode> Q = new LinkedList<TreeNode>();
        Q.offer(root);
        Q.offer(null); // dummy node
```

```
        List<Integer> level = new ArrayList<Integer>();
        while (!Q.isEmpty()) {
            TreeNode node = Q.poll();
            if (node == null) {
                if (level.size() == 0) {
                    break;
                }
                result.add(level);
                level = new ArrayList<Integer>();
                Q.offer(null); // add a new dummy node
                continue;
            }

            level.add(node.val);
            if (node.left != null) {
                Q.offer(node.left);
            }
            if (node.right != null) {
                Q.offer(node.right);
            }
        }

        return result;
    }
}
```

获赞 2　　9 条评论

李助教
更新于 6/9/2020, 7:03:53 AM

dfs的办法： dfs函数每次将target_level层的遍历结果放进level中 target_level从0开始，依次增加 当target_level超出二叉树的高度时，dfs不会向level中加入数据，从而跳出死循环。

```
/**
* 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:
    """
    @param root: A Tree
    @return: Level order a list of lists of integer
    """
    def levelOrder(self, root):
        # write your code here
        result = []
        target_level = 0
        while True:
            level = []
            self.dfs(root, 0, target_level, level)
            if not level:
                break
            result.append(level)
            target_level += 1
        return result

    def dfs(self, root, cur_level, target_level, level):
        if not root or cur_level > target_level:
            return
        if cur_level == target_level:
            level.append(root.val)
        self.dfs(root.left, cur_level + 1, target_level, level)
        self.dfs(root.right, cur_level + 1, target_level, level)
```

获赞 1　　添加评论

令狐冲
更新于 6/9/2020, 7:04:30 AM

Given a binary tree, return the level order traversal of its nodes' values. (ie, from left to right, level by level). 考点：

- 二叉树的层次遍历

For example: Given binary tree {3,9,20,#,#,15,7}, 3 / \ 9 20 / \ 15 7 return its level order traversal as: [3 (), 9,20 (), 15,7 ()] 分析:二叉树的层次遍历通常实现方式为使用队列不断压入节点遍历,每次取出队列首个元素遍历左右子节点，继续压入子节点即可。

```
/**
* 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 {
    /**
     * @param root: The root of binary tree.
     * @return: Level order a list of lists of integer
     */
public:
    vector<vector<int>> levelOrder(TreeNode *root) {
        vector<vector<int>> result;
        if (root == NULL) {
            return result;
        }

        queue<TreeNode *> Q;
        Q.push(root);
        while (!Q.empty()) {
            int size = Q.size();
            vector<int> level;

            for (int i = 0; i < size; i++) {
                TreeNode *head = Q.front(); Q.pop();
                level.push_back(head->val);
                if (head->left != NULL) {
                    Q.push(head->left);
                }
                if (head->right != NULL) {
                    Q.push(head->right);
                }
            }

            result.push_back(level);
        }

        return result;
    }
};
```

获赞 0 添加评论

九章用户QXN6C1
更新于 6/9/2020, 7:03:47 AM

lintcode 里面的两个challenge解法：challenge1：use only one queue, challenge2：use 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
*/
# # Challenge 1: with one queue
# from collections import deque
# class Solution:
#     """
#     @param: root: A Tree
#     @return: Level order a list of lists of integer
#     """
#     def levelOrder(self, root):
#         # write your code here
#         result = []

#         if root is None:
#             return result

#         q = deque([root])
#         while q:
#             result.append([item.val for item in q])
#             len_q = len(q)
#             for i in range(len_q):
#                 node = q.popleft()
#                 if node.left is not None:
#                     q.append(node.left)
#                 if node.right is not None:
#                     q.append(node.right)
#         return result

# Challenge 2: with DFS
class Solution:
    """
    @param: root: A Tree
    @return: Level order a list of lists of integer
    """
    def levelOrder(self, root):
        # write your code here
        result = []

        if root is None:
            return result

        thisLevel = 0
        self.DFS(root, result, thisLevel)
        return result

    def DFS(self, root, result, thisLevel):
        if root is None:
            return
        if thisLevel > len(result) - 1:
            result.append([])
        result[thisLevel].append(root.val)
        thisLevel += 1
        self.DFS(root.left, result, thisLevel)
        self.DFS(root.right, result, thisLevel)
        return
```

获赞 4　　添加评论

九章用户AJSYNX
更新于 6/9/2020, 7:03:50 AM

DFS方法： 其实是pre-order遍历，pre-order开始会一层层向下走，level==res.size()的时候初始化一下内层的列表。

```
/**
* 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: A Tree
     * @return: Level order a list of lists of integer
     */
    public List<List<Integer>> levelOrder(TreeNode root) {
        // write your code here
        List<List<Integer>> res = new ArrayList<>();
        if (root == null) {
            return res;
        }

        dfs(root, res, 0);

        return res;
    }

    private void dfs(TreeNode root, List<List<Integer>> res, int level) {
        if (root == null) {
            return;
        }
        if (level == res.size()) {
            res.add(new ArrayList<>());
        }
        res.get(level).add(root.val);

        dfs(root.left, res, level + 1);
        dfs(root.right, res, level + 1);
    }
}
```

获赞 2　　3 条评论

cc189
更新于 6/9/2020, 7:03:50 AM

使用 python queue package

给出一棵二叉树，返回其节点值的层次遍历（逐层从左往右访问）

```
/**
* 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
*/
from queue import Queue
class Solution:
    def levelOrder(self, root):
        # null
        if not root: return []

        # init
        res = []
        queue = Queue()
        queue.put(root)

        # loop
        while queue.qsize() > 0:
            size = queue.qsize()
            level = []
            for i in range(size):
                cur = queue.get()
                level.append(cur.val)
                if cur.left:
                    queue.put(cur.left)
                if cur.right:
                    queue.put(cur.right)
            res += [level]

        return res
```

获赞 2　　添加评论

九章用户RBODW7
更新于 7/29/2020, 3:37:41 PM

BFS的一个队列，用dummy node间隔每层的方法，注意append的时候要用result.append(level_list.copy())

```
/**
* 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: A Tree
    @return: Level order a list of lists of integer
    """
    def levelOrder(self, root):
        # write your code here
        if not root:
            return []

        result = []
        level_list = []
        queue = deque([root, None])

        while queue:
            node = queue.popleft()
            if node:
                level_list.append(node.val)
                if node.left:
                   queue.append(node.left)
                if node.right:
                    queue.append(node.right)
            else:
                result.append(level_list.copy())
                level_list.clear()
                if queue:
                    queue.append(None)

        return result
```

获赞 1　　添加评论

九章用户EBTP6U
更新于 6/9/2020, 7:03:57 AM

使用了BFS, 用了一个队列存放需要被访问的节点。时间复杂度为O(N), 空间复杂度为O(N).

```
/**
* 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<vector<int>> levelOrder(TreeNode * root) {
        // write your code here
        if (!root) return {};
        queue<TreeNode *> q;
        vector<vector<int>> results;
        q.push(root);
        while(!q.empty()) {
            results.push_back({});
            auto &result = results.back();
            int size = q.size();
            for (int i = 0; i < size; i++) {
                auto cur = q.front();
                result.push_back(cur->val);
                if (cur->left) q.push(cur->left);
                if (cur->right) q.push(cur->right);
                q.pop();
            }
        }
        return results;
    }
```

获赞 1　　添加评论

社会我喵哥
更新于 6/9/2020, 7:03:56 AM

BFS, 传统的用一个Queue push & pop, pop前先取一下这一层的数量

```
/**
* 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 levelOrder(self, root):
        # write your code here
        if not root:
            return []
        queue = []
        result = []
        queue.append(root)
        while queue:
            size = len(queue)
            level = []
            for i in range(size):
                node = queue.pop(0)
                level.append(node.val)
                if node.left != None:
                    queue.append(node.left)
                if node.right != None:
                    queue.append(node.right)
            result.append(level)
        return result
```

获赞 1　　添加评论

九章用户3IHROM
更新于 11/26/2020, 2:44:14 AM

```
/**
* 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
*/
import collections

class Solution(object):
    def levelOrder(self, root):
        """
        :type root: TreeNode
        :rtype: List[List[int]]
        """
        if not root:
            return []

        result = []
        queue = collections.deque([root])

        while queue:
            result.append([node.val for node in queue])
            for i in range(len(queue)):
                node = queue.popleft()
                if node.left:
                    queue.append(node.left)
                if node.right:
                    queue.append(node.right)

        return result
```

获赞 0 添加评论

S同学
更新于 6/9/2020, 7:04:24 AM

DFS. 与题解答案不同，遍历一次。时间复杂度O(n).....

```
/**
* 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: A Tree
     * @return: Level order a list of lists of integer
     */
    public List<List<Integer>> levelOrder(TreeNode root) {
        // write your code here
        List<List<Integer>> result = new ArrayList<>();

        if (root == null) {
            return result;
        }

        dfs(root, result, 1);

        return result;
    }

    private void dfs(TreeNode node, List<List<Integer>> result, int level) {
        if (node == null) {
            return;
        }

        if (result.size() < level) {
            List<Integer> list = new ArrayList<Integer>();
            list.add(node.val);
            result.add(list);
        } else {
            result.get(level - 1).add(node.val);
        }

        dfs(node.left, result, level + 1);
        dfs(node.right, result, level + 1);
    }
}
```

获赞 0　　添加评论

刷题ing
更新于 6/9/2020, 7:04:24 AM

这个代码比较简单。速度好像一般。时间复杂度依然不会分析。。。。

```
/**
* 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<vector<int>> levelOrder(TreeNode* root) {
        vector<vector<int>>ret;
        buildVector(root, 0,ret);
            return ret;
    }

    void buildVector(TreeNode*root, int depth, vector<vector<int>>&ret){
        if(root == NULL) return;
                if(depth == ret.size())
            ret.push_back(vector<int>());

            ret[depth].push_back(root->val);
            buildVector(root->left, depth + 1,ret);
            buildVector(root->right, depth + 1,ret);
    }
```

获赞 0　　添加评论

马同学
更新于 6/9/2020, 7:04:23 AM

对照老师BFS方法写的python版本，要注意pop(0)，因为在python里面pop()的默认Index是-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):
        self.val = val
        self.left, self.right = None, None
"""

class Solution:
    """
    @param root: A Tree
    @return: Level order a list of lists of integer
    """
    def levelOrder(self, root):
        # write your code here
        result = []

        if root is None:
            return result

        queue = []
        queue.append(root)

        while queue:
            level = []
            size = len(queue)
            for i in range(0, size):
                head = queue.pop(0)
                level.append(head.val)
                if head.left is not None:
                    queue.append(head.left)
                if head.right is not None:
                    queue.append(head.right)

            result.append(level)

        return result
```

获赞 0　　添加评论

九章用户RBODW7
更新于 6/9/2020, 7:04:21 AM

这是最基本的深度优先搜索解法，基本就是翻译了一下精选的DFS的Java版本

```
/**
* 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: A Tree
    @return: Level order a list of lists of integer
    """
    def levelOrder(self, root):
        # write your code here
        if not root:
            return []

        result = []
        max_level = 0
        while True:
            level_list = []
            self.dfs(root, level_list, 0, max_level)

            if not level_list:
                break

            result.append(level_list)
            max_level += 1

        return result


    def dfs(self, root, level_list, curr_level, max_level):
        if not root or curr_level > max_level:
            return

        if curr_level == max_level:
            level_list.append(root.val)

        self.dfs(root.left, level_list, curr_level + 1, max_level)
        self.dfs(root.right, level_list, curr_level + 1, max_level)
```

获赞 0 添加评论

九章用户RBODW7
更新于 6/9/2020, 7:04:21 AM

上传一下Python版本用deque写的2个queue的BFS方法

```
/**
* 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
"""
from collections import deque
class Solution:
    """
    @param root: A Tree
    @return: Level order a list of lists of integer
    """
    def levelOrder(self, root):
        # write your code here
        if not root:
            return []

        result = []
        queue1 = deque([root])
        queue2 = deque()

        while queue1:
            level_list = [node.val for node in queue1]
            for node in queue1:
                if node.left:
                    queue2.append(node.left)
                if node.right:
                    queue2.append(node.right)

            result.append(level_list)
            queue1, queue2 = queue2, queue1
            queue2.clear()

        return result
```

获赞 0　　添加评论

Charlie
更新于 6/9/2020, 7:04:21 AM

通用的 Graph 的 BFS。

使用 None 来分隔层次。

```
/**
* 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
*/
    while q:
        node = q.popleft()
        if node == None:
            A.append(curr_level)
            curr_level = []
            if not q:
                break;
            q.append(None)
            continue
```

```
/**
* 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
"""
import collections

class Solution:
    """
    @param root: A Tree
    @return: Level order a list of lists of integer
    """
    def levelOrder(self, root):
        # write your code here
        if root is None:
            return []
        q = collections.deque([root])
        q.append(None)
        A = []
        curr_level = []
        while q:
            node = q.popleft()
            if node == None:
                A.append(curr_level)
                curr_level = []
                if not q:
                    break;
                q.append(None)
                continue
            curr_level.append(node.val)
            if node.left:
                q.append(node.left)
            if node.right:
                q.append(node.right)
        return(A)
```

获赞 0 添加评论

Soton Zepler
更新于 6/9/2020, 7:04:05 AM

两个队列的实现方法。和九章解法不同在于我用第一个队列判断左右节点丢到第二个队列中。本层的值在第一个队列时判断。 反转两个队列的条件是当第一个队列为空时。反转两个队列时顺便把本层的输出值清空。

```
/**
* 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:
    """
    @param root: A Tree
    @return: Level order a list of lists of integer
    """
    def levelOrder(self, root):
        # write your code here

        if not root:
            return []

        currentLevel, nextLevel = [], []

        currentLevel = [root]
        results = []
        values = []

        while currentLevel:
            node = currentLevel.pop(0)
            if node.left:
                nextLevel.append(node.left)
            if node.right:
                nextLevel.append(node.right)

            values.append(node.val)
            if not currentLevel:
                currentLevel, nextLevel = nextLevel, currentLevel
                results.append(values)
                values = []

        return results
```

获赞 0 添加评论

Soton Zepler
更新于 6/9/2020, 7:04:05 AM

用一个队列存储所有的节点。只是把两个队列换成了两个counter。在本层counter为零时反转这两个counter的值即可。

```
/**
* 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
*/
class Solution:
    """
    @param root: A Tree
    @return: Level order a list of lists of integer
    """
    def levelOrder(self, root):
        # write your code here

        if not root:
            return []

        currentLevel, nextLevel = 0, 0

        queue = [root]
        results = []
        values = []
        currentLevel = len(queue)

        while currentLevel:
            node = queue.pop(0)
            currentLevel -= 1
            if node.left:
                queue.append(node.left)
                nextLevel += 1
            if node.right:
                queue.append(node.right)
                nextLevel += 1

            values.append(node.val)
            if currentLevel == 0:
                currentLevel, nextLevel = nextLevel, currentLevel
                results.append(values)
                values = []

        return results
```

获赞 0　　添加评论

chenmengniu
更新于 6/9/2020, 7:04:03 AM

bfs 二叉树层级遍历模板程序，希望可以吸引些更好的Go答案

```
/**
* 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.
 * type TreeNode struct {
 *     Val int
 *     Left *TreeNode
 *     Right *TreeNode
 * }
 */

/**
 * @param root: A Tree
 * @return: Level order a list of lists of integer
 */

import (
     "container/list"
)

func levelOrder (root *TreeNode) [][]int {
    // write your code here
    ans := make([][]int, 0)

    if root == nil {
        return ans
    }

    queue := list.New()
    queue.PushBack(*root)

    for queue.Len() > 0 {
        size := queue.Len()

        level := make([]int, 0)
        for i := 0; i < size; i++ {
            currentElement := queue.Front()
            currentNode, _ := queue.Remove(currentElement).(Pair)

            level = append(level, currentNode.Val)

            if currentNode.Left != nil {
                queue.PushBack(*currentNode.Left)
            }

            if currentNode.Right != nil {
                queue.PushBack(*currentNode.Right)
            }
        }
        ans = append(ans, level)
    }
    return ans
}
```

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

(/)

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

(http://weibo.com/ninechapter) 知 (https://www.zhihu.com/people/crackinterview/)