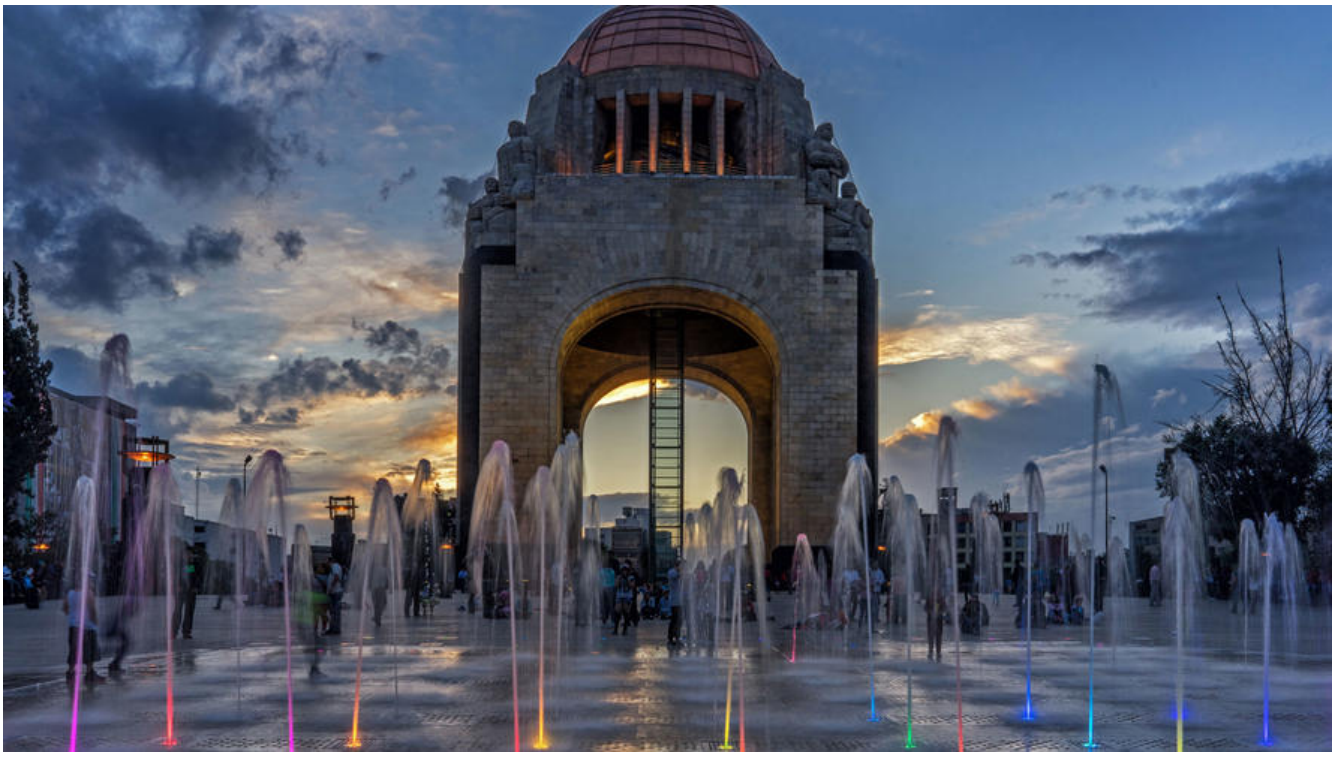
链滴

遍历二叉树

作者：cdq
原文链接：https://ld246.com/article/1623986316063
来源网站：链滴



广度遍历二叉树和广度遍历二叉树后进行分层

首先，函数接收到的是二叉树的根节点，我们需要将二叉树从第一层开始从左到右的打印出来，这里会用到广度优先搜索。

```
 例如:
给定二叉树：[3,9,20,null,null,15,7],
      3
//   / \
//  9  20
//    /  \
//   15   7
返回：[3,9,20,15,7]
```

这里可能就有人会想，我直接判断数组是否为空就行了，不为空的加入新的数组，最后返回这个数组行了，记住这里函数接收到的是二叉树的根节点，并不是一个数组，所以这种方法是不行的。我们在题里会用到队列来帮我们，队列的特性是先进先出，我们可以先将一层的节点从左到右全部放入队列再对队首的节点进行处理。

```
/**
 * @param {TreeNode} root
 * @return {number[]}
 */
var levelOrder = function(root) {
    if (!root){
        return [] // 如果二叉树为空，直接返回空数组
    }

    let queue = [root] ,res = []

    while (queue.length){//判断队列是否为空，即二叉树全部被遍历过后，条件不满足。
```

```
        const temp = queue.shift()//取出队首元素。
        res.push(temp.val)//将队首元素的值加入我们的结果集中。
        temp.left && queue.push(temp.left)//判断队首元素的左节点是否为空，如果不为空则加入
列
        temp.right && queue.push(temp.right)

    }
    return res
};
```

上面就是我们对二叉树从左到右的广度优先遍历，现在有个新问题，那就是上面的答案并不能看出来些节点是在同一层，所以下面我们来改进一下代码，让它有层次感。

```
/**
 * @param {TreeNode} root
 * @return {number[][]}
 */
var levelOrder = function(root) {
    if (!root){
        return []//根节点为空，直接返回[]
    }

    let queue = [root] ,res = [] ,level = 0//这里的level是指的二维数组的层，也就是二叉树的层

    while (queue.length){//循环一直执行到队列为空的时候
        res[level] = []//创建二维数组的第几层
        let nodeNum = queue.length//获取这一层的节点个数

        while (nodeNum --){//循环直到节点个数为0
            let temp = queue.shift()//从队列里取出队首节点
            res[level].push(temp.val)//将节点的值加入结果集

            temp.left && queue.push(temp.left)//判断该节点的左子树是否为空，不为空则加入队列
            temp.right && queue.push(temp.right)//判断该节点的右子树是否为空，不为空则加入队

        }
        level++//层数加1
    }
    return res

};
```

上面的函数实现了对二叉树的分层广度优先遍历，可以更好的看出二叉树的结构。一样的使用了队列特性来进行从左到右的遍历，层数则是通过该层队列的长度来控制。