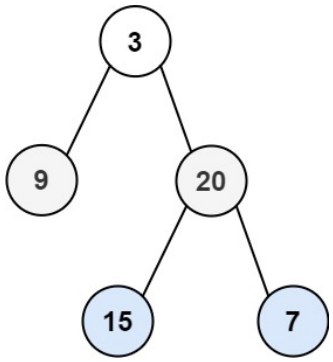
102. Binary Tree Level Order Traversal

Created	@October 28, 2021 9:54 PM
Difficulty	Medium
LC Url	https://leetcode.com/problems/binary-tree-level-order-traversal/
Importance	****
Tag	BFS NEET Queue Tree
Video	https://www.youtube.com/watch?v=MBZ-gBkjdMc

Given the `root` of a binary tree, return *the level order traversal of its nodes' values*. (i.e., from left to right, level by level).

Example 1:




```
Input: root = [3,9,20,null,null,15,7]
Output: [[3],[9,20],[15,7]]
```

Example 2:

```
Input: root = [1]
Output: [[1]]
```

Example 3:

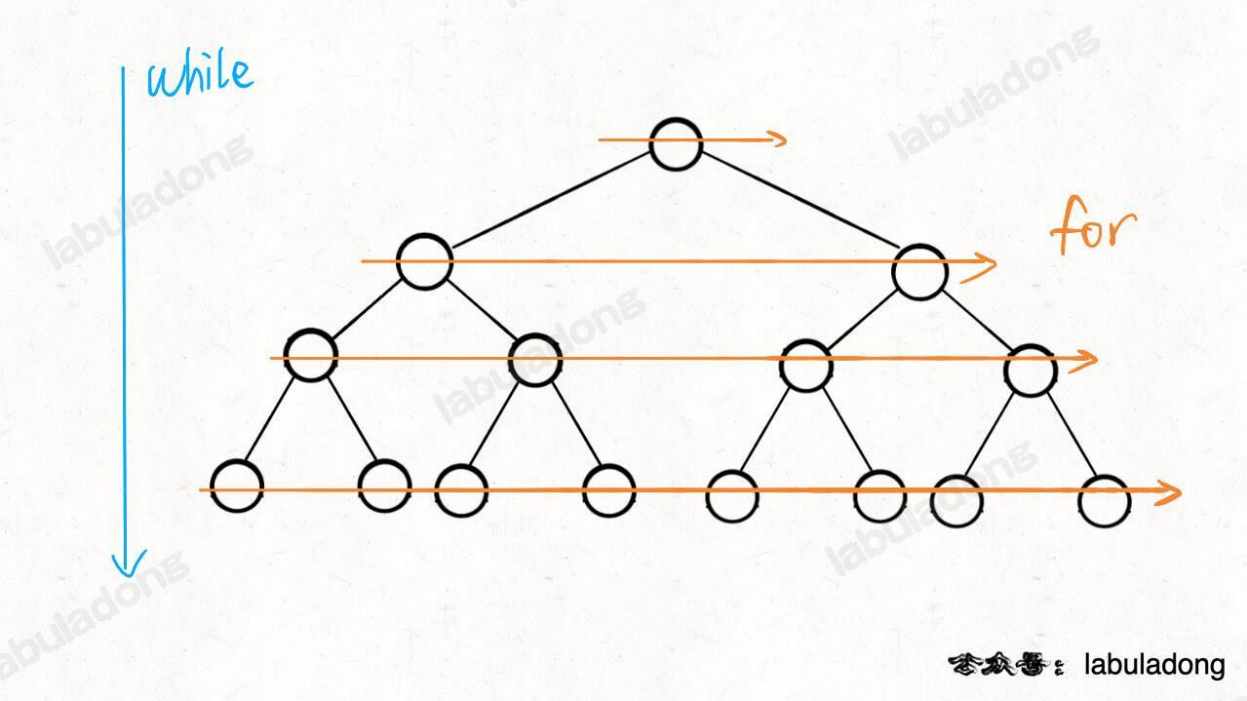
```
Input: root = []
Output: []
```

Constraints:

- The number of nodes in the tree is in the range `[0, 2000]`.
- `1000 <= Node.val <= 1000`

Solution

BFS




```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def levelOrder(self, root: Optional[TreeNode]) -> List[List[int]]:
        # 作者：fuxuemingzhu
        # 链接：https://leetcode.cn/problems/binary-tree-level-order-traversal/solution/tao-mo-ban-bfs-he-dfs-du-ke-yi-jie-jue-by-fuxuemin/
        queue = collections.deque()
        queue.append(root)
        res = []
        while queue:
            size = len(queue)
            level = []
            for _ in range(size):
                cur = queue.popleft()
                if not cur:
                    continue
                level.append(cur.val)
                queue.append(cur.left)
                queue.append(cur.right)
            if level:
                res.append(level)
        return res
```

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def levelOrder(self, root: Optional[TreeNode]) -> List[List[int]]:
        if root is None:
            return []

        queue = [root]
        next_queue = []
        level = []
        result = []
```

```
        while queue != []:
            for root in queue:
                level.append(root.val)
                if root.left is not None:
                    next_queue.append(root.left)
                if root.right is not None:
                    next_queue.append(root.right)
            result.append(level)
            level = []
            queue = next_queue
            next_queue = []
        return result
```