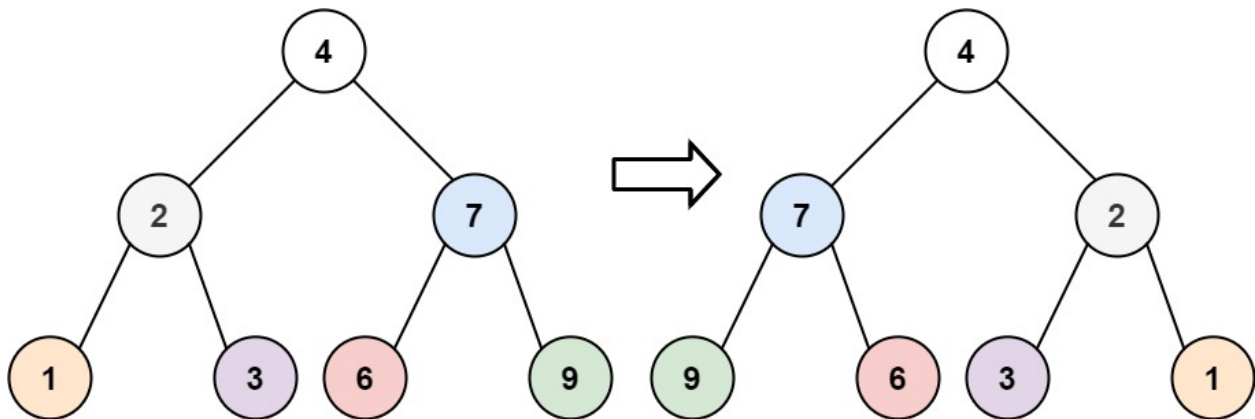


226. Invert Binary Tree

🕒 Created	@September 1, 2022 10:29 PM
📌 Difficulty	Easy
🔗 LC Url	https://leetcode.com/problems/invert-binary-tree/
📌 Importance	
🏷 Tag	DFS NEET Recursion Tree
📺 Video	

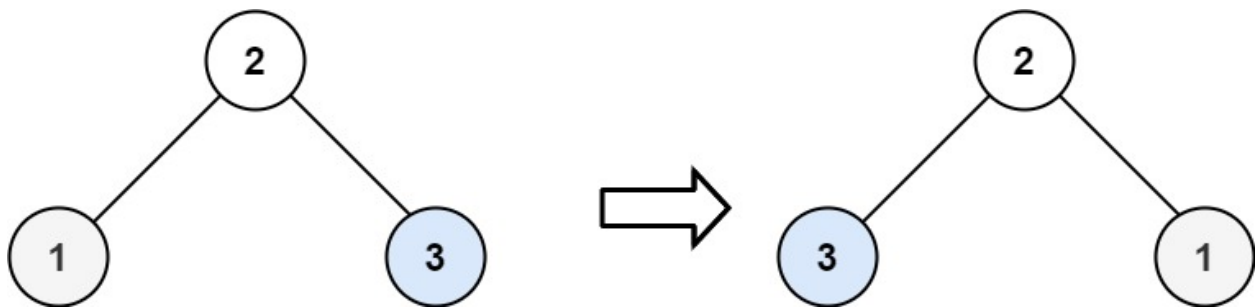
Given the `root` of a binary tree, invert the tree, and return *its root*.

Example 1:



Input: root = [4,2,7,1,3,6,9]
Output: [4,7,2,9,6,3,1]

Example 2:



```
Input: root = [2,1,3]
Output: [2,3,1]
```

Example 3:

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

Constraints:

- The number of nodes in the tree is in the range `[0, 100]`.
- `100 <= Node.val <= 100`

Solution

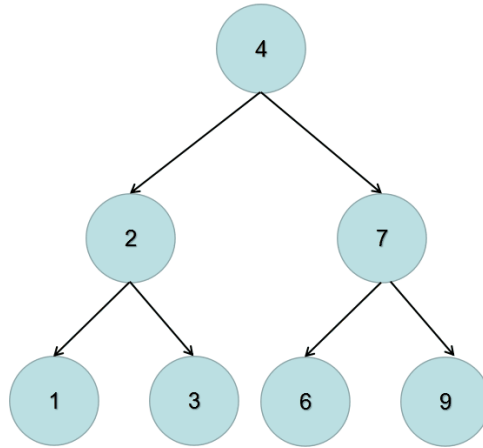
递归-DFS

- 终止条件：当前节点为 `null` 时返回
- 交换当前节点的左右节点，再递归的交换当前节点的左节点，递归的交换当前节点的右节点

时间复杂度：每个元素都必须访问一次，所以是 $O(n)$

空间复杂度：最坏的情况下，需要存放 $O(h)$ 个函数调用(h 是树的高度)，所以是 $O(h)$

代码实现如下：



```
# 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 invertTree(self, root: Optional[TreeNode]) -> Optional[TreeNode]:
        if not root:
            return None

        root.left, root.right = root.right, root.left
        self.invertTree(root.left)
        self.invertTree(root.right)
        return root
```

层序遍历-BFS

递归实现也就是深度优先遍历的方式，那么对应的就是广度优先遍历。

广度优先遍历需要额外的数据结构--队列，来存放临时遍历到的元素。

深度优先遍历的特点是一竿子插到底，不行了再退回来继续；而广度优先遍历的特点是层层扫荡。

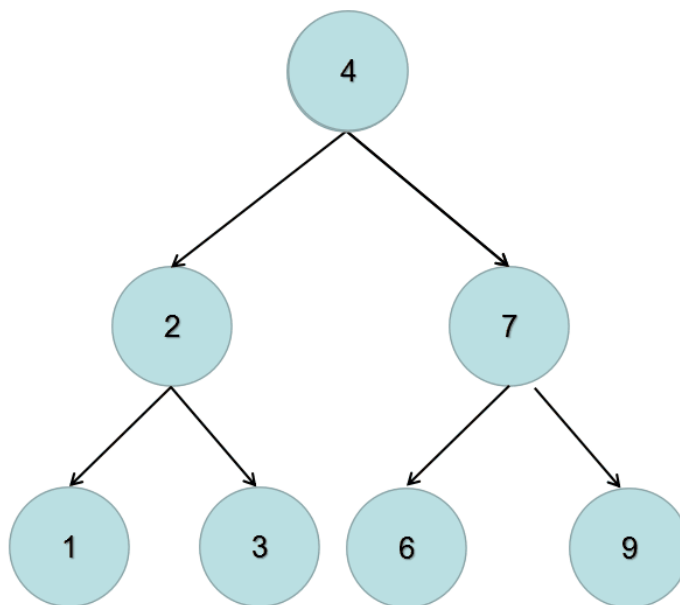
所以，我们需要先将根节点放入到队列中，然后不断的迭代队列中的元素。

对当前元素调换其左右子树的位置，然后：

- 判断其左子树是否为空，不为空就放入队列中
- 判断其右子树是否为空，不为空就放入队列中

时间复杂度：同样每个节点都需要入队列/出队列一次，所以是 $O(n)$

空间复杂度：最坏的情况下会包含所有的叶子节点，完全二叉树叶子节点是 $n/2$ 个，所以时间复杂度是 $O(n)$




```
# 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 invertTree(self, root: Optional[TreeNode]) -> Optional[TreeNode]:
        if not root:
            return None

        queue = [root]
        while queue:
            tmp = queue.pop(0)
            tmp.left, tmp.right = tmp.right, tmp.left
            if tmp.left:
                queue.append(tmp.left)
            if tmp.right:
```

```
        queue.append(tmp.right)
    return root
```

力扣

 <https://leetcode.cn/problems/invert-binary-tree/solution/dong-hua-yan-shi-liang-chong-shi-xian-226-fan-zhu-a/>