중앙을 기준으로 이진 트리를 반전시켜라.

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

```
# 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
```

1.dfs

```
import collections

class Solution:
    def invertTree(self, root: TreeNode) -> TreeNode:
        def dfs(node):
            if node == None:
                return
            dfs(node.left)
            dfs(node.right)
            node.left, node.right = node.right, node.left

        dfs(root)
        return root
```

```
class Solution:
    def invertTree(self, root: TreeNode) -> TreeNode:
        stack = collections.deque([root])
        while stack:
            node = stack.pop()
            if node:
                node.left, node.right = node.right, node.left
                stack.append(node.left)
                stack.append(node.right)
        return root
```

```
class Solution:
    def invertTree(self, root: TreeNode) -> TreeNode:
        stack = collections.deque([root])
        while stack:
            node = stack.pop()
            if node:
                stack.append(node.left)
                stack.append(node.right)
                node.left, node.right = node.right, node.left
        return root
```

2.bfs

```
class Solution:
    def invertTree(self, root: TreeNode) -> TreeNode:
        q = collections.deque([root])
        while q:
            node = q.popleft()
            if node:
                node.left, node.right = node.right, node.left
                q.append(node.left)
                q.append(node.right)
        return root
```

```
class Solution:
    def invertTree(self, root: TreeNode) -> TreeNode:
        q = collections.deque([root])
        while q:
            node = q.popleft()
            if node:
                q.append(node.left)
                q.append(node.right)
                node.left, node.right = node.right, node.left
        return root
```

3.파이썬 다운

```
class Solution:
    def invertTree(self, root: TreeNode) -> TreeNode:
        if root:
            root.left, root.right = self.invertTree(root.right), self.invertTree(root.left)
            return root
        return None
```