572. Subtree of Another Tree
We will use recursion to solve this question. We will preOrder traverse our tree and if root's values are the same, then we will compare these two subtrees. TC is O(ST)

```
class Solution:
    def isSubtree(self, s: TreeNode, t: TreeNode) -> bool:
        stack = [s]
        while stack:
            level = []
            for node in stack:
                if node.val == t.val:
                    if self.compareTree(node, t):
                        return True
                if node.left:
                    level.append(node.left)
                if node.right:
                    level.append(node.right)
            stack = level
        return False

    def compareTree(self, s, t):
        if not s or not t:
            return s == t
        return s.val == t.val and self.compareTree(s.left, t.left) and self.compareTree(s.right, t.right)
```

965. Univalued Binary Tree
Use recursion, straightforward.

```
class Solution:
    def isUnivalTree(self, root: TreeNode) -> bool:
        self.pre = None
        def traverse(node):
            if not node:
                return True
            if not traverse(node.left):
                return False
            if node.val == self.pre or self.pre is None:
                self.pre = node.val
            else:
                return False
            if not traverse(node.right):
                return False
            return True

        return traverse(root)
```

102. Binary Tree Level Order Traversal
Use level traversal.

```
class Solution:
    def levelOrder(self, root: TreeNode) -> List[List[int]]:
        result = []
        if not root:
            return result

        stack = [root]
        while stack:
            next_level = []
            cur_level = []
            for node in stack:
                cur_level.append(node.val)
                if node.left:
                    next_level.append(node.left)
                if node.right:
                    next_level.append(node.right)
            result.append(cur_level)
            stack = next_level
        return result
```

107. Binary Tree Level Order Traversal II
Same, using level traversal and return reversed(result)

```
class Solution:
    def levelOrderBottom(self, root: TreeNode) -> List[List[int]]:
        result = []
        if not root:
            return result

        stack = [root]
        while stack:
            next_level = []
            cur_level = []
            for node in stack:
                cur_level.append(node.val)
                if node.left:
                    next_level.append(node.left)
                if node.right:
                    next_level.append(node.right)
            result.append(cur_level)
            stack = next_level
        return reversed(result)
```

429. N-ary Tree Level Order Traversal

Same as the previous one. Use level traversal. TC is O(n).

```
class Solution:
    def levelOrder(self, root: 'Node') -> List[List[int]]:
        result = []
        if not root:
            return result

        stack = [root]
        while stack:
            next_level = []
            cur_level = []
            for node in stack:
                cur_level.append(node.val)
                next_level.extend(node.children)
            result.append(cur_level)
            stack = next_level
        return result
```