103. Binary Tree Zigzag Level Order Traversal
We will use iteration. TC is O(n)

```
class Solution:
    def zigzagLevelOrder(self, root: TreeNode) -> List[List[int]]:
        if not root:
            return []
        cur = [root]
        result = []
        reversed_mark = True
        while cur:
            next_ite = []
            cur_level_vals = []
            for node in cur:
                cur_level_vals.append(node.val)
                if node.left:
                    next_ite.append(node.left)
                if node.right:
                    next_ite.append(node.right)
            if reversed_mark:
                result.append(cur_level_vals)
            else:
                result.append(cur_level_vals[::-1])

            reversed_mark = not reversed_mark
            cur = next_ite
        return result
```

33. Search in Rotated Sorted Array
We will use set num as INF or -INF if nums[mid] and target are not in the same part. Then we will do the same binary search. TC is O(logn), SC is O(1)

```
class Solution:
    def search(self, nums: List[int], target: int) -> int:
        left, right = 0, len(nums)

        while left < right:
            mid = (left + right) // 2
            num = nums[mid]
            if not (nums[mid] < nums[0]) == (target < nums[0]):
                if target < nums[0]:
                    num = -float('inf')
                else:
                    num = float('inf')
            if num == target:
                return mid
            elif num > target:
                right = mid
            elif num < target:
                left = mid + 1
        return -1
```

75. Sort Colors

We will start from both ends and iterate from beginning to end. We will exchange with index of both ends if it’s 0 or 2. TC is O(n), SC is O(1)

```
 class Solution:
    def sortColors(self, nums: List[int]) -> None:
        """
        Do not return anything, modify nums in-place instead.
        """
        zero_idx, two_idx = 0, len(nums) - 1
        i = 0
        for i in range(len(nums)):
            while nums[i] == 2 and i < two_idx:
                nums[i], nums[two_idx] = nums[two_idx], nums[i]
                two_idx -= 1
            while nums[i] == 0 and i > zero_idx:
                nums[i], nums[zero_idx] = nums[zero_idx], nums[i]
                zero_idx += 1
```

402. Remove K Digits

We will use greedy and stack to get incremental subsequent array after deleting k digits. TC is O(n), SC is O(n)

```
from collections import deque
class Solution:
    def removeKdigits(self, num: str, k: int) -> str:
        if len(num) == k:
            return '0'
        stack = deque([])
        for i in num:
            while stack and i < stack[-1] and k > 0:
                stack.pop()
                k -= 1
            stack.append(i)

        while k > 0:
            stack.pop()
            k -= 1
        while stack and stack[0] == '0':
            stack.popleft()
        return ''.join(stack) if stack else '0'
```

19. Remove Nth Node From End of List

We will remove nth node from end of list by using two pointers. We will make the fast one move n first and then move the slow and fast at the same time until fast.next if None. Then delete the slower pointer’s next node. TC is O(n), SC is O(1)

```
class Solution:
    def removeNthFromEnd(self, head: ListNode, n: int) -> ListNode:
        dummy = ListNode(0)
        fast, slow = dummy, dummy
        dummy.next = head
        while n > 0:
            fast = fast.next
            n -= 1
        while fast.next:
            slow = slow.next
            fast = fast.next
        slow.next = slow.next.next
        return dummy.next
```