**Linked List**

2. Add two numbers

class Solution:

    def addTwoNumbers(self, l1: ListNode, l2: ListNode) -> ListNode:

        current = ListNode(0)

        temp = current

        carry = 0

        while l1 or l2 or carry:

            num1 = l1.val if l1 else 0

            num2 = l2.val if l2 else 0

            value = num1 + num2 + carry

            # when value > 9, only ten's digit is stored otherwise entire num

            current.next = ListNode(value%10)

            # Store the carry

            carry = value // 10

            # increment the linked lists

            l1 = l1.next if l1 else None

            l2 = l2.next if l2 else None

        return temp.next

19. Remove Nth node from the end

class Solution:

    def removeNthFromEnd(self, head: ListNode, n: int) -> ListNode:

        '''

        Approach 1: Move first till nth node and

        then move both first and second till first reaches end

        TC: O(n)

        SC: O(1)

        '''

        # base case

        if head.next is None:

            return

        temp = ListNode(0)

        temp.next = head

        i = 0

        first = temp

        second = temp

        while first.next is not None:

            if i < n:

                first = first.next

                i += 1

            else:

                second = second.next

                first = first.next

        # second will move till a node lesser than the node to be removed.

        # update the second's next with address of the next next node from second

        second.next = second.next.next

        return temp.next

237 Delete Node in a Linked List

328 Odd Even Linked List

class Solution:

    def oddEvenList(self, head: ListNode) -> ListNode:

        '''

        Approach 1:

        TC: O(n)

        SC: O(1)

        '''

        # base case

        if head is None:

            return

        odd\_head = head     # Initialize odd head

        even\_head = head.next # initialize even head

        temp = even\_head

        while  odd\_head.next is not None and temp.next is not None:

            odd\_head.next = temp.next

            odd\_head = odd\_head.next

            temp.next = odd\_head.next

            temp = temp.next

        # after odd nodes are linked and even nodes are linked. link the odd\_head to even\_head

        odd\_head.next = even\_head

        return head

21. Merge two sorted lists

23. Merge k sorted lists

141. Linked List cycle

class Solution:

    def hasCycle(self, head: ListNode) -> bool:

        # fast and slow pointer

        fast, slow = head, head

        # loop until the fast pointer reaches the last node

        while fast is not None and fast.next is not None:

            slow = slow.next

            fast = fast.next.next

            if slow == fast:

                return True

        return False

148. Linked List cycle 2

class Solution(object):

    def getIntersect(self, head):

        tortoise = head

        hare = head

        # A fast pointer will either loop around a cycle and meet the slow

        # pointer or reach the `null` at the end of a non-cyclic list.

        while hare is not None and hare.next is not None:

            tortoise = tortoise.next

            hare = hare.next.next

            if tortoise == hare:

                return tortoise

        return None

    def detectCycle(self, head):

        if head is None:

            return None

        # If there is a cycle, the fast/slow pointers will intersect at some

        # node. Otherwise, there is no cycle, so we cannot find an entrance to

        # a cycle.

        intersect = self.getIntersect(head)

        if intersect is None:

            return None

        # To find the entrance to the cycle, we have two pointers traverse at

        # the same speed -- one from the front of the list, and the other from

        # the point of intersection.

        ptr1 = head

        ptr2 = intersect

        while ptr1 != ptr2:

            ptr1 = ptr1.next

            ptr2 = ptr2.next

        return ptr1

Sort list

160. Intersection of Two Linked Lists

206. Reverse Linked List

234

Palindrome Linked List