Approach: Reversing the Linked list

1. Reverse both LL because we perform addition LSB to MSB and not the other way
2. While either of ll exist: keep adding it with carry(initially carry is 0), create new nodes for the sum generated at each step and keep connecting them
3. Finally if carry is not empty, suppose 5 + 5 = 10 so sum = 0 carry = 1 then we create a node for carry as well.
4. Now reverse the summed up linked list and return it

CODE:  
 l1 = self.reverse(l1)

l2 = self.reverse(l2)

res= sumhead = ListNode()

cur\_sum = 0

carry = 0

while l1 or l2:

cur\_sum = 0

if l1:

cur\_sum += l1.val

l1 = l1.next

if l2:

cur\_sum += l2.val

l2 = l2.next

cur\_sum += carry

carry = cur\_sum // 10

cur\_sum = cur\_sum % 10

sumhead.next = ListNode(cur\_sum)

sumhead = sumhead.next

if carry:

sumhead.next = ListNode(carry)

return self.reverse(res.next)

Approach: Without reversing’

1. Use a stack to store the values in each LL.
2. Because stack stores msb first it stays at the bottom and LSB comes at the top
3. Now perform addition while either stack is non empty
4. Follow same procedure as the previous

Time => O(m+n)

Space =< O(m+n)