

■ Add Two Numbers

Add two numbers stored as reversed linked lists — digit by digit

● Linked List

■ MEDIUM

■ Intuition First

Numbers are stored in REVERSE order: 342 is stored as 2→4→3. This is actually convenient — we add from least significant digit first (just like grade school addition). Walk both lists simultaneously, add digits plus carry, record the result digit, carry the overflow.

l1: 2 → 4 → 3 (represents 342)

l2: 5 → 6 → 4 (represents 465)

342 + 465 = 807

Step 1: 2+5+carry(0) = 7, carry=0 → node(7)

Step 2: 4+6+carry(0) = 10, carry=1 → node(0)

Step 3: 3+4+carry(1) = 8, carry=0 → node(8)

Both lists empty, carry=0 → stop

Result: 7 → 0 → 8 (represents 807) ✓

■ Java Code

```
1  class Solution {
2  public ListNode addTwoNumbers(ListNode l1, ListNode l2) {
3      ListNode dummy = new ListNode(0);
4      ListNode curr = dummy;
5      int carry = 0;
6
7      while (l1 != null || l2 != null || carry != 0) {
8          int val1 = (l1 != null) ? l1.val : 0;
9          int val2 = (l2 != null) ? l2.val : 0;
10
11         int sum = val1 + val2 + carry;
12         carry = sum / 10; // 1 if sum >= 10
13         curr.next = new ListNode(sum % 10);
14         curr = curr.next;
15
16         if (l1 != null) l1 = l1.next;
17         if (l2 != null) l2 = l2.next;
18     }
19     return dummy.next;
20 }
21 }
```

■ ■ Complexity

Type	Value	Meaning
Time	$O(\max(m,n))$	Process digits of the longer number
Space	$O(\max(m,n))$	Result list stores $\max(m,n)+1$ nodes

■ ■ Don't forget!

Continue the loop while $\text{carry} \neq 0$ even after both lists are exhausted. $999+1=1000$ — the carry creates an extra node!

■ Quick Revision

✓	Lists are in REVERSE order — add from head (least significant)
✓	$\text{sum} = l1.\text{val} + l2.\text{val} + \text{carry}$ at each step
✓	$\text{carry} = \text{sum}/10$, $\text{new digit} = \text{sum}\%10$
✓	Handle different length lists with null checks
✓	Loop continues while $\text{carry} \neq 0$ after lists end