Sunday, January 9, 2022 8:32 PM

# 2. Add Two Numbers

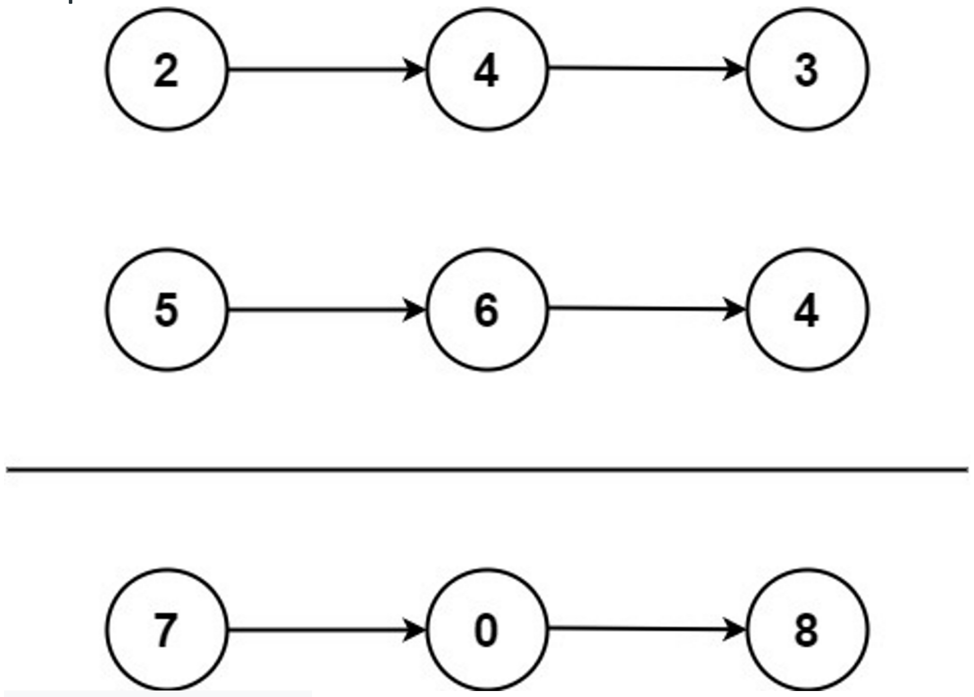
Medium

156593387Add to ListShare

You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order**, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

**Example 1:**



**Input:** l1 = [2,4,3], l2 = [5,6,4]
**Output:** [7,0,8]
**Explanation:** 342 + 465 = 807.
**Example 2:**
**Input:** l1 = [0], l2 = [0]
**Output:** [0]
**Example 3:**
**Input:** l1 = [9,9,9,9,9,9,9], l2 = [9,9,9,9]
**Output:** [8,9,9,9,0,0,0,1]

**Constraints:**

- The number of nodes in each linked list is in the range [1, 100].
- 0 <= Node.val <= 9
- It is guaranteed that the list represents a number that does not have leading zeros.

Accepted
2,415,624
Submissions
6,432,773

Seen this question in a real interview before?
Yes
No
Companies
Related Topics
[Linked ListMathRecursion]
Similar Questions
[Multiply Strings]
Medium

[Add Binary]
Easy

[Sum of Two Integers]
Medium

[Add Strings]
Easy

[Add Two Numbers II]
Medium

[Add to Array-Form of Integer]
Easy

[Add Two Polynomials Represented as Linked Lists]
Medium

From <https://leetcode.com/problems/add-two-numbers/>

# Solution

## Approach 1: Elementary Math

**Intuition**
Keep track of the carry using a variable and simulate digits-by-digits sum starting from the head of list, which contains the least-significant digit.
*Figure 1. Visualization of the addition of two numbers: 342 + 465 = 807342+465=807.*
*Each node contains a single digit and the digits are stored in reverse order.*

**Algorithm**
Just like how you would sum two numbers on a piece of paper, we begin by summing the least-significant digits, which is the head of l1/1 and l2/2. Since each digit is in the range of 0 \ldots 90…9, summing two digits may "overflow". For example 5 + 7 = 125+7=12. In this case, we set the current digit to 22 and bring over the carry = 1*carry*=1 to the next iteration. carry*carry* must be either 00 or 11 because the largest possible sum of two digits (including the carry) is 9 + 9 + 1 = 199+9+1=19.
The pseudocode is as following:

- Initialize current node to dummy head of the returning list.

- Initialize carry to 00.
- Initialize *pp* and *qq* to head of l1*l*1 and l2*l*2 respectively.
- Loop through lists l1*l*1 and l2*l*2 until you reach both ends.
    - Set *xx* to node *pp*'s value. If *pp* has reached the end of l1*l*1, set to 00.
    - Set *yy* to node *qq*'s value. If *qq* has reached the end of l2*l*2, set to 00.
    - Set sum = x + y + carry*sum*=*x*+*y*+*carry*.
    - Update carry = sum / 10*carry*=*sum*/10.
    - Create a new node with the digit value of (sum \bmod 10)(*sum*mod10) and set it to current node's next, then advance current node to next.
    - Advance both *pp* and *qq*.
- Check if carry = 1*carry*=1, if so append a new node with digit 11 to the returning list.
- Return dummy head's next node.

Note that we use a dummy head to simplify the code. Without a dummy head, you would have to write extra conditional statements to initialize the head's value.
Take extra caution of the following cases:

| **Test case** | **Explanation** |
|---|---|
| l1=[0,1]*l*1=[0,1]<br>l2=[0,1,2]*l*2=[0,1,2] | When one list is longer than the other. |
| l1=[]*l*1=[]<br>l2=[0,1]*l*2=[0,1] | When one list is null, which means an empty list. |
| l1=[9,9]*l*1=[9,9]<br>l2=[1]*l*2=[1] | The sum could have an extra carry of one at the end, which is easy to forget. |

**Complexity Analysis**

- Time complexity : O(\max(m, n))*O*(max(*m*,*n*)). Assume that m*m* and n*n* represents the length of l1*l*1 and l2*l*2 respectively, the algorithm above iterates at most \max(m, n)max(*m*,*n*) times.
- Space complexity : O(\max(m, n))*O*(max(*m*,*n*)). The length of the new list is at most \max(m,n) + 1max(*m*,*n*)+1.

**Follow up**

What if the the digits in the linked list are stored in non-reversed order? For example:
(3 \to 4 \to 2) + (4 \to 6 \to 5) = 8 \to 0 \to 7(3→4→2)+(4→6→5)=8→0→7

From <https://leetcode.com/problems/add-two-numbers/solution/>

```java
public class addTwoNumbers{
    public ListNode addTwoNumbers(ListNode l1, ListNode l2) {
        ListNode dummyHead = new ListNode(0);
        ListNode p = l1, q = l2, curr = dummyHead;
        int carry = 0;
        while (p != null || q != null) {
            int x = (p != null) ? p.val : 0;
            int y = (q != null) ? q.val : 0;
            int sum = carry + x + y;
            carry = sum / 10;
            curr.next = new ListNode(sum % 10);
            curr = curr.next;
            if (p != null) p = p.next;
            if (q != null) q = q.next;
        }
        if (carry > 0) {
            curr.next = new ListNode(carry);
        }
        return dummyHead.next;
    }
    public static void main(String[] args) {

    }
}
```