2. Add Two Numbers

April 5, 2016 | 1.1M views

Average Rating: 4.67 (699 votes)

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 contain a single digit. Add the two numbers and return it as a linked list.

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

Example:

```
Input: (2 -> 4 -> 3) + (5 -> 6 -> 4)
Output: 7 -> 0 -> 8
Explanation: 342 + 465 = 807.
```

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.

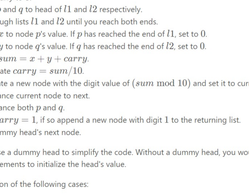

Figure 1. Visualization of the addition of two numbers: $342 + 465 = 807$. Each node contains a single digit and the digits are stored in reverse order.

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 $l1$ and $l2$. Since each digit is in the range of $0 \ldots 9$, summing two digits may "overflow". For example $5 + 7 = 12$. In this case, we set the current digit to 2 and bring over the $carry = 1$ to the next iteration. $carry$ must be either 0 or 1 because the largest possible sum of two digits (including the carry) is $9 + 9 + 1 = 19$.

The pseudocode is as following:

- Initialize current node to dummy head of the returning list.
- Initialize carry to 0.
- Initialize p and q to head of $l1$ and $l2$ respectively.
- Loop through lists $l1$ and $l2$ until you reach both ends.
  - Set x to node p's value. If p has reached the end of $l1$, set to 0.
  - Set y to node q's value. If q has reached the end of $l2$, set to 0.
  - Set $sum = x + y + carry$.
  - Update $carry = sum / 10$.
  - Create a new node with the digit value of $(sum \bmod 10)$ and set it to current node's next, then advance current node to next.
  - Advance both p and q.
- Check if $carry = 1$, if so append a new node with digit 1 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
$l1 = [0, 1]$ $l2 = [0, 1, 2]$	When one list is longer than the other.
$l1 = []$ $l2 = [0, 1]$	When one list is null, which means an empty list.
$l1 = [9, 9]$ $l2 = [1]$	The sum could have an extra carry of one at the end, which is easy to forget.

Java

Copy

```
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;
}
```

Complexity Analysis

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

Follow up

What if the the digits in the linked list are stored in non-reversed order? For example:

$$(3 \to 4 \to 2) + (4 \to 6 \to 5) = 8 \to 0 \to 7$$

Rate this article:

Comments: 939

Sort By

Type comment here... (Markdown is supported)

Preview Post

sanwave ★ 333 November 24, 2018 9:20 AM

My answer(C++)

```
class Solution {
public:
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
```

Read More

330 Share Reply

SHOW 10 REPLIES

MikeZLin ★ 333 November 20, 2018 9:40 AM

My Python3 implementation - 102ms

```
class Solution:
    def addTwoNumbers(self, l1, l2):
        """
```

Read More

331 Share Reply

SHOW 27 REPLIES

anhduc130 ★ 284 December 31, 2018 10:04 AM

My JS solution takes 112ms (faster than 99.54% of JS submissions)

```
/**
 * Definition for singly-linked list.
 * function ListNode(val) {
```

Read More

283 Share Reply

SHOW 10 REPLIES

badrabbit ★ 357 December 22, 2018 11:19 AM

runtime 68ms python

```
# Definition for singly-linked list.
# class ListNode(object):
#     def __init__(self, x):
```

Read More

357 Share Reply

SHOW 22 REPLIES

cgio ★ 121 July 24, 2018 10:21 PM

Here's my Python 3 solution. Not the fastest due to conversions, but the most readable here I've seen and 10 lines. What do you think?

```
class Solution:
    def addTwoNumbers(self, l1, l2):
```

Read More

120 Share Reply

SHOW 9 REPLIES

X-N2O ★ 125 May 15, 2019 8:11 PM

Alternate solution:

Instead of dealing with all the edge cases, just convert the linked lists to integers, perform the addition, and convert the sum to a linked list and return it.

Read More

125 Share Reply

SHOW 14 REPLIES

fly_uper ★ 81 November 16, 2018 10:31 PM

C++

81 Share Reply

SHOW 2 REPLIES

Navice ★ 42 December 13, 2018 7:43 PM

carry 即进位

42 Share Reply

SHOW 3 REPLIES

b45i ★ 40 August 10, 2018 5:03 PM

Python:

```
class Solution(object):
    def to_num(self, l1):
        """
```

Read More

40 Share Reply

SHOW 1 REPLY

AliaYoung ★ 48 June 29, 2018 1:00 PM

Here's my Python version using recursion. Complexity is O(max(m,n)) where m and n are the lengths of the list.

```
# Definition for singly-linked list.
# class ListNode(object):
```

Read More

48 Share Reply

SHOW 3 REPLIES