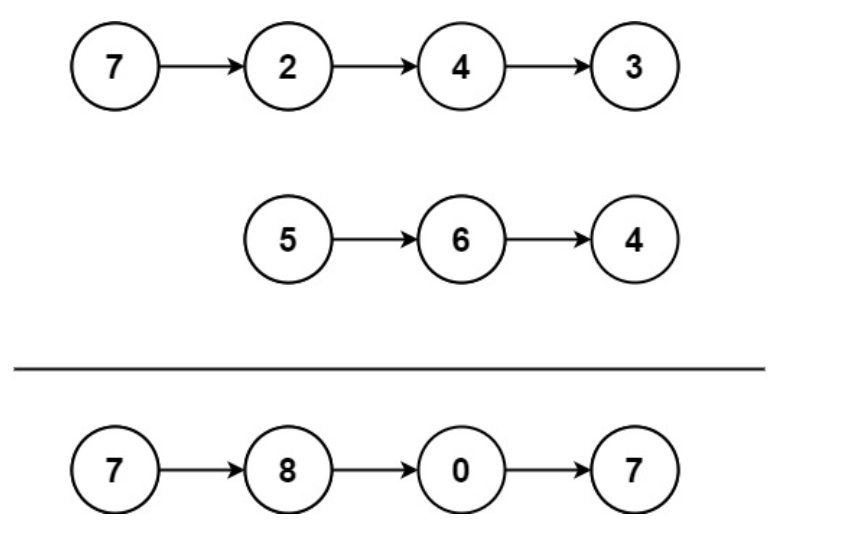
# 题目

给你两个 非空 链表来代表两个非负整数。数字最高位位于链表开始位置。它们的每个节点只存储一位数字。将这两数相加会返回一个新的链表。

你可以假设除了数字 0 之外，这两个数字都不会以零开头。

示例1：



输入：l1 = [7,2,4,3], l2 = [5,6,4]

输出：[7,8,0,7]

示例2：

输入：l1 = [2,4,3], l2 = [5,6,4]

输出：[8,0,7]

示例3：

输入：l1 = [0], l2 = [0]

输出：[0]

提示：

链表的长度范围为 [1, 100]

0 <= node.val <= 9

输入数据保证链表代表的数字无前导 0

进阶：如果输入链表不能翻转该如何解决？

# 分析

## 方法一：栈

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

\* ListNode() : val(0), next(nullptr) {}

\* ListNode(int x) : val(x), next(nullptr) {}

\* ListNode(int x, ListNode \*next) : val(x), next(next) {}

\* };

\*/

class Solution {

public:

ListNode\* addTwoNumbers(ListNode\* l1, ListNode\* l2) {

stack<int> stack1, stack2;

// 将链表节点值依次入栈

while (l1) {

stack1.push(l1->val);

l1 = l1->next;

}

while (l2) {

stack2.push(l2->val);

l2 = l2->next;

}

ListNode\* dummy = nullptr;

int carry = 0;

// 从栈中依次取出节点值相加，并构建新链表

while (!stack1.empty() || !stack2.empty() || carry != 0) {

int sum = carry;

if (!stack1.empty()) {

sum += stack1.top();

stack1.pop();

}

if (!stack2.empty()) {

sum += stack2.top();

stack2.pop();

}

carry = sum / 10;

ListNode\* newNode = new ListNode(sum % 10);

newNode->next = dummy;

dummy = newNode;

}

return dummy;

}

};