# 题目

给定两个用链表表示的整数，每个节点包含一个数位。

这些数位是反向存放的，也就是个位排在链表首部。

编写函数对这两个整数求和，并用链表形式返回结果。

**示例：**

输入：(7 -> 1 -> 6) + (5 -> 9 -> 2)，即617 + 295

输出：2 -> 1 -> 9，即912

进阶：假设这些数位是正向存放的，请再做一遍。

**示例：**

输入：(6 -> 1 -> 7) + (2 -> 9 -> 5)，即617 + 295

输出：9 -> 1 -> 2，即912

# 分析

**思路：**

先对应位求和（位数少的数对应位不存在就用0加）

1、加上上一次的进位

2、得到当前位

3、记录进位

当位数大的数遍历完（遍历完较长的链表）且进位也为0的时候就可以停止了。

**代码：**

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

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

\* };

\*/

class Solution

{

public:

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

{

ListNode \*head = new ListNode(-1), \*p1 = l1, \*p2 = l2, \*p = head;//用带头节点的可以少一点初始的特判

int sum = 0, carr = 0;

while (p1 || p2 || carr) //如果改用&&则while结束还要多一些特判

{

sum = 0;//当前两位数字和

if(p1)

{

sum += (p1->val);

p1 = p1->next;

}

if(p2)

{

sum += (p2->val);

p2 = p2->next;

}

sum += carr; //加上上一位的进位

ListNode \*t = new ListNode(sum % 10); //得到当前位数字

carr = sum / 10; //得到当前位对下一位的进位

p->next = t;//当前位连接上去

p = p->next;//游标指针更新

}

return head->next;

}

};