# 题目

给出两个非空的链表用来表示两个非负的整数。其中，它们各自的位数是按照逆序的方式存储的，并且它们的每个节点只能存储 一位 数字。

如果，我们将这两个数相加起来，则会返回一个新的链表来表示它们的和。

您可以假设除了数字0之外，这两个数都不会以0开头。

示例：

输入：(2 -> 4 -> 3) + (5 -> 6 -> 4)

输出：7 -> 0 -> 8

原因：342 + 465 = 807

# 分析

## 方法一：遍历

**思路：**

由于输入的两个链表都是逆序存储数字的位数的，因此两个链表中同一位置的数字可以直接相加。

我们同时遍历两个链表，逐位计算它们的和，并与当前位置的进位值相加。具体而言，如果当前两个链表处相应位置的数字为n1,n2，进位值为carry，则它们的和为n1+n2+carry；其中，答案链表处相应位置的数字为(n1+n2+carry)mod10，而新的进位值为[n1+n2+carry]/10。

如果两个链表的长度不同，则可以认为长度短的链表的后面有若干个0。

此外，如果链表遍历结束后，有carry>0，还需要在答案链表的后面附加一个节点，节点的值为carry。

**代码：**

class Solution {

public:

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

ListNode \*head = nullptr, \*tail = nullptr;

int carry = 0;

while (l1 || l2) {

int n1 = l1 ? l1->val: 0;

int n2 = l2 ? l2->val: 0;

int sum = n1 + n2 + carry;

if (!head) {

head = tail = new ListNode(sum % 10);

} else {

tail->next = new ListNode(sum % 10);

tail = tail->next;

}

carry = sum / 10;

if (l1) {

l1 = l1->next;

}

if (l2) {

l2 = l2->next;

}

}

if (carry > 0) {

tail->next = new ListNode(carry);

}

return head;

}

};

**复杂度分析：**

时间复杂度：O(max(m,n))，其中m和n分别为两个链表的长度。我们要遍历两个链表的全部位置，而处理每个位置只需要O(1)的时间。

空间复杂度：O(1)。注意返回值不计入空间复杂度。

另一种写法：

class Solution {

public:

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

ListNode\* head=new ListNode(-1);//存放结果的链表

ListNode\* h=head;//移动指针

int sum=0;//每个位的加和结果

bool carry=false;//进位标志

while(l1!=NULL||l2!=NULL)

{

sum=0;

if(l1!=NULL)

{

sum+=l1->val;

l1=l1->next;

}

if(l2!=NULL)

{

sum+=l2->val;

l2=l2->next;

}

if(carry)

sum++;

h->next=new ListNode(sum%10);

h=h->next;

carry=sum>=10?true:false;

}

if(carry)

{

h->next=new ListNode(1);

}

return head->next;

}

};

## 方法二：双栈

**思路：**

**代码：**