1. 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

1. 思路
2. 实质是将两个链表相加，生成一个新的链表。
3. 链表中的每一个节点存储一位数字，两个链表对应节点相加，则两个链表的长度需要对齐，将短的链表缺失部分补为0
4. 数字相加有进位需要考虑，有进位，则向后面的节点相加结果进位1。
5. 当两个链表都遍历完毕后，注意最后还可能有一个进位。
6. C++代码实现

/\*\*

 \* 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) {

        //新建节点表示新的链表,用0初始化

        ListNode\* l3 = new ListNode(0);

        //参数准备,两个链表相加

        ListNode\* p = l1, \*q = l2, \*curr = l3;

        //进位的标志

        int mov = 0;

        //当链表都没有到达末尾时

        while(p != NULL || q != NULL){

            //当一个链表到达末尾后，之后的值都为0

            int x = p != NULL ? p->val : 0;

            int y = q != NULL ? q->val : 0;

            //每次计算综合需要考虑上一次的mov,第一次为0

            int sum = x + y + mov;

            //进位用商来求

            mov = sum / 10;

            //新节点的值用mod来求

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

            //链接

            curr = curr->next;

            //两个链表节点后移

            if(p!= NULL){

                p = p->next;

            }

            if(q!= NULL){

                q = q->next;

            }

        }

        //两个链表都遍历完毕后,注意最后的相加可能还会进位

        if(mov > 0){

            curr->next = new ListNode(mov);

        }

        return l3->next;

    }

};

1. Java代码实现

/\*\*

 \* Definition for singly-linked list.

 \* public class ListNode {

 \*     int val;

 \*     ListNode next;

 \*     ListNode(int x) { val = x; }

 \* }

 \*/

class Solution {

    public ListNode addTwoNumbers(ListNode l1, ListNode l2) {

        //新建节点表示新的链表,用0初始化

        ListNode l3 = new ListNode(0);

        //参数准备,两个链表相加

        ListNode p = l1, q = l2, curr = l3;

        //进位的标志

        int mov = 0;

        //当链表都没有到达末尾时

        while(p != null || q != null){

            //当一个链表到达末尾后，之后的值都为0

            int x = p != null ? p.val : 0;

            int y = q != null ? q.val : 0;

            //每次计算综合需要考虑上一次的mov,第一次为0

            int sum = x + y + mov;

            //进位用商来求

            mov = sum / 10;

            //新节点的值用mod来求

            curr.next = new ListNode(sum % 10);

            //链接

            curr = curr.next;

            //两个链表节点后移

            if(p!= null){

                p = p.next;

            }

            if(q!= null){

                q = q.next;

            }

        }

        //两个链表都遍历完毕后,注意最后的相加可能还会进位

        if(mov > 0){

            curr.next = new ListNode(mov);

        }

        return l3.next;

    }

}