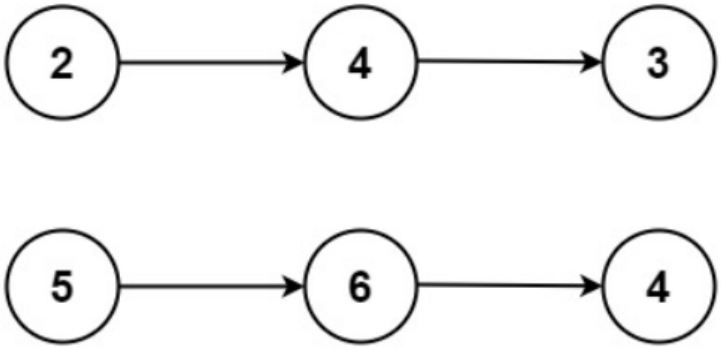
2. 两数相加

给你两个 **非空** 的链表，表示两个非负的整数。它们每位数字都是按照 **逆序** 的方式存储的，并且每个节点只能存储 **一位** 数字。

请你将两个数相加，并以相同形式返回一个表示和的链表。

你可以假设除了数字 0 之外，这两个数都不会以 0 开头。

示例 1：




```
//用法:提供一个head,tail思路,当head不存在,head,tail都指向同一个,存在的时候就是把tail->next添加
class Solution {
public:
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
        int carry = 0;
        ListNode *head = nullptr, *tail = nullptr;
        while (l1 || l2) {
            int n1 = l1 ? l1->val : 0;
            int n2 = l2 ? l2->val : 0;
            int sum = n1 + n2 + carry;
            if (head) {
                tail->next = new ListNode(sum % 10);
                tail = tail->next;
            } else head = tail = new ListNode(sum % 10);
            carry = sum / 10;
            if (l1) l1 = l1->next;
            if (l2) l2 = l2->next;
        }
        if(carry) tail->next = new ListNode(carry);
        return head;
    }
};
```

```
//递归
//对于链表连说,返回的是的头,而递归正好是从后往前算的,所以最后一次返回就是头节点,就是答案
```

```
//这里需要注意的是递归结束的条件,就是当l1,和l2都为空时,再判断carry,如果!=0,就要在new一个新的节点
//还需要注意的是,这里是以l1作为答案返回,一定得确保l1长于l2,如果发现l1为空就交换l2,
class Solution {
public:
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2, int carry = 0) {
        if (l1 == nullptr && l2 == nullptr) return carry ? new ListNode(carry) : nullptr;

        if(l1 == nullptr) swap(l1, l2);
        carry += l1->val + (l2 ? l2->val : 0);
        l1->val = carry % 10;
        l1->next = addTwoNumbers(l1->next, (l2 ? l2->next : nullptr), carry / 10);
        return l1;
    }
};
```