2 两数相加

Label：链表

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：

输入：l1 = [2,4,3], l2 = [5,6,4]
输出：[7,0,8]

输入：l1 = [9,9,9,9,9,9,9], l2 = [9,9,9,9]
输出：[8,9,9,9,0,0,0,1]

- 遍历，用变量存储进位，普通的加法运算本身就是从个位开始进行

```
class Solution {
    public ListNode addTwoNumbers(ListNode l1, ListNode l2) {
        ListNode current1 = l1;
        ListNode current2 = l2;
        ListNode head = new ListNode();
        ListNode current = head;
        int carry = 0;

        while (current1 != null || current2 != null || carry!= 0 ) {
            int addResult = 0;

            if (current1 == null && current2 != null) {
                addResult = 0 + current2.val + carry;
            } else if (current1 != null && current2 == null) {
                addResult = current1.val + 0 + carry;
            } else if(current1 != null && current2 != null){
                addResult = current1.val + current2.val + carry;
            } else{  // current1 == null && current2 == null
                addResult = carry;
            }

            current.val = addResult % 10; // 余数作为当前节点的值
            // new carry
            carry = addResult / 10;

            if (current1 != null) current1 = current1.next;
            if (current2 != null) current2 = current2.next;

            if (current1 != null || current2 != null || carry!= 0) {
                current.next = new ListNode();
                current = current.next;
            }
        }
        return head;
    }
}
```

- 递归

```
/**
 * 1.因为两个数字相加会产生进位，所以使用i来保存进位。
 * 2.当前位的值为(l1.val + l2.val + i) % 10
 * 3.进位值为(l1.val + l2.val + i) / 10
 * 4.建立新node，然后将进位传入下一层
*/

class Solution {
    public ListNode addTwoNumbers(ListNode l1, ListNode l2) {
        return dfs(l1, l2, 0);
    }

    ListNode dfs(ListNode l1, ListNode l2, int i) {

        // 退出条件
        if (l1 == null && l2 == null && i == 0)
            return null;

        int sum = (l1 != null ? l1.val : 0) + (l2 != null ? l2.val : 0) + i;
        ListNode node = new ListNode(sum % 10);

        // 继续递归
        node.next = dfs(l1 != null ? l1.next : null, l2 != null ? l2.next :
null, sum / 10);

        return node;

    }
}
```

- 队列

```
class Solution {
    public ListNode addTwoNumbers(ListNode l1, ListNode l2) {

        Queue<ListNode> queue1 = new LinkedList<>();
        Queue<ListNode> queue2 = new LinkedList<>();
        // 入队
        while (l1 != null){
            queue1.add(l1);
            l1 = l1.next; // 后续使用不到，直接覆盖
        }
        while (l2 != null){
            queue2.add(l2);
            l2 = l2.next;
        }

        // 先进先出
        int carry = 0;
        ListNode head = null;
        ListNode curr = head;

        while (!queue1.isEmpty() || !queue2.isEmpty()) {
            int l1Val = queue1.isEmpty()? 0 : queue1.poll().val;
            int l2Val = queue2.isEmpty()? 0 : queue2.poll().val;
            int add = l1Val + l2Val + carry;

            if (head == null){
                head = new ListNode(add%10);
                curr = head;
            }else { // 不是第一次
                curr.next = new ListNode(add%10);
                curr = curr.next;
            }
            carry = add / 10;
        }

        if (carry != 0) {
            curr.next = new ListNode(carry);
        }
        return head;
    }
}
```