# 题目

输入两个递增排序的链表，合并这两个链表并使新链表中的节点仍然是递增排序的。

**示例1：**

**输入：**1->2->4, 1->3->4

**输出：**1->1->2->3->4->4

**限制：**

0 <= 链表长度 <= 1000

**类似题目：Leetcode 21**

# 分析

## 方法一：递归

**思路：**

编写递归的第一步，应当是明确当前函数应当完成的功能。

**函数功能：**

返回l1指向的结点和l2指向的结点中，值较小的结点

并将从下级函数获得的返回值，链接到当前结点尾部

**函数结束条件：**

当l1为空，或l2为空，函数结束

返回l1或l2中剩下的部分

**代码：**

class Solution {

public:

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

if (l1 == NULL) {

return l2;

}

if (l2 == NULL) {

return l1;

}

if (l1->val <= l2->val) {

l1->next = mergeTwoLists(l1->next, l2);

return l1;

}

l2->next = mergeTwoLists(l1, l2->next);

return l2;

}

};

## 方法二：迭代

**思路：**

1、定义头结点

2、若l1指向的结点值<l2指向的结点值，则将l1链接到头结点的next位置

3、否则将l2链接到头结点的next位置

4、循环进行，直至l1或 l2为NULL

5、最后，将l1或l2中剩下的部分，链接到头结点后面

**代码：**

class Solution {

public:

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

ListNode\* head = new ListNode(1);

ListNode\* ret = head;

while (l1 != NULL && l2 != NULL) {

if (l1->val < l2->val) {

head->next = l1;

l1 = l1->next;

} else {

head->next = l2;

l2 = l2->next;

}

head = head->next;

}

head->next = l1 == NULL ? l2 : l1;

return ret->next;

}

};