21-合并两个有序链表

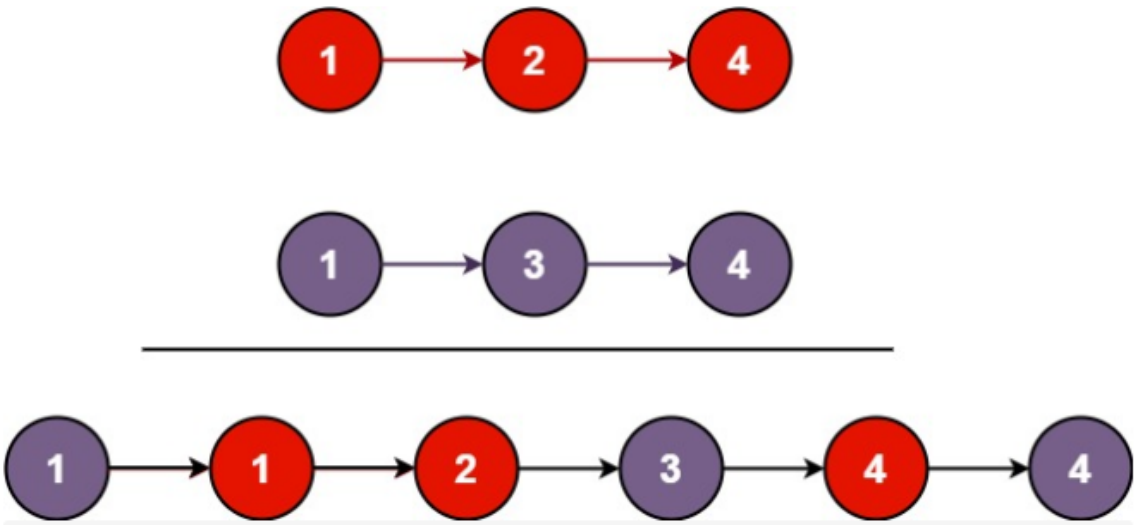
题述

21. 合并两个有序链表

难度 简单 1879 收藏 分享 切换为英文 接收动态 反馈

将两个升序链表合并为一个新的 **升序** 链表并返回。新链表是通过拼接给定的两个链表的所有节点组成的。

示例 1：




```
输入：l1 = [1,2,4], l2 = [1,3,4]
输出：[1,1,2,3,4,4]
```

示例 2：

```
输入：l1 = [], l2 = []
输出：[]
```

示例 3：

```
输入：l1 = [], l2 = [0]
输出：[0]
```

浅析

解法一：递归

$$\begin{cases} list1[0] + merge(list1[1:], list2) & list1[0] < list2[0] \\ list2[0] + merge(list1, list2[1:]) & otherwise \end{cases}$$

两个链表头部值较小的一个节点与剩下元素的merge操作结果合并。

如果l1或者l2一开始就是空链表，那么没有任何操作需要合并，只需返回非空链表。

代码

解法一：递归

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* l1, ListNode* l2)
    {
        //解法一：递归合并链表
        if(l1==nullptr)
        {
            return l2;
        }
        else if(l2==nullptr)
        {
            return l1;
        }
        else if(l1->val < l2->val)
        {
            l1->next=mergeTwoLists(l1->next,l2);
            return l1;
        }
        else
        {
            l2->next=mergeTwoLists(l1,l2->next);
            return l2;
        }

    }
};
```

解法二：迭代

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
```

```
 */
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* l1, ListNode* l2)
    {
        ListNode* preHead = new ListNode(-1);

        ListNode* prev = preHead;
        while (l1 != nullptr && l2 != nullptr) {
            if (l1->val < l2->val) {
                prev->next = l1;
                l1 = l1->next;
            } else {
                prev->next = l2;
                l2 = l2->next;
            }
            prev = prev->next;
        }

        // 合并后 l1 和 l2 最多只有一个还未被合并完，我们直接将链表末尾指向未合并完的链表即可
        prev->next = l1 == nullptr ? l2 : l1;

        return preHead->next;
    }
};
```

AC

题目描述 | 评论 (1.8k) | 题解 (3.3k) | 提交记录

执行结果： 通过 显示详情 › 添加备

执行用时： 4 ms , 在所有 C++ 提交中击败了 94.66% 的用户

内存消耗： 14.4 MB , 在所有 C++ 提交中击败了 67.61% 的用户

通过测试用例： 208 / 208

炫耀一下：

写题解，分享我的解题思路

提交结果	执行用时	内存消耗	语言	提交时间	备注
通过	4 ms	14.4 MB	C++	2021/09/02 14:31	添
解答错误	N/A	N/A	C++	2021/09/02 14:30	添
通过	8 ms	14.5 MB	C++	2021/09/02 14:22	添
通过	0 ms	14.4 MB	C++	2021/09/02 14:22	添