面试题25：合并两个排序的链表

题目描述

输入两个单调递增的链表，输出两个链表合成后的链表，当然我们需要合成后的链表满足单调不减规则。

解题思路

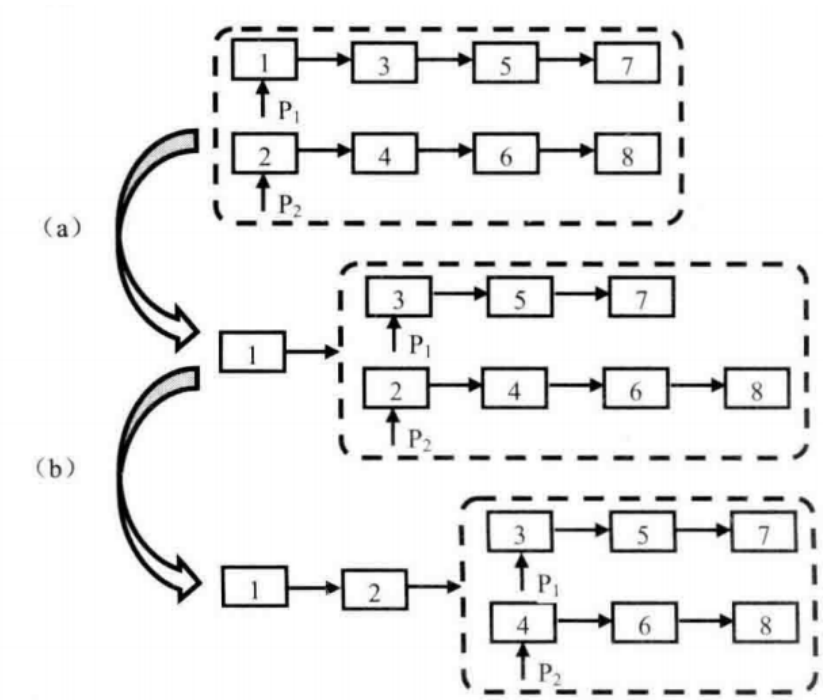

图 3.12　合并两个递增链表的过程

注：（a）链表 1 的头节点的值小于链表 2 的头节点的值，因此链表 1 的头节点是合并后链表的头节点。（b）在剩余的节点中，链表 2 的头节点的值小于链表 1 的头节点的值，因此链表 2 的头节点是剩余节点的头节点，把这个节点和之前已经合并好的链表的尾节点链接起来。

这是一道经常被各公司采用的面试题。在面试过程中，我们发现应聘者最容易犯两种错误：一是在写代码之前没有想清楚合并的过程，最终合并出来的链表要么中间断开了、要么并没有做到递增排序；二是代码在鲁棒性方面存在问题，程序一旦有特殊的输入（如空链表）就会崩溃。接下来分析如何解决这两个问题。

Java

```
/*
public class ListNode {
    int val;
    ListNode next = null;

    ListNode(int val) {
        this.val = val;
    }
}*/
public class Solution {
    public ListNode Merge(ListNode
list1,ListNode list2) {
        if(list1 == null){
            return list2;
        }
        if(list2 == null){
            return list1;
        }
        if(list1.val <= list2.val){
```

```
            list1.next = Merge(list1.next,
list2);
            return list1;
        }else{
            list2.next = Merge(list1,
list2.next);
            return list2;
        }
    }
}
```

C++

```
/*
struct ListNode {
    int val;
    struct ListNode *next;
    ListNode(int x) :
            val(x), next(NULL) {
    }
};*/
class Solution {
public:
    ListNode* Merge(ListNode* pHead1, ListNode*
pHead2)
    {
        ListNode* node=NULL;
```

```
        if(pHead1==NULL){return node=pHead2;}
        if(pHead2==NULL){return node=pHead1;}
        if(pHead1->val>pHead2->val){
            node=pHead2;
            node->next=Merge(pHead1,pHead2->next);
        }else
            {
            node=pHead1;
            node->next=Merge(pHead1->next,pHead2);
        }
        return node;

    }
};
```