合并两个有序链表

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。

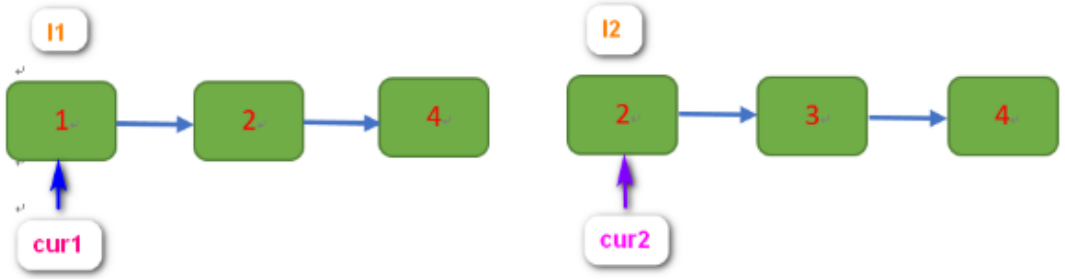
示例：

输入：`1->2->4, 1->3->4`

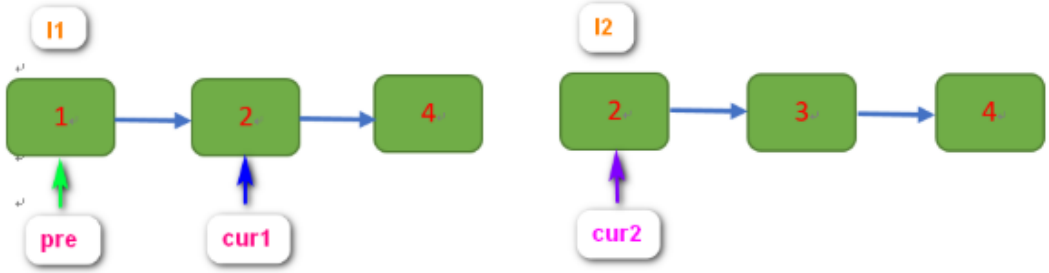
输出：`1->1->2->3->4->4`

比较两个链表头节点值的大小，值小的节点作为合并之后链表的头节点；之后哪个链表的头节点的值更小，另一链表的节点依次插入到这个链表中。
以链表 l1: 1->2->4->NULL 和链表 l2: 2->3->4->NULL 为例子，其合并过程如下图所示：

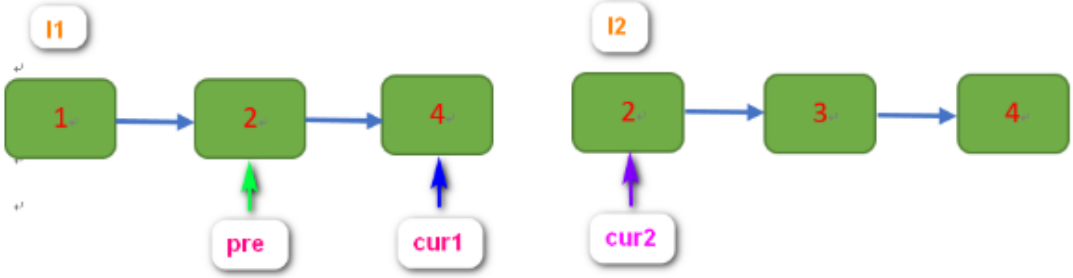
通过比较两链表头节点的值，获取 cur1 和 cur2，其中 cur1 用于记录两个链表中节点值较小的节点



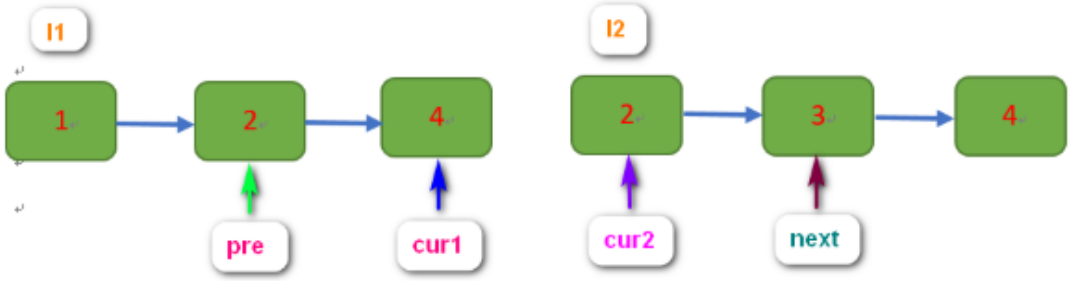
由于 cur1->val = 1 < cur2->val = 2，指针 cur1 右移，指针 pre 记录 cur1 右移之前 cur1 的位置

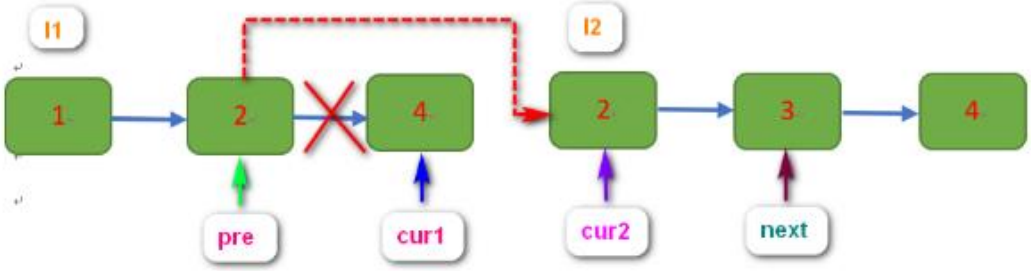


此时 cur1->val = 2 <= cur2->val = 2，指针 cur1 继续右移，pre 记录 cur1 右移之前 cur1 的位置

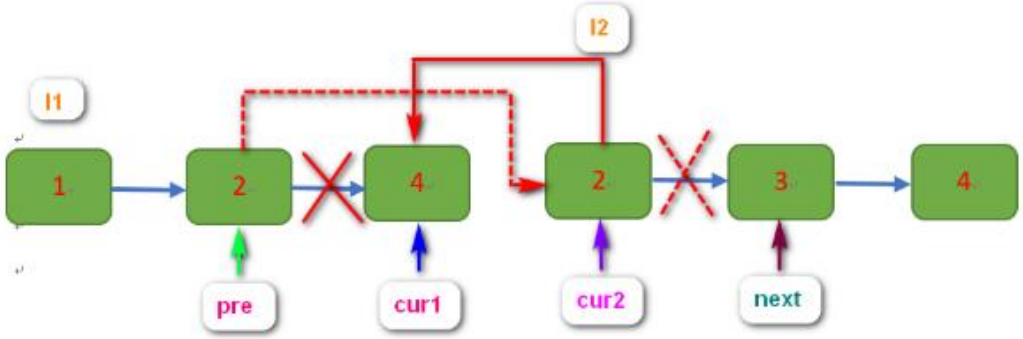


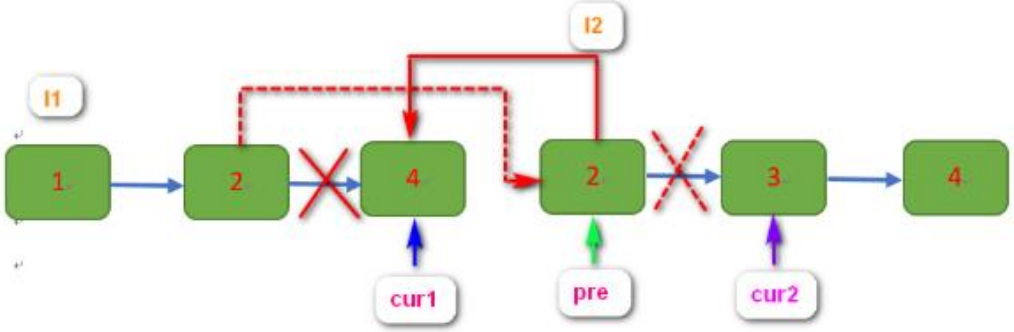
这时，cur1->val = 4 > cur2->val = 2，用一个指针 next 记录 cur2 的下一个节点，同时指针 pre 指向 cur2



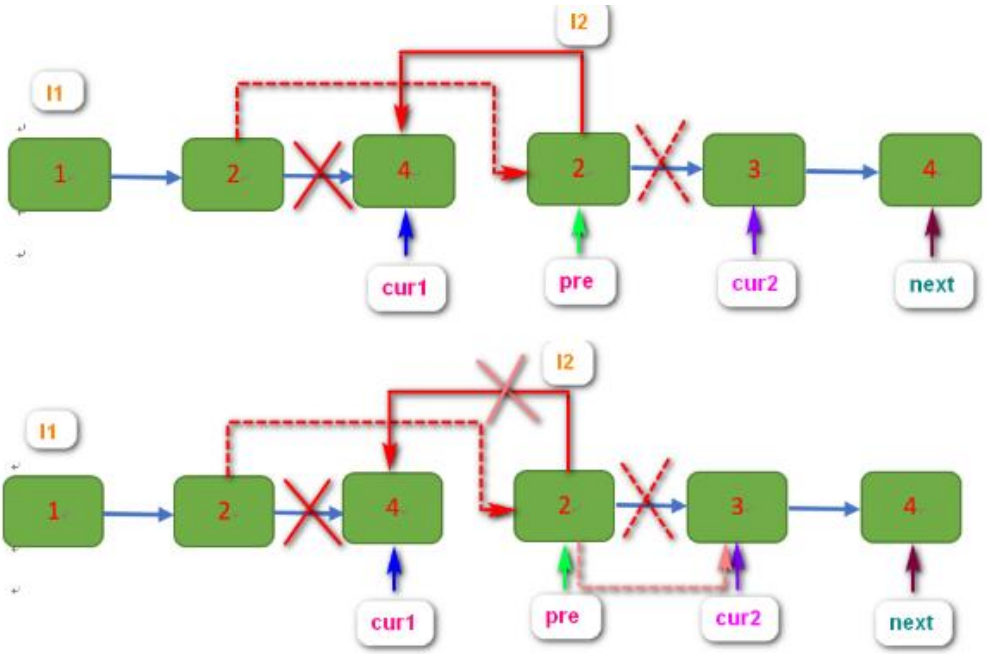


与此同时，指针 cur2 指向 cur1，pre 移动到 cur2，cur2 移动到 next

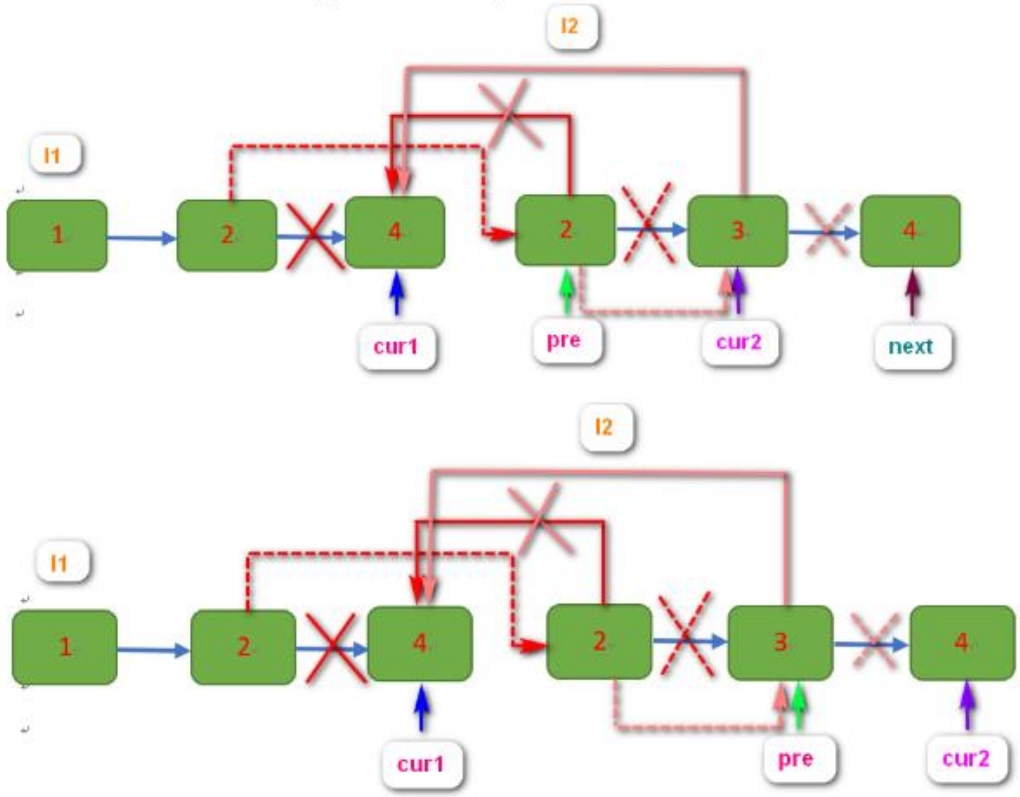




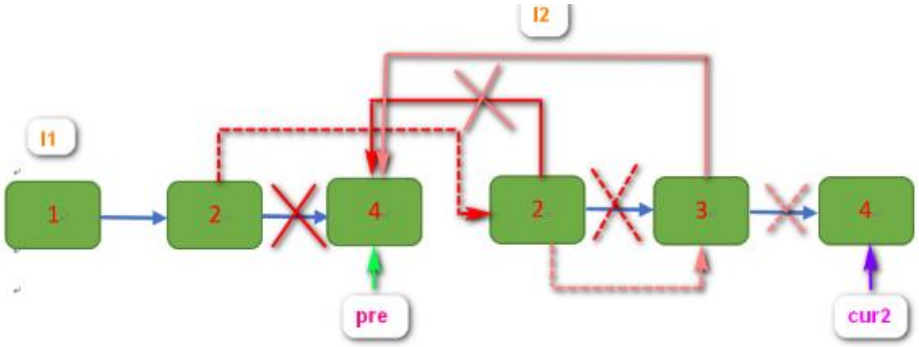
由于 cur1 还不为空，所以继续遍历。此时 cur1->val = 4 > cur2->val = 3，指针 next 记录 cur2 的下一个节点，同时指针 pre 指向 cur2



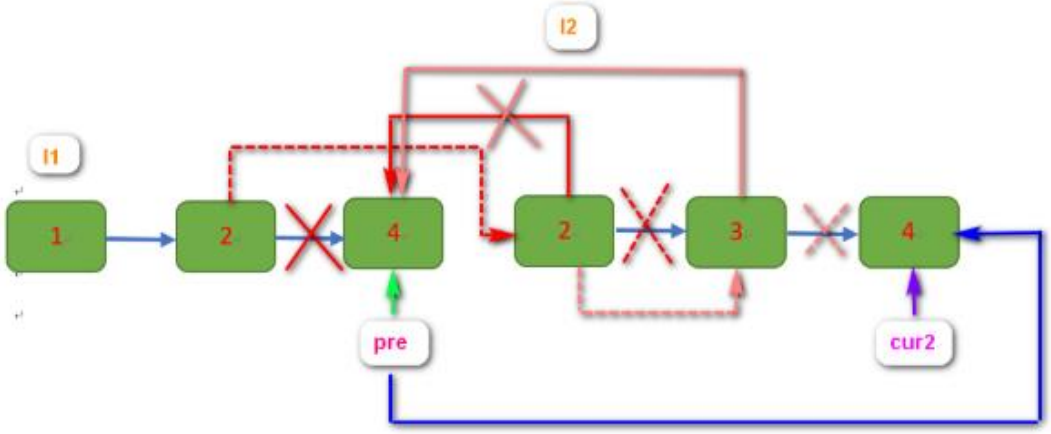
与此同时，指针 cur2 指向 cur1，pre 移动到 cur2，cur2 移动到 next



此时 cur1->val = 4 <= cur2->val = 4，指针 cur1 右移到空，指针 pre 记录 cur1 右移之前 cur1 的位置



至此，由于 cur1 为空，不再遍历，直接将指针 pre 指向 cur2 即可



合并之后的链表如下图示：


```
struct ListNode* mergeTwoLists(struct ListNode* l1, struct ListNode* l2){
    /* 特判 */
    if (l1 == NULL || l2 == NULL) {
        return l1 != NULL ? l1 : l2;
    }

    /* 获取合并之后新链表的头节点 */
    struct ListNode* mergeHead = l1->val > l2->val ? l2 : l1;
    /* 记录两个链表中节点值较小的节点，用于遍历链表 */
    struct ListNode* cur1 = mergeHead == l1 ? l1 : l2;
    struct ListNode* cur2 = mergeHead == l1 ? l2 : l1;
    /* 记录上次比较时节点值小的节点 */
    struct ListNode* pre = NULL;
    /* 记录当前节点值大的节点的下一节点，防止该链表断开之后无法找到后面的节
点 */
    struct ListNode* next = NULL;
    while (cur1 != NULL && cur2 != NULL) {
```

```
        if (cur1->val <= cur2->val) {
            pre = cur1;
            cur1 = cur1->next;
        } else {
            next = cur2->next;
            /* 将当前节点值较小的节点连接到另一链表中节点值大的节点 */
            pre->next = cur2;
            /* 将另一链表中节点值大的节点连接到当前节点值较小的节点的下一节点 */
            cur2->next = cur1;
            pre = cur2;
            cur2 = next;
        }
    }
    /* 合并后 l1 和 l2 最多只有一个还未被合并完，直接将链表末尾指向未合并完的链
表 */
    pre->next = cur1 == NULL ? cur2 : cur1;

    return mergeHead;
}
```