Merge Two Sorted List

Abbos Aliboev 2023041080

2025.01.16

Problem Definition (1)

- Source: Leetcode - 21
- Title: Merge Two Sorted List
- Difficulty: Easy

Recursion, Linked List, Data Structer

Problem Definition (2)

You are given the heads of two sorted linked lists ***list1*** and ***list2.***

Merge the two lists into one **sorted** list. The list should be made by splicing together the nodes of the first two lists.

Return the head of the merged linked list.

Problem Definition (2)

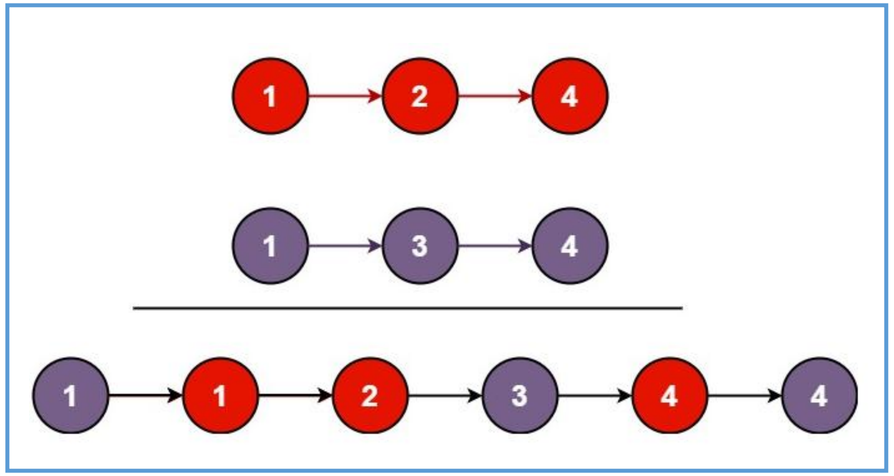
Example 1:
Input: list1 = [1,2,4], list2 = [1,3,4]
Output: [1,1,2,3,4,4]

Example 2:
Input: list1 = [], list2 = []
Output: []

Example 3:
Input: list1 = [], list2 = [0]
Output: [0]

Constraints:
- The number of nodes in both lists is in the range [0, 50].
- -100 <= Node.val <= 100
- Both list1 and list2 are sorted in **non-decreasing** order.

Example 1:



Solution (1-0)

1. Create a **dummy** node (**value = 0**) to simplify the merging process.

2. Initialize **current** pointer to point to the dummy node.

3. While both **list1** and **list2** are not NULL:

   - Compare **list1->val** and **list2->val**.

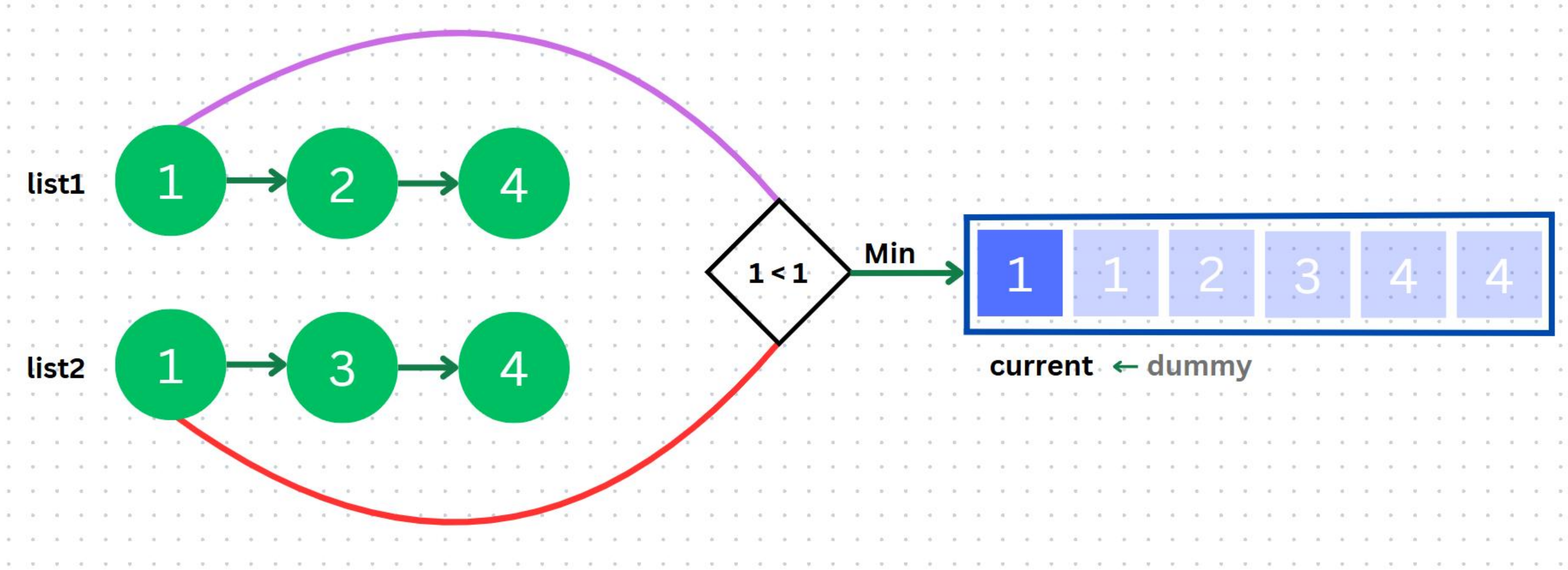
   - Add the smaller node to **current->next**.

   - Move the pointer of the list from which the node was taken.

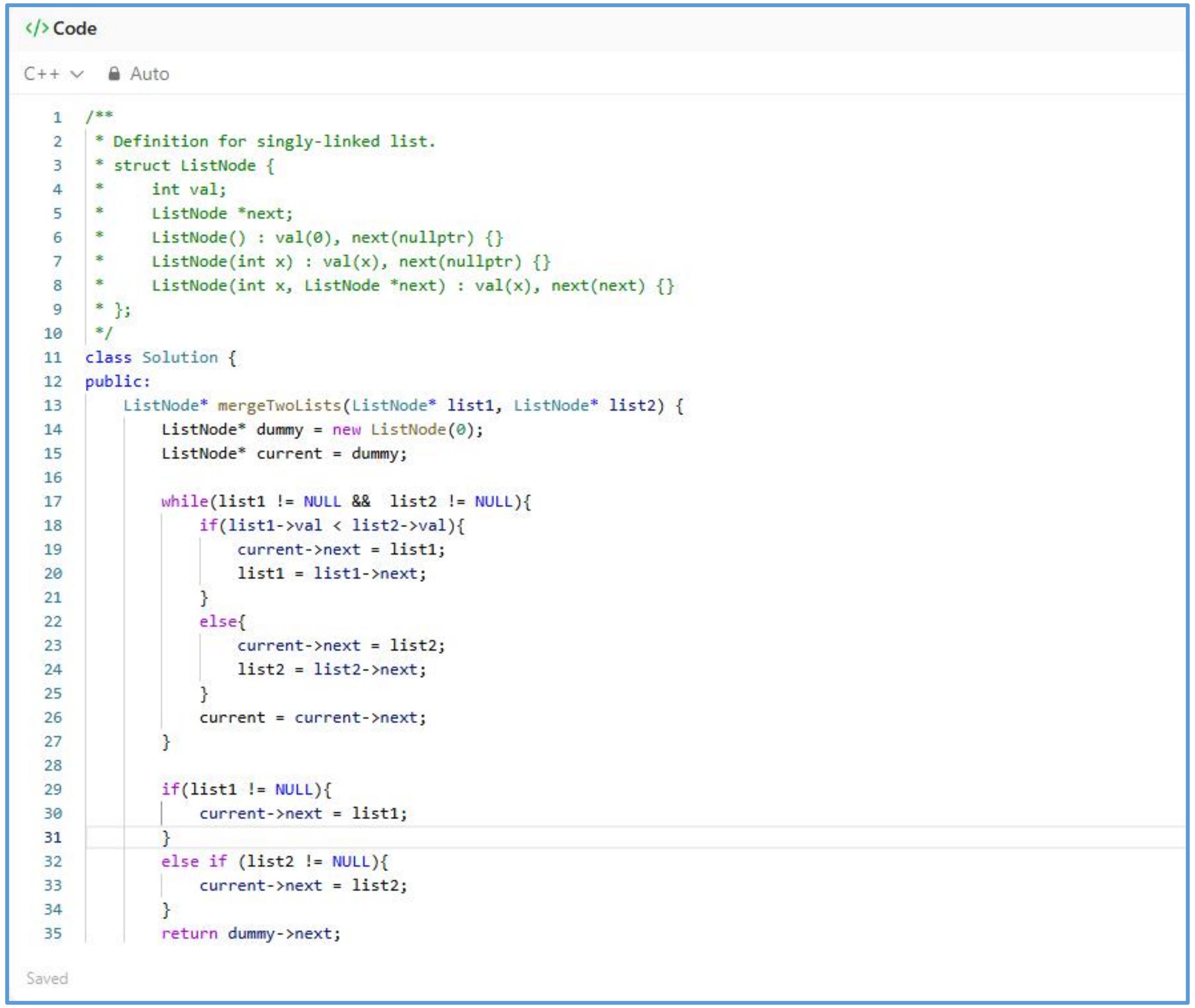
4. If either **list1** or **list2** still has remaining nodes, append them to **current->next**.

5. Return **dummy->next**, which is the head of the merged list.

Solution (1-1)



Solution (2-0)



Code

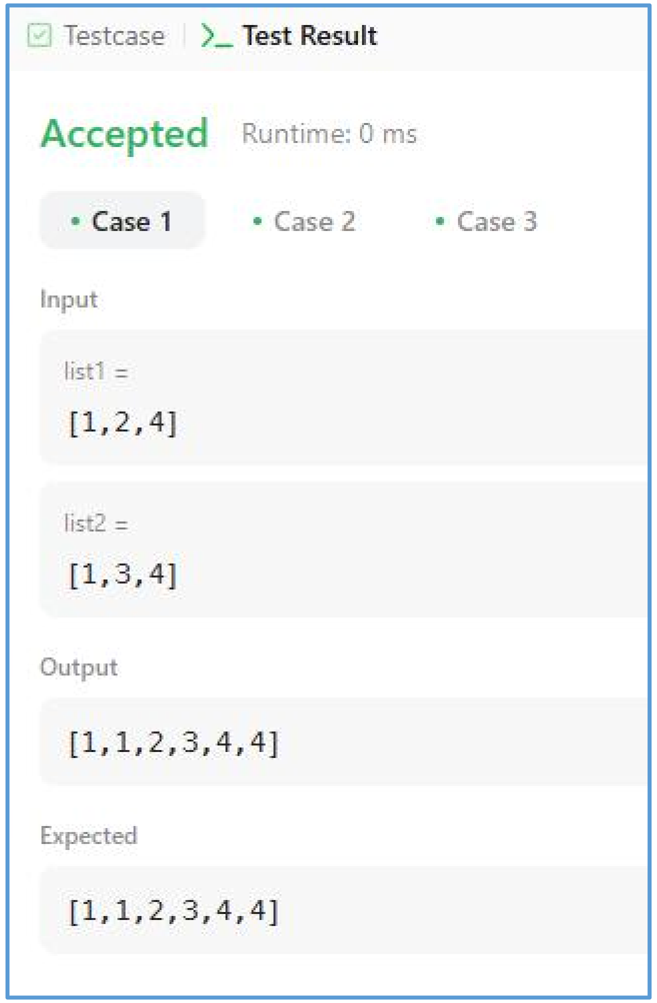
C++ Auto

```cpp
/**
 * 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* list1, ListNode* list2) {
        ListNode* dummy = new ListNode(0);
        ListNode* current = dummy;

        while(list1 != NULL &&  list2 != NULL){
            if(list1->val < list2->val){
                current->next = list1;
                list1 = list1->next;
            }
            else{
                current->next = list2;
                list2 = list2->next;
            }
            current = current->next;
        }

        if(list1 != NULL){
            current->next = list1;
        }
        else if (list2 != NULL){
            current->next = list2;
        }
        return dummy->next;
```

Saved



Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

list1 =

```
[1,2,4]
```

list2 =

```
[1,3,4]
```

Output

```
[1,1,2,3,4,4]
```

Expected

```
[1,1,2,3,4,4]
```

What you have learned

1. **Dummy** node simplifies merging and keeps track of the merged list's head.

2. **Merging** two lists involves comparing nodes, **adding** the smaller one, and **appending** remaining nodes.

3. Improved understanding of **pointer** manipulation and efficient linked list operations.

Questions and Answers

Greetings