21. Merge Two Sorted Lists

Tags	
Property	@August 23, 2022

Question

原文：

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

Merge the two lists in a 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*.

我的理解：

會拿到兩個以排序好的link list陣列(陣列可能為空)，排序好之後回傳排好的新陣列

翻譯：

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

Merge the two lists in a 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*.

自評翻譯正確性：

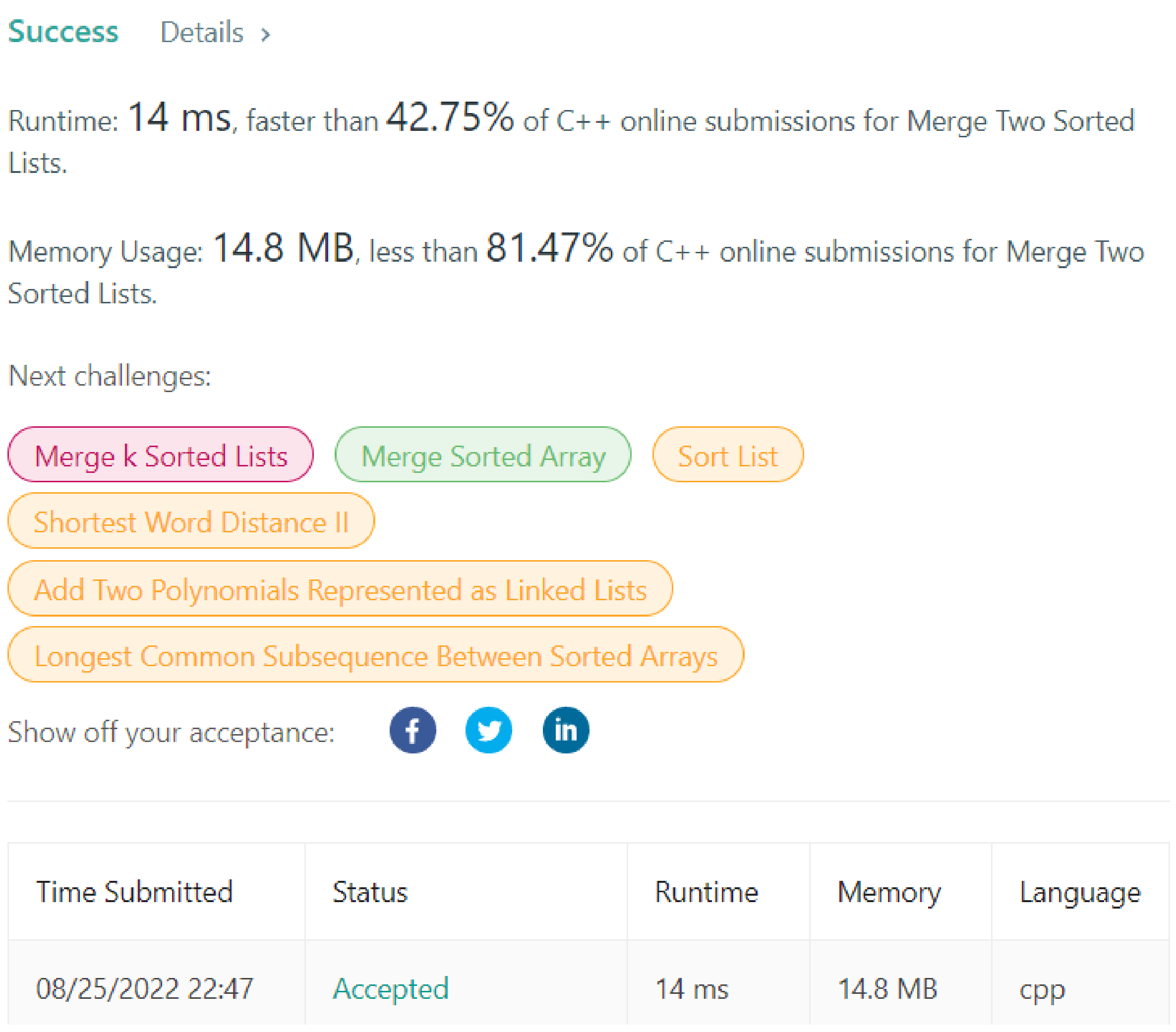
- Word Memory：
    - splicing拼接

Code

```
/**
 * 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) {
        //確認L1 L2有無NULL
        if(list1==NULL){
            return list2;
        }

        if(list2==NULL){
            return list1;
        }
        //開一個新的指標 head
        ListNode * head;
        //head 指向L1與L2中，第一個ListNode Val 數值較小的那個（照順序排的話 也會是合併後的第一個）
        if(list1->val > list2->val){
            head=list2;
            list2 = list2 -> next;
        }
        else{
            head=list1;
            list1 = list1 -> next;
        }
        //宣告一個臨時指標 curr 用來延長head之用
        ListNode * curr = head;




        while((list1!=NULL)&&(list2!=NULL)){//L1 L2沒有任何一方被排完就繼續while
            //哪邊小排哪邊
            if(list1->val < list2->val){
                curr->next=list1;//臨時指標指向下一個節點
                list1=list1->next;//節點被使用的 list 向下推移一個節點
            }

            else{
                curr->next=list2;
                list2=list2->next;
            }
            //臨時指標推移至剛剛指向的下一節點
            curr=curr->next;
        }

        //跳出迴圈就是有某方排序完了，另一方還沒排完的整串接在後面
        if(!list1){
            curr->next=list2;
        }
        else{
            curr->next=list1;
        }
        return head;
    }
};
```

> 思路：開一個新的指標指向兩個list第一個節點中較小的那個，作為合併後list的開頭，之後有點類似織衣服，哪邊小就把哪邊整串拿過來但只織一個節點進合併的list，一路比到有一條線（list）被用完，跳出迴圈，然後把另一條還有剩的線（list）通通皆在合併後的list後面



Success Details ›

Runtime: 14 ms, faster than 42.75% of C++ online submissions for Merge Two Sorted Lists.

Memory Usage: 14.8 MB, less than 81.47% of C++ online submissions for Merge Two Sorted Lists.

Next challenges:

Merge k Sorted Lists | Merge Sorted Array | Sort List | Shortest Word Distance II | Add Two Polynomials Represented as Linked Lists | Longest Common Subsequence Between Sorted Arrays

Show off your acceptance:

Time Submitted	Status	Runtime	Memory	Language
08/25/2022 22:47	Accepted	14 ms	14.8 MB	cpp

優良code參考

```
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
```

> 思路：基本照搬了這個人 所以思路同上

```
       // if list1 happen to be NULL
     // we will simply return list2.
         if(list1 == NULL)
             return list2;

     // if list2 happen to be NULL
     // we will simply return list1.
         if(list2 == NULL)
             return list1;

         ListNode * ptr = list1;
         if(list1 -> val > list2 -> val)
         {
             ptr = list2;
             list2 = list2 -> next;
         }
         else
         {
             list1 = list1 -> next;
         }
         ListNode *curr = ptr;

     // till one of the list doesn't reaches NULL
         while(list1 &&  list2)
         {
             if(list1 -> val < list2 -> val){
                 curr->next = list1;
                 list1 = list1 -> next;
             }
             else{
                 curr->next = list2;
                 list2 = list2 -> next;
             }
             curr = curr -> next;

         }

     // adding remaining elements of bigger list.
         if(!list1)
             curr -> next = list2;
         else
             curr -> next = list1;

         return ptr;

     }
};
```

> 思路：基本照搬了這個人 所以思路同上