

Description Accepted Editorial Solutions Submissions

← All Submissions

Accepted 134 / 134 testcases passed

Anirudh Singh submitted at Mar 06, 2025 22:45

Editorial Solution


Runtime

4 ms | Beats 48.98%

Analyze Complexity

Memory

18.51 MB | Beats 50.86%



75%  
50%  
25%  
0%

1ms 50ms 100ms 150ms

Code

C++ Auto

```
16 class Solution {
17 public:
18     ListNode* mergeKLists(vector<ListNode*>& lists) {
19         priority_queue<ListNode*, vector<ListNode*>, Compare> pq;
20
21         for (ListNode* list : lists) {
22             if (list) pq.push(list);
23         }
24
25         ListNode dummy; // Dummy node for ease of handling
26         ListNode* tail = &dummy;
27
28         while (!pq.empty()) {
29             ListNode* current = pq.top();
30             pq.pop();
31
32             tail->next = current;
33             tail = current;
34
35             if (current->next) {
36                 pq.push(current->next);
37             }
38         }
39
40         return dummy.next;
41     }
```

Saved

Ln 1, Col 1

Problem

Editorial

Submissions

Comments

Output Window

Compilation Results

Custom Input

Y.O.G.I. (AI Bot)

Problem Solved Successfully

Test Cases Passed

1112 / 1112

Attempts : Correct / Total

1 / 1

Accuracy : 100%

Points Scored

1 / 1

Your Total Score: 7

Time Taken

0.07

Solve Next

C++ (g++ 5.4)

Start Timer

```
1 // } Driver Code Ends
19
20 /*
21 struct Node {
22     int data;
23     struct Node* next;
24
25     Node(int x) {
26         data = x;
27         next = nullptr;
28     }
29 };
30 */
31 /*
32 Print elements of a linked list on console
33 Head pointer input could be NULL as well for empty list
34 */
35
36 class Solution {
37 public:
38     // Function to display the elements of a linked list in same line
39     void printList(Node *head) {
40         Node* temp = head;
41         while (temp != nullptr) {
42             cout << temp->data << " ";
43             temp = temp->next;
44             // your code goes here
45         }
46     }
47 };
48
49 // } Driver Code Ends
```

← All Submissions

**Accepted** 168 / 168 testcases passed

Anirudh Singh submitted at Mar 06, 2025 22:50

Editorial

Solution

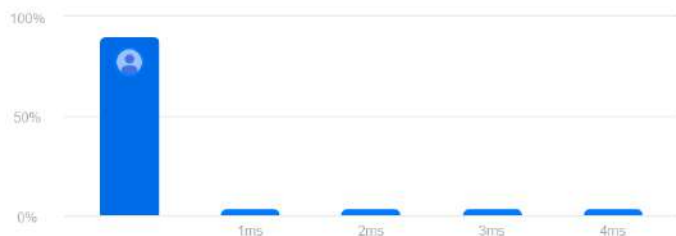
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

16.36 MB | Beats 11.13%



C++ v Auto

```
7  * ListNode(int x) : val(x), next(nullptr) {}
8  * ListNode(int x, ListNode *next) : val(x), next(next) {}
9  * };
10 */
11 class Solution {
12 public:
13     ListNode* deleteDuplicates(ListNode* head) {
14         ListNode* current = head;
15         while (current && current->next) {
16             if (current->val == current->next->val) {
17                 ListNode* temp = current->next;
18                 current->next = current->next->next;
19                 delete temp;
20             } else {
21                 current = current->next;
22             }
23         }
24         return head;
25     }
26 };
27
```

Saved

Ln 1, Col 1

← All Submissions

Accepted 28 / 28 testcases passed

Anirudh Singh submitted at Mar 06, 2025 22:52

Editorial

Solution

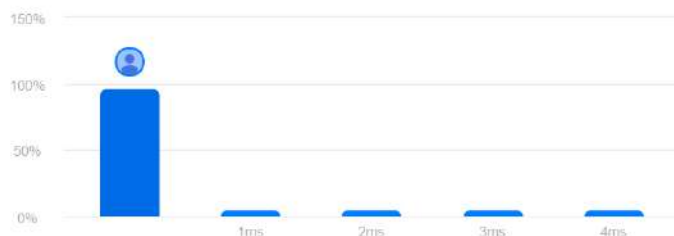
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

13.45 MB | Beats 39.75%

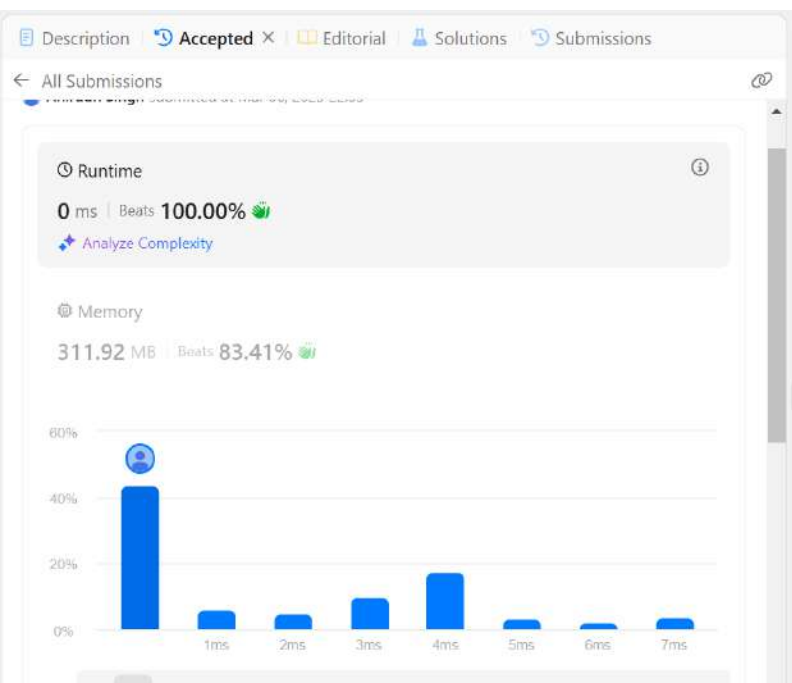


C++ Auto

```
5     ListNode* next;  
6     *  
7     *  
8     *  
9     *  
10    */  
11    class Solution {  
12    public:  
13        ListNode* reverseList(ListNode* head) {  
14            ListNode* prev = nullptr;  
15            ListNode* current = head;  
16            while (current) {  
17                ListNode* nextNode = current->next;  
18                current->next = prev;  
19                prev = current;  
20                current = nextNode;  
21            }  
22            return prev;  
23        }  
24    }  
25    };
```

Saved

Ln 22, Col 17



Solve today's Daily Challenge to refresh your streak!

C++ Auto

```
9  *;;
10 */
11 class Solution {
12 public:
13     ListNode* deleteMiddle(ListNode* head) {
14         if (!head || !head->next) return nullptr; // If only one node,
return nullptr
15
16         ListNode* slow = head, *fast = head, *prev = nullptr;
17         while (fast && fast->next) {
18             prev = slow;
19             slow = slow->next;
20             fast = fast->next->next;
21         }
22
23         prev->next = slow->next;
24         delete slow;
25         return head;
26     }
27 }
28 ;;
```

← Description Accepted × Editorial Solutions Submissions

← All Submissions

Accepted 208 / 208 testcases passed

Anirudh Singh submitted at Mar 06, 2025 22:56

Editorial

Solution

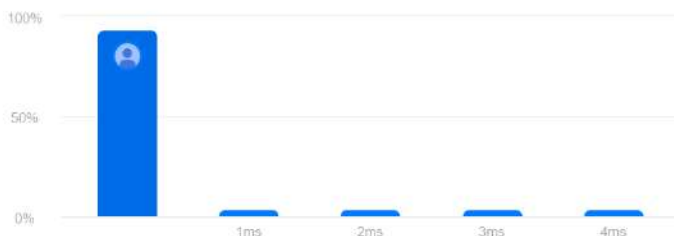
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

19.50 MB | Beats 62.23%



</> Code

C++ Auto

```
public:
    ListNode* mergeTwoLists(ListNode* l1, ListNode* l2) {
        if (!l1) return l2;
        if (!l2) return l1;

        ListNode* dummy = new ListNode(-1); // Dummy node
        ListNode* current = dummy;

        while (l1 && l2) {
            if (l1->val <= l2->val) {
                current->next = l1;
                l1 = l1->next;
            } else {
                current->next = l2;
                l2 = l2->next;
            }
            current = current->next;
        }

        // Attach the remaining part of the list
        current->next = l1 ? l1 : l2;

        return dummy->next; // The merged list starts after the dummy node
    }
};
```

Saved Upgrade to Cloud Saving

Ln 27, Col 1

All Submissions

Accepted 29 / 29 testcases passed

Anirudh Singh submitted at Mar 06, 2025 22:57

Editorial

Solution

Runtime

10 ms Beats 43.64%

Analyze Complexity

Memory

11.78 MB Beats 79.94%



Code

C++

Auto

```
15 // slow = head, fast = head,
16
17 while (fast && fast->next) {
18     slow = slow->next; // Move slow pointer one step
19     fast = fast->next->next; // Move fast pointer two steps
20     if (slow == fast) return true; // Cycle detected
21 }
22
23 return false; // No cycle found
24
```

Saved

Ln 23, Col 40

Testcase Test Result

Accepted

Runtime: 0 ms

Case 1

Case 2

Case 3

Input

head =  
[3,2,0,-4]

pos =

← All Submissions

Accepted 232 / 232 testcases passed

Anirudh Singh submitted at Mar 06, 2025 22:58

Editorial

Solution


Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

16.44 MB | Beats 31.91%



C++

Auto

```
9         while (tail->next) {
10             tail = tail->next;
11             length++;
12         }
13
14         // Step 2: Make the list circular
15         tail->next = head;
16
17         // Step 3: Find the new head (break the loop at the new tail)
18         k = k % length; // Optimize k to be within range
19         int stepsToNewHead = length - k;
20         ListNode* newTail = head;
21
22         for (int i = 1; i < stepsToNewHead; i++) {
23             newTail = newTail->next;
24         }
25
26         ListNode* newHead = newTail->next;
27         newTail->next = nullptr; // Break the loop
28
29         return newHead;
30     }
31 }
32
```

Saved

Ln 32, Col 1



Description Accepted Editorial Solutions Submissions

All Submissions

Accepted 30 / 30 testcases passed

Anirudh Singh submitted at Mar 06, 2025 22:59

Editorial Solution


Runtime

61 ms Beats 9.32%

Analyze Complexity

Memory

75.71 MB Beats 21.72%



Code

C++ Auto

```
1 class Solution {
2 public:
3     ListNode* merge(ListNode* l1, ListNode* l2) {
4         if (!l1) return l2;
5         if (!l2) return l1;
6
7         ListNode* dummy = new ListNode(-1);
8         ListNode* current = dummy;
9
10        while (l1 && l2) {
11            if (l1->val <= l2->val) {
12                current->next = l1;
13                l1 = l1->next;
14            } else {
15                current->next = l2;
16                l2 = l2->next;
17            }
18            current = current->next;
19        }
20        current->next = l1 ? l1 : l2;
21        return dummy->next;
22    }
23
24    ListNode* findMiddle(ListNode* head) {
25        ListNode* slow = head;
```

Saved

Ln 52, Col 1