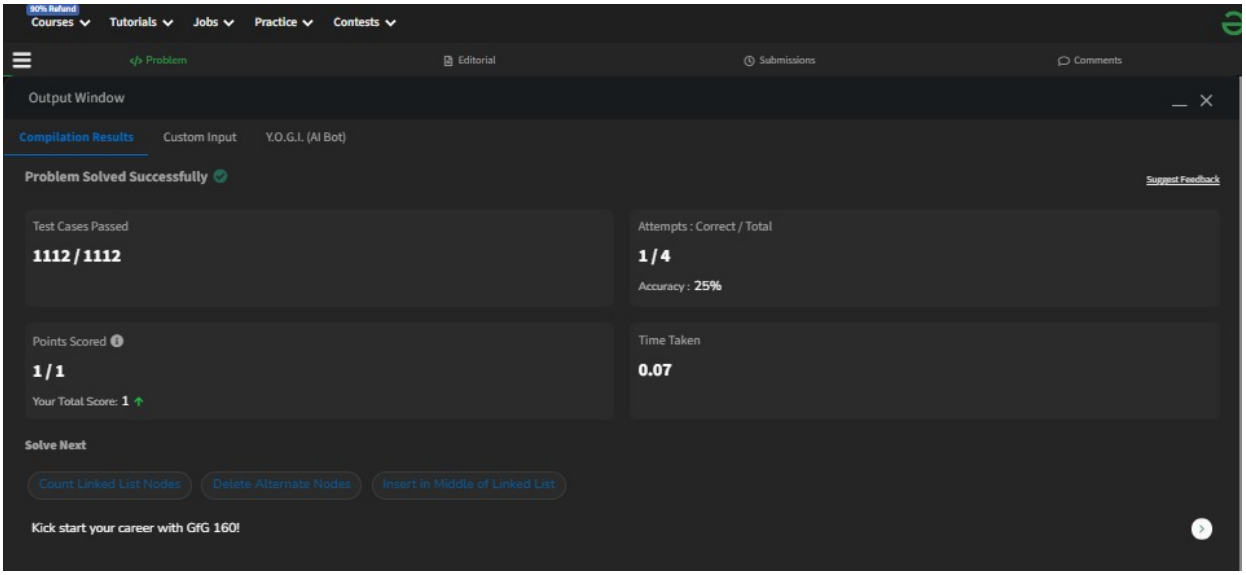
Experiment 3

Student Name: Saurav Ashiwal **UID: 22BCS13250**
Branch: CSE **Section/Group: 614/B**
Semester: 6 **Date of Performance: 14/02/2025**
Subject Name: AP lab-2 **Subject Code: 22CSP-351**

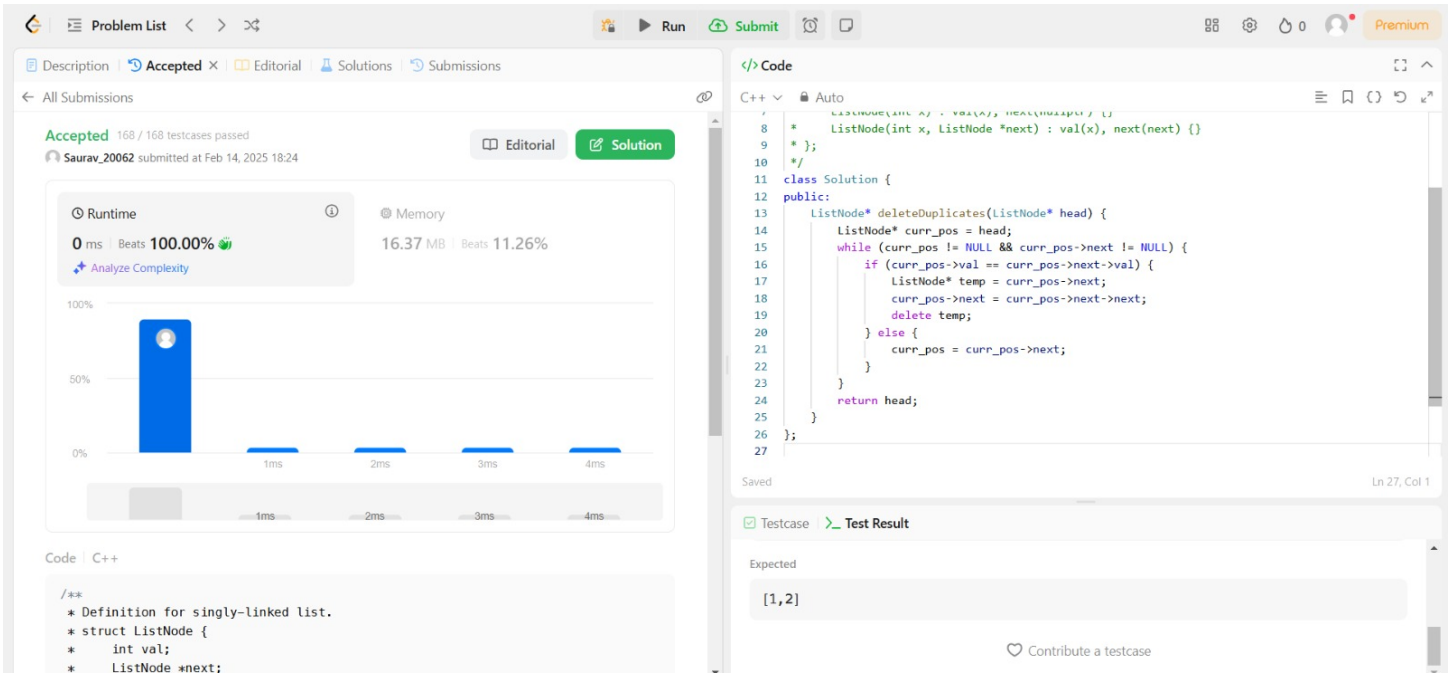
Q 1.

C++ (g++ 5.4) Start Timer

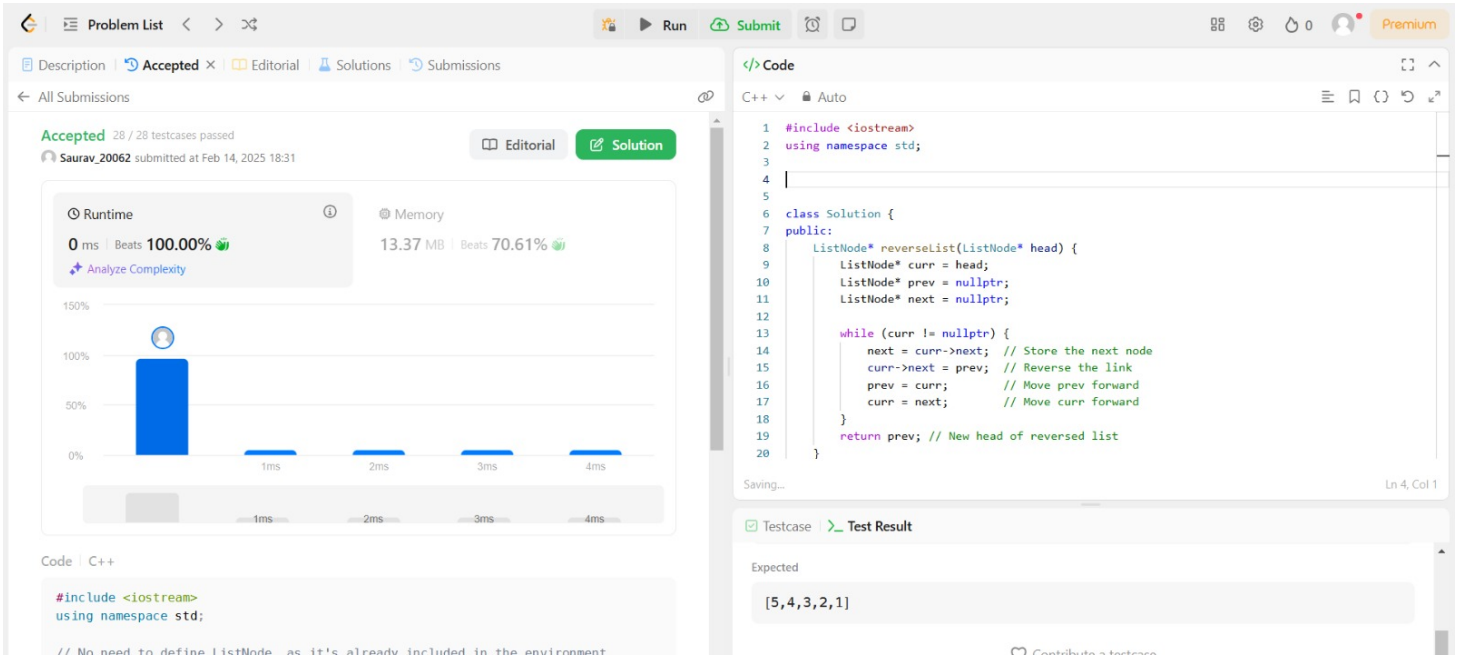
```
// } Driver Code Ends
/*
struct Node {
    int data;
    struct Node* next;

    Node(int x) {
        data = x;
        next = nullptr;
    }
};
*/
/*
    Print elements of a linked list on console
    Head pointer input could be NULL as well for empty list
*/

class Solution {
  public:
    // Function to display the elements of a linked list in same line
    void printList(Node *head) {
        Node*temp=head;
        while(temp!=NULL)
        {
            cout<< temp->data<<" ";
            temp=temp->next;
        }

    }
};
// } Driver Code Ends
```



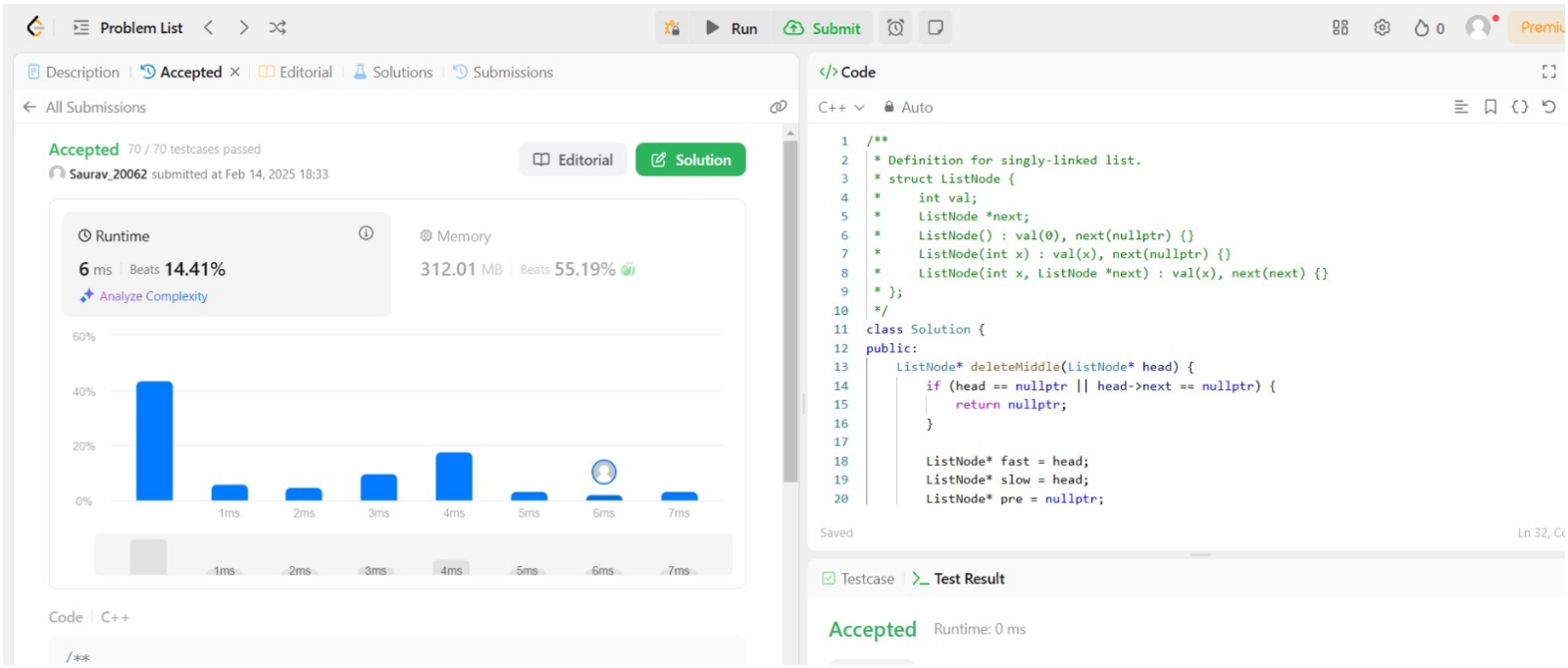
Q 2.


Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 168 / 168 testcases passed
Saurav_20062 submitted at Feb 14, 2025 18:24
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
16.37 MB | Beats 11.26%
Code
C++
Auto
class Solution {
public:
ListNode* deleteDuplicates(ListNode* head) {
ListNode* curr_pos = head;
while (curr_pos != NULL && curr_pos->next != NULL) {
if (curr_pos->val == curr_pos->next->val) {
ListNode* temp = curr_pos->next;
curr_pos->next = curr_pos->next->next;
delete temp;
} else {
curr_pos = curr_pos->next;
}
}
return head;
}
};
Saved
Ln 27, Col 1
Testcase
Test Result
Expected
[1,2]
Contribute a testcase

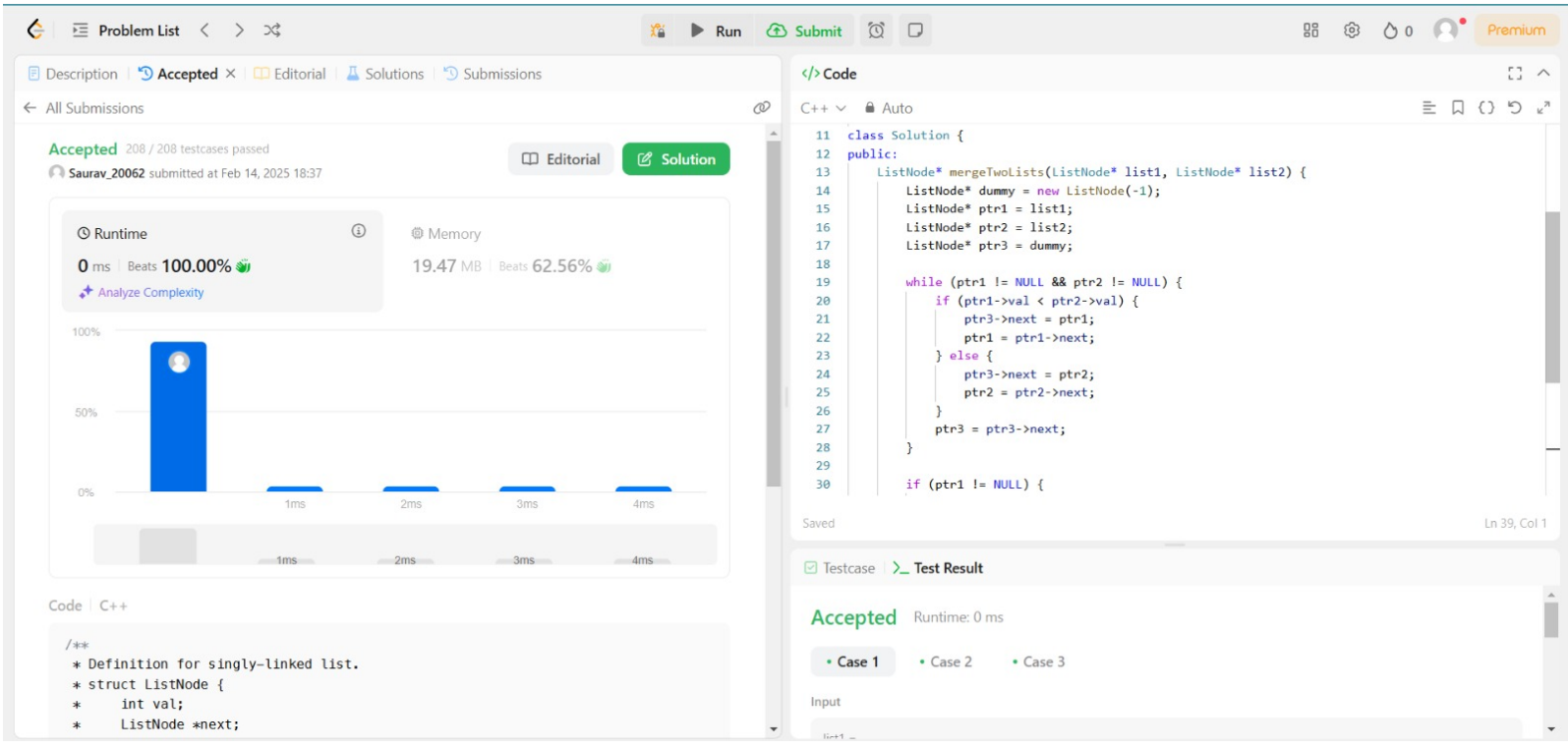
Q 3.


Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 28 / 28 testcases passed
Saurav_20062 submitted at Feb 14, 2025 18:31
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
13.37 MB | Beats 70.61%
Code
C++
Auto
#include <iostream>
using namespace std;
class Solution {
public:
ListNode* reverseList(ListNode* head) {
ListNode* curr = head;
ListNode* prev = nullptr;
ListNode* next = nullptr;
while (curr != nullptr) {
next = curr->next; // Store the next node
curr->next = prev; // Reverse the link
prev = curr; // Move prev forward
curr = next; // Move curr forward
}
return prev; // New head of reversed list
}
Saving...
Ln 4, Col 1
Testcase
Test Result
Expected
[5,4,3,2,1]

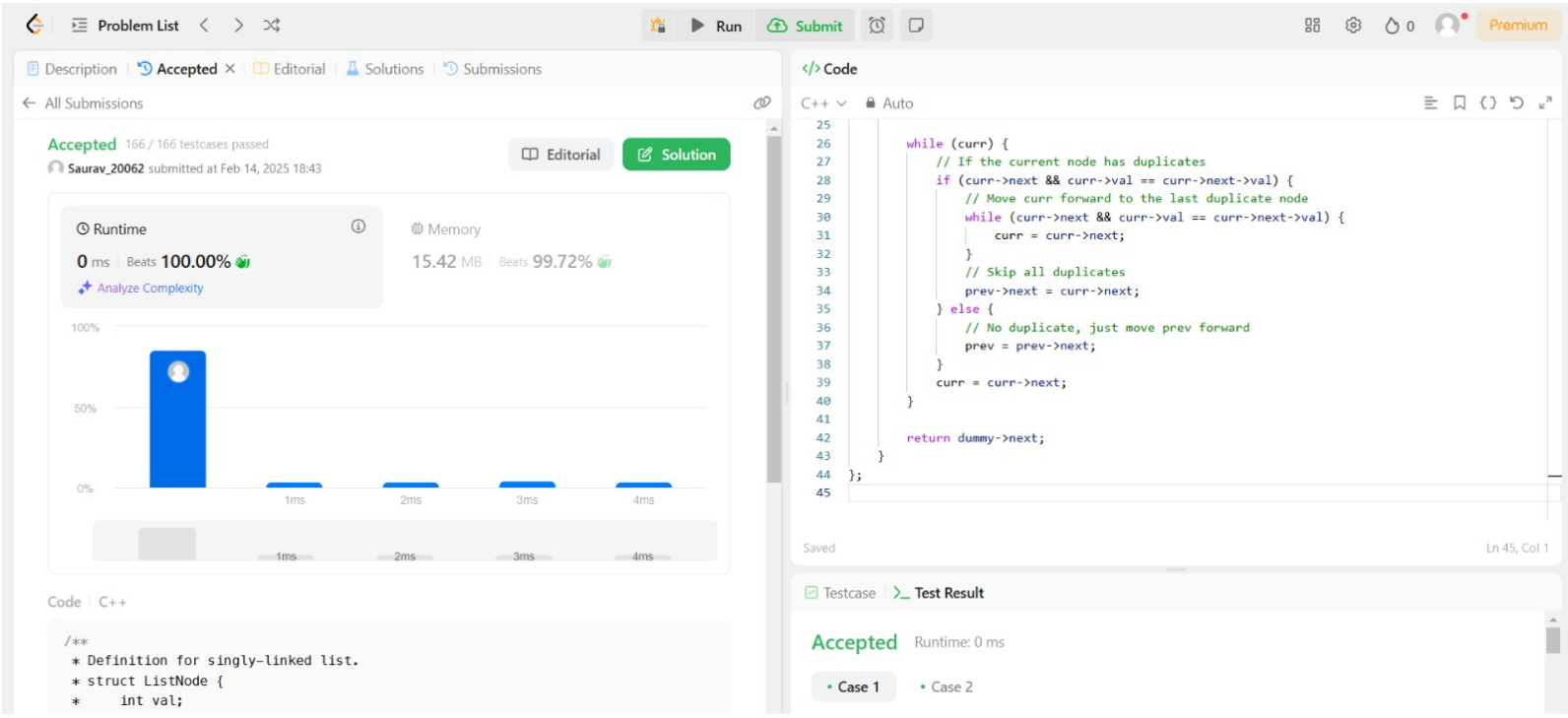
Q 4.


Problem List
Run
Submit
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 70 / 70 testcases passed
Saurav_20062 submitted at Feb 14, 2025 18:33
Editorial
Solution
Runtime
6 ms | Beats 14.41%
Analyze Complexity
Memory
312.01 MB | Beats 55.19%
Code
C++
Auto
/**
* 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* deleteMiddle(ListNode* head) {
if (head == nullptr || head->next == nullptr) {
return nullptr;
}
ListNode* fast = head;
ListNode* slow = head;
ListNode* pre = nullptr;
Saved
Testcase
Test Result
Accepted Runtime: 0 ms

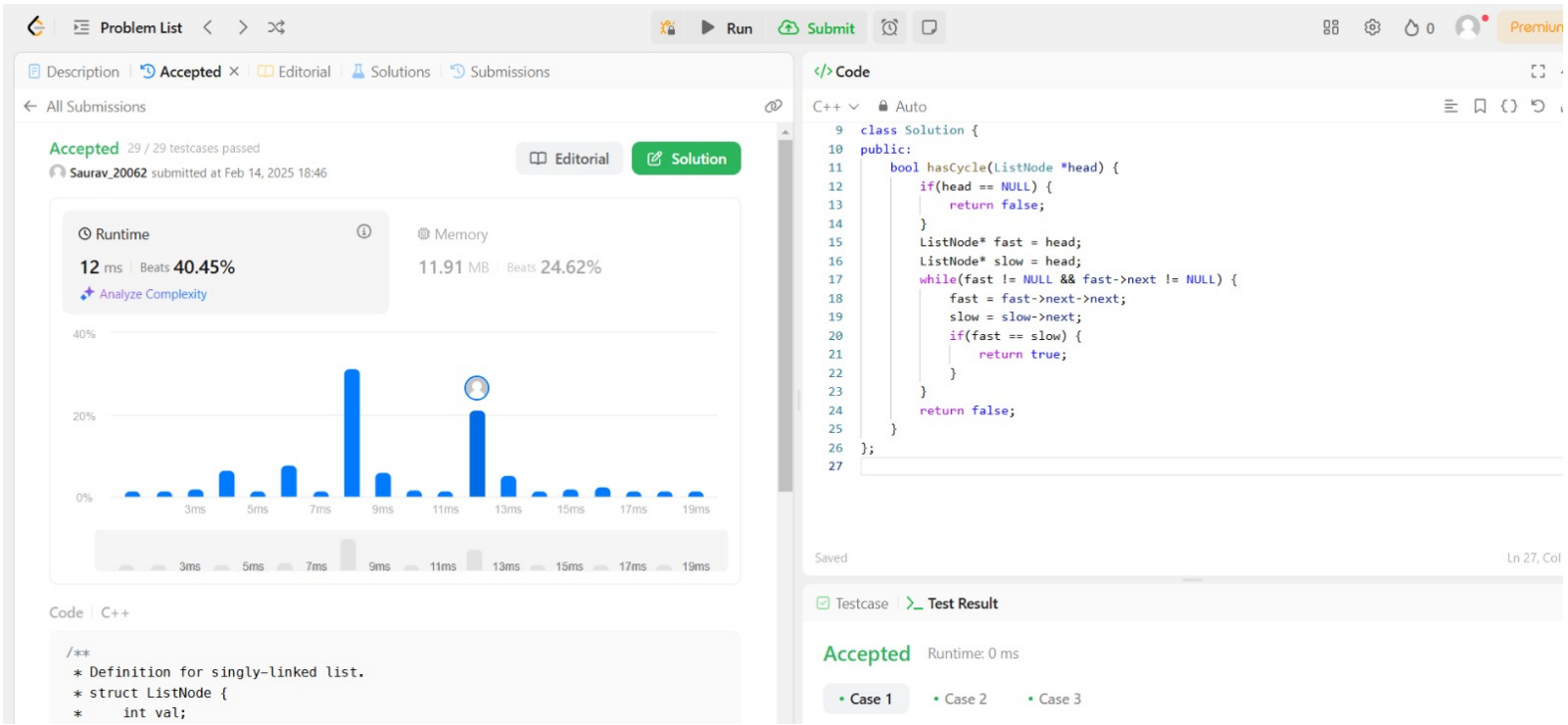
Q 5.


Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 208 / 208 testcases passed
Saurav_20062 submitted at Feb 14, 2025 18:37
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
19.47 MB | Beats 62.56%
Code
C++
Auto
class Solution {
public:
ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
ListNode* dummy = new ListNode(-1);
ListNode* ptr1 = list1;
ListNode* ptr2 = list2;
ListNode* ptr3 = dummy;
while (ptr1 != NULL && ptr2 != NULL) {
if (ptr1->val < ptr2->val) {
ptr3->next = ptr1;
ptr1 = ptr1->next;
} else {
ptr3->next = ptr2;
ptr2 = ptr2->next;
}
ptr3 = ptr3->next;
}
if (ptr1 != NULL) {
Saved
Testcase
Test Result
Accepted Runtime: 0 ms
Case 1
Case 2
Case 3
Input
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
* ListNode *next;

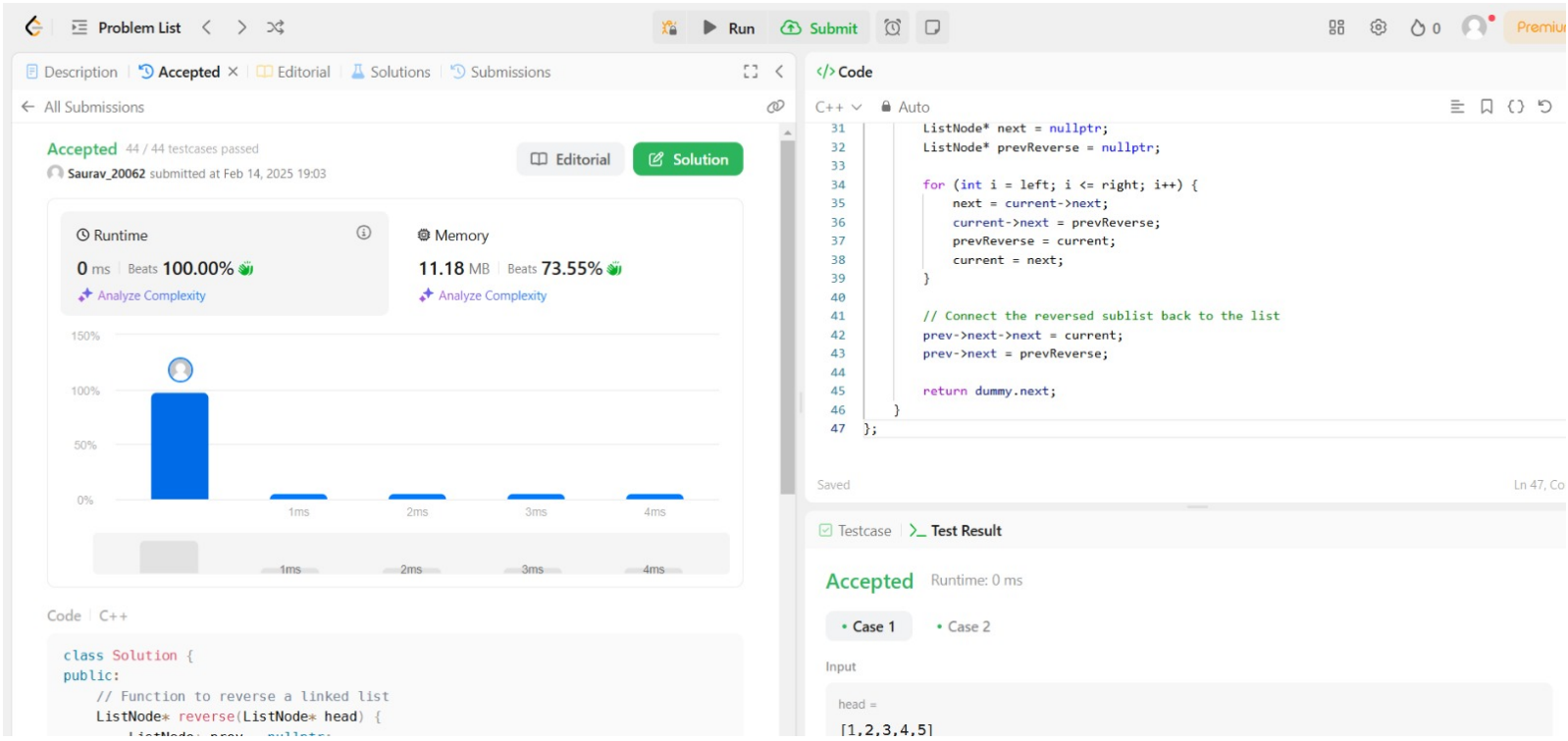
Q 6.


Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 166 / 166 testcases passed
Saurav_20062 submitted at Feb 14, 2025 18:43
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
15.42 MB | Beats 99.72%
Code
C++
Auto
while (curr) {
// If the current node has duplicates
if (curr->next && curr->val == curr->next->val) {
// Move curr forward to the last duplicate node
while (curr->next && curr->val == curr->next->val) {
curr = curr->next;
}
// Skip all duplicates
prev->next = curr->next;
} else {
// No duplicate, just move prev forward
prev = prev->next;
}
curr = curr->next;
}
return dummy->next;
}
};
Saved
Ln 45, Col 1
Testcase
Test Result
Accepted Runtime: 0 ms
Case 1
Case 2
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;

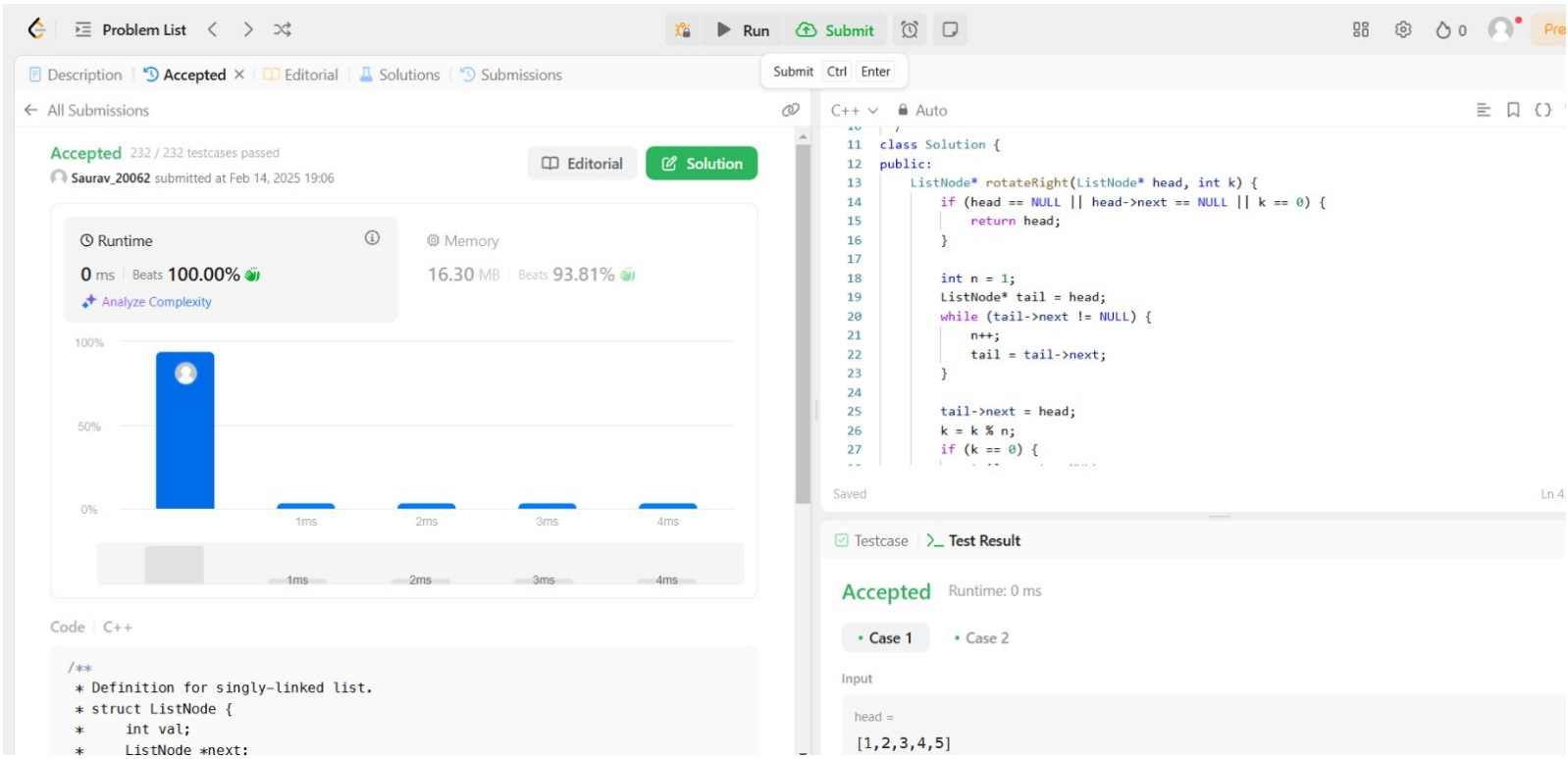
Q7.


Problem List
Run
Submit
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 29 / 29 testcases passed
Saurav_20062 submitted at Feb 14, 2025 18:46
Editorial
Solution
Runtime
12 ms | Beats 40.45%
Analyze Complexity
Memory
11.91 MB | Beats 24.62%
Code
C++
Auto
class Solution {
public:
bool hasCycle(ListNode *head) {
if(head == NULL) {
return false;
}
ListNode* fast = head;
ListNode* slow = head;
while(fast != NULL && fast->next != NULL) {
fast = fast->next->next;
slow = slow->next;
if(fast == slow) {
return true;
}
}
return false;
}
};
Saved
Testcase
Test Result
Accepted Runtime: 0 ms
Case 1
Case 2
Case 3
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;

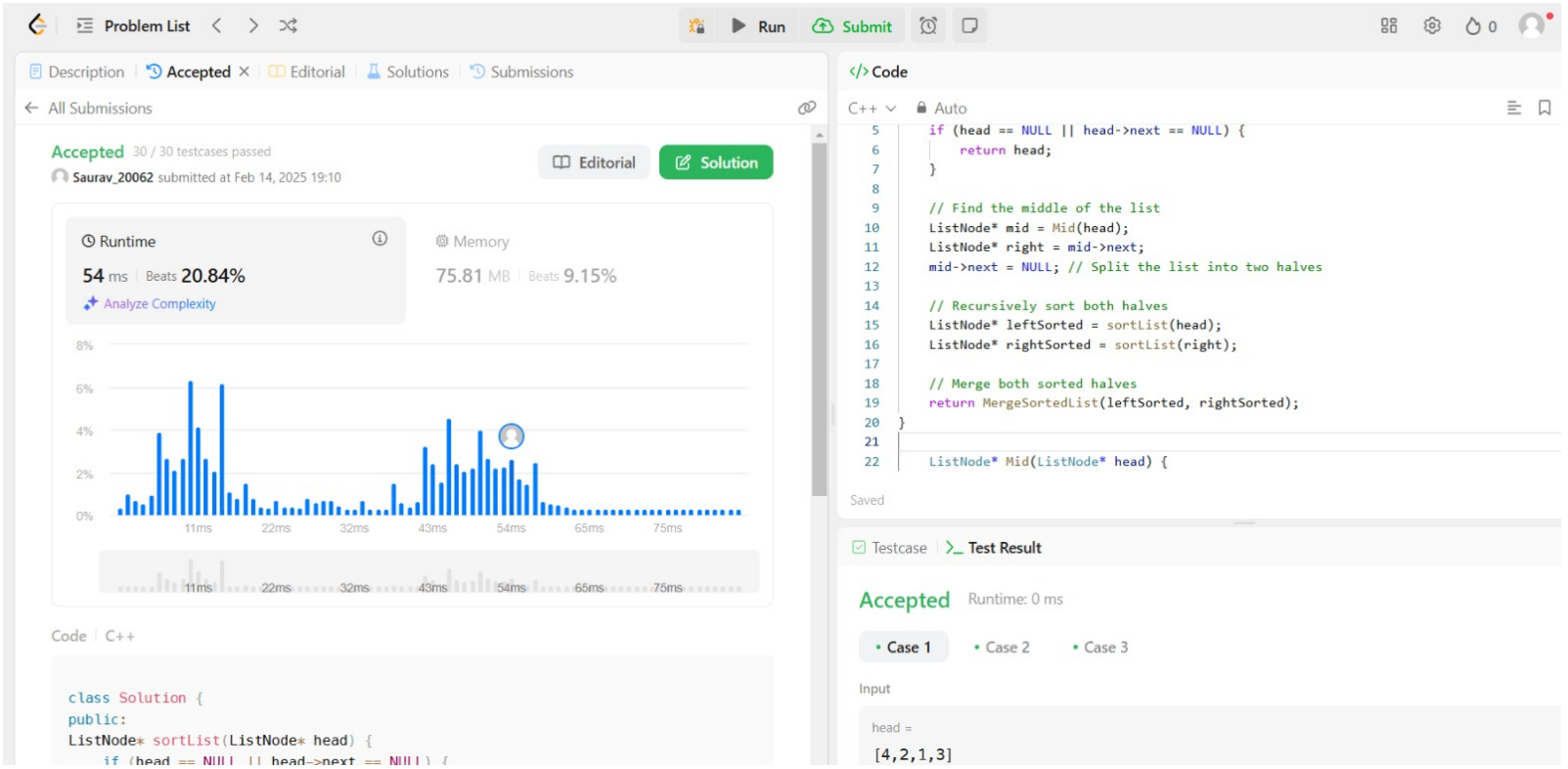
Q 8.


Problem List
Run
Submit
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 44 / 44 testcases passed
Saurav_20062 submitted at Feb 14, 2025 19:03
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
11.18 MB | Beats 73.55%
Analyze Complexity
150%
100%
50%
0%
1ms
2ms
3ms
4ms
Code | C++
class Solution {
public:
// Function to reverse a linked list
ListNode* reverse(ListNode* head) {
Code
C++
Auto
ListNode* next = nullptr;
ListNode* prevReverse = nullptr;
for (int i = left; i <= right; i++) {
next = current->next;
current->next = prevReverse;
prevReverse = current;
current = next;
}
// Connect the reversed sublist back to the list
prev->next->next = current;
prev->next = prevReverse;
return dummy.next;
}
};
Saved
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
head =
[1,2,3,4,5]

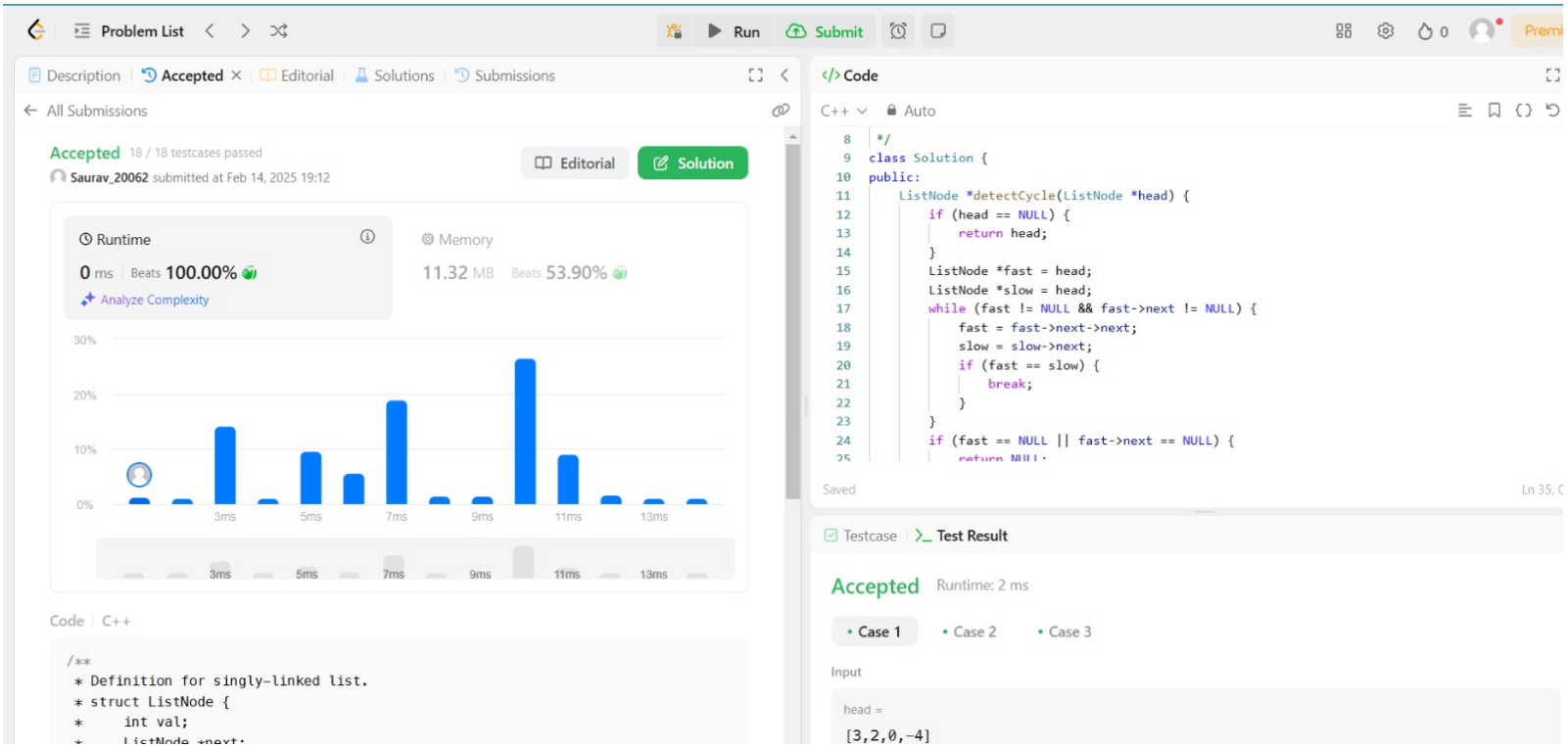
Q 9.


Problem List
Run
Submit
Description
Accepted
Editorial
Solutions
Submissions
Submit Ctrl Enter
All Submissions
Accepted 232 / 232 testcases passed
Saurav_20062 submitted at Feb 14, 2025 19:06
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
16.30 MB | Beats 93.81%
100%
50%
0%
1ms
2ms
3ms
4ms
Code | C++
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
* ListNode *next;
C++
Auto
class Solution {
public:
ListNode* rotateRight(ListNode* head, int k) {
if (head == NULL || head->next == NULL || k == 0) {
return head;
}
int n = 1;
ListNode* tail = head;
while (tail->next != NULL) {
n++;
tail = tail->next;
}
tail->next = head;
k = k % n;
if (k == 0) {
Saved
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
head =
[1,2,3,4,5]

Q 10.


Problem List
Run
Submit
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 30 / 30 testcases passed
Saurav_20062 submitted at Feb 14, 2025 19:10
Editorial
Solution
Runtime
54 ms | Beats 20.84%
Analyze Complexity
Memory
75.81 MB | Beats 9.15%
Code | C++
class Solution {
public:
ListNode* sortList(ListNode* head) {
if (head == NULL || head->next == NULL) {
return head;
}
// Find the middle of the list
ListNode* mid = Mid(head);
ListNode* right = mid->next;
mid->next = NULL; // Split the list into two halves
// Recursively sort both halves
ListNode* leftSorted = sortList(head);
ListNode* rightSorted = sortList(right);
// Merge both sorted halves
return MergeSortedList(leftSorted, rightSorted);
}
ListNode* Mid(ListNode* head) {
Saved
Testcase
Test Result
Accepted Runtime: 0 ms
Case 1
Case 2
Case 3
Input
head =
[4,2,1,3]

Q 11.


Problem List
Run
Submit
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 18 / 18 testcases passed
Saurav_20062 submitted at Feb 14, 2025 19:12
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
11.32 MB | Beats 53.90%
Code | C++
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
*/
class Solution {
public:
ListNode *detectCycle(ListNode *head) {
if (head == NULL) {
return head;
}
ListNode *fast = head;
ListNode *slow = head;
while (fast != NULL && fast->next != NULL) {
fast = fast->next->next;
slow = slow->next;
if (fast == slow) {
break;
}
}
if (fast == NULL || fast->next == NULL) {
Saved
Testcase
Test Result
Accepted Runtime: 2 ms
Case 1
Case 2
Case 3
Input
head =
[3,2,0,-4]