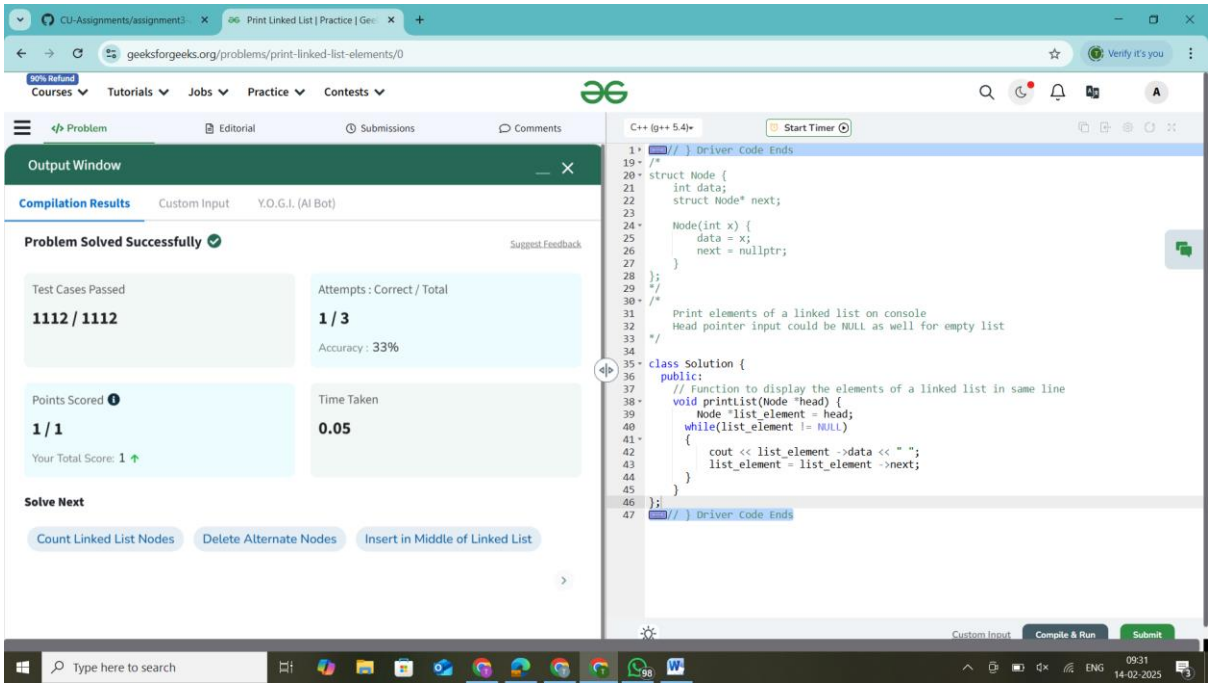
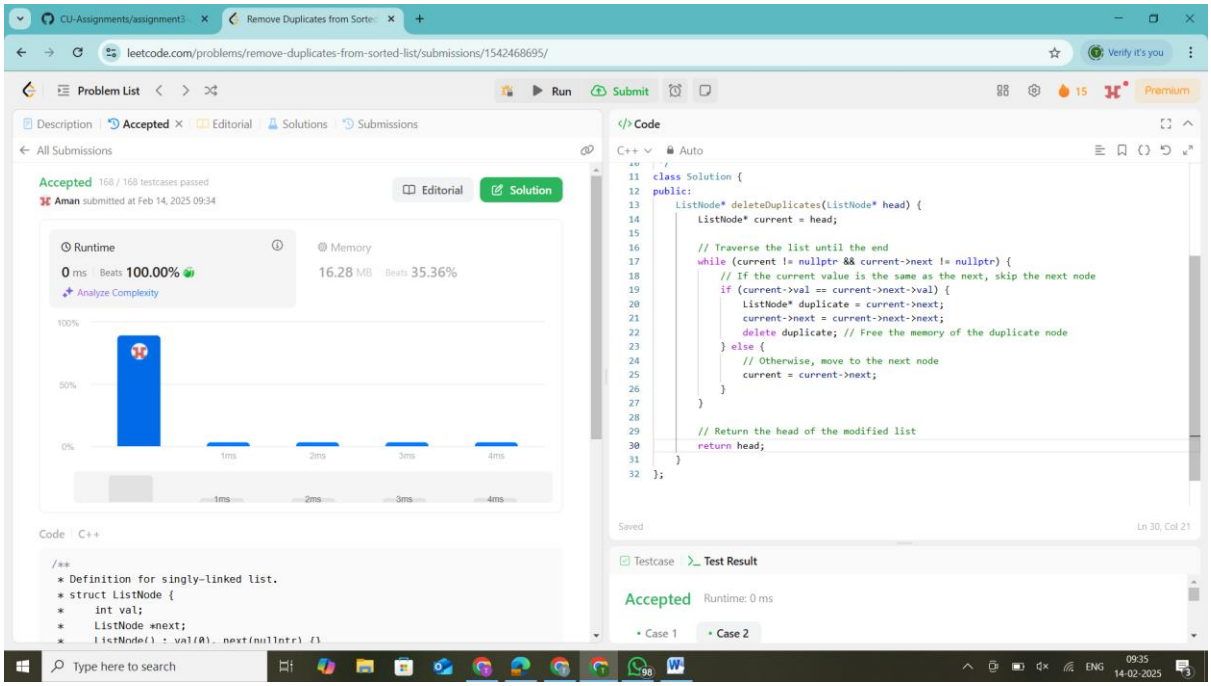
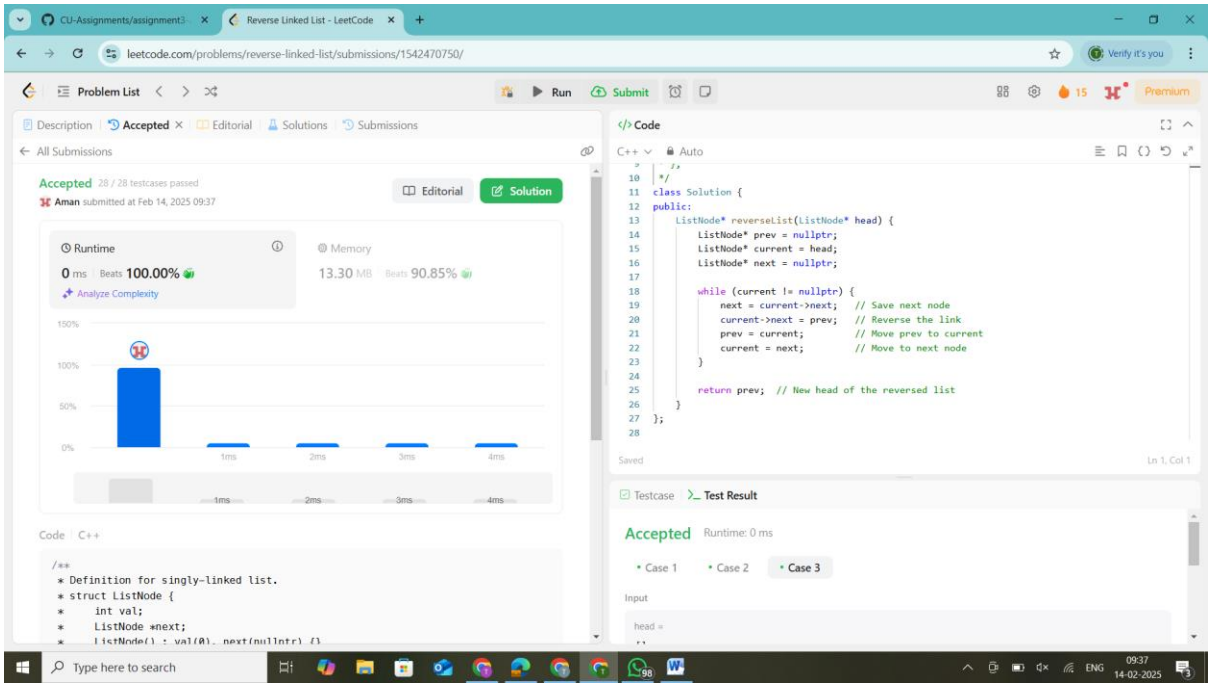
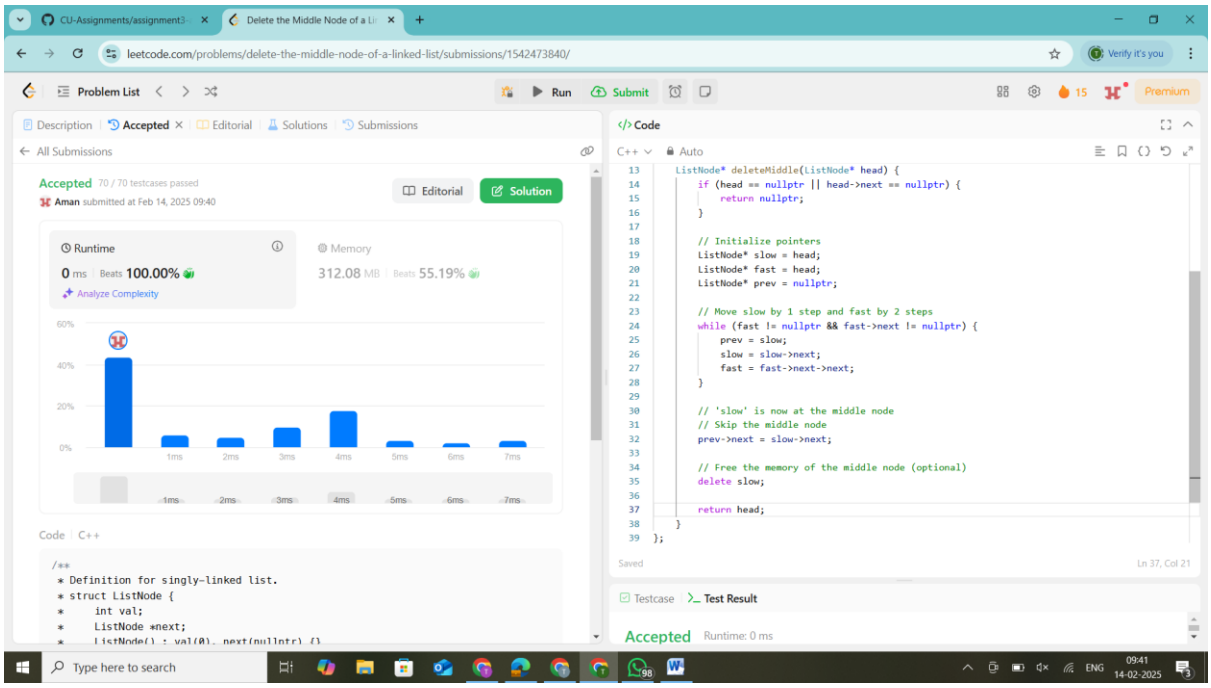
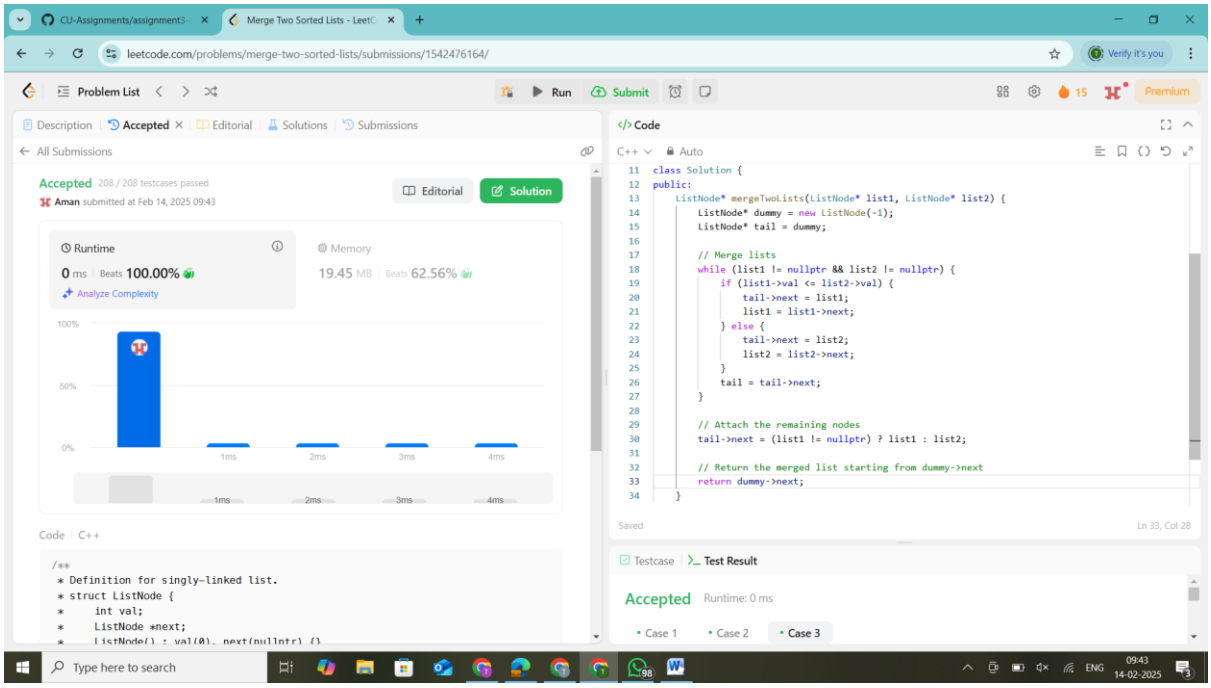
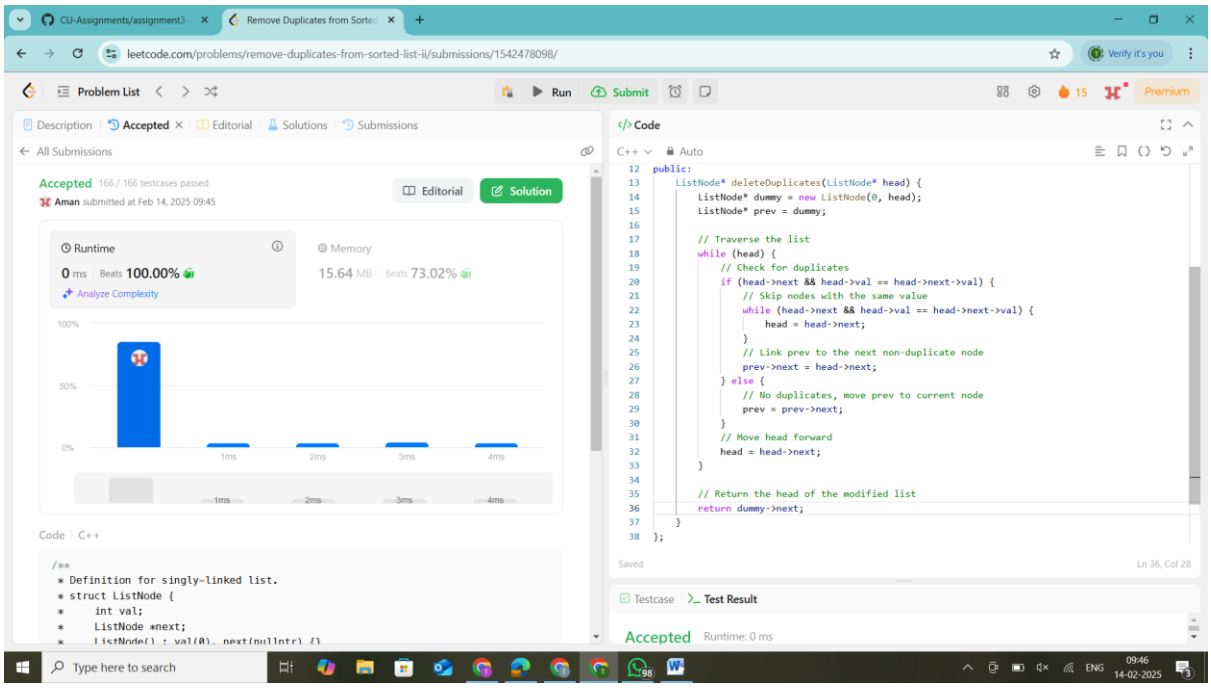

geeksforgeeks.org/problems/print-linked-list-elements/0
Problem
Editorial
Submissions
Comments
Output Window
Compilation Results
Custom Input
Y.O.G.I. (AI Bot)
Problem Solved Successfully
Suggest Feedback
Test Cases Passed
1112 / 1112
Attempts : Correct / Total
1 / 3
Accuracy : 33%
Points Scored
1 / 1
Your Total Score: 1
Time Taken
0.05
Solve Next
Count Linked List Nodes
Delete Alternate Nodes
Insert in Middle of Linked List
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 *list_element = head;
while(list_element != NULL)
{
cout << list_element ->data << " ";
list_element = list_element ->next;
}
}
};
// } Driver Code Ends
Custom Input
Compile & Run
Submit
Type here to search
ENG
09:31
14-02-2025

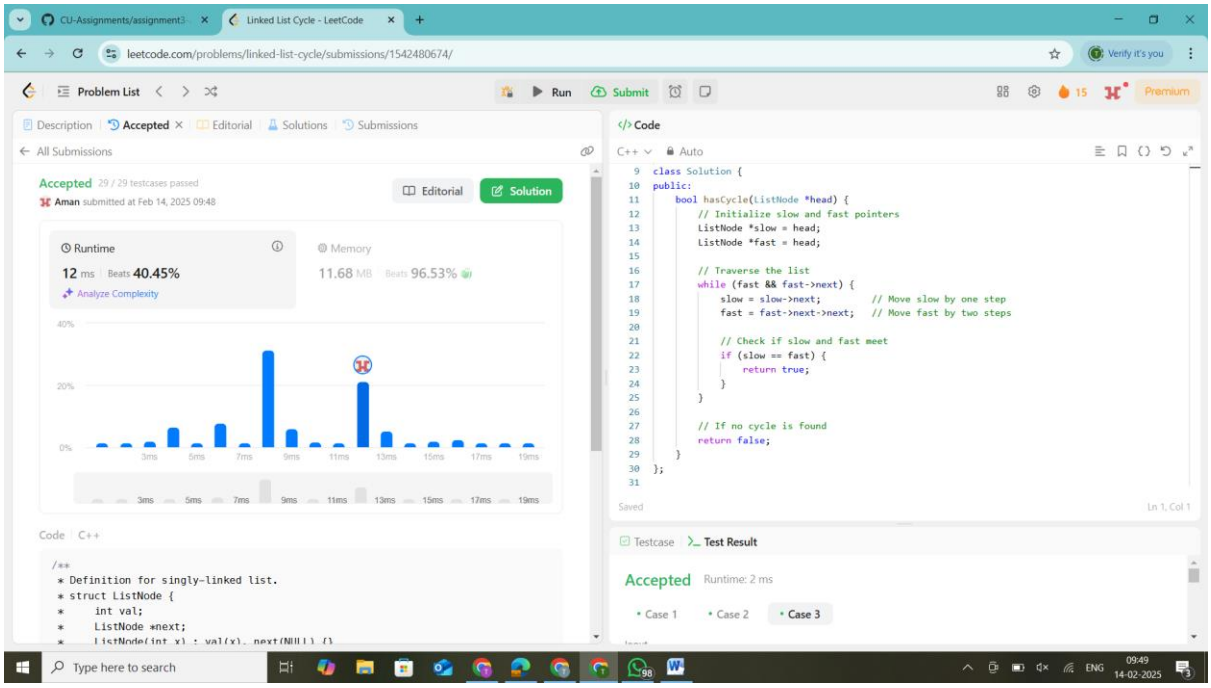

leetcode.com/problems/remove-duplicates-from-sorted-list/submissions/1542468695/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 168 / 168 testcases passed
Aman submitted at Feb 14, 2025 09:34
Editorial
Solution
Runtime
0 ms Beats 100.00%
Analyze Complexity
Memory
16.28 MB Beats 35.36%
Code
C++
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
* ListNode *next;
Code
C++
Auto
class Solution {
public:
ListNode* deleteDuplicates(ListNode* head) {
ListNode* current = head;
// Traverse the list until the end
while (current != nullptr && current->next != nullptr) {
// If the current value is the same as the next, skip the next node
if (current->val == current->next->val) {
ListNode* duplicate = current->next;
current->next = current->next->next;
delete duplicate; // Free the memory of the duplicate node
} else {
// Otherwise, move to the next node
current = current->next;
}
}
// Return the head of the modified list
return head;
}
};
Saved
Ln 30, Col 21
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Type here to search
ENG
09:35
14-02-2025

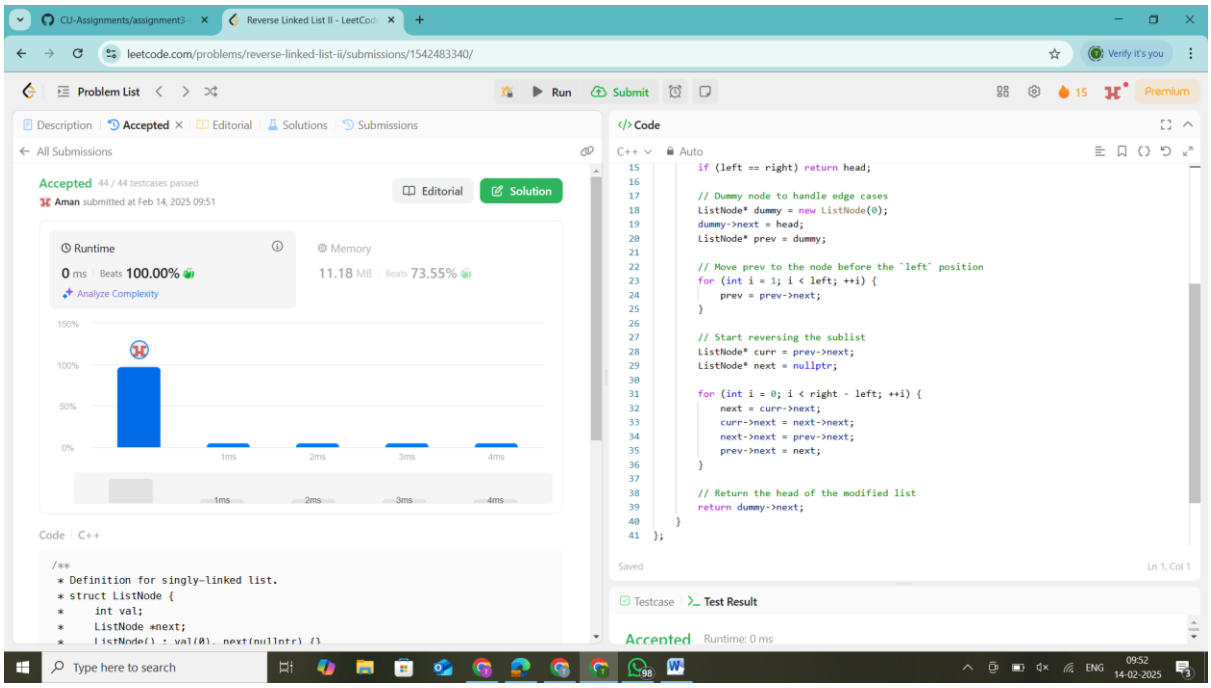

CU-Assignments/assignment3
Reverse Linked List - LeetCode
leetcode.com/problems/reverse-linked-list/submissions/1542470750/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 28 / 28 testcases passed
Aman submitted at Feb 14, 2025 09:37
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
13.30 MB | Beats 90.85%
Code | C++
/**
* Definition for singly-linked list.
* struct ListNode {
*   int val;
*   ListNode *next;
*   ListNode() : val(0), next(nullptr) {}
Code
C++
Auto
*/
class Solution {
public:
ListNode* reverseList(ListNode* head) {
ListNode* prev = nullptr;
ListNode* current = head;
ListNode* next = nullptr;
while (current != nullptr) {
next = current->next; // Save next node
current->next = prev; // Reverse the link
prev = current; // Move prev to current
current = next; // Move to next node
}
return prev; // New head of the reversed list
}
};
Saved
Ln 1, Col 1
Testcase
Test Result
Accepted Runtime: 0 ms
Case 1
Case 2
Case 3
Input
head =
Type here to search
ENG
09:37
14-02-2025

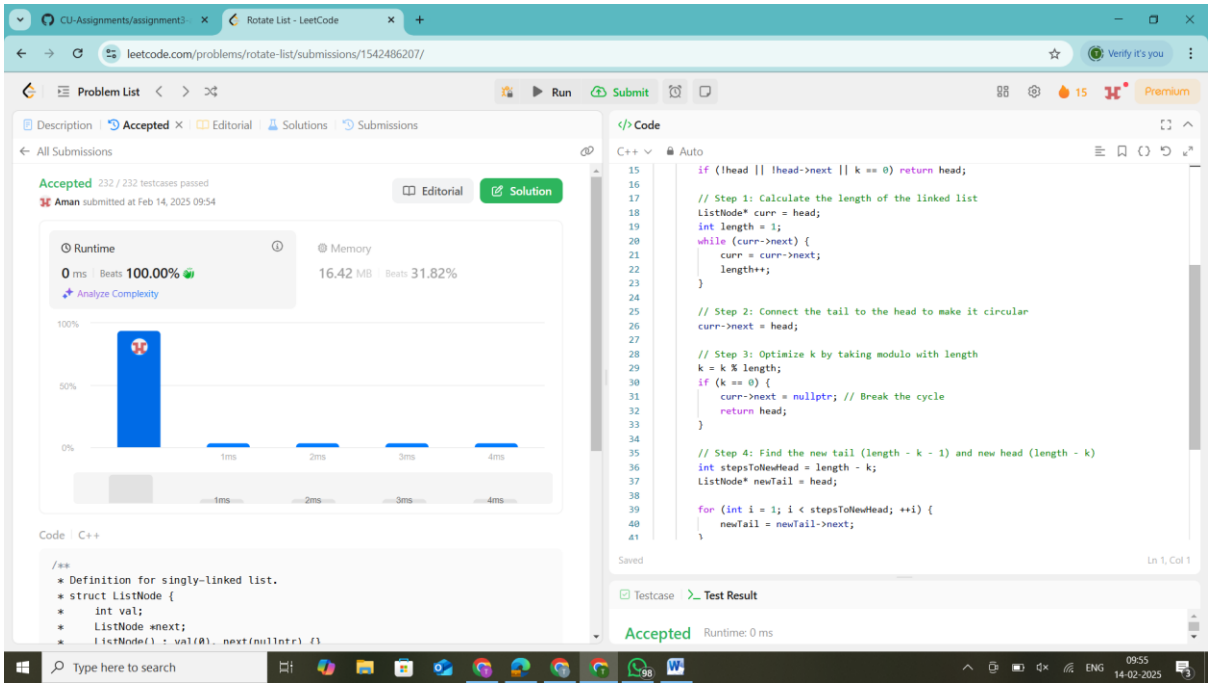

CU-Assignments/assignment3
Delete the Middle Node of a Li
leetcode.com/problems/delete-the-middle-node-of-a-linked-list/submissions/1542473840/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 70 / 70 testcases passed
Aman submitted at Feb 14, 2025 09:40
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
312.08 MB | Beats 55.19%
Code | C++
/**
* Definition for singly-linked list.
* struct ListNode {
*   int val;
*   ListNode *next;
*   ListNode() : val(0), next(nullptr) {}
Code
C++
Auto
ListNode* deleteMiddle(ListNode* head) {
if (head == nullptr || head->next == nullptr) {
return nullptr;
}
// Initialize pointers
ListNode* slow = head;
ListNode* fast = head;
ListNode* prev = nullptr;
// Move slow by 1 step and fast by 2 steps
while (fast != nullptr && fast->next != nullptr) {
prev = slow;
slow = slow->next;
fast = fast->next->next;
}
// 'slow' is now at the middle node
// Skip the middle node
prev->next = slow->next;
// Free the memory of the middle node (optional)
delete slow;
return head;
}
};
Saved
Ln 37, Col 21
Testcase
Test Result
Accepted Runtime: 0 ms
Type here to search
ENG
09:41
14-02-2025

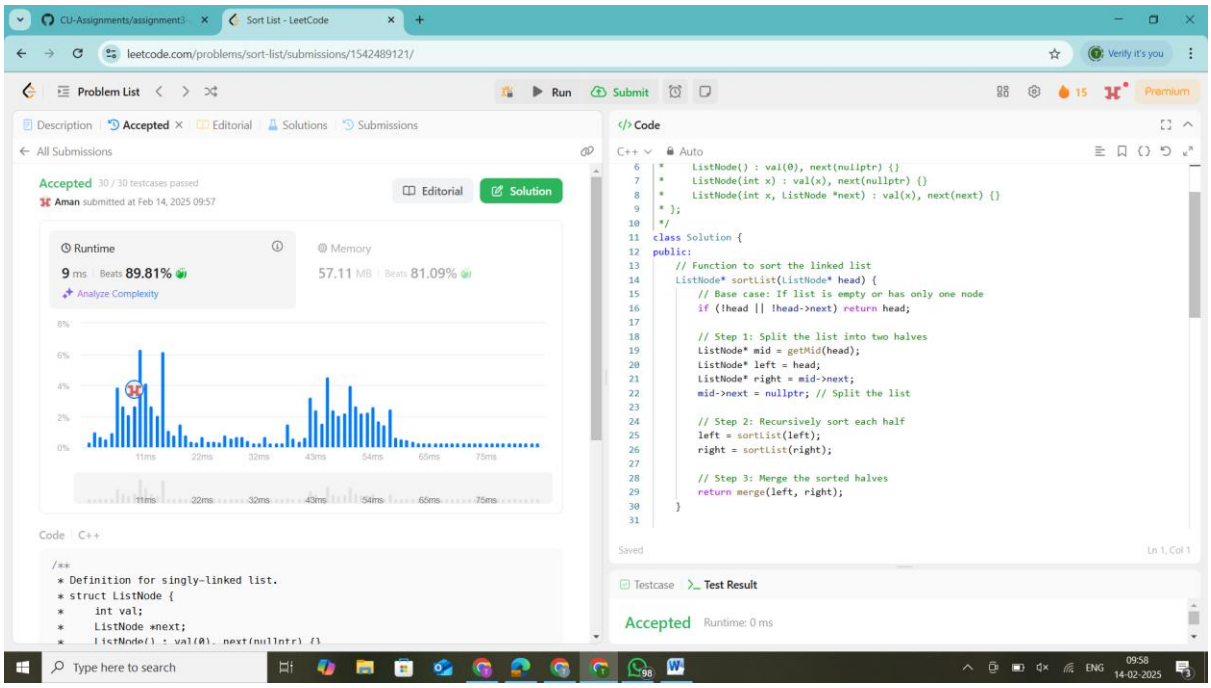

leetcode.com/problems/merge-two-sorted-lists/submissions/1542476164/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 208 / 208 testcases passed
Aman submitted at Feb 14, 2025 09:43
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
19.45 MB | Beats 62.56%
Code
C++
Auto
class Solution {
public:
ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
ListNode* dummy = new ListNode(-1);
ListNode* tail = dummy;
// Merge lists
while (list1 != nullptr && list2 != nullptr) {
if (list1->val <= list2->val) {
tail->next = list1;
list1 = list1->next;
} else {
tail->next = list2;
list2 = list2->next;
}
tail = tail->next;
}
// Attach the remaining nodes
tail->next = (list1 != nullptr) ? list1 : list2;
// Return the merged list starting from dummy->next
return dummy->next;
}
Saved
Ln 33, Col 28
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
* ListNode *next;
Type here to search
ENG
09:43
14-02-2025

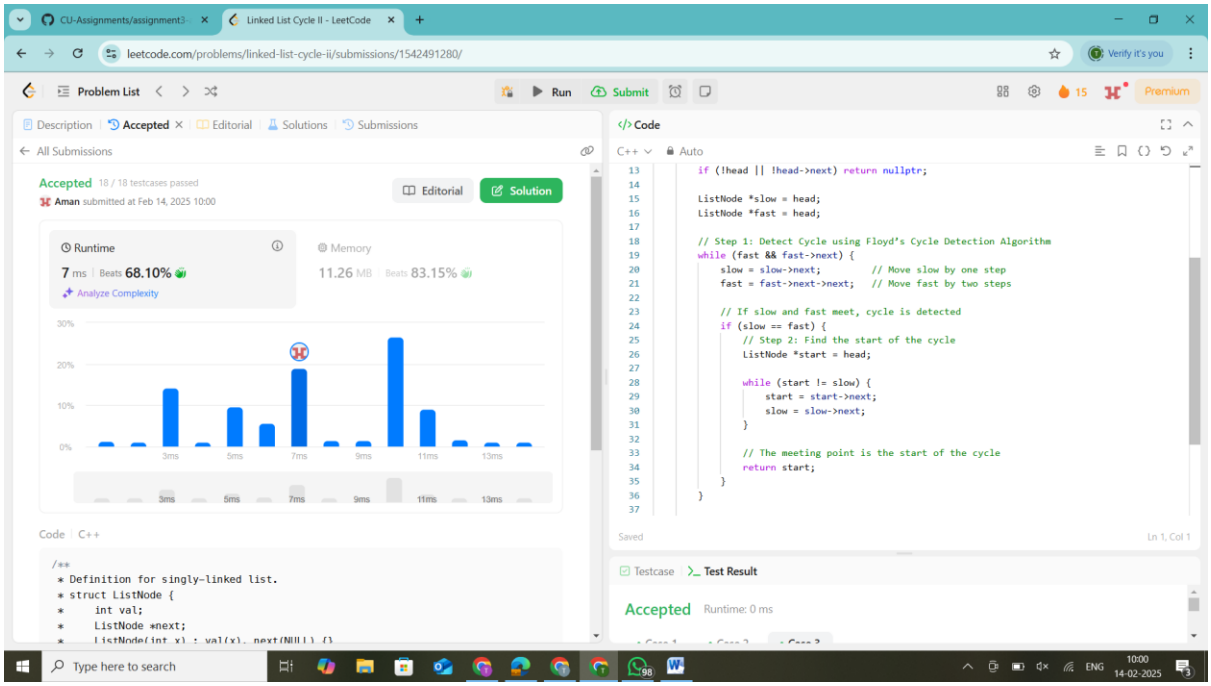

leetcode.com/problems/remove-duplicates-from-sorted-list-ii/submissions/1542478098/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 166 / 166 testcases passed
Aman submitted at Feb 14, 2025 09:45
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
15.64 MB | Beats 73.02%
Code
C++
Auto
public:
ListNode* deleteDuplicates(ListNode* head) {
ListNode* dummy = new ListNode(0, head);
ListNode* prev = dummy;
// Traverse the list
while (head) {
// Check for duplicates
if (head->next && head->val == head->next->val) {
// Skip nodes with the same value
while (head->next && head->val == head->next->val) {
head = head->next;
}
// Link prev to the next non-duplicate node
prev->next = head->next;
} else {
// No duplicates, move prev to current node
prev = prev->next;
}
// Move head forward
head = head->next;
}
// Return the head of the modified list
return dummy->next;
}
};
Saved
Ln 36, Col 28
Testcase
Test Result
Accepted
Runtime: 0 ms
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
* ListNode *next;
Type here to search
ENG
09:46
14-02-2025


leetcode.com/problems/linked-list-cycle/submissions/1542480674/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 29 / 29 testcases passed
Aman submitted at Feb 14, 2025 09:48
Editorial
Solution
Runtime
12 ms | Beats 40.45%
Analyze Complexity
Memory
11.68 MB | Beats 96.53%
Code
C++
Auto
class Solution {
public:
bool hasCycle(ListNode *head) {
// Initialize slow and fast pointers
ListNode *slow = head;
ListNode *fast = head;
// Traverse the list
while (fast && fast->next) {
slow = slow->next; // Move slow by one step
fast = fast->next->next; // Move fast by two steps
// Check if slow and fast meet
if (slow == fast) {
return true;
}
}
// If no cycle is found
return false;
}
};
Saved
Ln 1, Col 1
Testcase
Test Result
Accepted
Runtime: 2 ms
Case 1
Case 2
Case 3
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
* ListNode *next;
Type here to search
ENG
09:49
14-02-2025


leetcode.com/problems/reverse-linked-list-ii/submissions/1542483340/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 44 / 44 testcases passed
Aman submitted at Feb 14, 2025 09:51
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
11.18 MB | Beats 73.55%
Code
C++
Auto
if (left == right) return head;
// Dummy node to handle edge cases
ListNode* dummy = new ListNode(0);
dummy->next = head;
ListNode* prev = dummy;
// Move prev to the node before the `left` position
for (int i = 1; i < left; ++i) {
prev = prev->next;
}
// Start reversing the sublist
ListNode* curr = prev->next;
ListNode* next = nullptr;
for (int i = 0; i < right - left; ++i) {
next = curr->next;
curr->next = next->next;
next->next = prev->next;
prev->next = next;
}
// Return the head of the modified list
return dummy->next;
}
};
Saved
Ln 1, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
* ListNode *next;
Type here to search
ENG
09:52
14-02-2025


Rotate List - LeetCode
leetcode.com/problems/rotate-list/submissions/1542486207/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 232 / 232 testcases passed
Aman submitted at Feb 14, 2025 09:54
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
16.42 MB | Beats 31.82%
Code | C++
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
* ListNode *next;
Code
C++
Auto
if (!head || !head->next || k == 0) return head;
// Step 1: Calculate the length of the linked list
ListNode* curr = head;
int length = 1;
while (curr->next) {
curr = curr->next;
length++;
}
// Step 2: Connect the tail to the head to make it circular
curr->next = head;
// Step 3: Optimize k by taking modulo with length
k = k % length;
if (k == 0) {
curr->next = nullptr; // Break the cycle
return head;
}
// Step 4: Find the new tail (length - k - 1) and new head (length - k)
int stepsToNewHead = length - k;
ListNode* newTail = head;
for (int i = 1; i < stepsToNewHead; ++i) {
newTail = newTail->next;
Saved
Ln 1, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Type here to search
ENG
09:55
14-02-2025


Sort List - LeetCode
leetcode.com/problems/sort-list/submissions/1542489121/
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 30 / 30 testcases passed
Aman submitted at Feb 14, 2025 09:57
Editorial
Solution
Runtime
9 ms | Beats 89.81%
Analyze Complexity
Memory
57.11 MB | Beats 81.09%
Code | C++
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
* ListNode *next;
Code
C++
Auto
* ListNode() : val(0), next(nullptr) {}
* ListNode(int x) : val(x), next(nullptr) {}
* ListNode(int x, ListNode *next) : val(x), next(next) {}
* };
*/
class Solution {
public:
// Function to sort the linked list
ListNode* sortList(ListNode* head) {
// Base case: If list is empty or has only one node
if (!head || !head->next) return head;
// Step 1: Split the list into two halves
ListNode* mid = getMid(head);
ListNode* left = head;
ListNode* right = mid->next;
mid->next = nullptr; // Split the list
// Step 2: Recursively sort each half
left = sortList(left);
right = sortList(right);
// Step 3: Merge the sorted halves
return merge(left, right);
}
Saved
Ln 1, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Type here to search
ENG
09:58
14-02-2025


CU-Assignments/assignment3
Linked List Cycle II - LeetCode
leetcode.com/problems/linked-list-cycle-ii/submissions/1542491280/
Verify it's you
Problem List
Run
Submit
15
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 18 / 18 testcases passed
Aman submitted at Feb 14, 2025 10:00
Editorial
Solution
Runtime
7 ms | Beats 68.10%
Analyze Complexity
Memory
11.26 MB | Beats 83.15%
30%
20%
10%
0%
3ms
5ms
7ms
9ms
11ms
13ms
Code | C++
/**
* Definition for singly-linked list.
* struct ListNode {
*   int val;
*   ListNode *next;
*   ListNode(int x) : val(x), next(NULL) {}
Code
C++
Auto
13 if (!head || !head->next) return nullptr;
14
15 ListNode *slow = head;
16 ListNode *fast = head;
17
18 // Step 1: Detect Cycle using Floyd's Cycle Detection Algorithm
19 while (fast && fast->next) {
20 slow = slow->next; // Move slow by one step
21 fast = fast->next->next; // Move fast by two steps
22
23 // If slow and fast meet, cycle is detected
24 if (slow == fast) {
25 // Step 2: Find the start of the cycle
26 ListNode *start = head;
27
28 while (start != slow) {
29 start = start->next;
30 slow = slow->next;
31 }
32
33 // The meeting point is the start of the cycle
34 return start;
35 }
36 }
37
Saved
Ln 1, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Type here to search
ENG
10:00
14-02-2025