**Assignment 9(Complex Problem)**

**Name – Aditya Taragi Uid – 22BCS10422.**

**Problem 1: Set Matrix Zeroes (**[**https://leetcode.com/problems/set-matrix-zeroes/**](https://leetcode.com/problems/set-matrix-zeroes/) **)**

**Code:***class Solution {*

*public:*

*void setZeroes(vector<vector<int>>& matrix) {*

*int m = matrix.size();*

*int n = matrix[0].size();*

*bool firstRow = false, firstCol = false;*

*for (int j = 0; j < n; ++j) {*

*if (matrix[0][j] == 0) {*

*firstRow = true;*

*break;*

*}*

*}*

*for (int i = 0; i < m; ++i) {*

*if (matrix[i][0] == 0) {*

*firstCol = true;*

*break;*

*}*

*}*

*for (int i = 1; i < m; ++i) {*

*for (int j = 1; j < n; ++j) {*

*if (matrix[i][j] == 0) {*

*matrix[i][0] = 0;*

*matrix[0][j] = 0;*

*}*

*}*

*}*

*for (int i = 1; i < m; ++i) {*

*for (int j = 1; j < n; ++j) {*

*if (matrix[i][0] == 0 || matrix[0][j] == 0) {*

*matrix[i][j] = 0;*

*}*

*}*

*}*

*if (firstRow) {*

*for (int j = 0; j < n; ++j) {*

*matrix[0][j] = 0;*

*}*

*}*

*if (firstCol) {*

*for (int i = 0; i < m; ++i) {*

*matrix[i][0] = 0;*

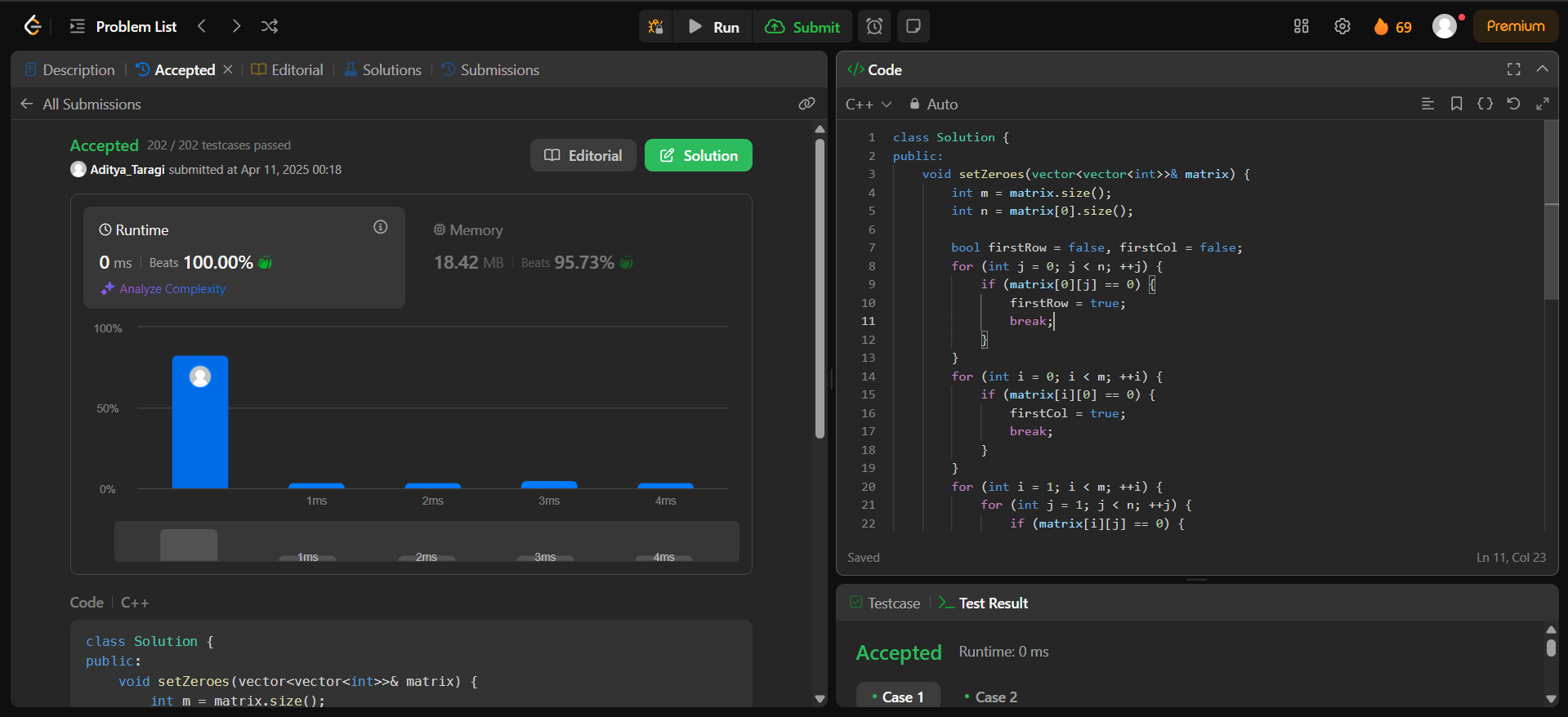
*}*

*}*

*}*

*};*

**Screenshot:**



**Problem 2: Longest Substring Without Repeating Characters (**[**https://leetcode.com/problems/longest-substring-without-repeating-characters/**](https://leetcode.com/problems/longest-substring-without-repeating-characters/) **)**

**Code:***class Solution {*

*public:*

*int lengthOfLongestSubstring(string s) {*

*if(s.length() == 0) return 0;*

*int count = 1;*

*int maxi = 1;*

*for(int i = 0;i<s.length();i++) {*

*count = 1;*

*for(int j = i+1;j<s.length();j++) {*

*bool isDuplicate = false;*

*for(int k = i;k<j;k++) {*

*if(s[k] == s[j]) {*

*isDuplicate = true;*

*break;*

*}*

*}*

*if(!isDuplicate){*

*count++;*

*}*

*else {*

*break;*

*}*

*}*

*maxi = max(maxi,count);*

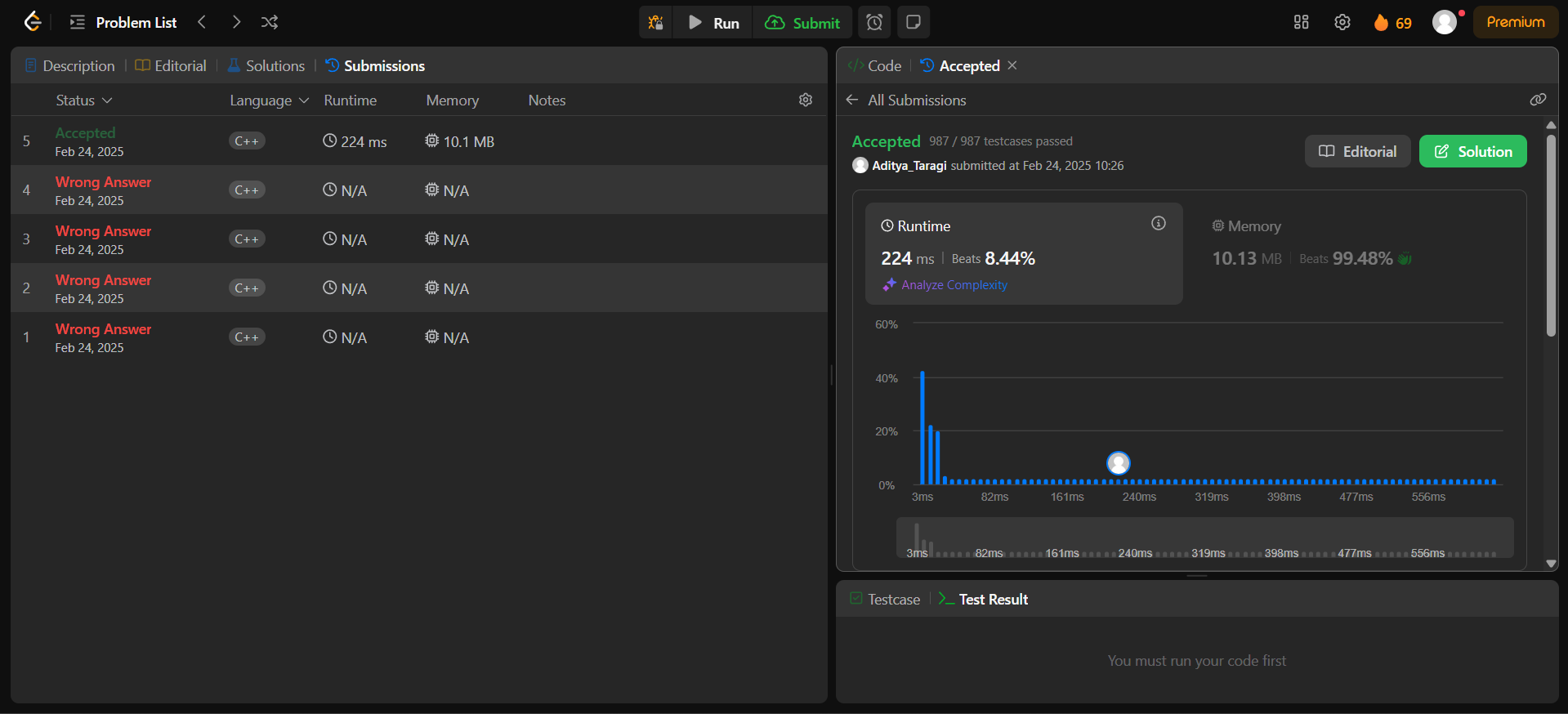
*}*

*return maxi;*

*}*

*};*

**Screenshot:**



**Problem 3: Reverse Linked List II (**[**https://leetcode.com/problems/reverse-linked-list-ii/**](https://leetcode.com/problems/reverse-linked-list-ii/) **)**

**Code:**class Solution {

public:

    ListNode\* reverseBetween(ListNode\* head, int left, int right) {

        if (!head || left == right) return head;

        ListNode\* dummy = new ListNode(0);

        dummy->next = head;

        ListNode\* prev = dummy;

        for (int i = 1; i < left; ++i) {

            prev = prev->next;

        }

        ListNode\* current = prev->next;

        ListNode\* nextNode = nullptr;

        for (int i = 0; i < right - left; ++i) {

            nextNode = current->next;

            current->next = nextNode->next;

            nextNode->next = prev->next;

            prev->next = nextNode;

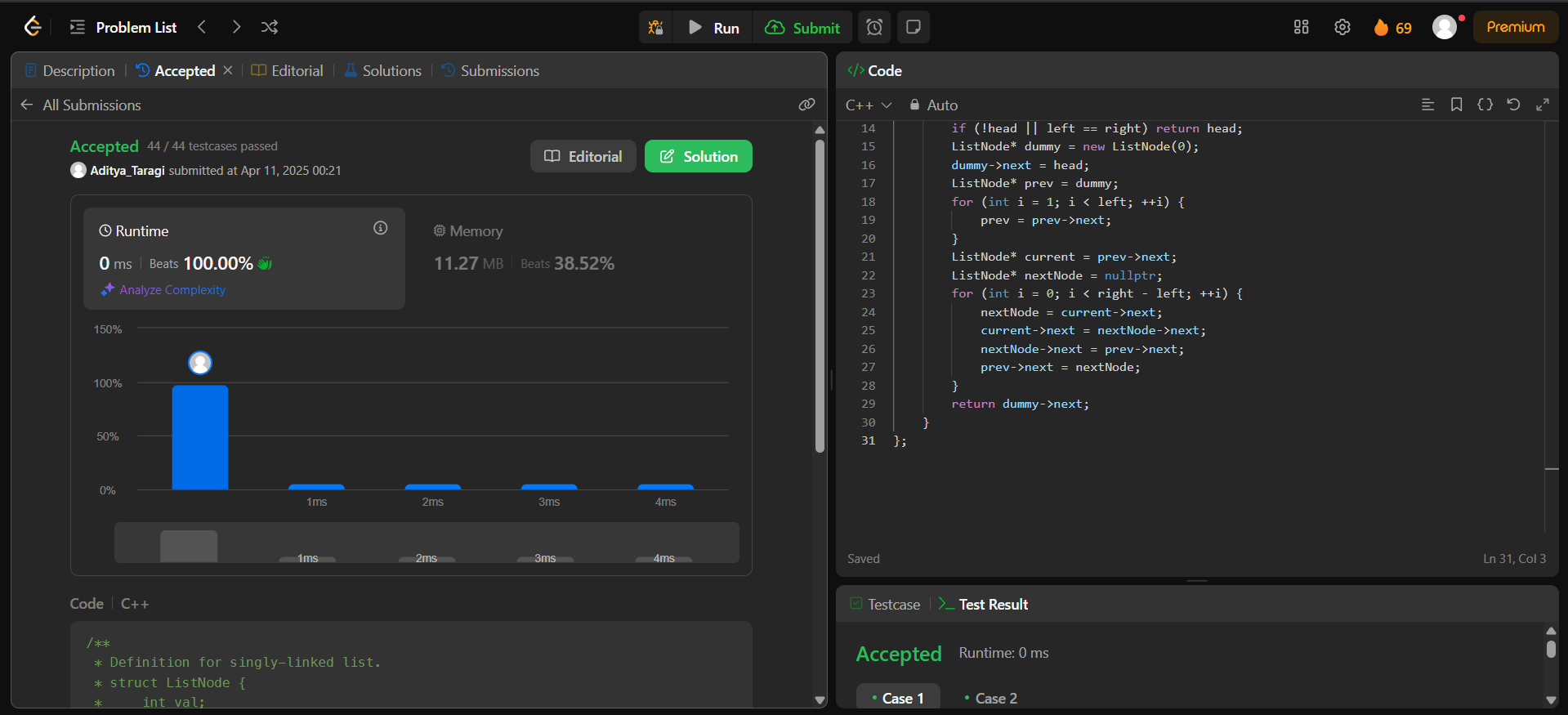
        }

        return dummy->next;

    }

};

**Screenshot:**



**Problem 4: Linked List Cycle (**[**https://leetcode.com/problems/linked-list-cycle/**](https://leetcode.com/problems/linked-list-cycle/) **)**

**Code:***bool* hasCycle(ListNode \**head*) {

    if (!*head* || !*head*->next) return false;

    ListNode\* slow = *head*;

    ListNode\* fast = *head*->next;

    while (slow != fast) {

        if (!fast || !fast->next)

            return false;

        slow = slow->next;

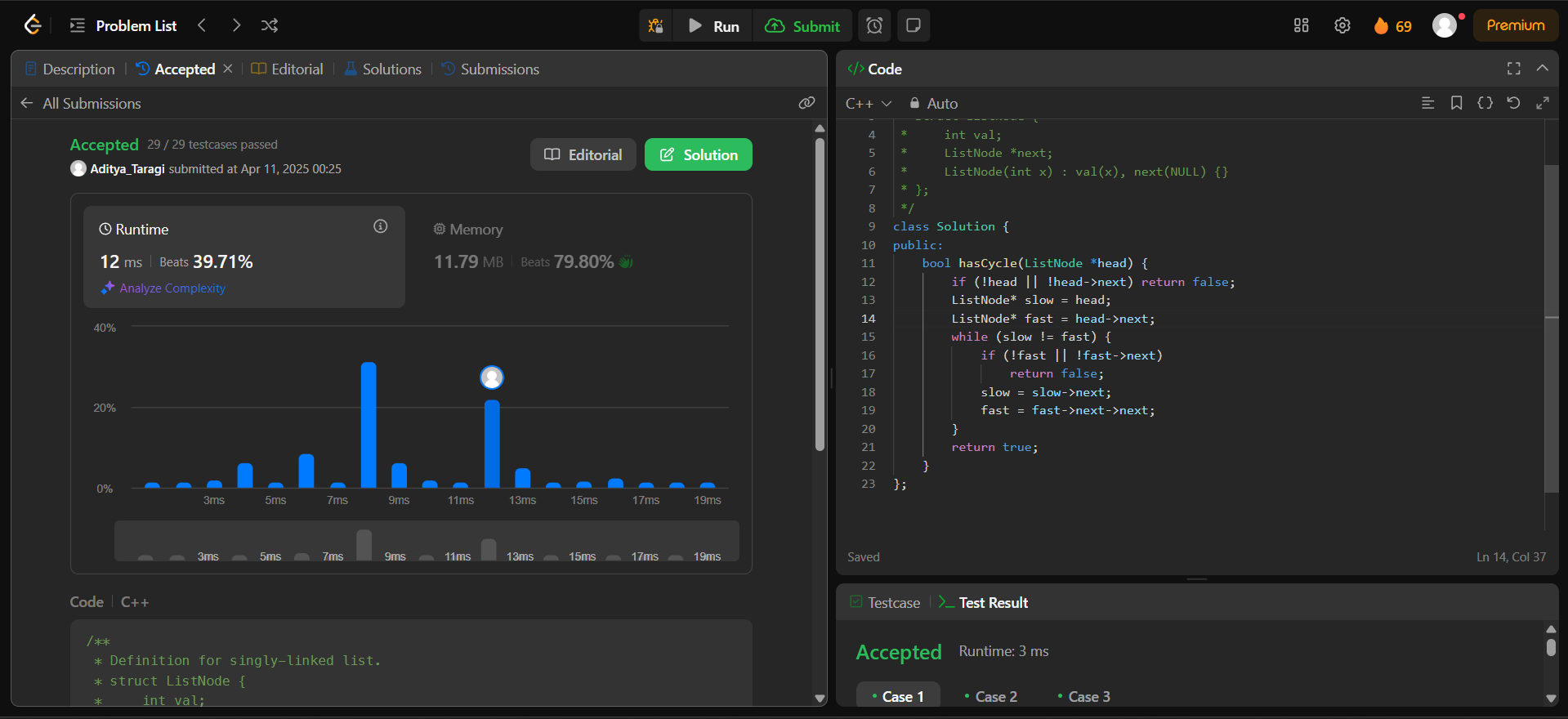
        fast = fast->next->next;

    }

    return true;

}

**Screenshot:**

****

**Problem 5: The Skyline Problem (**[**https://leetcode.com/problems/the-skyline-problem/**](https://leetcode.com/problems/the-skyline-problem/) **)**

**Code:**class Solution {

public:

    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {

        vector<pair<int,int>> events;

        for(auto& b: buildings) {

            events.push\_back({b[0], -b[2]});

            events.push\_back({b[1],b[2]});

        }

        sort(events.begin(),events.end());

        multiset<int> heights = {0};

        vector<vector<int>> result;

        int prevHeight = 0;

        for(auto& [x,h] : events) {

            if(h<0) heights.insert(-h);

            else heights.erase(heights.find(h));

            int maxHeight = \*heights.rbegin();

            if(maxHeight != prevHeight) {

                result.push\_back({x, maxHeight});

                prevHeight = maxHeight;

            }

        }

        return result;

    }

}

**Screenshot:**

