**Name –** Harsh Shaw

**UID –** 22BCS15788

**Section –** 601-‘B’

**Experiment –** 1

1. **Remove duplicates from a sorted array: https://leetcode.com/problems/remove-duplicates-from-sorted-array/**

class Solution {

public:

int removeDuplicates(vector<int>& nums) {

if(nums.empty()) return 0;

int res = 0;

for(int i = 1; i < nums.size(); i++) {

if(nums[res] == nums[i]) continue;

res++;

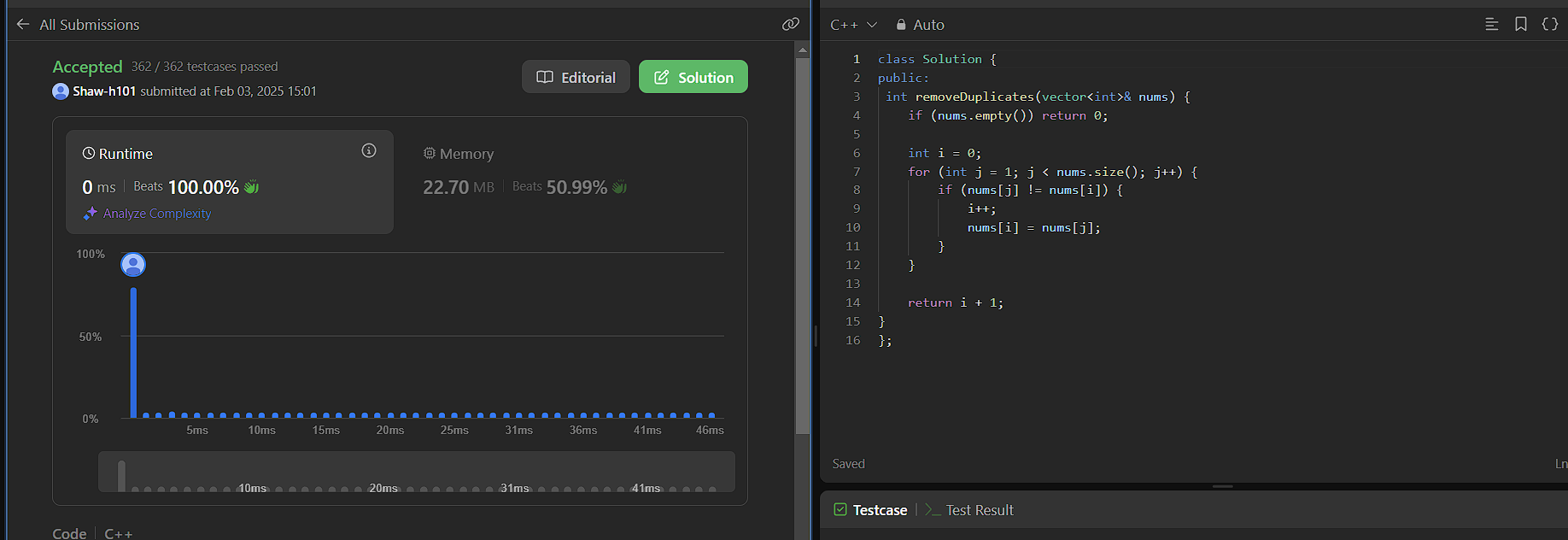
nums[res] = nums[i];

}

return res + 1;

}

};



1. **Implementing insertion sort: https://www.geeksforgeeks.org/problems/insertion-sort/1**

void insertionSort(vector<int>& arr) {

int n = arr.size();

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

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

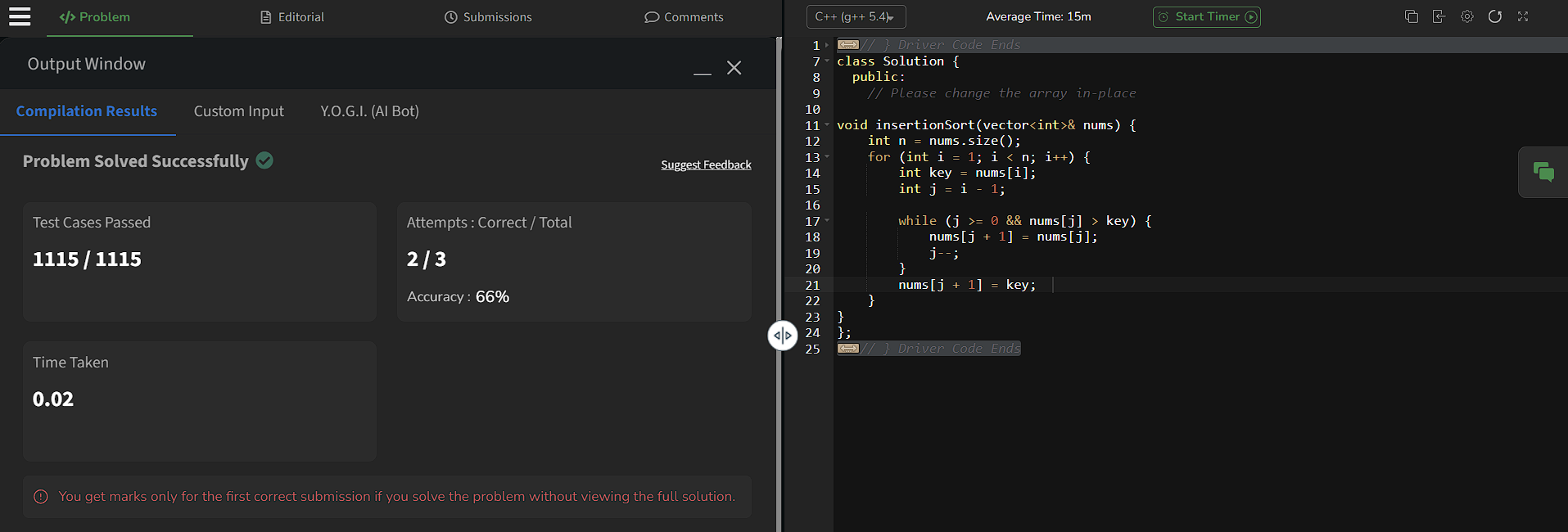
j--;

}

arr[j + 1] = key;

}

}



1. **Contains duplicate: https://leetcode.com/problems/contains-duplicate/description/**

class Solution {

public:

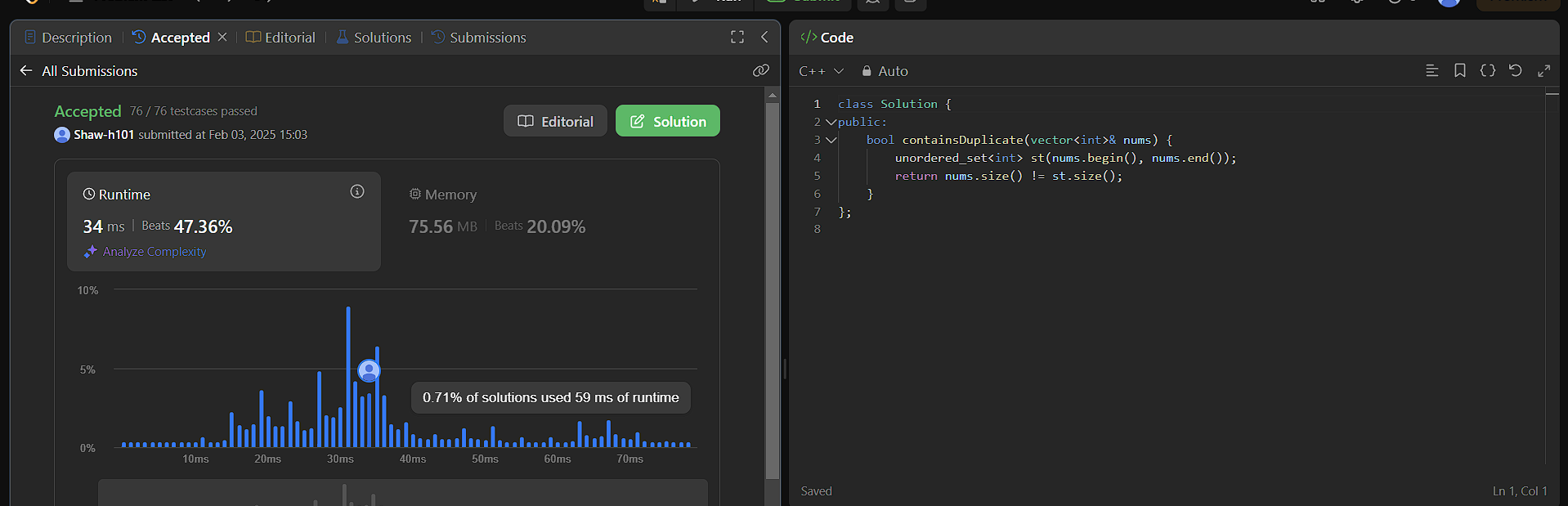
bool containsDuplicate(vector<int>& nums) {

unordered\_set<int> set(nums.begin(), nums.end());

return nums.size() !=set.size();

}

};



1. **Two Sum: https://leetcode.com/problems/two-sum/description/**

class Solution {

public:

vector<int> twoSum(vector<int>& nums, int target) {

unordered\_map<int, int> map;

for (int i = 0; i < nums.size(); i++) {

int left = target - nums[i];

if (map.find(left) != map.end()) {

return {map[left], i};

}

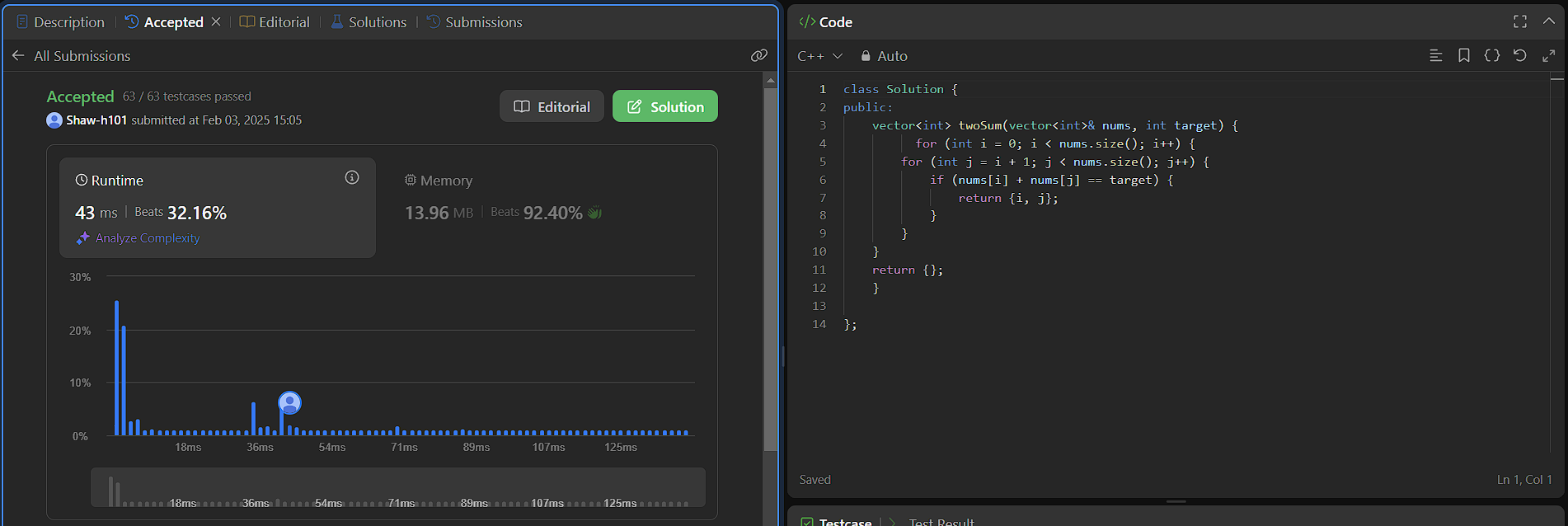
map[nums[i]] = i;

}

return {};

}

};



1. **Jump Game: https://leetcode.com/problems/jump-game/description/**

class Solution {

public:

bool canJump(vector<int>& nums) {

int jump = 0;

for (int i = 0; i < nums.size(); i++) {

if (i > jump) return false;

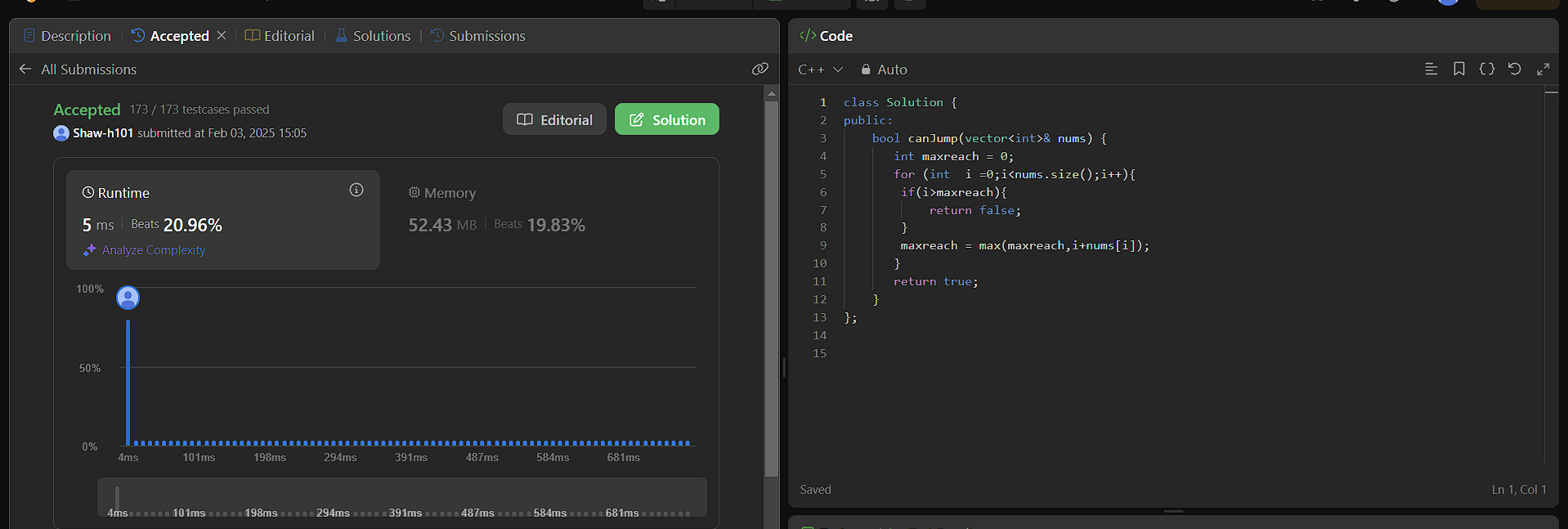
jump = max(jump, i + nums[i]);

}

return true;

}

};



1. **Majority Elements: https://leetcode.com/problems/majority-element/description/**

class Solution {

public:

int majorityElement(vector<int>& nums) {

unordered\_map<int, int> map;

for (int num : nums) {

map[num]++;

}

for (auto& pair : map) {

if (pair.second > nums.size() / 2) {

return pair.first;

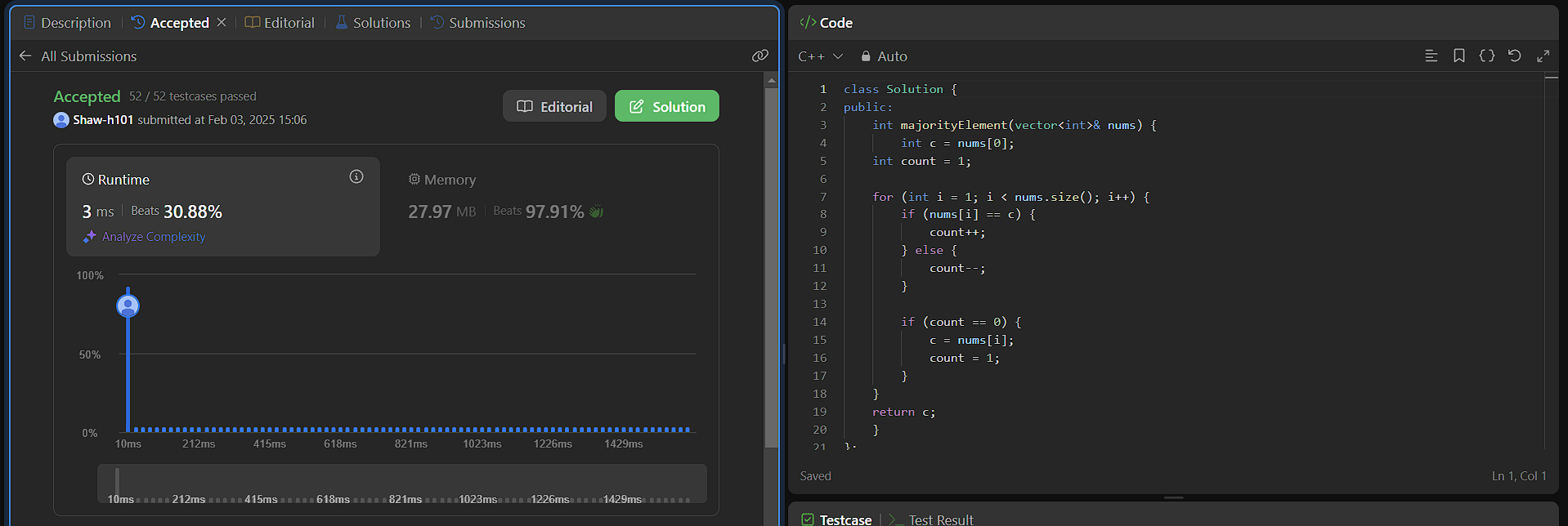
}

}

return 0;

}

};



1. **Valid Palindrome: https://leetcode.com/problems/valid-palindrome/description/**

class Solution {

public:

bool isPalindrome(string s) {

int i = 0, j = s.size() - 1;

while (i < j) {

if (!isalnum(s[i])) i++;

else if (!isalnum(s[j])) j--;

else if (tolower(s[i]) != tolower(s[j])) return false;

else {

i++;

j--;

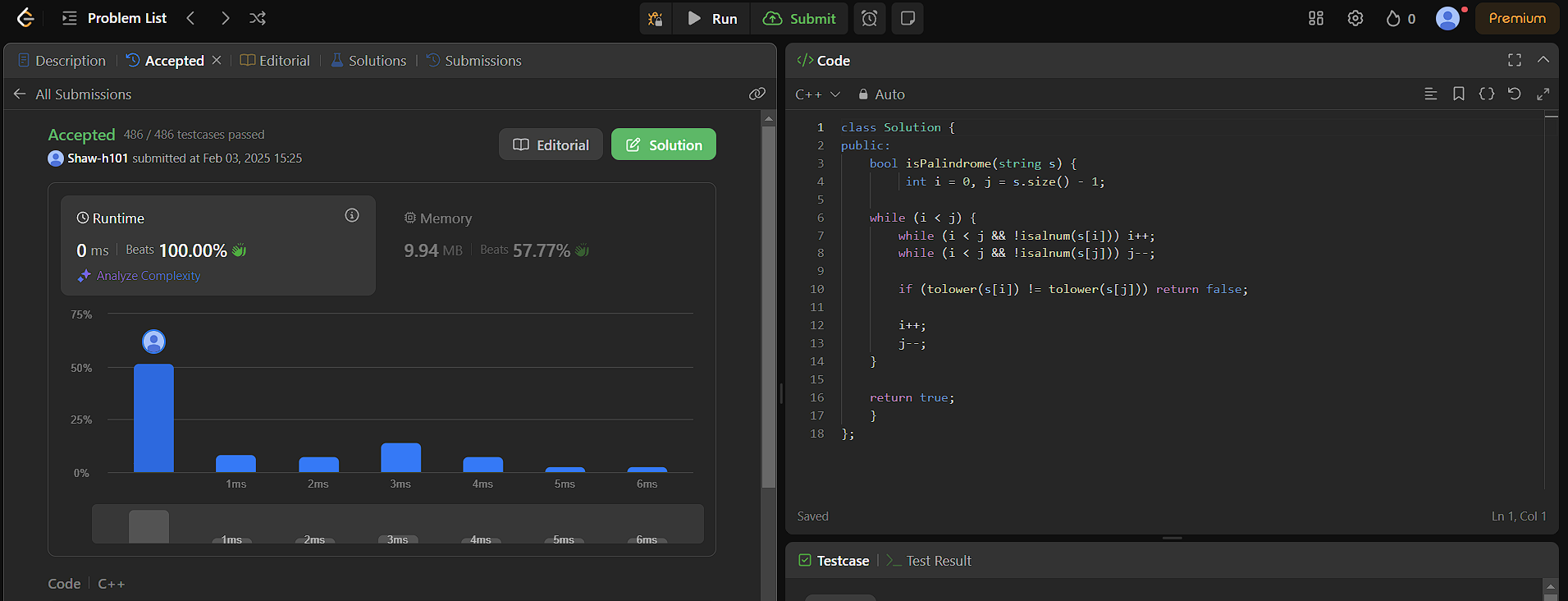
}

}

return true;

}

};



1. **Jump Game 2: https://leetcode.com/problems/jump-game-ii/description/**

class Solution {

public:

int jump(vector<int>& nums) {

int n = nums.size();

vector<int> dp(n, -1);

dp[n - 1] = 0;

for (int i = n - 2; i >= 0; i--) {

int steps = nums[i];

int ans = INT\_MAX;

for (int j = i + 1; j <= i + steps && j < n; j++) {

if (dp[j] != -1) {

ans = min(ans, dp[j] + 1);

}

}

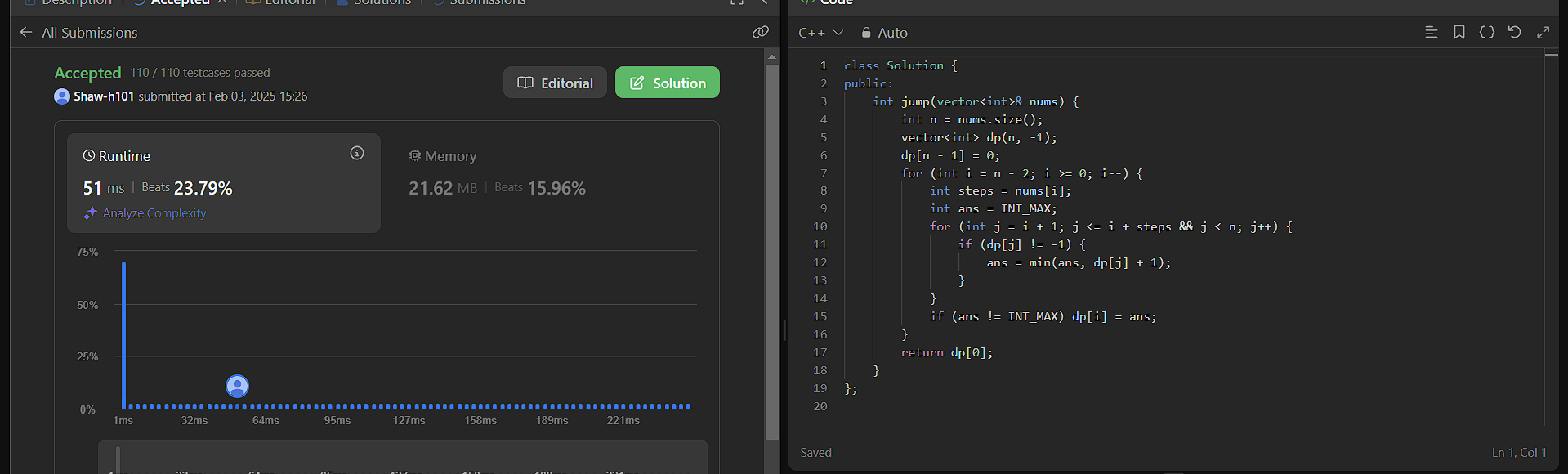
if (ans != INT\_MAX) dp[i] = ans;

}

return dp[0];

}

};



1. **3Sum: https://leetcode.com/problems/3sum/description/**

class Solution {

public:

vector<vector<int>> threeSum(vector<int>& nums) {

vector<vector<int>> result;

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

for (int i = 0; i < nums.size() - 2; i++) {

if (i > 0 && nums[i] == nums[i - 1]) continue;

int left = i + 1, right = nums.size() - 1;

while (left < right) {

int sum = nums[i] + nums[left] + nums[right];

if (sum == 0) {

result.push\_back({nums[i], nums[left], nums[right]});

while (left < right && nums[left] == nums[left + 1]) left++;

while (left < right && nums[right] == nums[right - 1]) right--;

left++; right--;

} else if (sum < 0) left++;

else right--;

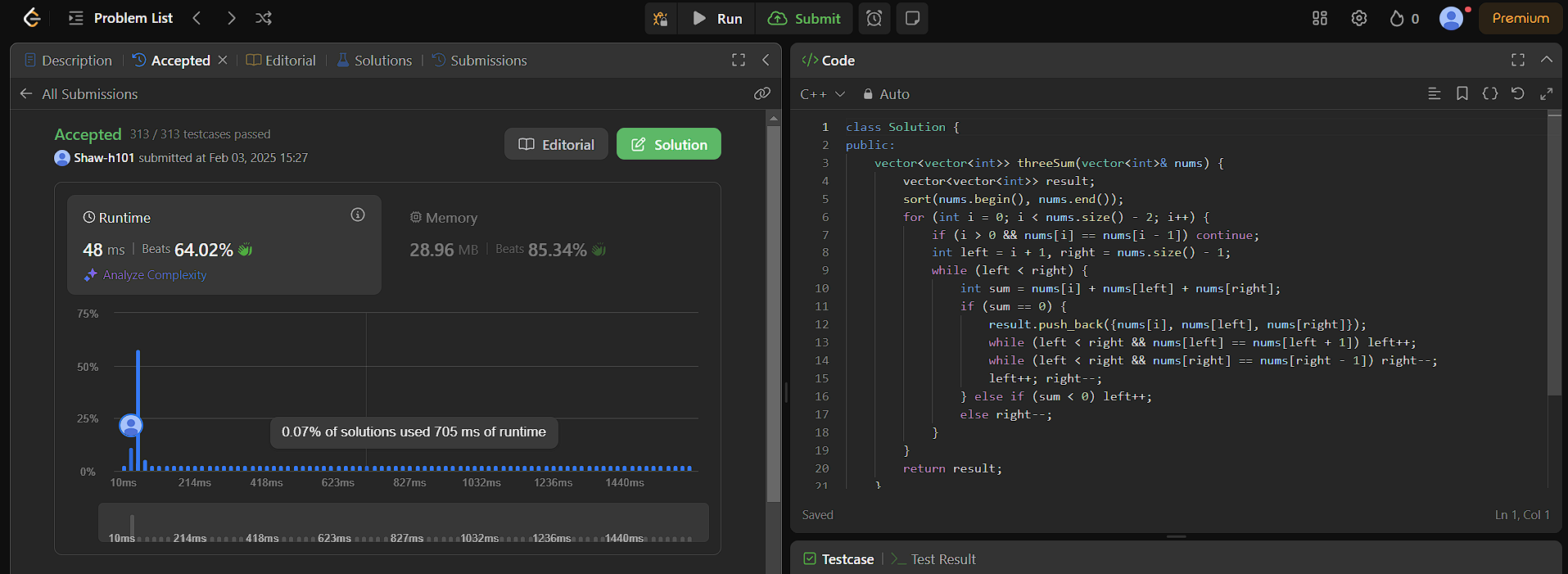
}

}

return result;

}

};



1. **Set Matrix Zeroes: https://leetcode.com/problems/set-matrix-zeroes/description/**

class Solution {

public:

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

bool fr = false, fc = false;

for (int i = 0; i < matrix.size(); i++) {

for (int j = 0; j < matrix[0].size(); j++) {

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

if (i == 0) fr = true;

if (j == 0) fc = true;

matrix[0][j] = 0;

matrix[i][0] = 0;

}

}

}

for (int i = 1; i < matrix.size(); i++) {

for (int j = 1; j < matrix[0].size(); j++) {

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

matrix[i][j] = 0;

}

}

}

if (fr) fill(matrix[0].begin(), matrix[0].end(), 0);

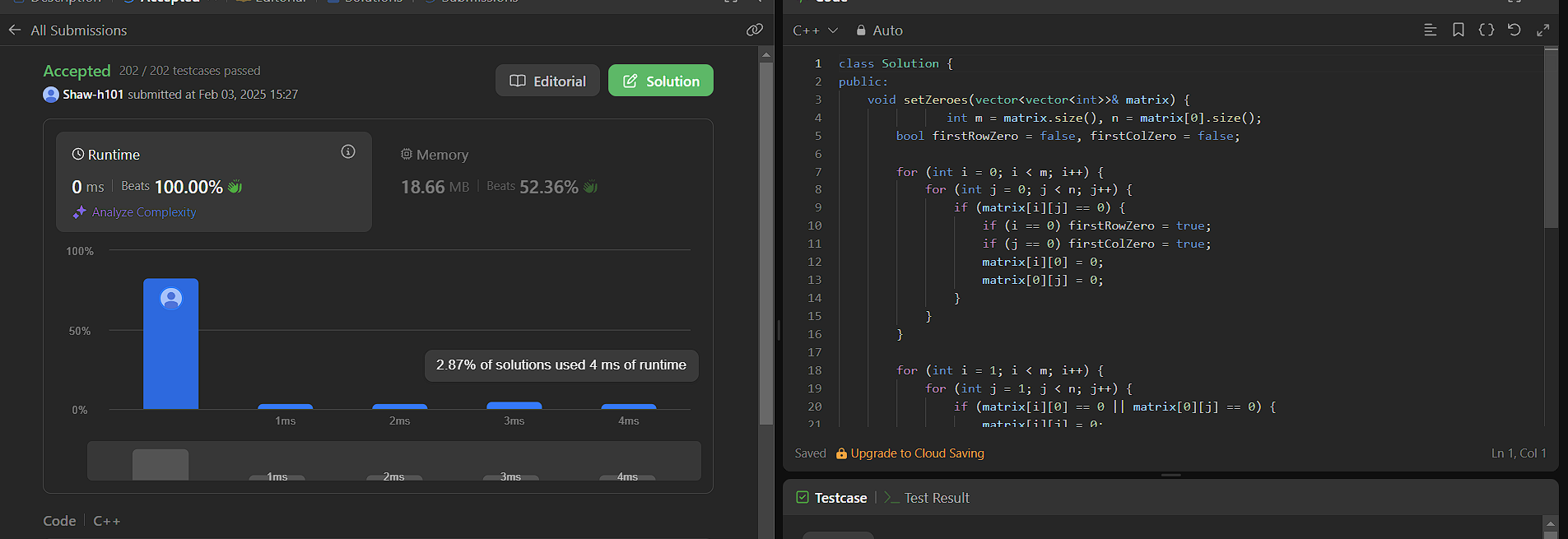
if (fc) {

for (int i = 0; i < matrix.size(); i++) matrix[i][0] = 0;

}

}

};



1. **Longest substring without repeating characters: https://leetcode.com/problems/longest-substring-without-repeating-characters/description/**

class Solution {

public:

int lengthOfLongestSubstring(string s) {

unordered\_set<char> charSet;

int left = 0, maxLength = 0;

for (int right = 0; right < s.size(); right++) {

while (charSet.count(s[right])) {

charSet.erase(s[left]);

left++;

}

charSet.insert(s[right]);

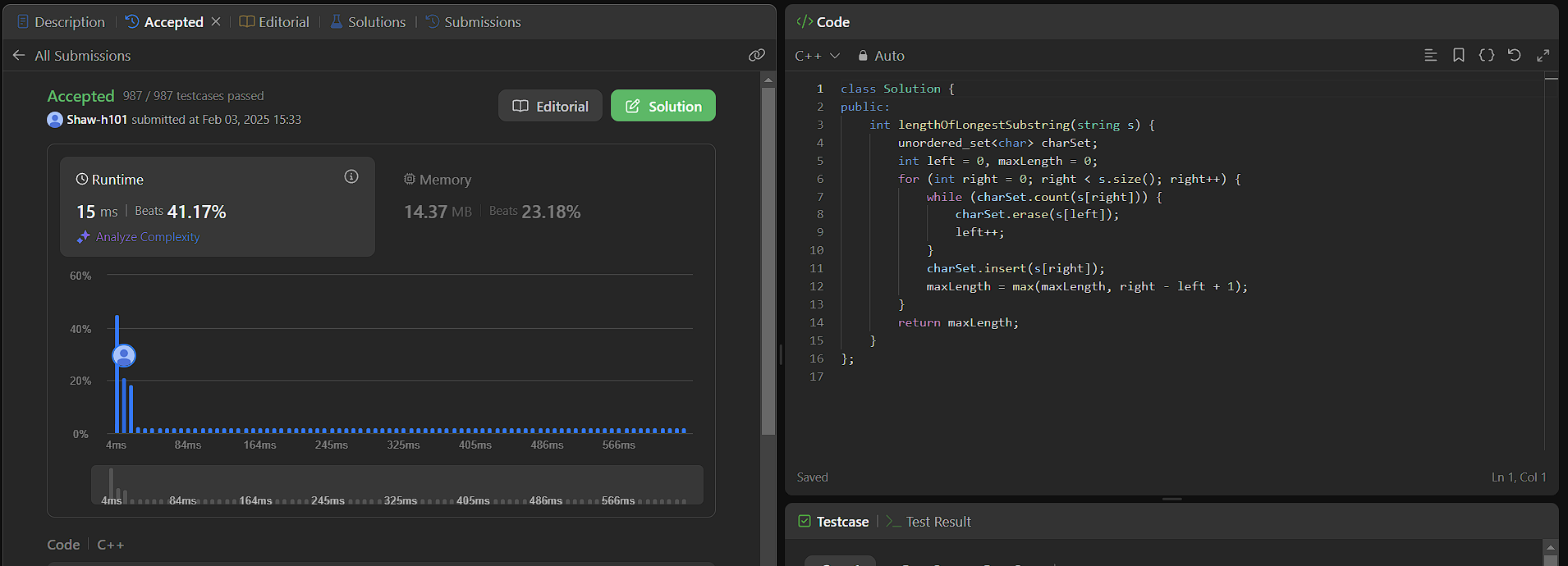
maxLength = max(maxLength, right - left + 1);

}

return maxLength;

}

};



1. **Finding duplicate number: https://leetcode.com/problems/find-the-duplicate-number/description/**

class Solution {

public:

int findDuplicate(vector<int>& nums) {

vector<bool> isUnique(nums.size(), false);

for (int num : nums) {

if (isUnique[num]) return num;

isUnique[num] = true;

}

return -1;

}

};

