**Name:** Yashita

**UID:** 22BCS15024

**Section:** FL\_IOT\_601 - A

**Assignment – 5 Solutions:-**

1. **[Find the Difference](https://leetcode.com/problems/find-the-difference/):-**

class Solution {

public:

char findTheDifference(string s, string t) {

char result = 0;

for (char c : s) result ^= c;

for (char c : t) result ^= c;

return result;

}

};

Result:-

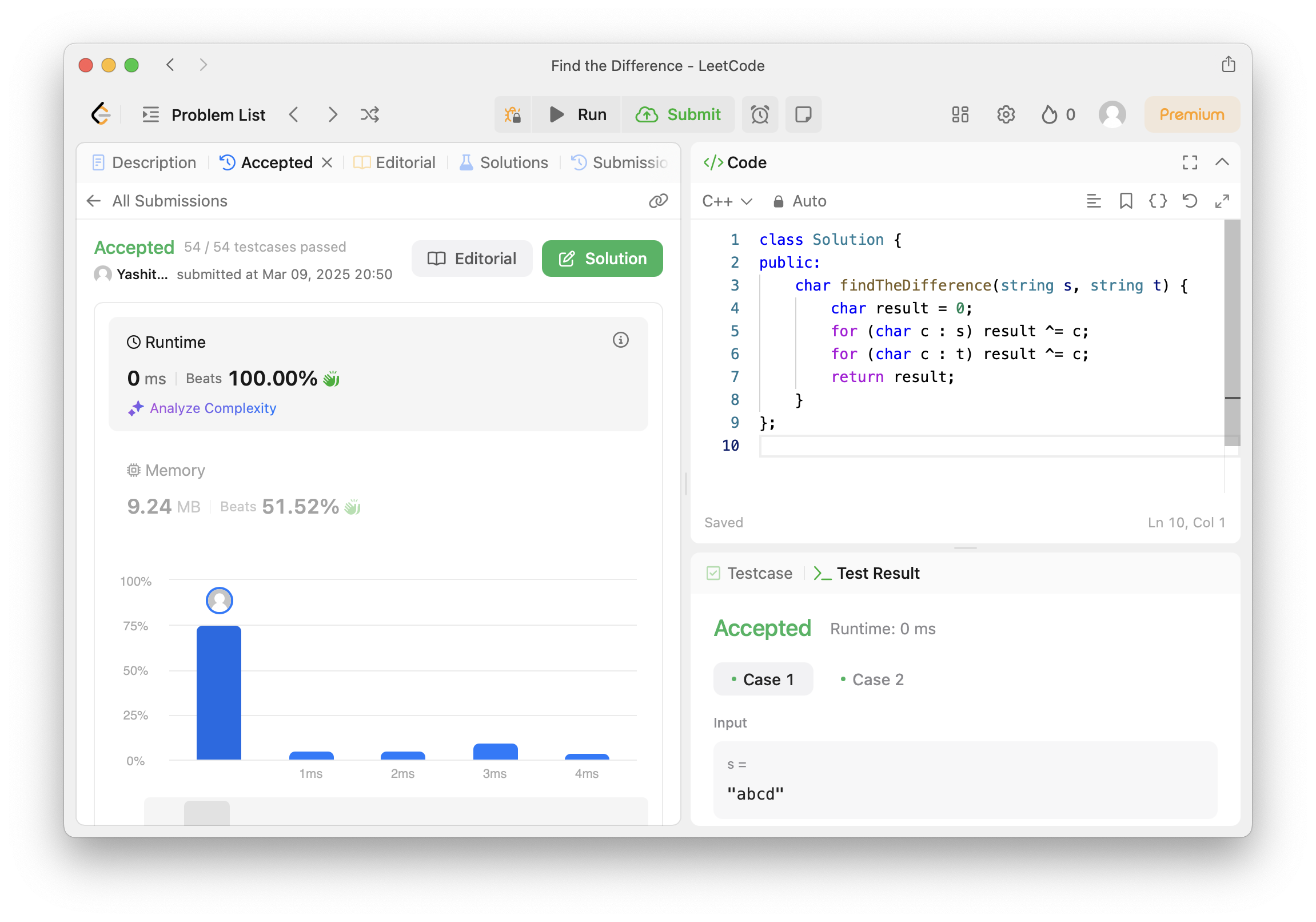
1. **[Largest Perimeter Triangle](https://leetcode.com/problems/largest-perimeter-triangle/):**

class Solution {

public:

    int largestPerimeter(vector<int>& nums) {

        sort(nums.begin(), nums.end(), greater<int>());

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

            if (nums[i] < nums[i + 1] + nums[i + 2]) {

                return nums[i] + nums[i + 1] + nums[i + 2];

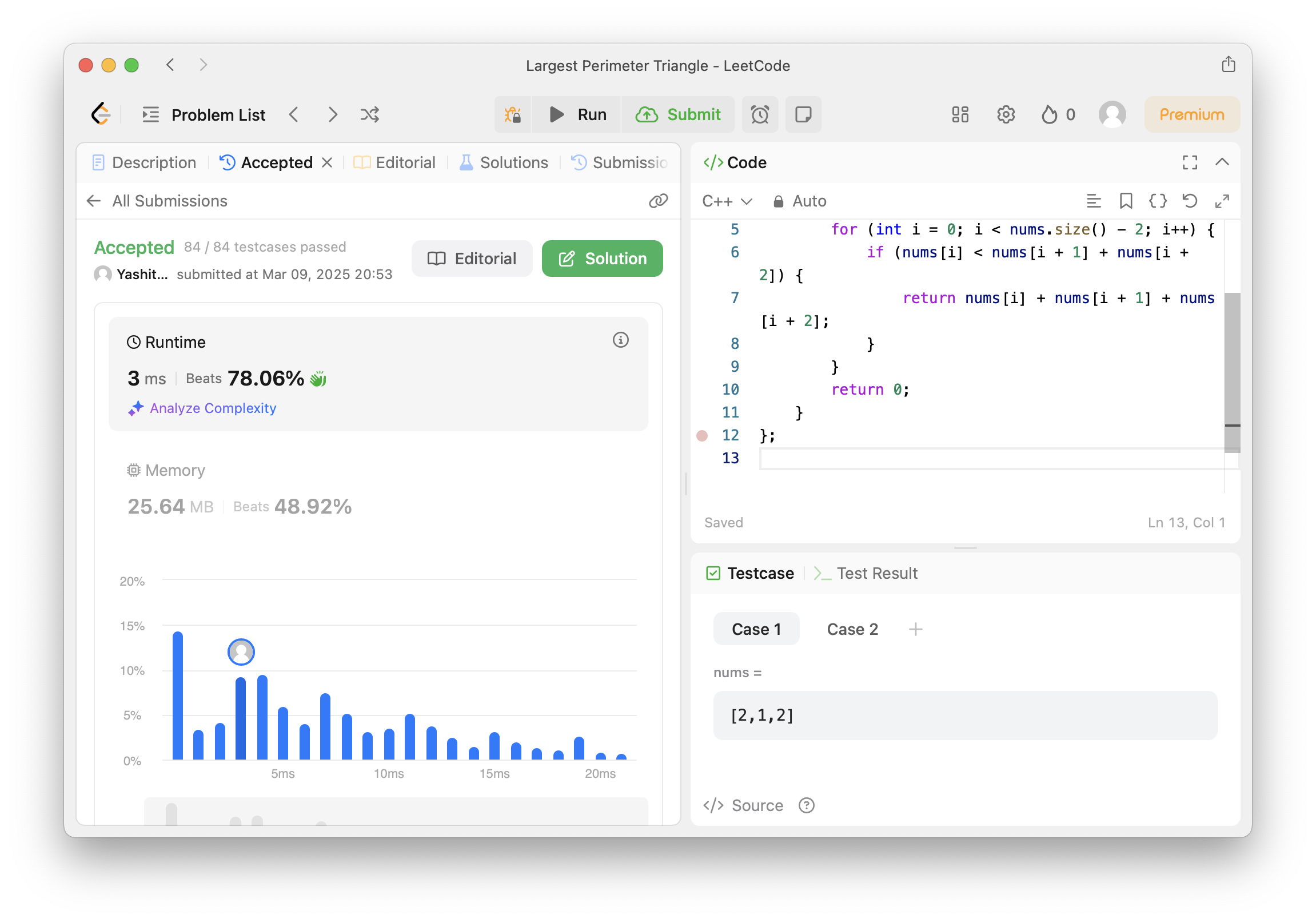
            }

        }

        return 0;

    }

};

Result:

1. **[Third Maximum Number](https://leetcode.com/problems/third-maximum-number/):**

class Solution {

public:

int thirdMax(vector<int>& nums) {

long first = LONG\_MIN, second = LONG\_MIN, third = LONG\_MIN;

for (int num : nums) {

if (num == first || num == second || num == third) continue;

if (num > first) {

third = second;

second = first;

first = num;

} else if (num > second) {

third = second;

second = num;

} else if (num > third) {

third = num;

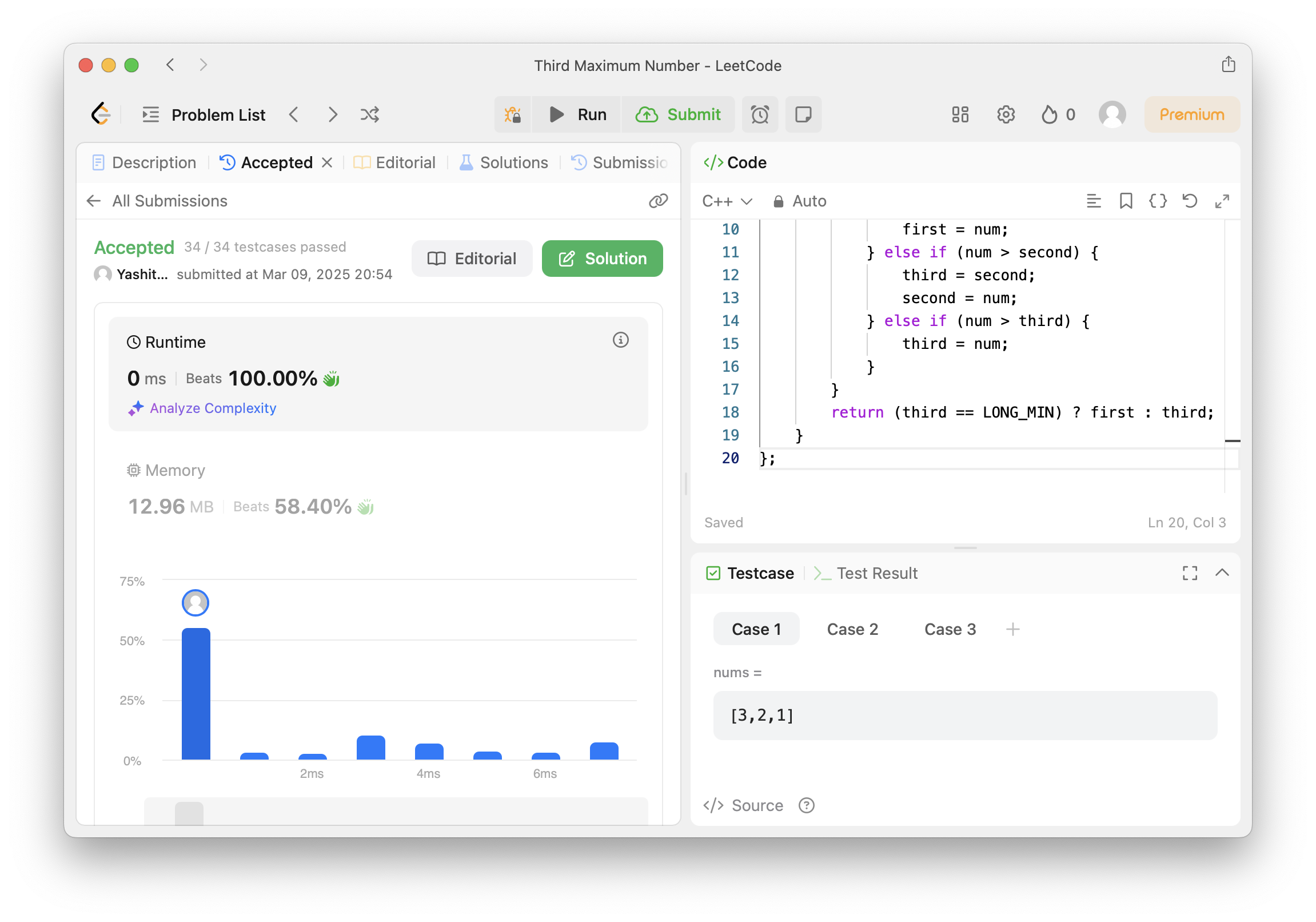
}

}

return (third == LONG\_MIN) ? first : third;

}

};

Result:

1. **[Sort Characters By Frequency](https://leetcode.com/problems/sort-characters-by-frequency/):**

class Solution {

public:

string frequencySort(string s) {

unordered\_map<char, int> freq;

for (char c : s) freq[c]++;

priority\_queue<pair<int, char>> maxHeap;

for (auto& [ch, count] : freq) {

maxHeap.push({count, ch});

}

string result;

while (!maxHeap.empty()) {

auto [count, ch] = maxHeap.top();

maxHeap.pop();

result.append(count, ch);

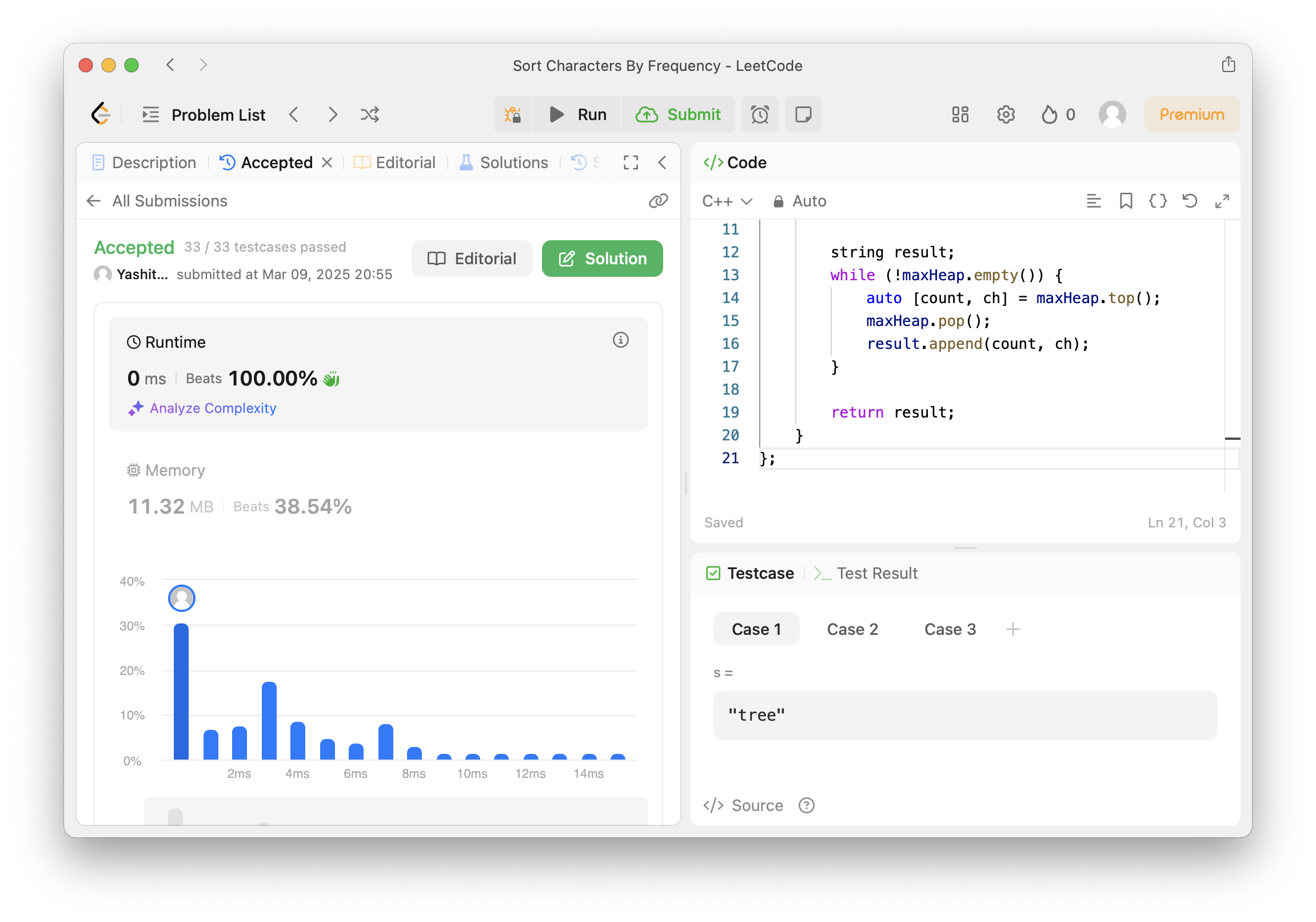
}

return result;

}

};

Result:



1. **[Minimum Number of Arrows to Burst Balloons](https://leetcode.com/problems/minimum-number-of-arrows-to-burst-balloons/):**

class Solution {

public:

int findMinArrowShots(vector<vector<int>>& points) {

if (points.empty()) return 0;

sort(points.begin(), points.end(), [](const vector<int>& a, const vector<int>& b) {

return a[1] < b[1];

});

int arrows = 1;

int end = points[0][1];

for (const auto& balloon : points) {

if (balloon[0] > end) {

arrows++;

end = balloon[1];

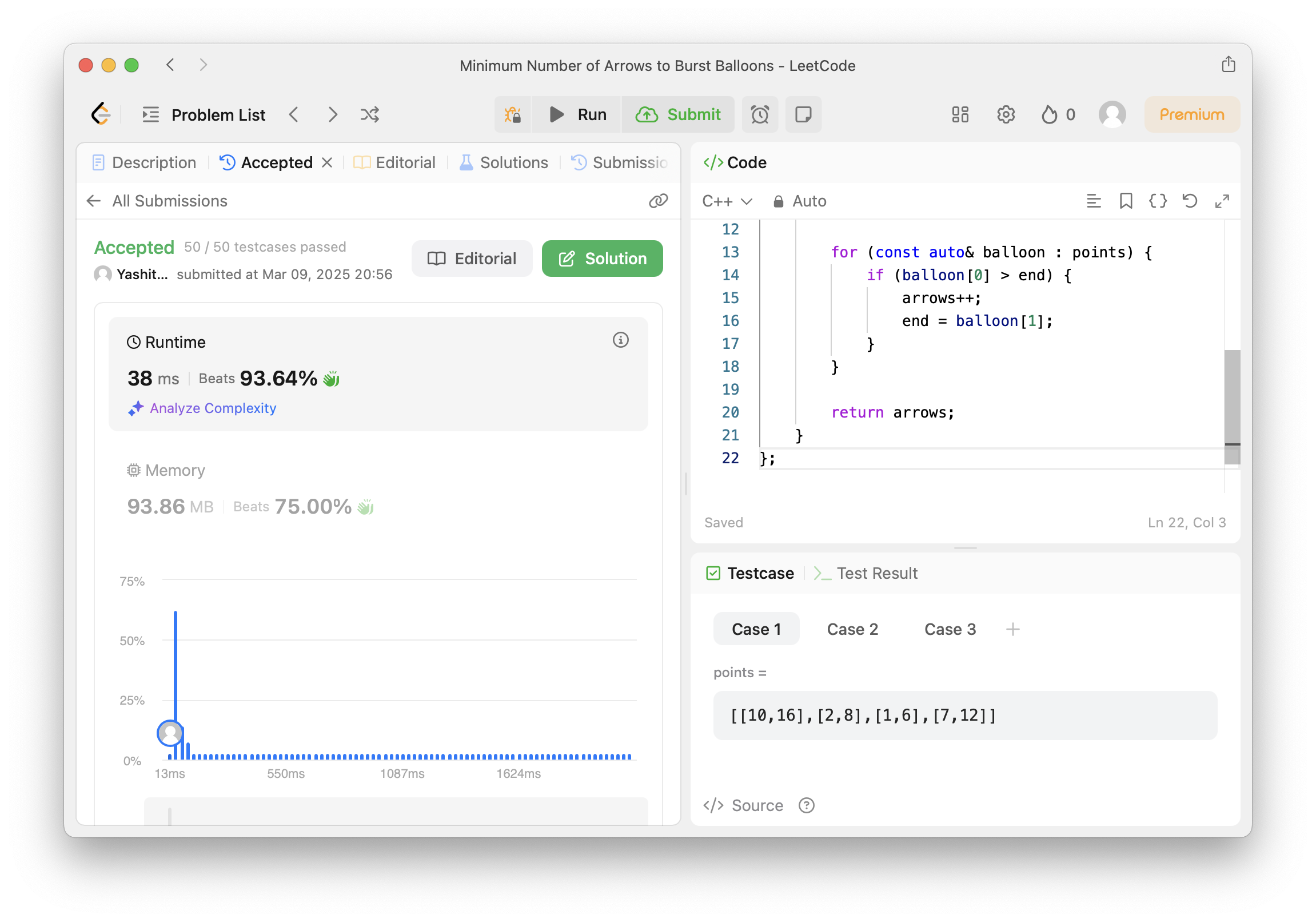
}

}

return arrows;

}

};

Result:

1. **[Boats to Save People](https://leetcode.com/problems/boats-to-save-people/):**

class Solution {

public:

int numRescueBoats(vector<int>& people, int limit) {

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

int left = 0, right = people.size() - 1;

int boats = 0;

while (left <= right) {

if (people[left] + people[right] <= limit) {

left++;

}

right--;

boats++;

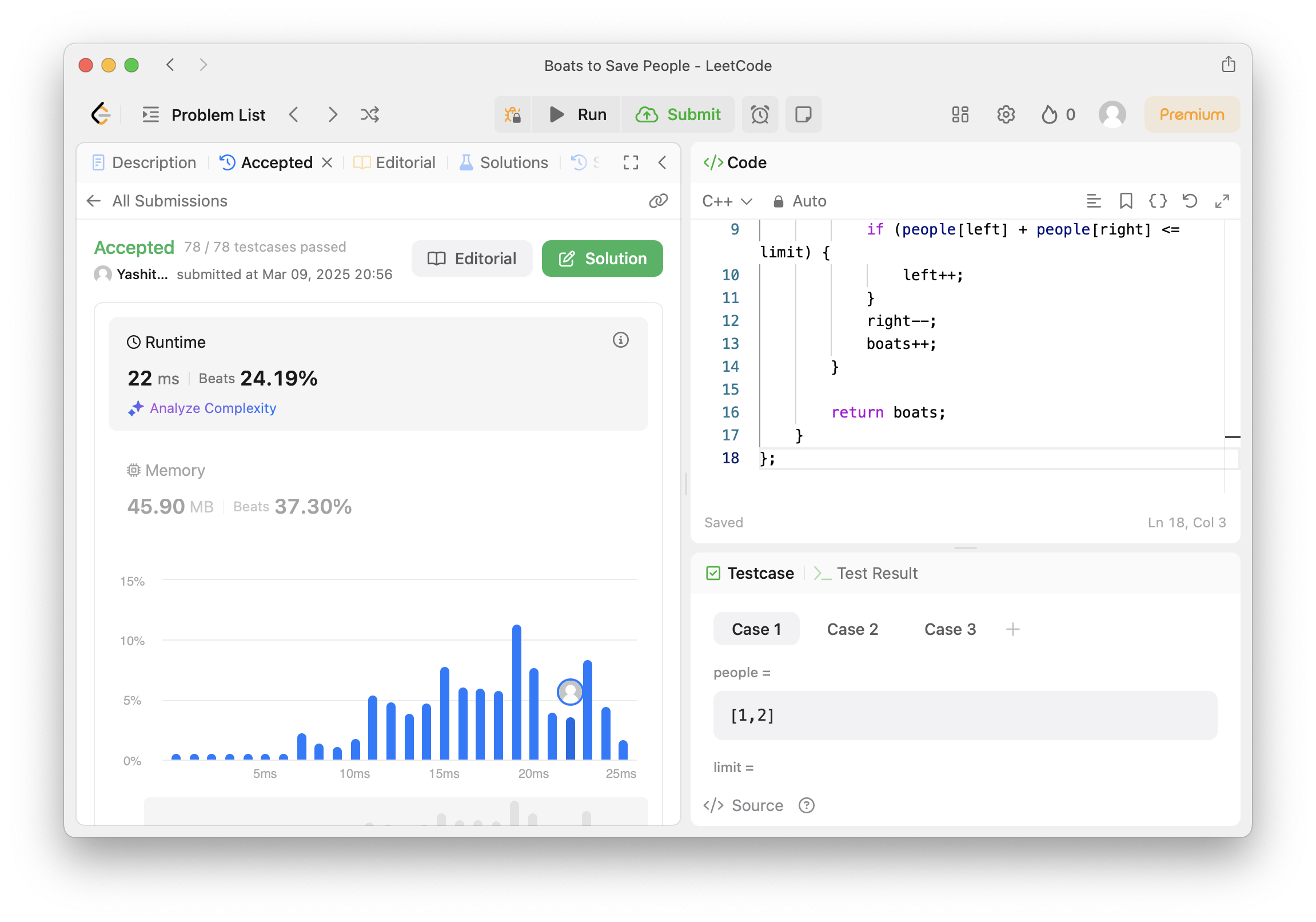
}

return boats;

}

};

Result:



1. **[K Closest Points to Origin](https://leetcode.com/problems/k-closest-points-to-origin/):**

class Solution {

public:

vector<vector<int>> kClosest(vector<vector<int>>& points, int k) {

priority\_queue<pair<int, vector<int>>> maxHeap;

for (auto& p : points) {

int dist = p[0] \* p[0] + p[1] \* p[1];

maxHeap.push({dist, p});

if (maxHeap.size() > k) {

maxHeap.pop();

}

}

vector<vector<int>> result;

while (!maxHeap.empty()) {

result.push\_back(maxHeap.top().second);

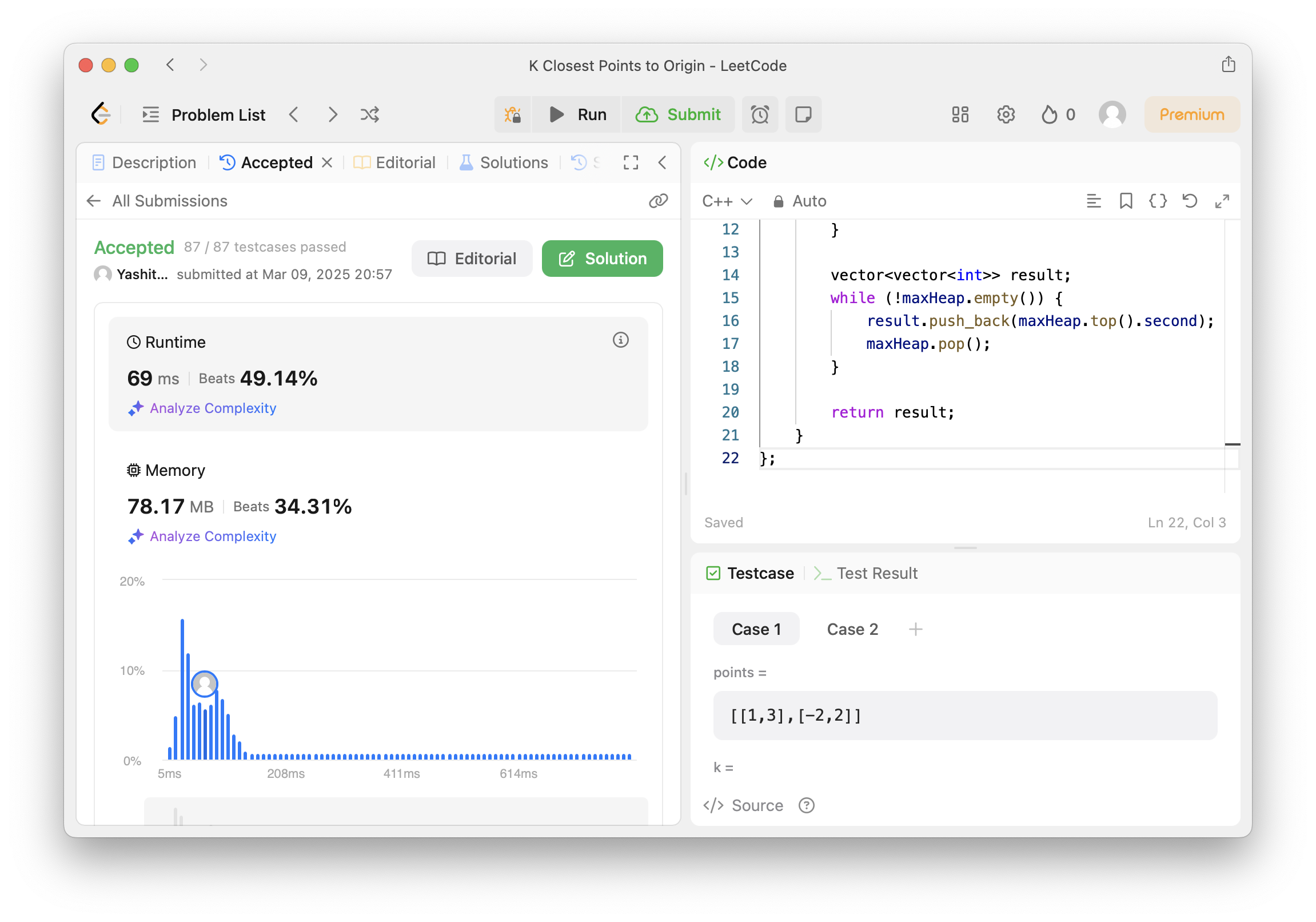
maxHeap.pop();

}

return result;

}

};

Result:

1. **[Reduce Array Size to The Half](https://leetcode.com/problems/reduce-array-size-to-the-half/):**

class Solution {

public:

int minSetSize(vector<int>& arr) {

unordered\_map<int, int> freq;

for (int num : arr) freq[num]++;

priority\_queue<int> maxHeap;

for (auto& [num, count] : freq) {

maxHeap.push(count);

}

int removed = 0, setSize = 0, halfSize = arr.size() / 2;

while (removed < halfSize) {

removed += maxHeap.top();

maxHeap.pop();

setSize++;

}

return setSize;

}

};

Result:-

