UNIVERSITY INSTITUTE OF ENGINEERING

**Department of Computer Science & Engineering**

**(BE-CSE/IT-6th Sem)**

****

**Subject Name:** Advanced Programming Lab - 2

**Subject Code:** 22CSP-351

**Submitted to: Submitted by:**

Mr. Vishal Name: Sanya Singh

UID: 23BCS14374

Section: FL\_IOT\_604 Group: A

**ASSIGNMENT-5**

**389.**[**Find the diffrence**](https://leetcode.com/problems/find-the-difference/description/)

class Solution {

public:

    char findTheDifference(string s, string t) {

        sort(t.begin(),t.end());

        sort(s.begin(),s.end());

        int c=0;

        for(int i=0;i<t.size();i++){

            if(s[i]!=t[i]){

                break;

            }

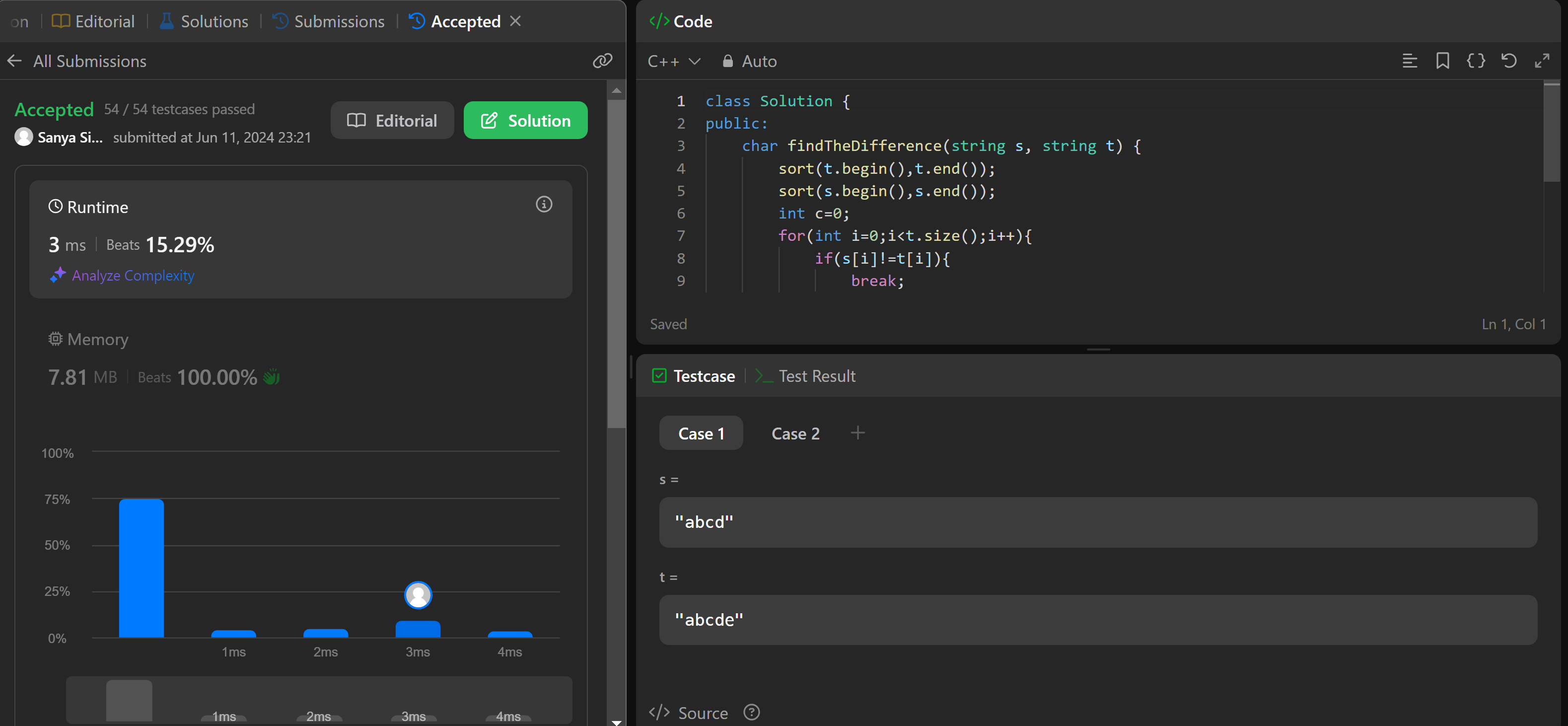
            c++;

        }

        return t[c];

    }

};



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

class Solution {

public:

    int largestPerimeter(vector<int>& nums) {

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

        for(int i=nums.size()-1;i>1;i--){

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

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

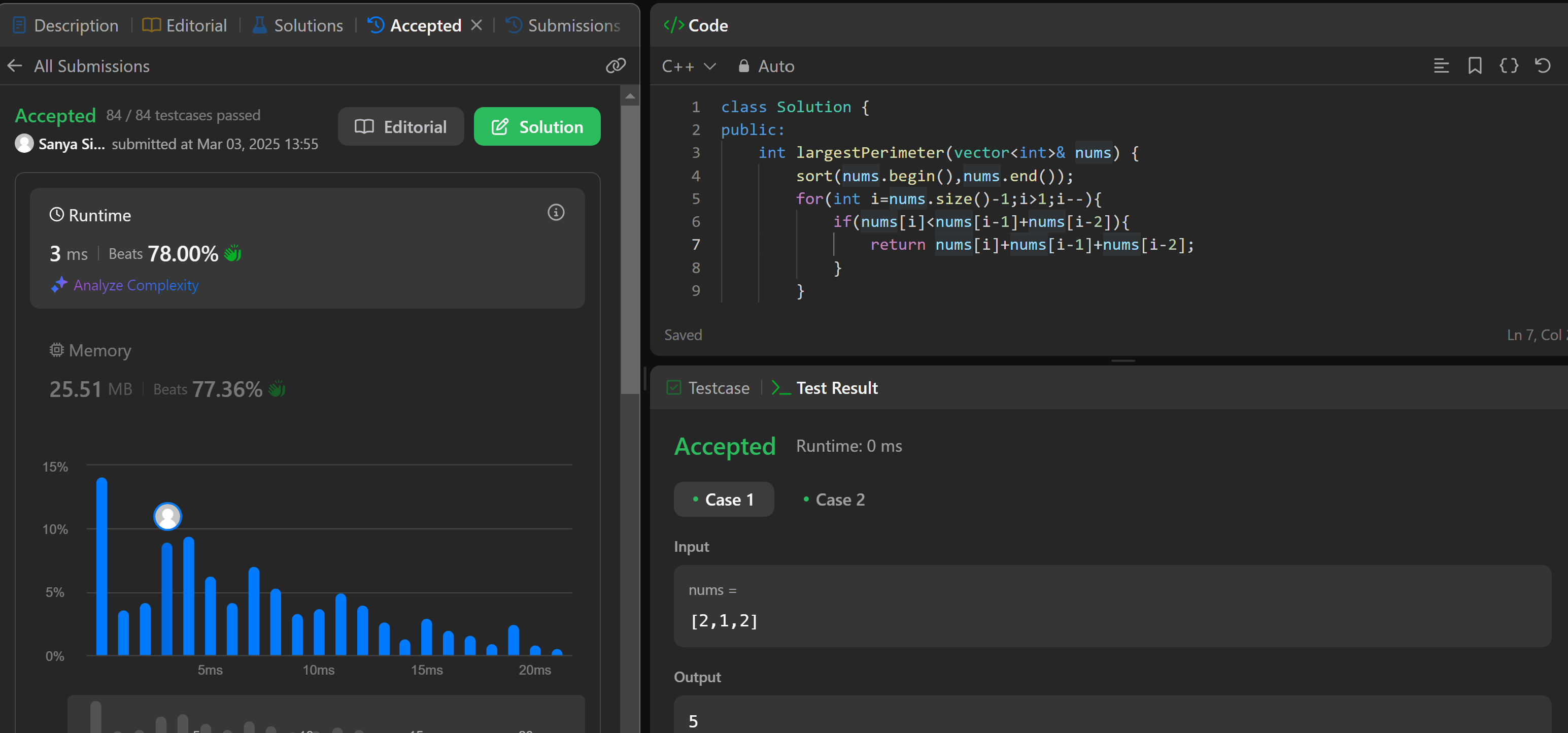
            }

        }

        return 0;

    }

};



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

class Solution {

public:

    int thirdMax(vector<int>& nums) {

        int thirdmax=0;

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

        nums.erase(unique(nums.begin(),nums.end()),nums.end());

        if(nums.size()<3){

            thirdmax=nums[nums.size()-1];

        }

        else{

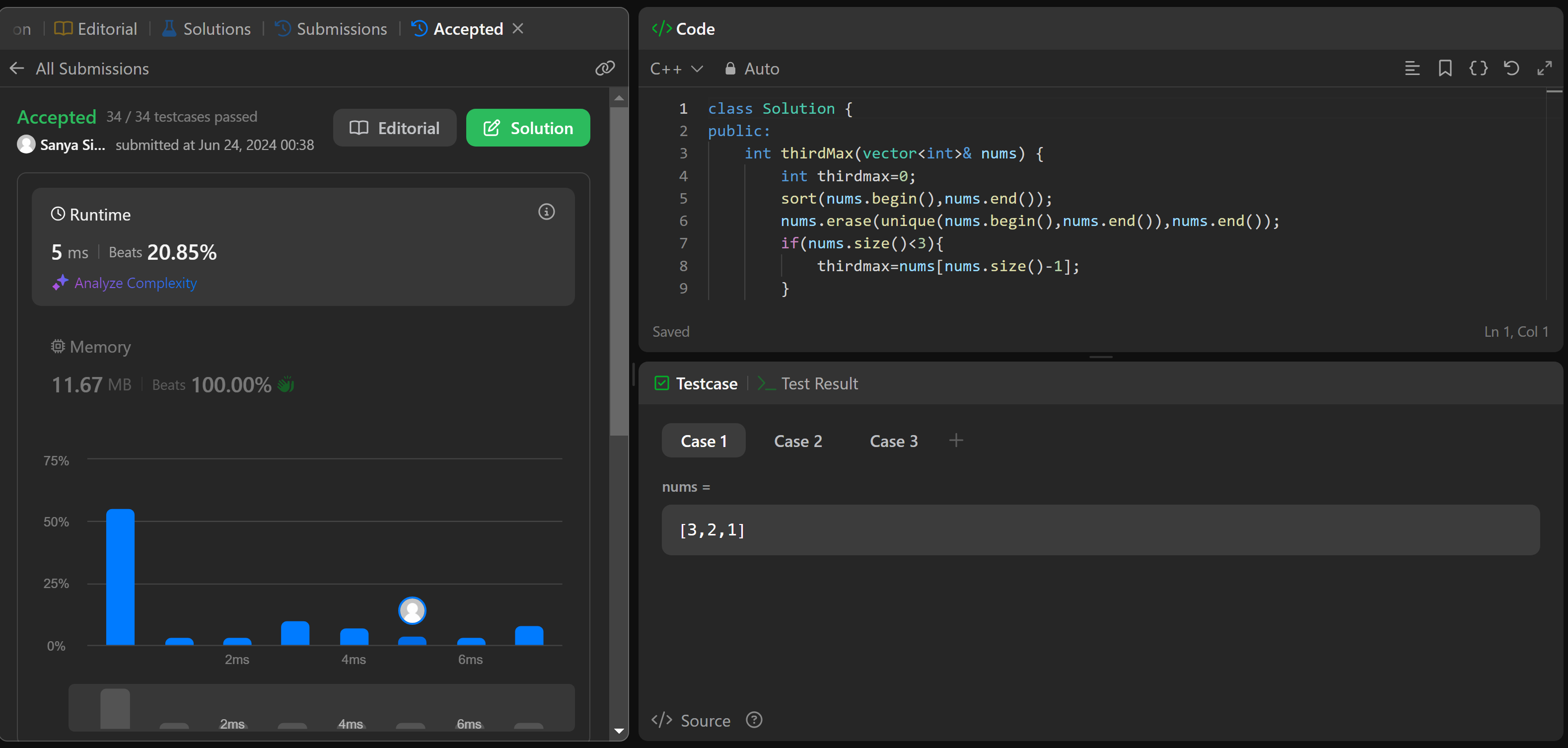
            thirdmax=nums[nums.size()-3];

        }

        return thirdmax;

    }

};

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

class Solution {

public:

    string frequencySort(string s) {

        map<char,int> freq;

        for(int i=0;i<s.size();i++){

            freq[s[i]]++;

        }

        string ans="";

        vector<pair<int,char>> freqvec;

        for(auto i:freq){

            freqvec.push\_back({i.second,i.first});

        }

        sort(freqvec.begin(),freqvec.end(),[](pair<int,char> &a,pair<int,char> &b){

            return a.first>b.first;

        });

        for(auto i:freqvec){

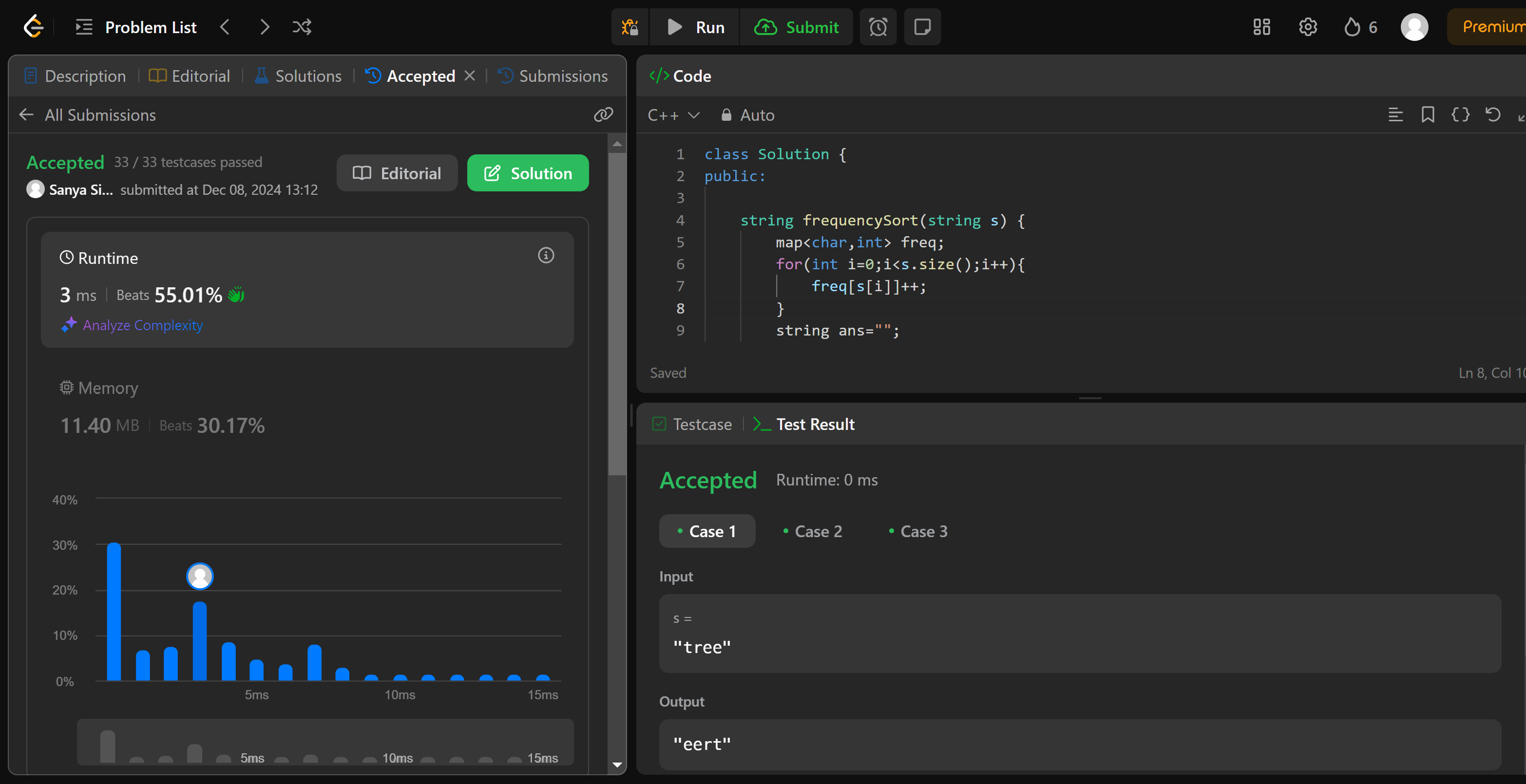
            ans.append(i.first, i.second);

        }

        return ans;

    }

};

****

**452.**[**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) {

        std::sort(points.begin(), points.end(), [](const auto& a, const auto& b) {

            return a[0] < b[0];

        });

        int arrows = 1;

        int end = points[0][1];

        for (size\_t i = 1; i < points.size(); ++i) {

            if (points[i][0] > end) {

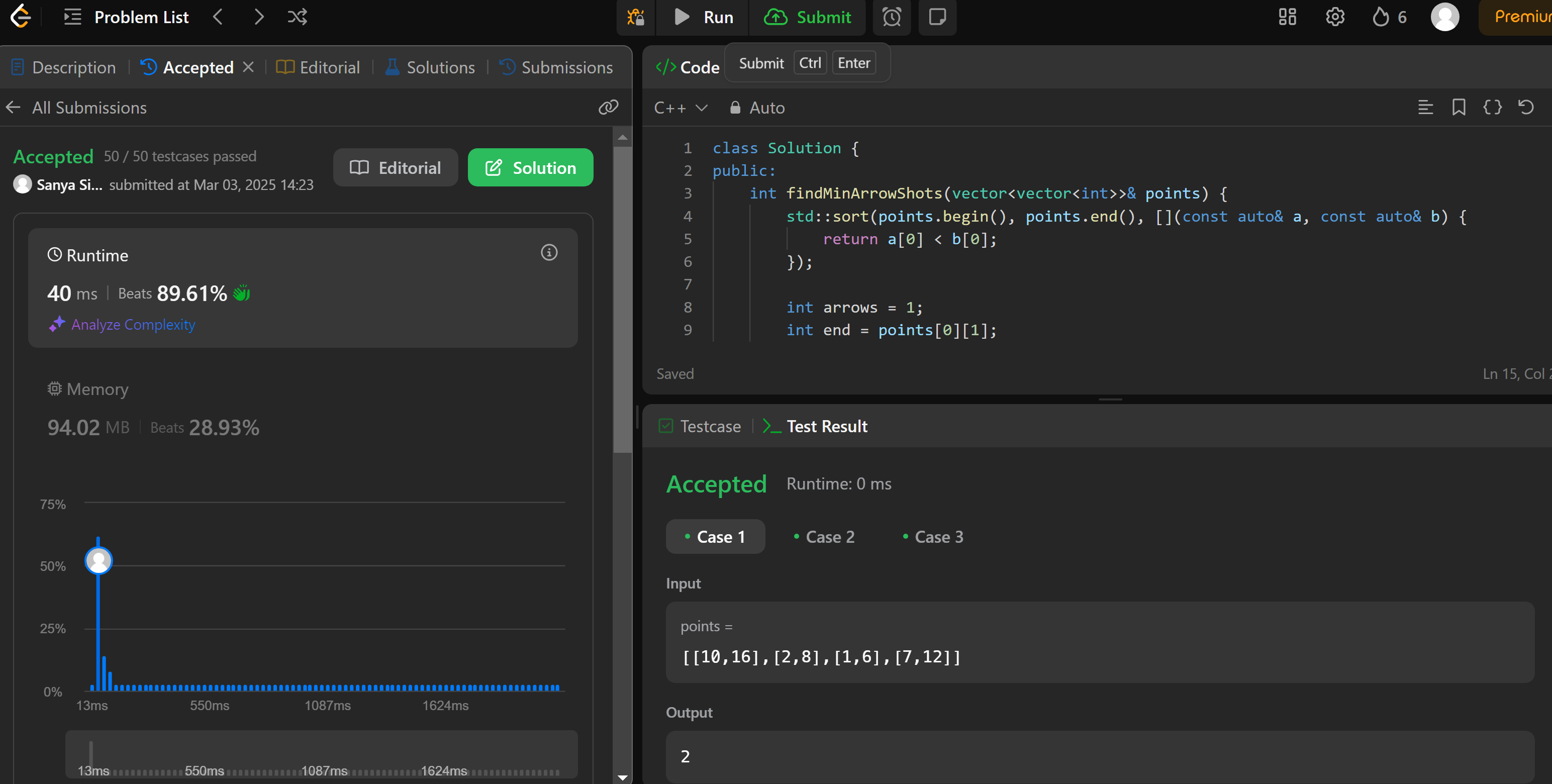
                arrows++;

                end = points[i][1];

            } else {

                end = std::min(end, points[i][1]);}}

        return arrows;}};

****

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

class Solution {

public:

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

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

        int left=0;

        int right=people.size()-1;

        int count=0;

        int sum=0;

        while(left<=right){

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

                left++;

            }

            right--;

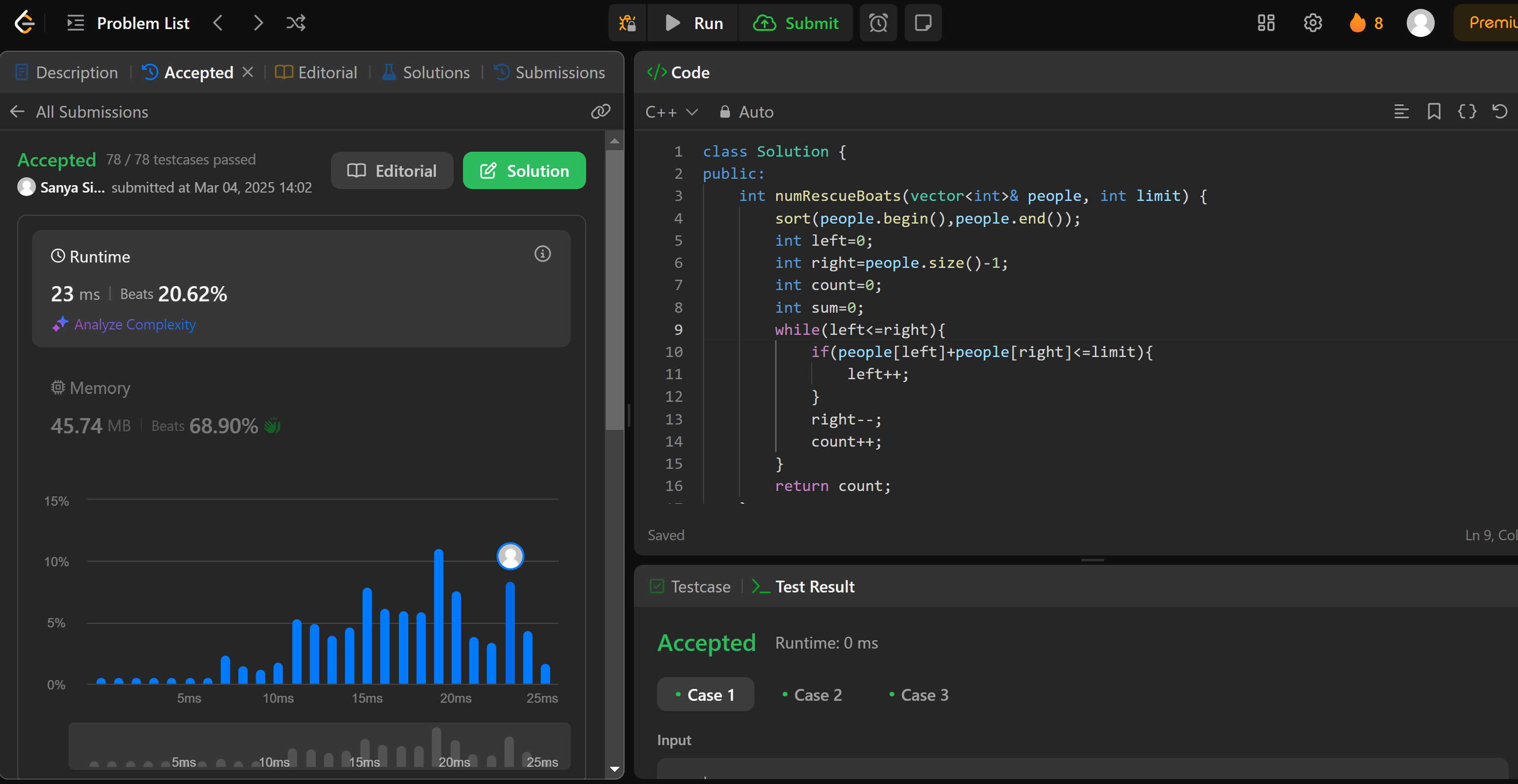
            count++;

        }

        return count;

    }

};

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

class Solution {

public:

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

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

        for (auto& point : points) {

            int distance = point[0] \* point[0] + point[1] \* point[1];

            maxHeap.push({distance, point});

            if (maxHeap.size() > k) maxHeap.pop();

        }

vector<vector<int>> ans;

        while (!maxHeap.empty()) {

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

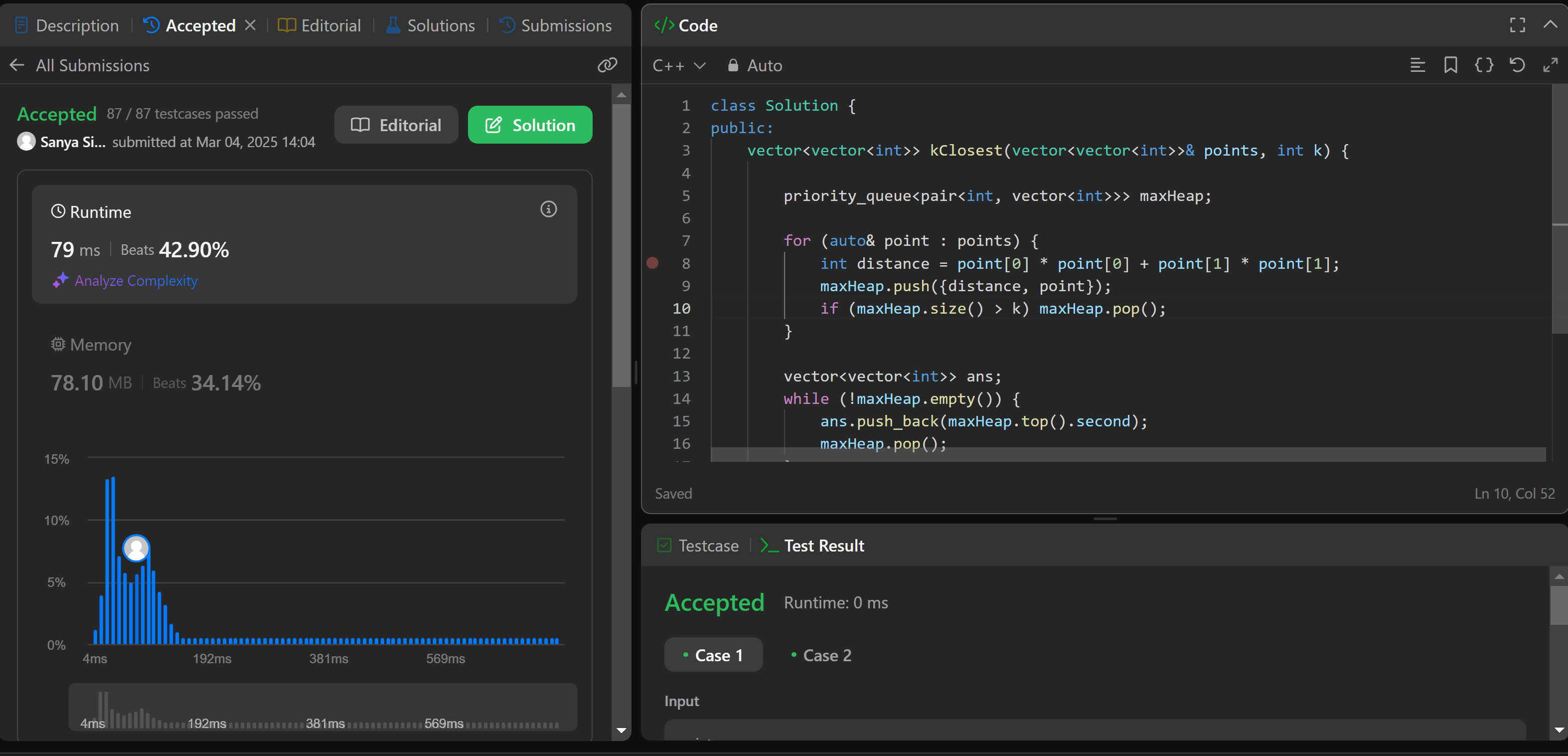
            maxHeap.pop();

        }

        return ans;

    }

};

****

**1338.**[**Reduce Array Size to The Half**](https://leetcode.com/problems/reduce-array-size-to-the-half/description/)

class Solution {

public:

    int minSetSize(vector<int>& arr) {

        int n = arr.size();

        unordered\_map<int, int> cnt;

        for (int x : arr) ++cnt[x];

        vector<int> counting(n + 1);

        for (auto [\_, freq] : cnt) ++counting[freq];

        int ans = 0, removed = 0, half = n / 2, freq = n;

        while (removed < half) {

            ans += 1;

            while (counting[freq] == 0) --freq;

            removed += freq;

            --counting[freq];

        }

        return ans;}};

