**Assignment 5**

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

**Problem 1: Find the Difference (**<https://leetcode.com/problems/find-the-difference/submissions/> **)**

**Code:**class Solution {

public:

    char findTheDifference(string s, string t) {

        unordered\_map<char,int>mpp;

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

            mpp[t[i]]++;

        }

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

            mpp[s[i]]--;

        }

        for(auto it:mpp){

            if(it.second>0){

                return it.first;

            }

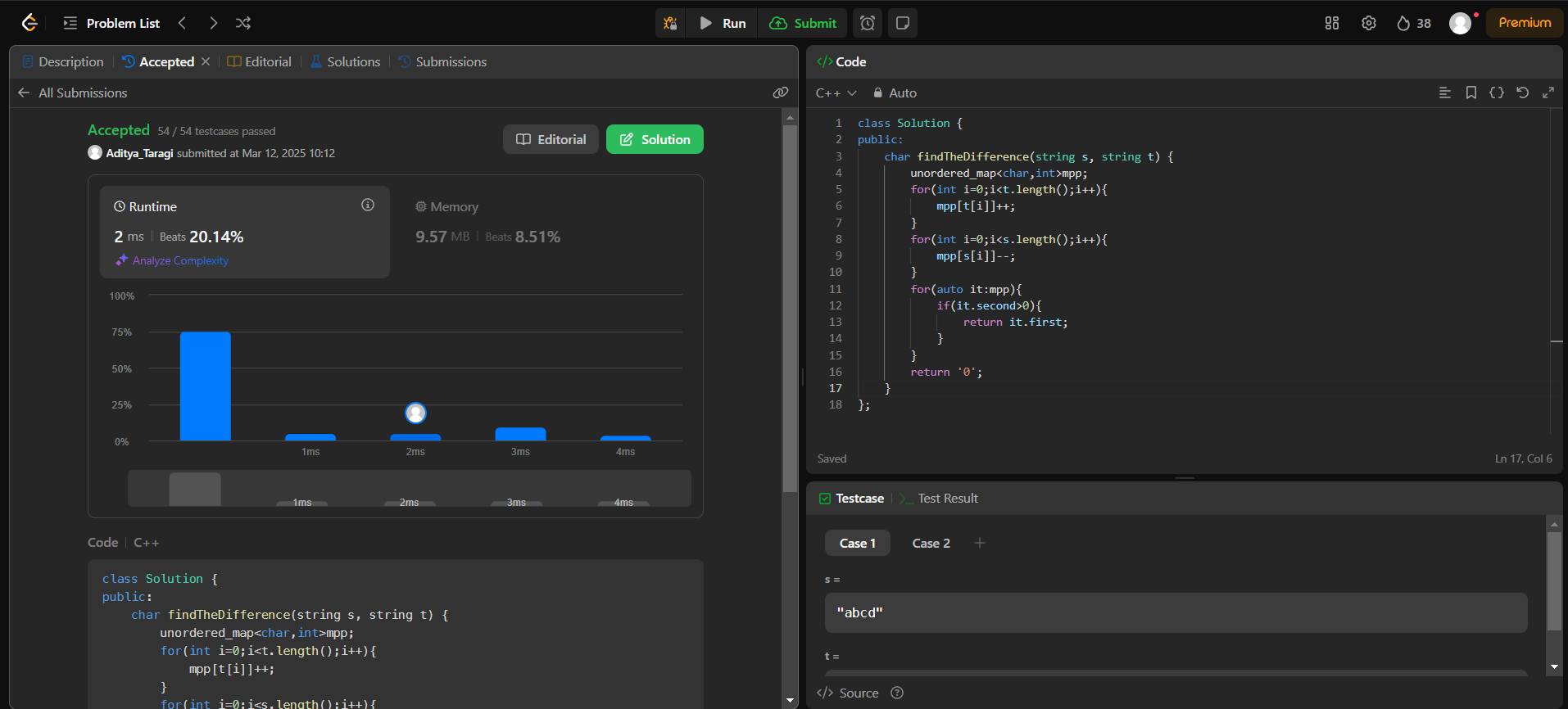
        }

        return '0';

    }

};

**Screenshot:**

****

**Problem 2: Largest Perimeter Triangle (**<https://leetcode.com/problems/largest-perimeter-triangle/> )

**Code:**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];

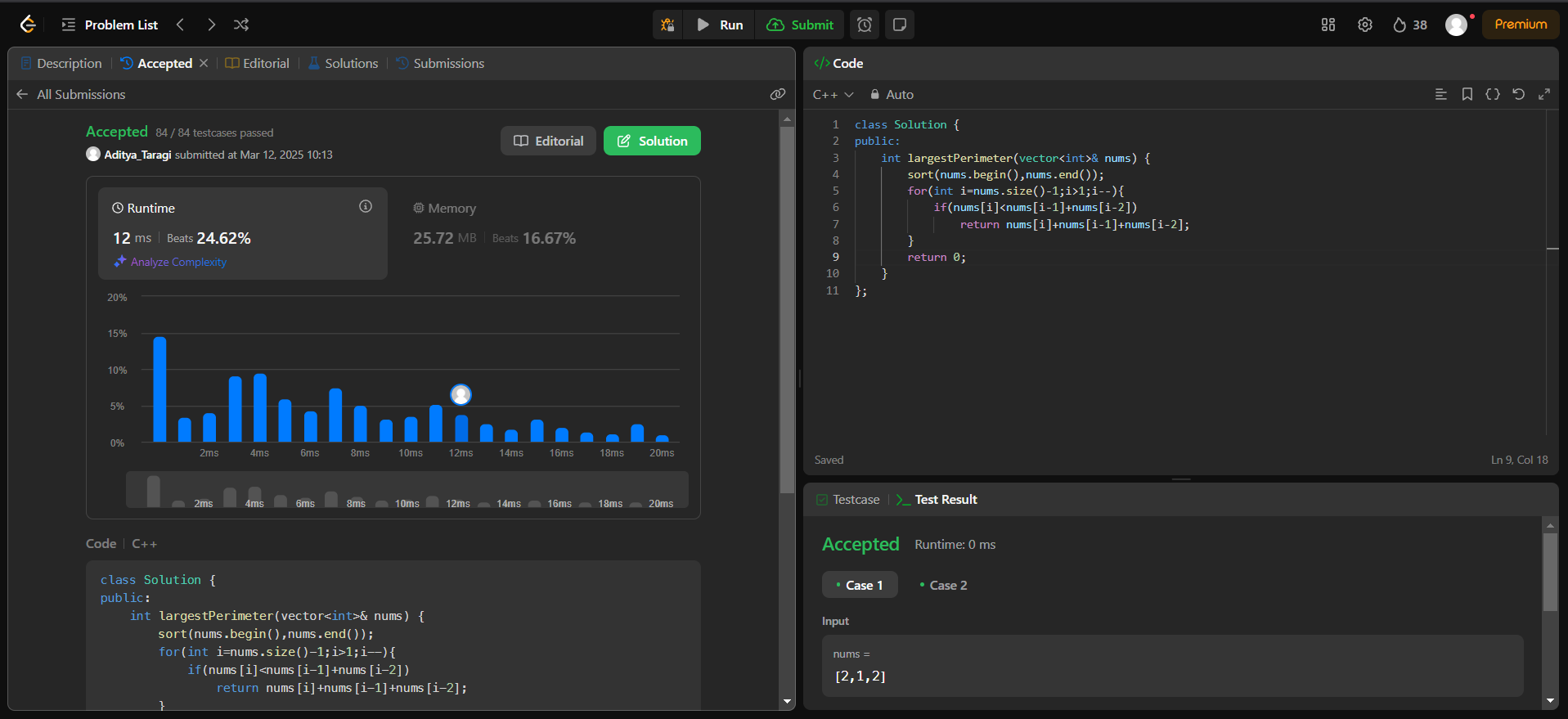
        }

        return 0;

    }

};

**Screenshot:**



**Problem 3: Third Maximum Number (**<https://leetcode.com/problems/third-maximum-number/> **)**

**Code:**class Solution {

public:

    int thirdMax(vector<int>& nums) {

        set<int>s;

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

            s.insert(nums[i]);

        }

        if(s.size()>=3){

            int Third\_index\_from\_last=s.size()-3;

            auto third\_maximum=next(s.begin(),Third\_index\_from\_last);

            return \*third\_maximum;

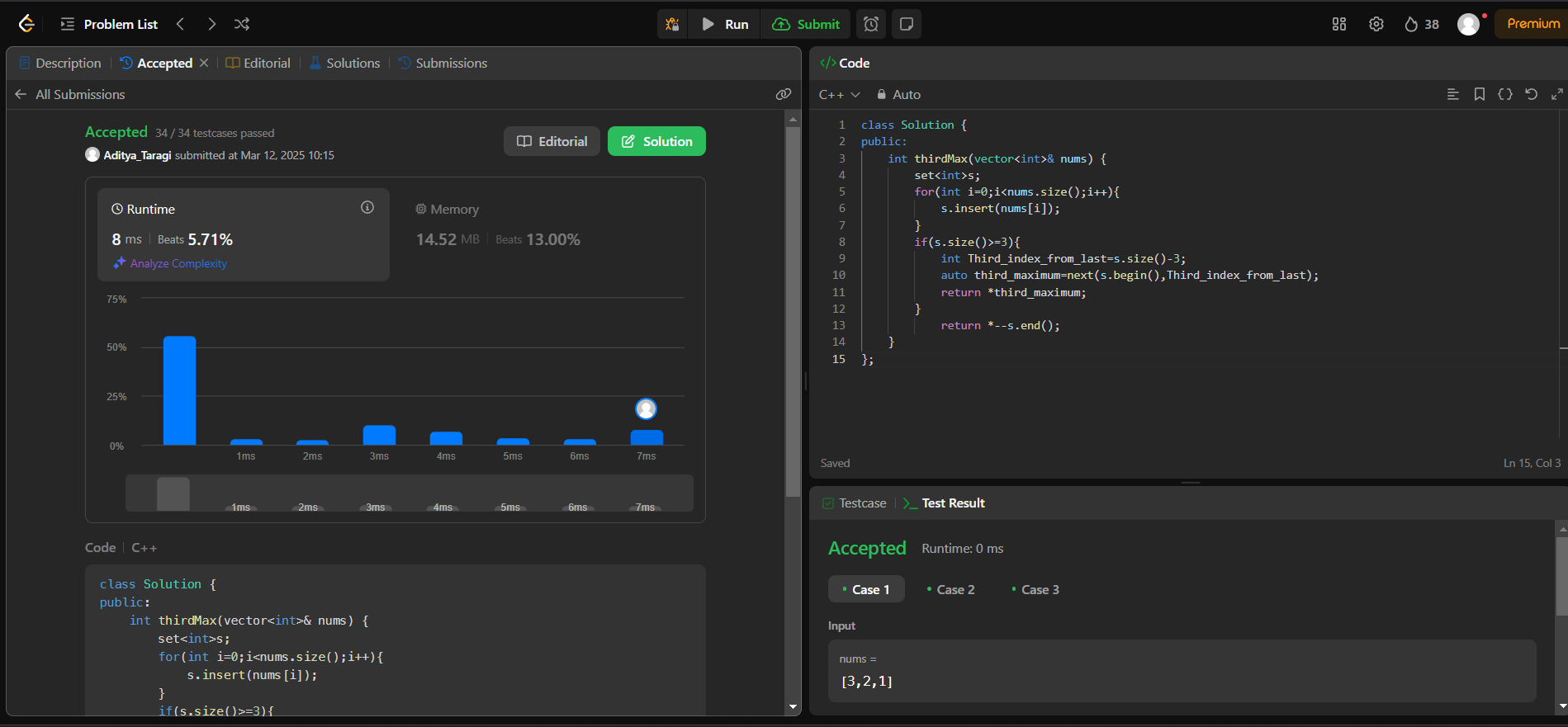
        }

            return \*--s.end();

    }

};

**Screenshot:**



**Problem 4: Sort Characters by Frequency (**<https://leetcode.com/problems/sort-characters-by-frequency/> **)**

**Code:**class Solution {

public:

    string frequencySort(string s) {

        auto cmp = [](const pair<char, int>& a, const pair<char, int>& b) {

            return a.second < b.second;

        };

        priority\_queue<pair<char, int>, vector<pair<char, int>>, decltype(cmp)> pq(cmp);

        unordered\_map<char, int> hm;

        for (char c : s) {

            hm[c]++;

        }

        for (const auto& entry : hm) {

            pq.push(make\_pair(entry.first, entry.second));

        }

        string result = "";

        while (!pq.empty()) {

            pair<char, int> p = pq.top();

            pq.pop();

            result.append(p.second, p.first);

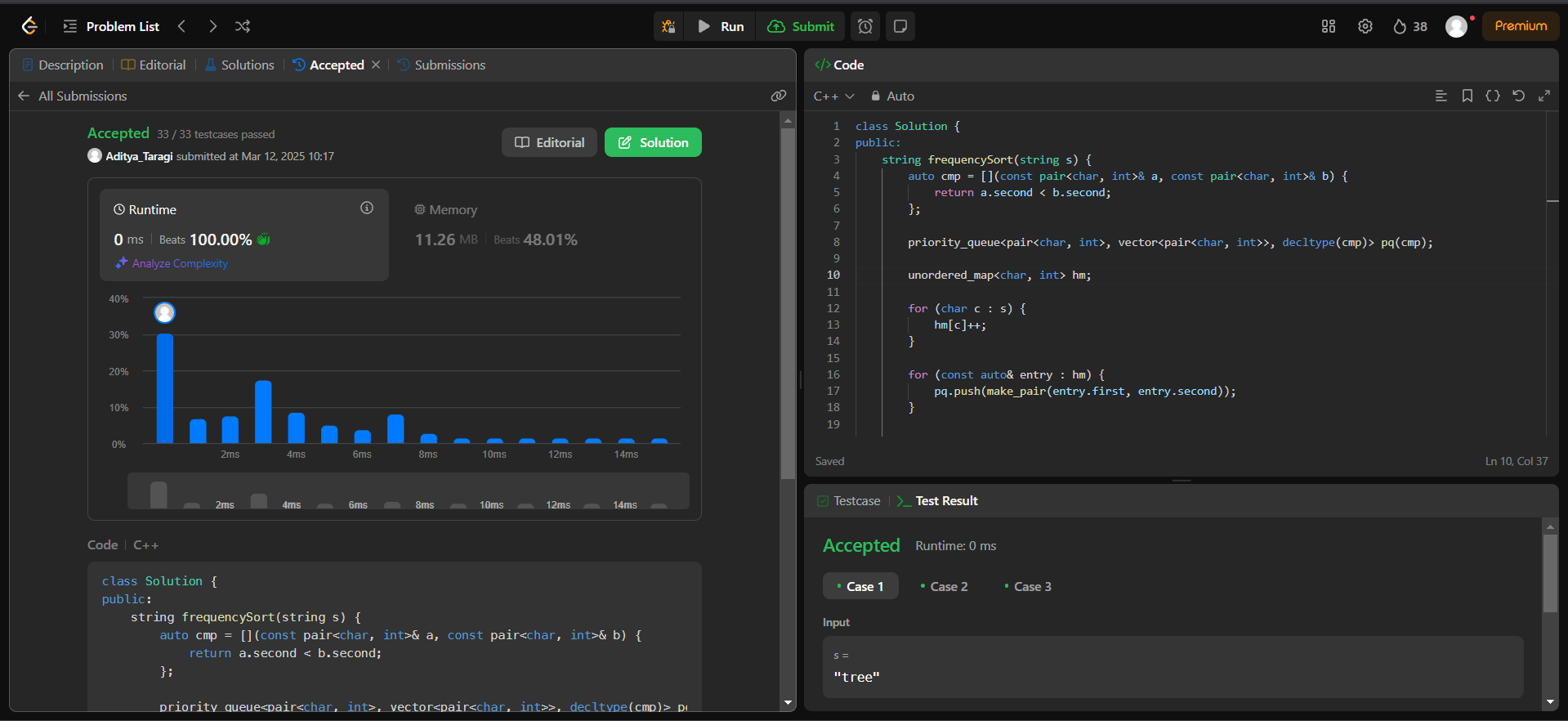
        }

        return result;

    }

};

**Screenshot:**



**Problem 5: Minimum Number of Arrows to Burst Balloons (**<https://leetcode.com/problems/minimum-number-of-arrows-to-burst-balloons/> **)**

**Code:**class Solution {

public:

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

        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]);

            }

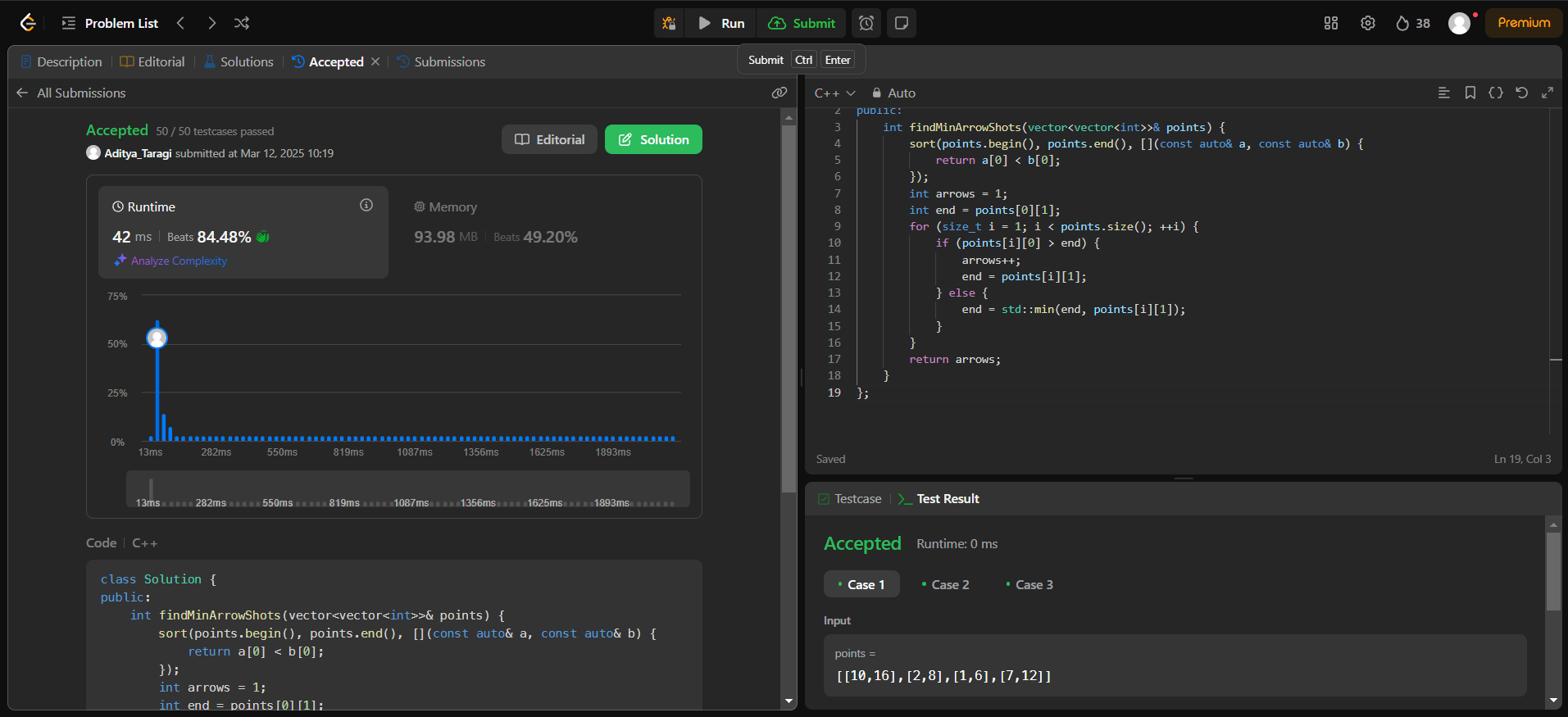
        }

        return arrows;

    }

};

**Screenshot:**

**Problem 6: Boats to Save People (**<https://leetcode.com/problems/boats-to-save-people/> **)**

**Code:**class Solution {

public:

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

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

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

        while(i <= j)

        {

            if(people[i] + people[j] <= limit)

            {

                ++i;

                --j;

            }

            else

                --j;

            ++n;

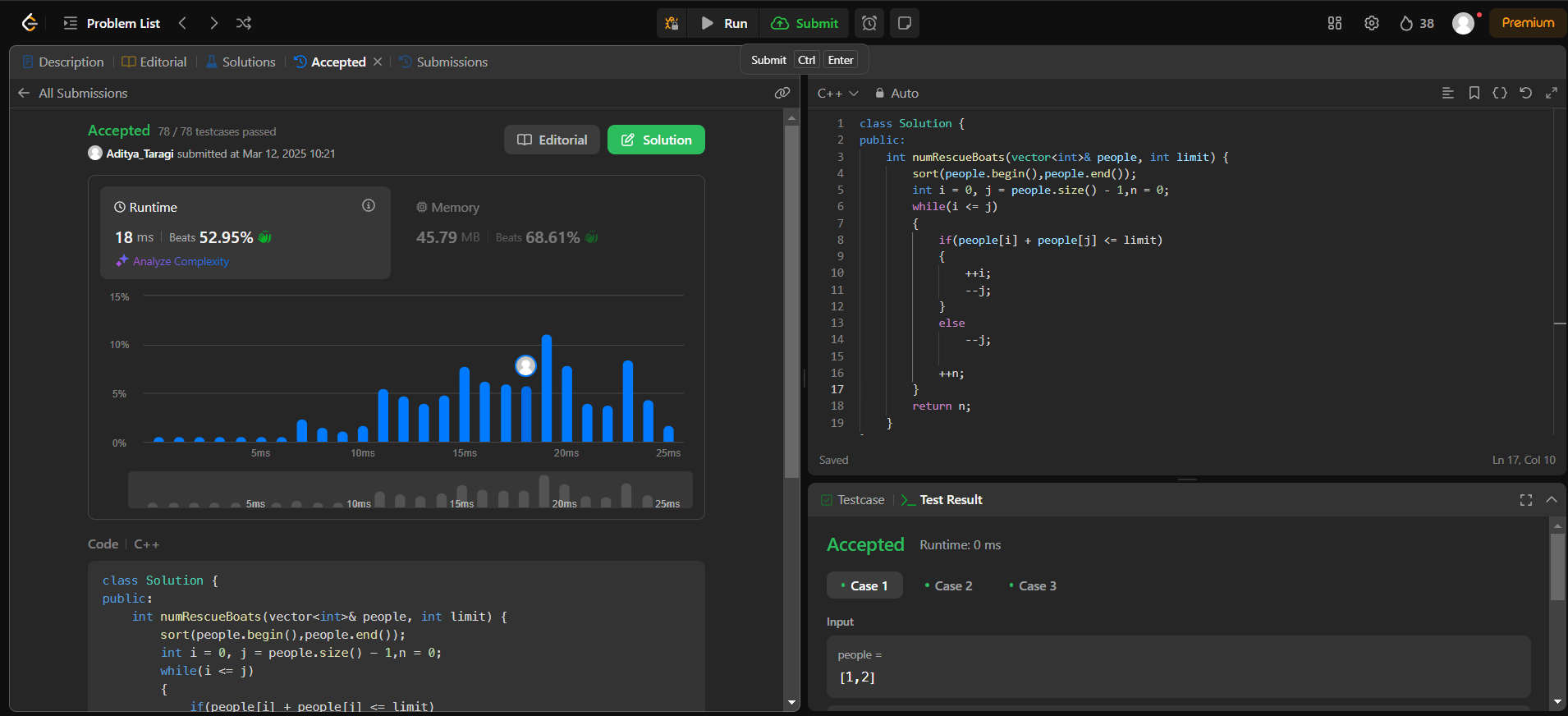
        }

        return n;

    }

};

**Screenshot:**



**Problem 7: k-closest Point to Origin (**<https://leetcode.com/problems/k-closest-points-to-origin/> **)**

**Code:**class Solution {

public:

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

        vector<vector<int>> result(k);

        priority\_queue<vector<int>> maxHeap;

        for (auto& p : points) {

            int x = p[0], y = p[1];

            maxHeap.push({x\*x + y\*y, x, y});

            if (maxHeap.size() > k) {

                maxHeap.pop();

            }

        }

        for (int i = 0; i < k; ++i) {

            vector<int> top = maxHeap.top();

            maxHeap.pop();

            result[i] = {top[1], top[2]};

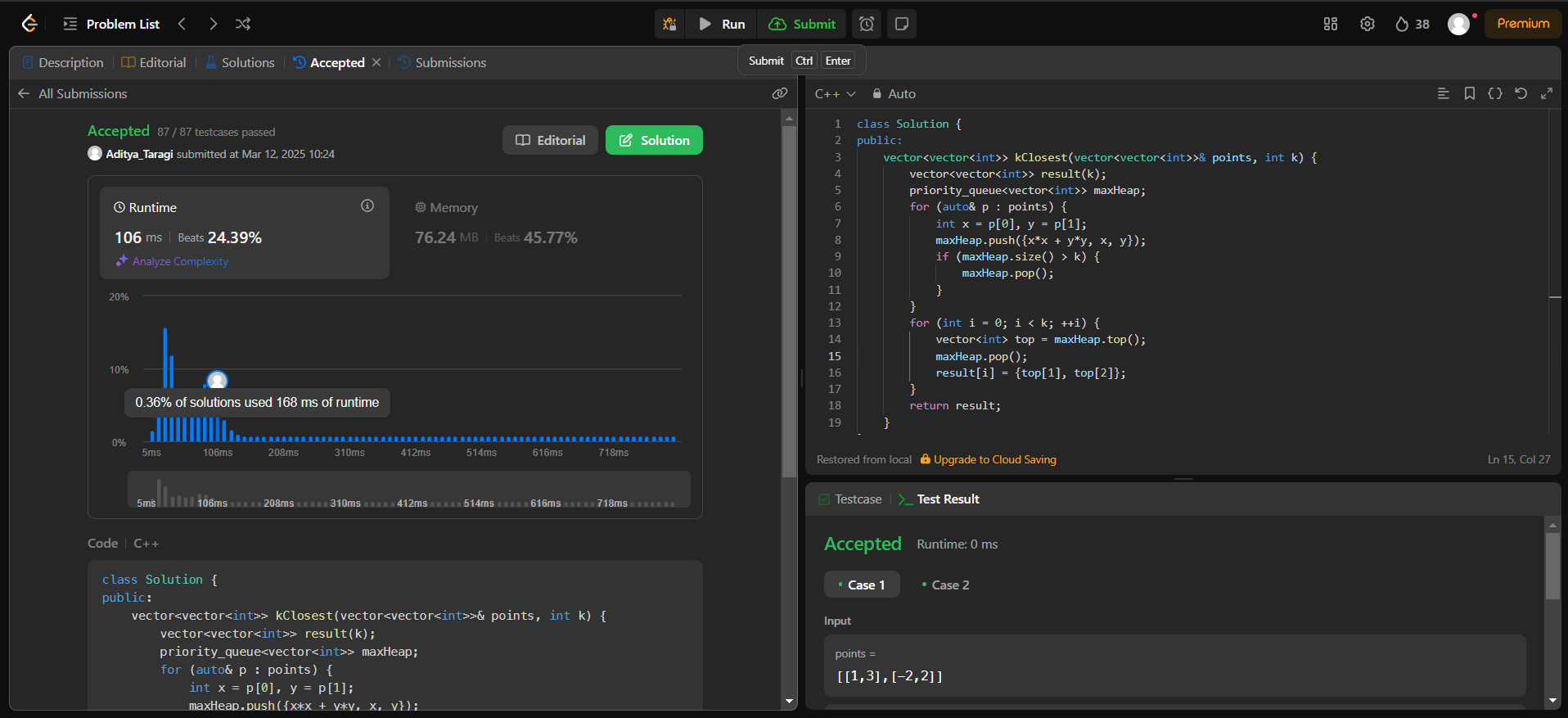
        }

        return result;

    }

};

**Screenshot:**



**Problem 8: Reduce Array Size to the Half (**<https://leetcode.com/problems/reduce-array-size-to-the-half/> **)**

**Code:**class Solution {

public:

    int minSetSize(vector<int>& arr) {

        map<int,int>mp;

        for(auto val:arr) mp[val]++;

        priority\_queue<int>pq;

        for(auto [val, cnt]:mp)

            pq.push(cnt);

        int ans = 0, need = arr.size()/2;

        while(need > 0)

        {

           ans++;

           need -= pq.top(); pq.pop();

        }

        return ans;

    }

};

**Screenshot:**

