# NAME- Akshit Boparai | UID- 22BCS14939 | SECTION- 601/A

# 1 FIND THE DIFFERENCE

class Solution {

public:

    char findTheDifference(string s, string t) {

        char result = 0;

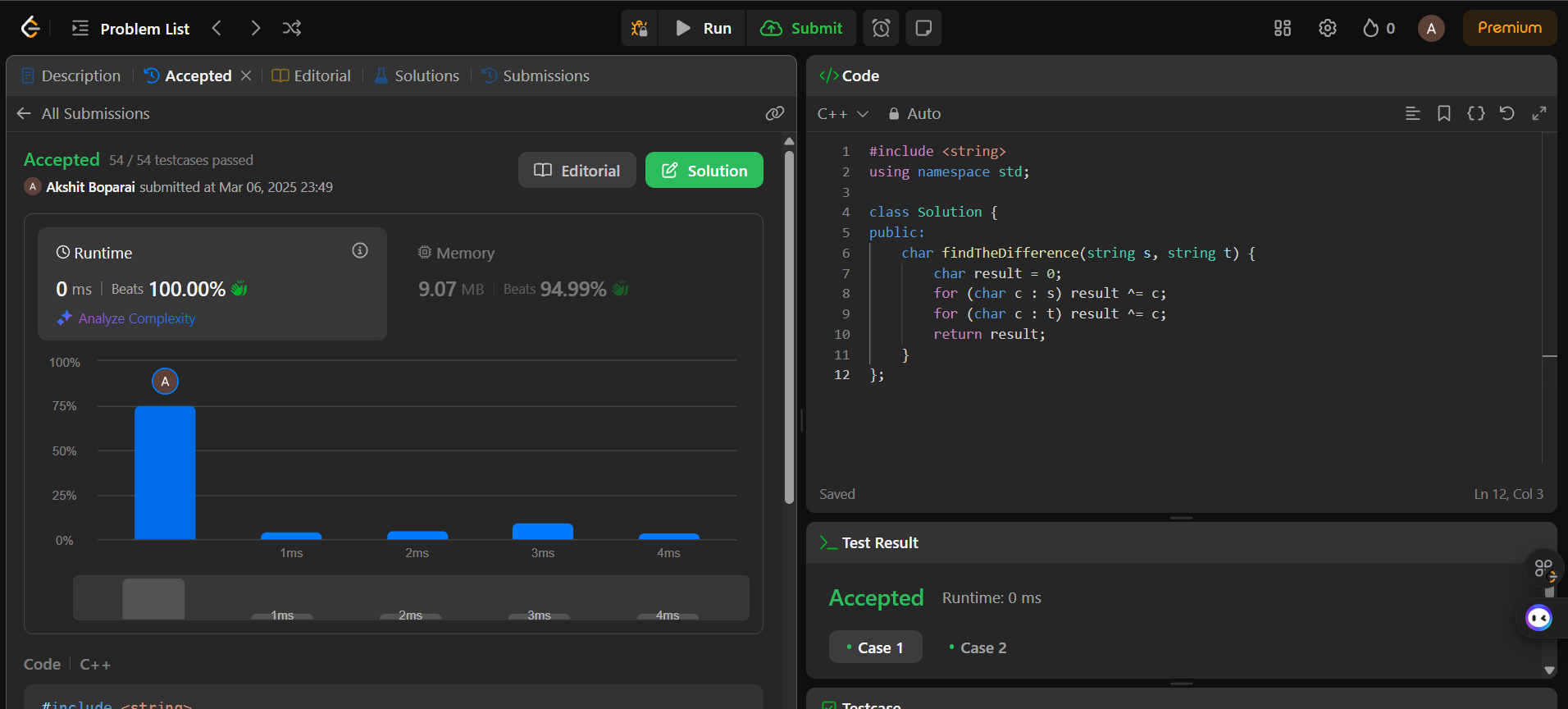
        for (char c : s) result ^= c;

        for (char c : t) result ^= c;

        return result;

    }

};



# 2. 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];

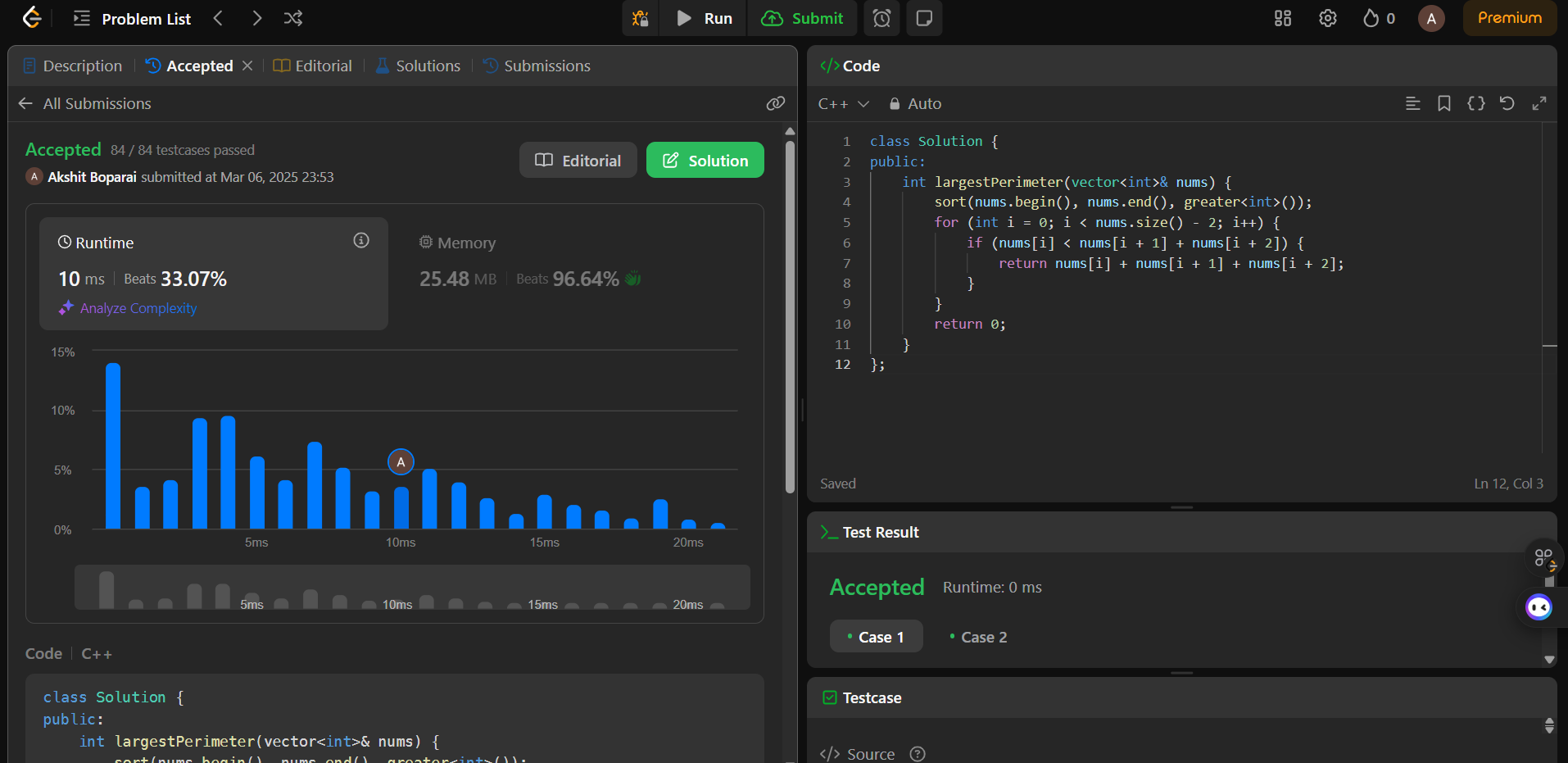
            }

        }

        return 0;

    }

};



# 3 THIRD MAXIMUM NUMBER

# class Solution {

# public:

# int thirdMax(vector<int>& nums) {

# set<int> uniqueNums(nums.begin(), nums.end());

# if (uniqueNums.size() < 3) {

# return \*uniqueNums.rbegin();

# }

# auto it = uniqueNums.end();

# advance(it, -3);

# return \*it;

# }

# };

# 

# SORT CHARACTER BY FREQUENCY

# class Solution {

# public:

# string frequencySort(string s) {

# unordered\_map<char, int> freq;

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

# sort(s.begin(), s.end(), [&](char a, char b) {

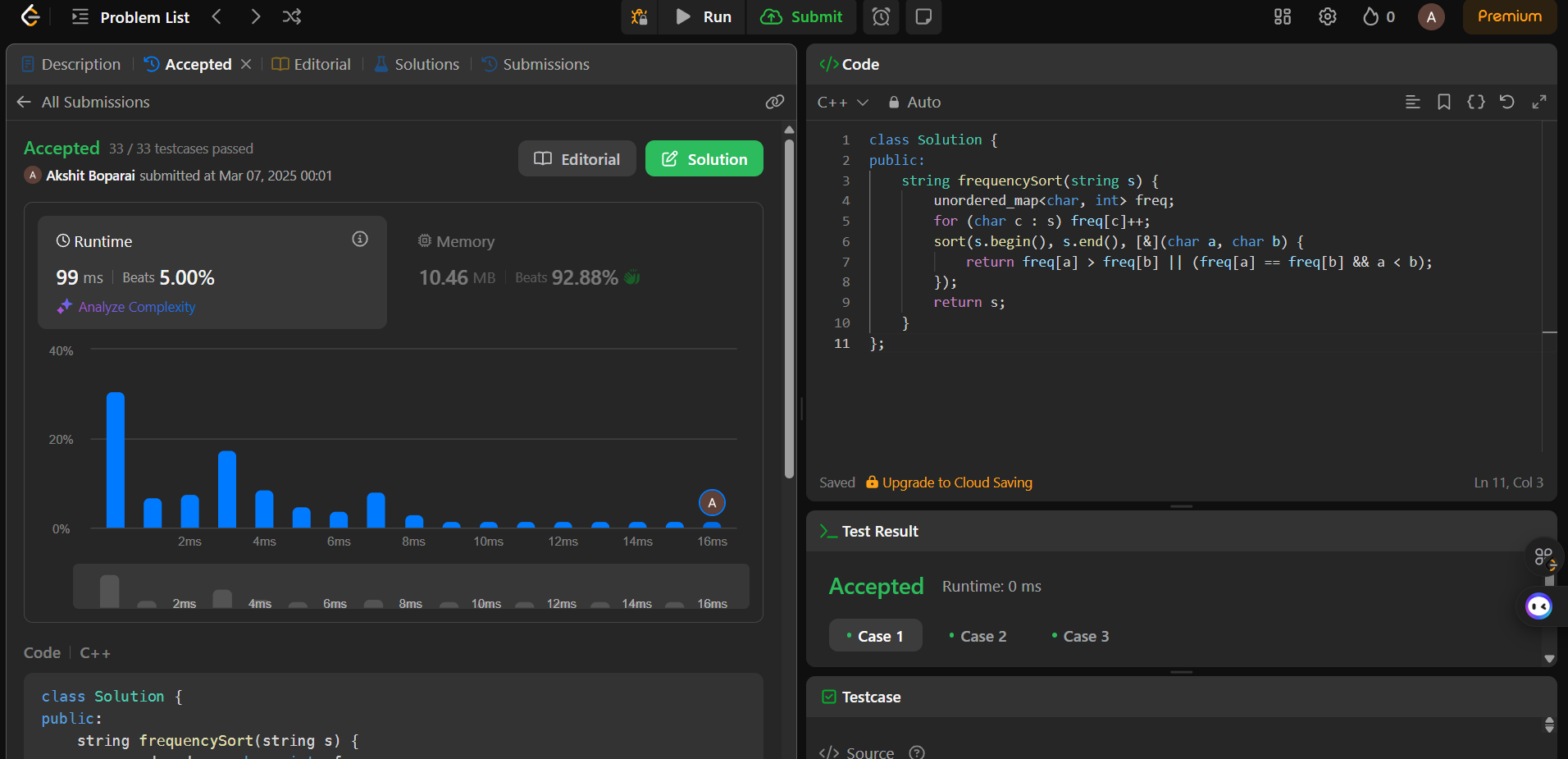
# return freq[a] > freq[b] || (freq[a] == freq[b] && a < b);

# });

# return s;

# }

# };



# 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& p : points) {

            if (p[0] > end) {

                arrows++;

                end = p[1];

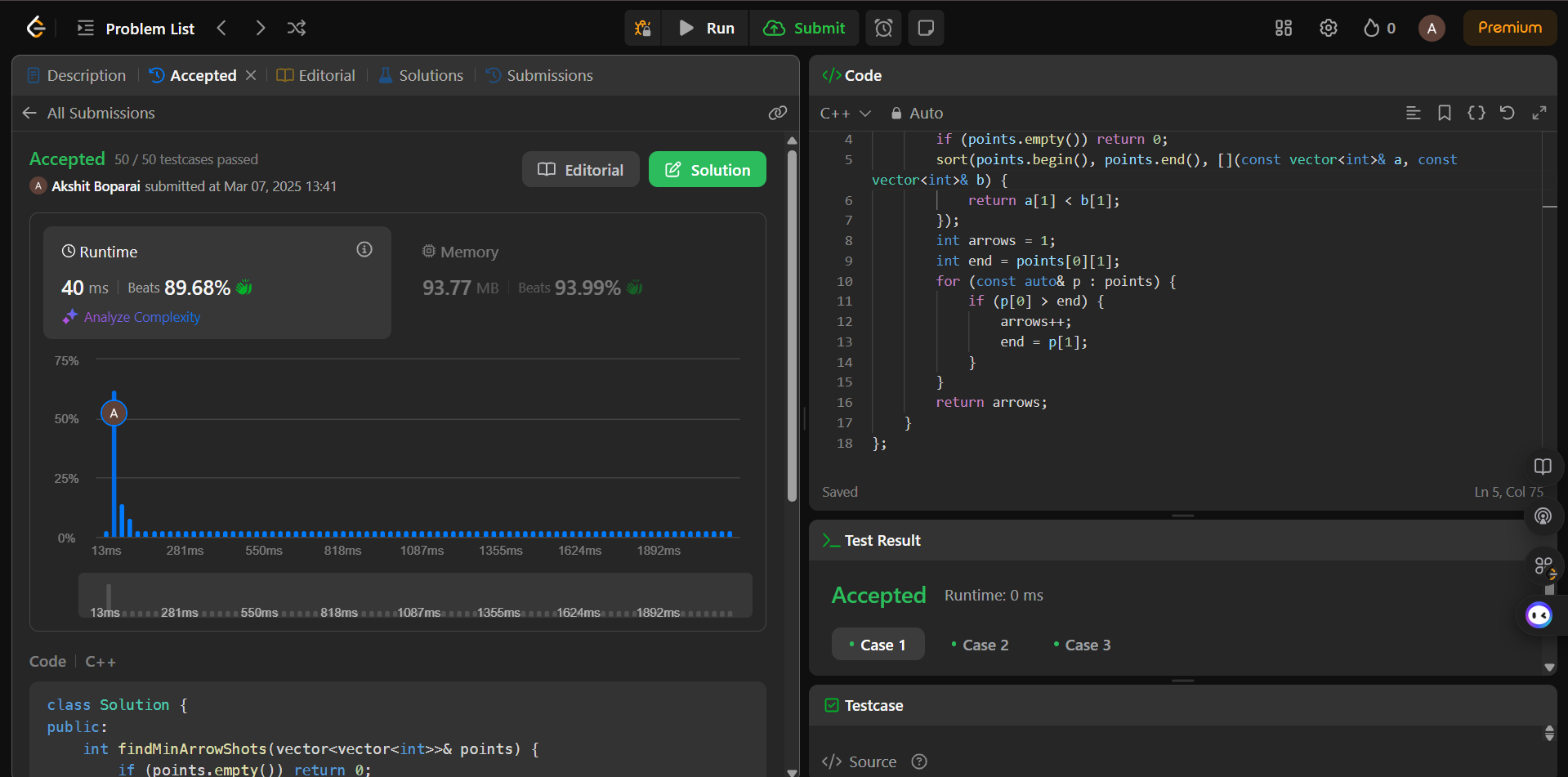
            }

        }

        return arrows;

    }

};



## 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--;

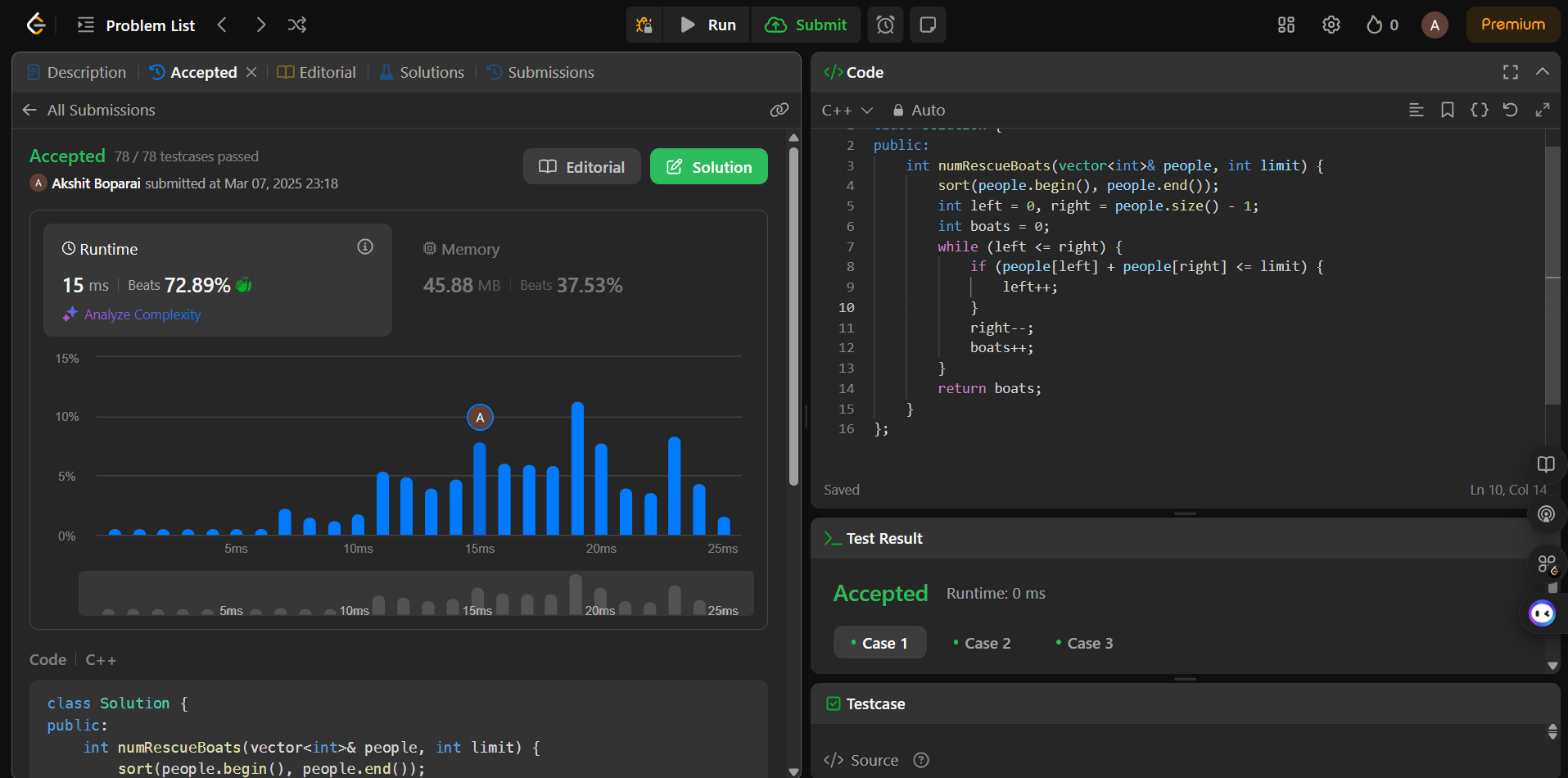
            boats++;

        }

        return boats;

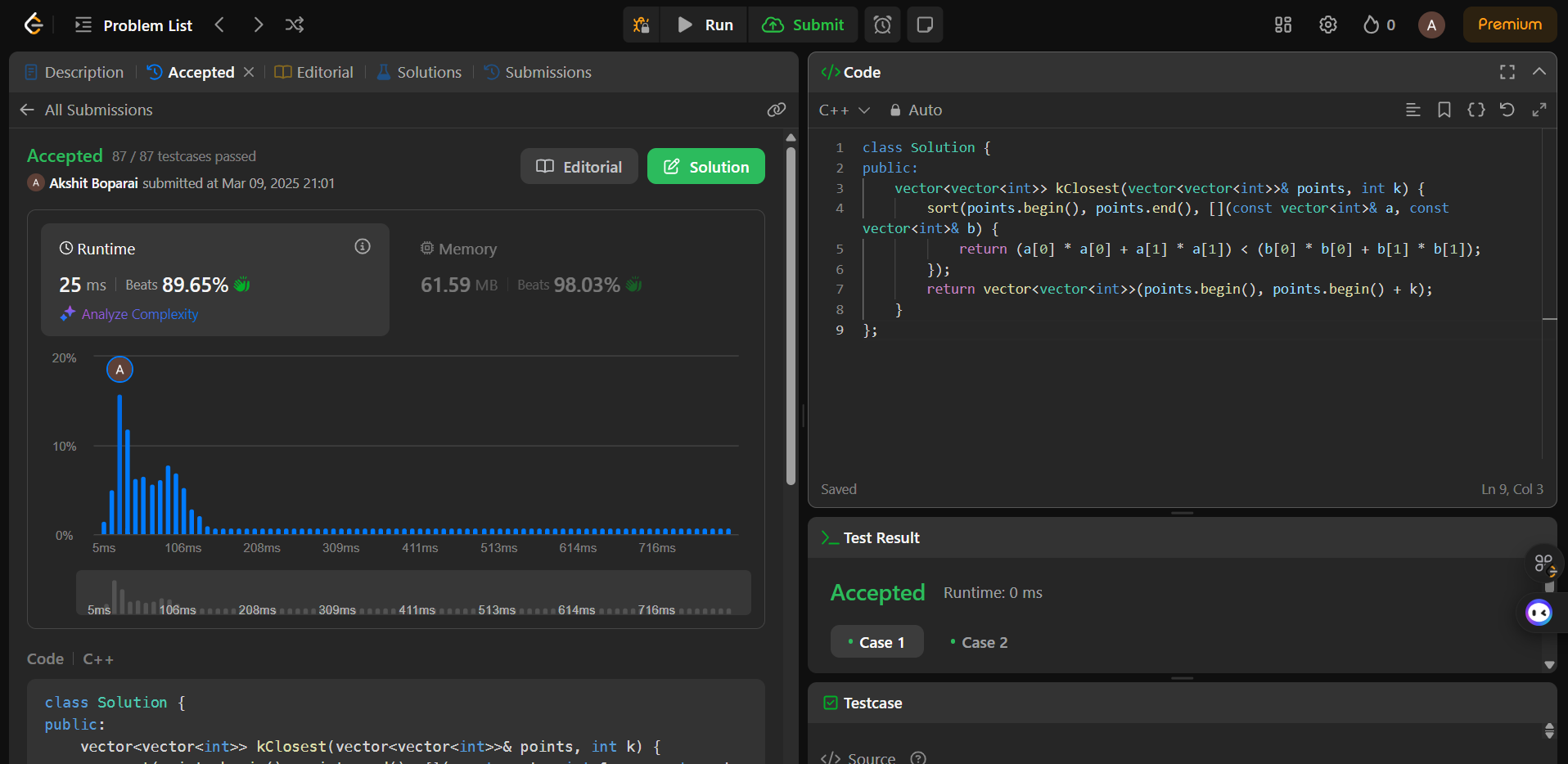
    }

};



## K CLOSEST POINT TO ORIGIN

class Solution {public: vector<vector<int>> kClosest(vector<vector<int>>& points, int k) { sort(points.begin(), points.end(), [](const vector<int>& a, const vector<int>& b) { return (a[0] \* a[0] + a[1] \* a[1]) < (b[0] \* b[0] + b[1] \* b[1]); }); return vector<vector<int>>(points.begin(), points.begin() + k); }};



## REDUCE ARRAY SIZE TO HALF

class Solution {public: int minSetSize(vector<int>& arr) { unordered\_map<int, int> freq; for (int num : arr) freq[num]++; vector<int> counts; for (auto& pair : freq) counts.push\_back(pair.second); sort(counts.begin(), counts.end(), greater<int>()); int total = 0, result = 0; for (int count : counts) { total += count; result++; if (total >= arr.size() / 2) break; } return result; }};

