**389. Find the Difference**

class Solution {

public:

    char findTheDifference(string s, string t) {

        char result = 0;

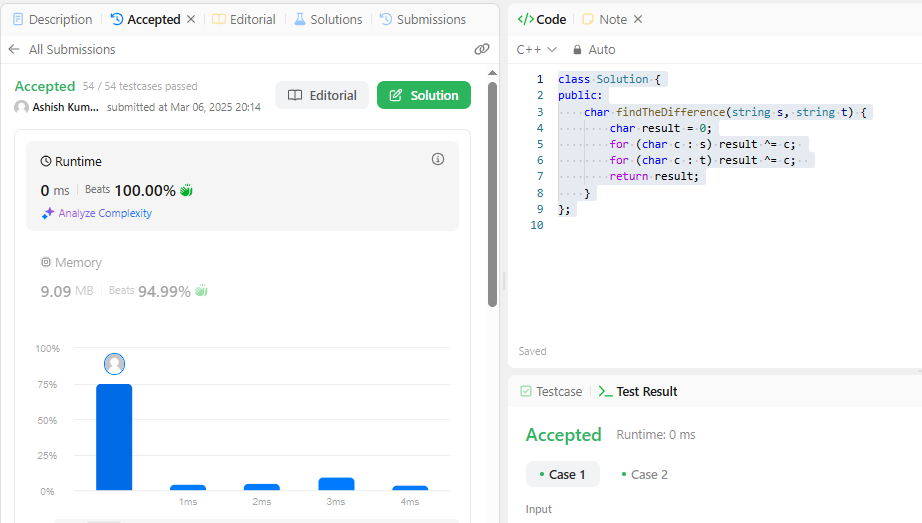
        for (char c : s) result ^= c;

        for (char c : t) result ^= c;

        return result;

    }

};



**976. 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];

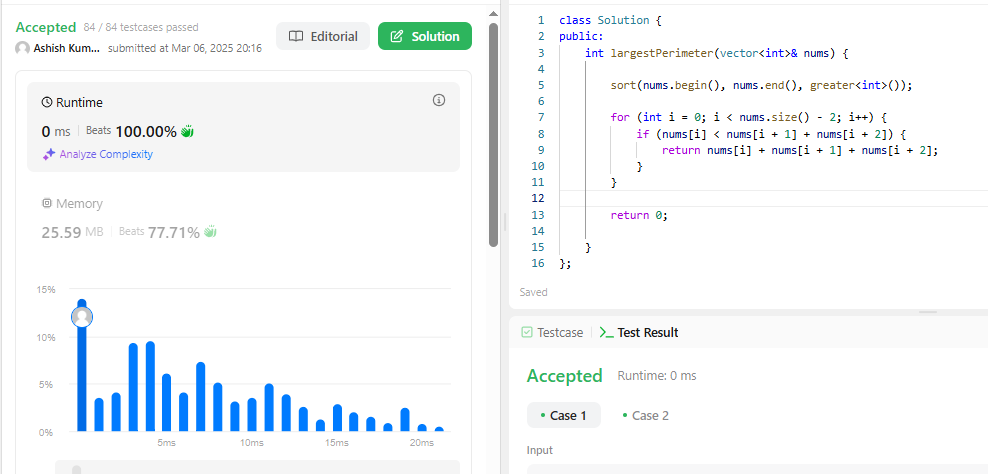
            }

        }

        return 0;

    }

};



**414. 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(first<num){

                third=second;

                second=first;

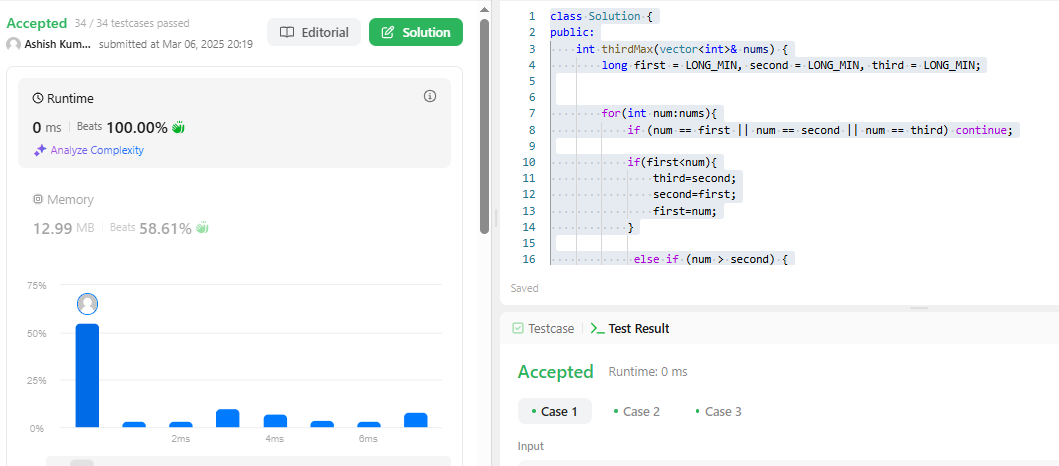
                first=num; }else if (num > second) {

                third = second;second = num;

            } else if (num > third) {

                third = num;  }}

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



**451. Sort Characters By Frequency**

class Solution {

public:

    string frequencySort(string s) {

        typedef pair<char, int> p;

        unordered\_map<char, int> mp;

        for (char c : s) {

            mp[c]++;

        }

        vector<p> lst;

        for (auto& p : mp) {

            lst.push\_back(p);

        }

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

            return (a.second == b.second) ? a.first < b.first : a.second > b.second;

        };

        sort(lst.begin(), lst.end(), compare);

        string result;

        for (auto& p : lst) {

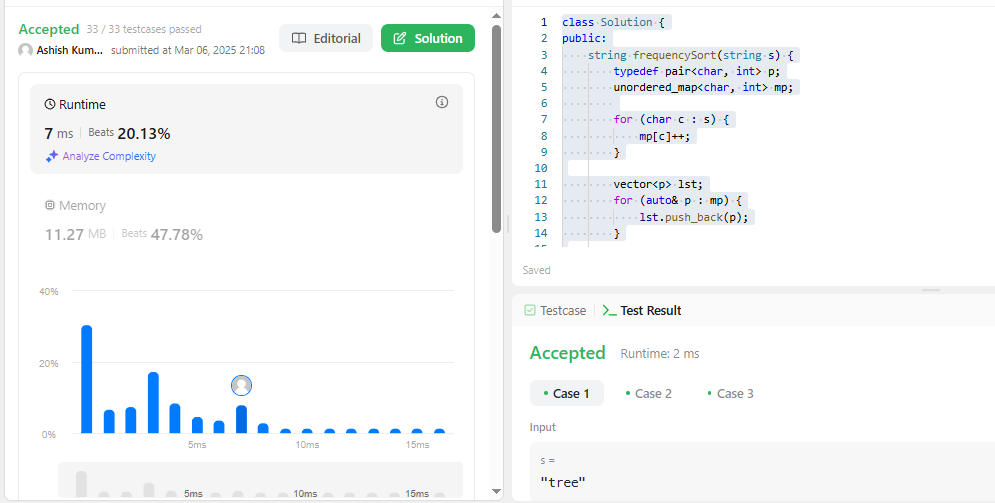
            result.append(p.second, p.first);

        }

        return result;

    }

};



**452. Minimum Number of Arrows to Burst Balloons**

class Solution {

public:

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

        int n=points.size();

        int arrow=1;

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

            return a[1]<b[1];

        });

        int prev=points[0][1];

         if (points.empty()) return 0;

        for(int i=0;i<n;i++){

            if(points[i][0]>prev){

                arrow++;

                prev=points[i][1];

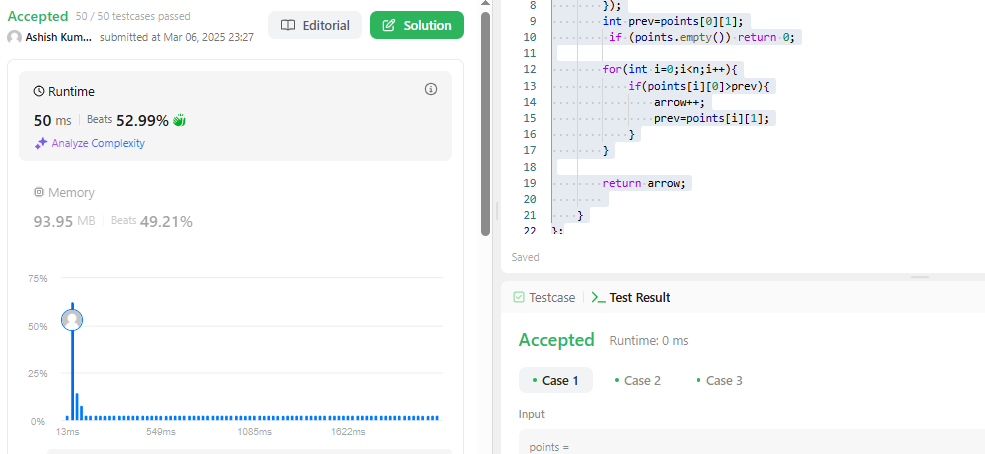
            }

        }

        return arrow;

    }

};



**881. 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, boats = 0;

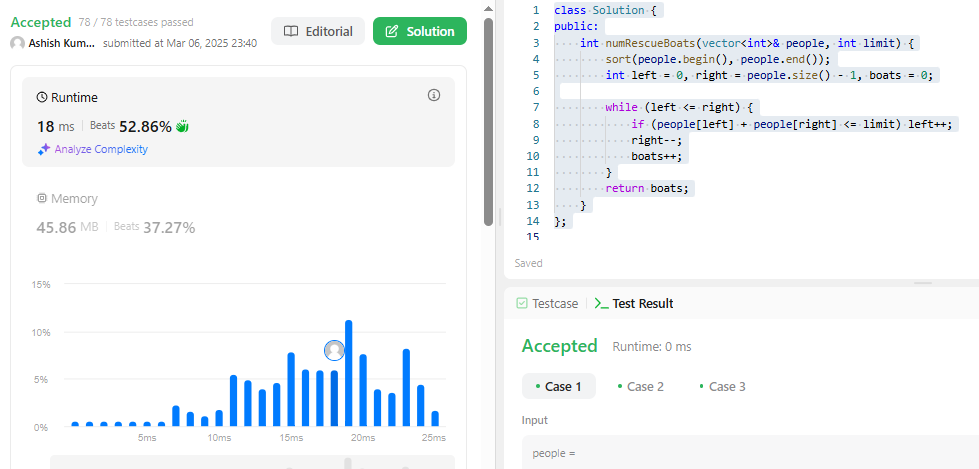
        while (left <= right) {

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

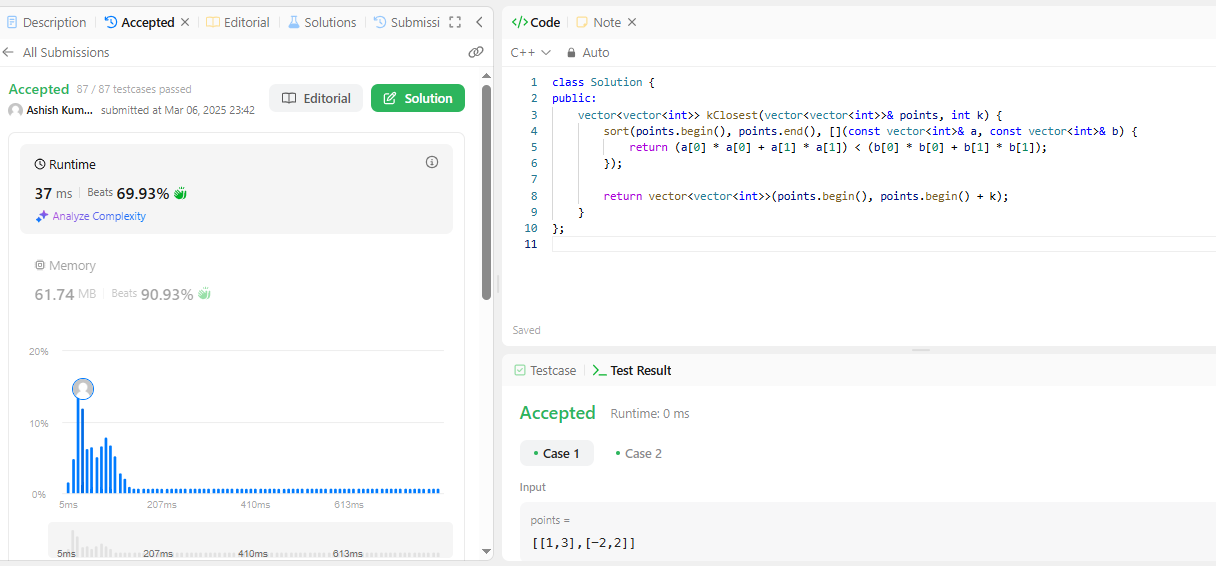
            right--;

            boats++;

        } return boats;}};



**973. K Closest Points to Origin**



**1338. 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]++;

        vector<int> counts;

        for (auto& p : freq) counts.push\_back(p.second);

        sort(counts.rbegin(), counts.rend());

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

        for (int count : counts) {

            removed += count;

            size++;

            if (removed >= half) break;

        }

        return size;

    }

};

