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

**CODE:**

class Solution {

public:

    char findTheDifference(string s, string t) {

        std::unordered\_map<char, int> count;

                for (char c : t) {

            count[c]++;

        }

              for (char c : s) {

            count[c]--;

            if (count[c] == 0) {

                count.erase(c);

            }

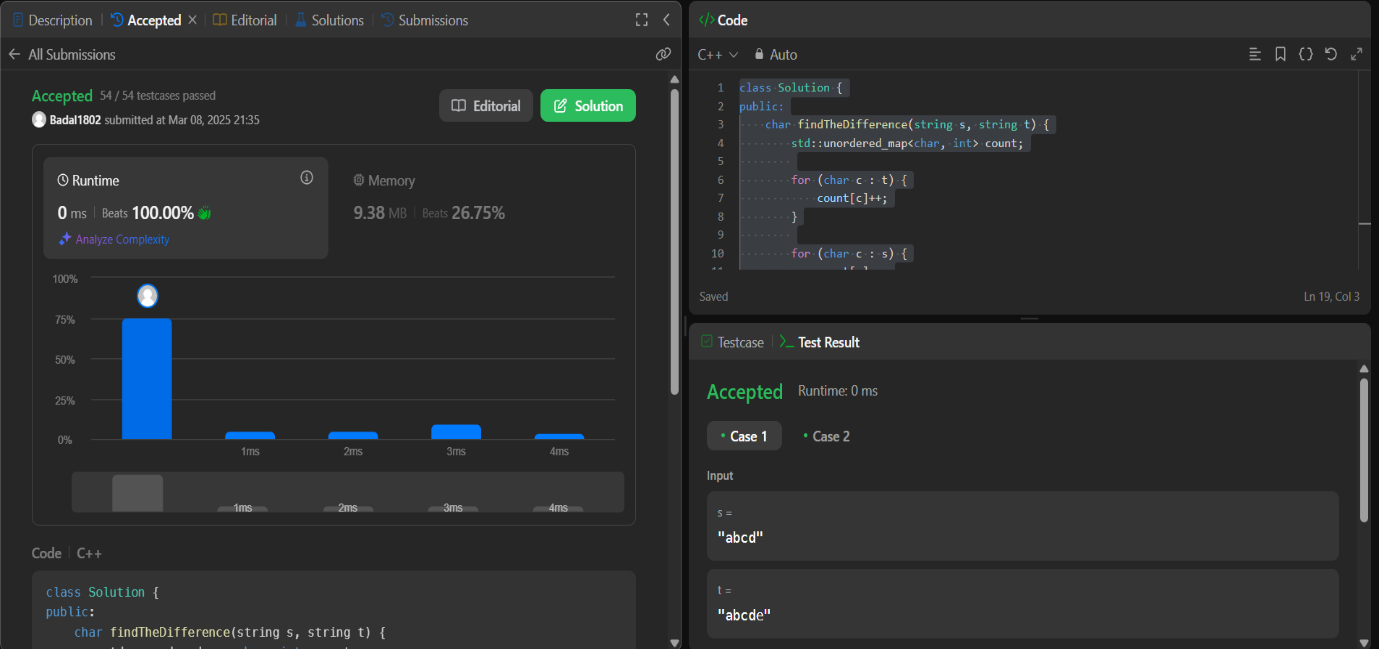
        }

        return count.begin()->first;

    }

};

**OUTPUT:**

****

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

**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];

            }

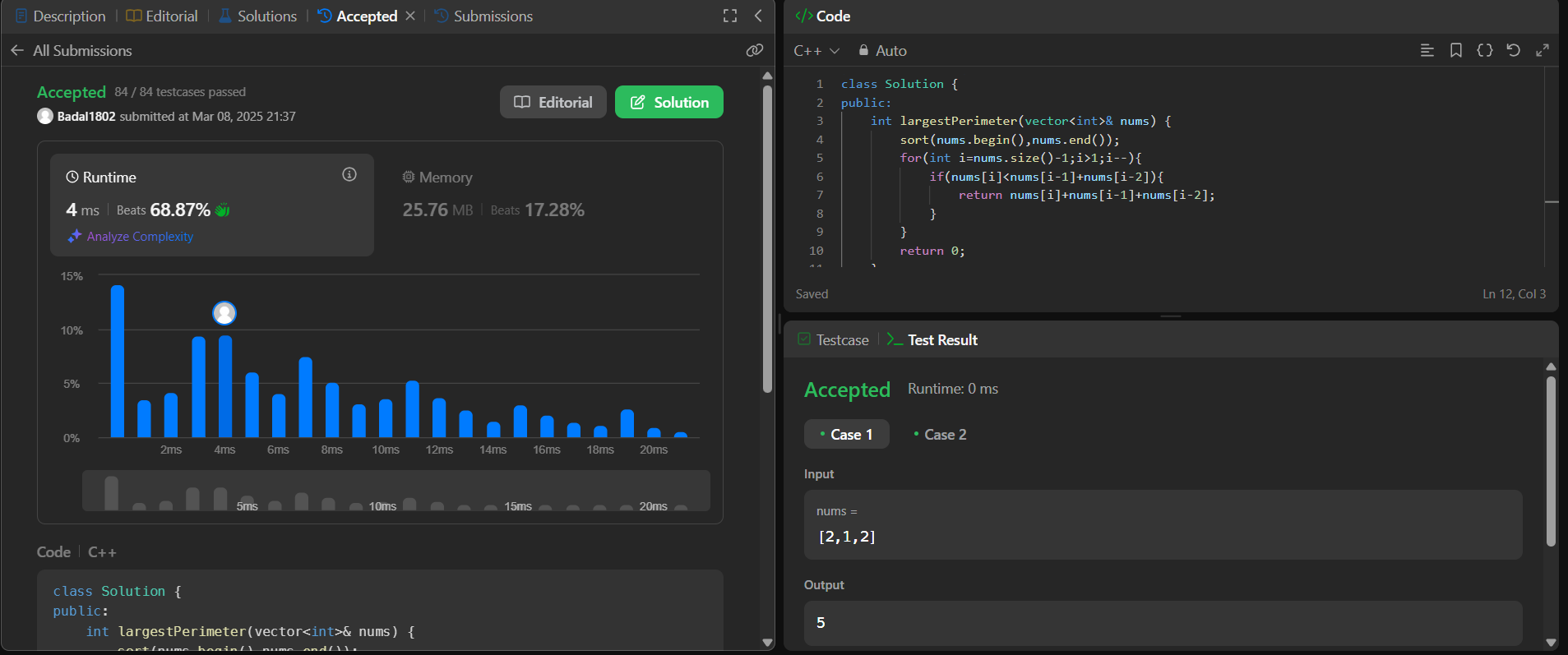
        }

        return 0;

    }

};

**OUTPUT:**

****

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

**CODE:**

class Solution {

public:

    int thirdMax(vector<int>& nums) {

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

        int count = 1;

        for (int i = 1; i < nums.size(); i++) {

        if (nums[i] != nums[i - 1]) {

        count++;

        if (count == 3) return nums[i];

        }

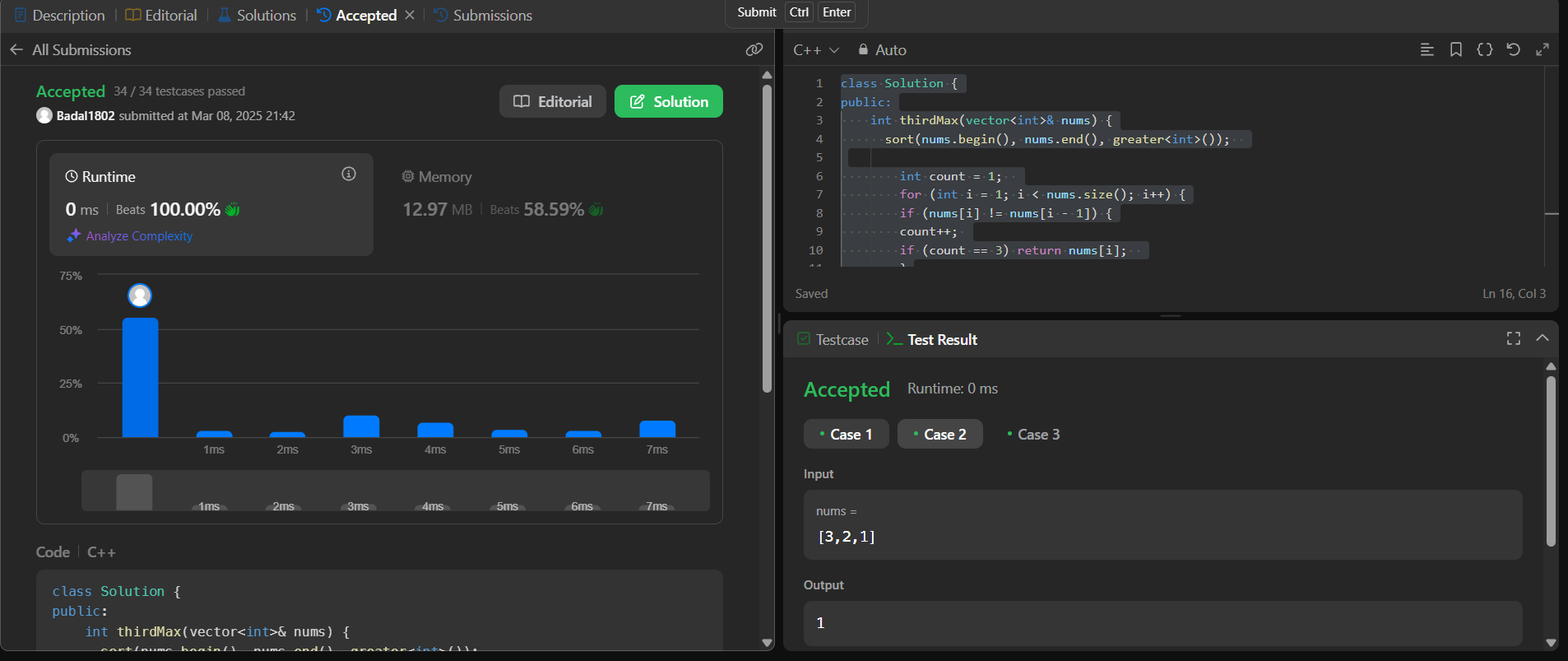
        }

         return nums[0];

    }

};

**OUTPUT:**

****

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

**CODE:**

class Solution {

public:

string frequencySort(string s) {

unordered\_map<char,int> mp;

multimap<int,char> r;

string ss="";

for(auto a : s)

mp[a]++;

for(auto a : mp)

r.insert({a.second, a.first});

for(auto it = r.rbegin(); it != r.rend(); ++it)

ss += string(it->first, it->second);

//for(auto it = r.rbegin(); it != r.rend(); ++it){

// for (int i = 0; i < it->first; ++i) {

// ss += it->second;

// }

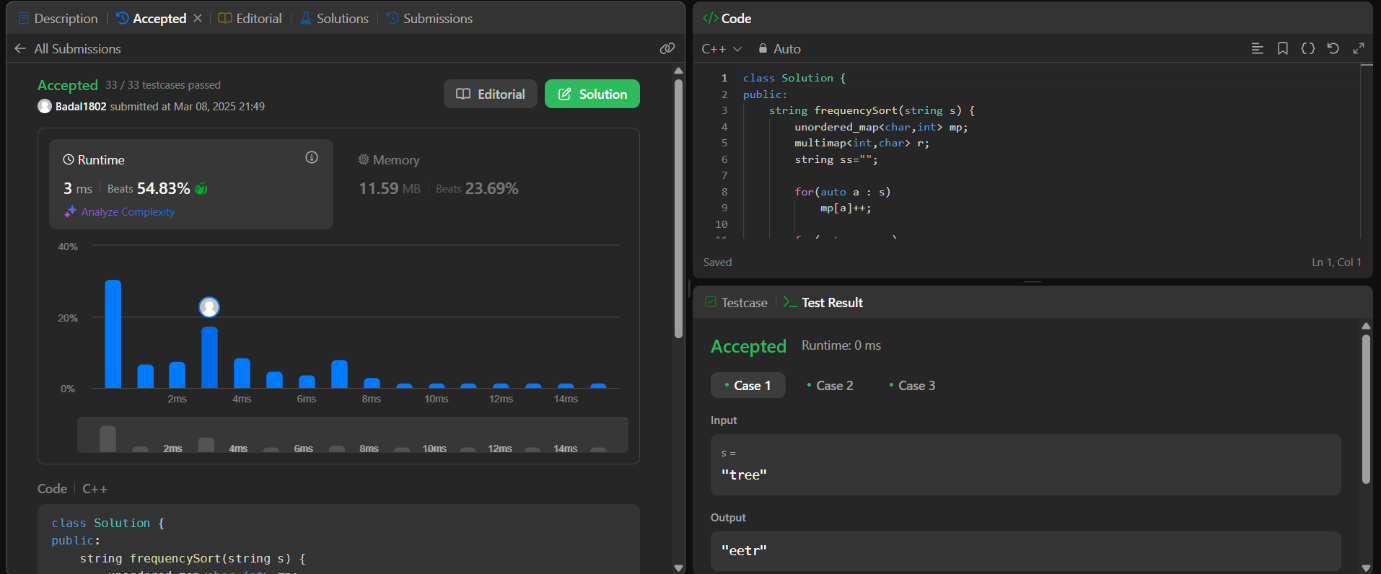
//}

return ss;

}

};

**OUTPUT:**

****

**452.**[**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) {

        int n = points.size();

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

        int cnt = 1;

        int x = points[0][0], y = points[0][1];

        for(int i = 1; i < n; i++){

            if((points[i][0] >= x && points[i][0] <= y) ||

               (points[i][1] >= x && points[i][1] <= y)){

                x = max(x, points[i][0]);

                y = min(y, points[i][1]);

            }

            else{

                x = points[i][0];

                y = points[i][1];

                cnt++;

            }

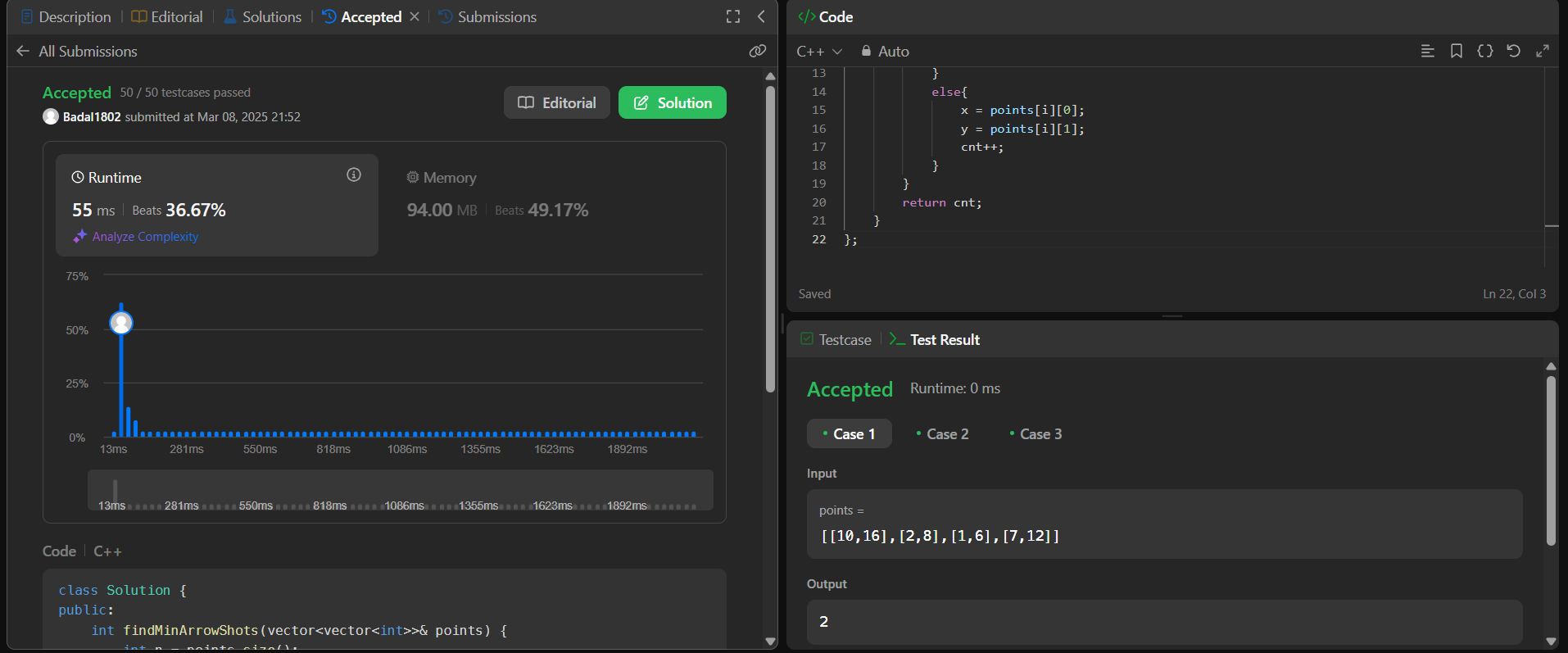
        }

        return cnt;

    }

};

**OUTPUT:**

****

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

**CODE:**

class Solution {

public:

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

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

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

while (i <= j) {

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

++i;

--j;

} else {

--j;

}

++cnt;

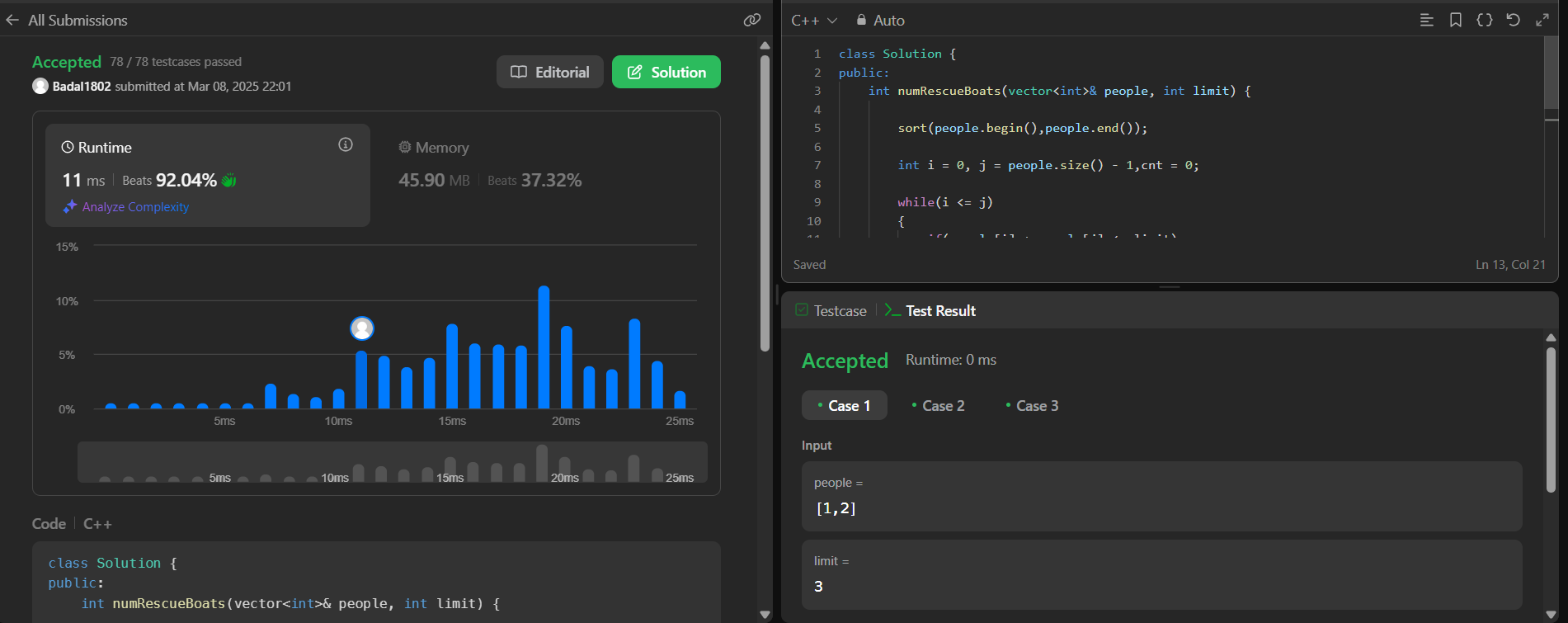
}

return cnt;

}

};

**OUTPUT:**

****

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

**CODE:**

class Solution {

public:

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

// Max heap to store distances and corresponding points

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(); // Remove farthest point if size > k

}

vector<vector<int>> ans;

while (!maxHeap.empty()) {

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

maxHeap.pop();

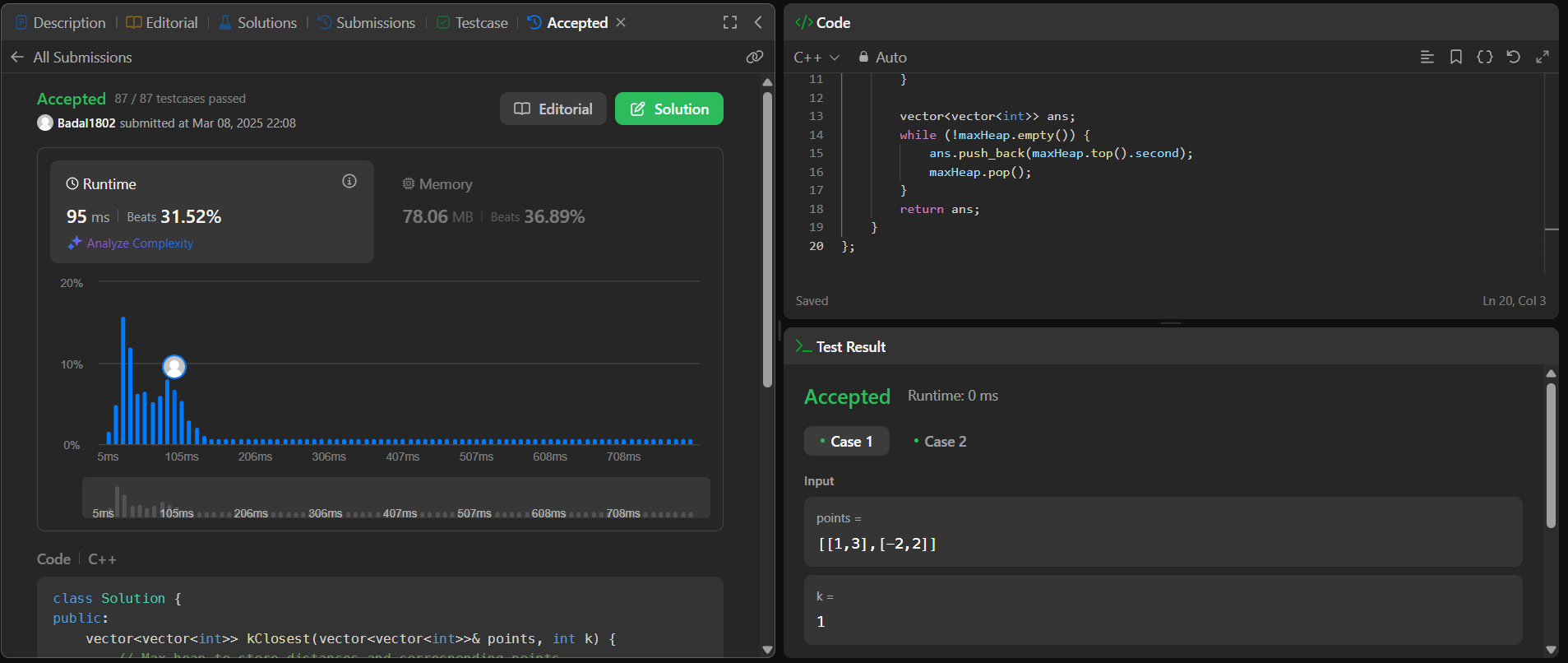
}

return ans;

}

};

**OUTPUT:**

****

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

**CODE:**

class Solution {

public:

int minSetSize(vector<int>& arr) {

unordered\_map<int,int>h;

for(int i = 0; i < arr.size(); i++) h[arr[i]]++;

priority\_queue<int> pq;

for(auto it: h) pq.push(it.second);

int ans = 0, minus = 0;

while(!pq.empty())

{

ans++;

minus += pq.top();

pq.pop();

if(minus >= (arr.size()/2)) break;

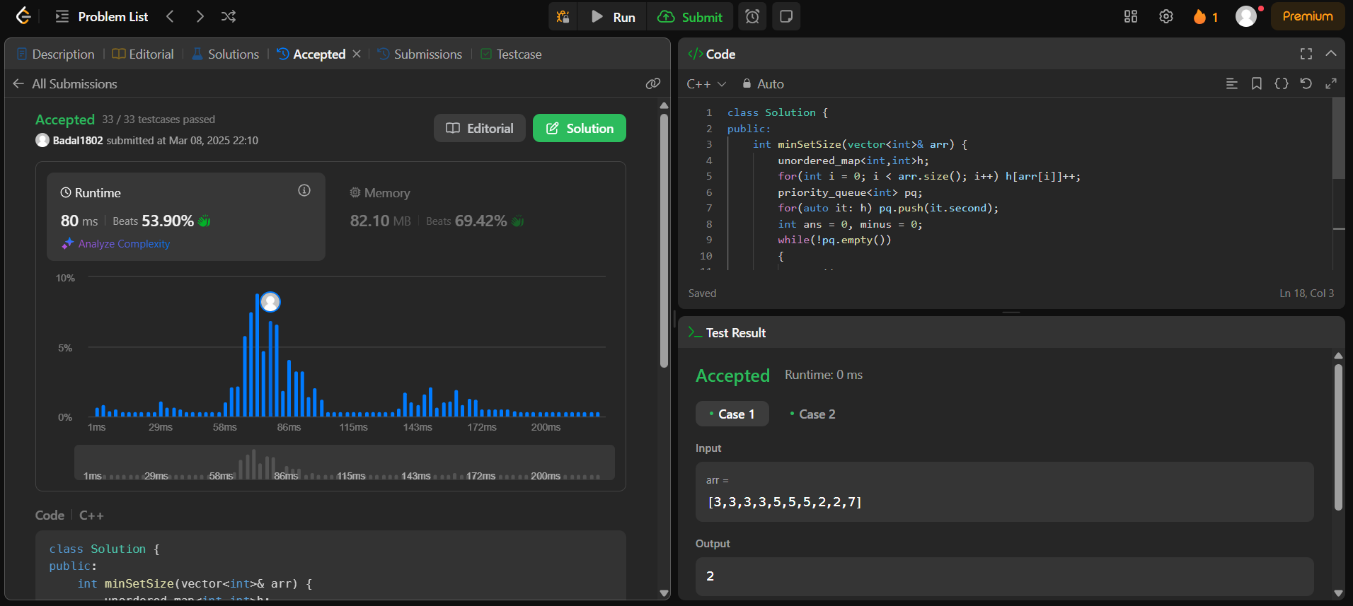
}

return ans;

}

};

**OUTPUT:**

****