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

* **Solution Code:**

class Solution {

public:

    char findTheDifference(string s, string t) {

        sort(s.begin(), s.end());

        sort(t.begin(), t.end());

        int i=0;

        while(s[i]==t[i] && i<s.length()){

            i++;

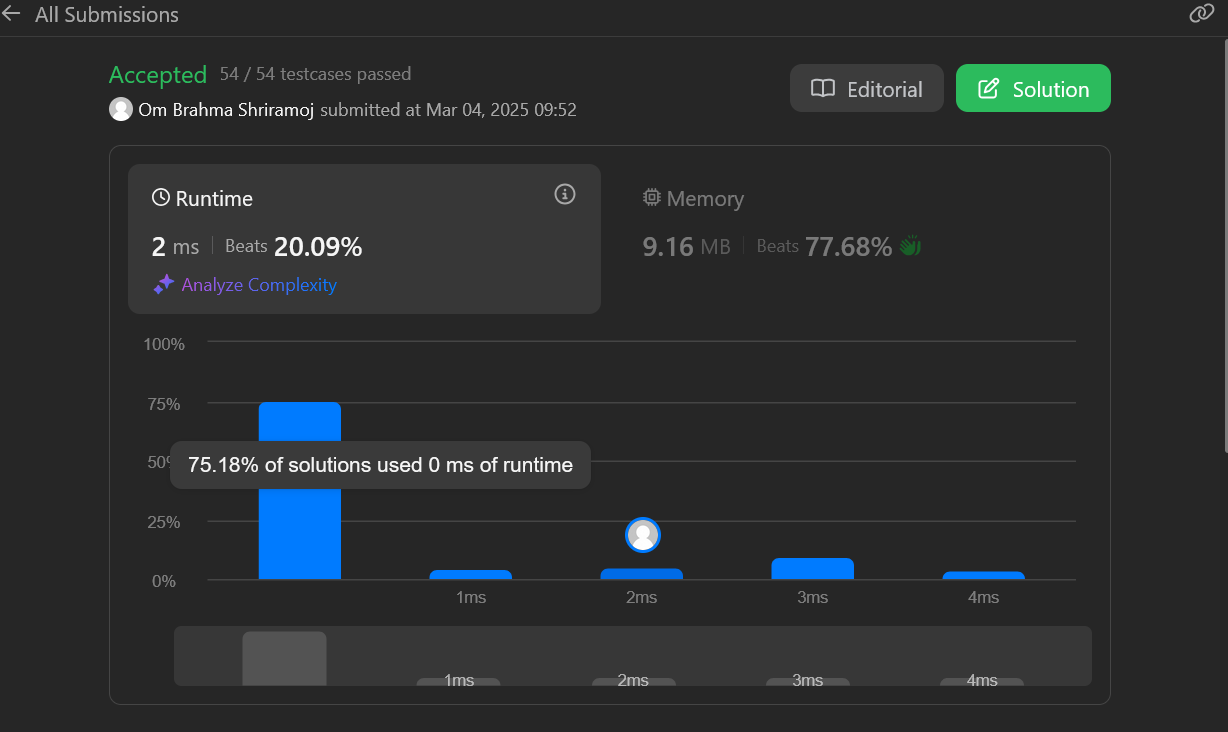
        }

        return t[i];

    }

};

* **Screenshot:**

****

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

* **Source Code:**

class Solution {

public:

    int largestPerimeter(vector<int>& nums) {

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

        for (int i = nums.size()-1; i >= 2; i--){

            if (nums[i-2] + nums[i-1] > nums[i]) return nums[i-2]+nums[i-1]+nums[i];

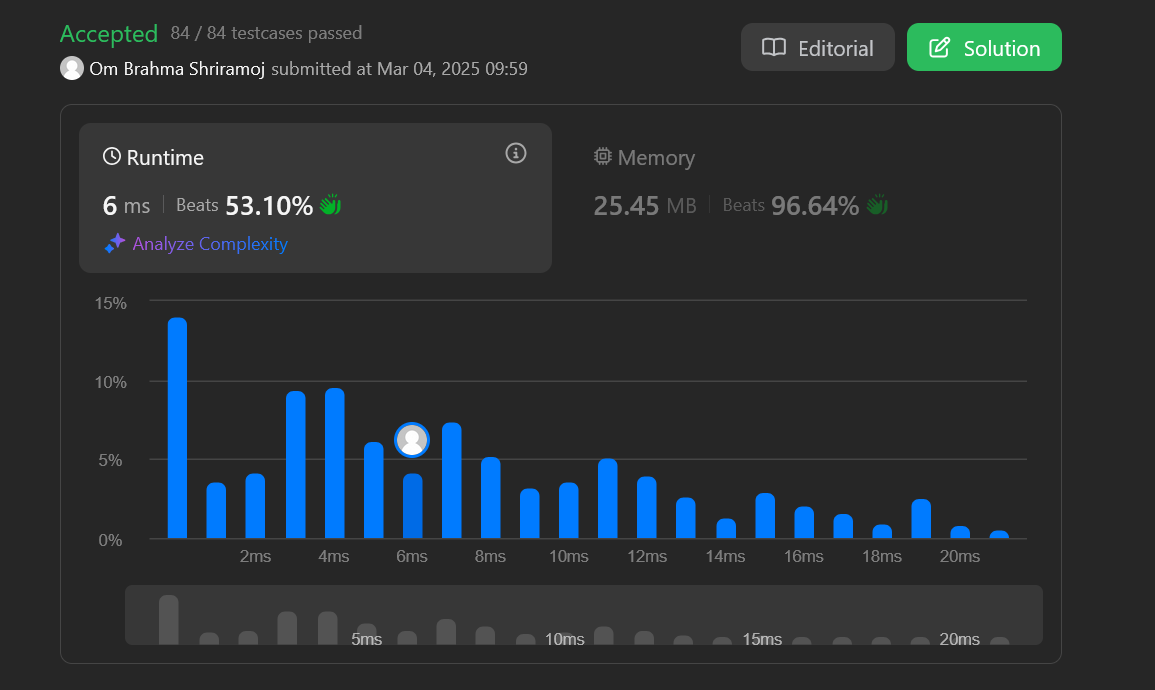
        }

        return 0;

    }

};

* **Screenshot:**

****

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

* **Source Code:**

class Solution {

public:

    int thirdMax(vector<int>& nums) {

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

        set<int> d;

        for(auto i:nums){

            d.insert(i);

        }

        auto s=d.rbegin();

        if(d.size()<3){

            return \*d.rbegin();

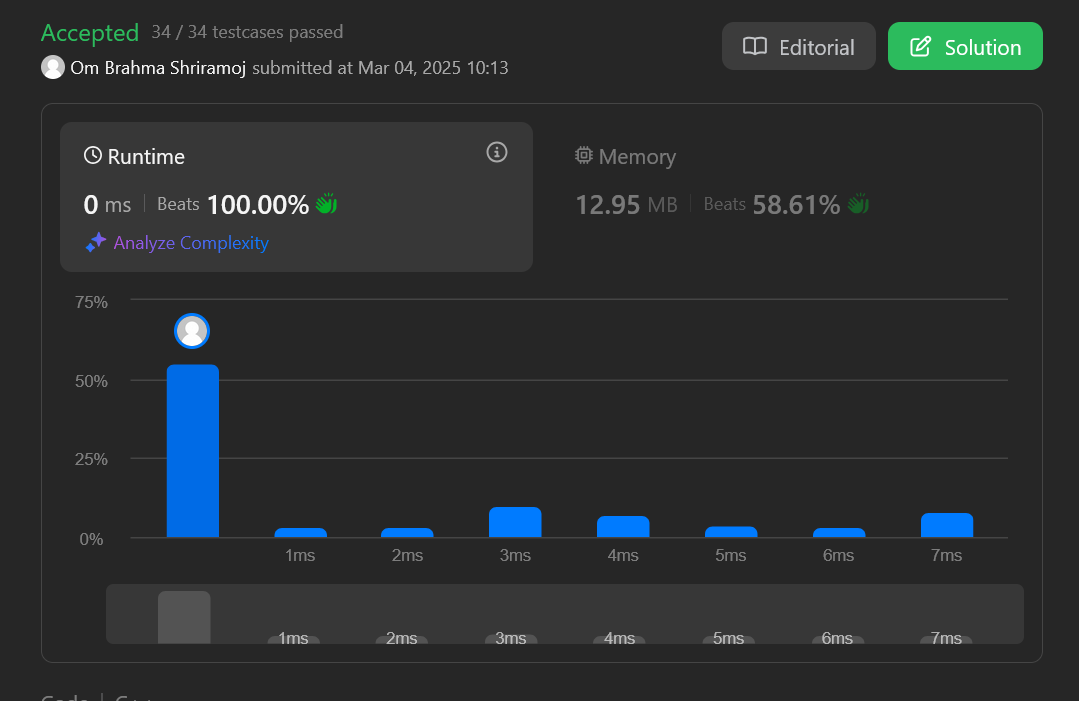
        }

        advance(s,2);

        return \*s;

    }

};

* **Screenshot:** 

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

* **Source 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);

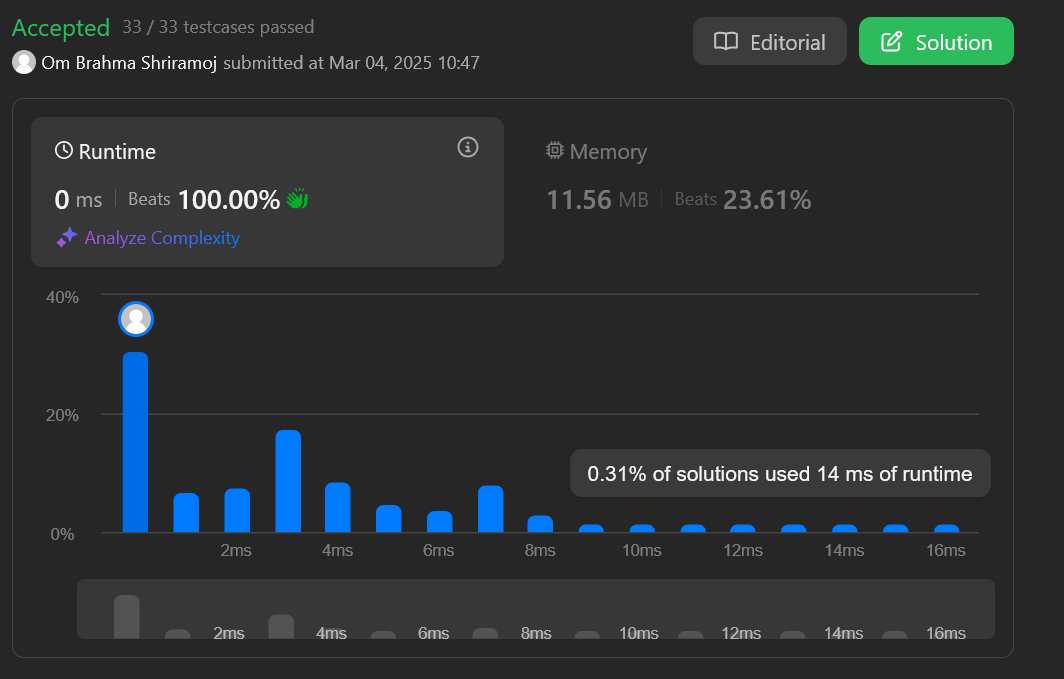
        }

        return result;

    }

};

* **Screenshot:**

****

**452.**[**Minimum Number of Arrows to Burst Balloons**](https://leetcode.com/problems/minimum-number-of-arrows-to-burst-balloons/)

* **Source Code:**

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]);

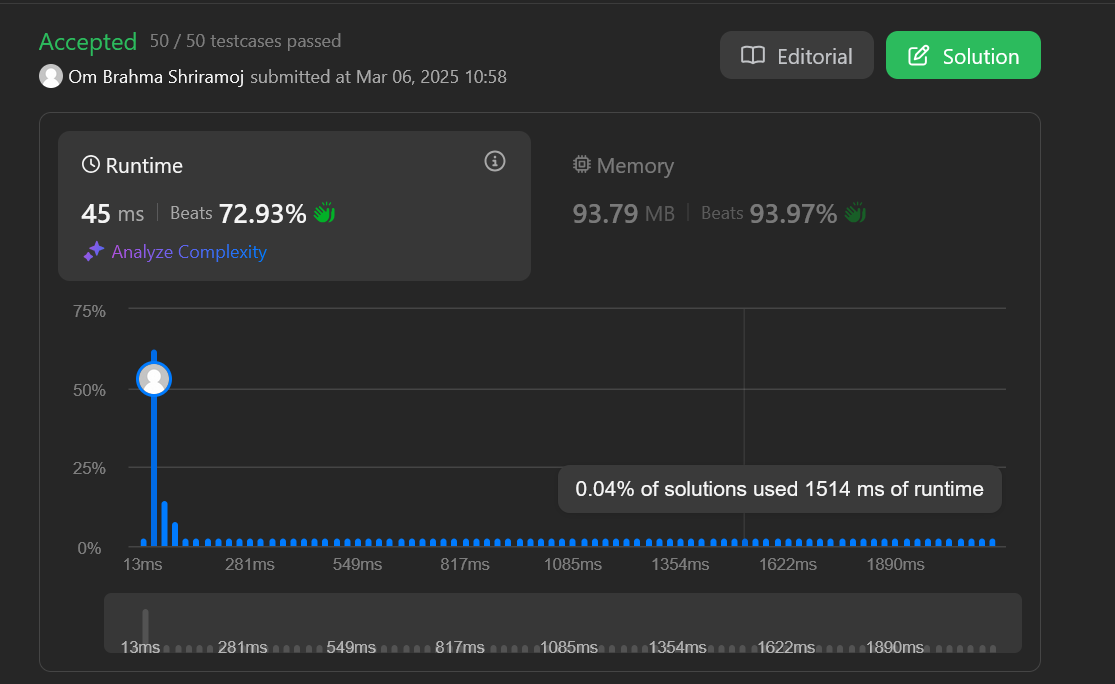
            }

        }

        return arrows;

    }

};

* **Screenshot:** 

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

* **Source Code:**

#pragma GCC optimize("O3", "unroll-loops")

class Solution {

public:

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

        unsigned freq[30001]={0};

        int maxW=0, minW=30001;

        for(int x: people){

            freq[x]++;

            maxW=max(maxW, x);

            minW=min(minW, x);

        }

        for (int i=minW, j=0; i<=maxW; i++){

            int f=freq[i];

            fill(people.begin()+j, people.begin()+j+f, i);

            j+=f;

        }

        int x=0;

        for(int l=0, r=people.size()-1;l<=r; r--){

            x++;

            if (people[l]+people[r]<=limit)

                l++;

        }

        return x;

    }

};

auto init = []() {

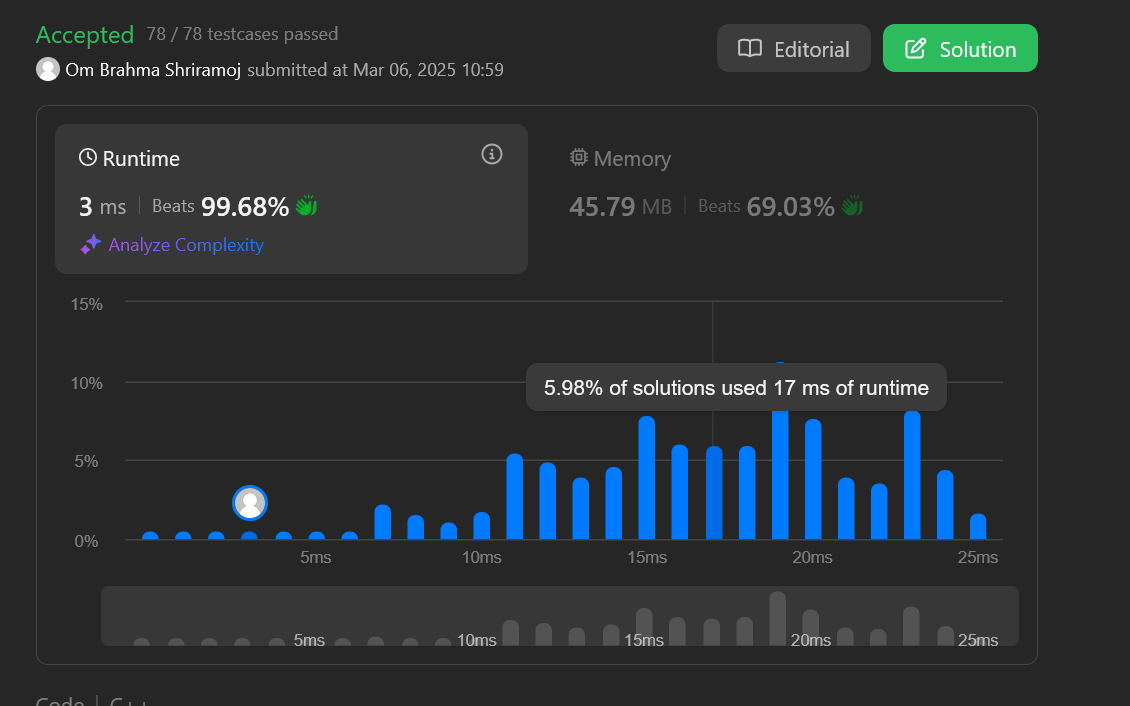
    ios::sync\_with\_stdio(0);

    cin.tie(0);

    cout.tie(0);

    return 'c';

}();

* **Screenshot:** 

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

* **Source Code:**

class Solution {

public:

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

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

        for(auto point:points)

        {

            int x\_cord = point[0];

            int y\_cord = point[1];

            int dist = x\_cord\*x\_cord+y\_cord\*y\_cord;

            if(closestpoints.size()<k)

            {

                closestpoints.push({dist,point});

            }

            else if(dist<closestpoints.top().first)

            {

                closestpoints.pop();

                closestpoints.push({dist,point});

            }

        }

        vector<vector<int>> resultClosePoints;

        while(k>0)

        {

            resultClosePoints.push\_back(closestpoints.top().second);

            closestpoints.pop();

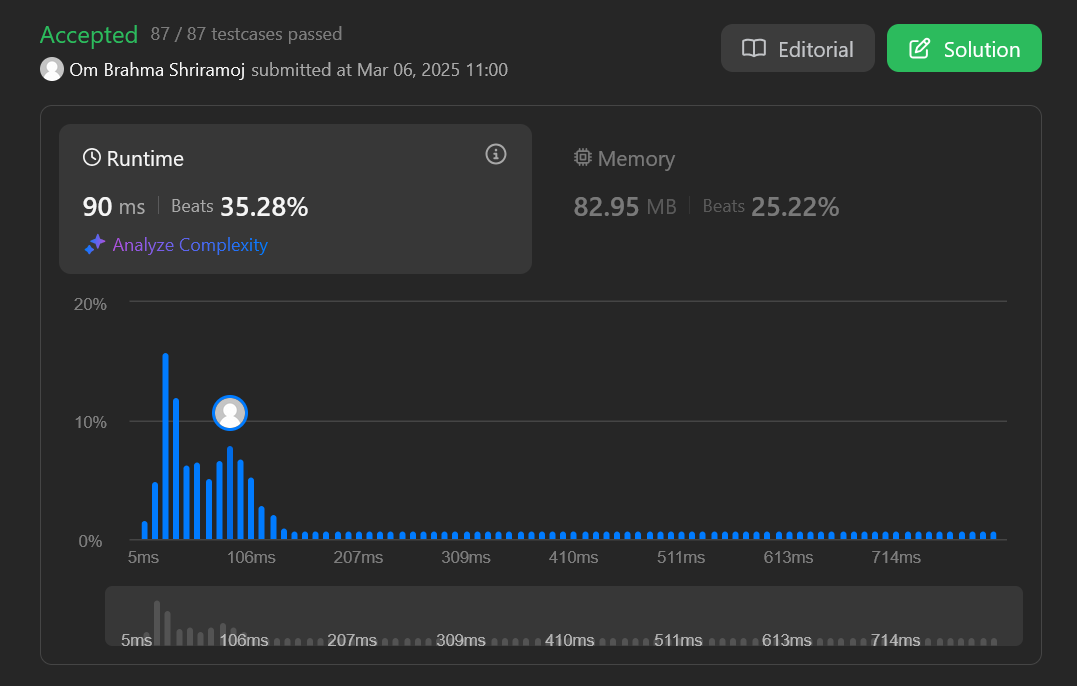
            k--;

        }

        return resultClosePoints;

    }

};

* **Screenshot:** 

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

* **Source 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;

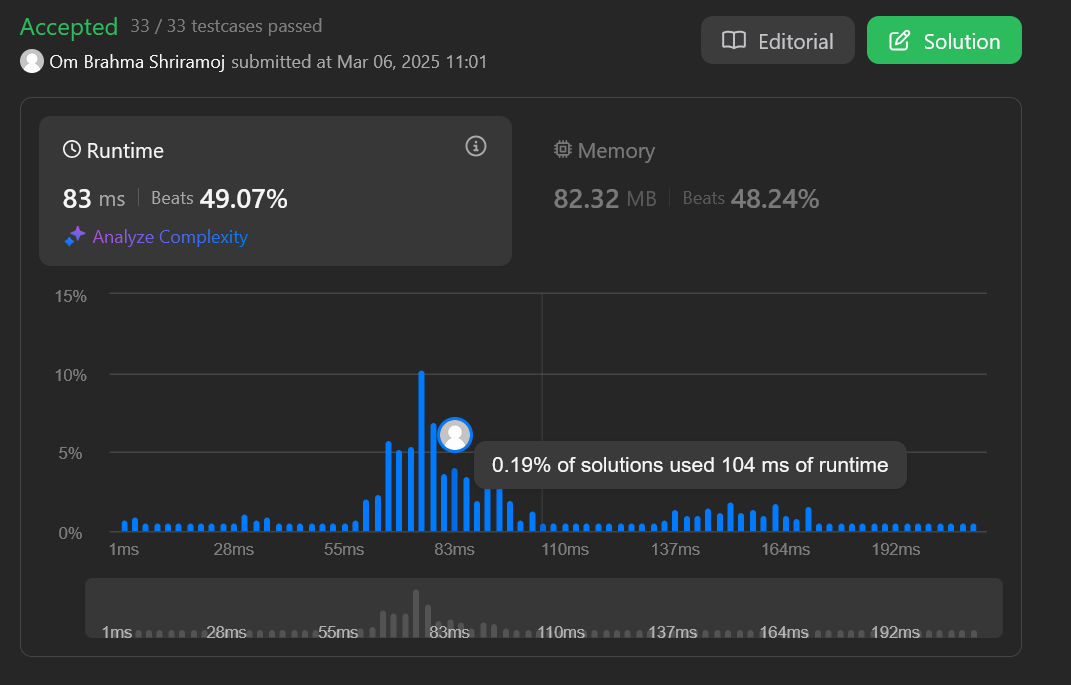
        }

        return ans;

    }

};

* **Screenshot:**

****