**Name : saksham Mishra**

**Uid : 22bcs15604**

<https://leetcode.com/problems/find-the-difference/description/>

Ans.

class Solution {

public:

char findTheDifference(string s, string t) {

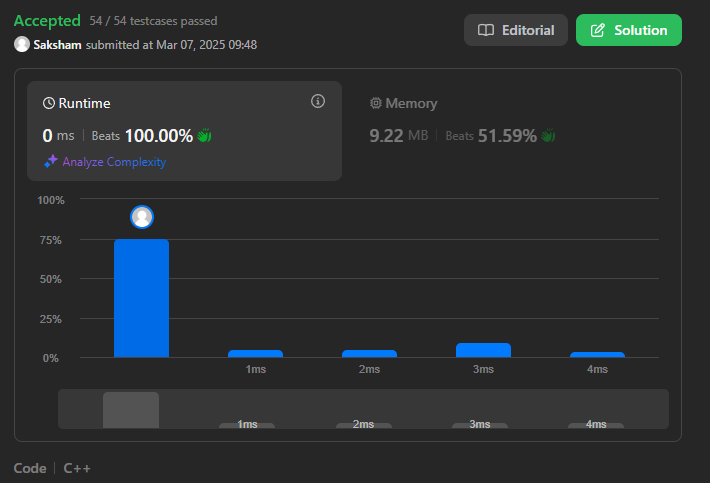
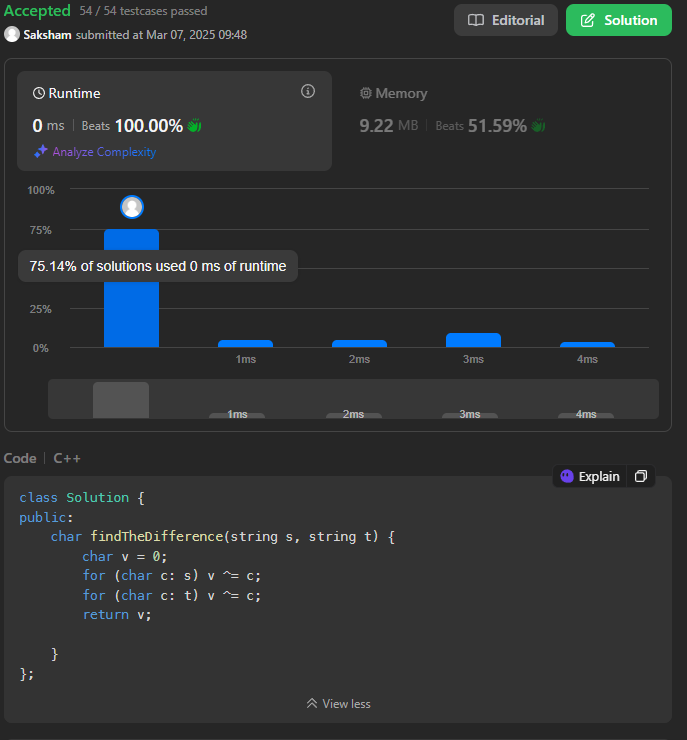
char v = 0;

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

for (char c: t) v ^= c;

return v;

}

};

Q2 - <https://leetcode.com/problems/largest-perimeter-triangle/description/>

Ans - 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])

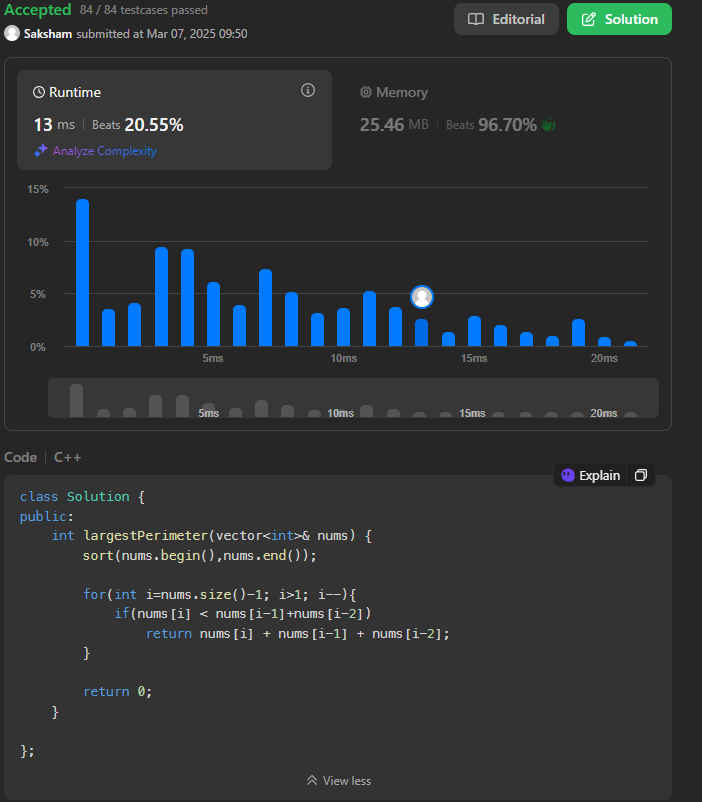
                return nums[i] + nums[i-1] + nums[i-2];

        }

        return 0;

    }

};



<https://leetcode.com/problems/third-maximum-number/description/>

class Solution {

public:

    int thirdMax(vector<int>& nums) {

        if(nums.size()==1)

            return nums[0];

        if(nums.size()==2)

            return max(nums[0],nums[1]);

        long m1=LONG\_MIN;

        long m2=LONG\_MIN;

        long m3=LONG\_MIN;

        for(int i:nums){

            if(i==m1 || i==m2 || i==m3)

                continue;

            if(i>m1){

                m3=m2;

                m2=m1;

                m1=i;

            }

            else if(i>m2){

                m3=m2;

                m2=i;

            }

            else if(i>m3)

                m3=i;

        }

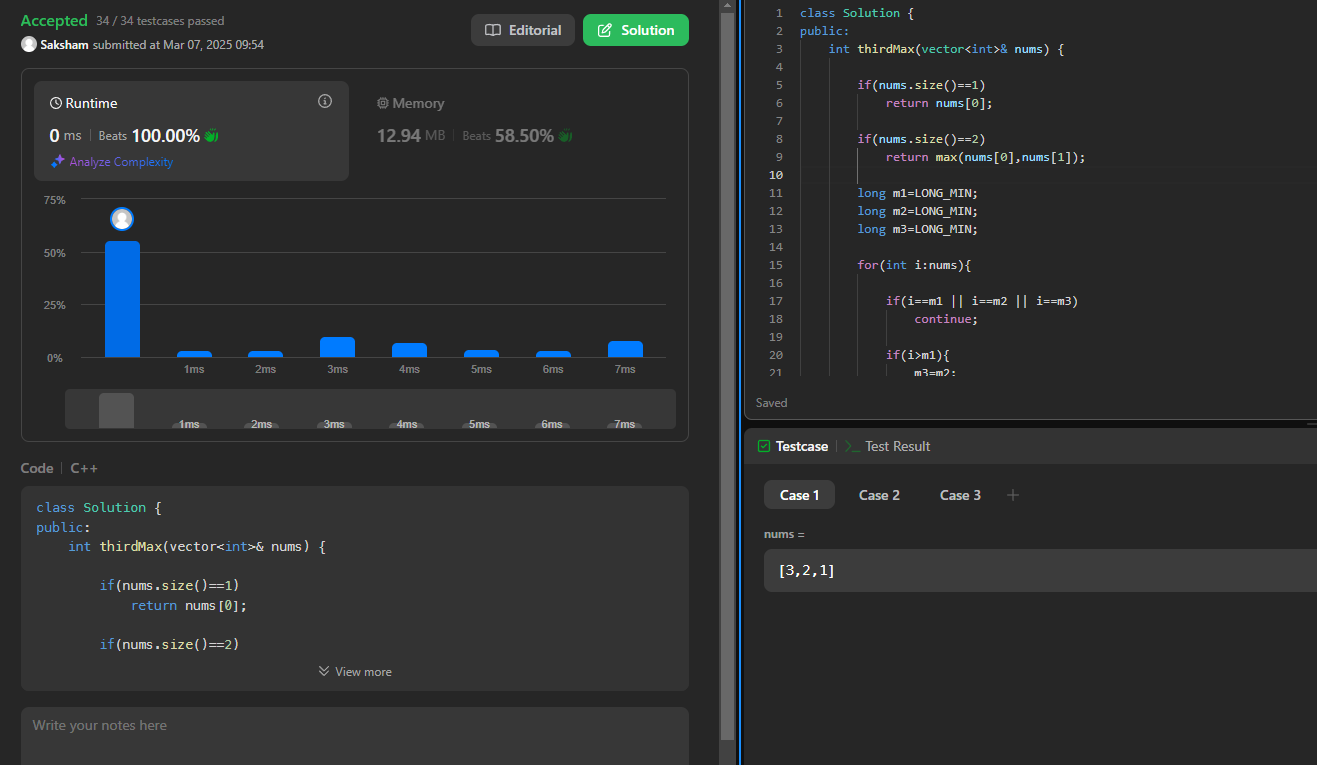
        if(m3==LONG\_MIN)

            return (int)m1;

        return (int)m3;

    }

};



<https://leetcode.com/problems/sort-characters-by-frequency/description/>

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();

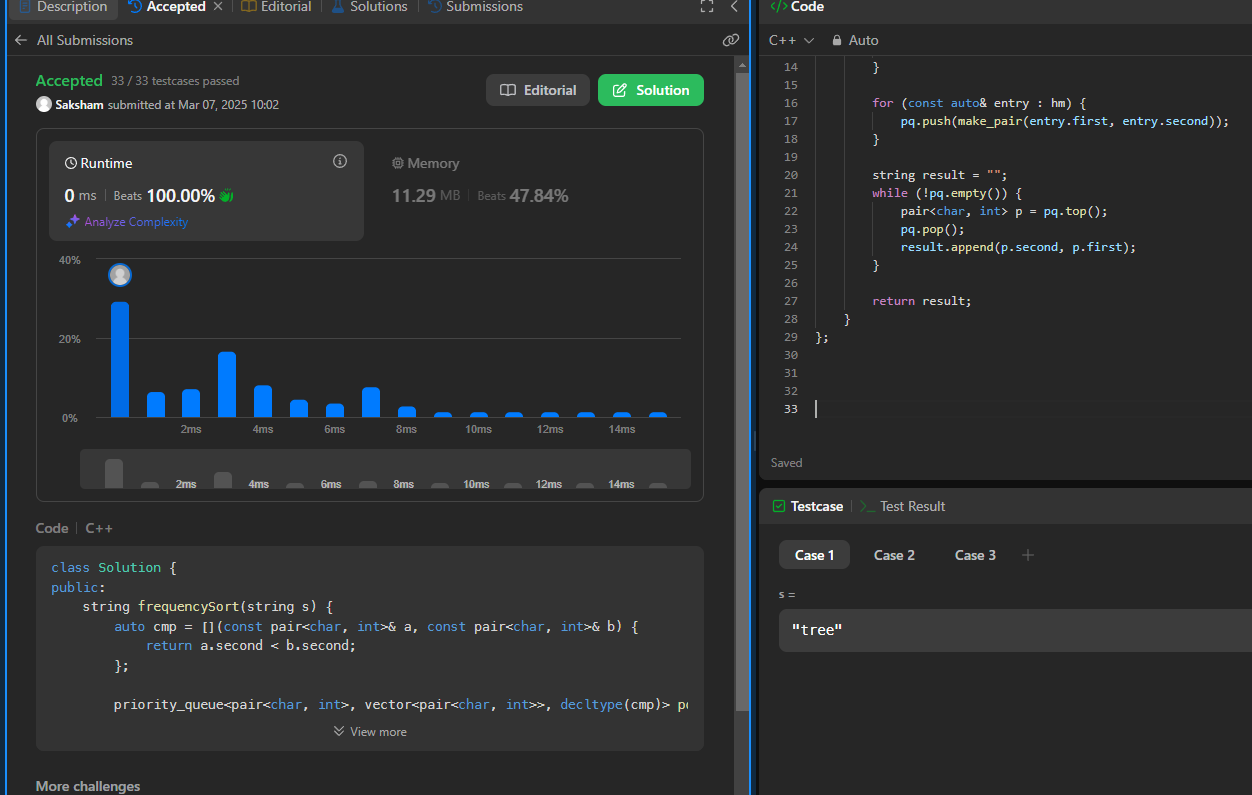
            result.append(p.second, p.first);

        }

        return result;

    }

};



<https://leetcode.com/problems/minimum-number-of-arrows-to-burst-balloons/submissions/1565571045/>

class Solution {

public:

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

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

int lastpoint = p[0][1];

int ans = 1;

for(auto point : p) {

if(point[0] > lastpoint) {

ans++;

lastpoint = point[1];

}

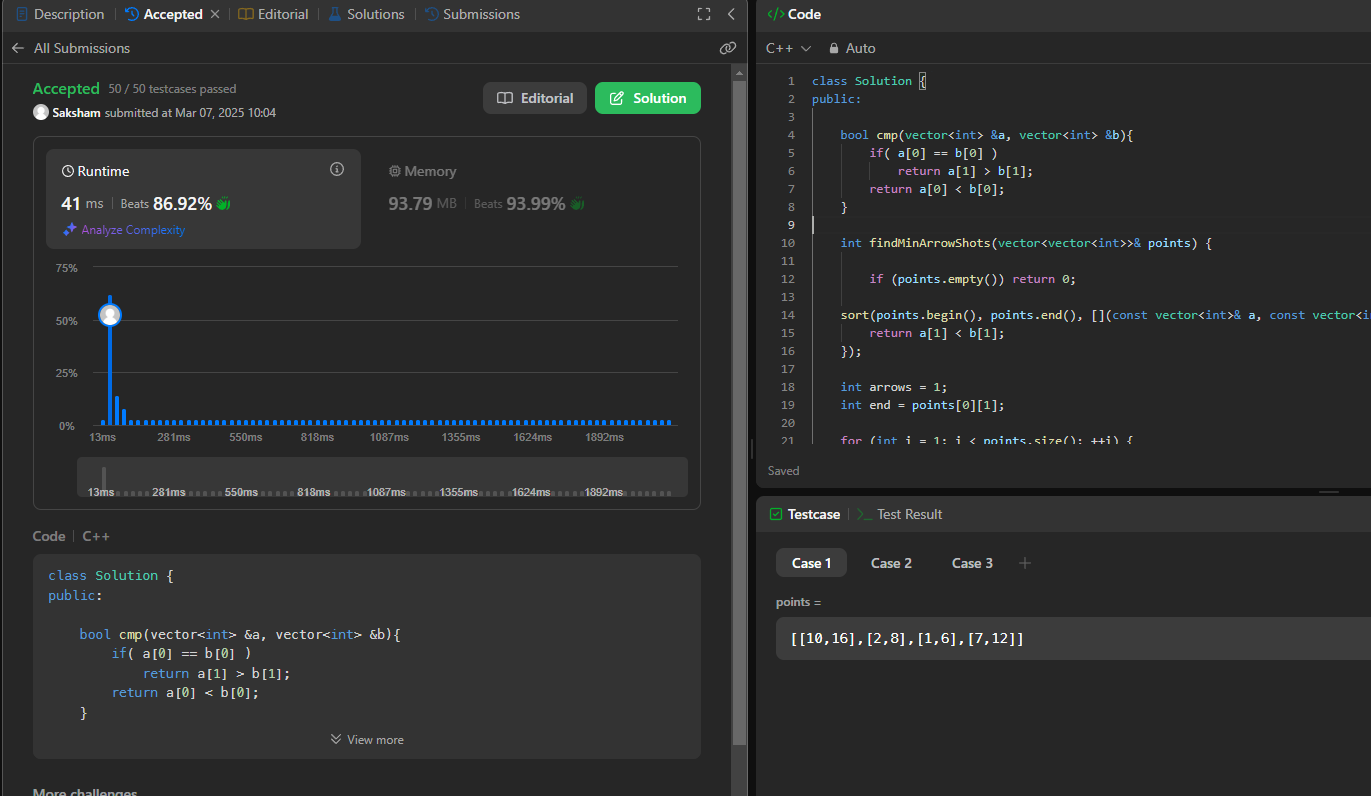
lastpoint = min(point[1],lastpoint);

}

return ans;

}

};



<https://leetcode.com/problems/boats-to-save-people/description/>

class Solution {

public:

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

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

        int n=people.size();

        int i=0,j=n-1;

        int count=0;

        while(i<=j){

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

                i++;

                j--;

            }

            else

                j--;

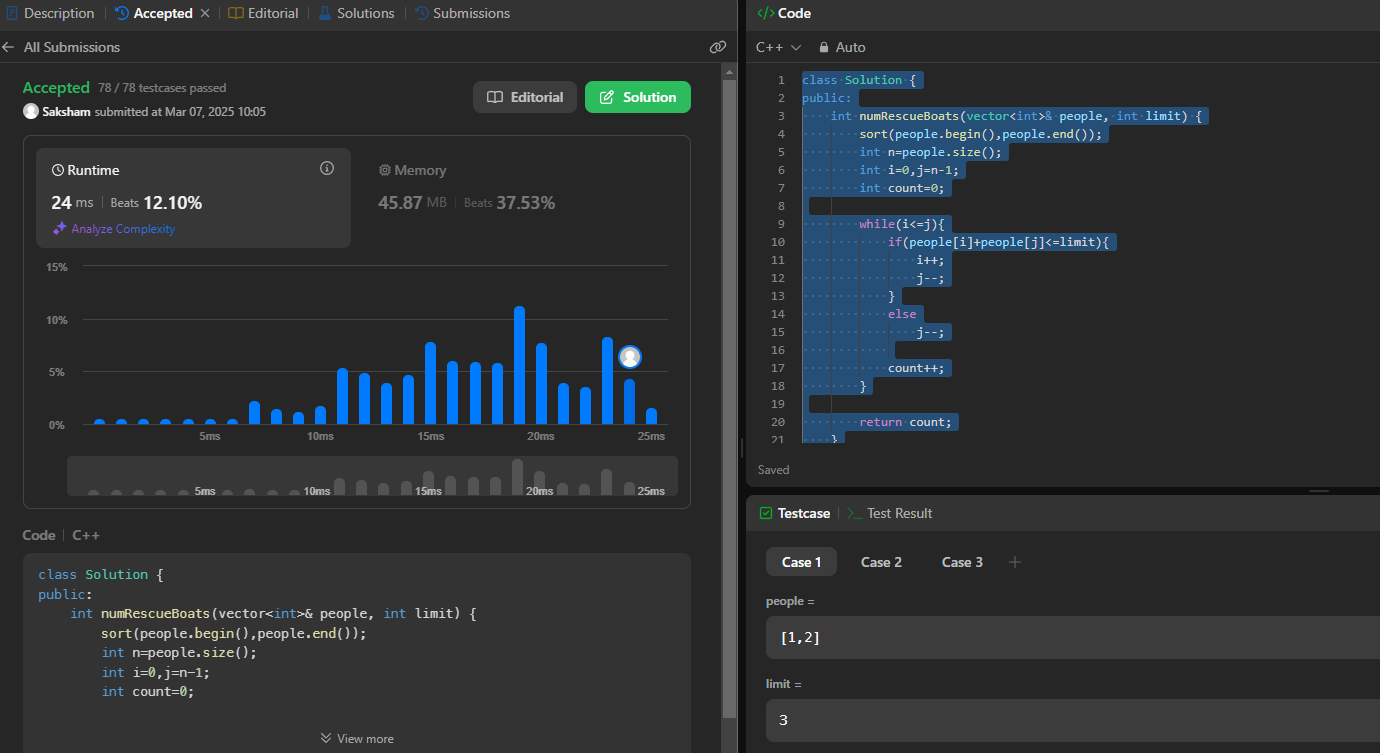
            count++;

        }

        return count;

    }

};



<https://leetcode.com/problems/k-closest-points-to-origin/description/>

//Upvote and Comment

class Solution {

public:

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

//Answer vector

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

//maxheap storage initialization

priority\_queue<vector<int>> maxHeap;

//Construction of 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();

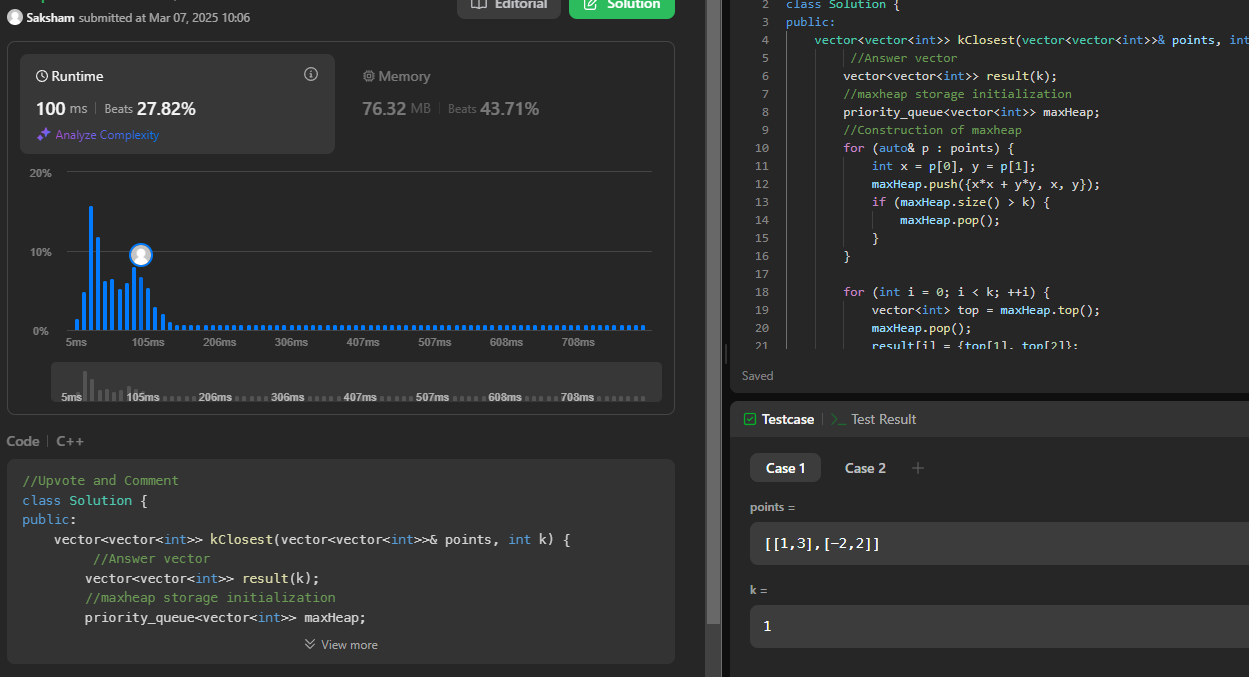
maxHeap.pop();

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

}

return result;

}

};

<https://leetcode.com/problems/reduce-array-size-to-the-half/description/>

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;

}

};

