Siddharth

22BCS15779

604\_FL\_A

Assignment - 5

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

class Solution {

public:

    char findTheDifference(string s, string t) {

        sort(s.begin(),s.end());

        sort(t.begin(),t.end());

        char res ;

        for(int i = 0;i<t.length();i++){

            if(s[i] != t[i]){

                return t[i];

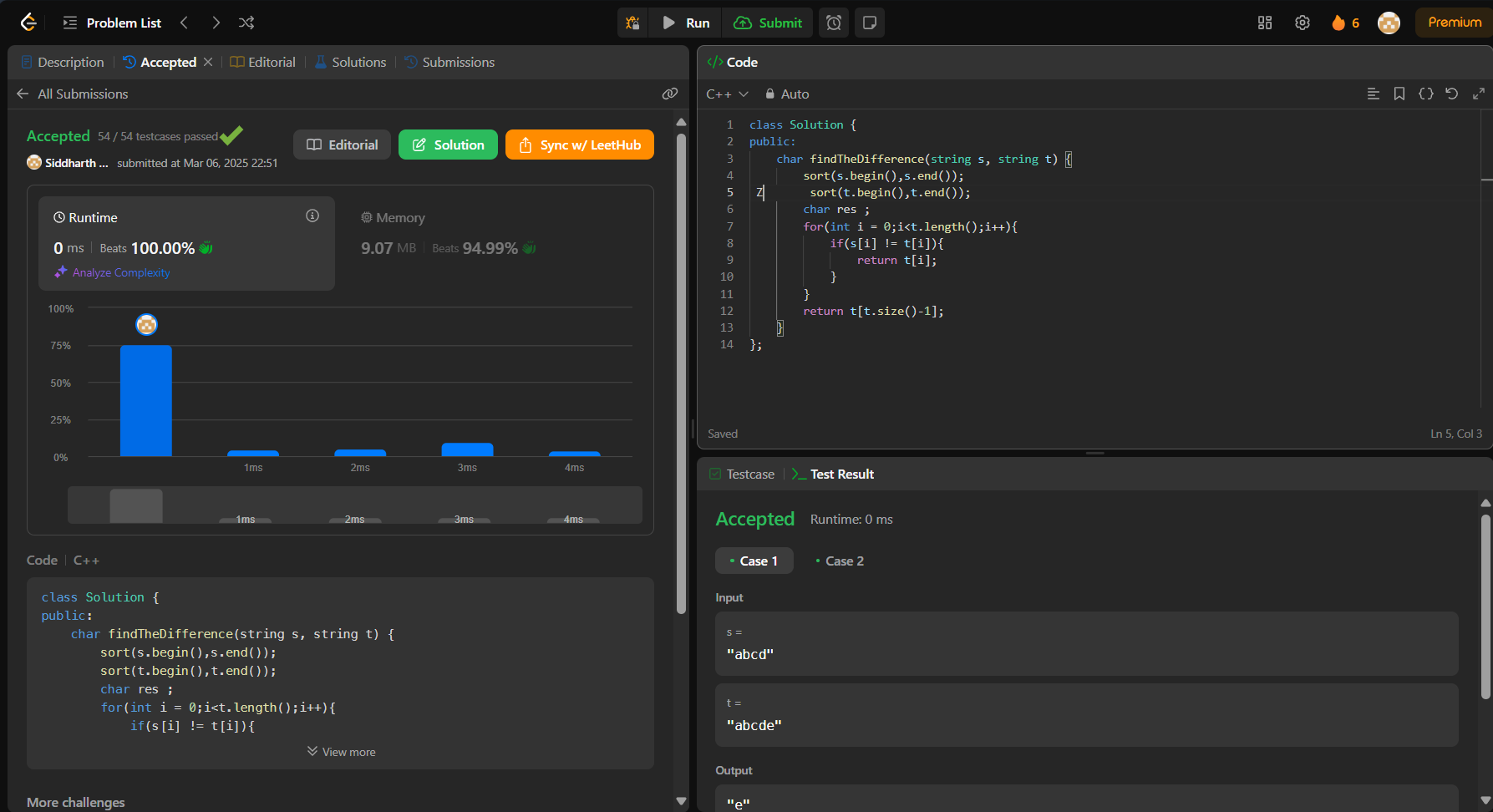
            }

        }

        return t[t.size()-1];

    }

};



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

class Solution {

public:

    int largestPerimeter(vector<int>& nums) {

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

        for(int i = 0; i < nums.size()-2; i++){

            if (nums[i] < nums[i+1] + nums[i+2])

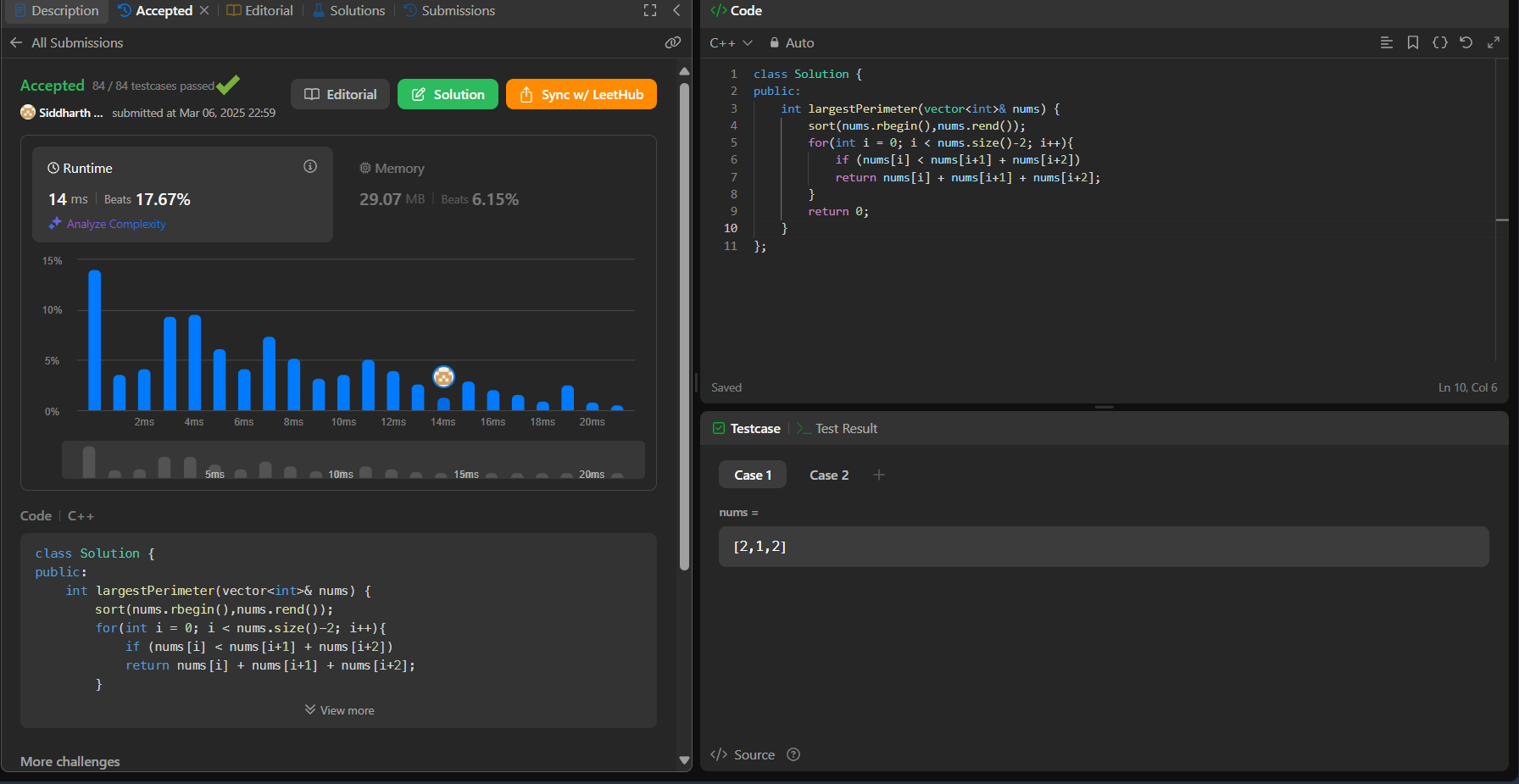
            return nums[i] + nums[i+1] + nums[i+2];

        }

        return 0;

    }

};



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

class Solution {

public:

    int thirdMax(vector<int>& nums) {

        double i,j,a=0.5,b=0.5,c=0.5,n=nums.size();

    sort(nums.begin(),nums.end());

        for(i=n-1;i>=0;i--){

            if(i==n-1)

             a=nums[n-1];

             else if(nums[i]!=a&&b==0.5)

             b=nums[i];

             else if(nums[i]!=a&&nums[i]!=b){

                c=nums[i];

                break;

             }

        }

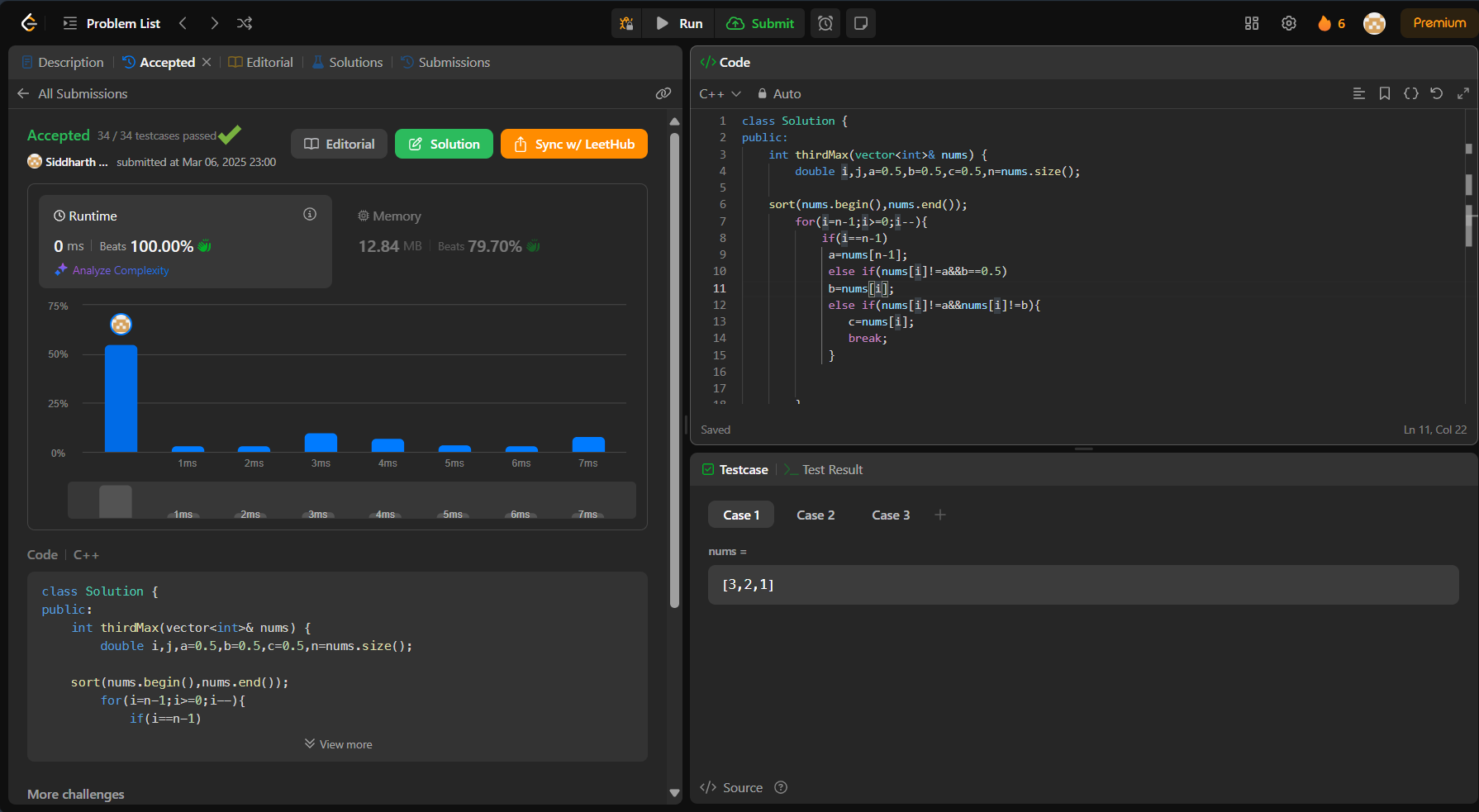
        if(c!=0.5)

        return c;

        return a;

    }

};



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

class Solution {

public:

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

        int boatCount = 0;

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

        int left = 0;

        int right = people.size() - 1;

        while(left <= right){

            int sum = people[left] + people[right];

            if(sum <= limit){

                boatCount++;

                left++;

                right--;

            }

            else{

                boatCount++;

                right--;

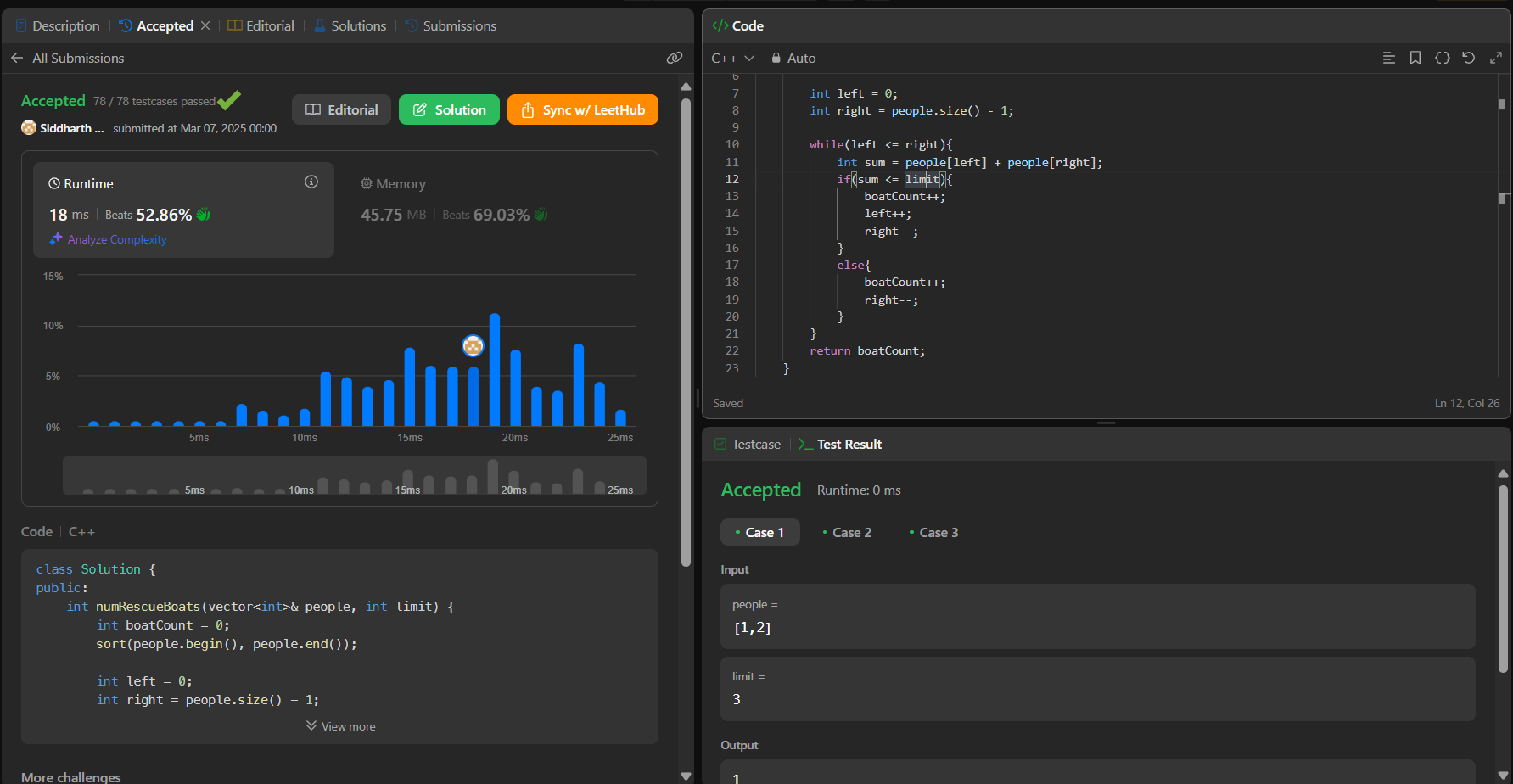
            }

        }

        return boatCount;

    }

};



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

class Solution {

public:

    int minSetSize(vector<int>& arr) {

        int n = arr.size();

        unordered\_map<int, int> cnt;

        for (int x : arr) ++cnt[x];

        vector<int> counting(n + 1);

        for (auto [\_, freq] : cnt) ++counting[freq];

        int ans = 0, removed = 0, half = n / 2, freq = n;

        while (removed < half) {

            ans += 1;

            while (counting[freq] == 0) --freq;

            removed += freq;

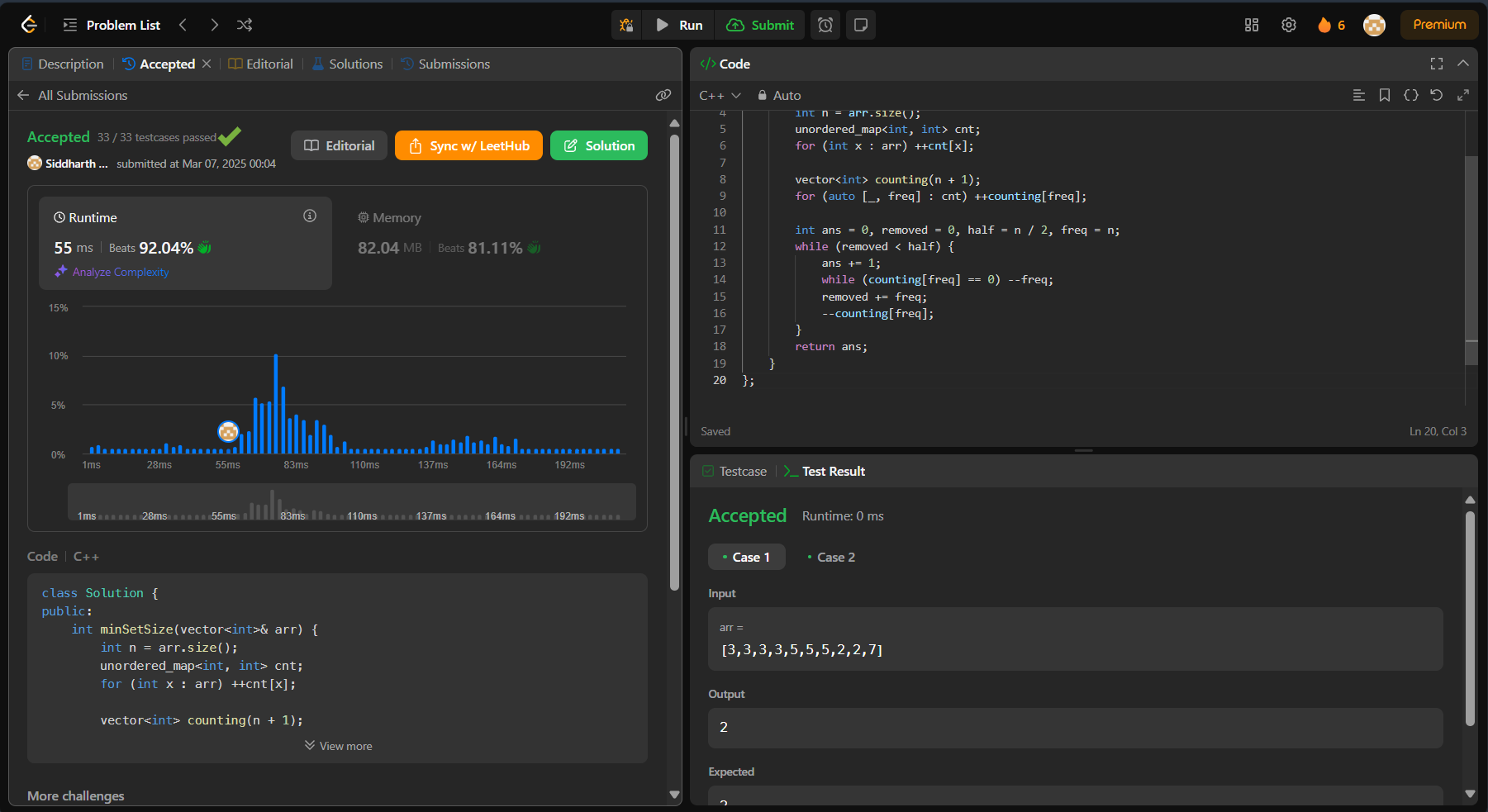
            --counting[freq];

        }

        return ans;

    }

};



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

class Solution {

public:

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

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

        for (auto& point : points) {

            int dist = point[0] \* point[0] + point[1] \* point[1];

            pq.push({dist, point});

            if (pq.size() > k) pq.pop();

        }

        vector<vector<int>> result;

        while (!pq.empty()) {

            result.push\_back(pq.top().second);

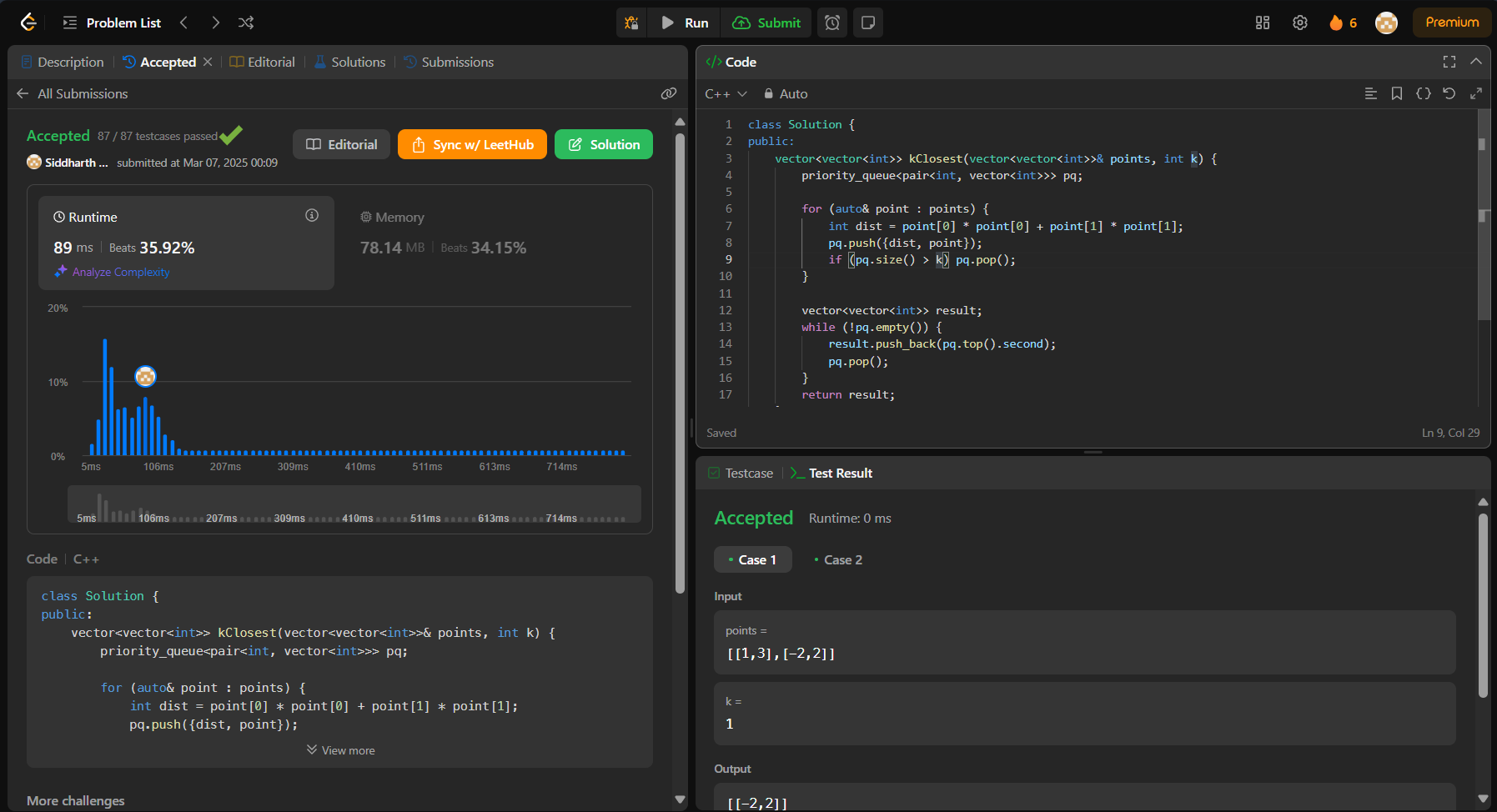
            pq.pop();

        }

        return result;

    }

};



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

class Solution {

public:

    static bool st(pair<char,int>& a,pair<char,int>& b)

    {

        if (a.second == b.second)  return a.first < b.first;

        return a.second > b.second;

    }

    string frequencySort(string s) {

        unordered\_map<char,int> mp;

        for(char c:s)

        {

            mp[c]++;

        }

        vector<pair<char,int>> arr(mp.begin(),mp.end());

        sort(arr.begin(),arr.end(),st);

        string s1;

        for(auto& it:arr)

        {

            for(int i=0;i<it.second;i++)

            {

                s1+=it.first;

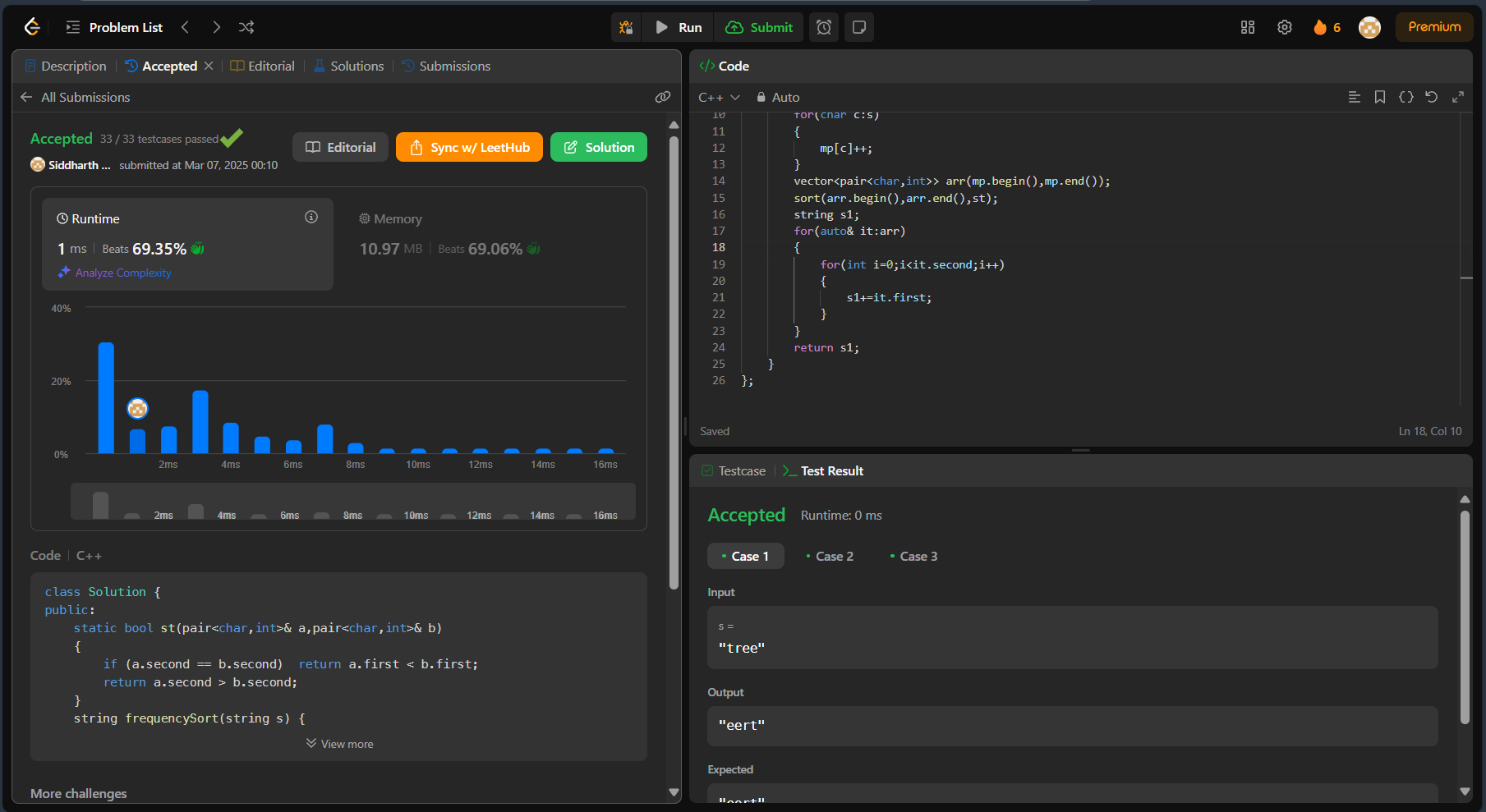
            }

        }

        return s1;

    }

};



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

class Solution {

public:

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

int n = intervals.size();

vector<int>v;

for(int i=0;i<n;i++){

v.push\_back(i);

}

sort(v.begin(),v.end(),[&](int a, int b){

return intervals[a][1]<intervals[b][1];

});

int count =0;

int endValue = intervals[v[0]][1];

for(int i=1;i<n;i++){

if(intervals[v[i]][0]>endValue){

endValue = intervals[v[i]][1];

count++;

}

}

return count+1;

}

};

