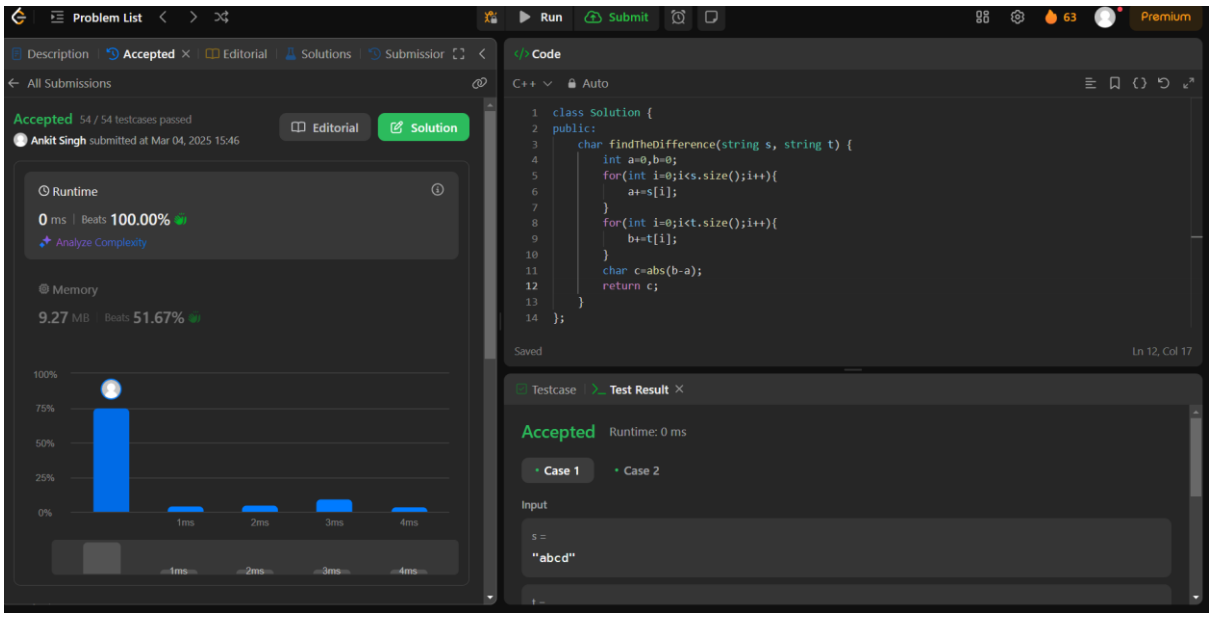
ASSIGNMENT -5 (ADVANCED PROGRAMMING)

Profile: https://leetcode.com/u/AnkitSingh101/

1. **Problem 1:** Find the Difference (389)
2. **Code:**

```
class Solution {
public:
    char findTheDifference(string s, string t) {
        int a=0,b=0;
        for(int i=0;i<s.size();i++){
            a+=s[i];
        }
        for(int i=0;i<t.size();i++){
            b+=t[i];
        }
        char c=abs(b-a);
        return c;
    }
};
```

3. **Output:**

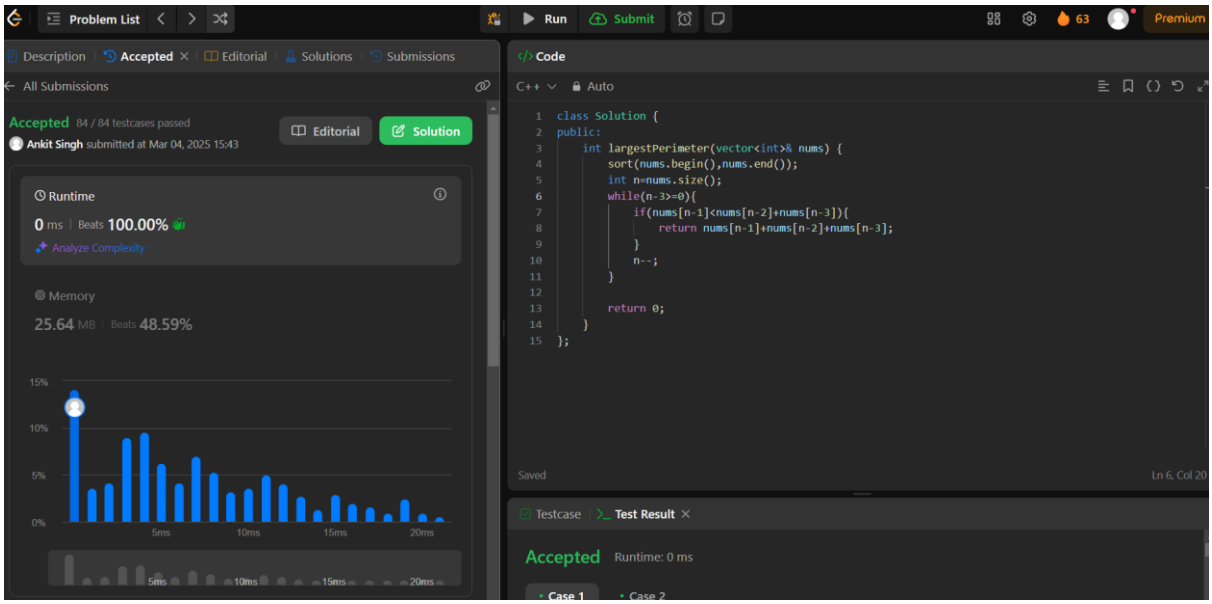


1. **Problem 2: Largest Perimeter Triangle (976)**
2. **Code:**

```
class Solution {
public:
    int largestPerimeter(vector<int>& nums) {
        sort(nums.begin(),nums.end());
        int n=nums.size();
        while(n-3>=0){
            if(nums[n-1]<nums[n-2]+nums[n-3]){
                return nums[n-1]+nums[n-2]+nums[n-3];
            }
            n--;
        }

        return 0;
    }
};
```

3. **Output:**

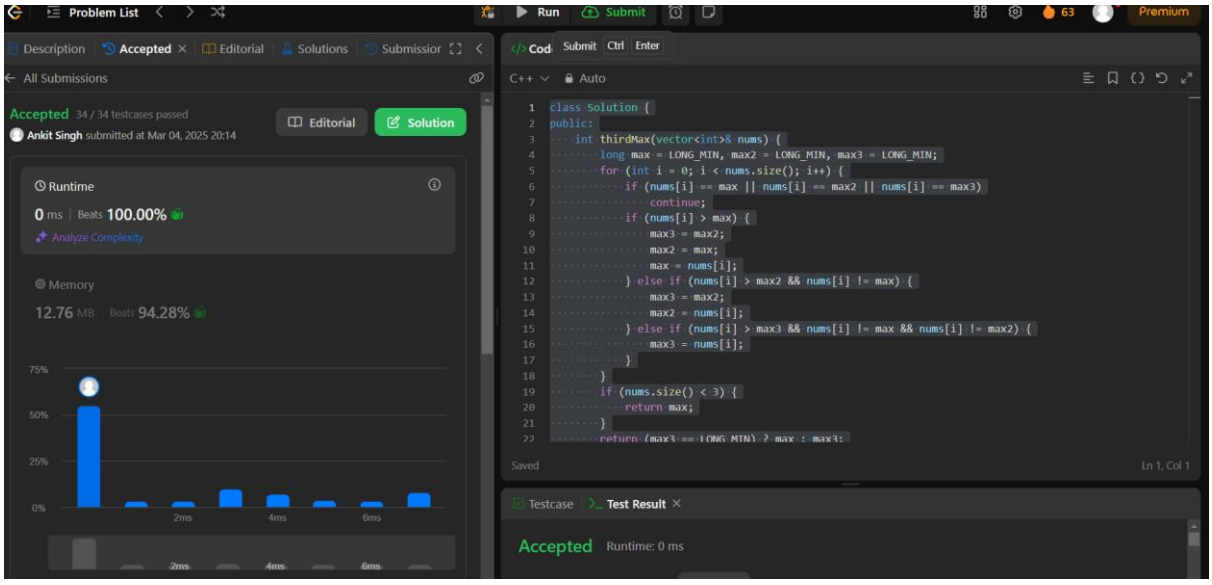


1. **Problem 3: [Third Maximum Number](#) (414)**

2. **Code:**

```
class Solution {
public:
    int thirdMax(vector<int>& nums) {
        long max = LONG_MIN, max2 = LONG_MIN, max3 = LONG_MIN;
        for (int i = 0; i < nums.size(); i++) {
            if (nums[i] == max || nums[i] == max2 || nums[i] == max3)
                continue;
            if (nums[i] > max) {
                max3 = max2;
                max2 = max;
                max = nums[i];
            } else if (nums[i] > max2 && nums[i] != max) {
                max3 = max2;
                max2 = nums[i];
            } else if (nums[i] > max3 && nums[i] != max && nums[i] != max2) {
                max3 = nums[i];
            }
        }
        if (nums.size() < 3) {
            return max;
        }
        return (max3 == LONG_MIN) ? max : max3;
    }
};
```

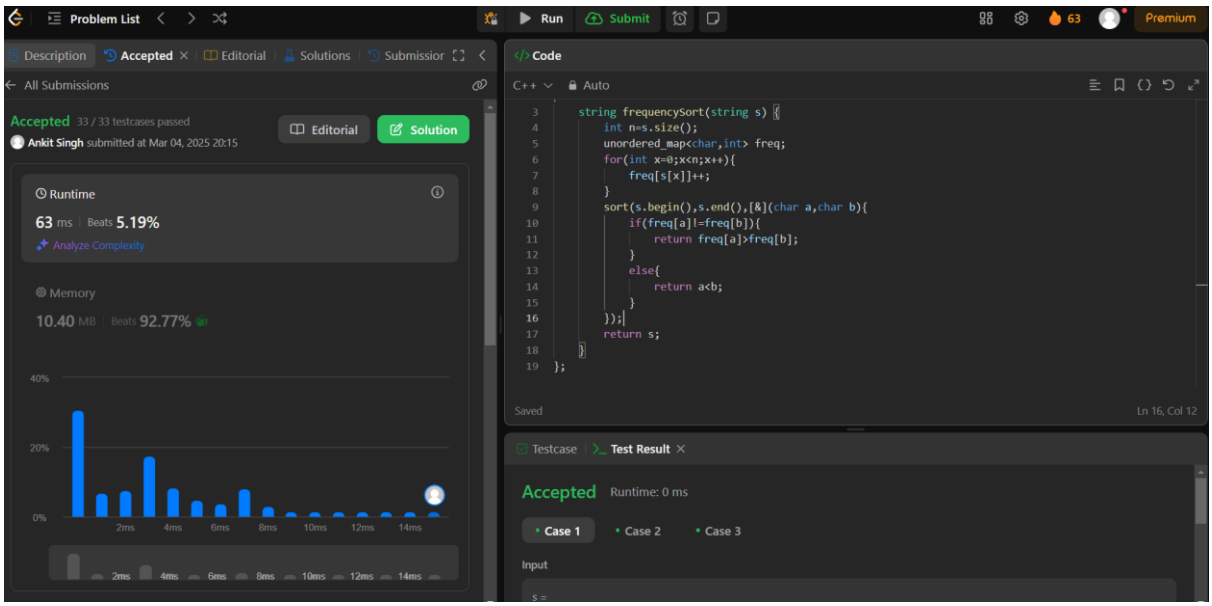
3. **Output:**



1. **Problem 4: Sort Characters By Frequency (451)**
2. **Code:**

```
class Solution {
public:
    string frequencySort(string s) {
        int n=s.size();
        unordered_map<char,int> freq;
        for(int x=0;x<n;x++){
            freq[s[x]]++;
        }
        sort(s.begin(),s.end(),[&](char a,char b){
            if(freq[a]!=freq[b]){
                return freq[a]>freq[b];
            }
            else{
                return a<b;
            }
        });
        return s;
    }
};
```

3. **Output:**



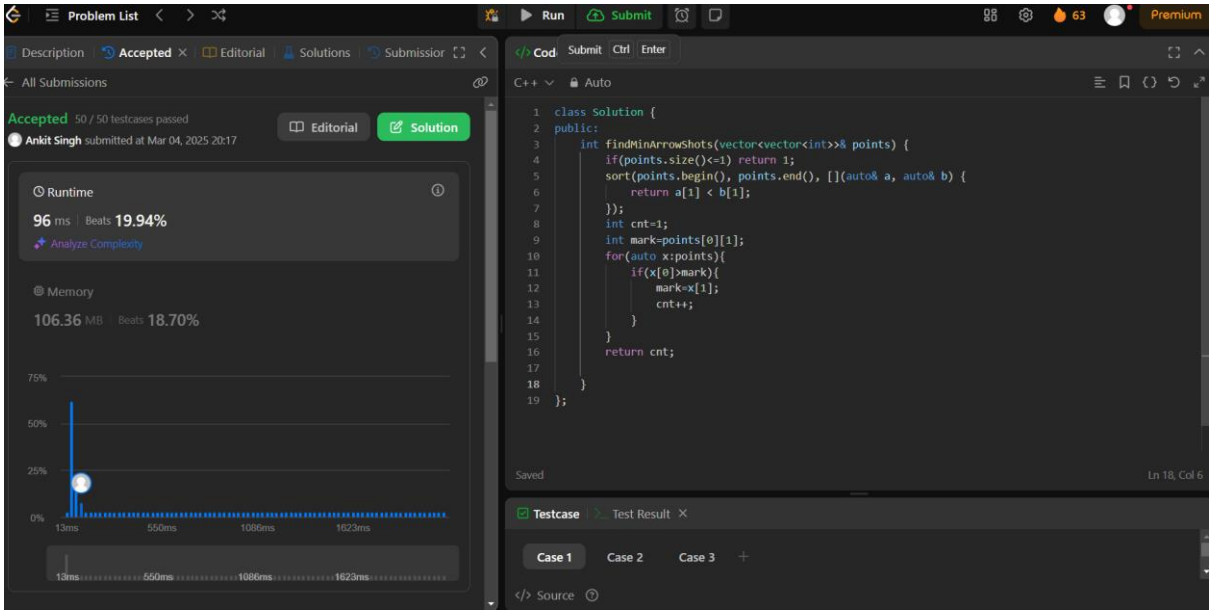
1.Problem 5: Minimum Number of Arrows to Burst Balloons (452)

2. Code:

```
class Solution {
public:
    int findMinArrowShots(vector<vector<int>>& points) {
        if(points.size()<=1) return 1;
        sort(points.begin(), points.end(), [](auto& a, auto& b) {
            return a[1] < b[1];
        });
        int cnt=1;
        int mark=points[0][1];
        for(auto x:points){
            if(x[0]>mark){
                mark=x[1];
                cnt++;
            }
        }
        return cnt;

    }
};
```

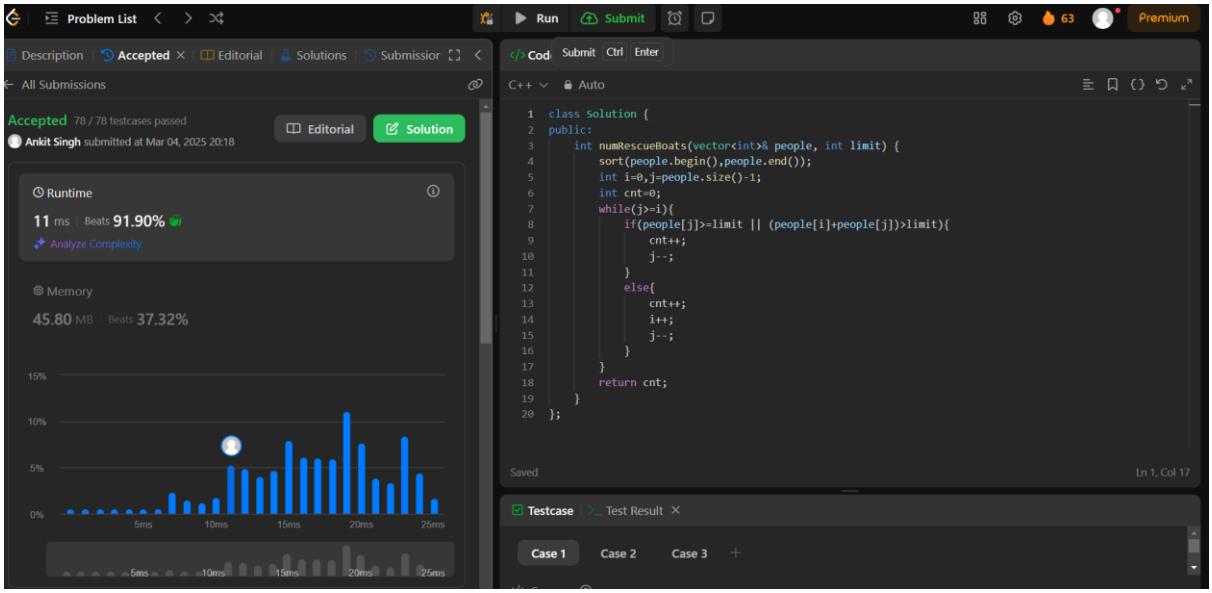
3. Output:



1. **Problem 6: Boats to Save People** (881)
2. **Code:**

```
class Solution {
public:
    int numRescueBoats(vector<int>& people, int limit) {
        sort(people.begin(),people.end());
        int i=0,j=people.size()-1;
        int cnt=0;
        while(j>=i){
            if(people[j]>=limit || (people[i]+people[j])>limit){
                cnt++;
                j--;
            }
            else{
                cnt++;
                i++;
                j--;
            }
        }
        return cnt;
    }
};
```

3. **Output:**



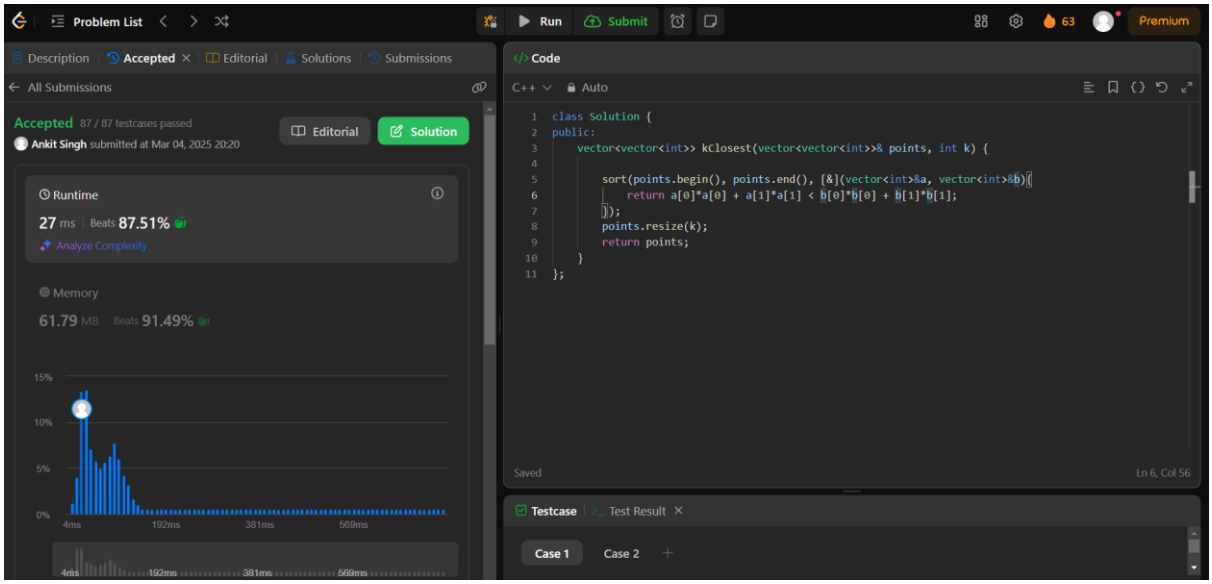
1. **Problem 7: [K Closest Points to Origin](#) (973)**

2. **Code:**

```
class Solution {
public:
    vector<vector<int>> kClosest(vector<vector<int>>& points, int k) {

        sort(points.begin(), points.end(), [&](vector<int>&a, vector<int>&b){
            return a[0]*a[0] + a[1]*a[1] < b[0]*b[0] + b[1]*b[1];
        });
        points.resize(k);
        return points;
    }
};
```

3. **Output:**



1. **Problem 8: Reduce Array Size to The Half** (1338)
2. **Code:**

```
class Solution {
public:
    int minSetSize(vector<int>& arr) {
        unordered_map<int, int> counter;
        priority_queue<int> q;
        int res = 0, removed = 0;

        for (auto a : arr) counter[a]++;
        for (auto c : counter) q.push(c.second);

        while (removed < arr.size() / 2) {
            removed += q.top();
            q.pop();
            res++;
        }

        return res;
    }
};
```

3. **Output:**