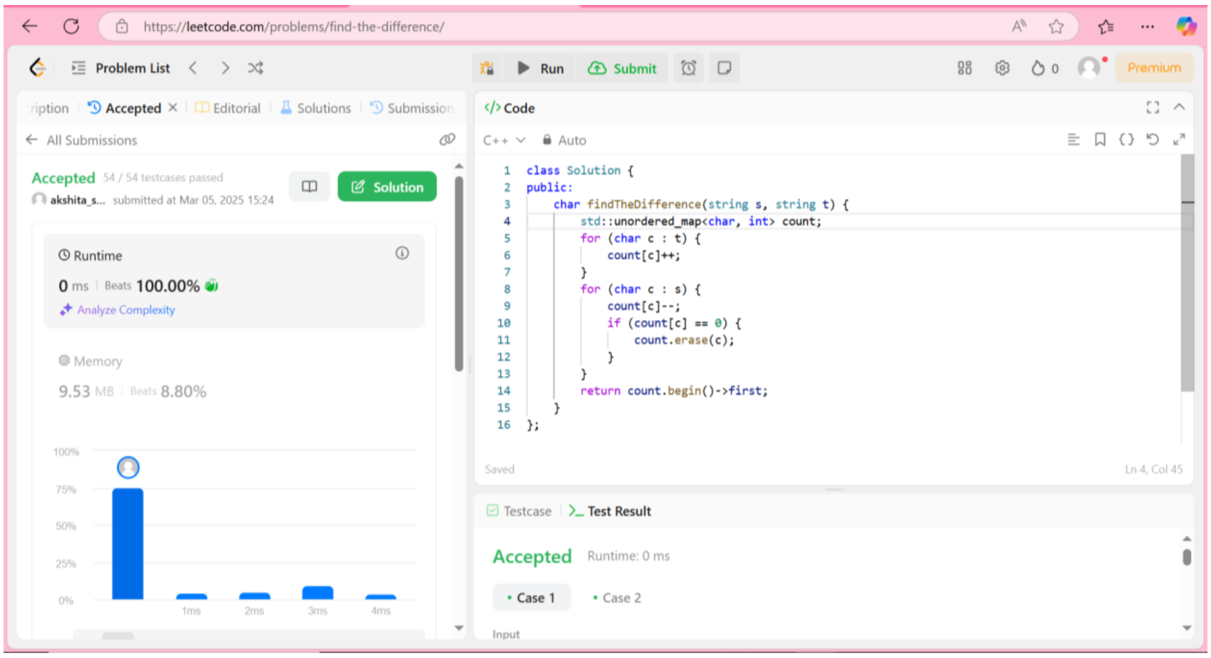
Name : Akshita Sood UID : 22BCS14993 Section : FL-IOT-601-A

389.Find the diffrence

Solution

```
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


https://leetcode.com/problems/find-the-difference/

Accepted 54 / 54 testcases passed
akshita_s... submitted at Mar 05, 2025 15:24

Runtime
0 ms | Beats 100.00%

Memory
9.53 MB | Beats 8.80%

```
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;
    }
};
```

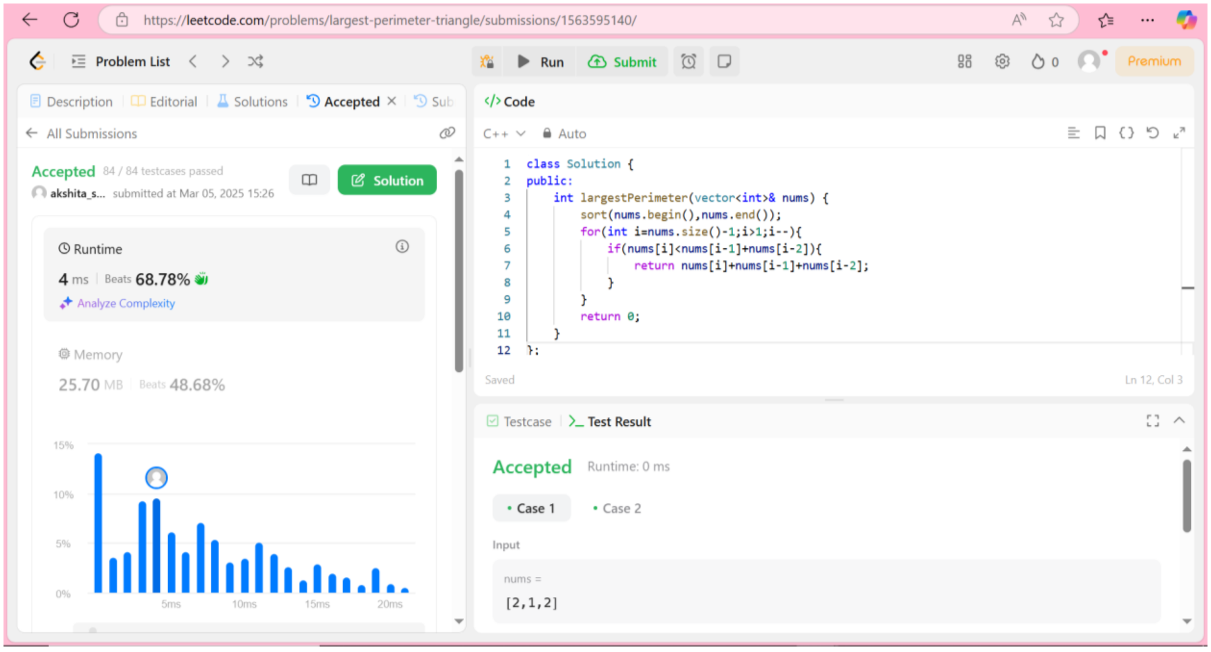
Test Result
Accepted Runtime: 0 ms
Case 1 Case 2

976.Largest Perimeter Triangle

Solution

```
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


https://leetcode.com/problems/largest-perimeter-triangle/submissions/1563595140/

Accepted 84 / 84 testcases passed
akshita_s... submitted at Mar 05, 2025 15:26

Runtime
4 ms | Beats 68.78%

Memory
25.70 MB | Beats 48.68%

C++ Auto

```
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;
    }
};
```

Saved Ln 12, Col 3

Accepted Runtime: 0 ms

Case 1 Case 2

Input

nums =

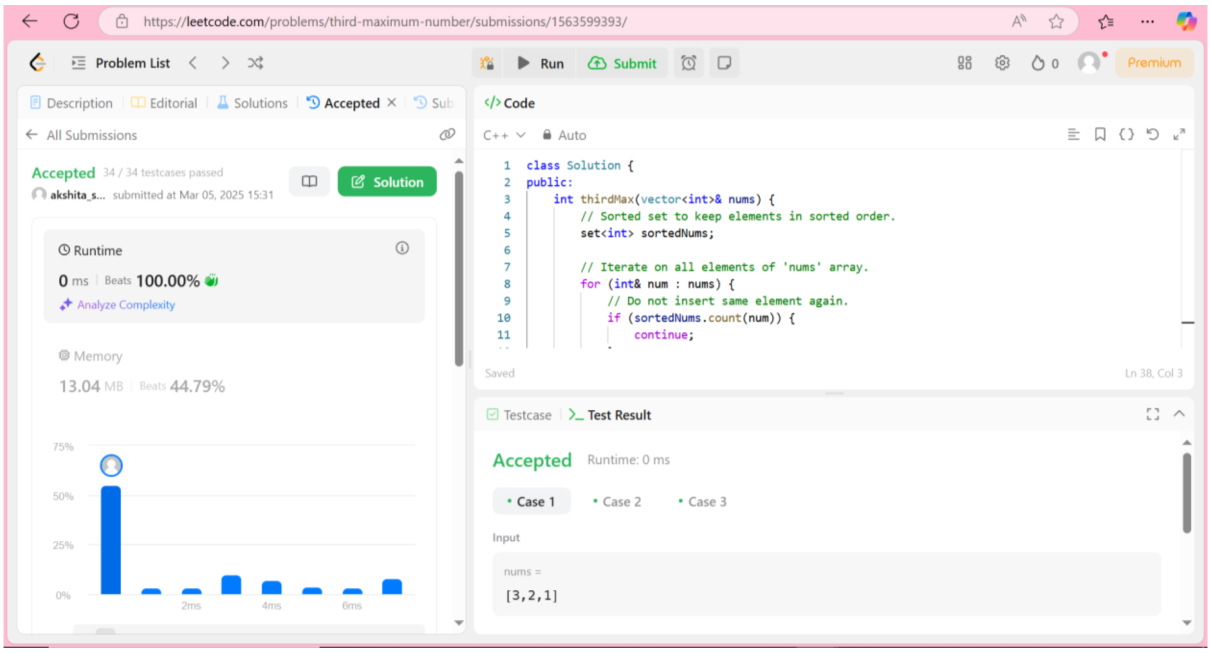
```
[2,1,2]
```

414.Third Maximum Number

Solution

```
class Solution {
public:
    int thirdMax(vector<int>& nums) {
        // Sorted set to keep elements in sorted order.
        set<int> sortedNums;

        // Iterate on all elements of 'nums' array.
        for (int& num : nums) {
            // Do not insert same element again.
            if (sortedNums.count(num)) {
                continue;
            }

            // If sorted set has 3 elements.
            if (sortedNums.size() == 3) {
                // And the smallest element is smaller than current element.
                if (*sortedNums.begin() < num) {
                    // Then remove the smallest element and push the current
element.
                    sortedNums.erase(sortedNums.begin());
                    sortedNums.insert(num);
                }

            }
            // Otherwise push the current element of nums array.
            else {
                sortedNums.insert(num);
            }
        }

        // If sorted set has three elements return the smallest among those 3.
        if (sortedNums.size() == 3) {
            return *sortedNums.begin();
        }

        // Otherwise return the biggest element of nums array.
        return *sortedNums.rbegin();
    }
};
```

Output



Accepted 34 / 34 testcases passed
akshita_s... submitted at Mar 05, 2025 15:31

Runtime
0 ms | Beats 100.00%
Analyze Complexity

Memory
13.04 MB | Beats 44.79%

```
class Solution {
public:
    int thirdMax(vector<int>& nums) {
        // Sorted set to keep elements in sorted order.
        set<int> sortedNums;

        // Iterate on all elements of 'nums' array.
        for (int& num : nums) {
            // Do not insert same element again.
            if (sortedNums.count(num)) {
                continue;
```

Saved
Ln 38, Col 3

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums =
[3,2,1]

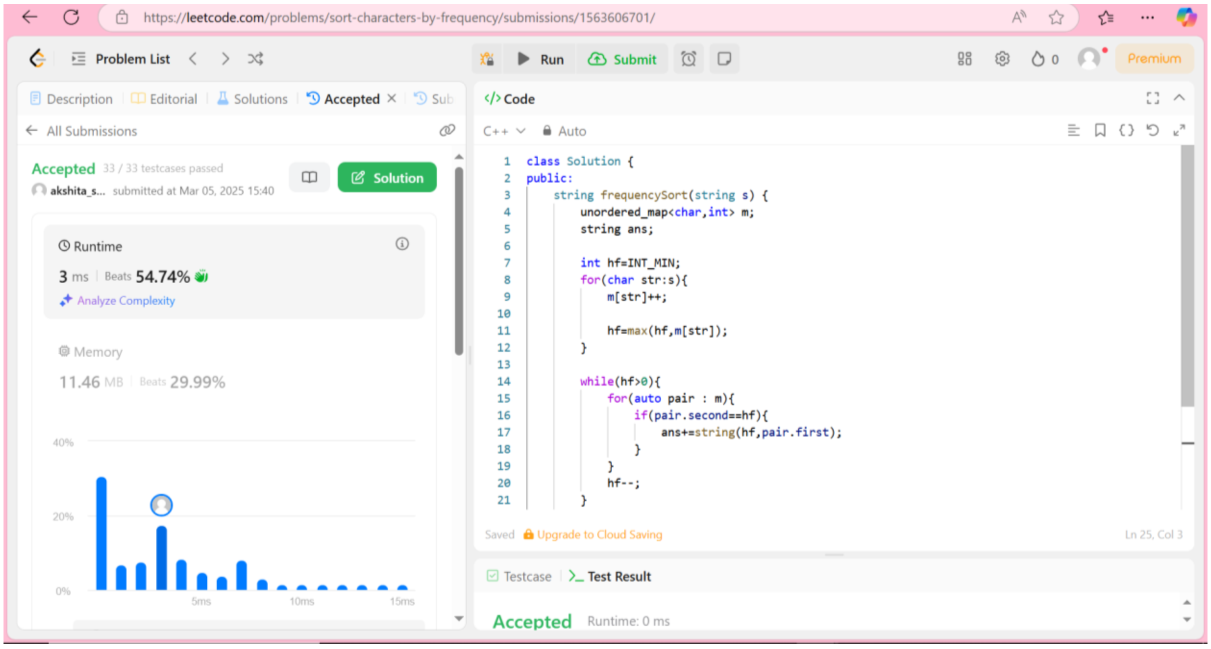
451.Sort Characters By Frequency

Solution

```
class Solution {
```

```
public:
    string frequencySort(string s) {
        unordered_map<char,int> m;
        string ans;

        int hf=INT_MIN;
        for(char str:s){
            m[str]++;

            hf=max(hf,m[str]);
        }

        while(hf>0){
            for(auto pair : m){
                if(pair.second==hf){
                    ans+=string(hf,pair.first);
                }
            }
            hf--;
        }

        return ans;
    }
};
```

Output



https://leetcode.com/problems/sort-characters-by-frequency/submissions/1563606701/

Accepted 33 / 33 testcases passed
akshita_s... submitted at Mar 05, 2025 15:40

Runtime
3 ms | Beats 54.74%

Memory
11.46 MB | Beats 29.99%

```
class Solution {
public:
    string frequencySort(string s) {
        unordered_map<char,int> m;
        string ans;

        int hf=INT_MIN;
        for(char str:s){
            m[str]++;

            hf=max(hf,m[str]);
        }

        while(hf>0){
            for(auto pair : m){
                if(pair.second==hf){
                    ans+=string(hf,pair.first);
                }
            }
            hf--;
        }
```

Saved   Ln 25, Col 3

Accepted   Runtime: 0 ms

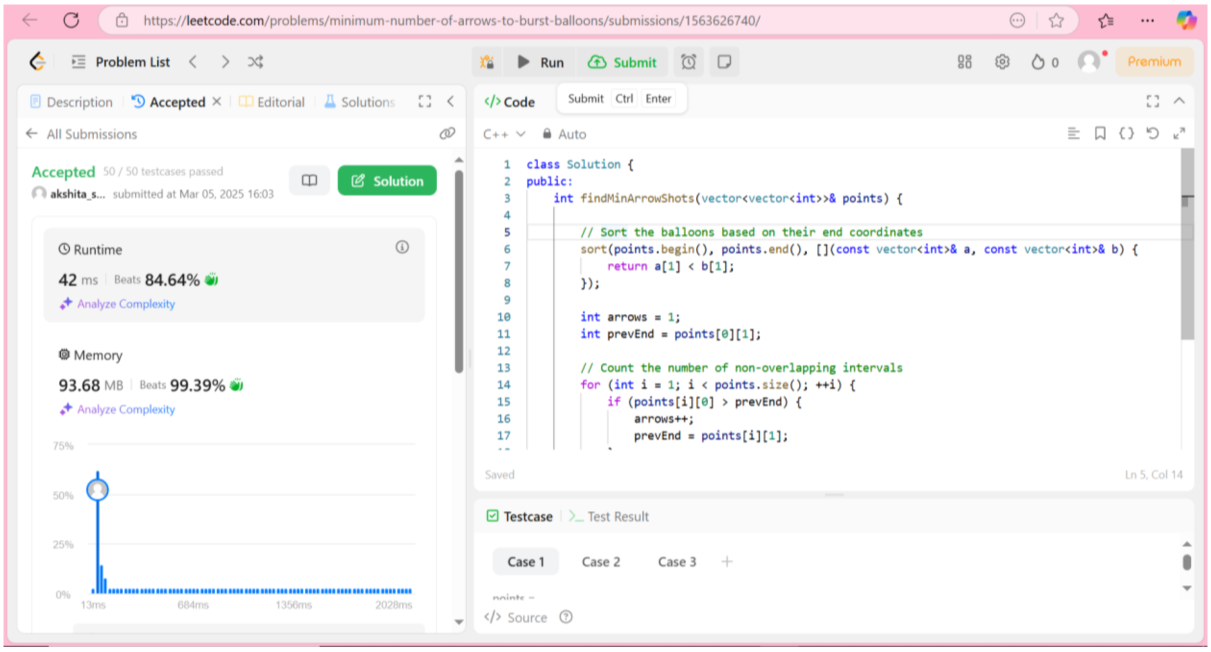
452.Minimum Number of Arrows to Burst Balloons

Solution

```
class Solution {
```

```cpp
public:
    int findMinArrowShots(vector<vector<int>>& points) {

        // Sort the balloons based on their end coordinates
        sort(points.begin(), points.end(), [](const vector<int>& a, const
vector<int>& b) {
            return a[1] < b[1];
        });

        int arrows = 1;
        int prevEnd = points[0][1];

        // Count the number of non-overlapping intervals
        for (int i = 1; i < points.size(); ++i) {
            if (points[i][0] > prevEnd) {
                arrows++;
                prevEnd = points[i][1];
            }
        }

        return arrows;
    }
};
```

Output



https://leetcode.com/problems/minimum-number-of-arrows-to-burst-balloons/submissions/1563626740/

Accepted 50 / 50 testcases passed
akshita_s... submitted at Mar 05, 2025 16:03

Runtime
42 ms | Beats 84.64%

Memory
93.68 MB | Beats 99.39%

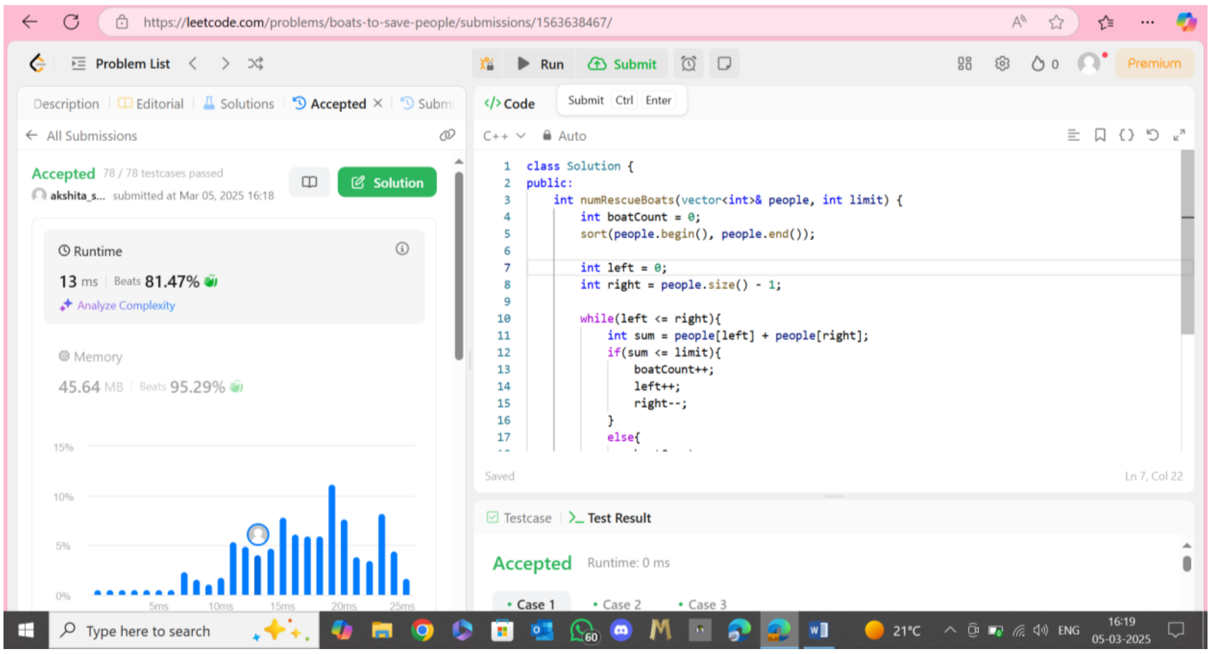
```cpp
class Solution {
public:
    int findMinArrowShots(vector<vector<int>>& points) {

        // Sort the balloons based on their end coordinates
        sort(points.begin(), points.end(), [](const vector<int>& a, const vector<int>& b) {
            return a[1] < b[1];
        });

        int arrows = 1;
        int prevEnd = points[0][1];

        // Count the number of non-overlapping intervals
        for (int i = 1; i < points.size(); ++i) {
            if (points[i][0] > prevEnd) {
                arrows++;
                prevEnd = points[i][1];
```

881.[Boats to Save People]()

Solution

```
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;
    }
};
```

Output


https://leetcode.com/problems/boats-to-save-people/submissions/1563638467/

Accepted 78 / 78 testcases passed
akshita_s... submitted at Mar 05, 2025 16:18

Runtime
13 ms | Beats 81.47%

Memory
45.64 MB | Beats 95.29%

```
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{
```

Saved   Ln 7, Col 22

Accepted Runtime: 0 ms

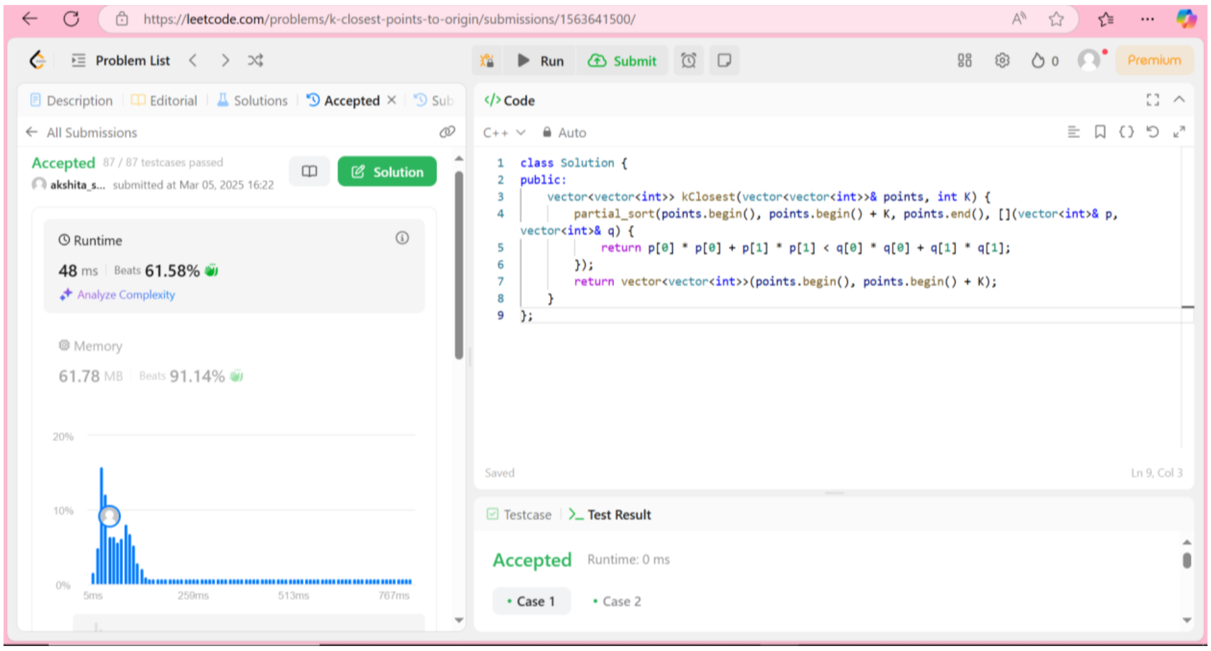
973.[K Closest Points to Origin]

Solution

```
class Solution {
```

```cpp
public:
    vector<vector<int>> kClosest(vector<vector<int>>& points, int K) {
        partial_sort(points.begin(), points.begin() + K, points.end(),
[](vector<int>& p, vector<int>& q) {
            return p[0] * p[0] + p[1] * p[1] < q[0] * q[0] + q[1] * q[1];
        });
        return vector<vector<int>>(points.begin(), points.begin() + K);
    }
};
```

Output



https://leetcode.com/problems/k-closest-points-to-origin/submissions/1563641500/

Accepted 87 / 87 testcases passed

akshita_s... submitted at Mar 05, 2025 16:22

Runtime 48 ms | Beats 61.58%

Memory 61.78 MB | Beats 91.14%

```cpp
class Solution {
public:
    vector<vector<int>> kClosest(vector<vector<int>>& points, int K) {
        partial_sort(points.begin(), points.begin() + K, points.end(), [](vector<int>& p,
vector<int>& q) {
            return p[0] * p[0] + p[1] * p[1] < q[0] * q[0] + q[1] * q[1];
        });
        return vector<vector<int>>(points.begin(), points.begin() + K);
    }
};
```

Test Result

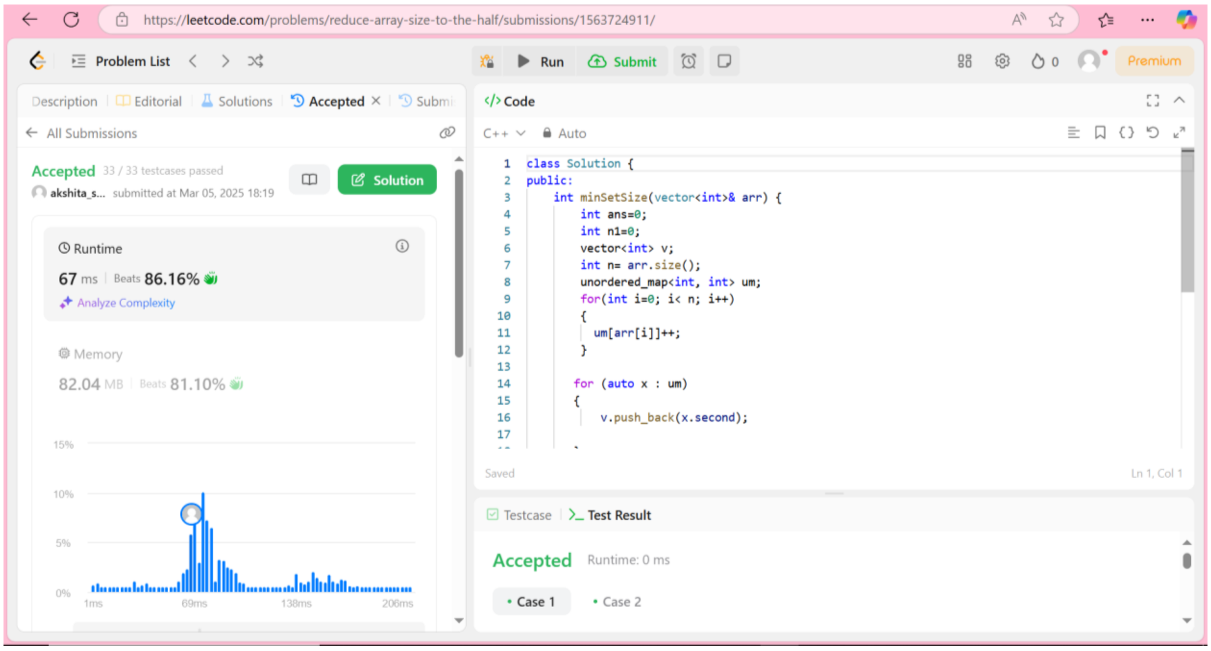
Accepted Runtime: 0 ms

1338.Reduce Array Size to The Half

Solution

```cpp
class Solution {
public:
    int minSetSize(vector<int>& arr) {
        int ans=0;
        int n1=0;
        vector<int> v;
        int n= arr.size();
        unordered_map<int, int> um;
        for(int i=0; i< n; i++)
        {
          um[arr[i]]++;
        }

       for (auto x : um)
       {
           v.push_back(x.second);
       }
        if(v.size()==1)return 1;
        sort(v.begin(), v.end());
        for(int i=v.size()-1; i>=0; i--)
        {
            if(ans>=n/2){
              return n1;
        }
        n1++;
        ans= ans+ v[i];

        }
        return 0;
    }
};
```

Output



Accepted 33 / 33 testcases passed
akshita_s... submitted at Mar 05, 2025 18:19

Runtime
67 ms | Beats 86.16%

Memory
82.04 MB | Beats 81.10%

```
class Solution {
public:
    int minSetSize(vector<int>& arr) {
        int ans=0;
        int n1=0;
        vector<int> v;
        int n= arr.size();
        unordered_map<int, int> um;
        for(int i=0; i< n; i++)
        {
          um[arr[i]]++;
        }

       for (auto x : um)
       {
           v.push_back(x.second);
```

Accepted Runtime: 0 ms
Case 1 Case 2