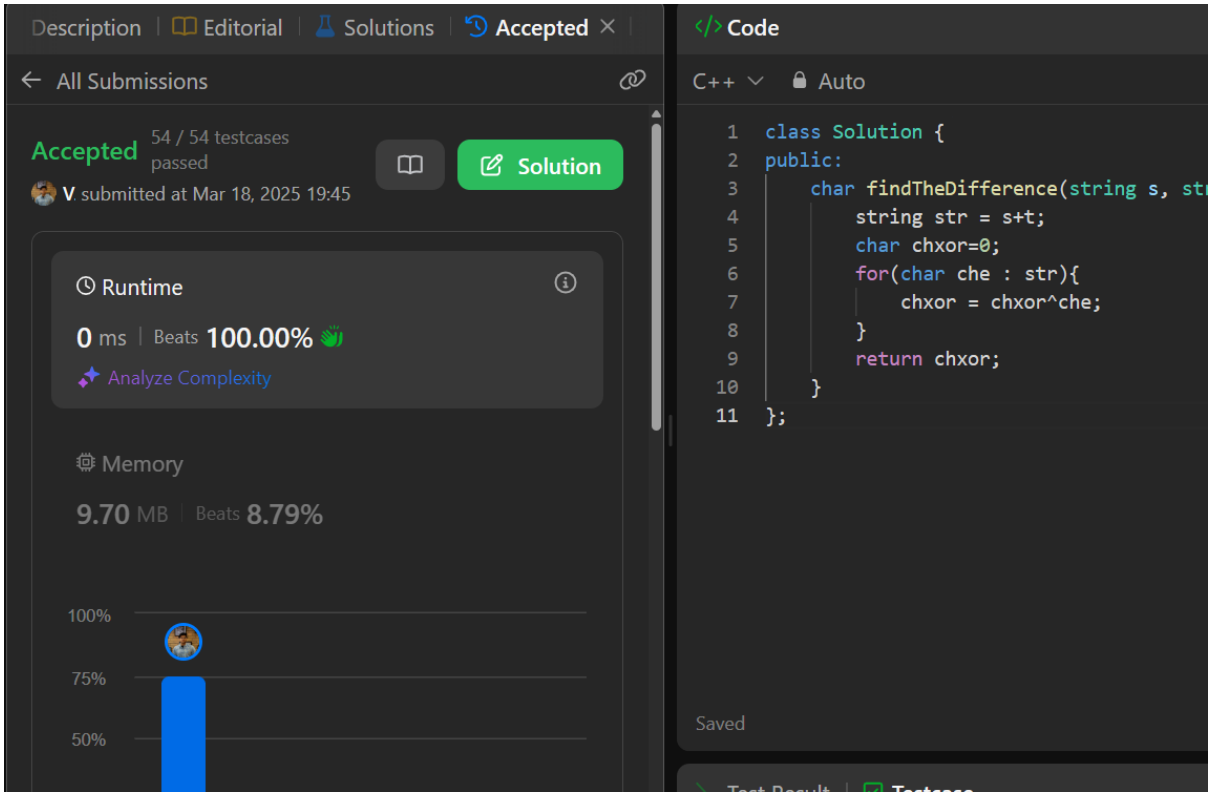
389.Find the diffrence

- **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:**

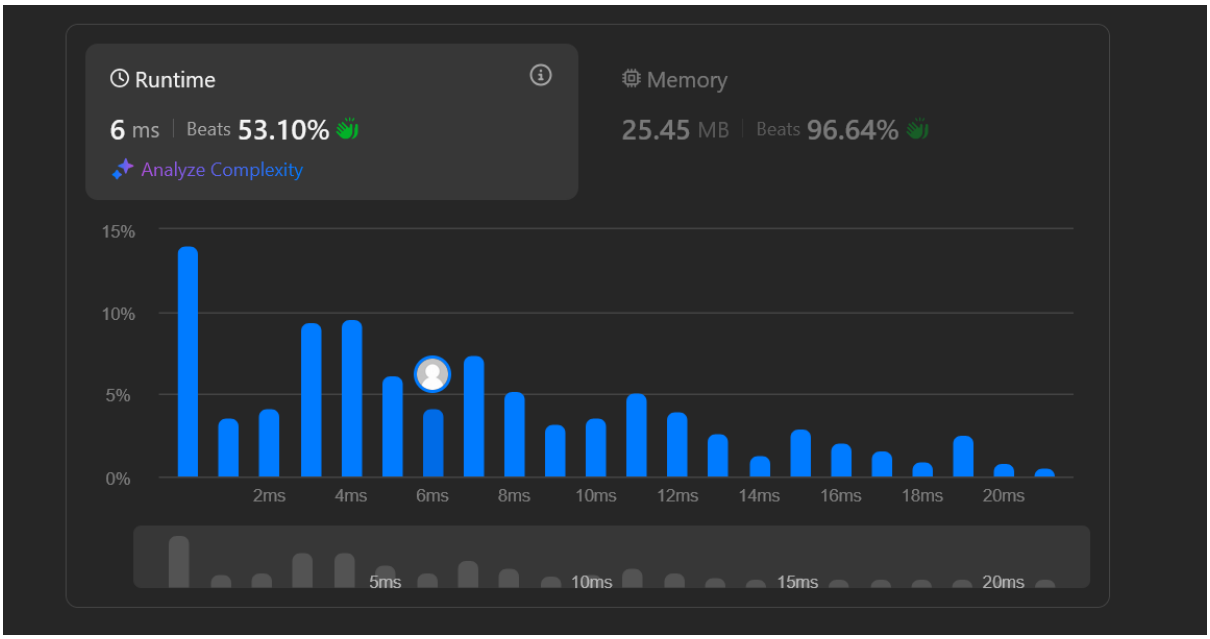


976.Largest Perimeter Triangle

- **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

- **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:**

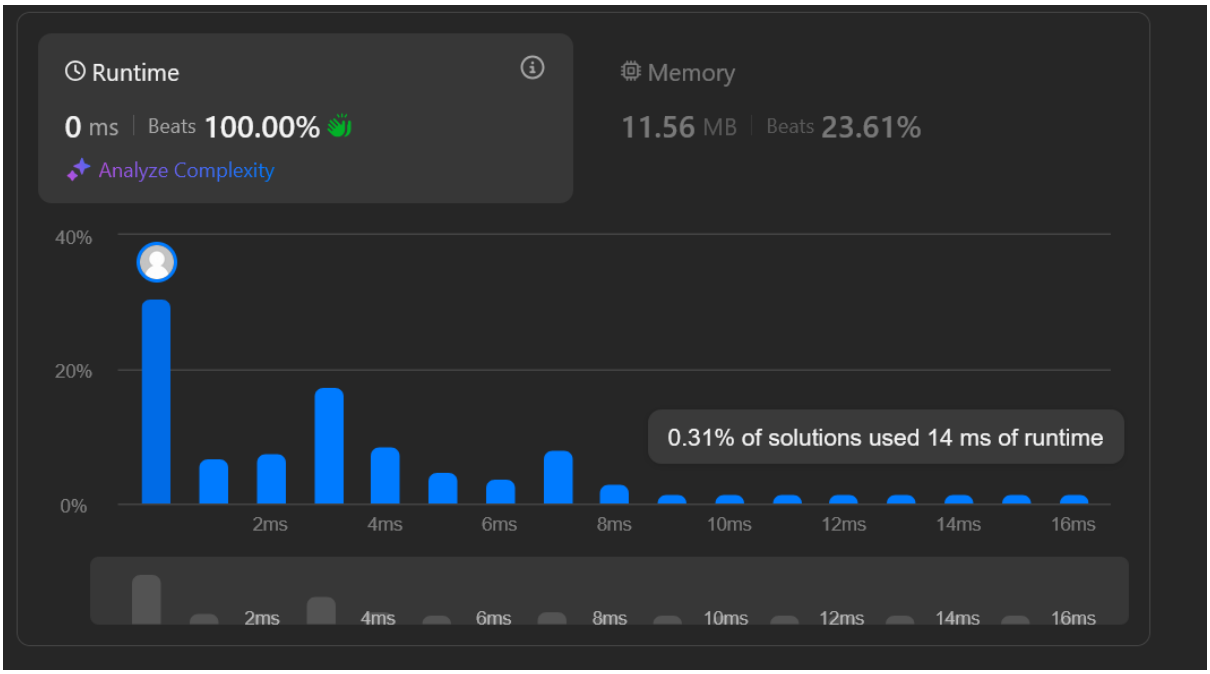
-
-

451.Sort Characters By Frequency

- **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:**

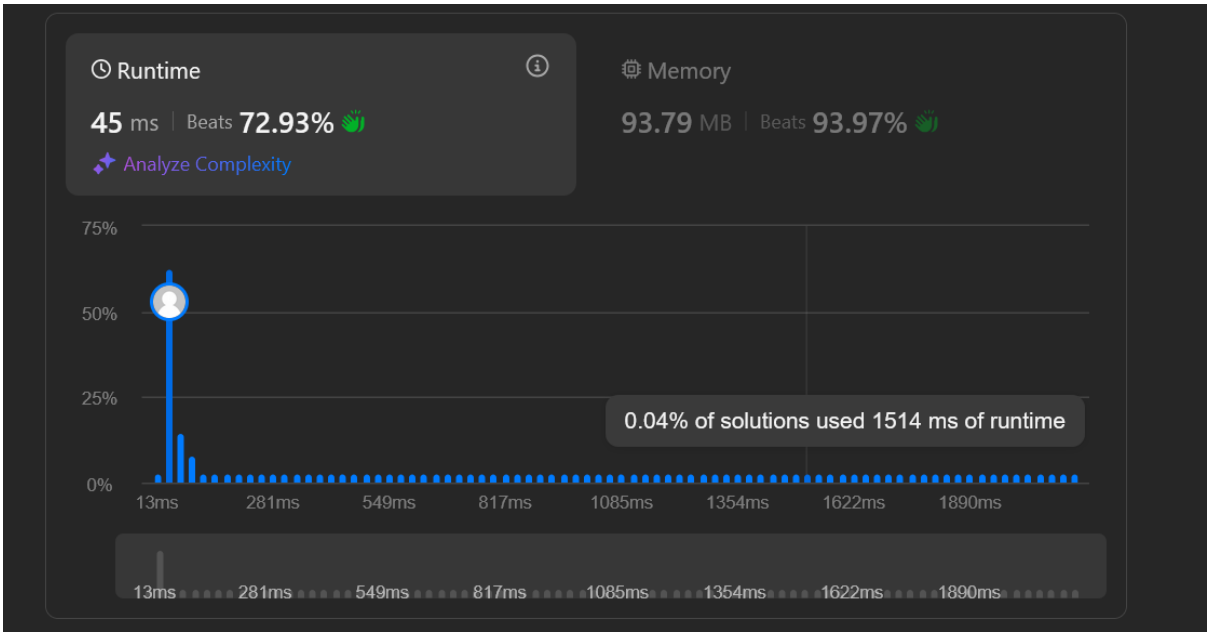

Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
11.56 MB | Beats 23.61%
40%
20%
0%
0.31% of solutions used 14 ms of runtime
2ms
4ms
6ms
8ms
10ms
12ms
14ms
16ms

452.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:**

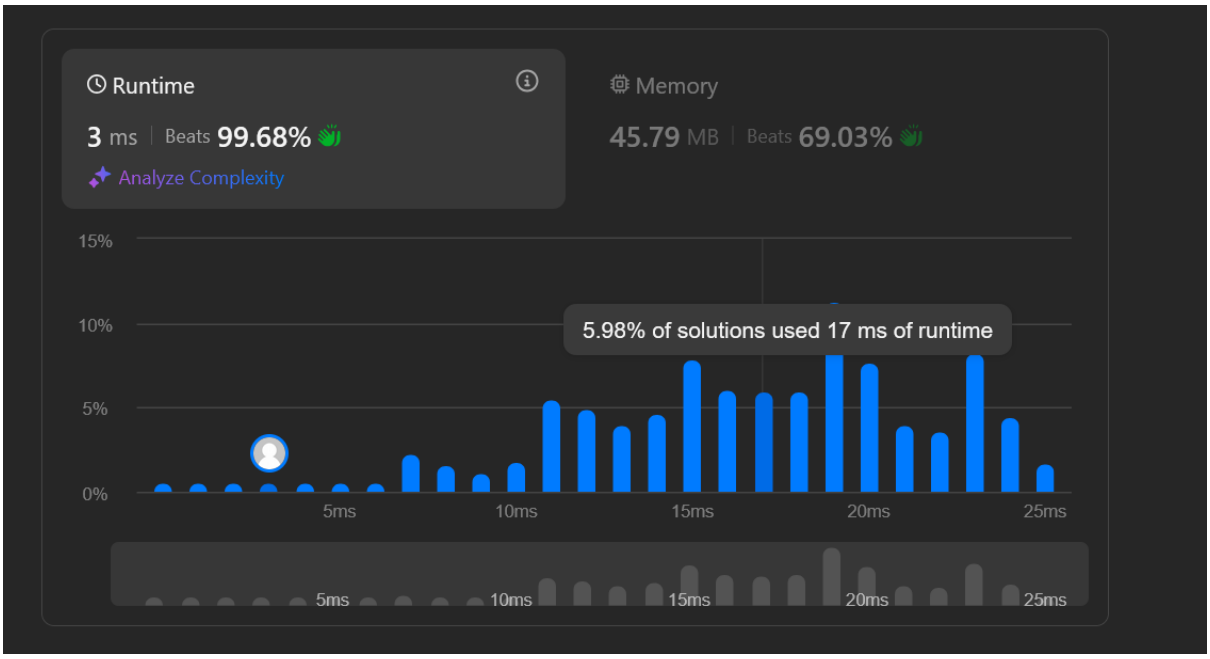


881.Boats to Save People

- **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';
}();
```

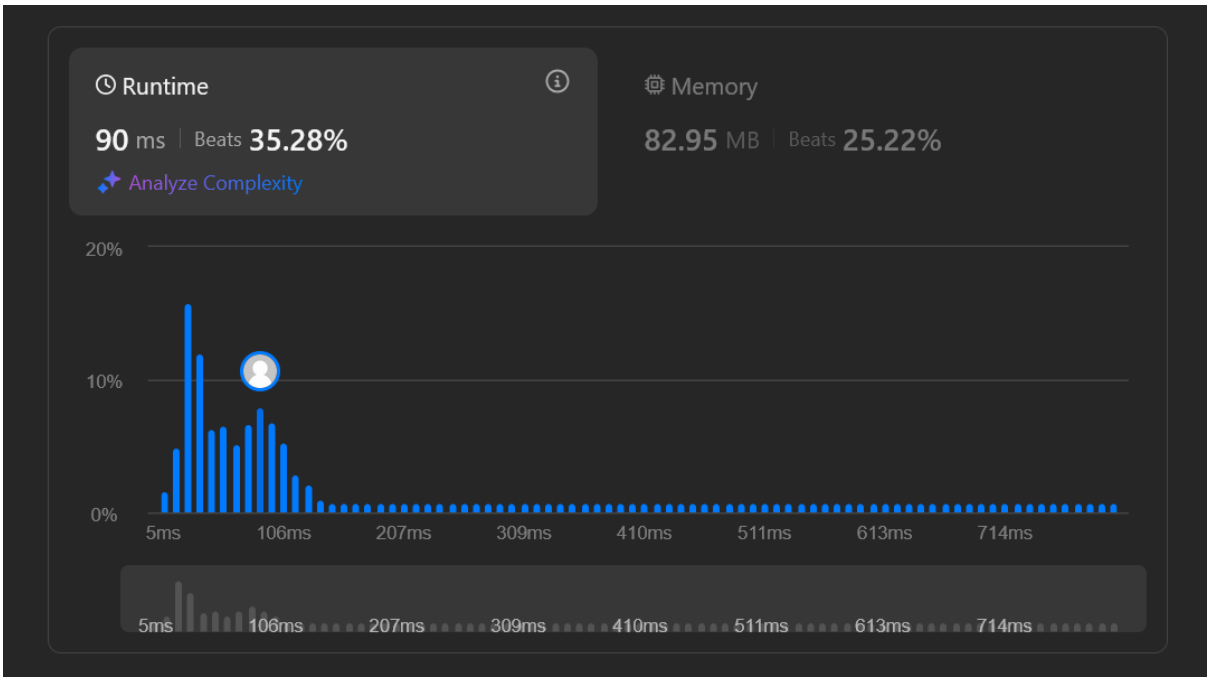
- **Screenshot:**


Runtime
3 ms | Beats 99.68%
Analyze Complexity
Memory
45.79 MB | Beats 69.03%
15%
10%
5%
0%
5.98% of solutions used 17 ms of runtime
5ms
10ms
15ms
20ms
25ms
5ms
10ms
15ms
20ms
25ms

973.K Closest Points to Origin

- **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:**
- 
Runtime
90 ms | Beats 35.28%
Analyze Complexity
Memory
82.95 MB | Beats 25.22%
20%
10%
0%
5ms
106ms
207ms
309ms
410ms
511ms
613ms
714ms

1338.Reduce Array Size to The Half

- **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:**