

Name: Semit Turkey

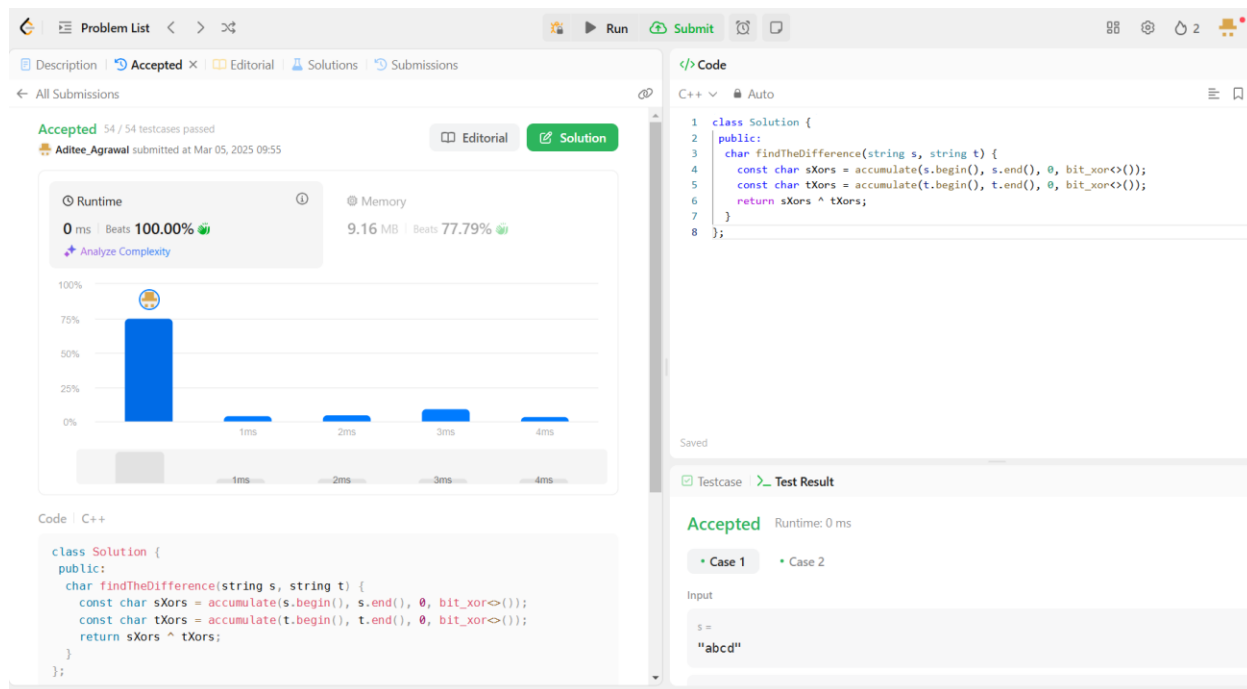
Uid : 22BCS13024

Fl_lot 601 A

AP assignment 5

1. Find the difference

```
class Solution {
public:
    char findTheDifference(string s, string t) {
        const char sXors = accumulate(s.begin(), s.end(), 0, bit_xor<>());
        const char tXors = accumulate(t.begin(), t.end(), 0, bit_xor<>());
        return sXors ^ tXors;
    }
};
```



2. Largest Perimeter Triangle

```
class Solution {
public:
    int largestPerimeter(vector<int>& nums) {
        ranges::sort(nums);

        for (int i = nums.size() - 1; i > 1; --i)
```

```

        if (nums[i - 2] + nums[i - 1] > nums[i])
            return nums[i - 2] + nums[i - 1] + nums[i];

    return 0;
}
};

```

Accepted 84 / 84 testcases passed
 Aditee_Agrawal submitted at Mar 05, 2025 10:07

Runtime: 8 ms | Beats 41.75%
 Memory: 25.41 MB | Beats 96.61%

Code | C++

```

class Solution {
public:
    int largestPerimeter(vector<int>& nums) {
        ranges::sort(nums);
        for (int i = nums.size() - 1; i > 1; --i)
            if (nums[i - 2] + nums[i - 1] > nums[i])
                return nums[i - 2] + nums[i - 1] + nums[i];
        return 0;
    }
};

```

Accepted Runtime: 0 ms

Case 1 • Case 2

Input

nums =
 [2,1,2]

3. Third Maximum Number

```

class Solution {
public:
    int thirdMax(vector<int>& nums) {
        long max1 = LONG_MIN;
        long max2 = LONG_MIN;
        long max3 = LONG_MIN;

        for (const int num : nums)
            if (num > max1) {
                max3 = max2;
                max2 = max1;
                max1 = num;
            } else if (max1 > num && num > max2) {
                max3 = max2;
                max2 = num;
            } else if (max2 > num && num > max3) {
                max3 = num;
            }
    }
};

```

```

    }

    return max3 == LONG_MIN ? max1 : max3;
}
};

```

← All Submissions

Accepted 34 / 34 testcases passed
 Aditee_Agrawal submitted at Mar 05, 2025 10:11

Editorial Solution

Runtime 0 ms | Beats 100.00%
 Memory 12.87 MB | Beats 79.72%

Analyze Complexity

Code | C++

```

class Solution {
public:
    int thirdMax(vector<int>& nums) {
        long max1 = LONG_MIN;
        long max2 = LONG_MIN;
        long max3 = LONG_MIN;

        for (const int num : nums)

```

Saved

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums =
 [3,2,1]

4. Sort Characters By Frequency

```

class Solution {
public:
    string frequencySort(string s) {
        const int n = s.length();
        string ans;
        vector<int> count(128);
        vector<vector<char>> buckets(n + 1);
        for (const char c : s)
            ++count[c];
        for (int i = 0; i < 128; ++i) {
            const int freq = count[i];
            if (freq > 0)
                buckets[freq].push_back((char)i);
        }
        for (int freq = n; freq > 0; --freq)
            for (const char c : buckets[freq])
                ans += string(freq, c);

        return ans;
    }
}

```

```
};
```

Accepted 33 / 33 testcases passed
Aditee_Agrawal submitted at Mar 05, 2025 10:13

Runtime: 4 ms | Beats 37.19%
Memory: 17.32 MB | Beats 5.29%

Code | C++

```
class Solution {
public:
    string frequencySort(string s) {
        const int n = s.length();
        string ans;
        vector<int> count(128);
        vector<vector<char>> buckets(n + 1);
        for (const char c : s)
            ++count[c];
        for (int i = 0; i < 128; ++i) {
            const int freq = count[i];
            if (freq > 0)
                buckets[freq].push_back((char)i);
        }
        for (int freq = n; freq > 0; --freq)
            for (const char c : buckets[freq])
                ans += string(freq, c);
        return ans;
    }
};
```

Testcase Test Result
Accepted Runtime: 0 ms
Case 1 Case 2 Case 3
Input
s =
"tree"

5. Minimum Number of Arrows to Burst Balloons

```
class Solution {
public:
    int findMinArrowShots(vector<vector<int>>& points) {
        ranges::sort(points,
            [](const auto& a, const auto& b) { return a[1] < b[1]; });
        int ans = 1;
        int arrowX = points[0][1];

        for (int i = 1; i < points.size(); ++i)
            if (points[i][0] > arrowX) {
                arrowX = points[i][1];
                ++ans;
            }
        return ans;
    }
};
```

Problem List < > < Run Submit < >

Description Accepted x Editorial Solutions Submissions

All Submissions

Accepted 50 / 50 testcases passed
Aditee_Agrawal submitted at Mar 05, 2025 10:16

Editorial Solution

Runtime 49 ms Beats 56.29%
Memory 94.02 MB Beats 28.84%

Analyze Complexity

Code C++ Auto

```

1 class Solution {
2 public:
3     int findMinArrowShots(vector<vector<int>>& points) {
4         ranges::sort(points,
5             [](const auto& a, const auto& b) { return a[1] < b[1]; });
6         int ans = 1;
7         int arrowX = points[0][1];
8
9         for (int i = 1; i < points.size(); ++i)
10             if (points[i][0] > arrowX) {
11                 arrowX = points[i][1];
12                 ++ans;
13             }
14         return ans;
15     }
16 };

```

Saved

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

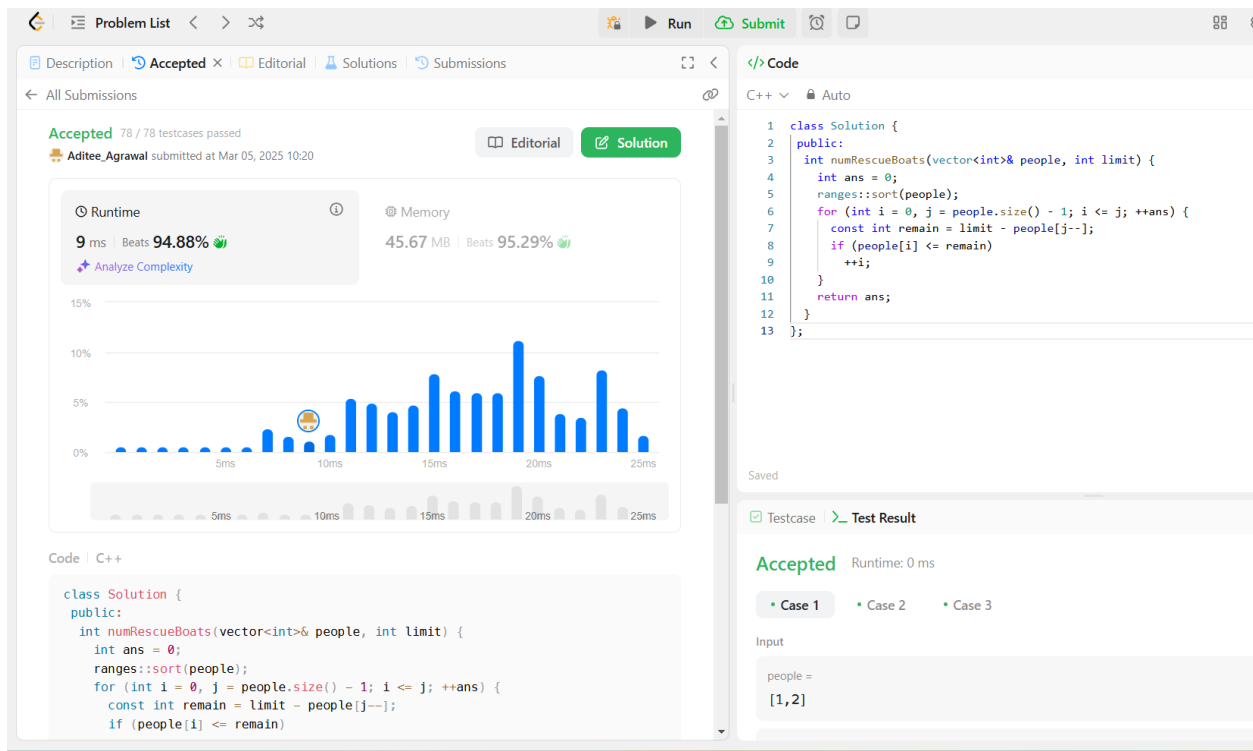
points =
[[10,16],[2,8],[1,6],[7,12]]

6. [Boats to Save People](#)

```

class Solution {
public:
    int numRescueBoats(vector<int>& people, int limit) {
        int ans = 0;
        ranges::sort(people);
        for (int i = 0, j = people.size() - 1; i <= j; ++ans) {
            const int remain = limit - people[j--];
            if (people[i] <= remain)
                ++i;
        }
        return ans;
    }
};

```



7. [K Closest Points to Origin](#)

```
class Solution {
public:
    vector<vector<int>> kClosest(vector<vector<int>>& points, int k) {
        vector<vector<int>> ans;
        auto compare = [&](const vector<int>& a, const vector<int>& b) {
            return squareDist(a) < squareDist(b);
        };
        priority_queue<vector<int>, vector<vector<int>>, decltype(compare)> maxHeap(
            compare);

        for (const vector<int>& point : points) {
            maxHeap.push(point);
            if (maxHeap.size() > k)
                maxHeap.pop();
        }

        while (!maxHeap.empty())
            ans.push_back(maxHeap.top()), maxHeap.pop();

        return ans;
    }
};
```

```
private:
    int squareDist(const vector<int>& p) {
        return p[0] * p[0] + p[1] * p[1];
    }
};
```

Accepted 87 / 87 testcases passed
Aditee_Agrawal submitted at Mar 05, 2025 10:22

Runtime: 99 ms | Beats 28.66%
Memory: 76.38 MB | Beats 43.83%

Code | C++

```
class Solution {
public:
    vector<vector<int>> kClosest(vector<vector<int>>& points, int k) {
        vector<vector<int>> ans;
        auto compare = [&](const vector<int>& a, const vector<int>& b) {
            return squareDist(a) < squareDist(b);
        };
        priority_queue<vector<int>, vector<vector<int>>, decltype(compare)>
            compare);
        for (const vector<int>& point : points) {
            maxHeap.push(point);
            if (maxHeap.size() > k)
                maxHeap.pop();
        }
        while (!maxHeap.empty())
            ans.push_back(maxHeap.top()), maxHeap.pop();
        return ans;
    };
private:
    int squareDist(const vector<int>& p) {
```

Testcase: Accepted Runtime: 0 ms
Case 1 Case 2
Input: points = [[1,3], [-2,2]]

8. Reduce Array Size to The Half

```
class Solution {
public:
    int minSetSize(vector<int>& arr) {
        const int n = arr.size();
        int sum = 0;
        unordered_map<int, int> count;
        vector<pair<int, int>> numAndFreqs;
        for (const int a : arr)
            ++count[a];
        for (const auto& [a, freq] : count)
            numAndFreqs.emplace_back(a, freq);
        ranges::sort(
            numAndFreqs, ranges::greater{},
```

```

        [](const pair<int, int>& numAndFreq) { return
numAndFreq.second; });
    for (int i = 0; i < numAndFreqs.size(); ++i) {
        sum += numAndFreqs[i].second;
        if (sum >= n / 2)
            return i + 1;
    }
    throw;
}
};

```

Description
Accepted
Editorial
Solutions
Submissions

All Submissions

Accepted 33 / 33 testcases passed
Aditee_Agrawal submitted at Mar 05, 2025 10:27

Runtime
76 ms | Beats 64.36%

Memory
84.93 MB | Beats 34.05%

Code | C++

```

class Solution {
public:
    int minSetSize(vector<int>& arr) {
        const int n = arr.size();
        int sum = 0;
        unordered_map<int, int> count;
        vector<pair<int, int>> numAndFreqs;
        for (const int a : arr)
            ++count[a];
        for (const auto& [a, freq] : count)
            numAndFreqs.emplace_back(a, freq);
        ranges::sort(
            numAndFreqs, ranges::greater{},
            [](const pair<int, int>& numAndFreq) { return numAndFreq.second; });
        for (int i = 0; i < numAndFreqs.size(); ++i) {
            sum += numAndFreqs[i].second;
            if (sum >= n / 2)
                return i + 1;
        }
        throw;
    }
};

```

Testcase
Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input
arr = [3,3,3,3,5,5,2,2,7]