



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment 5

Student Name: Kethrin Naharwal

Branch: BE-CSE

Semester: 6th

Subject Name: AP LAB-II

UID: 22BCS12653

Section: 22BCS_FL_IOT-603/A

Date of Performance: 04/03/25

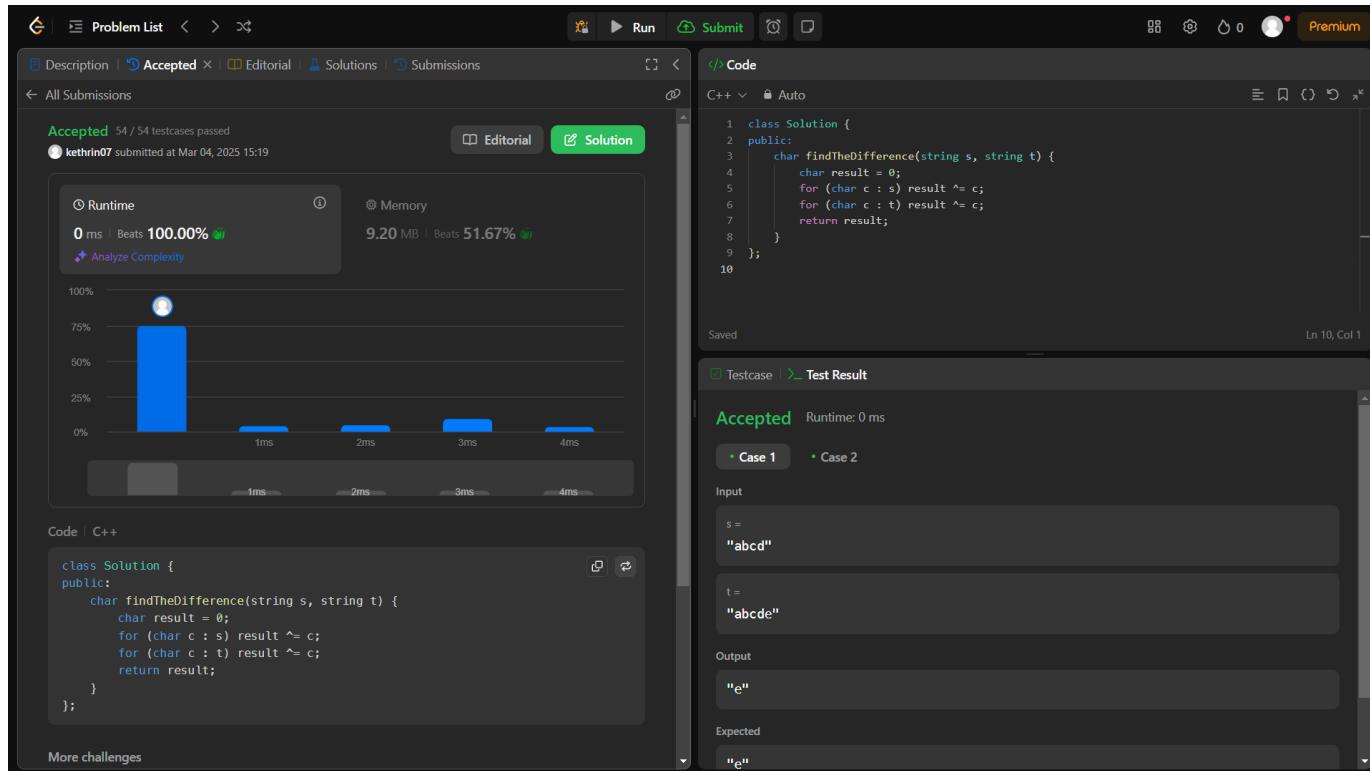
Subject Code: 22CSP-351

- **389. Find the difference**

CODE:

```
class Solution {
public:
    char findTheDifference(string s, string t) {
        char result = 0;
        for (char c : s) result ^= c;
        for (char c : t) result ^= c;
        return result;
    }
};
```

OUTPUT:





DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

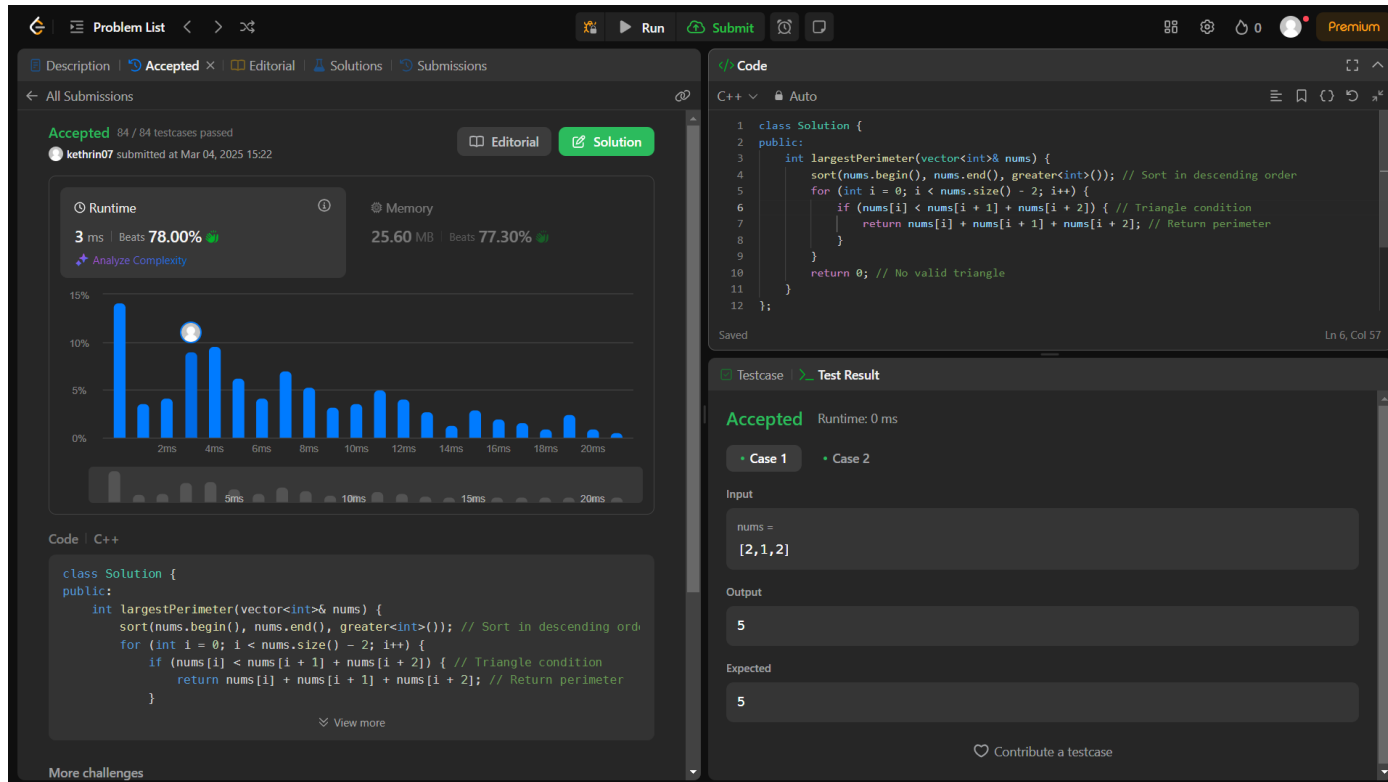
Discover. Learn. Empower.

- 976. [Largest Perimeter Triangle](#)

CODE:

```
class Solution {
public:
    int largestPerimeter(vector<int>& nums) {
        sort(nums.begin(), nums.end(), greater<int>()); // Sort in descending order
        for (int i = 0; i < nums.size() - 2; i++) {
            if (nums[i] < nums[i + 1] + nums[i + 2]) { // Triangle condition
                return nums[i] + nums[i + 1] + nums[i + 2]; // Return perimeter
            }
        }
        return 0; // No valid triangle
    }
};
```

OUTPUT:



- 414. [Third Maximum Number](#)

CODE:

```
class Solution {
public:
    int thirdMax(vector<int>& nums) {
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
long first = LONG_MIN, second = LONG_MIN, third = LONG_MIN; // Use  
LONG_MIN to handle edge cases
```

```
for (int num : nums) {  
    if (num == first || num == second || num == third) continue; // Skip duplicates
```

```
    if (num > first) {  
        third = second;  
        second = first;  
        first = num;  
    } else if (num > second) {  
        third = second;  
        second = num;  
    } else if (num > third) {  
        third = num;  
    }  
}
```

```
return (third == LONG_MIN) ? first : third; // Return third max if exists,  
otherwise max  
}  
};
```

OUTPUT:

The screenshot displays a C++ IDE interface. On the left, the 'Problem List' tab is active, showing a submission for 'Accepted' with 34/34 testcases passed. The submission was made by 'kethin07' on Mar 04, 2025 at 15:25. The 'Runtime' section shows 0 ms and 100.00% efficiency. The 'Memory' section shows 12.94 MB and 58.64% efficiency. A bar chart shows the runtime for different input sizes (1ms to 7ms). The 'Code' tab on the right shows the C++ code for the 'thirdMax' function. The code is as follows:

```
class Solution {  
public:  
    int thirdMax(vector<int>& nums) {  
        long first = LONG_MIN, second = LONG_MIN, third = LONG_MIN; // Use LONG_MIN to handle  
        edge cases  
        for (int num : nums) {  
            if (num == first || num == second || num == third) continue; // Skip duplicates  
            if (num > first) {  
                third = second;  
                second = first;  
                first = num;  
            } else if (num > second) {  
                third = second;  
                second = num;  
            } else if (num > third) {  
                third = num;  
            }  
        }  
        return (third == LONG_MIN) ? first : third; // Return third max if exists,  
        otherwise max  
    }  
};
```

The 'Testcase' tab on the right shows the test result for 'Case 1'. The input is 'nums = [3, 2, 1]' and the output is '1'. The test is marked as 'Accepted' with a runtime of 0 ms.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

- 451. [Sort Characters By Frequency](#)

CODE:

```
class Solution {
public:
    string frequencySort(string s) {
        unordered_map<char, int> freq;
        for (char c : s) freq[c]++;

        priority_queue<pair<int, char>> maxHeap;
        for (auto [c, f] : freq) maxHeap.push({f, c});

        string result;
        while (!maxHeap.empty()) {
            auto [f, c] = maxHeap.top();
            maxHeap.pop();
            result.append(f, c);
        }
        return result;
    }
};
```

OUTPUT:

The screenshot displays a web browser window with the URL <https://leetcode.com/problems/sort-characters-by-frequency/submissions/1562444595/>. The page shows the submission details for the problem "Sort Characters By Frequency". The submission is marked as "Accepted" with 33/33 test cases passed. The user "kethrin07" submitted it on Mar 04, 2025, at 15:26. The runtime is 0 ms, and the memory usage is 11.33 MB. A bar chart shows the runtime distribution, with the highest frequency at 0 ms. The code is written in C++ and uses a priority queue to sort characters by frequency. The test case shows the input "tree" and the output "eetr".

```
class Solution {
public:
    string frequencySort(string s) {
        unordered_map<char, int> freq;
        for (char c : s) freq[c]++;

        priority_queue<pair<int, char>> maxHeap;
        for (auto [c, f] : freq) maxHeap.push({f, c});

        string result;
        while (!maxHeap.empty()) {
            auto [f, c] = maxHeap.top();
            maxHeap.pop();
            result.append(f, c);
        }
        return result;
    }
};
```

Testcase: Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input: s = "tree"

Output: "eetr"

Expected: "eetr"



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

- 452. [Minimum Number of Arrows to Burst Balloons](#)

CODE:

```
class Solution {
public:
    int findMinArrowShots(vector<vector<int>>& points) {
        sort(points.begin(), points.end(), [](auto &a, auto &b) { return a[1] < b[1]; });

        int arrows = 1, end = points[0][1];
        for (auto &p : points) {
            if (p[0] > end) {
                arrows++;
                end = p[1];
            }
        }
        return arrows;
    }
};
```

OUTPUT:

The screenshot displays a web browser window with the URL <https://leetcode.com/problems/minimum-number-of-arrows-to-burst-balloons/submissions/1562448903/>. The page shows the submission details for the problem "Minimum Number of Arrows to Burst Balloons". The submission is marked as "Accepted" and was submitted by user "kethrin07" on March 04, 2025, at 15:33. The runtime is 44 ms, which beats 76.85% of other submissions, and the memory usage is 93.90 MB, which beats 75.34%. A bar chart shows the distribution of runtimes, with the user's submission being the fastest. The code is written in C++ and matches the code provided in the previous block. The test case shows an input of points = [[10,16], [2,8], [1,6], [7,12]] and an output of 2.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

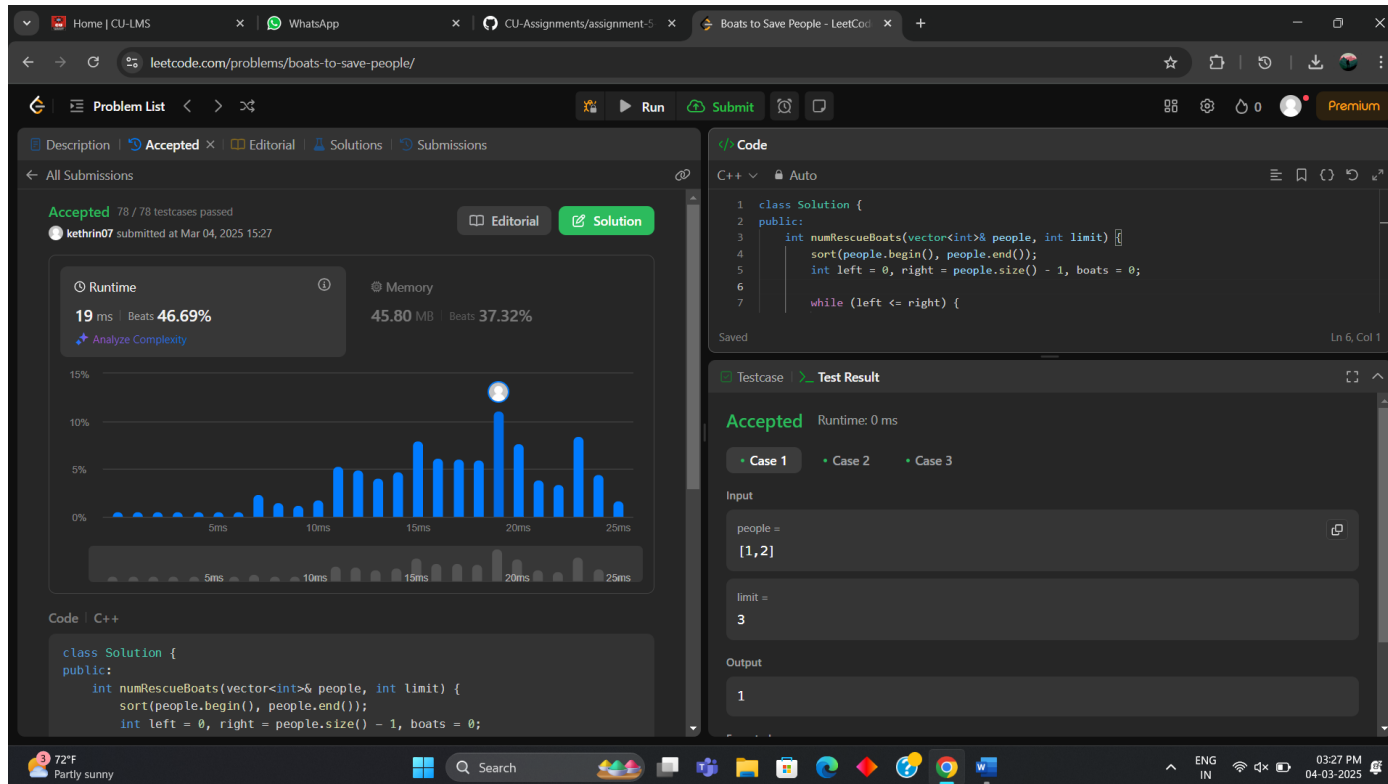
- 881. [Boats to Save People](#)

CODE:

```
class Solution {
public:
    int numRescueBoats(vector<int>& people, int limit) {
        sort(people.begin(), people.end());
        int left = 0, right = people.size() - 1, boats = 0;

        while (left <= right) {
            if (people[left] + people[right] <= limit) left++;
            right--;
            boats++;
        }
        return boats;
    }
};
```

OUTPUT:



- 973. [K Closest Points to Origin](#)

CODE:



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

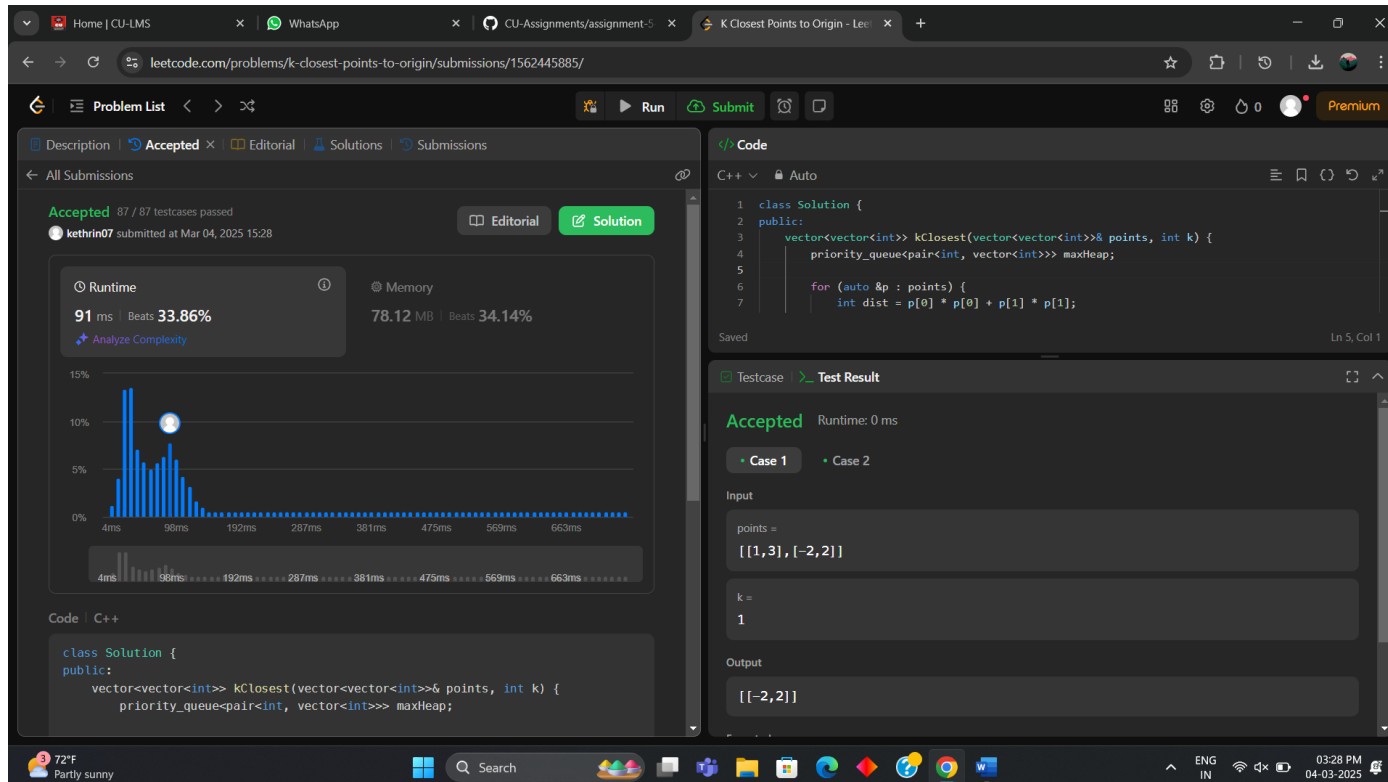
Discover. Learn. Empower.

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

        for (auto &p : points) {
            int dist = p[0] * p[0] + p[1] * p[1];
            maxHeap.push({dist, p});
            if (maxHeap.size() > k) maxHeap.pop();
        }

        vector<vector<int>> result;
        while (!maxHeap.empty()) {
            result.push_back(maxHeap.top().second);
            maxHeap.pop();
        }
        return result;
    }
};
```

OUTPUT:



- 1338. [Reduce Array Size to The Half](#)



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

CODE:

```
class Solution {
public:
    int minSetSize(vector<int>& arr) {
        unordered_map<int, int> freq;
        for (int num : arr) freq[num]++;

        vector<int> counts;
        for (auto [num, f] : freq) counts.push_back(f);

        sort(counts.rbegin(), counts.rend()); // Sort in descending order

        int removed = 0, setSize = 0, half = arr.size() / 2;
        for (int f : counts) {
            removed += f;
            setSize++;
            if (removed >= half) return setSize;
        }
        return setSize;
    }
};
```

OUTPUT:

The screenshot shows a web browser window with the URL <https://leetcode.com/problems/reduce-array-size-to-the-half/>. The page displays the problem description, a bar chart of runtime performance (88 ms, 42.86% beats), and the C++ code. The test result shows the input array [3, 3, 3, 3, 5, 5, 2, 2, 7] and the output 2.

Runtime Performance:

- Runtime: 88 ms
- Beats: 42.86%
- Memory: 85.58 MB
- Beats: 24.74%

Test Result:

- Accepted
- Runtime: 0 ms
- Case 1: Input [3, 3, 3, 3, 5, 5, 2, 2, 7], Output 2