

## Experiment 5

**Student Name:** Barenya Behera

**Branch:** BE-CSE

**Semester:** 6<sup>th</sup>

**Subject Name:** Advanced Programming

**Lab-2**

**UID:** 22BCS15121

**Section/Group:** FL\_IOT-602/A

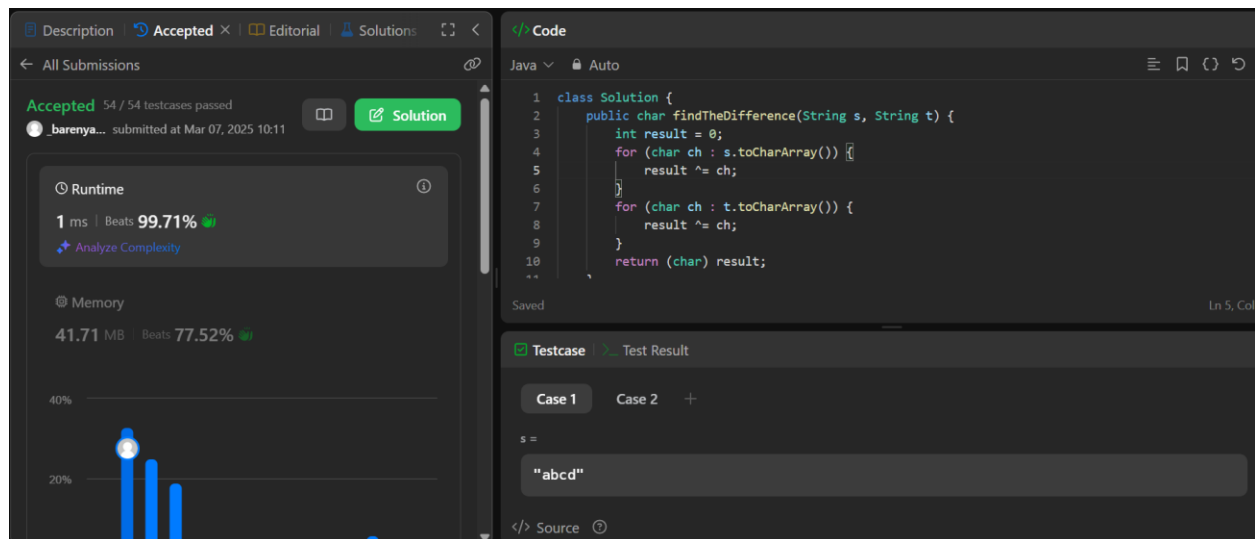
**Date of Performance:** 4/03/25

**Subject Code:** 22CSP-351

### 1. Implementation/Code:

#### i. Find the difference

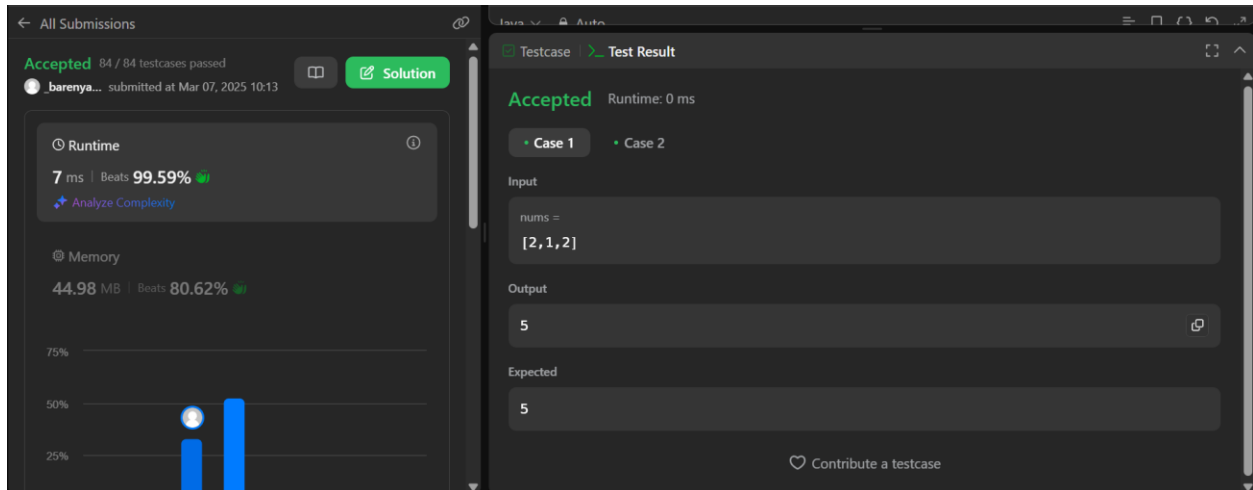
```
class Solution {  
    public char findTheDifference(String s, String t) {  
        int result = 0;  
        for (char ch : s.toCharArray()) {  
            result ^= ch;  
        }  
        for (char ch : t.toCharArray()) {  
            result ^= ch;  
        }  
        return (char) result;  
    }  
}
```



The screenshot displays a coding platform interface with a dark theme. On the left, the 'Description' tab is active, showing 'Accepted' status for 54/54 testcases. The submission is by user '\_barenya...' and was submitted on Mar 07, 2025 at 10:11. Performance metrics are shown: Runtime is 1 ms (Beats 99.71%) and Memory is 41.71 MB (Beats 77.52%). A bar chart at the bottom shows the user's performance relative to others. On the right, the 'Code' tab is active, displaying the Java code for the 'findTheDifference' method. The code uses XOR to find the character that appears only once in the two strings. Below the code, the 'Testcase' tab is active, showing 'Case 1' with input 's = "abcd"'. The 'Source' button is visible at the bottom right.

## ii. Largest Perimeter Traiangle

```
class Solution {  
    public int largestPerimeter(int[] nums) {  
        Arrays.sort(nums);  
        for (int i = nums.length - 1; i >= 2; i--) {  
            if (nums[i - 2] + nums[i - 1] > nums[i]) {  
                return nums[i - 2] + nums[i - 1] + nums[i];  
            }  
        }  
        return 0;  
    }  
}
```



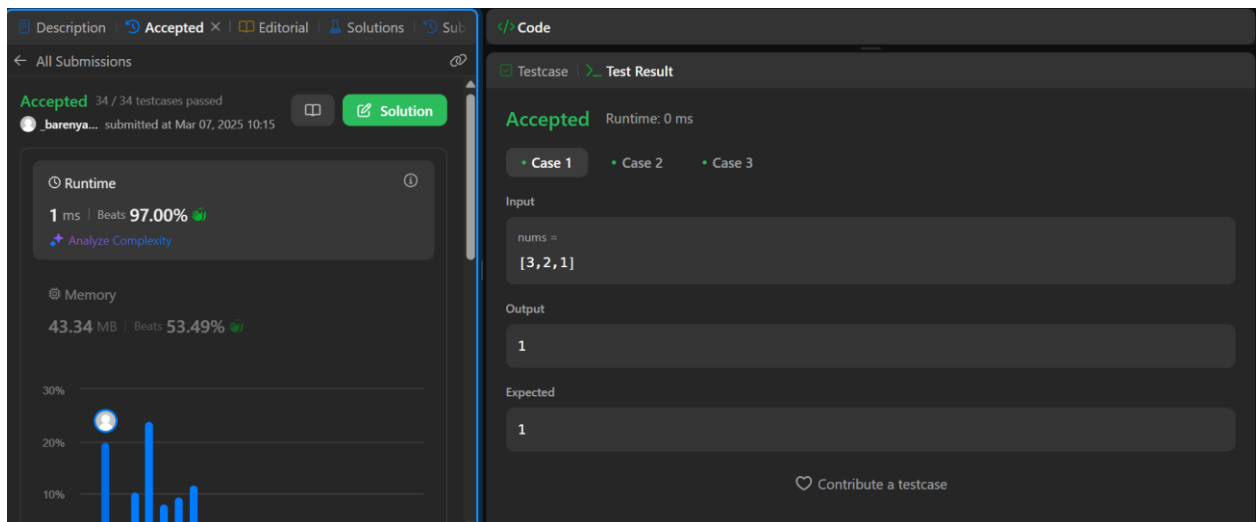
## iii. Third Maximum Number

```
class Solution {  
    public int thirdMax(int[] nums) {  
        long first = Long.MIN_VALUE, second = Long.MIN_VALUE, third =  
        Long.MIN_VALUE;  
        for (int num : nums) {  
            if (num == first || num == second || num == third) continue;  
            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_VALUE ? (int) first : (int) third;
}
}

```



#### iv. Sort Characters By Frequency

```
class Solution {
```

```
    public String frequencySort(String s) {
```

```
        Map<Character, Integer> freqMap = new HashMap<>();
```

```
        for (char c : s.toCharArray()) {
```

```
            freqMap.put(c, freqMap.getOrDefault(c, 0) + 1);
```

```
        }
```

```
        PriorityQueue<Character> maxHeap = new PriorityQueue<>((a, b) ->
            freqMap.get(b) - freqMap.get(a));
```

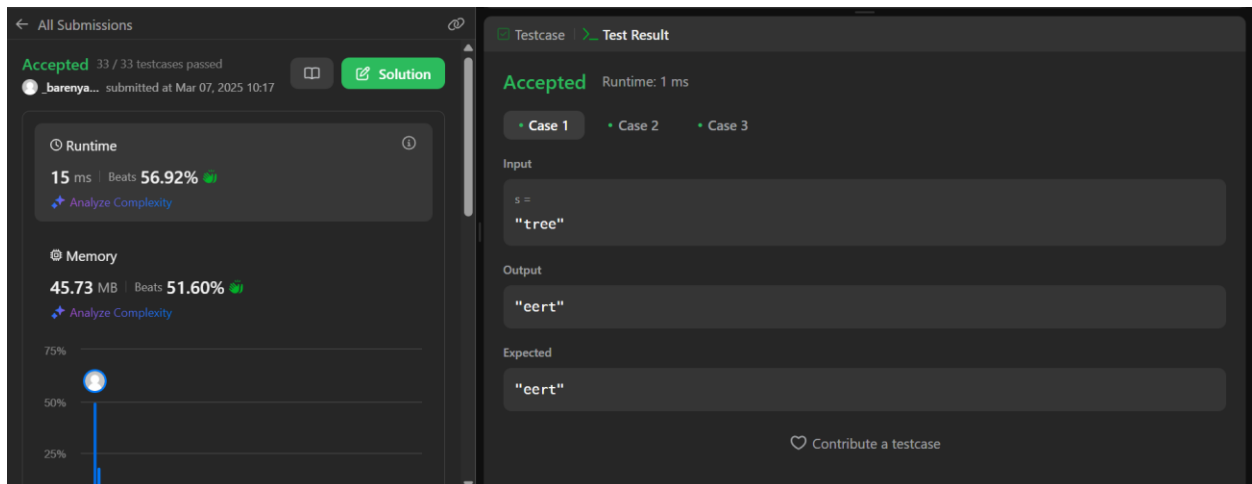
```

maxHeap.addAll(freqMap.keySet());

StringBuilder result = new StringBuilder();
while (!maxHeap.isEmpty()) {
    char c = maxHeap.poll();
    result.append(String.valueOf(c).repeat(freqMap.get(c)));
}

return result.toString();
}
}

```



## v. Minimum Number of Arrows to Burst Balloons

```

class Solution {
    public int findMinArrowShots(int[][] points) {
        Arrays.sort(points, (a, b) -> Integer.compare(a[1], b[1]));
        int arrows = 1;
        int end = points[0][1];

        for (int i = 1; i < points.length; i++) {
            if (points[i][0] > end) {
                arrows++;
            }
        }
    }
}

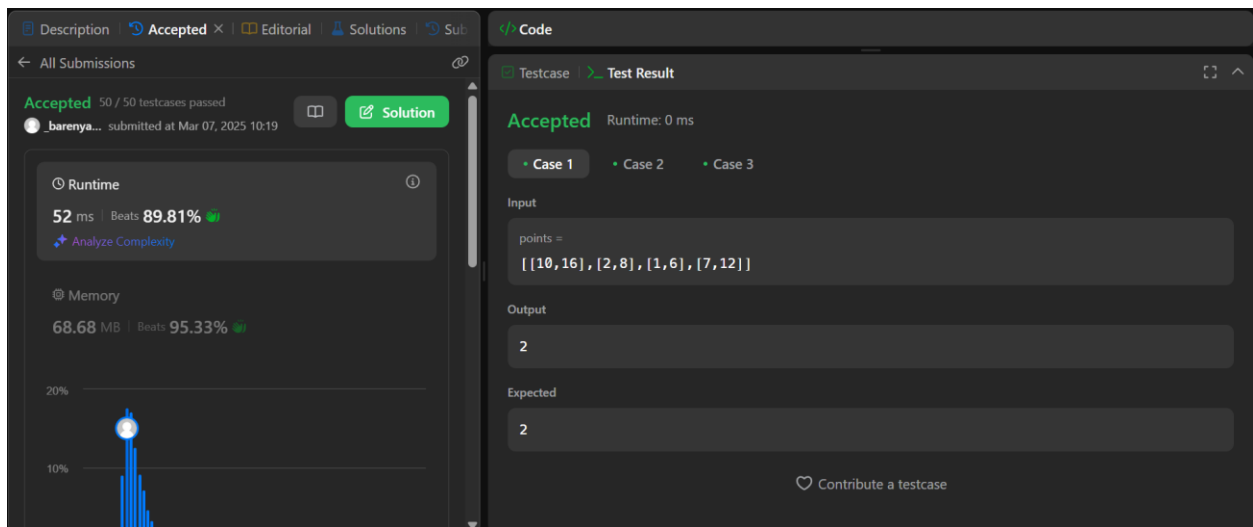
```

```

        end = points[i][1];
    }
}

return arrows;
}
}

```



The screenshot displays a submission result for a problem. On the left, the 'All Submissions' tab is active, showing a submission by user '\_barenya...' that was 'Accepted' on March 07, 2025, at 10:19. The submission details include a runtime of 52 ms (Beats 89.81%) and a memory usage of 68.68 MB (Beats 95.33%). A 'Solution' button is visible. On the right, the 'Testcase' tab is active, showing 'Accepted' with a runtime of 0 ms. The input is 'points = [[10,16],[2,8],[1,6],[7,12]]', the output is '2', and the expected output is '2'. A 'Contribute a testcase' link is at the bottom.

## vi. Boats to Save People

```

class Solution {
    public int numRescueBoats(int[] people, int limit) {
        Arrays.sort(people);
        int left = 0, right = people.length - 1;
        int boats = 0;

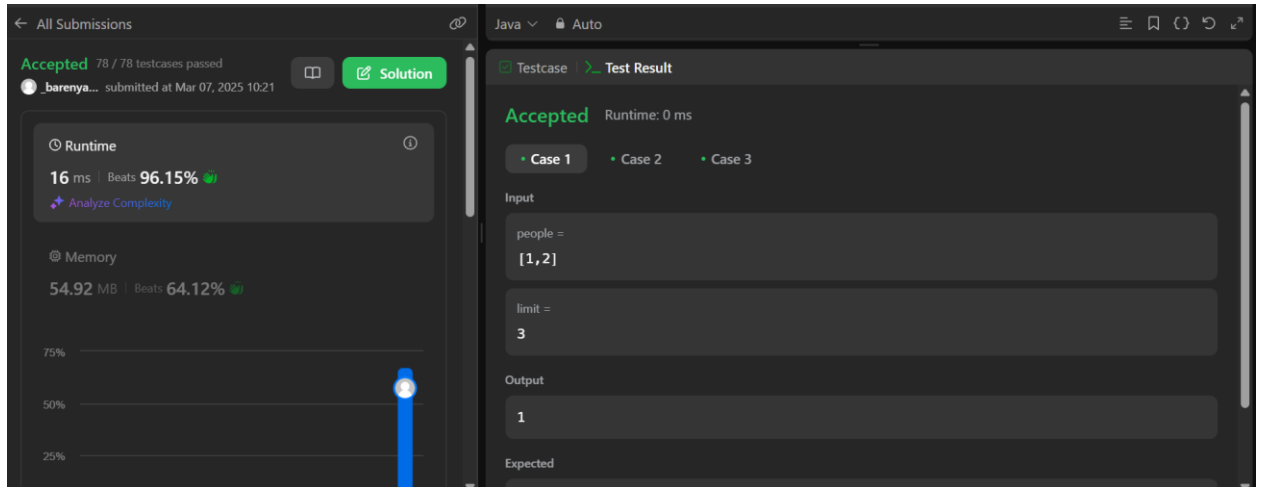
        while (left <= right) {
            if (people[left] + people[right] <= limit) {
                left++;
            }
            right--;
            boats++;
        }
    }
}

```

```

        return boats;
    }
}

```

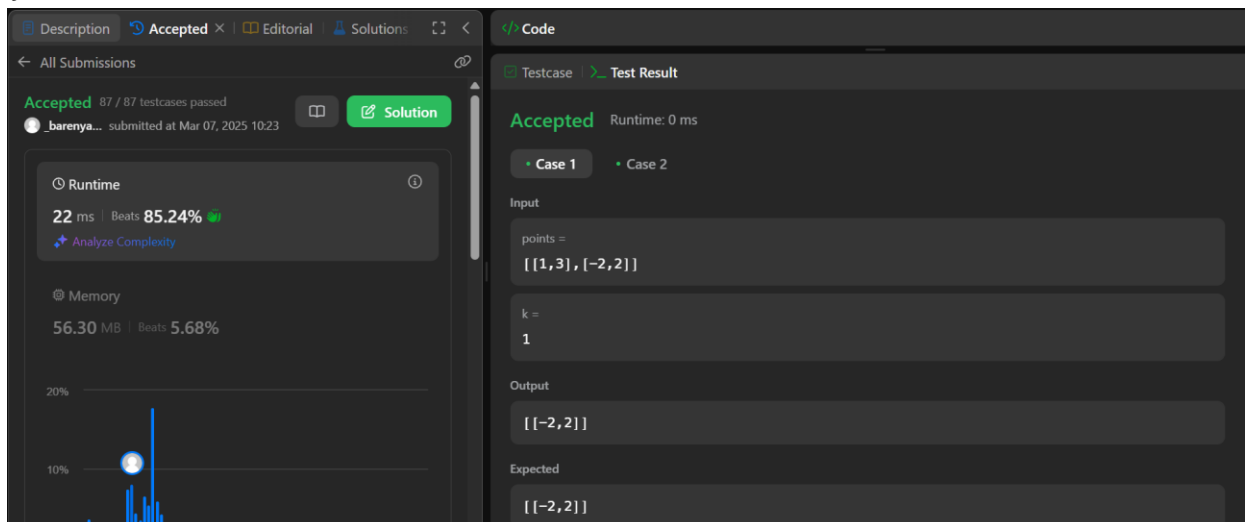


## vii. K Closest Points to Origin

```

class Solution {
    public int[][] kClosest(int[][] points, int k) {
        Arrays.sort(points, (a, b) -> Integer.compare(a[0] * a[0] + a[1] * a[1],
            b[0] * b[0] + b[1] * b[1]));
        return Arrays.copyOfRange(points, 0, k);
    }
}

```



## viii. Reduce Array Size to the Half

```
class Solution {
```

```
    public int minSetSize(int[] arr) {
```

```
        Map<Integer, Integer> freqMap = new HashMap<>();
```

```
        for (int num : arr) {
```

```
            freqMap.put(num, freqMap.getOrDefault(num, 0) + 1);
```

```
        }
```

```
        PriorityQueue<Integer> maxHeap = new
```

```
        PriorityQueue<>(Collections.reverseOrder());
```

```
        maxHeap.addAll(freqMap.values());
```

```
        int halfSize = arr.length / 2, removed = 0, count = 0;
```

```
        while (removed < halfSize) {
```

```
            removed += maxHeap.poll();
```

```
            count++;
```

```
        }
```

```
        return count;
```

```
    }
```

```
}
```

