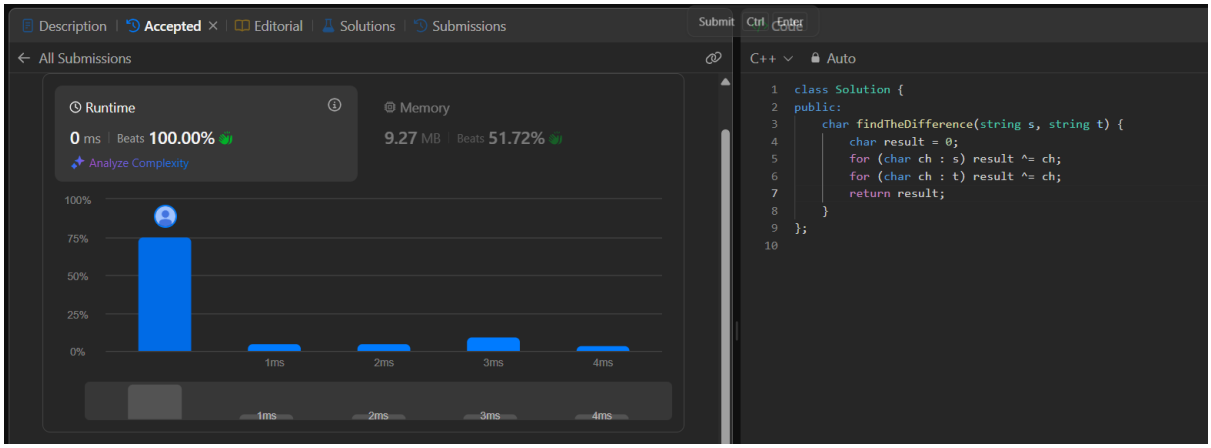
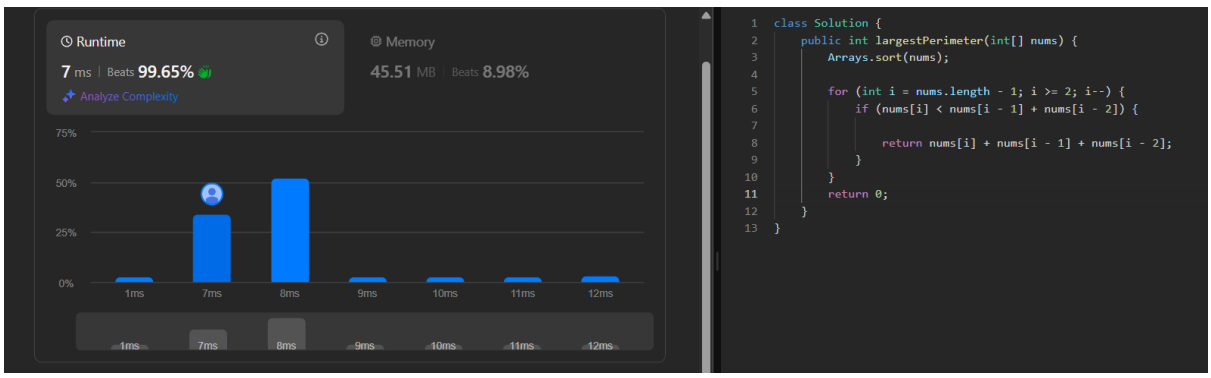
Assignment -5

Name – Chirag Singh
Class – 603_FL
UID – 22BCS15116

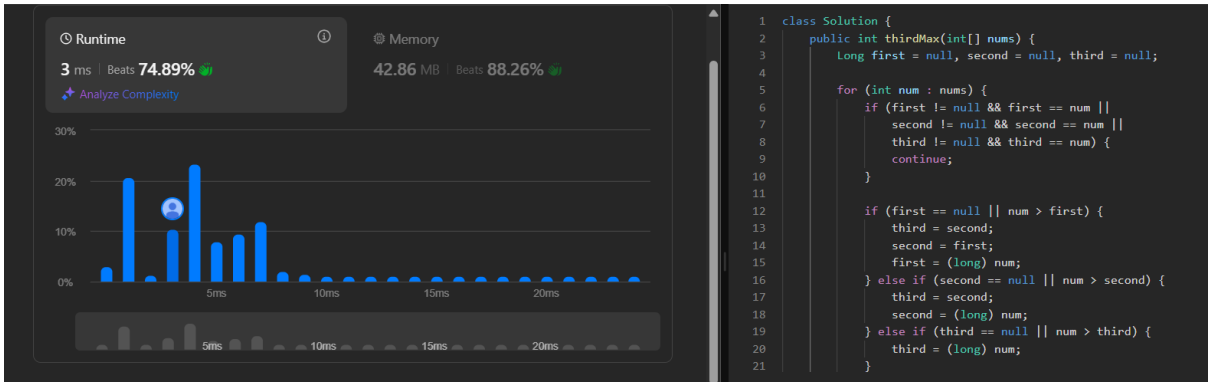
Find the diffrence



Largest Perimeter Triangle

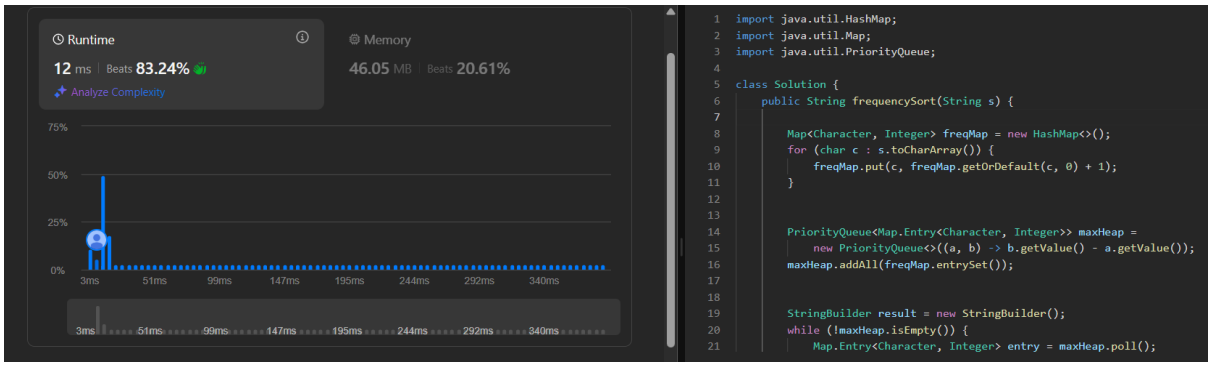


Third Maximum Number



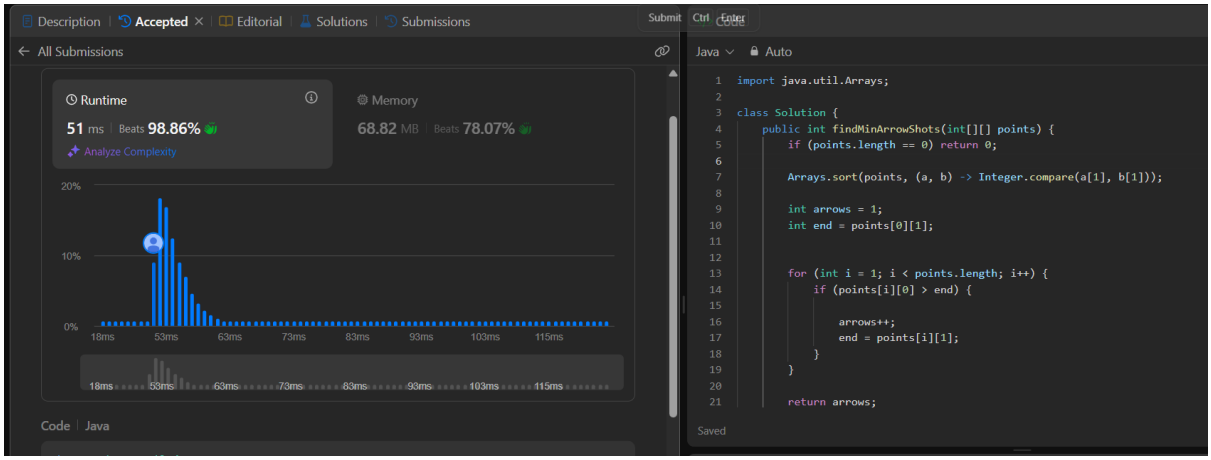

```java
class Solution {
    public int thirdMax(int[] nums) {
        Long first = null, second = null, third = null;

        for (int num : nums) {
            if (first != null && first == num ||
                second != null && second == num ||
                third != null && third == num) {
                continue;
            }

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

Sort Characters By Frequency



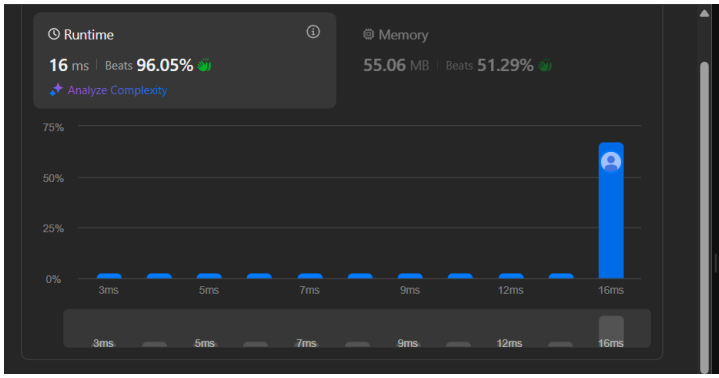

```java
import java.util.HashMap;
import java.util.Map;
import java.util.PriorityQueue;

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<Map.Entry<Character, Integer>> maxHeap =
            new PriorityQueue<>((a, b) -> b.getValue() - a.getValue());
        maxHeap.addAll(freqMap.entrySet());


        StringBuilder result = new StringBuilder();
        while (!maxHeap.isEmpty()) {
            Map.Entry<Character, Integer> entry = maxHeap.poll();
```

Minimum Number of Arrows to Burst Balloons



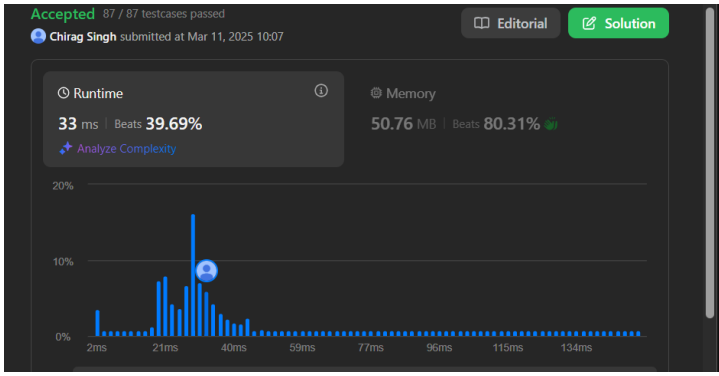

```java
import java.util.Arrays;

class Solution {
    public int findMinArrowShots(int[][] points) {
        if (points.length == 0) return 0;

        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;
```

Boats to Save People




```java
import java.util.Arrays;

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;
    }
}
```

K Closest Points to Origin




```java
import java.util.PriorityQueue;

class Solution {
    public int[][] kClosest(int[][] points, int k) {
        PriorityQueue<int[]> maxHeap = new PriorityQueue<>(
            (a, b) -> Integer.compare(
                (b[0] * b[0] + b[1] * b[1]),
                (a[0] * a[0] + a[1] * a[1])
            )
        );

        for (int[] point : points) {
            maxHeap.add(point);
            if (maxHeap.size() > k) {
                maxHeap.poll();
            }
        }

        int[][] result = new int[k][2];
        for (int i = 0; i < k; i++) {
            result[i] = maxHeap.poll();
```