**Assignment 5**

**Name: Aditi Singh UID: 22BCS10670**

**Section: FL\_IOT\_602 Group: A**

**Ques:** [Find the difference](https://leetcode.com/problems/find-the-difference/description/)

**Sol:** class Solution {

        public static char findTheDifference(String s, String t) {

        int sXOR = 0, tXOR = 0;

        for (char ch : s.toCharArray()) {

            sXOR ^= ch;

        }

        for (char ch : t.toCharArray()) {

            tXOR ^= ch;

        }

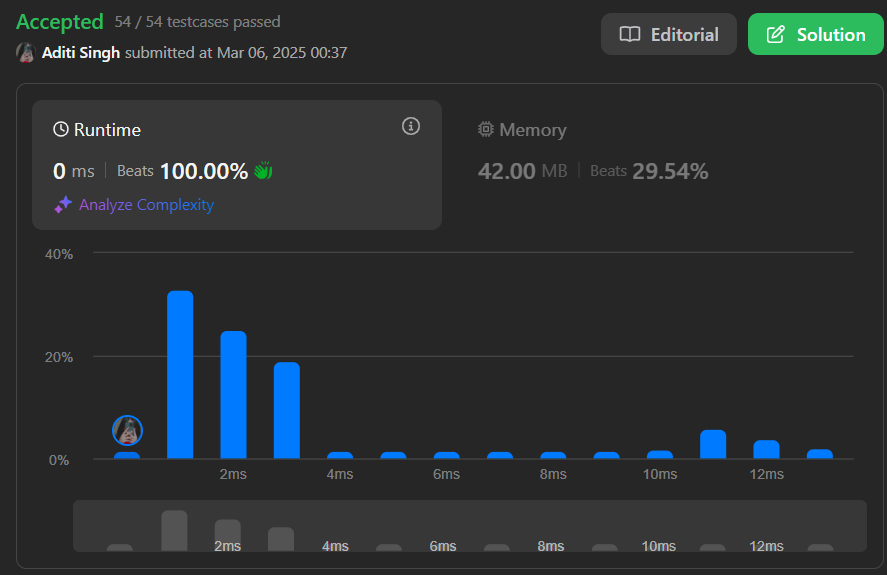
        char ans = (char)(sXOR ^ tXOR);

        return ans;

    }

}

**Output:**

****

**Ques:** [Largest Perimeter Triangle](https://leetcode.com/problems/largest-perimeter-triangle/description/)

**Sol:** class Solution {

    public int largestPerimeter(int[] nums) {

        Arrays.sort(nums);

        for (int i = nums.length - 3; i >= 0; --i) {

            if (nums[i] + nums[i + 1] > nums[i + 2])

                return nums[i] + nums[i + 1] + nums[i + 2];

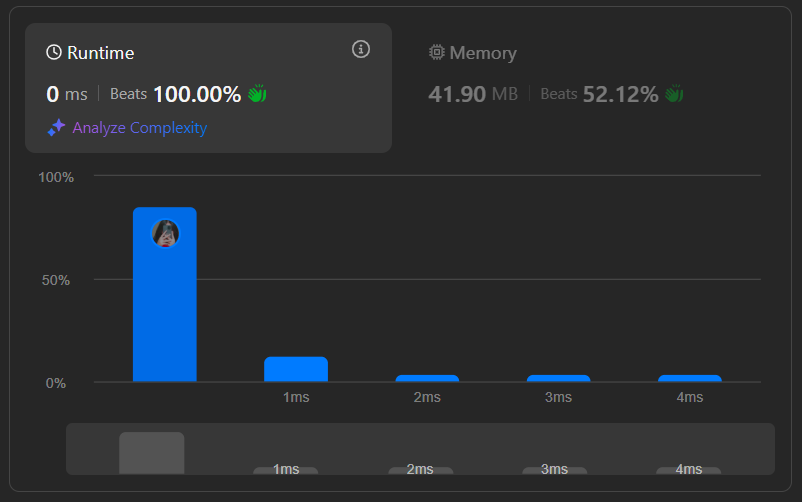
        }

        return 0;

    }

}

**Output:**

****

**Ques:** [Third Maximum Number](https://leetcode.com/problems/third-maximum-number/description/)

**Sol:** class Solution {

public int thirdMax(int[] nums) {

long max1 = Long.MIN\_VALUE;

long max2 = Long.MIN\_VALUE;

long max3 = Long.MIN\_VALUE;

for(int n : nums) {

if(n > max1) {

max3 = max2;

max2 = max1;

max1 = (long)n;

}

else if(n > max2 && n < max1) {

max3 = max2;

max2 = (long)n;

}

else if(n > max3 && n < max2)

max3 = (long)n;

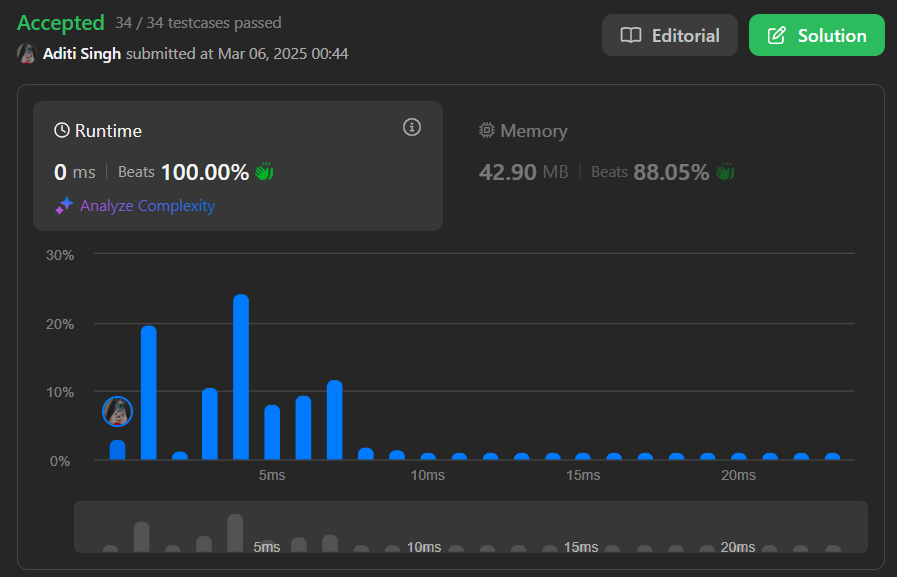
}

return max3 == Long.MIN\_VALUE ? (int)max1 : (int)max3;

}

}

**Output:**

****

**Ques:** [Sort Characters By Frequency](https://leetcode.com/problems/sort-characters-by-frequency/description/)

**Sol:**

class Solution {

    public String frequencySort(String s) {

        int[] freq = new int[128];

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

            freq[c]++;

        }

        List<Character>[] buckets = new ArrayList[s.length() + 1];

        for (int i = 0; i < 128; i++) {

            if (freq[i] > 0) {

                if (buckets[freq[i]] == null) {

                    buckets[freq[i]] = new ArrayList<>();

                }

                buckets[freq[i]].add((char) i);

            }

        }

        StringBuilder result = new StringBuilder();

        for (int i = s.length(); i > 0; i--) {

            if (buckets[i] != null) {

                for (char c : buckets[i]) {

                    result.append(String.valueOf(c).repeat(i));

                }

            }

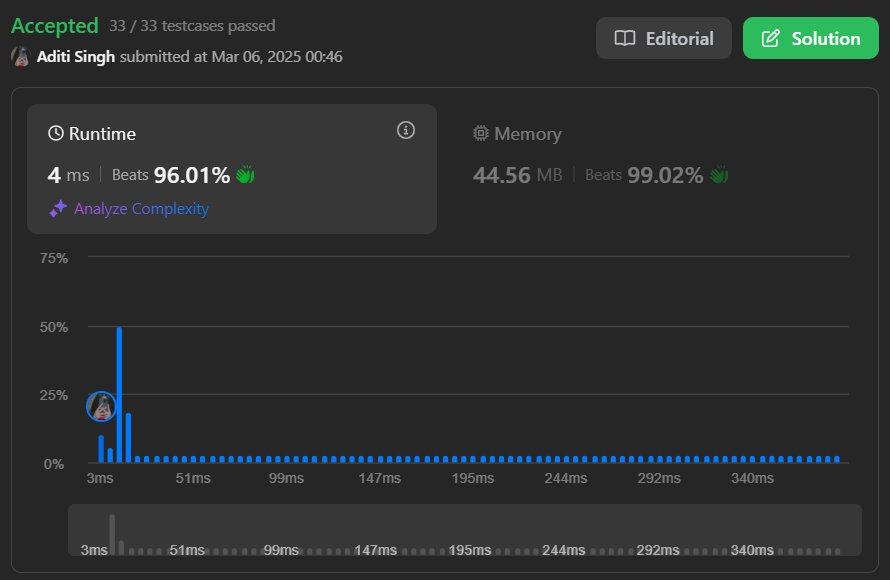
        }

              return result.toString();

    }

}

**Output:**

****

**Ques:**[Minimum Number of Arrows to Burst Balloons](https://leetcode.com/problems/minimum-number-of-arrows-to-burst-balloons/)

**Sol:** class Solution {

public int findMinArrowShots(int[][] segments) {

Arrays.sort(segments, (a, b) -> Integer.compare(a[1], b[1]));

int ans = 0, arrow = 0;

for (int i = 0; i < segments.length; i ++) {

if (ans == 0 || segments[i][0] > arrow) {

ans ++;

arrow = segments[i][1];

}

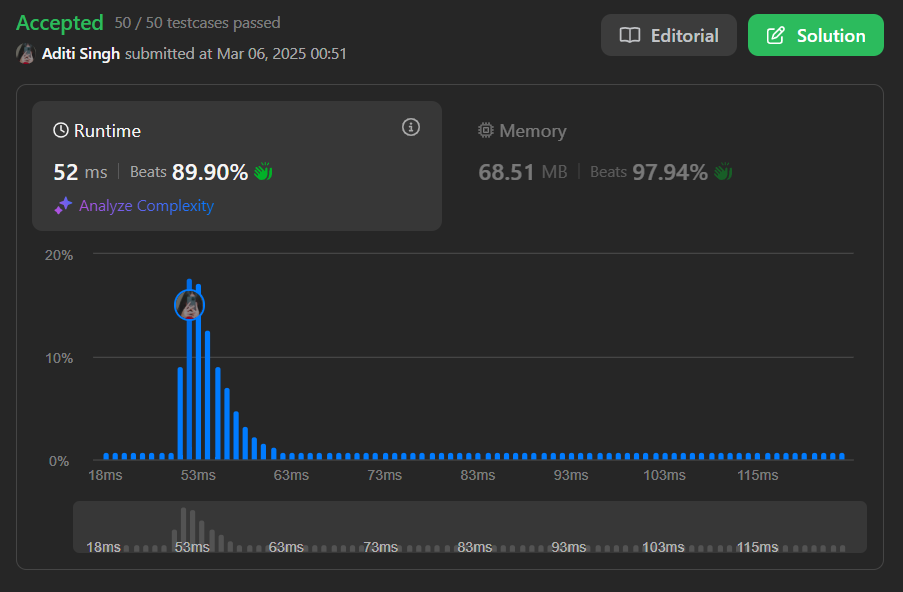
}

return ans;

}

}

**Output:**

****

**Ques:** [Boats to Save People](https://leetcode.com/problems/boats-to-save-people/description/)

**Sol:** class Solution {

    public int numRescueBoats(int[] people, int limit) {

        int boatCount = 0;

        Arrays.sort(people);

        int left = 0;

        int right = people.length - 1;

        while(left <= right){

            int sum = people[left] + people[right];

            if(sum <= limit){

                boatCount++;

                left++;

                right--;

            }

            else{

                boatCount++;

                right--;

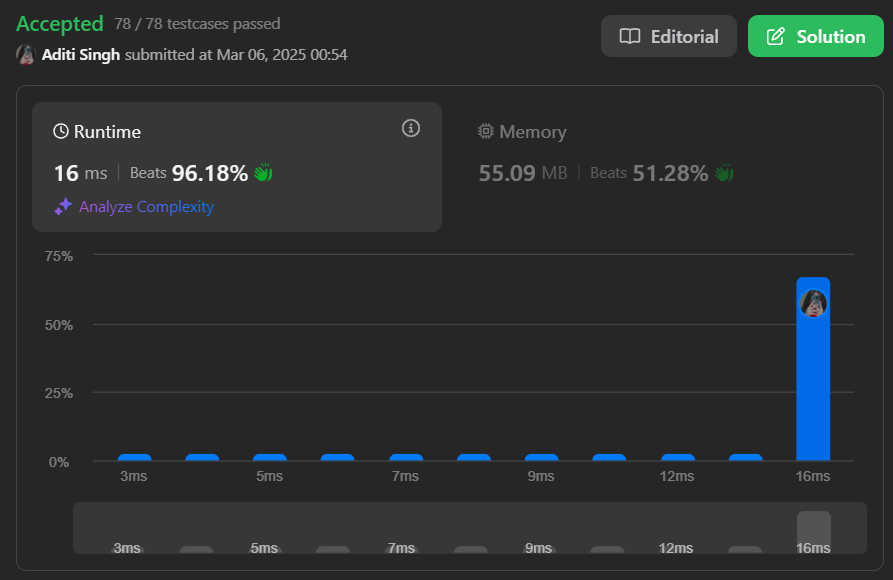
            }

        }

        return boatCount;

    }

}

**Output:**  

**Ques:** [K Closest Points to Origin](https://leetcode.com/problems/k-closest-points-to-origin/description/)

**Sol:**

class Solution {

public int[][] kClosest(int[][] points, int k) {

PriorityQueue<int[]> maxHeap = new PriorityQueue<>((p, q) -> Integer.compare(distance(q), distance(p)));

for (var point: points) {

maxHeap.offer(point);

if (maxHeap.size() > k) {

maxHeap.poll();

}

}

return maxHeap.toArray(new int[0][]);

}

private int distance(int[] point) {

int x = point[0];

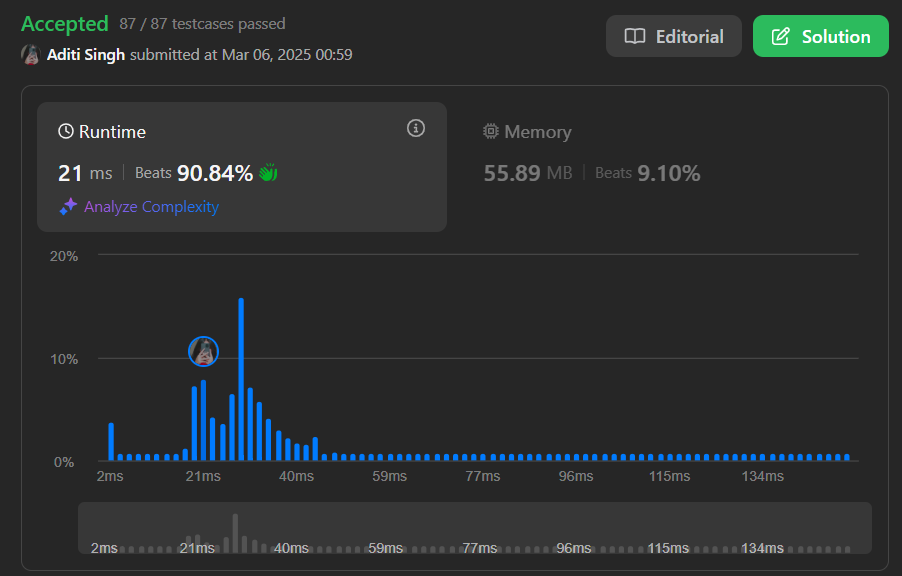
int y = point[1];

return x \* x + y \* y;

}

}

**Output:**

****

**Ques:** [Sort Characters By Frequency](https://leetcode.com/problems/sort-characters-by-frequency/description/)

**Sol:**

class Solution {

    public String frequencySort(String s) {

        int[] freq = new int[128];

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

            freq[c]++;

        }

        List<Character>[] buckets = new ArrayList[s.length() + 1];

        for (int i = 0; i < 128; i++) {

            if (freq[i] > 0) {

                if (buckets[freq[i]] == null) {

                    buckets[freq[i]] = new ArrayList<>();

                }

                buckets[freq[i]].add((char) i);

            }

        }

        StringBuilder result = new StringBuilder();

        for (int i = s.length(); i > 0; i--) {

            if (buckets[i] != null) {

                for (char c : buckets[i]) {

                    result.append(String.valueOf(c).repeat(i));

                }

            }

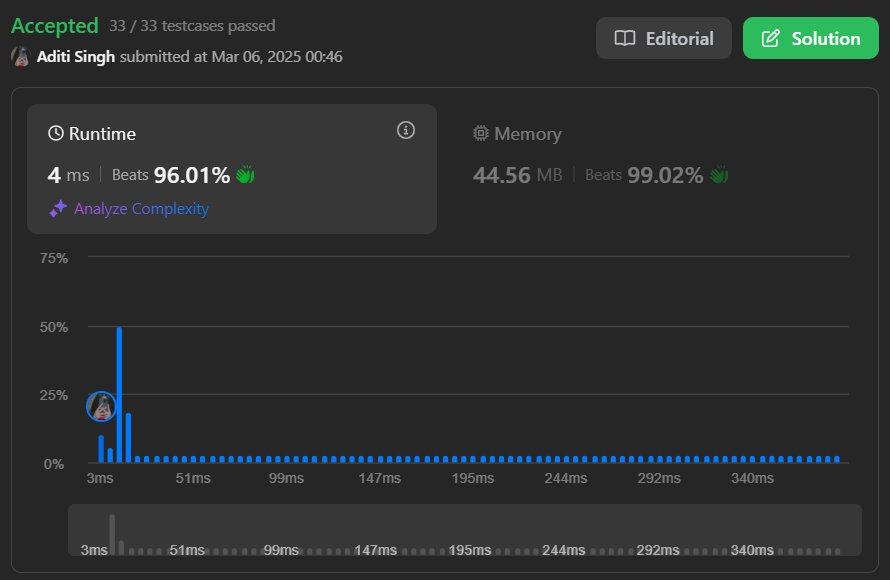
        }

              return result.toString();

    }

}

**Output:**

****

**Ques:** [Reduce Array Size to The Half](https://leetcode.com/problems/reduce-array-size-to-the-half/description/)

**Sol:** class Solution {

    public int minSetSize(int[] arr) {

        int max = arr[0];

        int min = max;

        for (int i : arr) {

            if (max < i) max = i;

            if (min > i) min = i;

        }

        int[] count = new int[max-min+1];

        int maxCount = 0;

        for (int i : arr) {

            maxCount = Math.max(maxCount,++count[i-min]);

        }

        int[] secondCount = new int[maxCount+1];

        for (int i : count) {

            if (i == 0) continue;

            secondCount[i]++;

        }

        int n = arr.length >> 1;

        int sum = 0;

        int setSize = 0;

        for (int i = maxCount; i > 0; i--) {

            if (secondCount[i] == 0) continue;

            int cnt = secondCount[i];

            while (cnt > 0) {

                sum += i;

                setSize++;

                if (sum >= n) {

                    return setSize;

                }

                cnt--;

            }

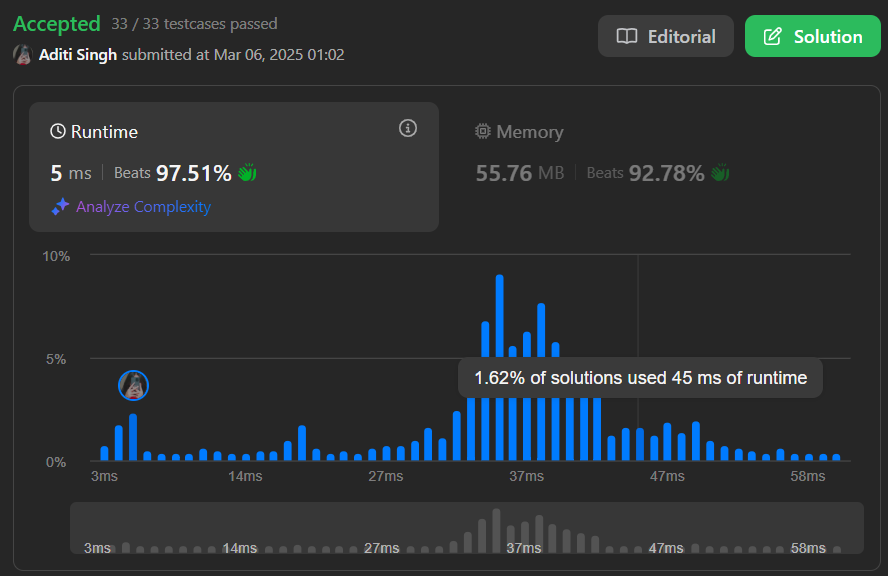
        }

        return 0;

    }

}

**Output:**

****