Assignment-5

Name: Ravi Prakash Singh Uid: 22BCS14299

Section: FL\_IOT\_602/A

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

Sol:

class Solution {

public char findTheDifference(String s, String t) {

char c = 0;

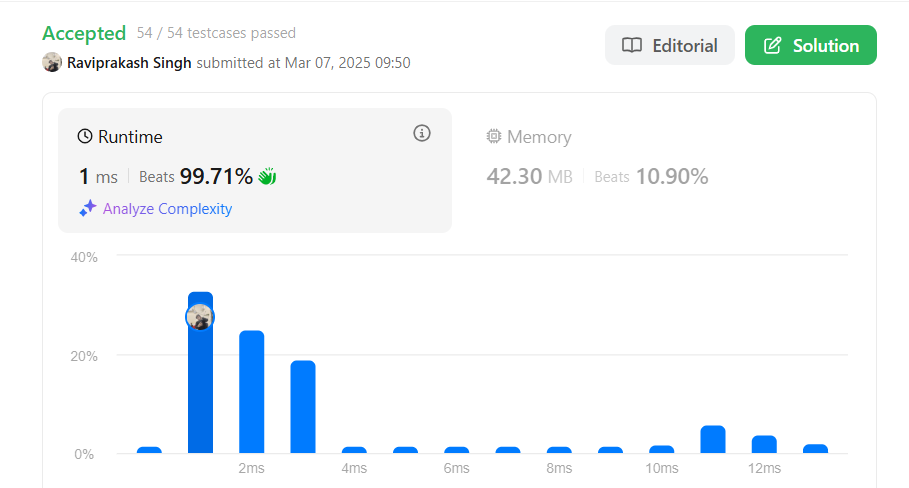
for(char cs : s.toCharArray()) c ^= cs;

for(char ct : t.toCharArray()) c ^= ct;

return c;

}

}



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

Sol:

class Solution {

public int largestPerimeter(int[] nums) {

//Sort the array first.

Arrays.sort(nums);

//Start traversing from back , so that we can get the largest value.

for(int i = nums.length-1; i>1; i--){

//Using triangle property to become valid sides

// The sum of the length of the two sides of a triangle is greater than the length of the third side.

if(nums[i] < nums[i-1] + nums[i-2])

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

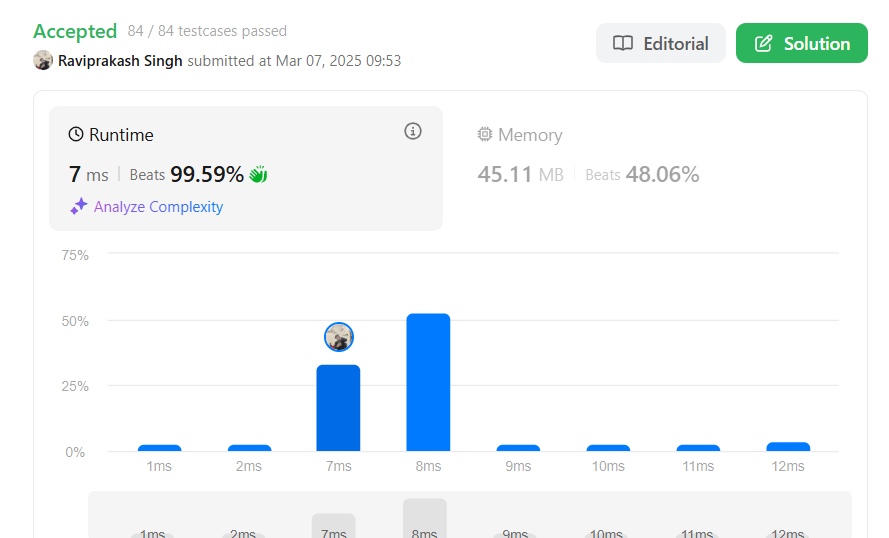
}

//If we didn't found anything we return 0.

return 0;

}

}



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

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 i = 0; i < nums.length; i++) {

if (nums[i] > max1) {

max3 = max2;

max2 = max1;

max1 = nums[i];

} else if (nums[i] > max2 && nums[i] != max1) {

max3 = max2;

max2 = nums[i];

} else if (nums[i] > max3 && nums[i] != max1 && nums[i] != max2) {

max3 = nums[i];

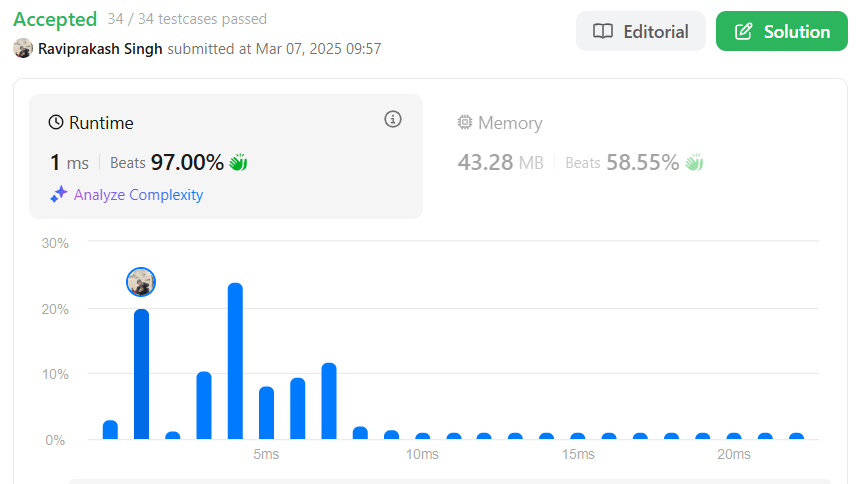
}

}

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

}

}



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

Sol:

class Solution {

public String frequencySort(String s) {

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

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

hm.put(c, hm.getOrDefault(c, 0) + 1);

}

PriorityQueue<Map.Entry<Character, Integer>> pq = new PriorityQueue<>(

(a, b) -> b.getValue() - a.getValue()

);

pq.addAll(hm.entrySet());

StringBuilder result = new StringBuilder();

while (!pq.isEmpty()) {

Map.Entry<Character, Integer> entry = pq.poll();

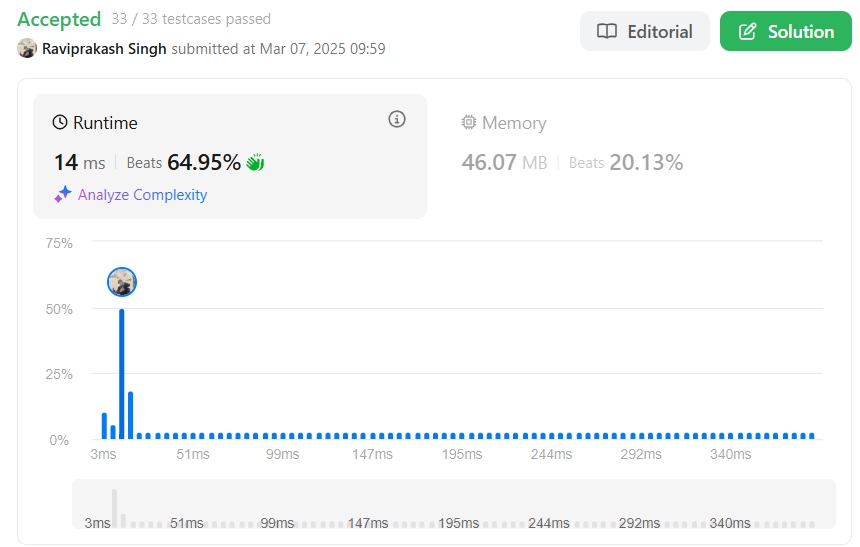
result.append(String.valueOf(entry.getKey()).repeat(entry.getValue()));

}

return result.toString();

}

}



[**452. 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[][] points) {

if (points == null || points.length == 0) return 0;

Arrays.sort(points, (int[] p1, int[] p2)->{

// Don't use p1[1 ] - p2[1]. It may overflow

return p1[1] < p2[1] ? -1 : 1;

});

int end = points[0][1];

// the reason res = 1 is that we need an arrow to destroy the last group.

int res = 1;

for (int[] point: points)

{

if (point[0] <= end) continue;

res++;

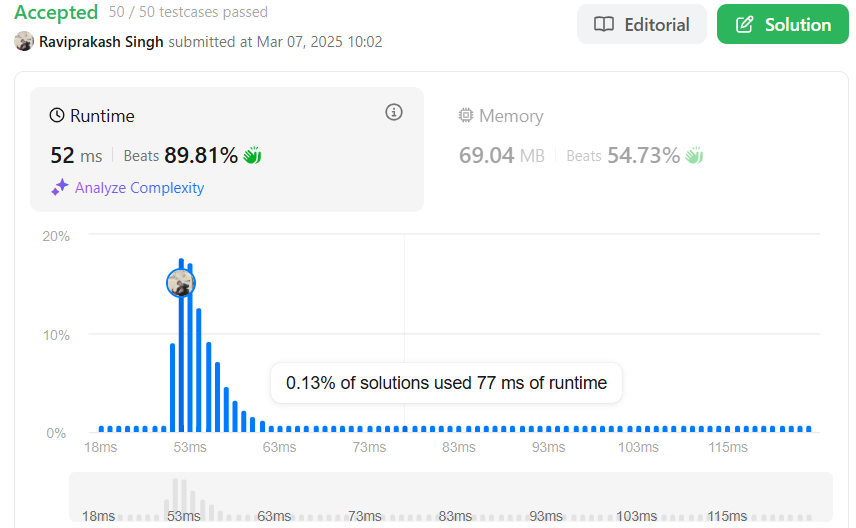
end = point[1];

}

return res;

}

}



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

class Solution {

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

int boats = 0;

Arrays.sort(people);

int i=0,j=people.length-1;

while(i<=j){

if((people[j]+people[i])<=limit){

i++;

}

j--;

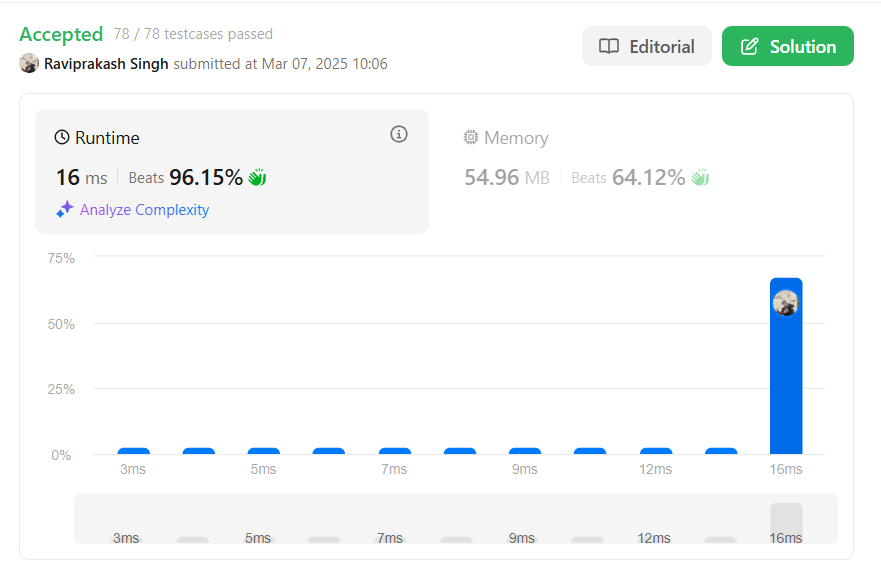
boats++;

}

return boats;

}

}



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

Sol:

class Solution {

public class Point implements Comparable<Point>{

int x;

int y;

int dis;

public Point(int x, int y, int dis){

this.x = x;

this.y = y;

this.dis = dis;

}

@Override

public int compareTo(Point p2){

return p2.dis - this.dis; // maxheap so that top will be larger one

}

}

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

//using priorityQueue

PriorityQueue<Point> pq = new PriorityQueue<>();

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

int x = points[i][0];

int y = points[i][1];

int distance = x\*x + y\*y;

pq.add(new Point(x,y,distance));

if( pq.size() > k){

pq.remove();

}

}

int j = 0;

int ans[][] = new int[k][2];

while(!pq.isEmpty() && j < k){

Point p = pq.remove();

ans[j][0] = p.x;

ans[j][1] = p.y;

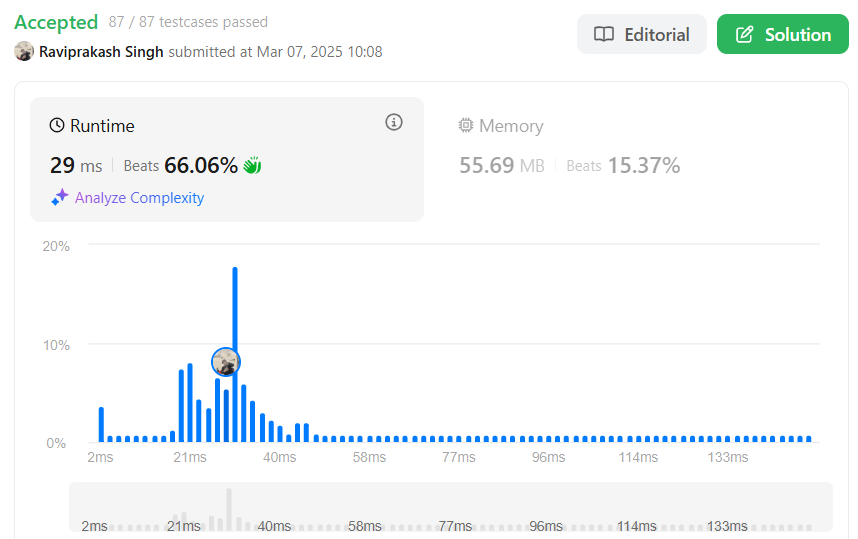
j++;

}

return ans;

}

}



[**1338. Reduce Array Size to The Half**](https://leetcode.com/problems/reduce-array-size-to-the-half/)

Sol:

class Solution {

public int minSetSize(int[] arr) {

int n = arr.length;

HashMap<Integer, Integer> cnt = new HashMap<>();

for (int x : arr) cnt.put(x, cnt.getOrDefault(x, 0) + 1);

int[] counting = new int[n + 1];

for (int freq : cnt.values()) ++counting[freq];

int ans = 0, removed = 0, half = n / 2, freq = n;

while (removed < half) {

ans += 1;

while (counting[freq] == 0) --freq;

removed += freq;

--counting[freq];

}

return ans;

}

}