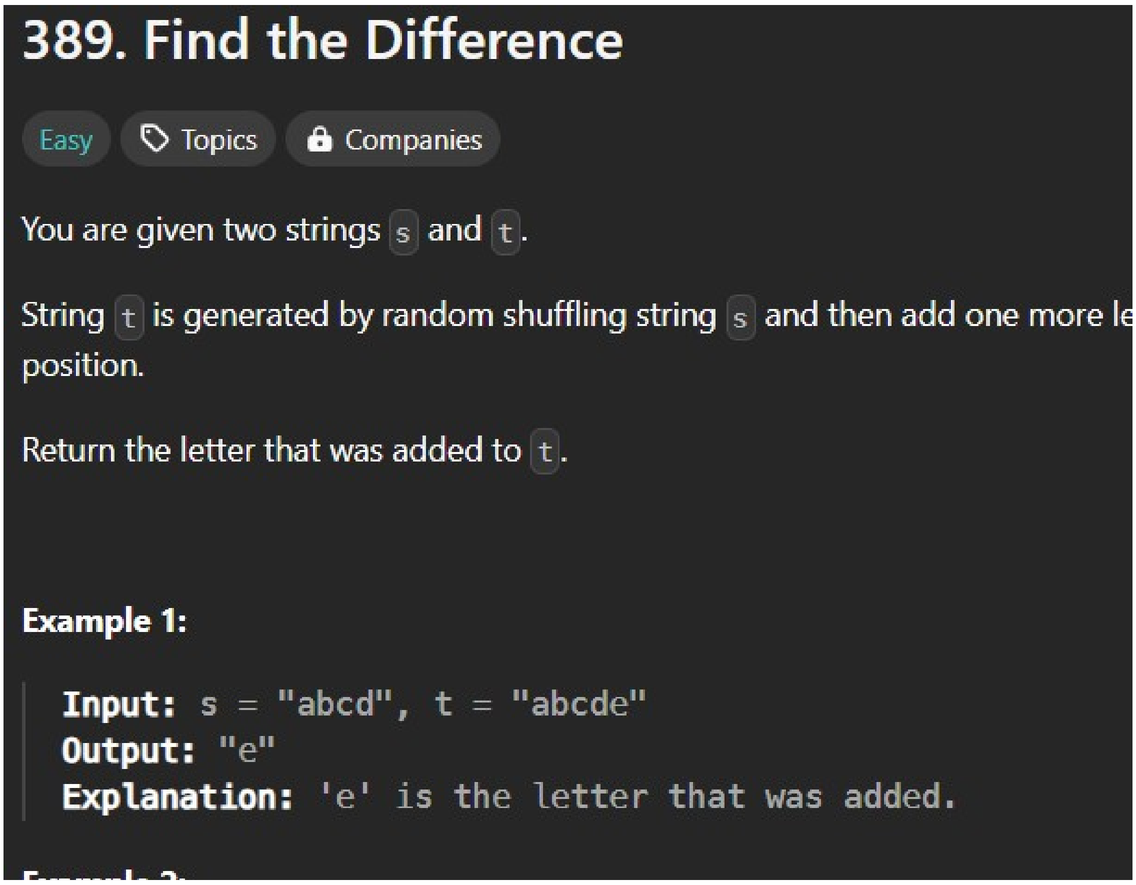
DEPARTMENT OF
COMPUTERSCIENCE & ENGINEERING
Discover. Learn. Empower.

Assignment-5

Student Name: Tanvi Chhabra **UID: 22BCS15679**
Branch: CSE **Section: 22BCS_IOT_605 B**
Semester: 6th **DOP: 05-03-25**
Subject: Advanced Programming **Subject Code: 22CSH-351**

1.Question:


389. Find the Difference

Easy · Topics · Companies

You are given two strings `s` and `t`.

String `t` is generated by random shuffling string `s` and then add one more le
position.

Return the letter that was added to `t`.

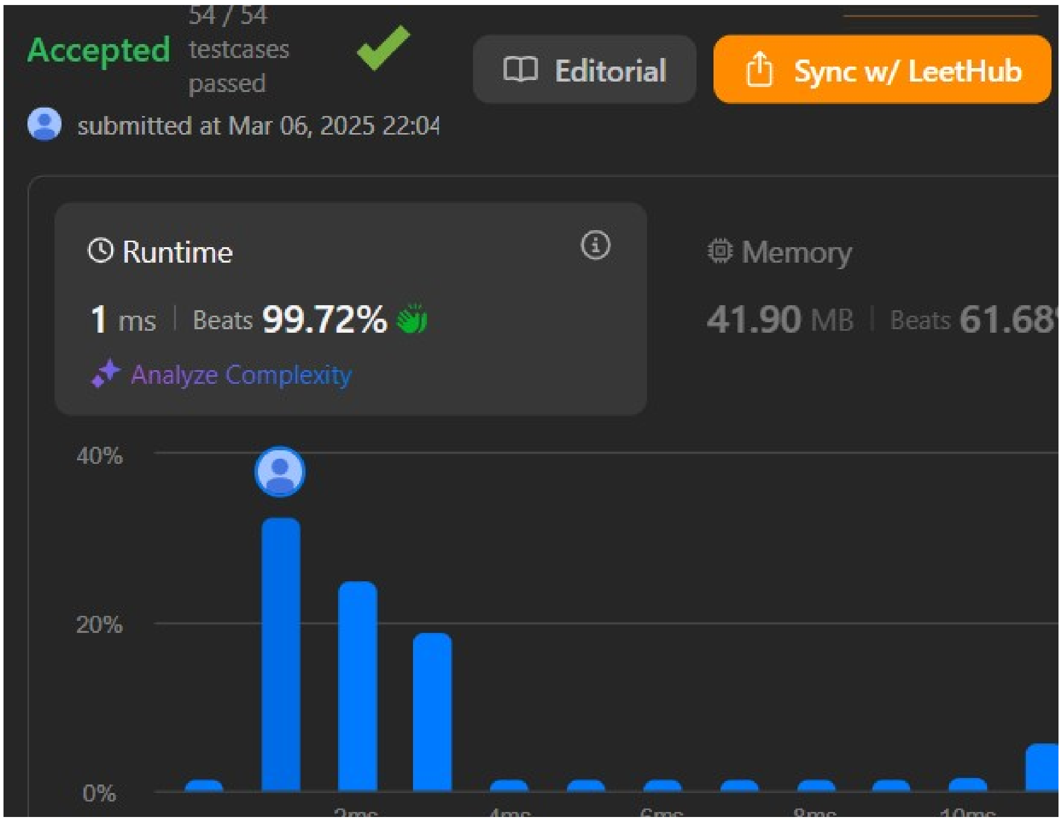
Example 1:

```
Input: s = "abcd", t = "abcde"
Output: "e"
Explanation: 'e' is the letter that was added.
```

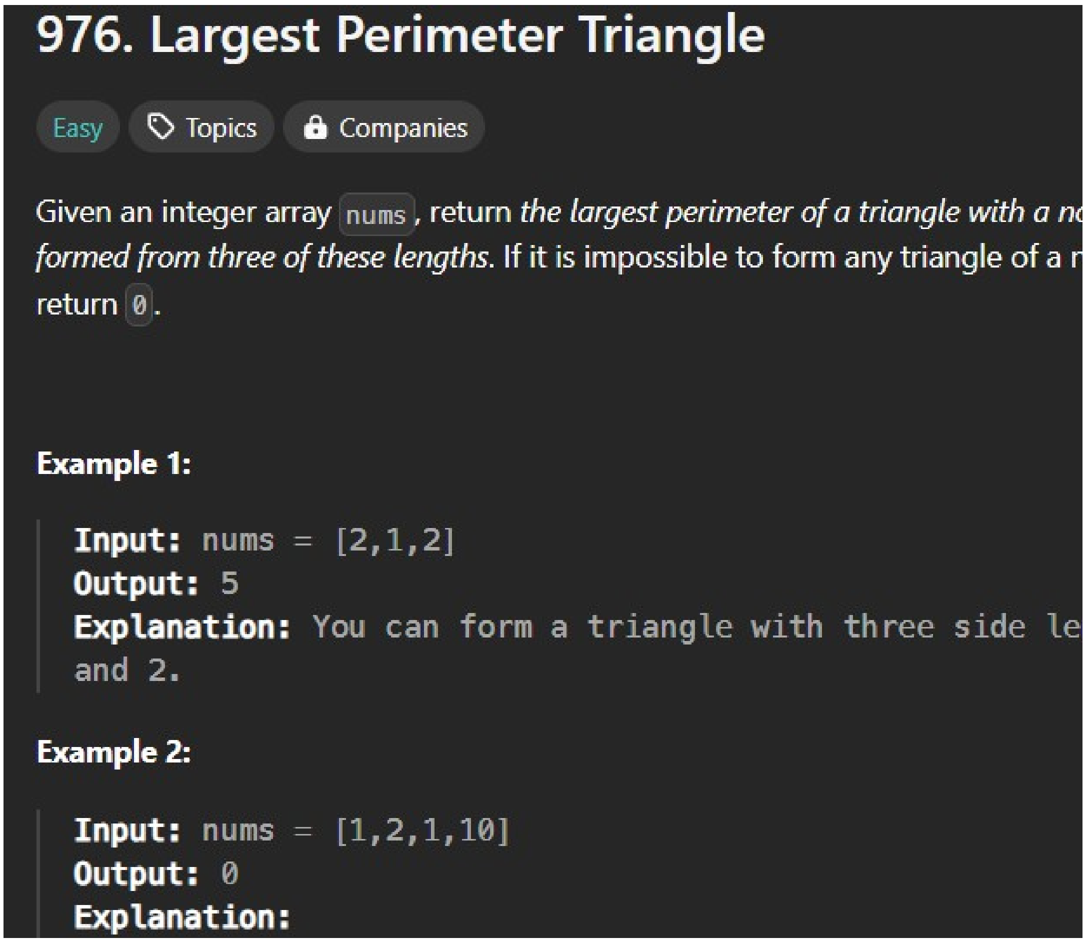
Code:

```
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;
    }
}
```

Output:


Accepted 54 / 54 testcases passed
submitted at Mar 06, 2025 22:04
Runtime
1 ms | Beats 99.72%
Memory
41.90 MB | Beats 61.68

2.Question:



976. Largest Perimeter Triangle

Given an integer array `nums`, return *the largest perimeter of a triangle with a n*
formed from three of these lengths. If it is impossible to form any triangle of a r
return `0`.

Example 1:

```
Input: nums = [2,1,2]
Output: 5
Explanation: You can form a triangle with three side le
and 2.
```

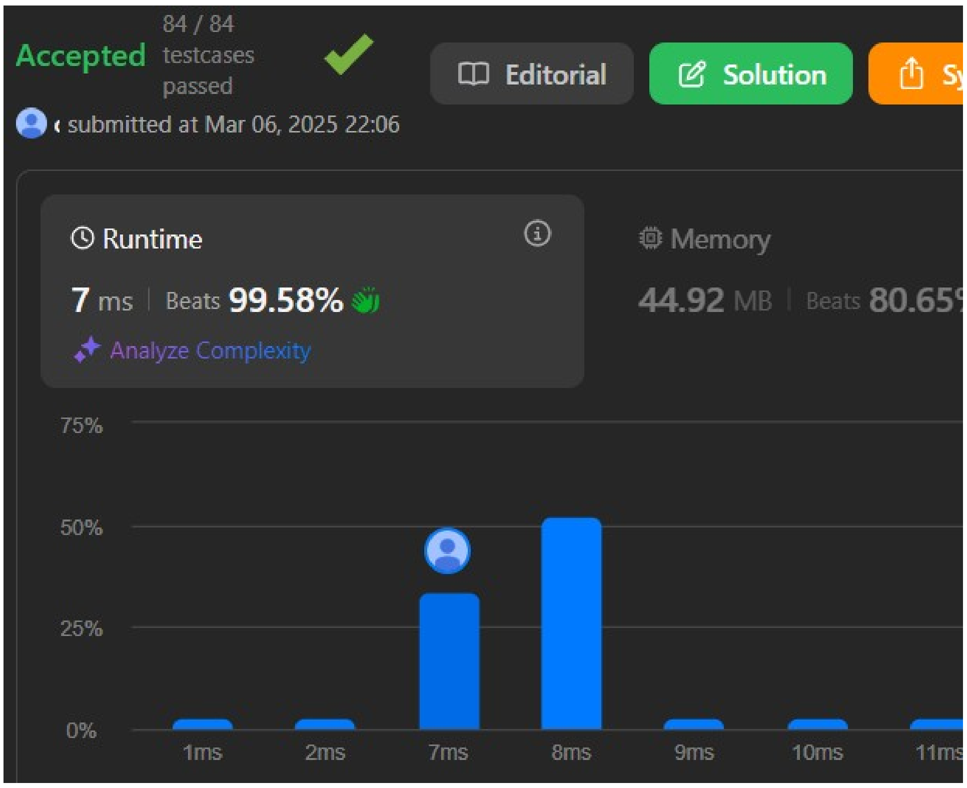
Example 2:

```
Input: nums = [1,2,1,10]
Output: 0
Explanation:
```

Code:

```
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;
   }

}
```

Output:



Accepted 84 / 84 testcases passed

submitted at Mar 06, 2025 22:06

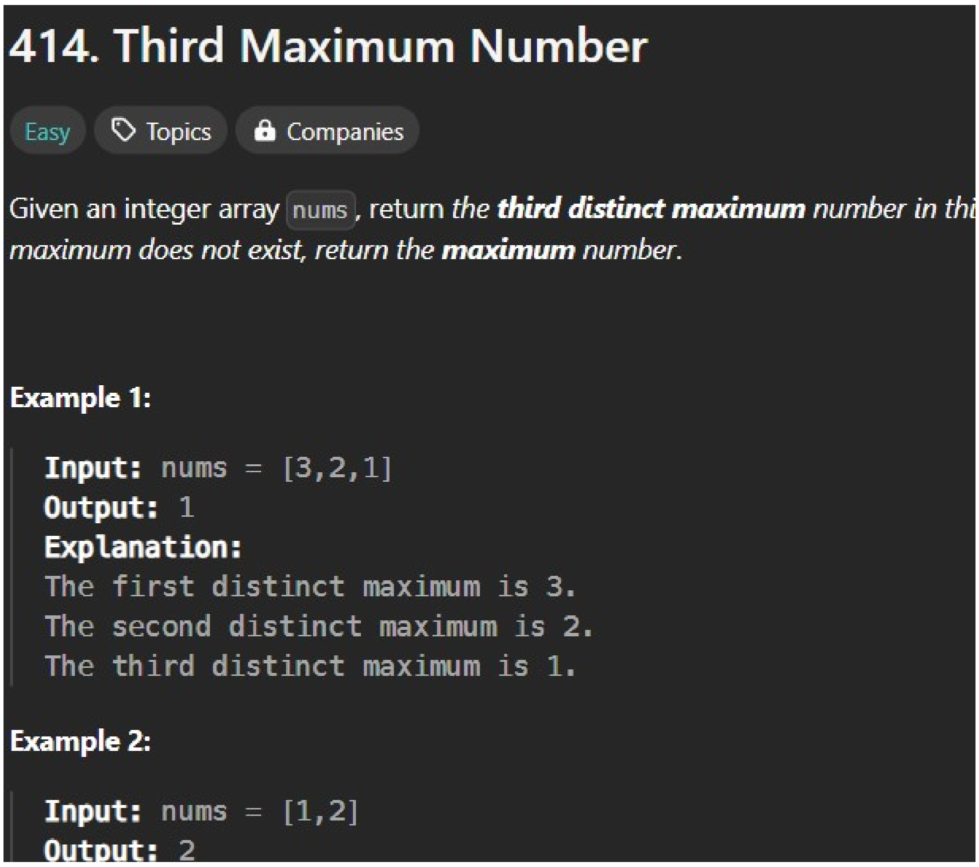
Runtime

7 ms | Beats 99.58%

Memory

44.92 MB | Beats 80.65

3.Question:



414. Third Maximum Number

Easy Topics Companies

Given an integer array `nums`, return *the* ***third distinct maximum*** *number in thi*
maximum does not exist, return the ***maximum*** *number.*

Example 1:

```
Input: nums = [3,2,1]
Output: 1
Explanation:
The first distinct maximum is 3.
The second distinct maximum is 2.
The third distinct maximum is 1.
```

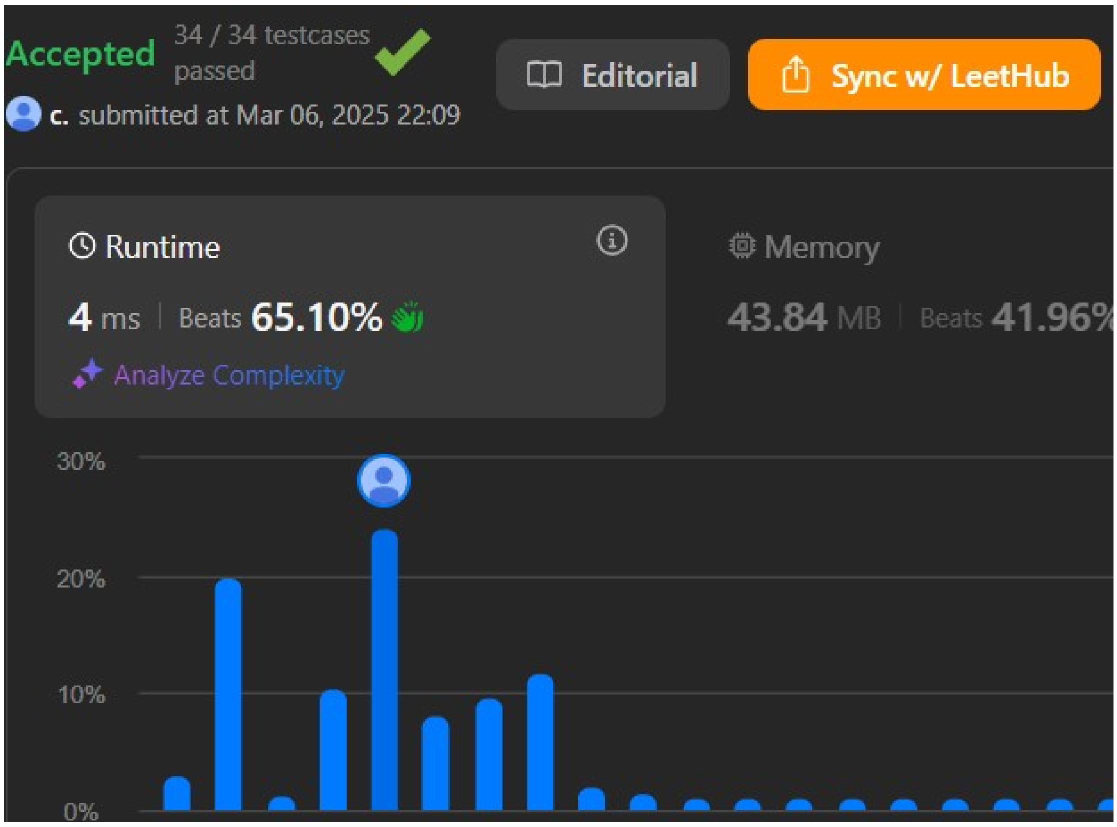
Example 2:

```
Input: nums = [1,2]
Output: 2
```

Code:

```
public int thirdMax(int[] nums) {
    Integer max1 = null;
    Integer max2 = null;
    Integer max3 = null;
    for (Integer n : nums) {
        if (n.equals(max1) || n.equals(max2) || n.equals(max3)) continue;
        if (max1 == null || n > max1) {
            max3 = max2;
            max2 = max1;
            max1 = n;
        } else if (max2 == null || n > max2) {
            max3 = max2;
            max2 = n;
        } else if (max3 == null || n > max3) {
            max3 = n;
        }
    }
    return max3 == null ? max1 : max3;
}
```

Output:

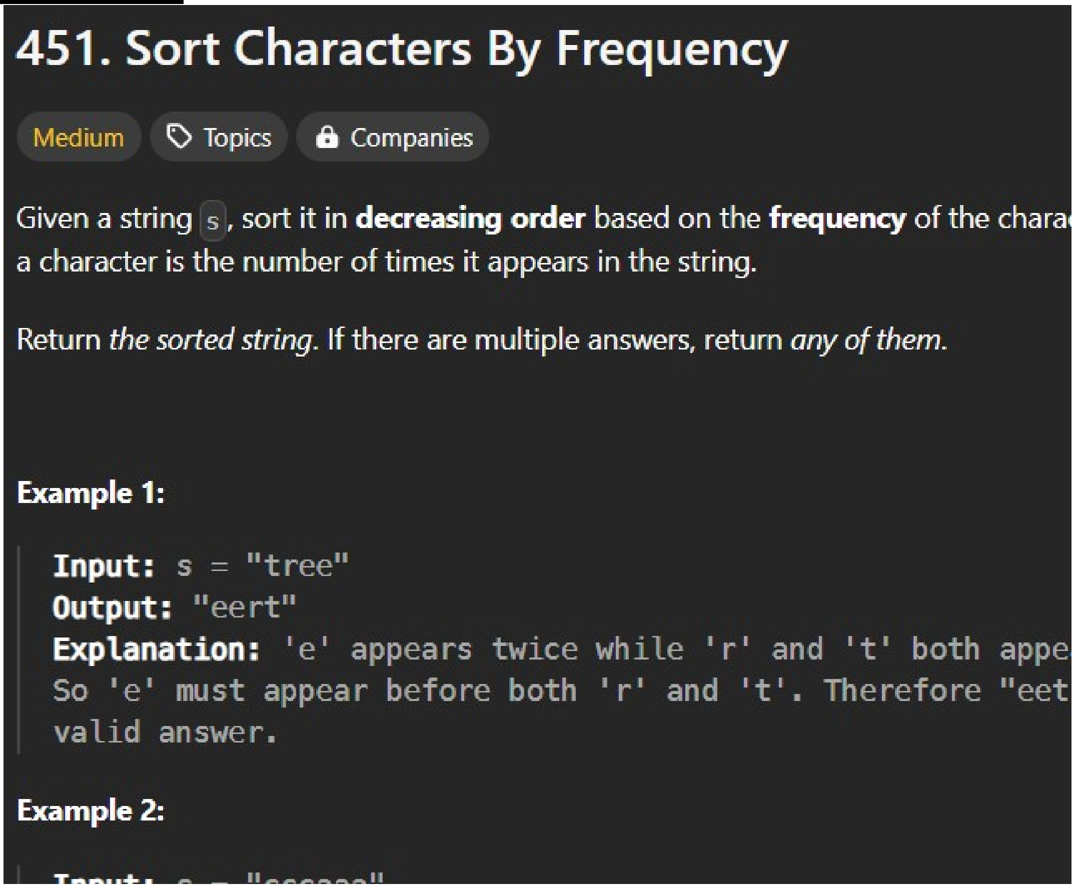


Accepted 34 / 34 testcases passed
c. submitted at Mar 06, 2025 22:09

Runtime
4 ms | Beats 65.10%

Memory
43.84 MB | Beats 41.96%

4.Question:



451. Sort Characters By Frequency

Medium Topics Companies

Given a string s, sort it in **decreasing order** based on the **frequency** of the chara
a character is the number of times it appears in the string.

Return *the sorted string*. If there are multiple answers, return *any of them*.

Example 1:

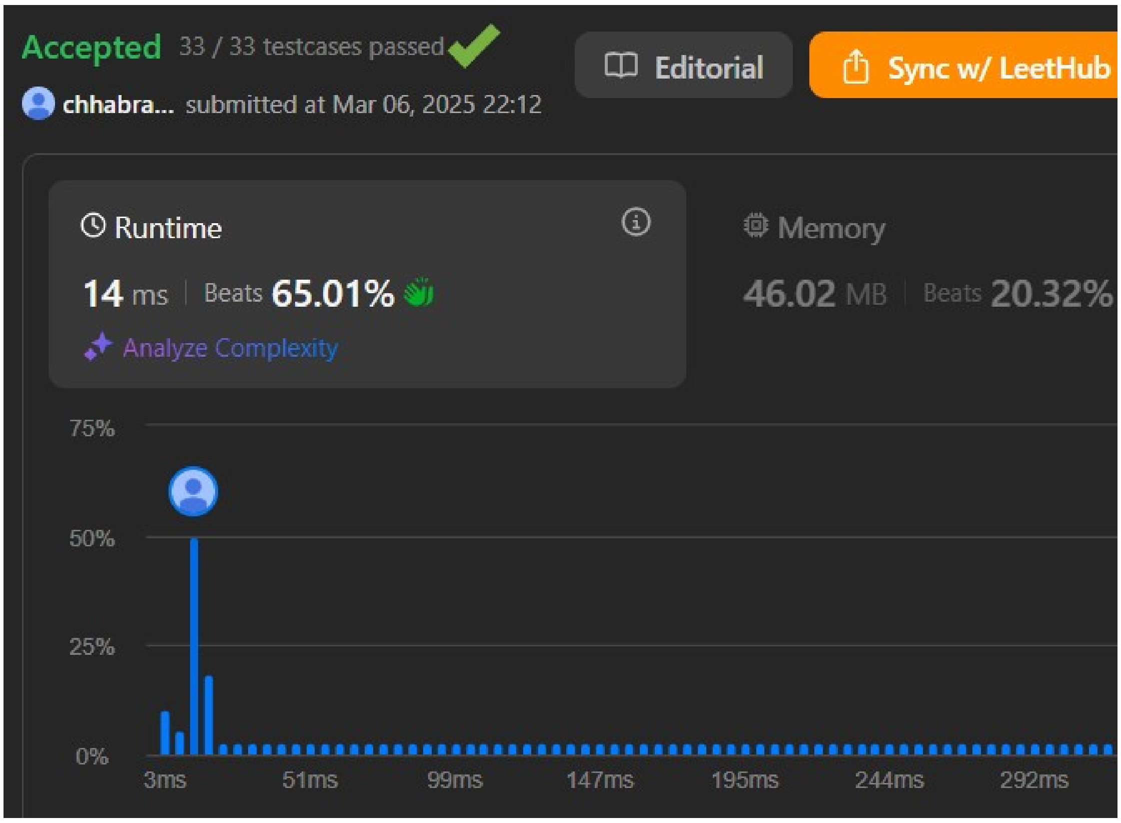
```
Input: s = "tree"
Output: "eert"
Explanation: 'e' appears twice while 'r' and 't' both appe
So 'e' must appear before both 'r' and 't'. Therefore "eet
valid answer.
```

Example 2:

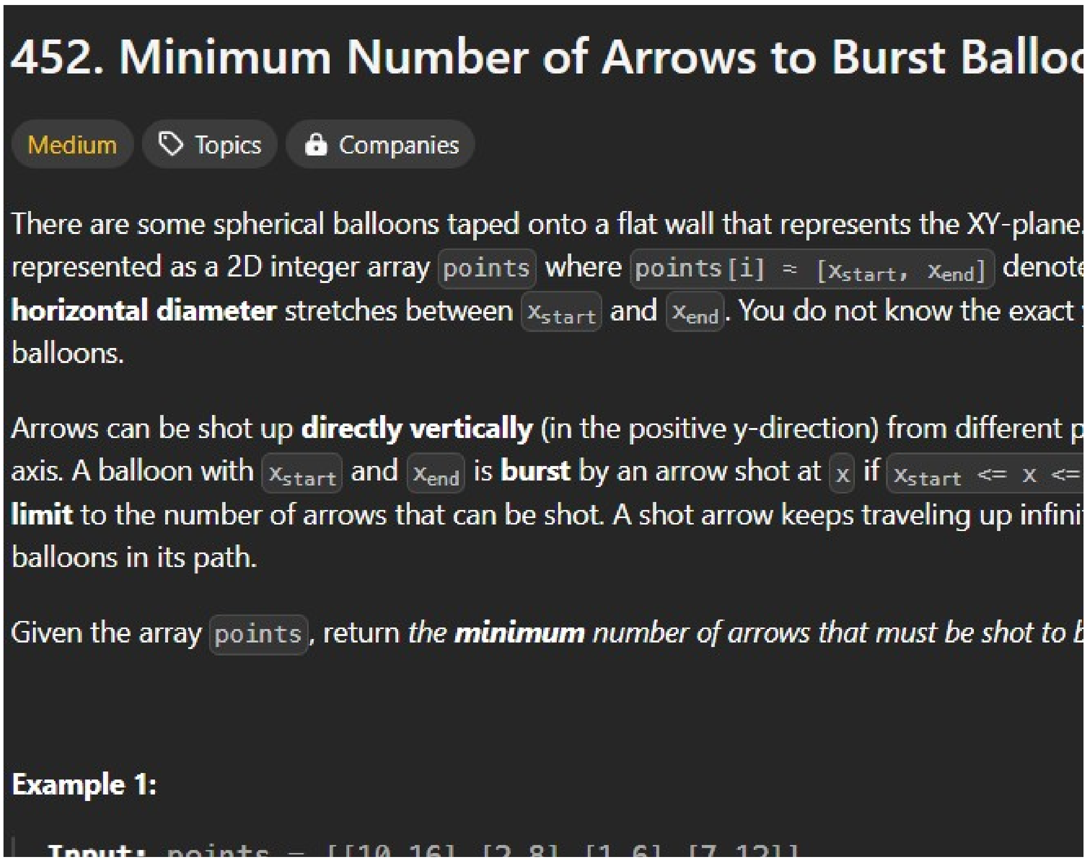
Code:

```
class pair{
    int freq;
    char ch;
    pair(int freq,char ch){
        this.freq=freq;
        this.ch=ch;
    }
}
class Solution {
    public String frequencySort(String s) {
        HashMap<Character,Integer> freq=new HashMap<>();
        int n=s.length();
        for(int i=0;i<n;i++){
            freq.put(s.charAt(i),freq.getOrDefault(s.charAt(i),0)+1);
        }
        PriorityQueue<pair> q=new PriorityQueue<>((a,b)->b.freq-a.freq);
        for(Map.Entry<Character,Integer> hm:freq.entrySet()){
            pair p=new pair(hm.getValue(),hm.getKey());
            q.offer(p);
        }
        StringBuilder sb=new StringBuilder();
        while(!q.isEmpty()){
            pair temp=q.poll();
            char ch=temp.ch;
            int fre=temp.freq;
            for(int i=0;i<fre;i++){
                sb.append(ch);
            }
        }
        return sb.toString();
    }
}
```

Output:


Accepted 33 / 33 testcases passed
chhabra... submitted at Mar 06, 2025 22:12

5.Question:



452. Minimum Number of Arrows to Burst Balloc

There are some spherical balloons taped onto a flat wall that represents the XY-plane. represented as a 2D integer array `points` where `points[i] = [xstart, xend]` denote **horizontal diameter** stretches between `xstart` and `xend`. You do not know the exact balloons.

Arrows can be shot up **directly vertically** (in the positive y-direction) from different p axis. A balloon with `xstart` and `xend` is **burst** by an arrow shot at `x` if `xstart <= x <=` **limit** to the number of arrows that can be shot. A shot arrow keeps traveling up infini balloons in its path.

Given the array `points`, return *the **minimum** number of arrows that must be shot to b*

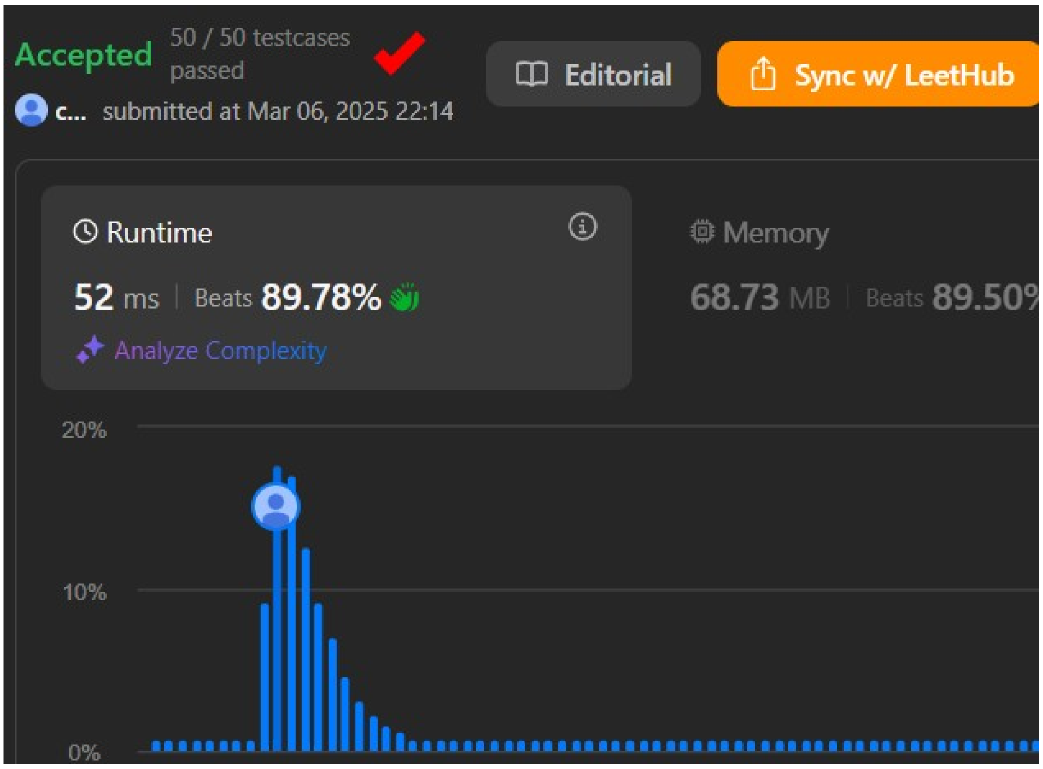
Example 1:

Input: points = [[10,16],[2,8],[1,6],[7,12]]

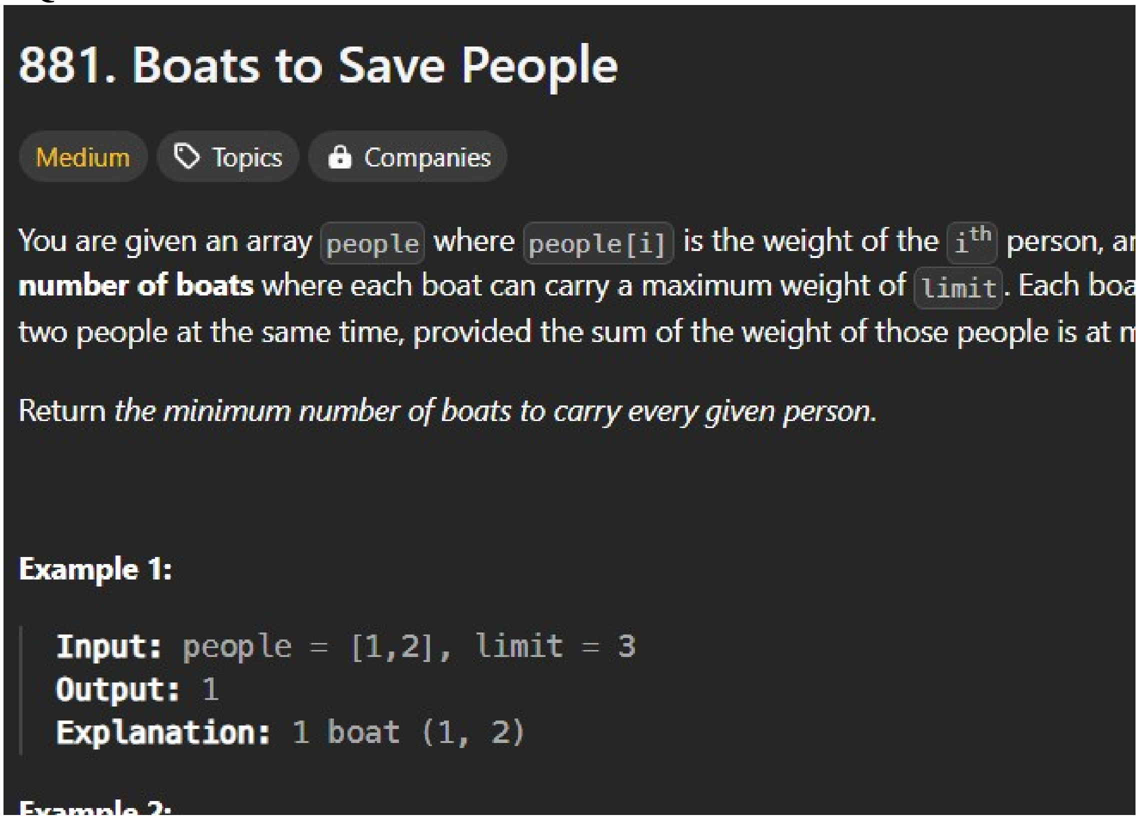
Code:

```
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:


Accepted 50 / 50 testcases passed
c... submitted at Mar 06, 2025 22:14

6.Question:



881. Boats to Save People

Medium Topics Companies

You are given an array `people` where `people[i]` is the weight of the `i`th person, ar
number of boats where each boat can carry a maximum weight of `limit`. Each boa
two people at the same time, provided the sum of the weight of those people is at m

Return *the minimum number of boats to carry every given person.*

Example 1:

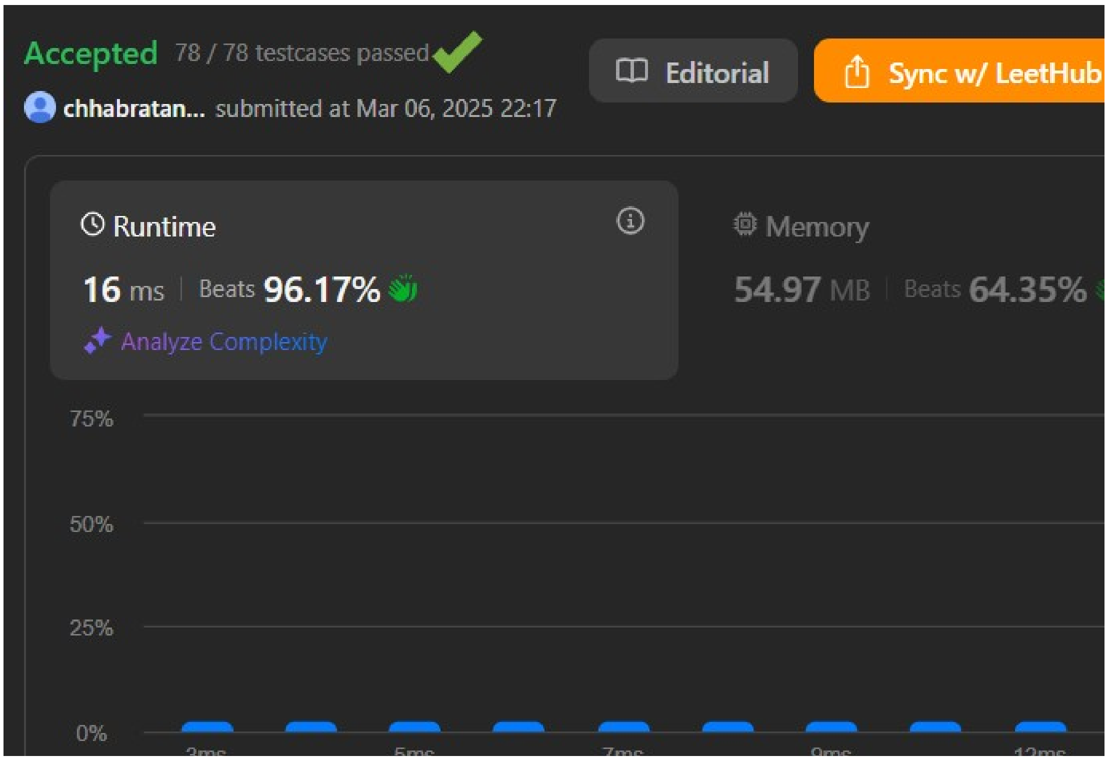
```
Input: people = [1,2], limit = 3
Output: 1
Explanation: 1 boat (1, 2)
```

Example 2:

Code:

```
class Solution {
        public int numRescueBoats(int[] people, int limit) {
        Arrays.sort(people);
        int i, j;
        for (i = 0, j = people.length - 1; i <= j; --j)
            if (people[i] + people[j] <= limit) ++i;
        return people.length - 1 - j;
    }
}
```

Output:



Accepted 78 / 78 testcases passed

chhabratan... submitted at Mar 06, 2025 22:17

Runtime

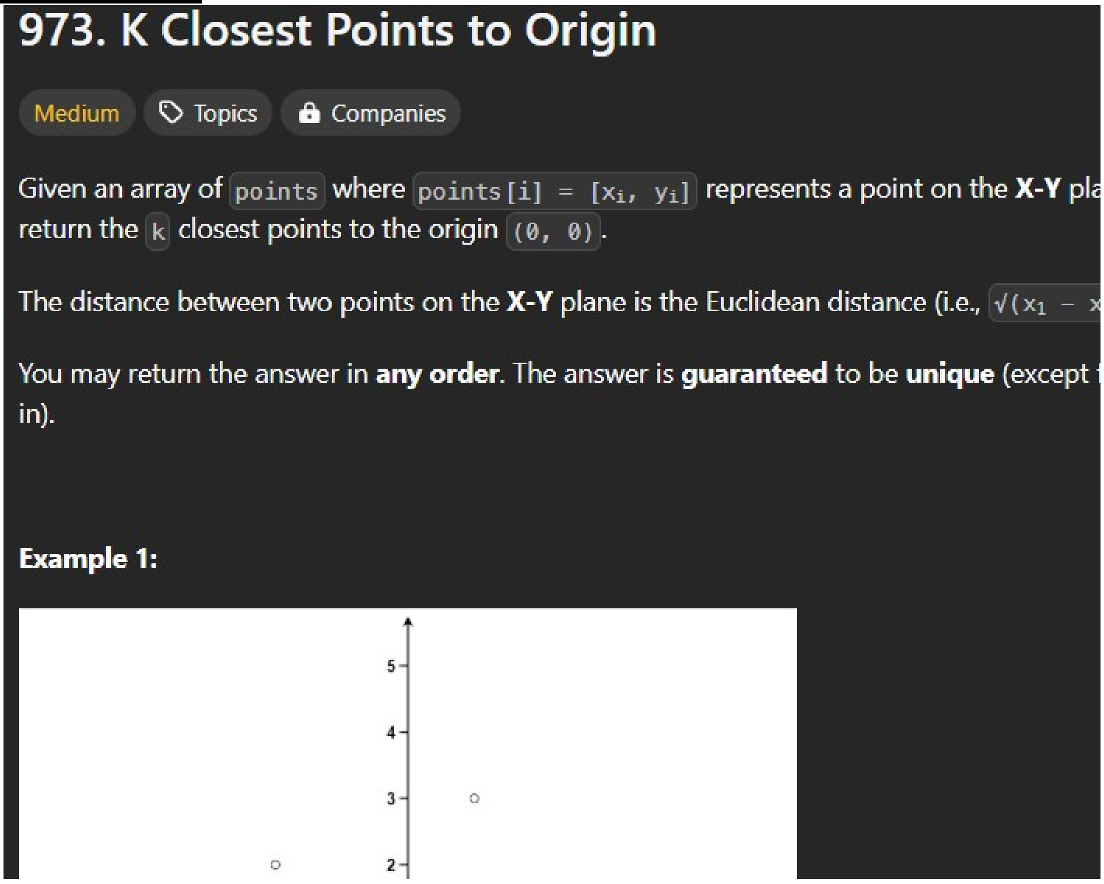
16 ms | Beats 96.17%

Analyze Complexity

Memory

54.97 MB | Beats 64.35%

7. Question:



973. K Closest Points to Origin

Medium · Topics · Companies

Given an array of `points` where `points[i] = [xi, yi]` represents a point on the **X-Y** pla
return the `k` closest points to the origin `(0, 0)`.

The distance between two points on the **X-Y** plane is the Euclidean distance (i.e., `√(x1 - x`

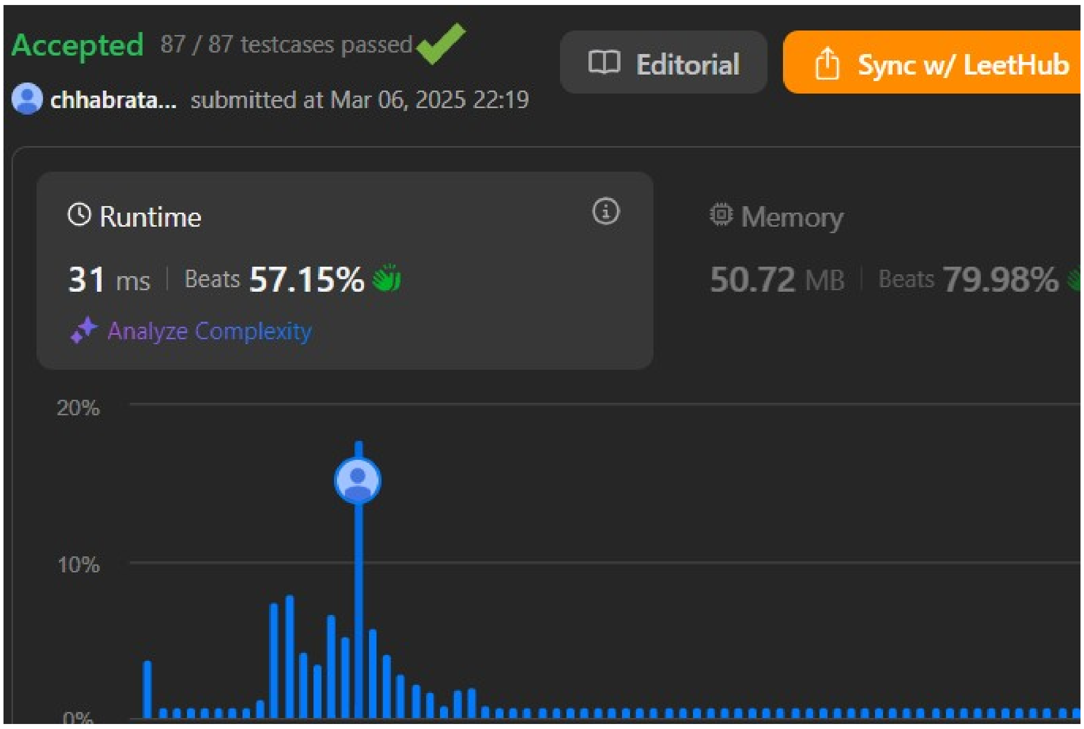
You may return the answer in **any order**. The answer is **guaranteed** to be **unique** (except
in).

Example 1:

Code:

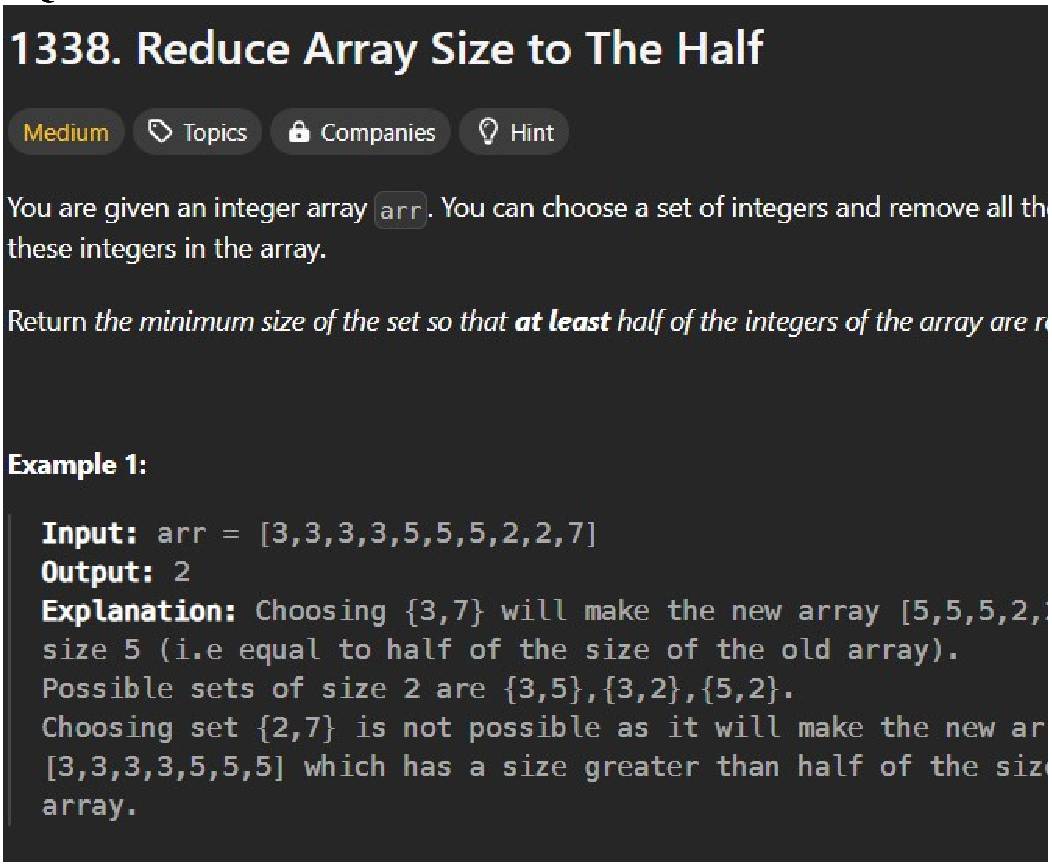
```
class Solution {
   public int[][] kClosest(int[][] points, int K) {
   PriorityQueue<int[]> pq = new PriorityQueue<int[]>((p1, p2) -> p2[0] * p2[0] + p2[1] *
p2[1] - p1[0] * p1[0] - p1[1] * p1[1]);
   for (int[] p : points) {
       pq.offer(p);
       if (pq.size() > K) {
           pq.poll();
       }
   }
   int[][] res = new int[K][2];
   while (K > 0) {
       res[--K] = pq.poll();
   }
   return res;
}
}
```

Output:



Accepted 87 / 87 testcases passed
chhabrata... submitted at Mar 06, 2025 22:19

8.Question:



1338. Reduce Array Size to The Half

Medium Topics Companies Hint

You are given an integer array `arr`. You can choose a set of integers and remove all th
these integers in the array.

Return *the minimum size of the set so that* ***at least*** *half of the integers of the array are r*

Example 1:

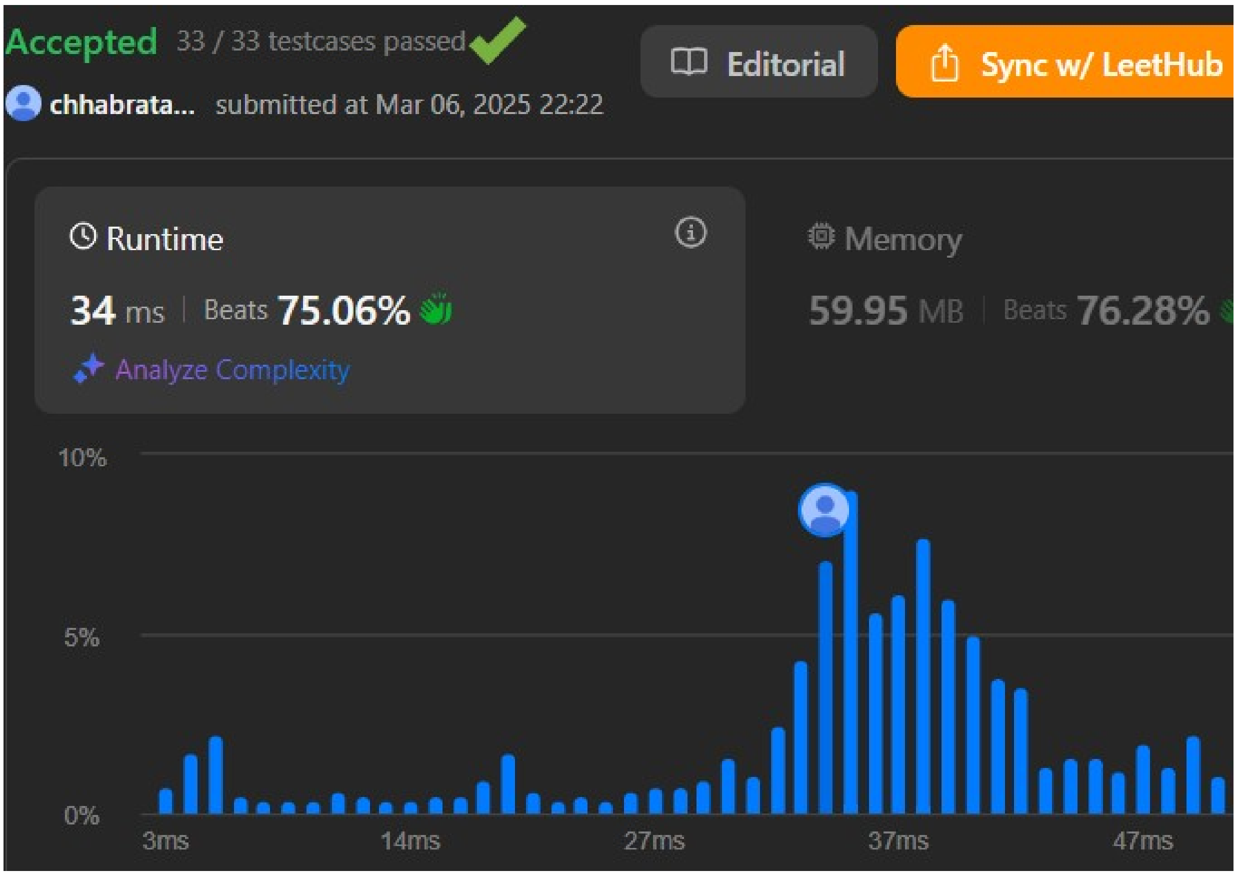
```
Input: arr = [3,3,3,3,5,5,5,2,2,7]
Output: 2
Explanation: Choosing {3,7} will make the new array [5,5,5,2,
size 5 (i.e equal to half of the size of the old array).
Possible sets of size 2 are {3,5},{3,2},{5,2}.
Choosing set {2,7} is not possible as it will make the new ar
[3,3,3,3,5,5,5] which has a size greater than half of the siz
array.
```

Code:

```
class Solution {
   public int minSetSize(int[] arr) {
         Map<Integer, Integer> map = new HashMap<>();
         ArrayList<Integer>[] list = new ArrayList[arr.length + 1];

         for (int num : arr) {
                map.put(num, map.getOrDefault(num, 0) + 1);
         }

         for (int num : map.keySet()) {
                int count = map.get(num);
                if (list[count] == null) {
                       list[count] = new ArrayList<Integer>();
                }
                list[count].add(num);
         }

         int steps = 0, res = 0;
         for (int i = arr.length; i > 0; i--) {
                List<Integer> cur = list[i];
                if (cur == null || cur.size() == 0) continue;
                for (int num : cur) {
                       steps += i;
                       res++;
```

```
                        if (steps >= arr.length / 2)
                            return res;
                }
        }
        return arr.length;
}
}
```

Output:



Accepted 33 / 33 testcases passed
chhabrata... submitted at Mar 06, 2025 22:22

Runtime
34 ms | Beats 75.06%

Memory
59.95 MB | Beats 76.28%