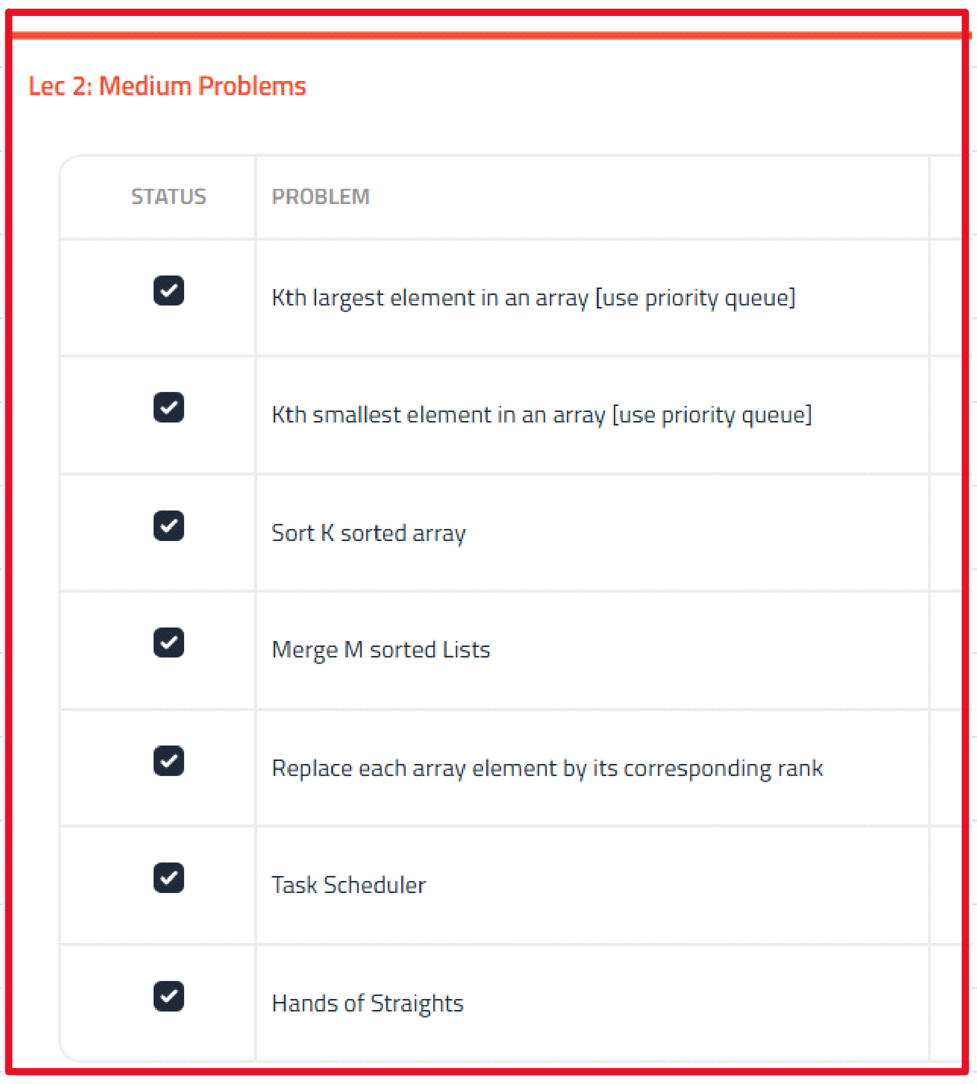
2. Medium Problems

19 March 2025 01:20

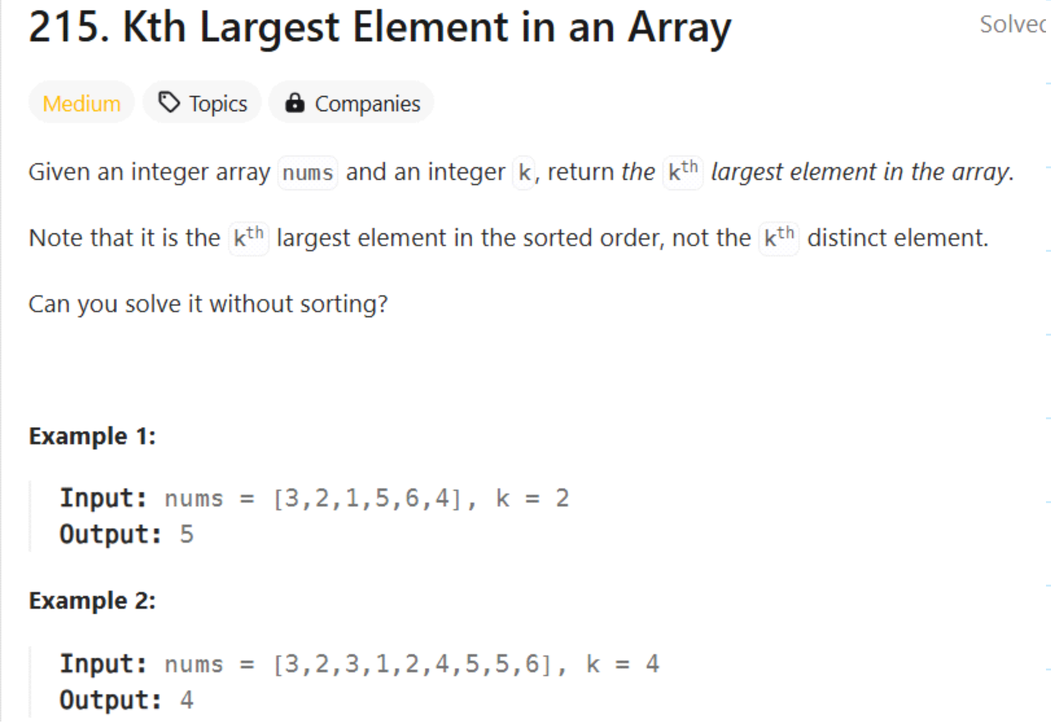


Lec 2: Medium Problems

STATUS	PROBLEM
☑	Kth largest element in an array [use priority queue]
☑	Kth smallest element in an array [use priority queue]
☑	Sort K sorted array
☑	Merge M sorted Lists
☑	Replace each array element by its corresponding rank
☑	Task Scheduler
☑	Hands of Straights

Question 1:

Kth largest element in an array [use priority queue]



215. Kth Largest Element in an Array

Solvec

Medium Topics Companies

Given an integer array `nums` and an integer `k`, return *the* k^{th} *largest element in the array.*

Note that it is the k^{th} largest element in the sorted order, not the k^{th} distinct element.

Can you solve it without sorting?

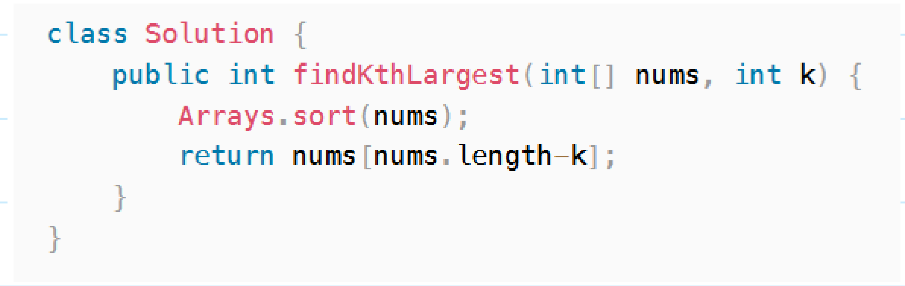
Example 1:

```
Input: nums = [3,2,1,5,6,4], k = 2
Output: 5
```

Example 2:

```
Input: nums = [3,2,3,1,2,4,5,5,6], k = 4
Output: 4
```

Sol1



```
class Solution {
    public int findKthLargest(int[] nums, int k) {
        Arrays.sort(nums);
        return nums[nums.length-k];
    }
}
```

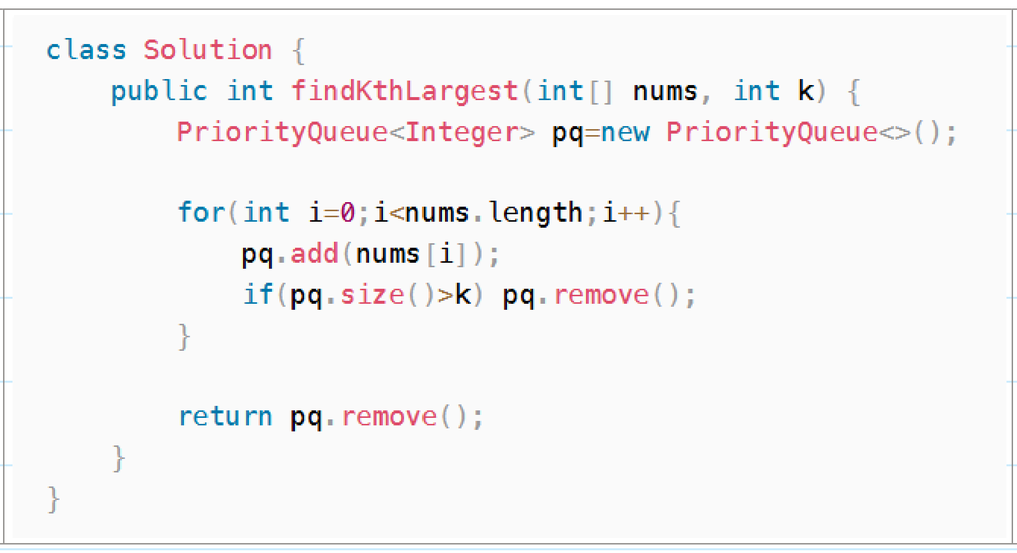
TC -O(NLogN)

Sol2

```
class Solution {
    public int findKthLargest(int[] nums, int k) {
        PriorityQueue<Integer> pq=new PriorityQueue<>
(Collections.reverseOrder());
        for(int i=0;i<nums.length;i++)
        pq.add(nums[i]);
        for(int i=0;i<k-1;i++){
            pq.remove();
        }
        return pq.remove();
    }
}
```

TC O(NLogN)

Sol3

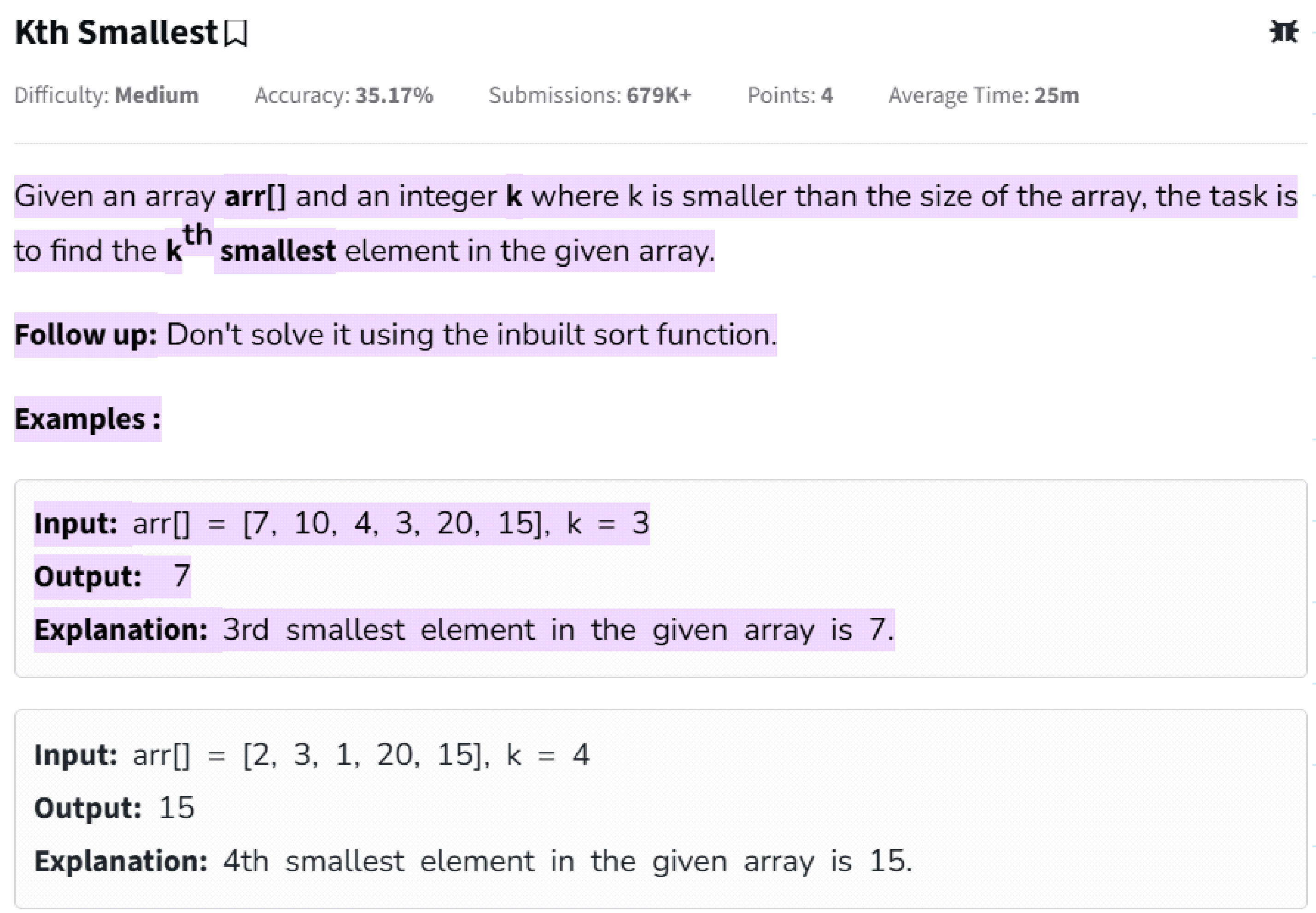


```
class Solution {
    public int findKthLargest(int[] nums, int k) {
        PriorityQueue<Integer> pq=new PriorityQueue<>();

        for(int i=0;i<nums.length;i++){
            pq.add(nums[i]);
            if(pq.size()>k) pq.remove();
        }

        return pq.remove();
    }
}
```

Tc 0(n*logK)

19 March 2025 01:31

Que2: Kth smallest element in an array [use priority queue]



Kth Smallest

Difficulty: **Medium** Accuracy: **35.17%** Submissions: **679K+** Points: **4** Average Time: **25m**

Given an array **arr[]** and an integer **k** where k is smaller than the size of the array, the task is to find the **k**th **smallest** element in the given array.

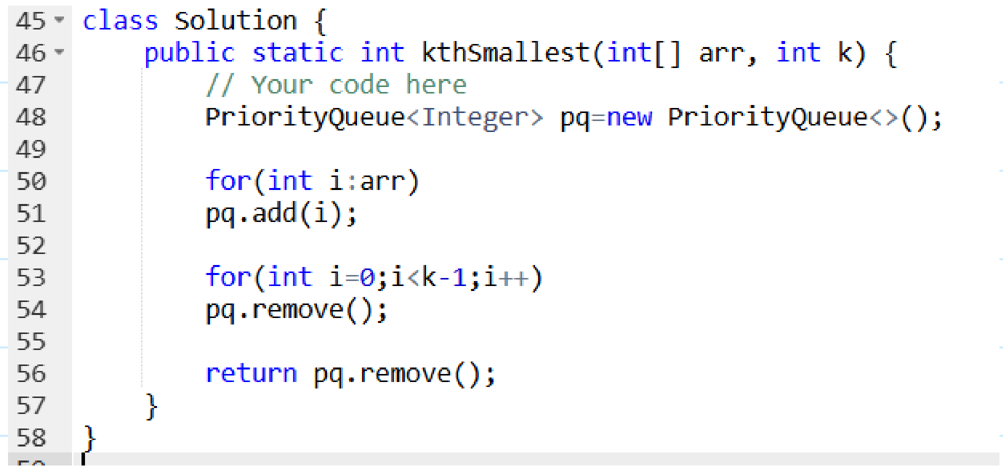
Follow up: Don't solve it using the inbuilt sort function.

Examples :

Input: arr[] = [7, 10, 4, 3, 20, 15], k = 3
Output: 7
Explanation: 3rd smallest element in the given array is 7.

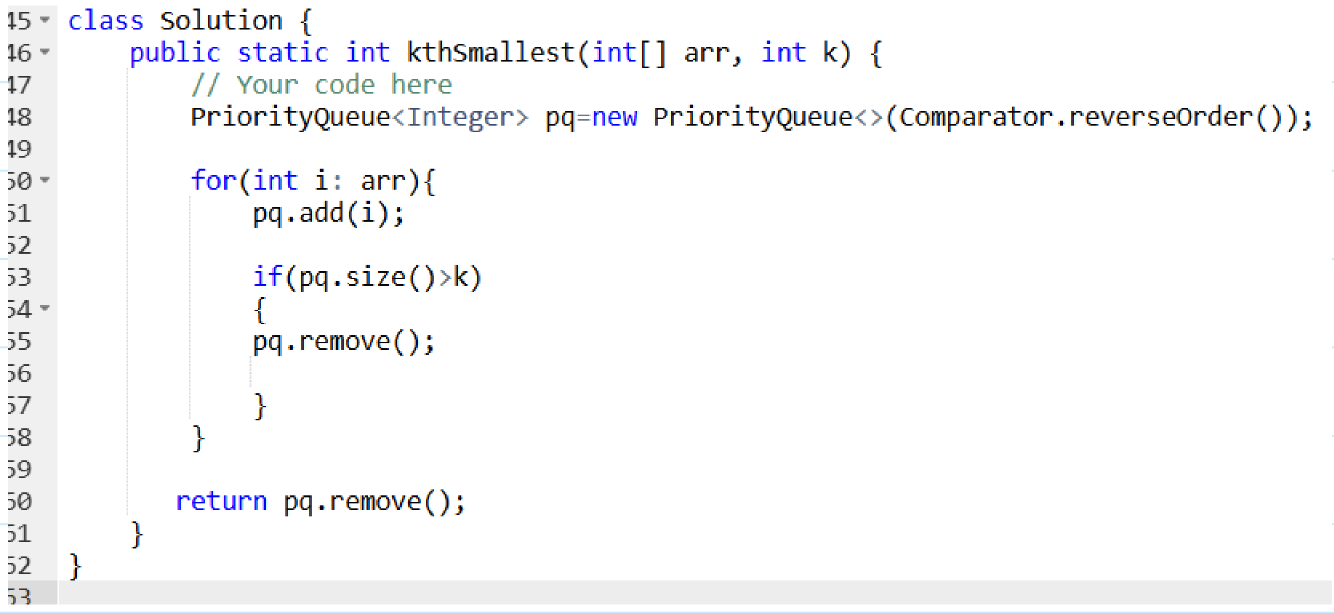
Input: arr[] = [2, 3, 1, 20, 15], k = 4
Output: 15
Explanation: 4th smallest element in the given array is 15.

Sol 1



```java
class Solution {
    public static int kthSmallest(int[] arr, int k) {
        // Your code here
        PriorityQueue<Integer> pq=new PriorityQueue<>();

        for(int i:arr)
        pq.add(i);

        for(int i=0;i<k-1;i++)
        pq.remove();

        return pq.remove();
    }
}
```

Sol2

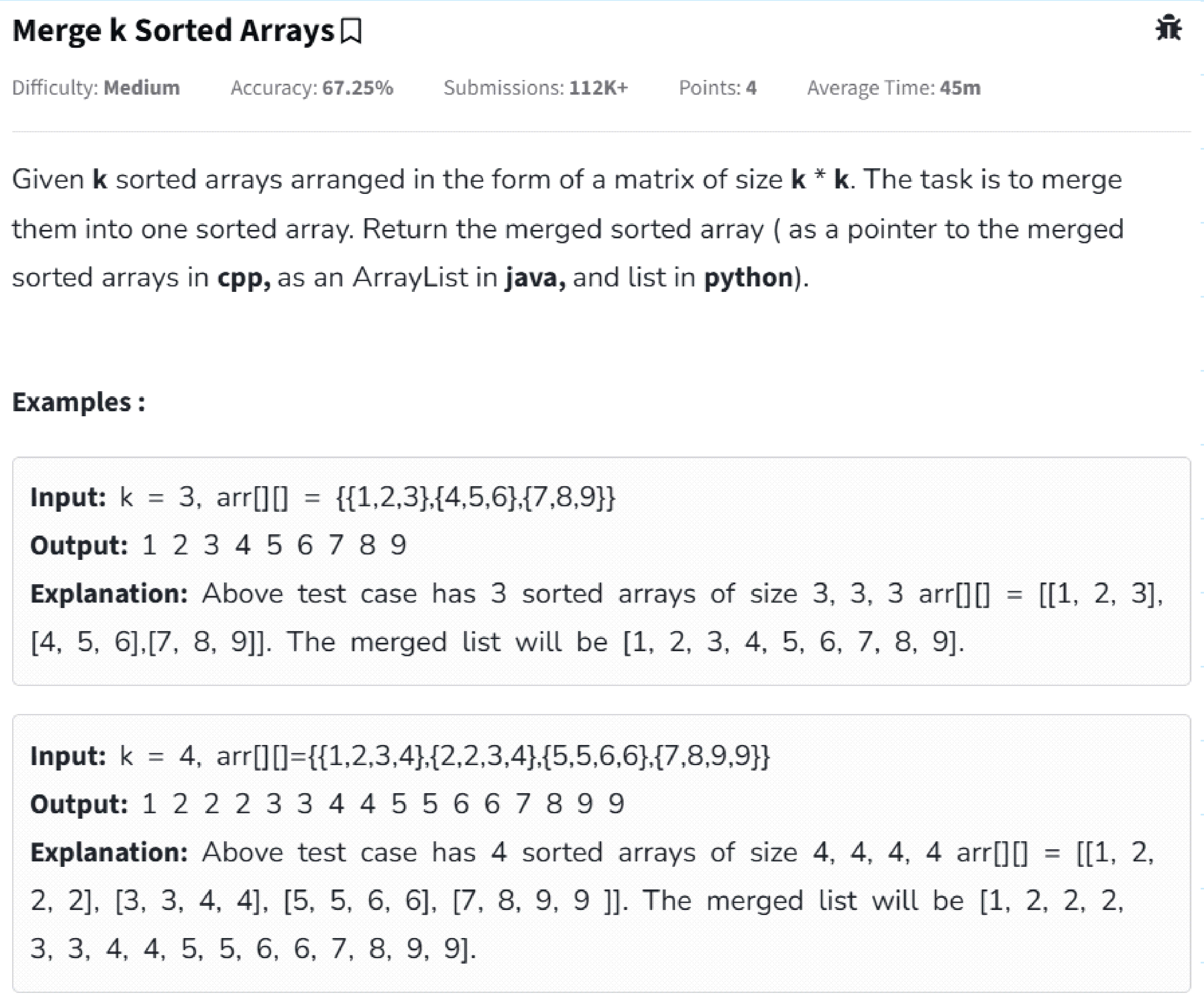


```java
class Solution {
    public static int kthSmallest(int[] arr, int k) {
        // Your code here
        PriorityQueue<Integer> pq=new PriorityQueue<>(Comparator.reverseOrder());

        for(int i: arr){
            pq.add(i);

            if(pq.size()>k)
            {
            pq.remove();

            }
        }

       return pq.remove();
    }
}
```

Tc --O(nlogk)

3rd Sort K sorted array

19 March 2025 01:38



Merge k Sorted Arrays

Difficulty: **Medium** Accuracy: **67.25%** Submissions: **112K+** Points: **4** Average Time: **45m**

Given **k** sorted arrays arranged in the form of a matrix of size **k** * **k**. The task is to merge them into one sorted array. Return the merged sorted array (as a pointer to the merged sorted arrays in **cpp,** as an ArrayList in **java,** and list in **python**).

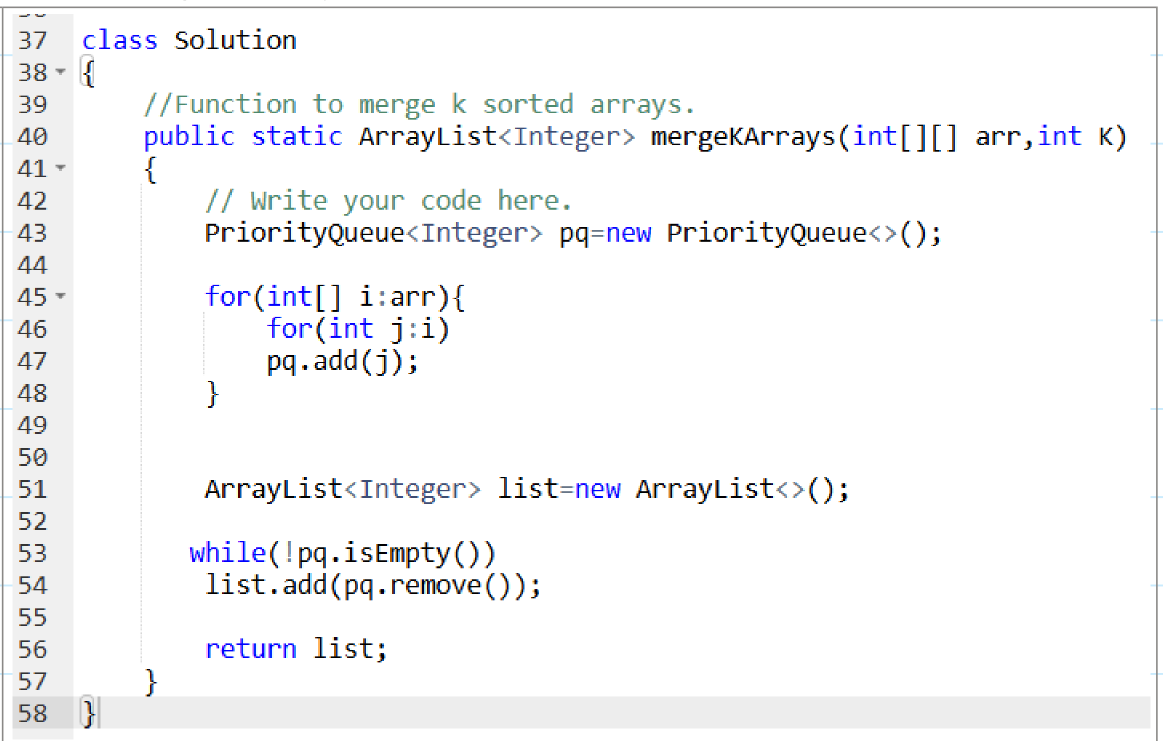
Examples :

Input: k = 3, arr[][] = {{1,2,3},{4,5,6},{7,8,9}}
Output: 1 2 3 4 5 6 7 8 9
Explanation: Above test case has 3 sorted arrays of size 3, 3, 3 arr[][] = [[1, 2, 3],[4, 5, 6],[7, 8, 9]]. The merged list will be [1, 2, 3, 4, 5, 6, 7, 8, 9].

Input: k = 4, arr[][]={{1,2,3,4},{2,2,3,4},{5,5,6,6},{7,8,9,9}}
Output: 1 2 2 2 3 3 4 4 5 5 6 6 7 8 9 9
Explanation: Above test case has 4 sorted arrays of size 4, 4, 4, 4 arr[][] = [[1, 2, 2, 2], [3, 3, 4, 4], [5, 5, 6, 6], [7, 8, 9, 9]]. The merged list will be [1, 2, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 8, 9, 9].

This is the solution using the Priority Queue

Sol1

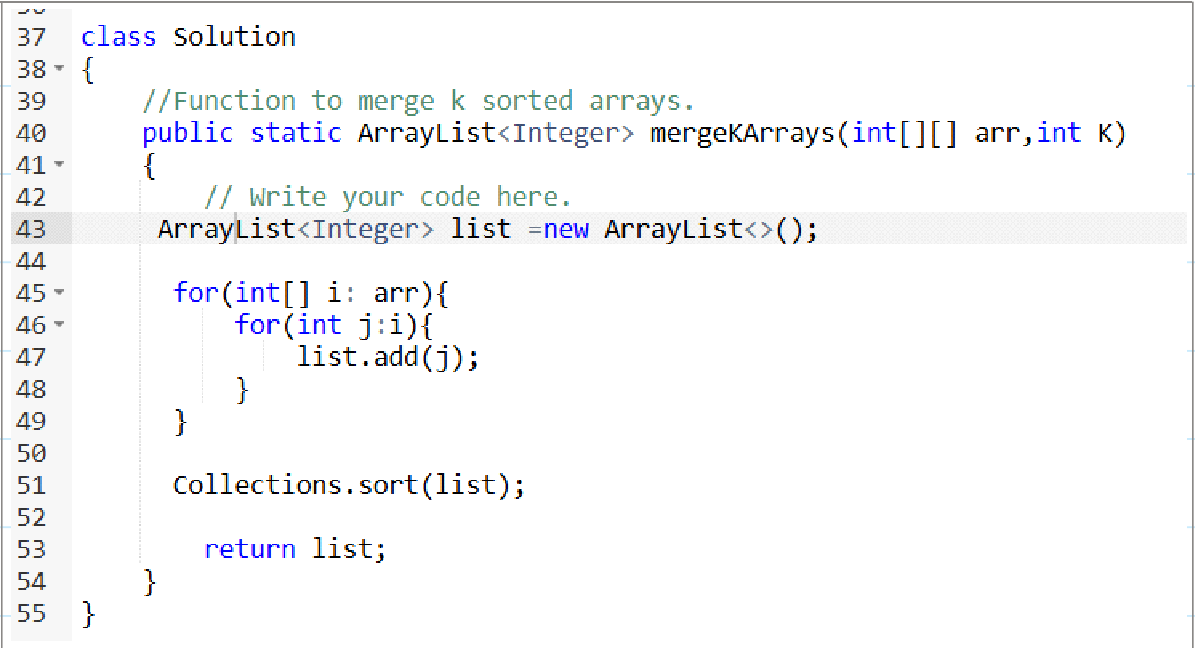


```
class Solution
{
    //Function to merge k sorted arrays.
    public static ArrayList<Integer> mergeKArrays(int[][] arr,int K)
    {
        // Write your code here.
        PriorityQueue<Integer> pq=new PriorityQueue<>();
        
        for(int[] i:arr){
            for(int j:i)
            pq.add(j);
        }
        
        
        ArrayList<Integer> list=new ArrayList<>();
        
       while(!pq.isEmpty())
        list.add(pq.remove());
        
        return list;
    }
}
```

Tc-(NlogN) where N is the total number of element in array

Insert in arraylist and sort

Sol2

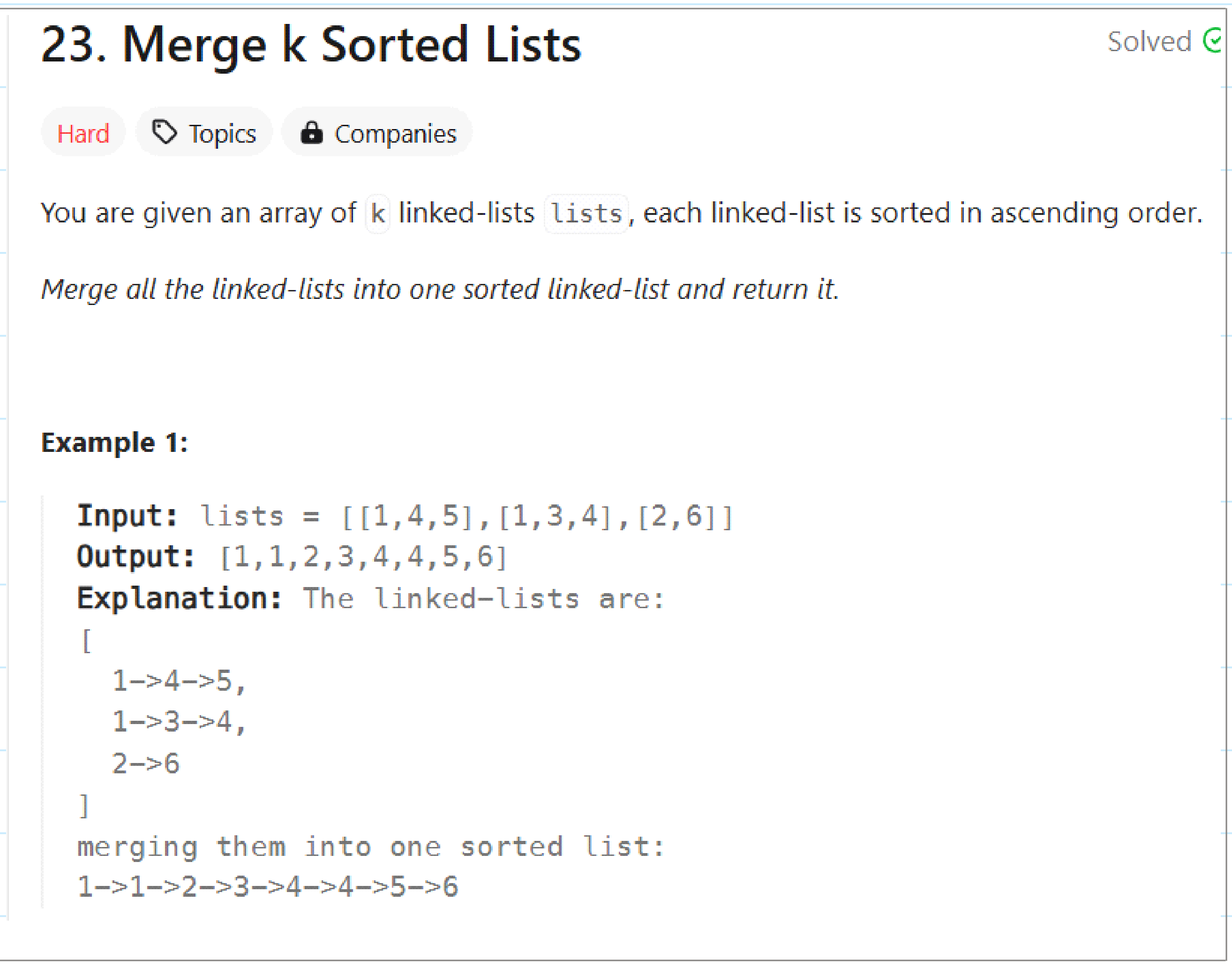


```
class Solution
{
    //Function to merge k sorted arrays.
    public static ArrayList<Integer> mergeKArrays(int[][] arr,int K)
    {
        // Write your code here.
     ArrayList<Integer> list =new ArrayList<>();
     
      for(int[] i: arr){
          for(int j:i){
              list.add(j);
          }
      }
      
      Collections.sort(list);
      
        return list;
    }
}
```

4. Merge M sorted Lists

19 March 2025 01:46

Question



23. Merge k Sorted Lists

Solved

Hard

You are given an array of `k` linked-lists `lists`, each linked-list is sorted in ascending order.

Merge all the linked-lists into one sorted linked-list and return it.

Example 1:

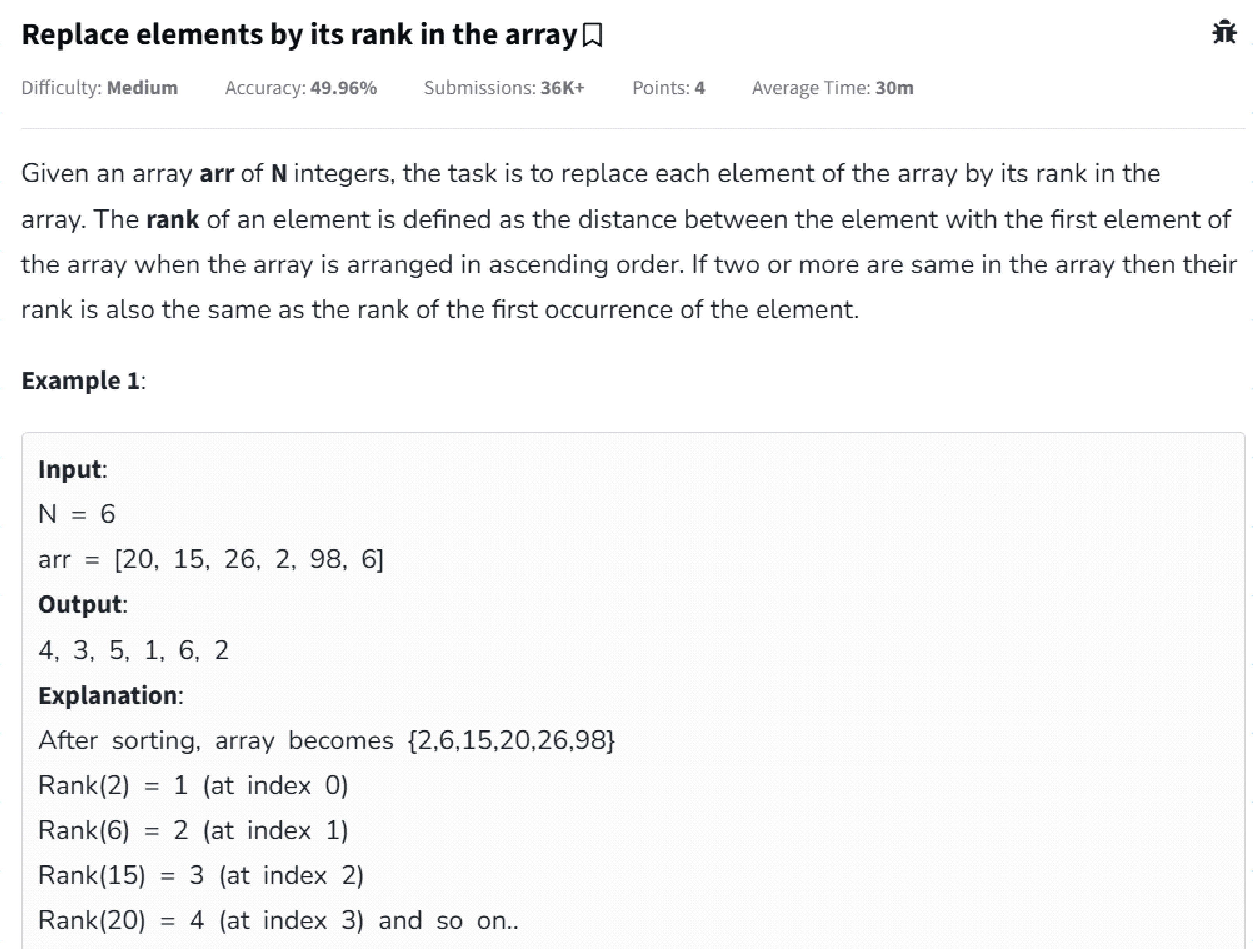
```
Input: lists = [[1,4,5],[1,3,4],[2,6]]
Output: [1,1,2,3,4,4,5,6]
Explanation: The linked-lists are:
[
  1->4->5,
  1->3->4,
  2->6
]
merging them into one sorted list:
1->1->2->3->4->4->5->6
```

Sol

```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {
    public ListNode mergeKLists(ListNode[] lists) {
        PriorityQueue<Integer> pq=new PriorityQueue<>();
        for(int i=0;i<lists.length;i++){
            ListNode temp=lists[i];
            while(temp!=null){
                pq.add(temp.val);
                temp=temp.next;
            }
        }

        ListNode head =new ListNode();
        ListNode temp=head;
        while(!pq.isEmpty())
        {
            ListNode n=new ListNode(pq.remove());
            temp.next=n;
            temp=n;
        }
        return head.next;
    }
}
```

5. Replace each array element by its corresponding rank

19 March 2025 01:56



Replace elements by its rank in the array

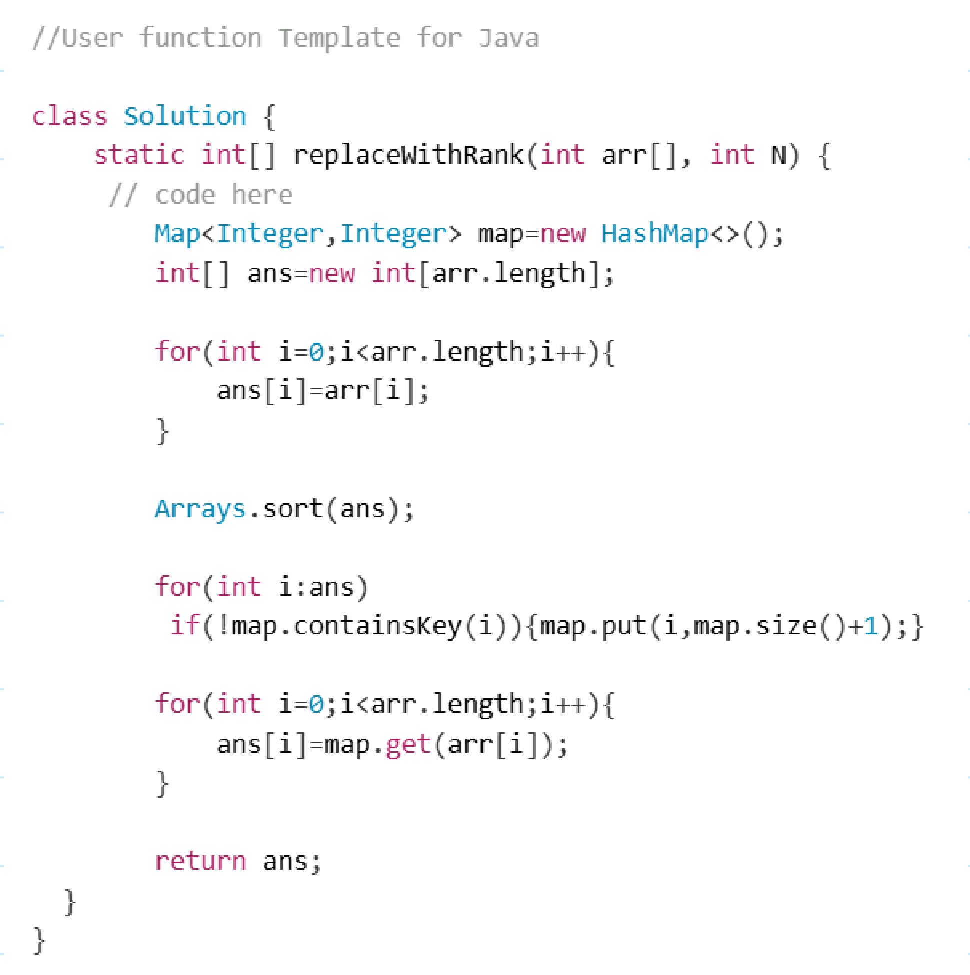
Difficulty: **Medium** Accuracy: **49.96%** Submissions: **36K+** Points: **4** Average Time: **30m**

Given an array **arr** of **N** integers, the task is to replace each element of the array by its rank in the array. The **rank** of an element is defined as the distance between the element with the first element of the array when the array is arranged in ascending order. If two or more are same in the array then their rank is also the same as the rank of the first occurrence of the element.

Example 1:

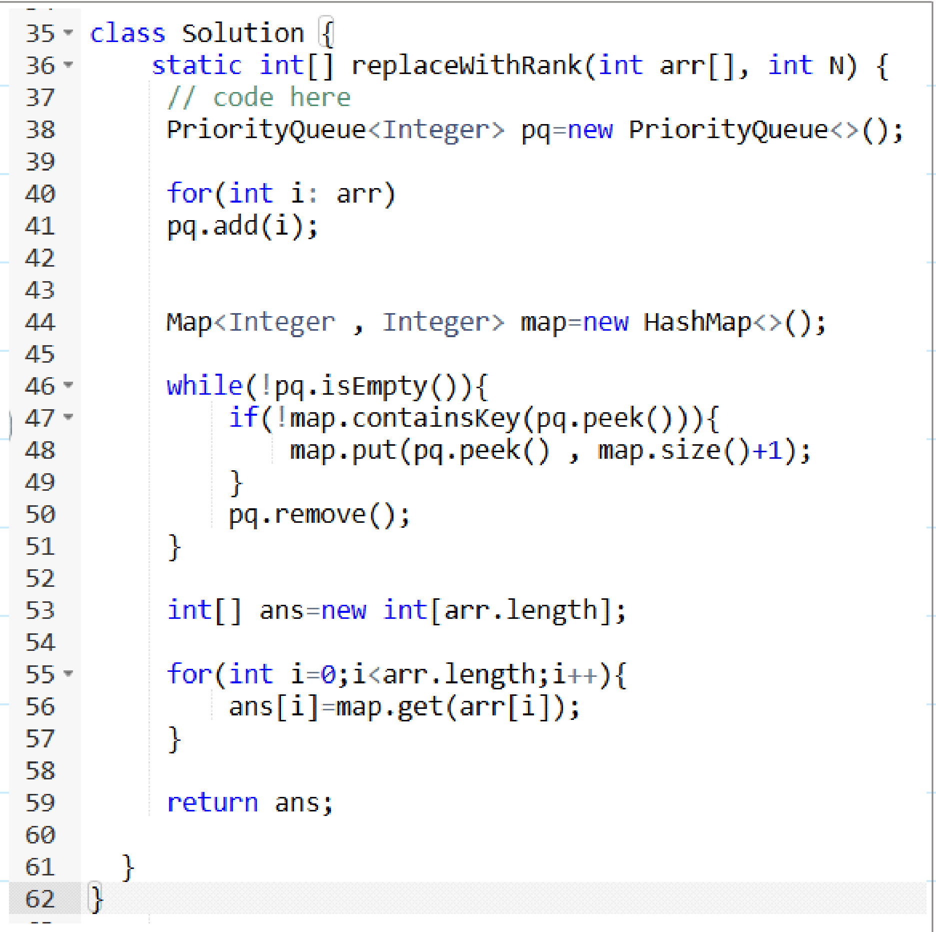
```
Input:
N = 6
arr = [20, 15, 26, 2, 98, 6]
Output:
4, 3, 5, 1, 6, 2
Explanation:
After sorting, array becomes {2,6,15,20,26,98}
Rank(2) = 1 (at index 0)
Rank(6) = 2 (at index 1)
Rank(15) = 3 (at index 2)
Rank(20) = 4 (at index 3) and so on..
```

Sol 1



```java
//User function Template for Java

class Solution {
    static int[] replaceWithRank(int arr[], int N) {
     // code here
        Map<Integer,Integer> map=new HashMap<>();
        int[] ans=new int[arr.length];

        for(int i=0;i<arr.length;i++){
            ans[i]=arr[i];
        }

        Arrays.sort(ans);

        for(int i:ans)
         if(!map.containsKey(i)){map.put(i,map.size()+1);}

        for(int i=0;i<arr.length;i++){
            ans[i]=map.get(arr[i]);
        }

        return ans;
  }
}
```

Sol2

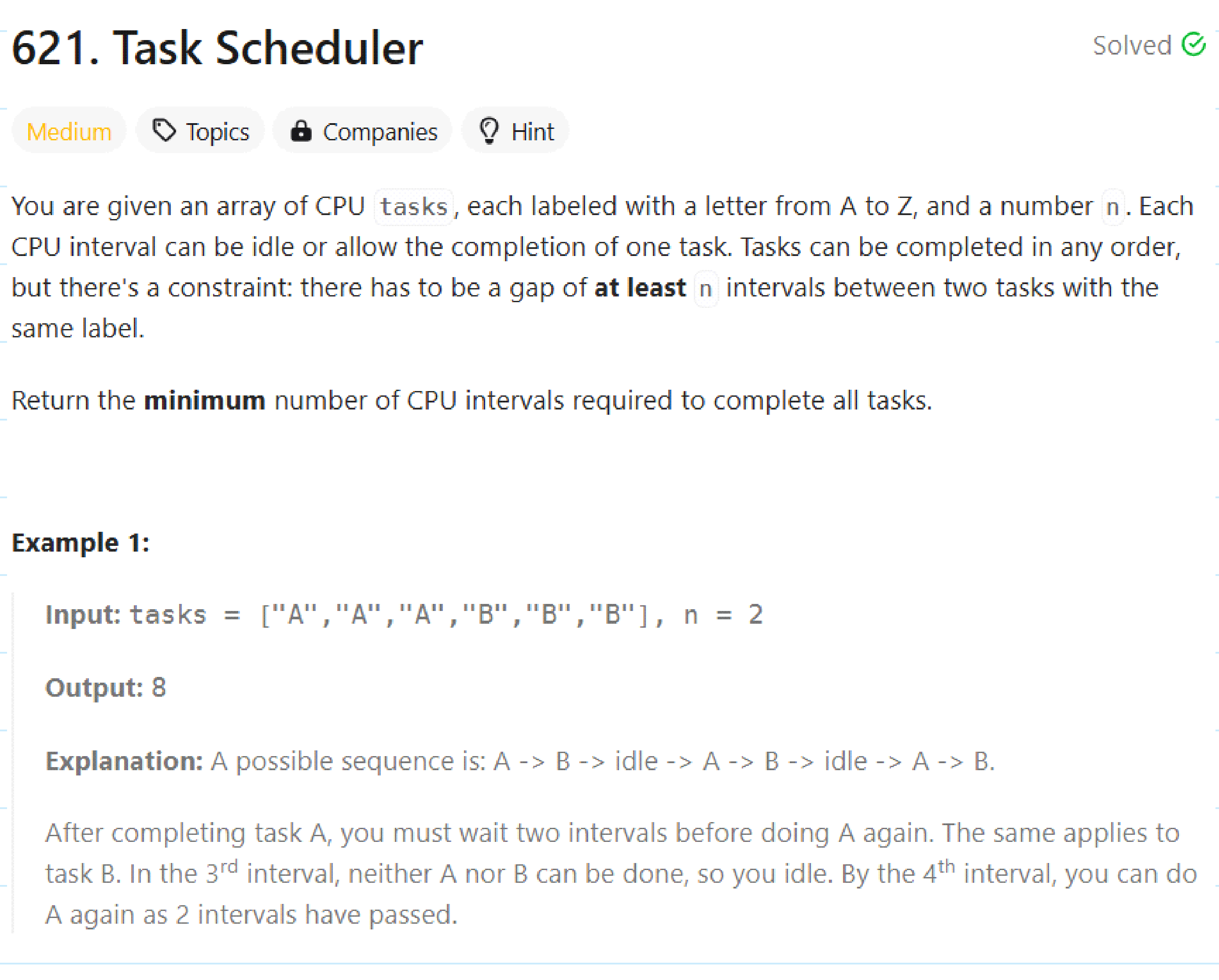


```java
class Solution {
    static int[] replaceWithRank(int arr[], int N) {
     // code here
     PriorityQueue<Integer> pq=new PriorityQueue<>();

     for(int i: arr)
     pq.add(i);


     Map<Integer , Integer> map=new HashMap<>();

     while(!pq.isEmpty()){
         if(!map.containsKey(pq.peek())){
             map.put(pq.peek() , map.size()+1);
         }
         pq.remove();
     }

     int[] ans=new int[arr.length];

     for(int i=0;i<arr.length;i++){
         ans[i]=map.get(arr[i]);
     }

     return ans;

   }
}
```

6. Task Scheduler

19 March 2025 02:02

Que



621. Task Scheduler

You are given an array of CPU `tasks`, each labeled with a letter from A to Z, and a number `n`. Each CPU interval can be idle or allow the completion of one task. Tasks can be completed in any order, but there's a constraint: there has to be a gap of **at least** `n` intervals between two tasks with the same label.

Return the **minimum** number of CPU intervals required to complete all tasks.

Example 1:

> **Input:** `tasks = ["A","A","A","B","B","B"], n = 2`
>
> **Output:** 8
>
> **Explanation:** A possible sequence is: A -> B -> idle -> A -> B -> idle -> A -> B.
>
> After completing task A, you must wait two intervals before doing A again. The same applies to task B. In the 3rd interval, neither A nor B can be done, so you idle. By the 4th interval, you can do A again as 2 intervals have passed.

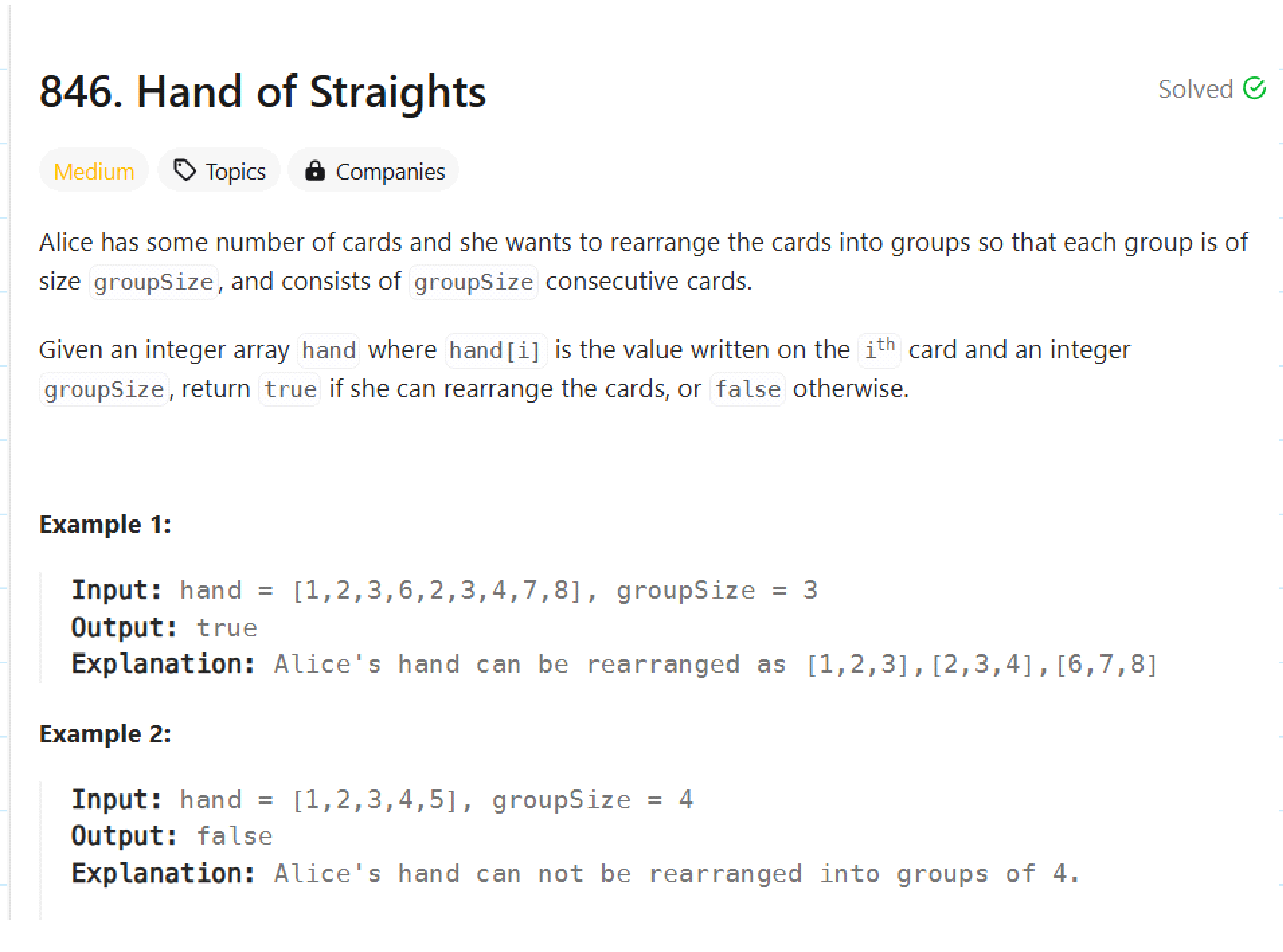
Sol

```
class Solution {
    public int leastInterval(char[] tasks, int n) {
        Map<Character , Integer> map=new HashMap<>();
        for(char ch: tasks){
            map.put(ch,map.getOrDefault(ch,0)+1);
        }
        PriorityQueue<Integer> pq=new PriorityQueue<>(Comparator.reverseOrder());
        for(int i:map.values()){
            pq.add(i);
        }
        int time=0;
        while(!pq.isEmpty()){
            List<Integer> list=new ArrayList<>();
            for(int i=0;i<=n;i++){
                if(!pq.isEmpty())
                list.add(pq.remove()-1);
            }
            for(int i: list){
                if(i!=0) pq.add(i);
            }

           if(pq.isEmpty()){
            time+=list.size();
           }else{
            time+=n+1;
           }
        }
        return time;
    }
}
```

7Hands of Straights

19 March 2025 02:08

Que



846. Hand of Straights

Solved

Medium | Topics | Companies

Alice has some number of cards and she wants to rearrange the cards into groups so that each group is of size `groupSize`, and consists of `groupSize` consecutive cards.

Given an integer array `hand` where `hand[i]` is the value written on the i^{th} card and an integer `groupSize`, return `true` if she can rearrange the cards, or `false` otherwise.

Example 1:

```
Input: hand = [1,2,3,6,2,3,4,7,8], groupSize = 3
Output: true
Explanation: Alice's hand can be rearranged as [1,2,3],[2,3,4],[6,7,8]
```

Example 2:

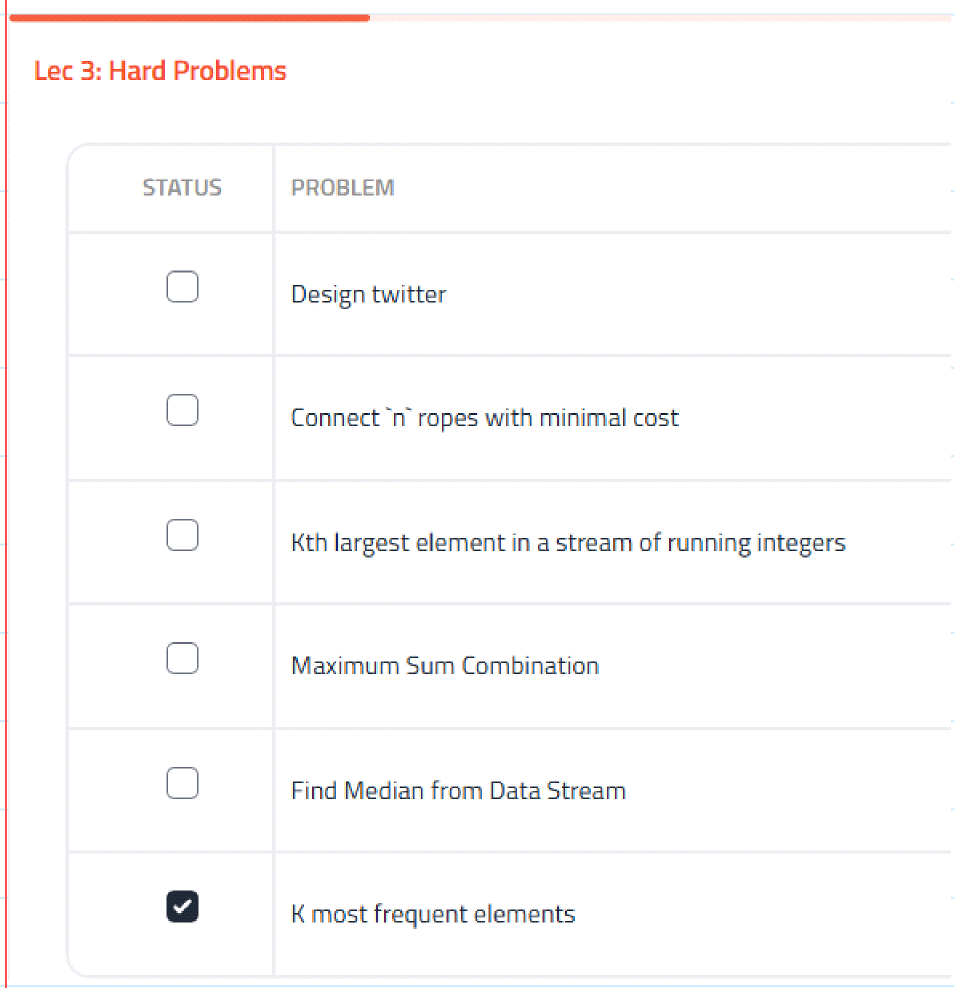
```
Input: hand = [1,2,3,4,5], groupSize = 4
Output: false
Explanation: Alice's hand can not be rearranged into groups of 4.
```

Sol

```java
class Solution {
    public boolean isNStraightHand(int[] hand, int groupSize) {
        if(hand.length%groupSize!=0) return false;
        Map<Integer , Integer> map=new HashMap<>();
        for(int i: hand)
        map.put(i,map.getOrDefault(i,0)+1);

        PriorityQueue<temp> pq=new PriorityQueue<>();
        for(Map.Entry<Integer, Integer> it : map.entrySet()){
            pq.add(new temp(it.getKey() , it.getValue()));
        }
        while(!pq.isEmpty()){
            List<temp> list=new ArrayList<>();
            int n=pq.peek().num-1;
            for(int i=0;i<groupSize;i++){
                if( pq.isEmpty() || n+1!=pq.peek().num) {

                    return false ;
                    }
                else{
                    list.add(new temp(pq.peek().num , pq.peek().fre-1));
                    n++;
                    pq.remove();
                }
            }
            for(int i=0;i<list.size();i++){
                if(list.get(i).fre!=0)
                pq.add(list.get(i));
            }
        }

        return true;
    }
    static class temp implements Comparable<temp>{
        int num;
        int fre;

        temp(int num ,int fre){
            this.num=num ;
            this.fre=fre;
        }
        public int compareTo(temp s2){
        return this.num-s2.num;}
    }
}
```

Hard Question

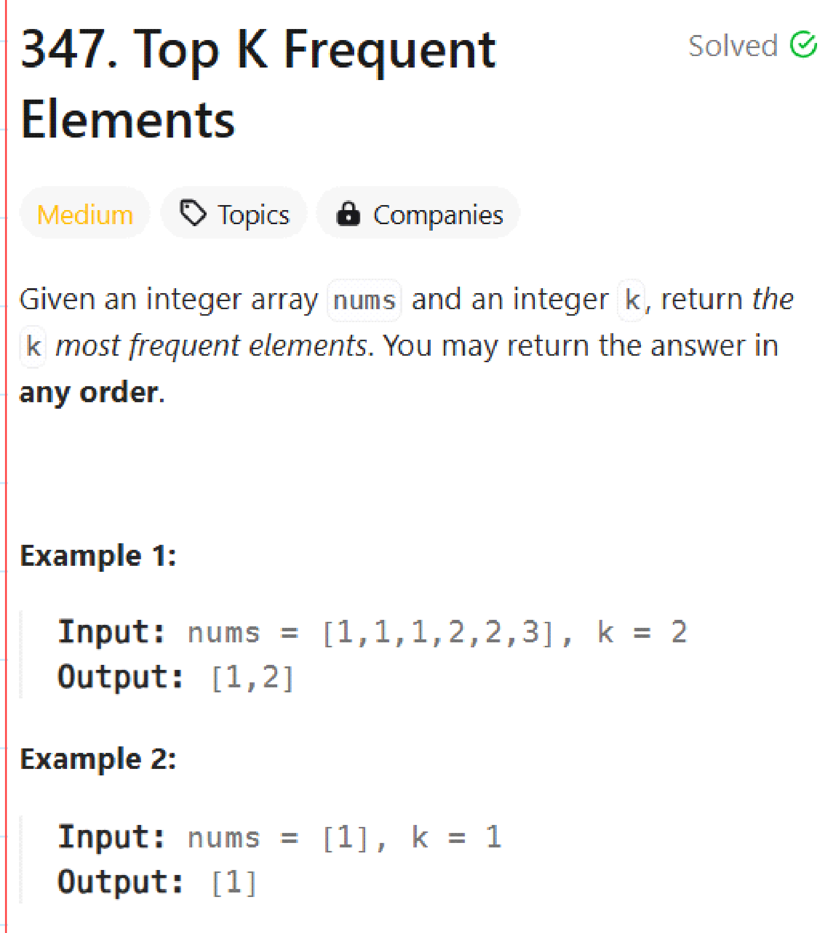
19 March 2025 02:11



Lec 3: Hard Problems

STATUS	PROBLEM
□	Design twitter
□	Connect `n` ropes with minimal cost
□	Kth largest element in a stream of running integers
□	Maximum Sum Combination
□	Find Median from Data Stream
☑	K most frequent elements

Last Question :
K most frequence element



347. Top K Frequent Elements

Solved ✓

Medium Topics Companies

Given an integer array `nums` and an integer `k`, return *the* `k` *most frequent elements*. You may return the answer in **any order**.

Example 1:

```
Input: nums = [1,1,1,2,2,3], k = 2
Output: [1,2]
```

Example 2:

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

Sol

```
class Solution {
    public int[] topKFrequent(int[] nums, int k) {
        Map<Integer ,Integer> map=new HashMap<>();
        for(int i:nums)
        map.put(i, map.getOrDefault(i,0)+1);

        PriorityQueue<Temp> pq=new PriorityQueue<>();
        for(Map.Entry<Integer, Integer> it : map.entrySet()){
            pq.add(new Temp(it.getKey() , it.getValue()));
            if(pq.size()>k) pq.remove();
        }
        int[] ans=new int[k];
        for(int i=0;i<k;i++){
            ans[i]=pq.remove().num;
        }
        return ans;
    }

    static class Temp implements Comparable<Temp>{
        int num;
        int freq;
        Temp(int num ,int freq){
            this.num=num;
            this.freq=freq;
        }
        public int compareTo(Temp t){
            return this.freq-t.freq;
        }
    }
}
```