

Q1. Merge Sorted Array

The screenshot shows a coding platform interface for problem 88, "Merge Sorted Array". The left panel contains the problem description, which states: "You are given two integer arrays `nums1` and `nums2`, sorted in **non-decreasing order**, and two integers `m` and `n`, representing the number of elements in `nums1` and `nums2` respectively. Merge `nums1` and `nums2` into a single array sorted in **non-decreasing order**. The final sorted array should not be returned by the function, but instead be stored *inside the array* `nums1`. To accommodate this, `nums1` has a length of `m + n`, where the first `m` elements denote the elements that should be merged, and the last `n` elements are set to `0` and should be ignored. `nums2` has a length of `n`." It includes two examples: Example 1 with input `nums1 = [1,2,3,0,0,0]`, `m = 3`, `nums2 = [2,5,6]`, `n = 3` and output `[1,2,2,3,5,6]`; and Example 2 with input `nums1 = [1]`, `m = 1`, `nums2 = []`, `n = 0` and output `[1]`. The right panel shows the code editor with a C++ solution:

```
1 class Solution {
2 public:
3     void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
```

 Below the code is the test result section, which shows "Accepted" with a runtime of 0 ms. The input fields are filled with the values from Example 1: `nums1 = [1,2,3,0,0,0]`, `m = 3`, `nums2 = [2,5,6]`, and `n = 3`.

Q2. First bad Version

The screenshot shows a coding platform interface for problem 278, "First Bad Version". The left panel contains the problem description, which states: "You are a product manager and currently leading a team to develop a new product. Unfortunately, the latest version of your product fails the quality check. Since each version is developed based on the previous version, all the versions after a bad version are also bad. Suppose you have `n` versions `[1, 2, ..., n]` and you want to find out the first bad one, which causes all the following ones to be bad. You are given an API `bool isBadVersion(version)` which returns whether `version` is bad. Implement a function to find the first bad version. You should minimize the number of calls to the API." It includes an example: Example 1 with input `n = 5`, `bad = 4` and output `4`. The right panel shows the code editor with a C++ solution:

```
1
2
3
4 class Solution {
5 public:
6     int firstBadVersion(int n) {
```

 Below the code is the test result section, which shows "Accepted" with a runtime of 0 ms. The input fields are filled with the values from Example 1: `n = 5` and `bad = 4`. The output field is filled with `4`.

Q3. Kth largest Element in Array

215. Kth Largest Element in an Array

Medium

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

Constraints:

- $1 \leq k \leq \text{nums.length} \leq 10^5$

17.7K | 275 | 381 Online

```
1 class Solution {
2 public:
3     int findKthLargest(std::vector<int>& nums, int k) {
```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

nums =
[3,2,1,5,6,4]

k =
2

Output

5

Expected

Q4. Find Peak Element

162. Find Peak Element

Medium

A peak element is an element that is strictly greater than its neighbors.

Given a 0-indexed integer array `nums`, find a peak element, and return its index. If the array contains multiple peaks, return the index to **any of the peaks**.

You may imagine that `nums[-1] = nums[n] = -∞`. In other words, an element is always considered to be strictly greater than a neighbor that is outside the array.

You must write an algorithm that runs in $O(\log n)$ time.

Example 1:

Input: `nums = [1,2,3,1]`
Output: 2
Explanation: 3 is a peak element and your function should return the index number 2.

Example 2:

Input: `nums = [1,2,1,3,5,6,4]`
Output: 5
Explanation: Your function can return either index number 1 where the value is 2 or index number 5 where the value is 6.

13K | 309 | 237 Online

```
1 class Solution {
2 public:
3     int findPeakElement(vector<int>& nums) {
```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

nums =
[1,2,3,1]

Output

2

Expected

2

Contribute a testcase

Q5. Merge Intervals

Problem List

56. Merge Intervals

Medium

Topics

Companies

Given an array of `intervals` where `intervals[i] = [starti, endi]`, merge all overlapping intervals, and return *an array of the non-overlapping intervals that cover all the intervals in the input*.

Example 1:

Input: `intervals = [[1,3],[2,6],[8,10],[15,18]]`
Output: `[[1,6],[8,10],[15,18]]`
Explanation: Since intervals `[1,3]` and `[2,6]` overlap, merge them into `[1,6]`.

Example 2:

Input: `intervals = [[1,4],[4,5]]`
Output: `[[1,5]]`
Explanation: Intervals `[1,4]` and `[4,5]` are considered overlapping.

Constraints:

- `1 <= intervals.length <= 104`

23K

178

423 Online

Code

```
1 class Solution {
2 public:
3     vector<vector<int>> merge(vector<vector<int>>& intervals) {
```

SavedLn 25, Col 3

Testcase

Test Result

AcceptedRuntime: 0 ms

Case 1Case 2

Input

intervals =

[[1,3],[2,6],[8,10],[15,18]]

Output

[[1,6],[8,10],[15,18]]

Expected

[[1,6],[8,10],[15,18]]

Contribute a testcase