

AP Assignment 5

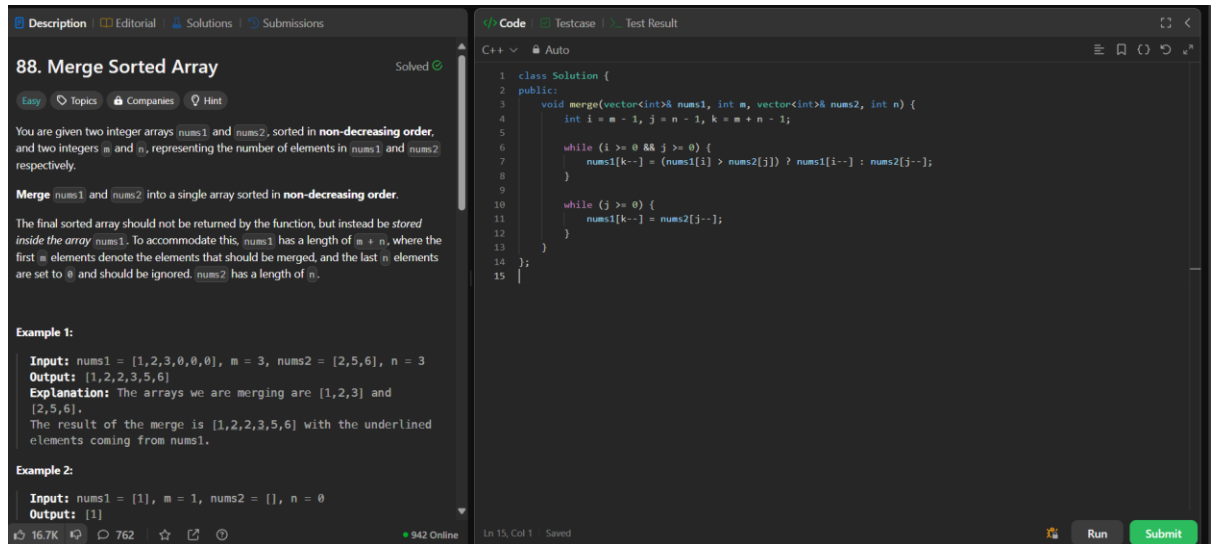
Name : Piyush Rawat

UID : 22BCS10750

Sec_grp – 608-B

Sorting and Searching:

1. Merge Sorted Array: <https://leetcode.com/problems/merge-sorted-array/>



88. Merge Sorted Array Solved

Easy Topics Companies Hint

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`.

Example 1:

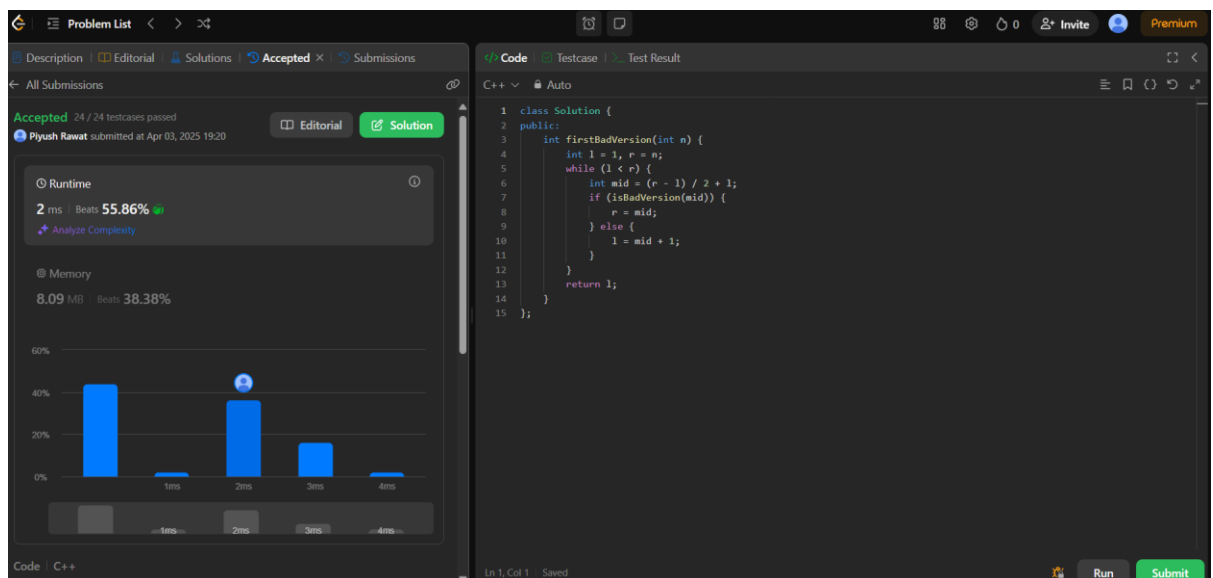
Input: `nums1 = [1,2,3,0,0,0]`, `m = 3`, `nums2 = [2,5,6]`, `n = 3`
Output: `[1,2,2,3,5,6]`
Explanation: The arrays we are merging are `[1,2,3]` and `[2,5,6]`.
The result of the merge is `[1,2,2,3,5,6]` with the underlined elements coming from `nums1`.

Example 2:

Input: `nums1 = [1]`, `m = 1`, `nums2 = []`, `n = 0`
Output: `[1]`

```
1 class Solution {
2 public:
3     void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
4         int i = m - 1, j = n - 1, k = m + n - 1;
5
6         while (i >= 0 && j >= 0) {
7             nums1[k--] = (nums1[i] > nums2[j]) ? nums1[i--] : nums2[j--];
8         }
9
10        while (j >= 0) {
11            nums1[k--] = nums2[j--];
12        }
13    };
14 }
15
```

2. First Bad Version: <https://leetcode.com/problems/first-bad-version/>



278. First Bad Version

Accepted 24 / 24 testcases passed

Piyush Rawat submitted at Apr 03, 2025 19:20

Runtime: 2 ms, Beats 55.86%

Memory: 8.09 MB, Beats 38.38%

Code: C++

```
1 class Solution {
2 public:
3     int firstBadVersion(int n) {
4         int l = 1, r = n;
5         while (l < r) {
6             int mid = (r - l) / 2 + l;
7             if (isBadVersion(mid)) {
8                 r = mid;
9             } else {
10                l = mid + 1;
11            }
12        }
13        return l;
14    };
15 }
```

3. Sort Colors: <https://leetcode.com/problems/sort-colors/>

AP Assignment 5

The screenshot shows a LeetCode submission for the 'Find Peak Element' problem. The left sidebar displays the submission status as 'Accepted' with 89/89 testcases passed, submitted by Piyush Rawat on Apr 03, 2025 at 19:21. The runtime is 0 ms, beating 100.00% of submissions, and the memory usage is 11.74 MB, beating 9.95%. A bar chart shows the runtime distribution. The main area displays the C++ code for the 'Find Peak Element' problem, which uses a linear search approach to find the index of the maximum element in the array.

```
1 class Solution {
2 public:
3     void sortColors(vector<int>& nums) {
4         unordered_map<int, int> colorCount;
5
6         for (int num : nums) {
7             colorCount[num]++;
8         }
9
10        int index = 0;
11        for (int color = 0; color <= 2; ++color) { //runs from 0 to 2
12            while (colorCount[color] > 0) { //jab tak frequency is greater then 0
13                nums[index++] = color; //when we put the number in array[output] index badta rahega
14                colorCount[color]--; //reduce the frequency of the number present in hash table
15            }
16        }
17    }
18 };
```

4. Find Peak Element: <https://leetcode.com/problems/find-peak-element/>

The screenshot shows a LeetCode submission for the 'Median of Two Sorted Arrays' problem. The left sidebar displays the submission status as 'Accepted' with 68/68 testcases passed, submitted by Piyush Rawat on Apr 03, 2025 at 19:21. The runtime is 0 ms, beating 100.00% of submissions, and the memory usage is 12.62 MB, beating 32.44%. A bar chart shows the runtime distribution. The main area displays the C++ code for the 'Median of Two Sorted Arrays' problem, which uses a binary search approach to find the median of the two sorted arrays.

```
1 class Solution {
2 public:
3     int findPeakElement(vector<int>& nums) {
4         int left = 0;
5         int right = nums.size() - 1;
6
7         while (left < right) {
8             int mid = left + (right - left) / 2;
9             if (nums[mid] > nums[mid + 1]) {
10                 right = mid;
11             } else {
12                 left = mid + 1;
13             }
14         }
15         return right; //left
16     }
17 };
```

5. Median of Two Sorted Arrays: <https://leetcode.com/problems/median-of-two-sorted-arrays/>

AP Assignment 5

Problem List

Description

Accepted

Editorial

Solutions

Submissions

All Submissions

Accepted 2096 / 2096 testcases passed

Piyush Rawat submitted at Apr 03, 2025 19:22

Editorial

Solution

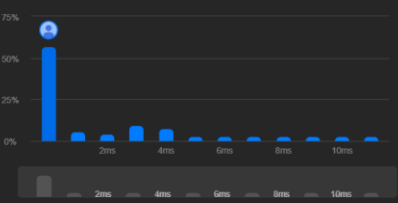
Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

95.10 MB | Beats 63.38%



Code

C++

Code

Testcase

Test Result

C++

Auto

```
1 class Solution {
2 public:
3     double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
4         int n = nums1.size();
5         int m = nums2.size();
6         int i = 0, j = 0, m1 = 0, m2 = 0;
7
8         for (int count = 0; count <= (n + m) / 2; count++) {
9             m2 = m1;
10            if (i != n && j != m) {
11                if (nums1[i] > nums2[j]) {
12                    m1 = nums2[j++];
13                } else {
14                    m1 = nums1[i++];
15                }
16            } else if (i < n) {
17                m1 = nums1[i++];
18            } else {
19                m1 = nums2[j++];
20            }
21        }
22
23        if ((n + m) % 2 == 1) {
24            return static_cast<double>(m1);
25        } else {
26            double ans = static_cast<double>(m1) + static_cast<double>(m2);
27            return ans / 2.0;
28        }
29    }
30};
```

Ln 1, Col 1 Saved

Run

Submit