

ASSIGNMENT - 5

NAME: ARCHI BANSAL

SECTION: 608/B

UID: 22BCS15264

SUBJECT: AP LAB

1. Merge Sorted Array

Accepted 59 / 59 testcases passed
archibansal submitted at Mar 05, 2025 23:09

Runtime 0 ms | Beats 100.00%
Memory 12.38 MB | Beats 39.28%

Code C++

```
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
        int i = m - 1; // Pointer for nums1 (actual elements)
        int j = n - 1; // Pointer for nums2
        int k = m + n - 1; // Pointer for placing elements in nums1

        // Merge from the end to avoid overwriting elements in nums1
    }
};
```

Testcase Test Result

Case 1 Case 2 Case 3 +

nums1 =
[1,2,3,0,0,0]

m =

n =

2. First Bad Version

Accepted 24 / 24 testcases passed
archibansal submitted at Apr 03, 2025 20:54

Runtime 2 ms | Beats 55.86%
Memory 7.86 MB | Beats 69.64%

Code C++

```
class Solution {
public:
    int firstBadVersion(int n) {
        int left = 1, right = n;
        while (left < right) { // Binary search loop
            int mid = left + (right - left) / 2; // Avoid overflow
            if (isBadVersion(mid)) {
                right = mid; // Move left to find the first bad version
            } else {
                left = mid + 1;
            }
        }
        return left; // left == right, the first bad version
    }
};
```

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

n =
5

ASSIGNMENT - 5

3. Sort Colors

The screenshot shows a C++ solution for the 'Sort Colors' problem. The interface includes a problem list, a description, and a solution editor. The solution is accepted, with a runtime of 0 ms and memory usage of 11.57 MB. A bar chart shows the runtime performance, indicating that 0.63% of solutions used 4 ms of runtime. The code implements a sorting algorithm using a while loop and conditional swaps.

Runtime: 0 ms | Beats: 100.00% | Memory: 11.57 MB | Beats: 75.11%

Code:

```
#include <vector>
using namespace std;

class Solution {
public:
    void sortColors(vector<int>& nums) {
        int low = 0, mid = 0, high = nums.size() - 1;

        while (mid <= high) {
            if (nums[mid] == 0) {
                swap(nums[low], nums[mid]);
                low++;
                mid++;
            } else if (nums[mid] == 1) {
                mid++;
            } else { // nums[mid] == 2
                swap(nums[mid], nums[high]);
                high--;
            }
        }
    }
};
```

Testcase: Accepted | Runtime: 0 ms

Case 1: Input: nums = [2,0,2,1,1,0]

4. Find Peak Element

The screenshot shows a C++ solution for the 'Find Peak Element' problem. The interface includes a problem list, a description, and a solution editor. The solution is accepted, with a runtime of 0 ms and memory usage of 12.50 MB. A bar chart shows the runtime performance, indicating that 0.43% of solutions used 2 ms of runtime. The code implements a binary search algorithm to find the peak element.

Runtime: 0 ms | Beats: 100.00% | Memory: 12.50 MB | Beats: 67.85%

Code:

```
#include <vector>
using namespace std;

class Solution {
public:
    int findPeakElement(vector<int>& nums) {
        int low = 0, high = nums.size() - 1;

        while (low < high) {
            int mid = low + (high - low) / 2; // Prevents overflow
            if (nums[mid] > nums[mid + 1]) {
                high = mid; // Peak is in the left half
            } else {
                low = mid + 1; // Peak is in the right half
            }
        }
        return low; // Peak index
    }
};
```

Testcase: Accepted | Runtime: 0 ms

Case 1: Input: nums = [1,2,3,1]

ASSIGNMENT - 5

5. Median of Two Sorted Arrays

The screenshot displays the LeetCode submission interface for the problem "Median of Two Sorted Arrays". The submission is marked as "Accepted" and shows a runtime of 1 ms, beating 42.81% of other submissions. The memory usage is 95.13 MB, beating 63.38% of other submissions. A bar chart shows the runtime distribution across different time limits. The C++ code is displayed on the right, showing a recursive solution for finding the median of two sorted arrays.

Runtime Performance:

- Runtime: 1 ms | Beats: 42.81%
- Memory: 95.13 MB | Beats: 63.38%

Code (C++):

```
1 #include <vector>
2 #include <limits>
3 using namespace std;
4
5 class Solution {
6 public:
7     double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
8         if (nums1.size() > nums2.size()) {
9             return findMedianSortedArrays(nums2, nums1); // Ensure nums1 is smaller
10        }
11
12        int n = nums1.size(), n2 = nums2.size();
13        int left = 0, right = n, half = (n + n2 + 1) / 2;
14
15        while (left <= right) {
16            int mid1 = (left + right) / 2;
17            int mid2 = half - mid1;
18
19            int L1 = (mid1 == 0) ? INT_MIN : nums1[mid1 - 1];
20            int R1 = (mid1 == n) ? INT_MAX : nums1[mid1];
21            int L2 = (mid2 == 0) ? INT_MIN : nums2[mid2 - 1];
22            int R2 = (mid2 == n2) ? INT_MAX : nums2[mid2];
23
24            if (L1 <= R2 && L2 <= R1) {
25                if ((n + n2) % 2 == 0) {
26                    return (max(L1, L2) + min(R1, R2)) / 2.0;
27                } else {
28                    return max(L1, L2);
29                }
30            } else if (L1 > R2) {
31                right = mid1 - 1;
32            } else {
33                left = mid1 + 1;
34            }
35        }
36    }
37};
```