

Assignment NO. 5

Name: Sahil

UID: 22BCS15237

Section: 608(B)

Subject: AP-2

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

```
1 class Solution {
2 public:
3     void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
4         int i = m - 1;
5         int j = n - 1;
6         int k = m + n - 1;
7
8         while (i >= 0 && j >= 0) {
9             if (nums1[i] > nums2[j]) {
10                 nums1[k--] = nums1[i--];
11             } else {
12                 nums1[k--] = nums2[j--];
13             }
14         }
15
16         while (j >= 0) {
17             nums1[k--] = nums2[j--];
18         }
19     }
20 };
```

Saved Ln 21, Col 1

Testcase

Test Result

Accepted Runtime: 0 ms

• Case 1

• Case 2

• Case 3

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

```
C++ v Auto
1 class Solution {
2 public:
3     int firstBadVersion(int n) {
4         int left = 1, right = n;
5         while (left < right) {
6             int mid = left + (right - left) / 2;
7             if (isBadVersion(mid)) {
8                 right = mid;
9             } else {
10                left = mid + 1;
11            }
12        }
13        return left;
14    }
15 };
16
```

Saved Ln 16, Col 1

Testcase Test Result

Accepted Runtime: 3 ms

• Case 1 • Case 2

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



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

Saved Ln 17, Col 1

☒ Testcase > Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

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

C++

Auto

```
1 class Solution {
2 public:
3     int findPeakElement(vector<int>& nums) {
4         int left = 0, right = nums.size() - 1;
5
6         while (left < right) {
7             int mid = left + (right - left) / 2;
8             if (nums[mid] > nums[mid + 1]) {
9                 right = mid;
10            } else {
11                left = mid + 1;
12            }
13        }
14
15        return left;
16    }
17 };
18
```

SavedLn 18, Col 1

☒ Testcase

>_ Test Result

AcceptedRuntime: 0 ms

• Case 1

• Case 2

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

</>Code

C++  Auto

```
1 class Solution {
2 public:
3     double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
4         int m = nums1.size(), n = nums2.size();
5         if (m > n) return findMedianSortedArrays(nums2, nums1);
6
7         int left = 0, right = m, total = (m + n + 1) / 2;
8
9         while (left <= right) {
10             int mid1 = left + (right - left) / 2;
11             int mid2 = total - mid1;
12
13             int l1 = (mid1 > 0) ? nums1[mid1 - 1] : INT_MIN;
14             int l2 = (mid2 > 0) ? nums2[mid2 - 1] : INT_MIN;
15             int r1 = (mid1 < m) ? nums1[mid1] : INT_MAX;
16             int r2 = (mid2 < n) ? nums2[mid2] : INT_MAX;
17
18             if (l1 <= r2 && l2 <= r1) {
19                 if ((m + n) % 2 == 0) {
20                     return (max(l1, l2) + min(r1, r2)) / 2.0;
21                 } else {
22                     return max(l1, l2);
23                 }
24             } else if (l1 > r2) {
25                 right = mid1 - 1;
26             } else {
27                 left = mid1 + 1;
28             }
29         }
30
31         return 0.0;
32     }
33 };
34
```