

Q.1. Sort Colors.

The screenshot shows the LeetCode interface for the 'Sort Colors' problem. The problem description is on the left, and the solution code is on the right. The solution is in Java and uses a counting sort approach. The runtime is 1 ms, and the memory is 42.19 MB. The test results show that the solution is accepted for all test cases.

```
class Solution {
    public void sortColors(int[] nums) {
        HashMap<Integer, Integer> count = new HashMap<>();
        count.put(0, 0);
        count.put(1, 0);
        count.put(2, 0);

        for (int num : nums) {
            count.put(num, count.get(num) + 1);
        }

        int idx = 0;
        for (int color = 0; color < 3; color++) {
            for (int freq = count.get(color); freq > 0; freq--) {
                nums[idx++] = color;
            }
        }
    }
}
```

Test Results:

Case	Input	Output	Expected
Case 1	[2,0,2,1,1,0]	[0,0,1,1,2,2]	[0,0,1,1,2,2]

Q.2. Merge Sorted Array.

The screenshot shows the LeetCode interface for the 'Merge Sorted Array' problem. The problem description is on the left, and the solution code is on the right. The solution is in Java and uses a two-pointer approach. The runtime is 0 ms, and the memory is 42.32 MB. The test results show that the solution is accepted for all test cases.

```
class Solution {
    public void merge(int[] nums1, int m, int[] nums2, int n) {
        int p1 = m-1;
        int p2 = n-1;
        int pmerge = m+n-1;

        while (p2 >= 0) {
            if (p1 >= 0 && nums1[p1] > nums2[p2]) {
                nums1[pmerge--] = nums1[p1--];
            } else {
                nums1[pmerge--] = nums2[p2--];
            }
        }
    }
}
```

Test Results:

Case	Input	Output
Case 1	[1]	[1]

Q.3. Binary Search.

The screenshot shows the LeetCode interface for the 'Binary Search' problem. The problem description is on the left, and the solution code is on the right. The solution is in Java and uses a binary search approach. The runtime is 0 ms, and the memory is 45.90 MB. The test results show that the solution is accepted for all test cases.

```
class Solution {
    public int search(int[] nums, int target) {
        int start = 0;
        int end = nums.length-1;

        while (start <= end) {
            int mid = start + (end-start)/2;
            if (nums[mid] == target) {
                return mid;
            } else if (nums[mid] < target) {
                start = mid + 1;
            } else {
                end = mid - 1;
            }
        }

        return -1;
    }
}
```

Test Results:

Case	Input	Output
Case 1	[-1,0,3,5,9,12]	4