

LeetCode Solution

The image displays two screenshots of the LeetCode website, showing solutions for two different problems. The top screenshot is for the problem "Median of Two Sorted Arrays" (Problem 4), and the bottom screenshot is for "How Many Numbers Are Smaller Than the Current Number" (Problem 912).

Top Screenshot: Median of Two Sorted Arrays

The problem page shows the submission history table with the following data:

Time Submitted	Status	Runtime	Memory	Language
07/12/2022 22:13	Accepted	5 ms	49.7 MB	java

The solution code is in Java, using a binary search approach to find the median of two sorted arrays. The code is as follows:

```
1 class Solution {
2     public double findMedianSortedArrays(int[] nums1, int[] nums2) {
3         int m = nums1.length, n = nums2.length;
4
5         if (m > n) {
6             int[] temp = nums1; nums1 = nums2; nums2 = temp;
7             int tmp = m; m = n; n = tmp;
8         }
9
10        int i_min = 0, i_max = m, half_len = (m + n + 1) / 2;
11
12        while (i_min <= i_max) {
13            int i = (i_min + i_max) / 2, j = half_len - i;
14
15            if (i < i_max && nums1[i] < nums2[j - 1]) {
16                i_min = i + 1;
17            }
18            else if (i > i_min && nums2[j] < nums1[i - 1]) {
19                i_max = i - 1;
20            }
21            else {
22                int maxleft = 0;
23
24                if (i == 0) {
25                    maxleft = nums2[j - 1];
26                }
27                else if (j == 0) {
28                    maxleft = nums1[i - 1];
29                }
30                else {
31                    maxleft = Math.max(nums1[i - 1], nums2[j - 1]);
32                }
33
34                if ((m + n) % 2 == 1) {
35                    return maxleft;
36                }
37
38                int minright = 0;
39
40                if (i == m) {
41                    minright = nums2[j];
42                }
43                else if (j == n) {
44                    minright = nums1[i];
45                }
46                else {
47                    minright = Math.min(nums1[i], nums2[j]);
48                }
49                return (maxleft + minright) / 2.0;
50            }
51        }
52    }
53 }
```

Bottom Screenshot: How Many Numbers Are Smaller Than the Current Number

The problem page shows the submission history table with the following data:

Time Submitted	Status	Runtime	Memory	Language
07/12/2022 13:17	Accepted	1 ms	44.1 MB	java

The solution code is in Java, using a nested loop to compare each element in the input array with all other elements to count how many are smaller. The code is as follows:

```
1 class Solution {
2     public int[] smallerNumbersThanCurrent(int[] nums) {
3         int[] counter = new int[nums.length];
4         for (int i : nums) {
5             counter[i]++;
6         }
7         int sum = 0;
8         int[] restSum = new int[nums.length];
9         for (int i = 0; i < 100; i++) {
10             restSum[i] = sum;
11             sum += counter[i];
12         }
13         int[] ans = new int[nums.length];
14         for (int i = 0; i < nums.length; i++) {
15             ans[i] = restSum[nums[i]];
16         }
17         return ans;
18     }
19 }
```