AlgoExpert   Quad Layout   00:00:00 | 00:00:00

Prompt   Scratchpad   Our Solution(s)   Video Explanation

Difficulty: 🟦 **Category:** Sorting **Successful Submissions:** 10,996+

Three Number Sort

You're given an array of integers and another array of three distinct integers. The first array is guaranteed to only contain integers that are in the second array, and the second array represents a desired order for the integers in the first array. For example, a second array of `[x, y, z]` represents a desired order of `[x, x, ..., x, y, y, ..., y, z, z, ..., z]` in the first array.

Write a function that sorts the first array according to the desired order in the second array.

The function should perform this in place (i.e., it should mutate the input array), and it shouldn't use any auxiliary space (i.e., it should run with constant space: `O(1)` space).

Note that the desired order won't necessarily be ascending or descending and that the first array won't necessarily contain all three integers found in the second array—it might only contain one or two.

Sample Input

```
array = [1, 0, 0, -1, -1, 0, 1, 1]
order = [0, 1, -1]
```

Sample Output

```
[0, 0, 0, 1, 1, 1, -1, -1]
```

Hints

Hint 1

What advantage does knowing the three values contained in the array give you, and how can you use that to solve this problem in linear time?

Hint 2

Try counting how many times each of the three values appears in the input array. Once you have these counts, you can repopulate the input array as need be.

Hint 3

Putting aside the first two hints, try conceptually splitting the original array into three subarrays and moving elements of each unique value into the correct subarray. You'll need to keep track of the respective starting indices of these subarrays.

Hint 4

Going off of Hint #3, you can solve this problem either with two passes through the input array or with a single pass. If you do two passes through the array, you'll specifically be positioning the first ordered element during the first pass and the third ordered element during the second pass. You'll be swapping elements from the left side of the array whenever you encounter the first element, and you'll be swapping elements from the right side of the array whenever you encounter the third element. You'll have to keep track of where you last placed a first element or a third element. With a single pass through the array, you'll have to implement both of these strategies and a little more all at once.

Optimal Space & Time Complexity

O(n) time | O(1) space - where n is the length of the array