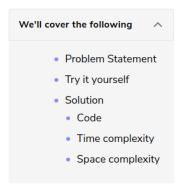




# **Dutch National Flag Problem (medium)**



### Problem Statement #

Given an array containing 0s, 1s and 2s, sort the array in-place. You should treat numbers of the array as objects, hence, we can't count 0s, 1s, and 2s to recreate the array.

The flag of the Netherlands consists of three colors: red, white and blue; and since our input array also consists of three different numbers that is why it is called <u>Dutch National Flag problem</u>.

### Example 1:

```
Input: [1, 0, 2, 1, 0]
Output: [0 0 1 1 2]
```

#### Example 2:

```
Input: [2, 2, 0, 1, 2, 0]
Output: [0 0 1 2 2 2 ]
```

# Try it yourself #

Try solving this question here:

### Solution #

The brute force solution will be to use an in-place sorting algorithm like Heapsort which will take O(N\*logN). Can we do better than this? Is it possible to sort the array in one iteration?

We can use a **Two Pointers** approach while iterating through the array. Let's say the two pointers are called **low** and **high** which are pointing to the first and the last element of the array respectively. So while iterating, we will move all 0s before **low** and all 2s after **high** so that in the end, all 1s will be between **low** and **high**.

### Code

Here is what our algorithm will look like:

## Time complexity

The time complexity of the above algorithm will be O(N) as we are iterating the input array only once.

## Space complexity #

The algorithm runs in constant space O(1).

