

Dutch National Flag Problem (medium)

We'll cover the following ^

- Problem Statement
- Try it yourself
- Solution
 - Code
 - Time complexity
 - Space complexity

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 [Dutch National Flag problem](#).

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:

 Java

 Python3

 JS

 C++

```
1 class DutchFlag {
2
3     public static void sort(int[] arr) {
4         // TODO: Write your code here
5     }
6 }
7
```

Test

Save

Reset




Solution


The brute force solution will be to use an in-place sorting algorithm like [Heapsort](#) which will take $O(N * \log N)$. 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:

 Java

 Python3

 C++

 JS

```
1 class DutchFlag {
2
3     public static void sort(int[] arr) {
4         // all elements < low are 0 and all elements > high are 2
5         // all elements from >= low < i are 1
6         int low = 0, high = arr.length - 1;
7         for (int i = 0; i <= high; i++) {
8             if (arr[i] == 0) {
9                 swap(arr, i, low);
```

```
10      // increment 'i' and 'low'
11      i++;
12      low++;
13  } else if (arr[i] == 1) {
14      i++;
15  } else { // the case for arr[i] == 2
16      swap(arr, i, high);
17      // decrement 'high' only, after the swap the number at index 'i' could be 0, 1 or 2
18      high--;
19  }
20  }
21  }
22
23  private static void swap(int[] arr, int i, int j) {
24      int temp = arr[i];
25      arr[i] = arr[j];
26      arr[j] = temp;
27  }
28  }
```

Run

SaveReset

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 $O(1)$.

← Back

Next →

Subarrays with Product Less than a Ta...

Problem Challenge 1

✓ Completed

! Report an Issue ? Ask a Question