

 Master algorithms together on [Binary Search](#)! Create a room, invite your friends, and race to finish the problems.



Daily Coding Problem

[Blog](#)

Daily Coding Problem #35

Problem

This problem was asked by Google.

Given an array of strictly the characters 'R', 'G', and 'B', segregate the values of the array so that all the Rs come first, the Gs come second, and the Bs come last. You can only swap elements of the array.

Do this in linear time and in-place.

For example, given the array ['G', 'B', 'R', 'R', 'B', 'R', 'G'], it should become ['R', 'R', 'R', 'G', 'G', 'B', 'B'].

Solution

It may be easier to first consider an easier problem: one with only two possible values, say 'R' and 'G'. Then we could maintain the following loop invariant quite easily:

- Maintain three sections of the array using two indices, `low` and `high`:
 - Strictly 'R's: `array[:low]`
 - Unknown: `array[low:high]`
 - Strictly 'G's: `array[high:]`

Initially, `low` will be 0 and `high` will be `len(array) - 1`, since the whole array is unknown. As we iterate over the array, we'll swap any 'G's we see to the third section and decrement `high`. If we see an 'R', then we just need to increment `low`, since that's where it belongs. We can terminate once `low` crosses `high`. So we can gradually shrink our unknown section through the following algorithm:

```
def partition(arr):  
    low, high = 0, len(arr) - 1  
    while low <= high:  
        if arr[low] == 'R':  
            low += 1  
        else:  
            arr[low], arr[high] = arr[high], arr[low]  
            high -= 1
```

This correctly partitions our array into two separate categories. How can we extend this to three partitions? Let's maintain four sections using 3 indices, `low`, `mid`, and `high`:

- Strictly 'R's: `array[:low]`
- Strictly 'G's: `array[low:mid]`
- Unknown: `array[mid:high]`
- Strictly 'B's: `array[high:]`

We'll initialize low and mid both to 0, and high to $\text{len}(\text{array}) - 1$ so that our unknown section is the whole array, as before. To maintain this invariant, we should do the following:

- Look at `array[mid]`:
 - If it's R, then swap `array[low]` with `array[mid]` and increment low and mid
 - If it's G, then just increment mid; it's where it should be
 - If it's B, then swap `array[mid]` with `array[high]` and decrement high

Once mid crosses over with high, then our unknown section is gone and we can terminate.

Our solution looks like this:

```
def partition(arr):
    low, mid, high = 0, 0, len(arr) - 1
    while mid <= high:
        if arr[mid] == 'R':
            arr[low], arr[mid] = arr[mid], arr[low]
            low += 1
            mid += 1
        elif arr[mid] == 'G':
            mid += 1
        else:
            arr[mid], arr[high] = arr[high], arr[mid]
            high -= 1
```

P.S. This problem is also called the [Dutch national flag problem](#)!

[© Daily Coding Problem 2019](#) [Privacy Policy](#) [Terms of Service](#) [Press](#)