LeetCode 75

Description

Given an array with n objects colored red, white or blue, sort them so that objects of the same color are adjacent, with the colors in the order red, white and blue.

Here, we will use the integers 0, 1, and 2 to represent the color red, white, and blue respectively.

Note:
You are not suppose to use the library's sort function for this problem.

Thought

One straight forward solution is the bucket sort, iterate through the array and save the count of each color in a hash map, then overwrite array with the total number of each color; however this solution will need extra space.

A better way is use three pointers, one in the start, one in the end, and one as current. When we are traversing the array, we swap it with start whenever current is 0, and we swap it with end whenever current is 2. One little trick here is that, when current swap with start, both++; while when current swap with end, end--, but current stay the same for one more check. Because, current has already checked ele in [start, current], but have not checked ele in [current, end]. Time complexity is O(n), space complexity is O(1).

Solution

```
class Solution {
  public void sortColors(int[] nums){
    if (nums == null || nums.length  == 0) {
      return;
    }

    int len = nums.length;
    int f = 0;
    int c = 0;
    int e = len - 1;

    while (c <= e){
      if (nums[c] == 0) {
        swap(nums, f, c);
        f++;
```

```
          c++;
        } else if(nums[c] == 2) {
          swap(nums, e, c);
          e--;
        } else {
          c++;
        }
      }
    }

    private void swap(int[] nums, int a, int b){
      int tmp = nums[a];
      nums[a] = nums[b];
      nums[b] = tmp;
    }
  }
```

Takeaways

- Bucket sort for integer sorting;
- Loop invariant when using pointers;