Documentation for "Sort Colors" Algorithm

Problem Statement

The task is to sort an array nums containing n objects colored red, white, or blue represented by integers 0, 1, and 2 respectively. The goal is to sort the array in-place so that objects of the same color are adjacent, with the colors arranged in the order red (0), white (1), and blue (2).

Constraints:

- The length of the array n is between 1 and 300 inclusive.
- Each element in nums is either 0, 1, or 2.
- The sorting must be done without using the library's sort function.
- The solution should ideally be a one-pass algorithm using only constant extra space.

Algorithm

The algorithm used here is known as the Dutch National Flag problem solution by Edsger W. Dijkstra. It involves using three pointers to sort the array in a single pass with constant extra space.

Detailed Steps

1. **Initialize Pointers:**

- low points to the beginning of the array and is used to place 0s (reds).
- high points to the end of the array and is used to place 2s (blues).
- current is used to traverse the array.

2. **Traverse the Array:**

- As long as current is less than or equal to high, perform the following checks:
- If nums[current] is 0, swap it with the element at the low pointer, increment both low and current.
- If nums[current] is 2, swap it with the element at the high pointer and decrement high. Note that current is not incremented in this case because the element swapped from high needs to be checked.
- If nums[current] is 1, simply move to the next element by incrementing current.

Example Usage

Example 1

nums = [2,0,2,1,1,0]

solution = Solution()

solution.sortColors(nums)

print(nums) # Output: [0, 0, 1, 1, 2, 2]

Example 2

nums = [2,0,1]

solution = Solution()

solution.sortColors(nums)

print(nums) # Output: [0, 1, 2]

Explanation of Time and Space Complexity

- **Time Complexity:** O(n), where n is the number of elements in the array. Each element is checked at most once.

- **Space Complexity:** O(1), as the sorting is done in-place with a constant amount of extra space used for the pointers.

This solution ensures the array is sorted in a single pass while maintaining constant extra space, meeting the problem's constraints and the follow-up requirement.