Experiment 4 A

Student Name: PARDEEP SINGH **UID: 22BCS16692**

Branch: CSE **Section/Group: Ntpp 602-A**

Semester: 6TH **Date of Performance:13/02/25**

Subject Name: AP Lab-2 **Subject Code: 22CSH-352**

1. TITLE:

Sort Colors

2. AIM:

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

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

You must solve this problem without using the library's sort function.

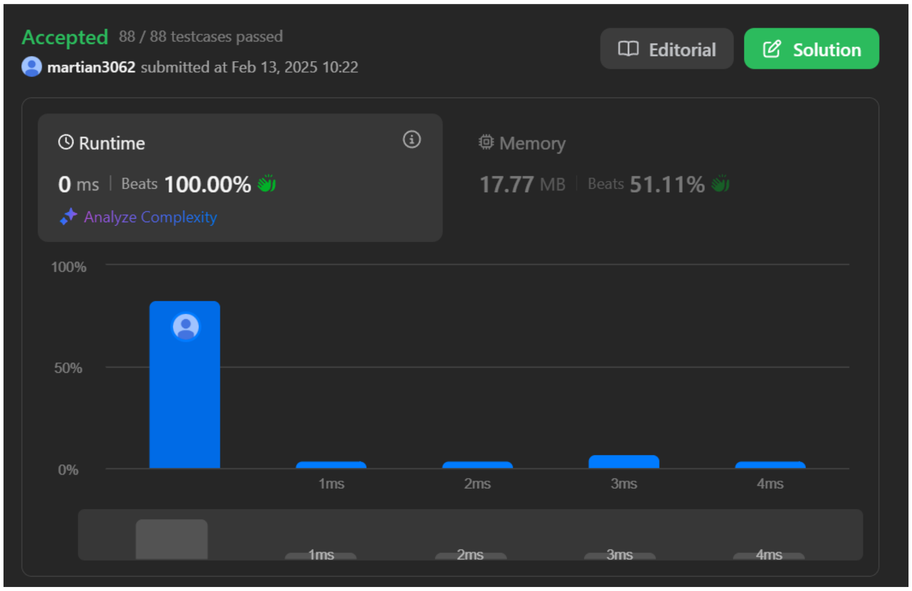
3. Algorithm

- Initialize counters for zeros and ones.
- Traverse the list and count the number of zeros and ones.
- Overwrite the original list: first with zeros, then ones, and finally twos (the remainder).

Implemetation/Code

```
class Solution:
def sortColors(self, nums: List[int]) -> None:
    zeros, ones, n = 0, 0, len(nums)
    for num in nums:
        if num == 0:
            zeros += 1
        elif num == 1:
            ones += 1
    for i in range(0, zeros):
        nums[i] = 0
    for i in range(zeros, zeros + ones):
        nums[i] = 1
    for i in range(zeros + ones, n):
        nums[i] = 2
```

Output



Accepted 88 / 88 testcases passed

martian3062 submitted at Feb 13, 2025 10:22

Runtime

0 ms | Beats 100.00%

Memory

17.77 MB | Beats 51.11%

Time Complexity : O(n)

Space Complexity : O(1)

Learning Outcomes:-

- Learn the principles behind counting sort.
- Manipulate array indices and values to sort in-place

Experiment 4 B

Student Name: PARDEEP SINGH	**UID: 22BCS16692**
Branch: CSE	**Section/Group: Ntpp 602-A**
Semester: 6TH	**Date of Performance:13/02/25**
Subject Name: AP Lab-2	**Subject Code: 22CSH-352**

1. TITLE:

Search in Rotated Sorted Array

AIM:

There is an integer array nums sorted in ascending order (with distinct values).

Prior to being passed to your function, nums is possibly rotated at an unknown pivot index k (1 <= k < nums.length) such that the resulting array is [nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]] (0-indexed). For example, [0,1,2,4,5,6,7] might be rotated at pivot index 3 and become [4,5,6,7,0,1,2].

2. Algorithm

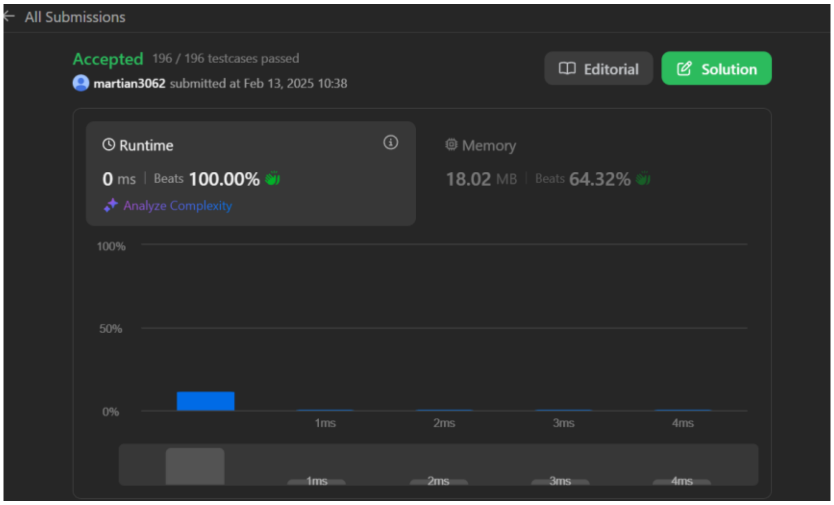
- Initialize two pointers, left at the start and right at the end of the list.
- Use a loop to repeatedly divide the list into halves.
- If the middle element matches the target, return its index.
- Adjust the left or right pointer based on the comparison between the target and the middle element,
- If the target is not found by the end of the loop, return -1.

Implemetation/Code:

```
class Solution:
    def search(self, nums: List[int], target: int) -> int:
        left, right = 0, len(nums) - 1
        while left <= right:
            mid = (left + right) // 2
            if nums[mid] == target:
                return mid
            if nums[left] <= nums[mid]:
```

```
        if nums[left] <= target < nums[mid]:
            right = mid - 1
        else:
            left = mid + 1
    else:
        if nums[mid] < target <= nums[right]:
            left = mid + 1
        else:
            right = mid - 1
return -1
```

Output



Accepted 196 / 196 testcases passed

martian3062 submitted at Feb 13, 2025 10:38

0 ms | Beats 100.00%

18.02 MB | Beats 64.32%

Time Complexity : O(log n)

Space Complexity : O(1)

Learning Outcomes:-

- Learn how to implement and utilize binary search in a potentially rotated sorted array
- Optimizing search operations significantly over linear scanning.
- Managing multiple conditions to direct search logic