Experiment 4 A

Student Name: Karanvir Singh **UID: 22BCS16269**

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

- Set up counters for ones and zeros.
- Count how many ones and zeros there are in the list.
- Replace the original list with zeros, ones, and then twos (the remainder) in order of precedence.

Implemetation/Code

```
class Solution {
public:
void sortColors(vector<int>& nums) {
int zero = -1;
int one = -1;
int two = -1;

for (const int num : nums)
if (num == 0) {
nums[++two] = 2;
```

```
nums[++one] = 1;
nums[++zero] = 0;
} else if (num == 1) {
nums[++two] = 2;
nums[++one] = 1;
} else {
nums[++two] = 2;
}
}
};
```

Output

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

DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Experiment 4 B

Student Name: Karanvir Singh **UID: 22BCS16269**

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

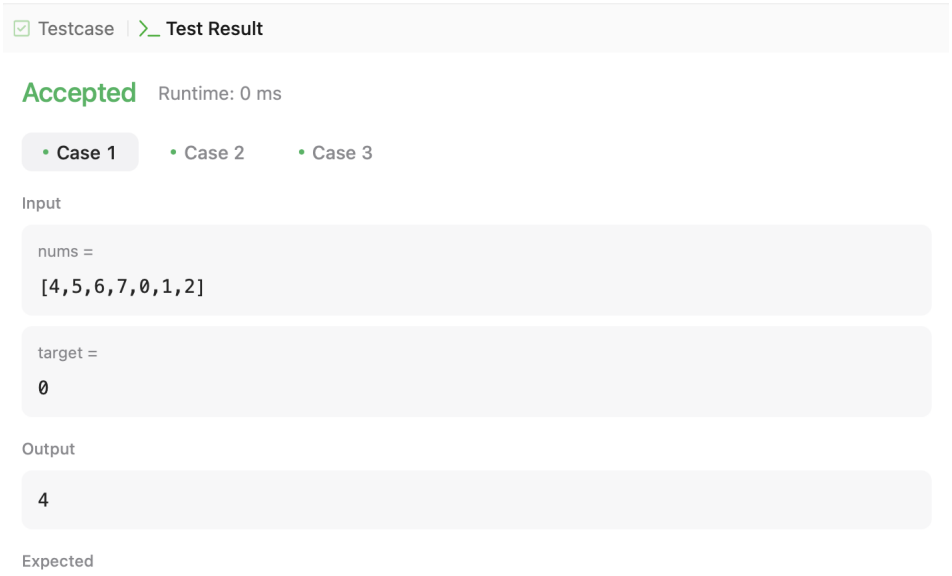
- Initialize two pointers, left at the start and right at the end of the list.
- Use a loop to repeatedly divide the list into halves.
- If the middle element matches the target, return its index.
- Adjust the left or right pointer based on the comparison between the target and the middle element,
-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

Implemetation/Code:

```
class Solution {
        public:
        int search(vector<int>& nums, int target) {
        int n = nums.size();
        int left = 0, right = n - 1;
        while (left < right) {
        int mid = (left + right) >> 1;
        if (nums[0] <= nums[mid]) {
        if (nums[0] <= target && target <= nums[mid])
        right = mid;
        else
```

```
left = mid + 1;
} else {
if (nums[mid] < target && target <= nums[n - 1])
left = mid + 1;
else
right = mid;
}
}
return nums[left] == target ? left : -1;
}
};
```

Output



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