Project Write-Up: Remove Duplicates from Sorted Array

Author(s): Shalom

Date: 12/19/2024

# Table of Contents

1. Executive Summary  
2. Introduction  
3. System Design and Architecture  
4. Implementation Details  
5. Testing and Validation  
6. Usage Instructions  
7. Challenges and Solutions  
8. Future Improvements  
9. Conclusion  
10. Appendix (if needed)

# 1. Executive Summary

This project addresses the problem of removing duplicate elements from a sorted array in-place while preserving the relative order of unique elements. The solution returns the count of unique elements and modifies the array so that the first `k` elements contain unique values. The solution runs in O(n) time complexity with O(1) space complexity.

# 2. Introduction

Objective: To implement a function that removes duplicates from a sorted array in-place, maintaining the order of unique elements.

Background: The problem focuses on in-place operations to minimize space usage while ensuring correctness of the result.

Scope: The function returns the count of unique elements and modifies the input array such that the first `k` elements contain unique values.

# 3. System Design and Architecture

High-Level Design: The solution iterates through the sorted array, comparing consecutive elements, and shifts unique elements to the front.

Component Overview:  
- `nums`: An array of integers sorted in non-decreasing order.  
- `count`: The counter tracking the number of unique elements.  
- `k`: The final number of unique elements.

Design Decisions:  
- In-place modification avoids using extra memory, ensuring space complexity of O(1).

# 4. Implementation Details

Code Structure:  
The method `removeDuplicates(int[] nums)` iterates through the array, comparing each element with the previous one, and shifting non-duplicate elements to the front.

Algorithm:  
- Step 1: Initialize `count` to 1 (the first element is always unique).  
- Step 2: Iterate through the array from the second element.  
- Step 3: If an element is not equal to the previous element, shift it to the front and increment `count`.  
- Step 4: Return `count` as the number of unique elements.

Code Snippet:

class Solution {  
 public int removeDuplicates(int[] nums) {  
 int count = 1;  
 for(int i = 1; i < nums.length; i++) {  
 if(nums[count-1] != nums[i]) {  
 nums[count] = nums[i];  
 count++;  
 }  
 }  
 return count;  
 }  
}

# 5. Testing and Validation

Testing Approach:  
The function is tested with various test cases, including arrays of different lengths, to ensure it handles all scenarios.

Test Cases:  
1. Input: `nums = [1, 1, 2]`  
 Output: `2, nums = [1, 2, \_]`  
2. Input: `nums = [0, 0, 1, 1, 1, 2, 2, 3, 3, 4]`  
 Output: `5, nums = [0, 1, 2, 3, 4, \_, \_, \_, \_, \_]`  
3. Input: `nums = [1, 1, 1]`  
 Output: `1, nums = [1, \_, \_]`

Known Issues: No known issues with the current implementation.

# 6. Usage Instructions

Installation: This function can be added to any Java project. No installation is required.

Running the Project: To use the `removeDuplicates` function, call it with an array of integers:

int[] nums = {1, 1, 2};  
Solution solution = new Solution();  
int k = solution.removeDuplicates(nums);  
System.out.println(k); // Output: 2

Example Output:  
2

# 7. Challenges and Solutions

Challenges:  
The main challenge is ensuring that duplicates are removed while maintaining the relative order of elements.

Solutions:  
The solution uses a single iteration through the array and modifies the array in-place, ensuring that the unique elements are retained in order.

# 8. Future Improvements

Optimizations: The current solution is optimal with O(n) time complexity and O(1) space complexity.

Known Issues: None. The implementation handles edge cases such as empty arrays and arrays with all identical elements.

# 9. Conclusion

The solution efficiently removes duplicates from a sorted array in-place while maintaining the relative order of unique elements. The implementation meets the problem's constraints and performs optimally.

# 10. Appendix

References: LeetCode Problem 26: Remove Duplicates from Sorted Array (https://leetcode.com/problems/remove-duplicates-from-sorted-array/)