AXSOS ACADEMY

Problem-Solving Patterns

Two Pointers

Remove Duplicates

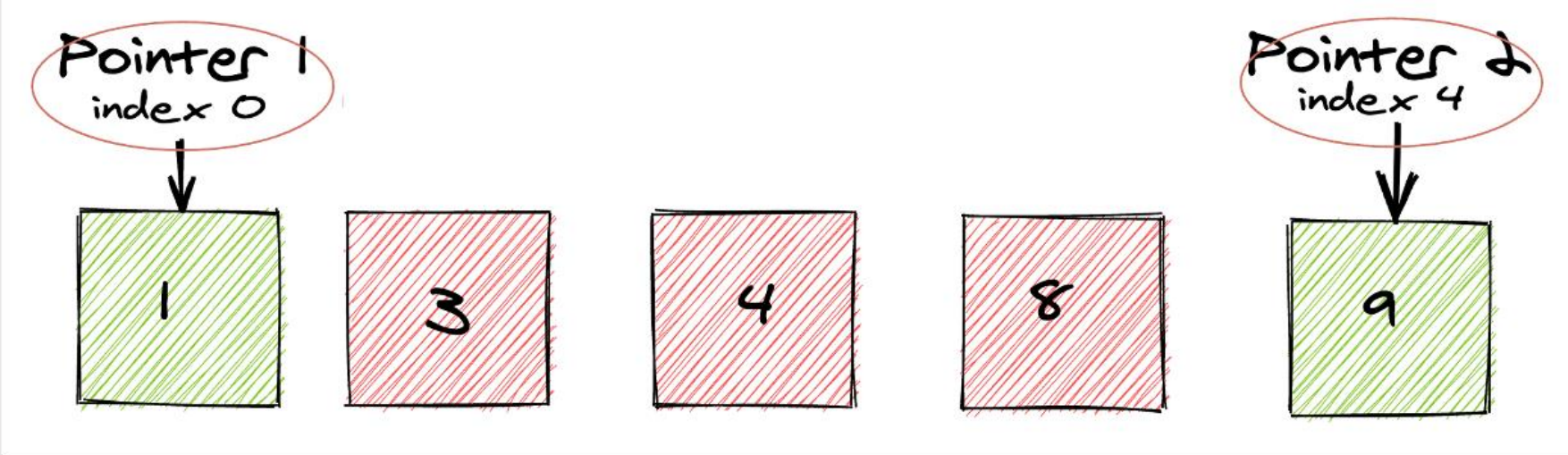
Outline

- Introduce the topic to the academy team including Idea, Problem statement, and solution. **(15 Minutes)**
- Practice a challenge with the team. **(15 Minutes)**
- Take feedback from the team and update later the slides and confluence accordingly. **(10 Minutes)**
- Team to evaluate the session. **(5 Minutes)**

- **Total time: 45 Minutes**

What is Two Pointers

- We can follow the **Two Pointers** approach if we **Sort the Array.**
- We will define two pointers the first one pointing to the beginning of the array and another pointing at the end.



Problem Statement

- You have a sorted array of integer numbers, write a function to **remove all duplicates** from the array without **using any extra space** after that **return** the new length of the array.

Example 1

Input: [2, 3, 3, 3, 6, 9, 9]
Output: 4

Explanation: The first four elements after removing the duplicates will be [2, 3, 6, 9].

Solution

We Can solve it with HashMap, but it will take more memory

```
Complexity is 4.1 Everything is cool!
function removeDuplicates(array) {
    const result = [];
    const map = {};
    for (let i = 0; i < array.length; i++) {
        if (map[array[i]]) {
            continue;
        } else {
            result.push(array[i]);
            map[array[i]] = true;
        }
    }
    return result;
}
```

It’s accepted BUT !

Do we have a better solution??

Solution Two Pointers

- In this problem theirs a sorted array. So, we can use two pointers technique.

- When we initialize the pointers at **the first and end**, it means we are using the two pointers technique to search for a particular element or subarray in a sorted array.

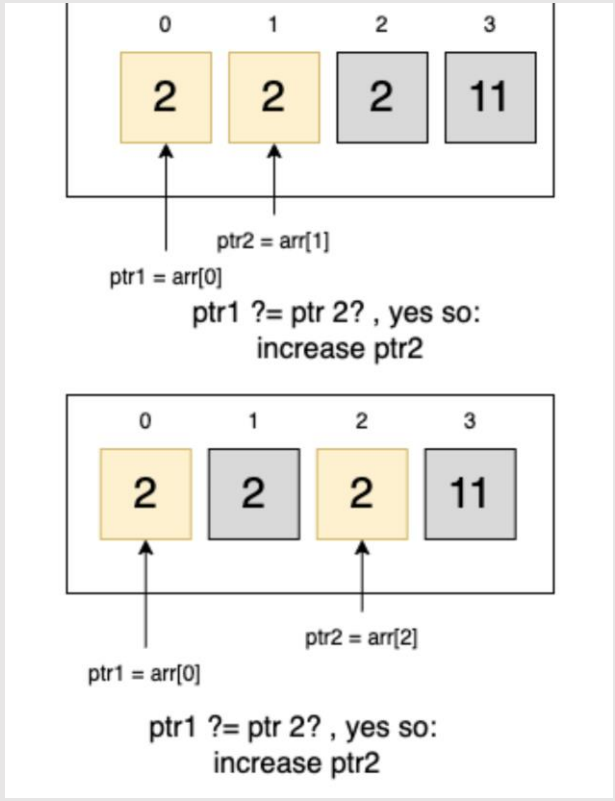
- On the other hand, when we initialize the two pointers at the first, it means we are using the two pointers technique to traverse an array or a string.

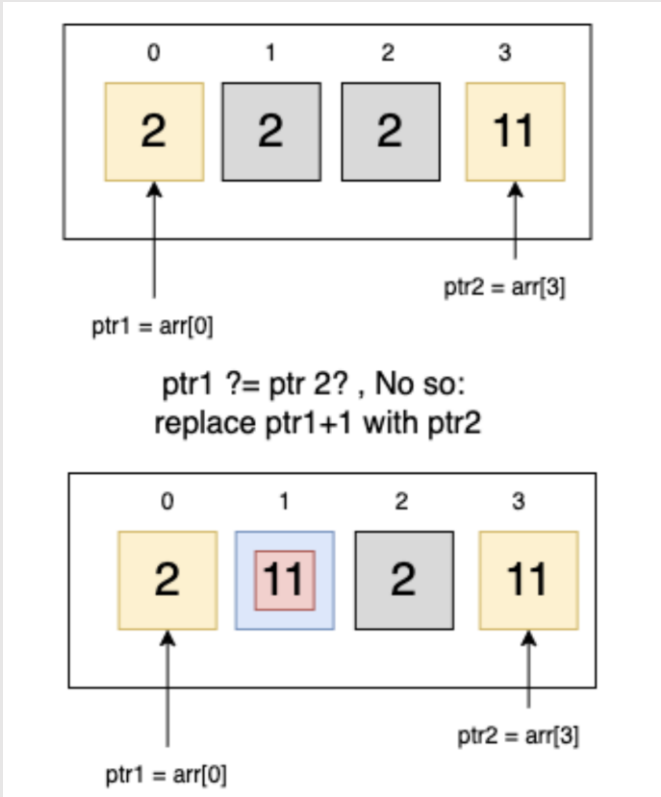
Solution Two Pointers cont.

- One pointer for iterating the array and another one for placing the next non-duplicate number.
- Our algorithm will be to iterate the array and whenever we see a non-duplicate number, we move it next to the last non-duplicate number we've seen.

Example

Example 1: Input: [2, 2, 2, 11]



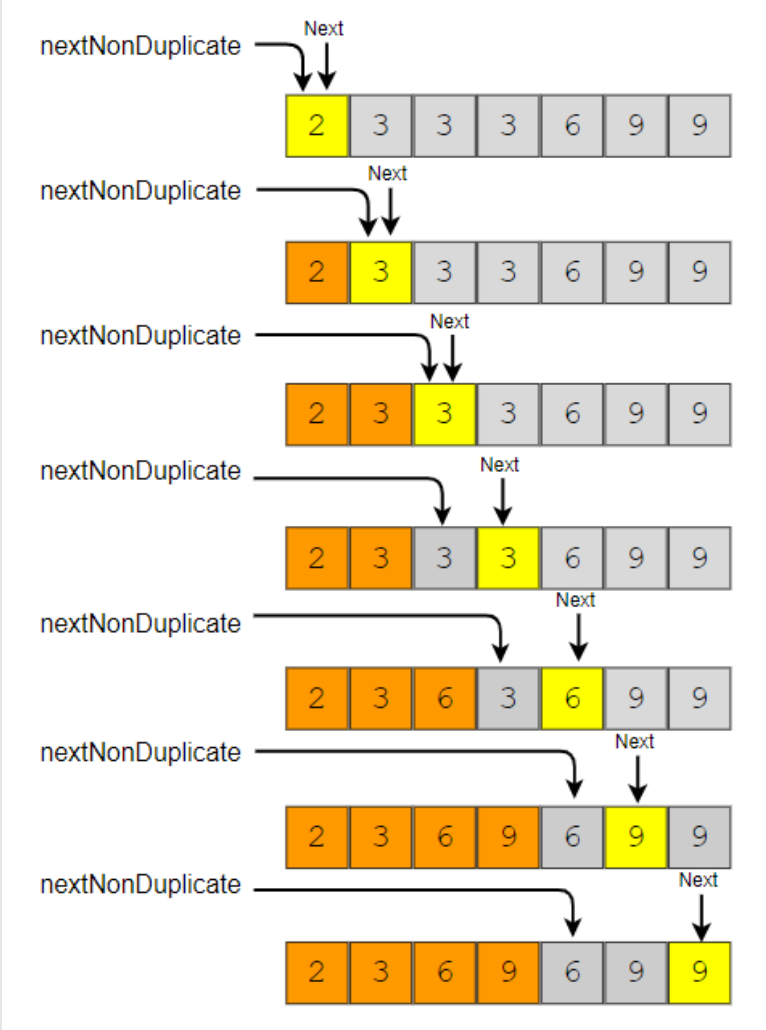


Solution

```
var removeDuplicates = function(nums) {
    let left = 0;
    let right = 1;

    while(right < nums.length){
        if(nums[left] !== nums[right]){
            left++;
            nums[left] = nums[right];
        }
        right++;
    }

    return left+1;
};
```



Solution

Time Complexity *O*(*N*)

where ‘N’ is the total number of elements in the given array.

Space Complexity *O*(1).

https://leetcode.com/problems/valid-palindrome/

Evaluation

- **Let us evaluate this session by filling out the survey.**

- **The aim of the evaluation is to enhance the content.**

Problem Solving Pattern Session evaluation

Any Questions?

Zettachring 6 · 70567 Stuttgart
Fon +49 711.901196 0
Fax +49 711.901196 111

Daimlerstraße 17/1 · 72581 Dettingen an der Erms
Fon +49 711.901196 0
Fax +49 711.901196 111

Neuenhofer Str. 11 · 42657 Solingen
Fon +49 212.2245505 0
Fax +49 212.2245505 110

Ougarit Bld. 1st floor · Al Irsal Street · Al Masayef · Ramallah
Fon +970 2 2988350
Fax +970 2 2988364

19 Hartom Street · Har Hotzvim · Jerusalem
Fon +970 2 2988350
Fax +970 2 2988364

Karatasou 7 · 54627 Thessaloniki
Fon +49 711.901196 0
Fax +49 711.901196 111