7 → Remove duplicates from sorted array -

1 → Question -

Given an integer array nums sorted in non-decreasing order, remove the duplicates in-place such that each unique element appears only once. The relative order of elements should be kept the same.

↑ =

$1\ 2\ 2\ 3 \rightarrow \boxed{1\ 2\ 3}$

Result to be placed in first part of the array nums.

If there are k elements after removing the duplicates, then the first k elements of nums should hold the final result. It does not matter what you leave beyond the first k elements.

Return $\boxed{k}$ after placing the final result

Donot allocate extra space for another array. Modifying the input array in-place with O(1) extra memory.

2. Example -

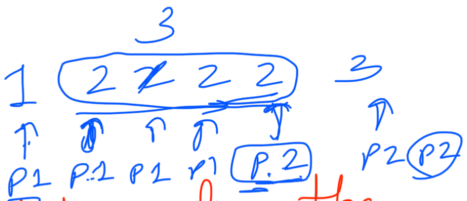
nums = [0, 0, 1, 1, 1, 2, 2, 3, 3, 4]

→ 5 nums = [0, 1, 2, 3, 4]

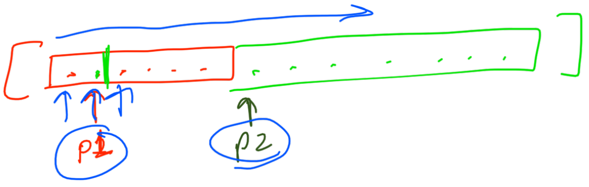
3. Intuition -

Two pointers ? why ?

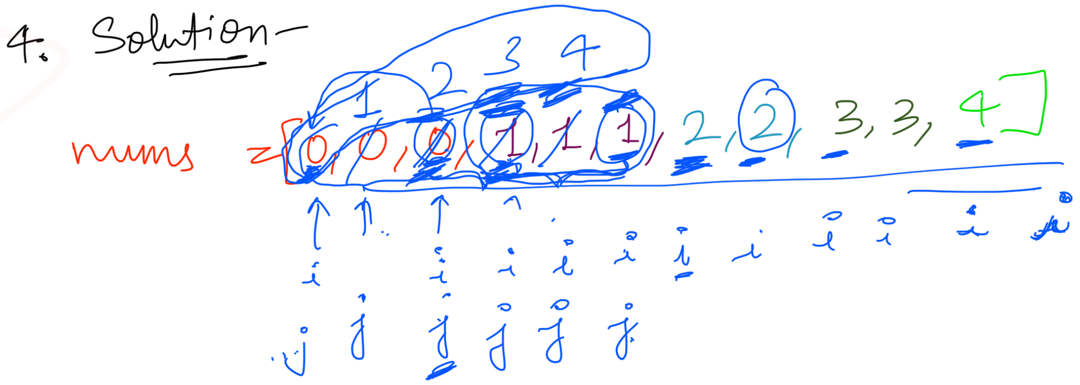
[. . .]



→ handle the duplicates and update only the first k positions



4. Solution-



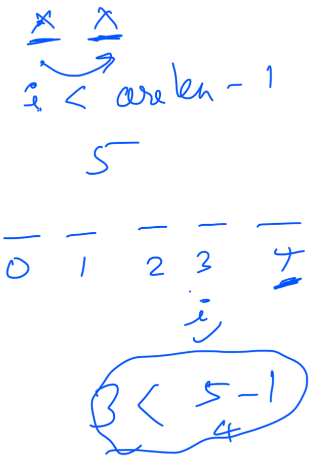
5. Time/space complexity-

T = O(n)

S = O(1)

6. Solution walkthrough-

```
int removeDuplicates (int[] nums) {
        i=0, j=0
        while (i < nums.length) {
            while (i < nums.length - 1 &&
                nums[i] == nums[i+1]) {
                i+=1;
            }
            nums[j] = nums[i];
            i+=1;
            j+=1;
        }
        return j;
}
```



i < arr.len - 1
5
0 1 2 3 4
i
3 < 5-1
4