<https://leetcode.com/problems/remove-duplicates-from-sorted-array-ii/>

Given an integer array nums sorted in **non-decreasing order**, remove some duplicates **[in-place](https://en.wikipedia.org/wiki/In-place_algorithm)** such that each unique element appears **at most twice**. The **relative order** of the elements should be kept the **same**.

Since it is impossible to change the length of the array in some languages, you must instead have the result be placed in the **first part** of the array nums. More formally, 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 k *after placing the final result in the first* k *slots of* nums.

Do **not** allocate extra space for another array. You must do this by **modifying the input array [in-place](https://en.wikipedia.org/wiki/In-place_algorithm)** with O(1) extra memory.

**Custom Judge:**

The judge will test your solution with the following code:

int[] nums = [...]; // Input array

int[] expectedNums = [...]; // The expected answer with correct length

int k = removeDuplicates(nums); // Calls your implementation

assert k == expectedNums.length;

for (int i = 0; i < k; i++) {

assert nums[i] == expectedNums[i];

}

If all assertions pass, then your solution will be **accepted**.

**Example 1:**

Input: nums = [1,1,1,2,2,3]

Output: 5, nums = [1,1,2,2,3,\_]

Explanation: Your function should return k = 5, with the first five elements of nums being 1, 1, 2, 2 and 3 respectively.

It does not matter what you leave beyond the returned k (hence they are underscores).

**Example 2:**

Input: nums = [0,0,1,1,1,1,2,3,3]

Output: 7, nums = [0,0,1,1,2,3,3,\_,\_]

Explanation: Your function should return k = 7, with the first seven elements of nums being 0, 0, 1, 1, 2, 3 and 3 respectively.

It does not matter what you leave beyond the returned k (hence they are underscores).

**Constraints:**

* 1 <= nums.length <= 3 \* 104
* -104 <= nums[i] <= 104
* nums is sorted in **non-decreasing** order.

**Attempt 1: 2023-02-23**

**Solution 1: Two Pointers (30 min)**

class Solution {

public int removeDuplicates(int[] nums) {

int i = 0;

for(int n : nums) {

if(i < 2 || n > nums[i - 2]) {

nums[i] = n;

i++;

}

}

return i;

}

}

**Refer to**

<https://leetcode.com/problems/remove-duplicates-from-sorted-array-ii/solutions/27976/3-6-easy-lines-c-java-python-ruby/comments/119993>

**i和n两个指针，一开始齐头并进，同时指向一个位置，当当前的数值不比他前前一个数值大的时候，意味着出现了3个或3个以上的相同值，此时不满足if条件，i停留在不满足的位置，等待下一个更大的数来替换，当出现下一个更大的数字时再次满足if条件，将i所指向的位置替换为该数字，i指向下一个等待替换，此时if条件再次用以检测用来替换**

**的数字，以保证不出现两次以上的重复。**