Remove Duplicates

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 the elements should be kept the **same**. Then return *the number of unique elements in* `nums`.

Consider the number of unique elements of `nums` to be `k`, to get accepted, you need to do the following things:

- Change the array `nums` such that the first `k` elements of `nums` contain the unique elements in the order they were present in `nums` initially. The remaining elements of `nums` are not important as well as the size of `nums`.
- Return `k`.

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 = [1,1,2]
Output: 2, nums = [1,2,_]
Explanation: Your function should return k = 2, with the first two elements of n
ums being 1 and 2 respectively.
It does not matter what you leave beyond the returned k (hence they are unde
rscores).
```

Example 2:

```
Input: nums = [0,0,1,1,1,2,2,3,3,4]
Output: 5, nums = [0,1,2,3,4,_,_,_,_,_]
Explanation: Your function should return k = 5, with the first five elements of n
ums being 0, 1, 2, 3, and 4 respectively.
It does not matter what you leave beyond the returned k (hence they are unde
rscores).
```

First declare two pointers in this case `i` and `j` these are the two variables going through our loop `i` will be our last unique element we had and j will be the current element we are on for example `{1, 2, 3, 4(i), 4, 4, 5, 5(j)}`

```
# i and j are two pointers
        # i is the last unique element
        # j is the current element
        i,j=0,1
```

Next we will make a while loop to say whilst `i ≤ j` and `j < len(nums)` (j < length of the array)

```
# while j is less than the length of nums
        # and i is less than j
        while i<=j and j<len(nums):
```

Our first condition will be to if `i` and `j` and the same we will increase `j` by one to check on the next element.

```
# if the current element is equal to the last unique element
            # increment j by one
            if nums[i]==nums[j]:
                j+=1
```

Otherwise if the `i` and `j` were not the same element we will increase `i` by one and `j` will be assigned the same value.

```
 # else if the current element is not equal to the last unique element
            # set the next element to the current element
            # increment i by one
            else:
                nums[i+1]=nums[j]
                i+=1
```

Don't be afraid to pull the code into your local machine and play around with the different inputs in the main method.

```
# Example usage
if __name__ == "__main__":
    # Create an instance of the Solution class
    solution = Solution()
    # Example input
    nums = [1, 1, 2, 3, 3, 4, 4, 5]

    # Call the removeDuplicates method
    length = solution.removeDuplicates(nums)

    # Print the result
    print("Length of array after removing duplicates:", length)
    print("Array after removing duplicates:", nums[:length])
```

Complete solution

```
class Solution:
    def removeDuplicates(self, nums: List[int]) → int:
        # i and j are two pointers
        # i is the last unique element
        # j is the current element
        i,j=0,1
        # while j is less than the length of nums
        # and i is less than j
        while i<=j and j<len(nums):
            # if the current element is equal to the last unique element
            # increment j by one
            if nums[i]==nums[j]:
                j+=1
            # else if the current element is not equal to the last unique element
            # set the next element to the current element
            # increment i by one
            else:
                nums[i+1]=nums[j]
                i+=1
        # return the length of the array
        return i+1
```