You are given two integer arrays `nums1` and `nums2`, sorted in **non-decreasing order**, and two integers `m` and `n`, representing the number of elements in `nums1` and `nums2` respectively.

Merge `nums1` and `nums2` into a single array sorted in **non-decreasing order**.

The final sorted array should not be returned by the function, but instead be *stored inside the array* `nums1`. To accommodate this, `nums1` has a length of `m + n`, where the first `m` elements denote the elements that should be merged, and the last `n` elements are set to `0` and should be ignored. `nums2` has a length of `n`.

Example 1:

```
Input: nums1 = [1,2,3,0,0,0], m = 3, nums2 = [2,5,6], n = 3
Output: [1,2,2,3,5,6]
Explanation: The arrays we are merging are [1,2,3] and [2,5,6].
The result of the merge is [1,2,2,3,5,6] with the underlined elements coming from nums1.
```

p1 = nums1[0]
p2 = nums2[0]

P1
nums1 = [1, 2, 3, 0, 0, 0]

P2
nums2 = [2, 5, 6]

1 from n1
2 from n1
➜ 2 from n2
3 n1
5 n2
6 n2

3 <= 2

```
result = []

// pointers num 1 and num 2
p1, p2 = 0,0

for i in range(m +n):
    if n1[p1] <= n2[p2]
        result.append( n1[p1])
        p1++
    else
        result.append(n2[p2])
        p2++

// outside the for loop
//Copy result back into nums1
nums1[:m+n] = result
```