3.Hard\08.Merge_2_sorted_array_without_space.cpp

```
/*
QUESTION:
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.

APPROACH:
To merge two sorted arrays, nums1 and nums2, into nums1, we can use a two-pointer approach.
1. Initialize three pointers: i, j, and k, where i points to the last valid element of nums1, j points to the last element of nums2, and k points to the last index of the merged array nums1.
2. Start from the end of the arrays and compare the elements at i and j.
3. If the element at nums1[i] is greater than the element at nums2[j], swap it with the element at nums1[k], decrement i and k.
4. Otherwise, swap the element at nums2[j] with the element at nums1[k], decrement j and k.
5. Repeat steps 3 and 4 until all elements in nums1 and nums2 have been merged.
6. If there are still elements remaining in nums2 after merging, copy them to the remaining positions in nums1.

CODE:
*/

void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
    int i = m - 1; // Pointer for nums1
    int j = n - 1; // Pointer for nums2
    int k = m + n - 1; // Pointer for merged array nums1

    while (i >= 0 && j >= 0) {
        if (nums1[i] > nums2[j]) {
            swap(nums1[i], nums1[k]);
            i--;
            k--;
        } else {
            swap(nums2[j], nums1[k]);
            j--;
            k--;
        }
    }

    // Copy remaining elements from nums2 to nums1 if any
    while (j >= 0) {
        swap(nums2[j], nums1[k]);
```

```
        j--;
        k--;
    }
}

/*
TIME COMPLEXITY: O(m + n), where m and n are the lengths of nums1 and nums2 respectively.
The merging process requires iterating through both arrays once.
SPACE COMPLEXITY: O(1)
The merge is performed in-place without using any additional space.

*/


```