Better solution → O(n+m) } leetcode sol^n

This solution can be achieved with the help of 3 points. Take i = m-1, j = n-1 and k = m+n-1. Traverse until any one of array finishes.

1) nums1[i] > nums2[j], then insert nums1[i] & decrement k and i both.
2) nums1[i] <= nums2[j], then insert nums2[j] & decrement k & j both.

Now if any one of the array is bigger, run loop for that also.

Dry run

nums1 → {1, 2, 3, 0, 0, 0}, m = 3

nums2 → {2, 5, 6} n = 3

{1, 2, 3, 0, 0, 0} {2, 5, 6}

↑i ↑k ↑j

1) nums1[i] = 3

nums2[j] = 6

Here nums2[j] > nums1[i] & hence insert nums2[j] & decrement both k & j.

	nums1]	nums2]
2)	{1, 2, 3, 0, 0, 6}	{2, 5, 6}
	↑ ↑	↑
	i k	j

nums1[i] = 3
nums2[j] = 5
nums2[j] > nums1[i] & hence insert nums2[j] & decrement both k & j

3)	{1, 2, 3, 0, 5, 6}	{2, 5, 6}
	↑ ↑	↑
	i k	j

nums1[i] = 3
nums2[j] = 2
nums1[i] > nums2[j] & hence insert nums1[i] and decrement both i & j.

4)	{1, 2, 3, 3, 5, 6}	{2, 5, 6}
	↑ ↑	↑
	i k	j

nums1[i] = 2
nums2[j] = 2
Both equal & hence insert nums2[j] & then decrement both j & k

5)	{1, 2, 2, 3, 5, 6}	{2, 5, 6}
	↑	↑
	i, k	j

6) Traverse fully from i & hence nums1 here remains same.

{1, 2, 2, 3, 5, 6} is the answer.

Code

```
void merge (vector <int> & nums1, int m,
vector <int> & nums2 , int n) {
    // Maintaining 3 pointers
    int i = m-1;
    int j = n-1;
    int k = m+n-1;
    // Traverse until any of the array finishes
    while (i>=0 && j>=0) {
        // Inserting element of nums1
        if(nums1[i] > nums2[j]) {
            nums1[k--] = nums1[i--];
        }
        // Inserting element of nums2
        else {
            nums1[k--] = nums2[j--];
        }
    }
    // For handling case in which one array is bigger than other array.
    while (i>=0) {
        nums1[k--] = nums1[i--];
    }
    while (j>=0) {
        nums1[k--] = nums2[j--];
    }
}
```

Time complexity = $O(n+m)$
Space complexity = $O(1)$