Merge sorted arrays :-

→ Given arrays num1 & num2 are sorted in ascending order.

→ m represents the size of num1 & n represents the size of num2.

→ Merge num1 & num2 such that all the elements are sorted in ascending order.

→ Do not create an additional array. Update num1 itself as the total size of num1 is (m+n).

→ Additional 0s are padded in num1 to accomodate (m+n) elements.

* Traditional approach to merge two sorted arrays :-

num1 = [1, 2, 4, 6] (i) num2 = [3, 5] (j) result = [ ] (k) (m+n)

→ if num1[i] <= num2[j], then push num1[i] to result.

num1 = [1, 2, 4, 6] (i) num2 = [3, 5] (j) result = [1,] (k)

Again, num1[i] <= num2[j], then push num1[i] to result.

num1 = [1, 2, 4, 6] (i) num2 = [3, 5] (j) result = [1, 2,] (k)

Now, num2[j] < num1[i], so push num2[j] to result.

num1 = [1, 2, 4, 6] (i) num2 = [3, 5] (j) result = [1, 2, 3,] (k)

num1[i] < num2[j], push num1[i] to result[k]

num1 = [1, 2, 4, 6] (i) num2 = [3, 5] (j) result = [1, 2, 3, 4,] (k)

num2[j] < num1[i], push num2[j] to result[k]

num1 = [1, 2, 4, 6] num2 = [3, 5] result = [1, 2, 3, 4, 5,]

nums2 is completed. Now, fill result[] with the remaining elements of nums1.

* Code :-

```
public static int[] mergeSortedArrays (int[] nums1, int[] nums2) {
    int m = nums1.length;
    int n = nums2.length;
    int i=0, j=0, k=0;
    int[] result = new int[m+n];

    while (i<m && j<n) {
      if (nums1[i] <= nums2[j]) {
        result[k++] = nums1[i++];
      } else {
        result[k++] = nums2[j++];
      }
    }

    while (i<m) {
      result[k++] = nums1[i++];
    }

    while (j<n) {
      result[k++] = nums2[j++];
    }

    return result;
}
```

Time complexity :- O(m+n)

Space complexity :- O(m+n)

* Problem specific approach :-

→ We are not supposed to create any additional array.

→ We need to modify & update nums1[ ].

1) nums1 = [1, 2, 3, 0, 0, 0] (3)
(i points to 3, k points to last 0)

nums2 = [2, 5, 6] (3)   k = (m+n) − 1 = 5
(j points to 6)

If nums1[i] <= nums2[j], then nums1[k] = nums2[j]

2) nums1 = [1, 2, 3, 0, 0, 6]
(i points to 3, k points to second 0)

nums2 = [2, 5, 6]
(j points to 5)

→ Again nums1[i] <= nums2[j], so nums1[k] = nums2[j]

3) nums1 = [1, 2, 3, 0, 5, 6]
(i points to 3, k points to 0)

nums2 = [2, 5, 6]
(j points to 2)

→ If nums1[i] > nums2[j], then nums1[k] = nums1[i]

4) nums1 = [1, 2, 3, 3, 5, 6]
(i points to 2, k points to 3)

nums2 = [2, 5, 6]
(j points to 2)

→ Again nums1[i] <= nums2[j], so nums1[k] = nums2[j]

5) nums1 = [1, 2, 2, 3, 5, 6]

nums2 = [2, 5, 6]

* Code :-

```
public static void mergeSortedArrays (int[] nums1, int m, int[] nums2, int n) {
    int i = m-1;
    int j = n-1;
    int k = (m+n)-1;
```

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