88. Merge Sorted Array

Merge num1 and num2 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 num1. To accommodate this, num1 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 num2 has a length of n.

Example 1:

Input: num1 =[1,2,3,0,0,0], m=3, num2 =[2,5,6], n=3

O/P: [1,2,2,3,5,6]

Explanation: the array 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 num1.

```
Public class MergeSortedArrays{
    public void merge(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--]=nums1[i--];
      }else{
         nums1[k--]=nums2[j--];
      }
    }
while(j>=0){
    nums1[k--]=nums2[j--];
}}
public static void main(String[]args){
 MergeSortedArray merger=new MergeSortedArray();
   int[] nums1={1,2,3,0,0,0};
   int m=3;
   int[] nums2 ={2,5,6};
   int n=3;
   merge.merge(nums1, m, nums2, n);
   for(int num: nums1){
        System.out.print(num+" ");
   }}}
```

Initial.

m=3   n=3.

i = m-1 = 3-1 = 2 $\boxed{i=2}$

j = n-1 = 3-1 = 2 $\boxed{j=2}$

k = m+n-1 = 3+3-1 = 6-1 $\boxed{k=5}$

while (i>=0 && j>=0)

  [illegible]>=0 && [illegible]>=0

  ✓   ✓.

nums1[i]=3
nums2[j]=6.

if (nums1[i]>nums2[j])

  [illegible] > [illegible] F.

k in nums1

while (2>=0) {

j-- = 1

k-- = 4.

nums2 = 5
nums1 = 4.

Iteration 2:

while (i >= 0 && j >= 0)
    3 >= 0 ✓ && 5 >= 0

if (nums1[i] > nums2[j])
    5 > 3 ✓ ~~[illegible]~~ k in nums1
    while (5 >= 0) ✓.
    nums1[k--] = nums2[j--]
        j-- = 0
        k-- = 3.

Iteration 3.

$\boxed{\text{nums2} = 2}$

while (i >= 0 && j >= 0)
    3 >= 0 ✓ && 0 >= 0 ✓.

if (nums1[i] > nums2[j])
    3 > 2 ✓ k in nums1.

while (2 >= 0)
    i-- = 1 → i now points to 2 in num1
    k-- = 2 → k now points to position 2 in nums1

Iteration 4:

if (nums[i] > nums2[j])
    2 >

```
Public class mergeSortedArray{
    Public static void main(String [] args)
  {
   int nums1[] = {1,2,3,0,0,0};
   int nums2[] = {2,3,5};
   int m=3
   int n=3
   mergeArrays(nums1, nums 2, m,n);
   for(int i=0; i<m+n; i++){
       System.out.print(nums1[i]+" ");
    }
 Public static void mergeArray(int [] nums1,
                   int[]nums2, int m, int n ){
   int i = m-1;
   int j = n-1;
   int k = m+n-1;
   while (j>=0){
      if(i>=0 && nums1[i]> nums2[j])
      {
          nums1[k] = nums1[i];
          k--;
          i--;
      }
     else{
          nums1[k] = nums2[j];
          k--;
          j--;
      }
```

i = m-1 = 3-1 = 2

i = 2

j = n-1 = 3-1 = 2

j = 2

k = m+n-1

= 3+3-1 = 5.

(j >= 0)

2 >= 0 ✓.

i >= 0 && nums1[i] > nums2[j]

2 >= 0 && 6 > 3 ✓.

✓

nums1[k] = 6.

k = 4

i = 1.

---

nums1[i] = 3 > nums2[j] = 5 ✗

2 >= 0 ✓ ✓.

else part.

nums2[j] at num1[k]

j = 0

k = 3.

---

nums1[i] = 3 > nums[j] = 2 ✓

2 >= 0 ✓.

nums1[k] = nums2[j];

nums1[i] at num1[k]

i = 1

k = 2

---

nums1[i] = 2 nums2[j] = 2

2 > 2 ✗

else part

---

nums2[j] at num1[k]

j = -1

k = 1.