**难度: 简单**

给你两个有序整数数组 nums1 和 nums2，请你将 nums2 合并到 nums1 中，使 nums1 成为一个有序数组。

说明：

初始化 nums1 和 nums2 的元素数量分别为 m 和 n 。

你可以假设 nums1 有足够的空间（空间大小大于或等于 m + n）来保存 nums2 中的元素。

示例：

输入：

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

nums2 = [2,5,6], n = 3

输出：[1,2,2,3,5,6]

提示：

-10^9 <= nums1[i], nums2[i] <= 10^9

nums1.length == m + n

nums2.length == n

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/merge-sorted-array

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

**思路1(效率低):**

1). 直接把nums2的值添加到nums1现有数组的后面

2). 然后调用Arrays的sort方法进行排序即可

|  |
| --- |
| public void merge(int[] nums1, int m, int[] nums2, int n) {  for(int i = m, j = 0; i < m+n; i++,j++) {  nums1[i] = nums2[j];  }  Arrays.sort(nums1);  } |

**思路2(使用指针,从尾部开始):**

1). 创建三个指针,分别指向nums1的m-1,指向nums2的n-1,指向nums1的最后个下标

2). 比较一下两个指针的值谁大,大的就把值放到num1s最后的下标,并且该下标-1,对于的指针也-1

3). 判断第二个数组的指针是否大于等于０，如果大于等于就继续在nums1下标-1,第二个指针的下表同样 -1

|  |
| --- |
| public void merge2(int[] nums1, int m, int[] nums2, int n) {  int p1 = m -1;  int p = nums1.length -1;  int p2 = n -1;  while(p2 >= 0 && p1 >= 0) {  if (nums1[p1] < nums2[p2]) {  nums1[p--] = nums2[p2--];  }else {  nums1[p--] = nums1[p1--];  }  }  while (p2 >= 0) {  nums1[p--] = nums2[p2--];  }  } |

**思路3(同样使用指针,但是头指针,并且需要创建一个临时数组):**

1). 三个指针刚开始全部都为0

2). 然后分别比较来移动指针

3). 最后判断指针有没有提前结束的,把提前结束的指针重新进行移动

|  |
| --- |
| public void merge3(int[] nums1, int m, int[] nums2, int n) {  int p1 = 0;  int p2 = 0;  int p = 0;  int[] arr = new int[m+n];  while(p2 < n && p1 < m) {  if (nums1[p1] < nums2[p2]) {  arr[p] = nums1[p1];  p1++;  }else {  arr[p] = nums2[p2];  p2++;  }  p++;  }  while (p1 < m) {  arr[p++] = nums1[p1++];  }  while (p2 < n) {  arr[p++] = nums2[p2++];  }  for (int i = 0; i < arr.length; i++) {  nums1[i] = arr[i];  }  } |