Array #5

Leetcode #88 ✓

Merge Sorted Array ✓

https://leetcode.com/problems/merge-sorted-array/description/ ✓

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 array should not be returned by the function, but stored inside the array nums1. To accomodate 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, 3, 4, 5, 6~~] 1 2 2 3 5 6

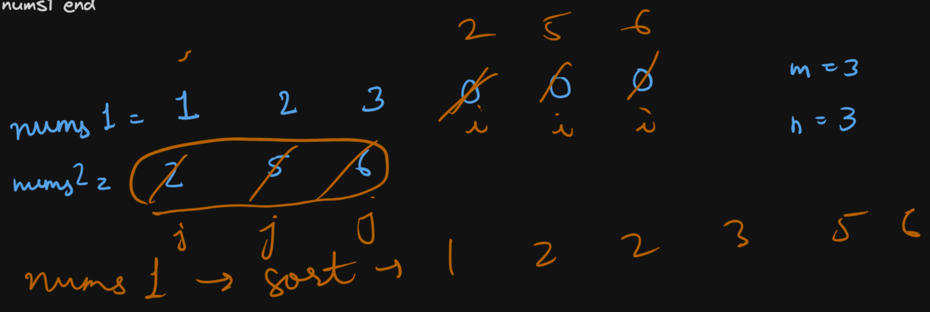
Example 2:
Input: nums1 = [1], m = 1, nums2 = [], n = 0
Output: [1]

Example 3:
Input: nums1 = [~~0~~], m = 0, nums2 = [1], n = 1
Output: [1]

Constraints:
nums1.length == m + n
nums2.length == n
0 <= m, n <= 200
1 <= m + n <= 200
-10^9 <= nums1[i], nums2[j] <= 10^9

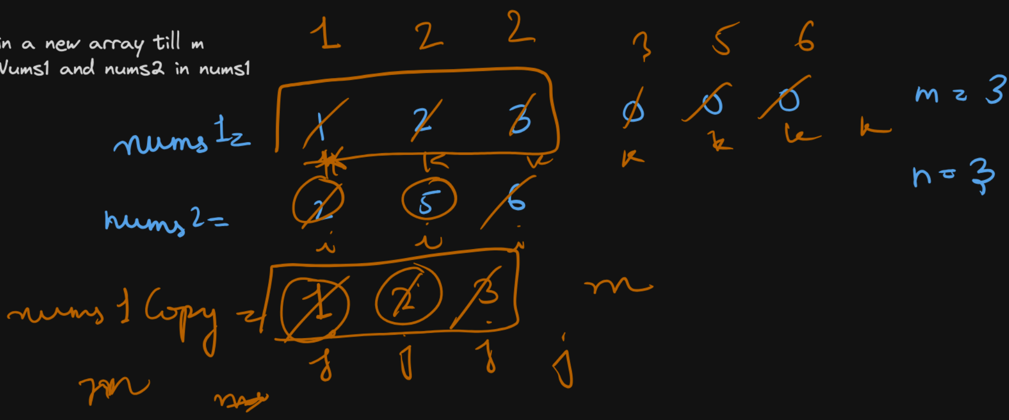
Companies:
Meta, Amazon, Google, Microsoft, Adobe, Apple, etc

Approach 1:
Copy nums2 in nums1 end
Sort nums1



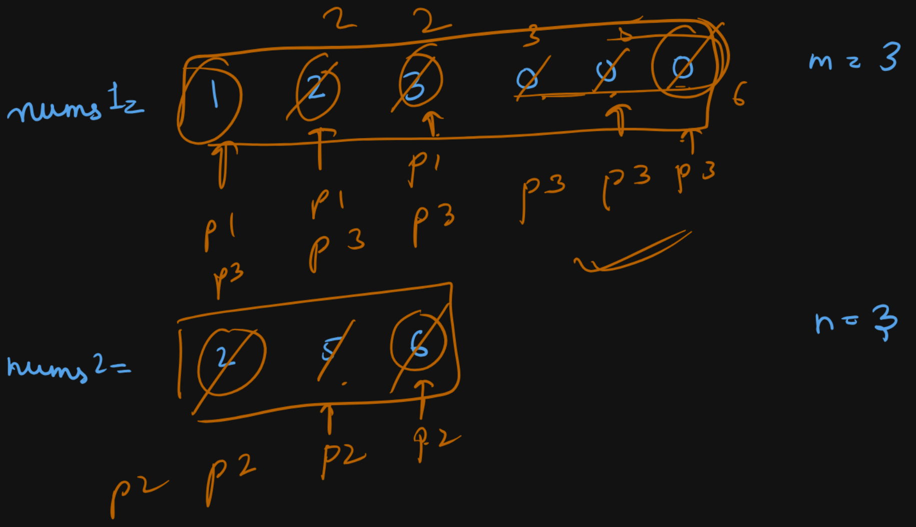
Time complexity:
O((M + N) (log(M + N)))
Space Complexity:
O(Depends upon sort algorithm of programming language sort function)

Approach 2:
Copy nums1 in a new array till m
Merge tempNums1 and nums2 in nums1



Time complexity:
O(m + n)
Space Complexity:
O(m)

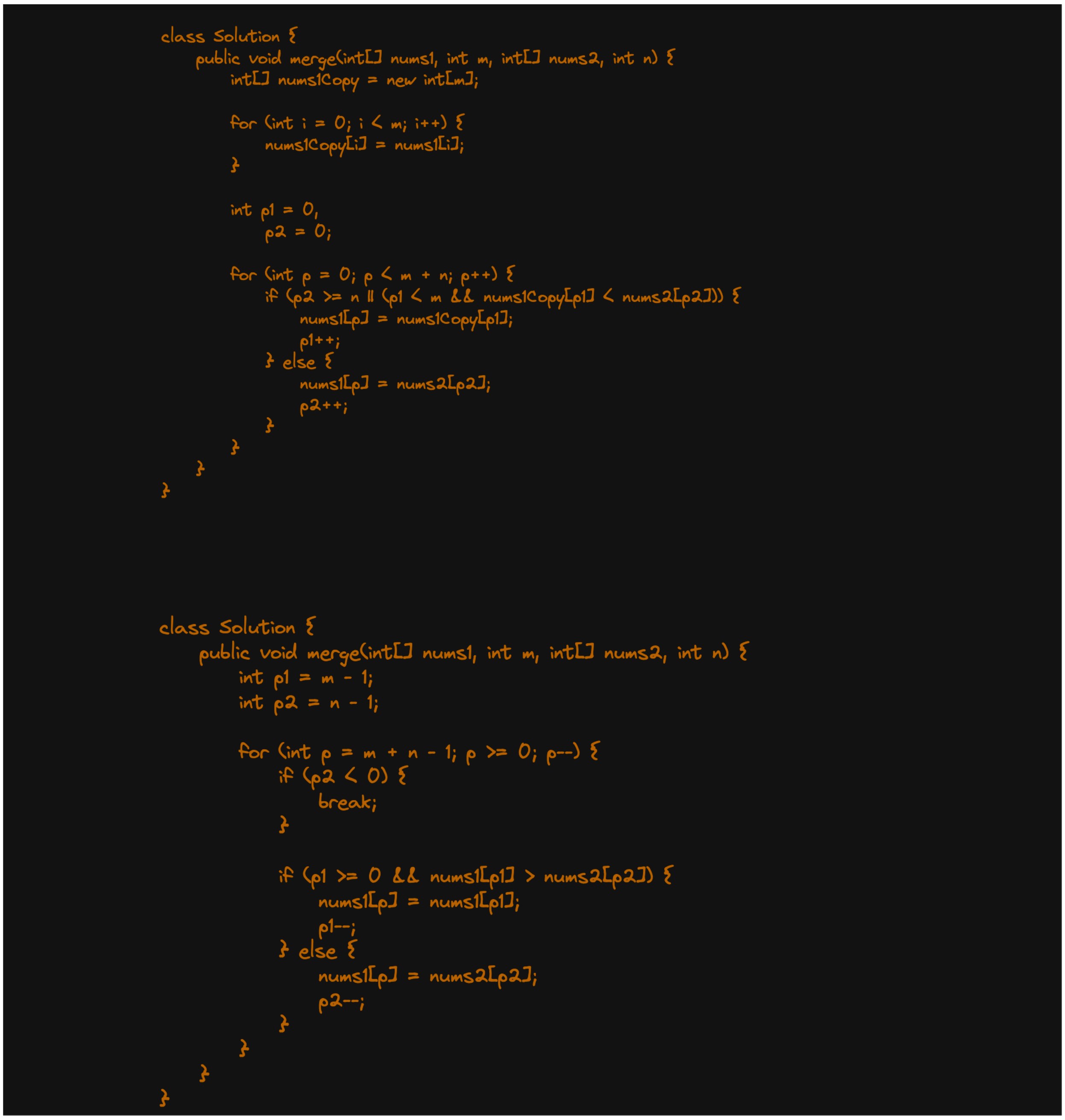
Approach 3:
Need three pointer from back of nums1 and nums2 and one from nums1 end where last element of nums1 is there



Time complexity:
O(m + n)
Space Complexity:
O(1)

```
class Solution {
    public void merge(int[] nums1, int m, int[] nums2, int n) {
        // approach 1
        for (int i = 0; i < n; i++) {
            nums1[i + m] = nums2[i];
        }

        Arrays.sort(nums1);
    }
}
```



```java
class Solution {
    public void merge(int[] nums1, int m, int[] nums2, int n) {
        int[] nums1Copy = new int[m];

        for (int i = 0; i < m; i++) {
            nums1Copy[i] = nums1[i];
        }

        int p1 = 0,
            p2 = 0;

        for (int p = 0; p < m + n; p++) {
            if (p2 >= n || (p1 < m && nums1Copy[p1] < nums2[p2])) {
                nums1[p] = nums1Copy[p1];
                p1++;
            } else {
                nums1[p] = nums2[p2];
                p2++;
            }
        }
    }
}
```

```java
class Solution {
    public void merge(int[] nums1, int m, int[] nums2, int n) {
        int p1 = m - 1;
        int p2 = n - 1;

        for (int p = m + n - 1; p >= 0; p--) {
            if (p2 < 0) {
                break;
            }

            if (p1 >= 0 && nums1[p1] > nums2[p2]) {
                nums1[p] = nums1[p1];
                p1--;
            } else {
                nums1[p] = nums2[p2];
                p2--;
            }
        }
    }
}
```