class Solution:

    def findMedianSortedArrays(self, nums1: List[int], nums2: List[int]) -> float:

        # Since one of the arrays can be significantly longer than

        # the other, we always perform binary search on the shorter

        # array to ensure that the time complexity remains

        # O(log(min(m, n))).

        if len(nums1) > len(nums2):

            return self.findMedianSortedArrays(nums2, nums1)

        len1, len2 = len(nums1), len(nums2)

        left, right = 0, len1

        while left <= right:

            part1 = (left + right) // 2

            part2 = (len1 + len2 + 1) // 2 - part1

            max\_left1 = float('-inf') if part1 == 0 else nums1[part1 - 1]

            min\_right1 = float('inf') if part1 == len1 else nums1[part1]

            max\_left2 = float('-inf') if part2 == 0 else nums2[part2 - 1]

            min\_right2 = float('inf') if part2 == len2 else nums2[part2]

            if max\_left1 <= min\_right2 and max\_left2 <= min\_right1:

                if (len1 + len2) % 2 == 0:

                    return (max(max\_left1, max\_left2) + min(min\_right1, min\_right2)) / 2

                else:

                    return max(max\_left1, max\_left2)

            elif max\_left1 > min\_right2:

                right = part1 - 1

            else:

                left = part1 + 1