4 寻找两个有序数组的中位数

题目描述

给定两个大小为 m 和 n 的有序数组`nums1`和`nums2`。

请你找出这两个有序数组的中位数，并且要求算法的时间复杂度为 O(log(m + n))。

你可以假设`nums1`和 `nums2`不会同时为空。

示例 1:

```
nums1 = [1, 3]
nums2 = [2]

则中位数是 2.0
```

示例 2:

```
nums1 = [1, 2]
nums2 = [3, 4]

则中位数是 (2 + 3)/2 = 2.5
```

代码

```
class Solution:
    def findMedianSortedArrays(self, nums1: List[int], nums2: List[int])
-> float:
        def get_kth_smallest(a_start, b_start, k):
            if k <= 0 or k > len(nums1) - a_start + len(nums2) - b_start:
                raise ValueError('k is out of the bounds of the input
lists')

            if a_start >= len(nums1):
                return nums2[b_start + k - 1]
            if b_start >= len(nums2):
                return nums1[a_start + k -1]
            if k == 1:
                return min(nums1[a_start], nums2[b_start])

            mid_nums1, mid_nums2 = float('inf'), float('inf')

            if k // 2 - 1 < len(nums1) - a_start:
                mid_nums1 = nums1[a_start + k // 2 - 1]
            if k // 2 - 1 < len(nums2) - b_start:
                mid_nums2 = nums2[b_start + k // 2 - 1]
```

```
            if mid_nums1 < mid_nums2:
                return get_kth_smallest(a_start + k // 2, b_start, k - k
// 2)
            return get_kth_smallest(a_start, b_start + k // 2, k - k //
2)

        right = get_kth_smallest(0, 0, 1 + (len(nums1) + len(nums2)) //
2)

        if (len(nums1) + len(nums2)) % 2 == 1:
            return right

        left = get_kth_smallest(0, 0, (len(nums1) + len(nums2)) // 2)
        return (left + right) / 2.0
```

成功 显示详情 ›

执行用时 : 76 ms, 在Median of Two Sorted Arrays的Python3提交中击败了93.78% 的用户

内存消耗 : 14.4 MB, 在Median of Two Sorted Arrays的Python3提交中击败了5.28% 的用户

进行下一个挑战：

匹配子序列的单词数 子数组的最小值之和 数组形式的整数加法

炫耀一下:

提交时间	状态	执行用时	内存消耗	语言
几秒前	通过	76 ms	14.4 MB	python3