4. Median of Two Sorted Arrays ★

Question | Editorial Solution

My Submissions (/problems/median-of-two-sorted-arrays/submissions/)

Total Accepted: **111266**   Total Submissions: **563498**   Difficulty: **Hard**

Notes

There are two sorted arrays **nums1** and **nums2** of size m and n respectively.

Find the median of the two sorted arrays. The overall run time complexity should be O(log (m+n)).

Example 1:

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

The median is 2.0
```

Example 2:

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

The median is (2 + 3)/2 = 2.5
```

Subscribe (/subscribe/) to see which companies asked this question

Show Tags

Have you met this question in a real interview? Yes No

Discuss (https://leetcode.com/discuss/questions/oj/median-of-two-sorted-arrays)

Pick One (/problems/random-one-question/)

C++

```
class Solution {
public:
    int findKthValue(const vector<int>::const_iterator& xbegin,const vector<int>
        ybegin,const vector<int>::const_iterator& yend, int K) {
        int xsize = xend - xbegin, ysize = yend - ybegin;
        assert(K <= xsize + ysize && K > 0);
        if (xsize == 0) {
            return *(ybegin + K - 1);
        }
        else if (ysize == 0) {
            return *(xbegin + K - 1);
        }
```

Send Feedback (mailto:admin@leetcode.com?subject=Feedback)

```
        else {
            if (K == 1) {
                return min(*xbegin, *ybegin);
            }
            else {
                int kHalf = K / 2 - 1;
                if (kHalf < xsize && kHalf < ysize) {
                    auto xkHalf = xbegin + kHalf, ykHalf = ybegin + kHalf;
                    if (*xkHalf > *ykHalf) {
                        return findKthValue(xbegin, xend, ykHalf+1, yend, K - (k
                    }
                    else if (*xkHalf < *ykHalf) {
                        return findKthValue(xkHalf+1, xend, ybegin, yend, K - (k
                    }
                    else { // ==
                        if (K % 2 == 0) {
                            return *xkHalf;
                        }
                        else {
                            return findKthValue(xkHalf + 1, xend, ykHalf + 1, ye
                        }
                    }
                }
                else if (xsize <= kHalf) {
                    auto ykHalf = ybegin + kHalf, xback = xend-1;
                    if (*ykHalf >= *xback) {
                        return findKthValue(xend, xend, ybegin, yend, K - xsize)
                    }
                    else {
                        return findKthValue(xbegin, xend, ykHalf+1, yend, K - (k
                    }
                }
                else { //ysize <= kHalf
                    auto xkHalf = xbegin + kHalf, yback = yend - 1;
                    if (*xkHalf >= *yback) {
                        return findKthValue(xbegin, xend, yend, yend, K - ysize)
                    }
                    else {
                        return findKthValue(xkHalf+1, xend, ybegin, yend, K - (k
                    }
                }
            }
        }
    }

    int findKthValue(const vector<int>& x,const vector<int>& y, int K) {
        return findKthValue(x.cbegin(), x.cend(), y.cbegin(), y.cend(), K);
    }

    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
        int K = nums1.size() + nums2.size();
        if (K % 2 != 0) { // 奇數
            return (double)findKthValue(nums1, nums2, (K + 1) / 2);
        }
        else {
            return ((double)findKthValue(nums1, nums2, K / 2) + (double)findKthV
        }
    }
```

Notes

Custom Testcase □

✉ Send Feedback (mailto:admin@leetcode.com?subject=Feedback)

Run Code Submit Solution