两个排序数组的中位数 · Median of two Sorted Arrays

中文

Adobe (/problem/?tags=adobe) 微软 (/problem/?tags=microsoft) Zenefits (/problem/?tags=zenefits) 掉盒子 (/problem/?tags=dropbox) 雅虎 (/problem/?tags=yahoo) 苹果 (/problem/?tags=apple) 谷歌 (/problem/?tags=google) 优步 (/problem/?tags=uber) 排序数组 (/problem/?tags=sorted-array) 分治法 (/problem/?tags=divide-and-conquer) 数组 (/problem/?tags=array)

描述

两个排序的数组A和B分别含有m和n个数，找到两个排序数组的中位数，要求时间复杂度应为O(log (m+n))。

样例

样例1

```
输入:
A = [1,2,3,4,5,6]
B = [2,3,4,5]
输出: 3.5
```

样例2

```
输入:
A = [1,2,3]
B = [4,5]
输出: 3
```

挑战

时间复杂度为O(log n)

说明

中位数的定义：

- **这里的 中位数 等同于数学定义里的 中位数 。**
- 中位数是排序后数组的中间值。
- 如果有数组中有n个数且n是奇数，则中位数为$\mathrm{A}[(\mathrm{n}-1)/2]$。
- 如果有数组中有n个数且n是偶数，则中位数为 $(\mathrm{A}[\mathrm{n}/2]+\mathrm{A}[\mathrm{n}/2+1])/2$.
- 比如：数组A=[1,2,3]的中位数是2，数组A=[1,19]的中位数是10。

在线评测地址: https://www.lintcode.com/problem/median-of-two-sorted-arrays/ (https://www.lintcode.com/problem/median-of-two-sorted-arrays/)

收起题目描述 ^

语言类型 ALL (39) python (20) cpp (10) java (9) 上传题解

令狐冲
更新于 11/27/2020, 5:56:46 PM

分治法。时间复杂度 $log(n+m)$

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
public class Solution {
    public double findMedianSortedArrays(int A[], int B[]) {
        int n = A.length + B.length;

        if (n % 2 == 0) {
            return (
                findKth(A, 0, B, 0, n / 2) +
                findKth(A, 0, B, 0, n / 2 + 1)
            ) / 2.0;
        }

        return findKth(A, 0, B, 0, n / 2 + 1);
    }

    // find kth number of two sorted array
    public static int findKth(int[] A, int startOfA,
                              int[] B, int startOfB,
                              int k){
        if (startOfA >= A.length) {
            return B[startOfB + k - 1];
        }
        if (startOfB >= B.length) {
            return A[startOfA + k - 1];
        }

        if (k == 1) {
            return Math.min(A[startOfA], B[startOfB]);
        }

        int halfKthOfA = startOfA + k / 2 - 1 < A.length
            ? A[startOfA + k / 2 - 1]
            : Integer.MAX_VALUE;
        int halfKthOfB = startOfB + k / 2 - 1 < B.length
            ? B[startOfB + k / 2 - 1]
            : Integer.MAX_VALUE;

        if (halfKthOfA < halfKthOfB) {
            return findKth(A, startOfA + k / 2, B, startOfB, k - k / 2);
        } else {
            return findKth(A, startOfA, B, startOfB + k / 2, k - k / 2);
        }
    }
}
```


令狐冲
更新于 10/3/2020, 1:17:00 PM

findMedian -> findKth

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param A: An integer array.
    @param B: An integer array.
    @return: a double whose format is *.5 or *.0
    """
    def findMedianSortedArrays(self, A, B):
        n = len(A) + len(B)
        if n % 2 == 1:
            return self.findKth(A, 0, B, 0, n // 2 + 1)
        else:
            smaller = self.findKth(A, 0, B, 0, n // 2)
            bigger = self.findKth(A, 0, B, 0, n // 2 + 1)
            return (smaller + bigger) / 2

    def findKth(self, A, index_a, B, index_b, k):
        if len(A) == index_a:
            return B[index_b  + k - 1]
        if len(B) == index_b:
            return A[index_a + k - 1]
        if k == 1:
            return min(A[index_a], B[index_b])

        a = A[index_a + k // 2 - 1] if index_a + k // 2 <= len(A) else None
        b = B[index_b + k // 2 - 1] if index_b + k // 2 <= len(B) else None

        if b is None or (a is not None and a < b):
            return self.findKth(A, index_a + k // 2, B, index_b, k - k // 2)
        return self.findKth(A, index_a, B, index_b + k // 2, k - k // 2)
```

获赞 15　　添加评论

令狐冲
更新于 8/22/2020, 5:45:35 AM

二分答案的方法，时间复杂度 $O(log(range) * (log(n) + log(m)))$

其中 range 为最小和最大的整数之间的范围。 可以拓展到 Median of K Sorted Arrays

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
public class Solution {
    /*
     * @param A: An integer array
     * @param B: An integer array
```

```
     * @return: a double whose format is *.5 or *.0
     */
    public double findMedianSortedArrays(int[] A, int[] B) {
        int n = A.length + B.length;

        if (n % 2 == 0) {
            return (findKth(A, B, n / 2) + findKth(A, B, n / 2 + 1)) / 2.0;
        }

        return findKth(A, B, n / 2 + 1);
    }

    // k is not zero-based, it starts from 1
    public int findKth(int[] A, int[] B, int k) {
        if (A.length == 0) {
            return B[k - 1];
        }
        if (B.length == 0) {
            return A[k - 1];
        }

        int start = Math.min(A[0], B[0]);
        int end = Math.max(A[A.length - 1], B[B.length - 1]);

        // find first x that >= k numbers is smaller or equal to x
        while (start + 1 < end) {
            int mid = start + (end - start) / 2;
            if (countSmallerOrEqual(A, mid) + countSmallerOrEqual(B, mid) < k) {
                start = mid;
            } else {
                end = mid;
            }
        }

        if (countSmallerOrEqual(A, start) + countSmallerOrEqual(B, start) >= k) {
            return start;
        }

        return end;
    }

    private int countSmallerOrEqual(int[] arr, int number) {
        int start = 0, end = arr.length - 1;

        // find first index that arr[index] > number;
        while (start + 1 < end) {
            int mid = start + (end - start) / 2;
            if (arr[mid] <= number) {
                start = mid;
            } else {
                end = mid;
            }
        }

        if (arr[start] > number) {
            return start;
        }

        if (arr[end] > number) {
            return end;
        }

        return arr.length;
    }
}
```

九章-小原
更新于 12/23/2020, 3:55:16 PM

归并

解题思路

- 最简单的思路是将两个数组合并，然后返回新数组的中位数。两个有序数组的合并也是经典归并排序算法的一步，我们可以新开一个数组，保存合并后的结果。但是我们这样会做额外的工作，因为我们不必保存整个新数组，只需要知道新数组的中位数即可。
- 因此，更简便的方法是，使用双指针分别对两个数组遍历，比较两个指针下的元素大小，每次移动更小的指针，通过移动次数确定中位数的位置。

算法流程

- 令指针 `p1` 和 `p2` 分别指向两个数组，初始指向位置 `0` 。我们需要遍历 `(m + n)/2 + 1` 次，每次比较两个位置的元素，在第 `k` 次比较时，较小的那个值就是两个数组中第 `k + 1` 小的数。如果一个指针已经走到了数组末尾，那么移动另一个指针，否则每次将指向较小数的指针后移，直到遍历到中位数。
- 为了将奇偶情况合并，在代码中用了 `left` 和 `right` 保存中间值，如果是奇数直接返回 `right` ，如果是偶数就返回 `(left + right) / 2` 。

复杂度分析

- 复杂度分析：
  - 时间复杂度：$O(m+n)$， `m` 和 `n` 分别是两个数组的长度。双指针遍历两个数组，指针移动次数是 `0(m+n)` 级。
  - 空间复杂度：$O(1)$。

代码

python | java | c++

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param: A: An integer array
    @param: B: An integer array
    @return: a double whose format is *.5 or *.0
    """
    def findMedianSortedArrays(self, A, B):
        m, n = len(A), len(B)
        p1, p2 = 0, 0
        left, right = -1, -1
        for i in range((m + n) // 2 + 1):
            left = right
            # p2 右移
            if p1 >= len(A) or p2 < len(B) and A[p1] > B[p2]:
                right = B[p2]
                p2 += 1
            # p1 右移
            else:
                right = A[p1]
                p1 += 1
        # 长度和是奇数
        if (m + n) % 2 == 1:
            return right
        # 长度和是偶数
        return (left + right) / 2
```

二分

解题思路

- 题目要求时间复杂度为$O(log(m+n))$，不难想到二分法。双指针方法中，我们一个一个的排除不可能的元素。如果充分利用数组的有序性，就能一半一半的排除。具体来说，假设我们要找第 k 小数，通过二分，可以每次循环排除掉 k/2 个数。

算法流程

- 建立辅助函数 getKth ，参数有数组 A ， A 的起始指针 start1 和终止指针 end1 , 相对应的有 B 、 start2 和 end2 ，以及整数 k ，目标是找到 A[start1:end1] 和 B[start2:end2] 中第 k 小的元素。
- 在主程序中，看 m + n 的奇偶性，并调用 getKth 函数。如果是奇数，返回数组 A 和 B 的第 (m + n) // 2 + 1 小元素；如果是偶数，返回数组 A 和 B 的第 (m + n) // 2 小和第 (m + n) // 2 + 1 小元素的均值。
- getKth(nums1, start1, end1, nums2, start2, end2, k) 函数的实现方法： *如果有数组在 [start:end] 范围内为空，说明该数组已经排除完毕，第 k 小的元素一定存在于另一数组中，计算好索引位置返回即可。* 如果 k 为1，说明已经找到第 k 小的数，那就是 A[start1] 和 B[start2] 中的较小值，直接返回即可。 *定义指针 i 和 j ，分别指向 A 和 B ，使得 A[start1:i] 和 B[start2:j] 的长度分别为 k // 2 ，通过比较 A[i] 和 B[j] 的大小，我们就可以确定排除哪段元素。* 如果 A[i] > B[j] ，说明 B[start2:j] 不可能为第 k 小元素。我们对 A[start1:end1] 和 B[j + 1:end2] 继续送入 getKth 进行递归， k 应该更新为 k - (j - start2 + 1) 。* 反之，说明 A[start1:i] 不可能为第 k 小元素。我们对 A[i + 1:end1] 和 B[start2:end2] 继续送入 getKth 进行递归， k 应该更新为 k - (i - start1 + 1) 。

复杂度

- 时间复杂度：$O(log(m+n))$。 m 和 n 分别是两个数组的长度。二分法的复杂度为$O(log(m+n))$。
- 空间复杂度：$O(1)$。

代码

python java c++

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param: A: An integer array
    @param: B: An integer array
    @return: a double whose format is *.5 or *.0
    """
    def findMedianSortedArrays(self, A, B):
        m, n = len(A), len(B)
        # 如果是奇数
        if (m + n) % 2 == 1:
            return self.getKth(A, 0, len(A) - 1, B, 0, len(B) - 1, (m + n) // 2  + 1)
        # 如果是偶数
        left = self.getKth(A, 0, len(A) - 1, B, 0, len(B) - 1, (m + n) // 2)
        right = self.getKth(A, 0, len(A) - 1, B, 0, len(B) - 1, (m + n) // 2 + 1)
        return (left + right) / 2

    def getKth(self, A, start1, end1, B, start2, end2, k):
        len1 = end1 - start1 + 1
        len2 = end2 - start2 + 1
        # 让 len1 的长度小于 len2，这样就能保证如果有数组空了，一定是 len1
        if (len1 > len2):
            return self.getKth(B, start2, end2, A, start1, end1, k)
        # A数组排除完毕
        if (len1 == 0):
            return B[start2 + k - 1]
        # 已经找到第k小的数
        if k == 1:
            return min(A[start1], B[start2])
        # 开始二分
        i = start1 + min(len1, k // 2) - 1
        j = start2 + min(len2, k // 2) - 1
        if (A[i] > B[j]):
            return self.getKth(A, start1, end1, B, j + 1, end2, k - (j - start2 + 1))
        else:
            return self.getKth(A, i + 1, end1, B, start2, end2, k - (i - start1 + 1))
```

快速选择

解题思路

- 对于长度为 m 的数组 A，我们把 A 划分成两个部分 `A1 = A[0, i-1]` 和 `A2 = A[i, m-1]`。对于长度为 n 的数组 B，将 B 划分成 `B1 = B[0, j-1]` 和 `B2 = B[j, n-1]`，使得 `len(A1) + len(B1) == len(A2) + len(B2)`（记作条件1），那么当 A 划分后， B 的划分位置就是确定的。
- 如果我们能够确定 `max(A1[:], B1[:]) <= min(A2[:], B2[:])`（记作条件2），说明我们已经找到合适的划分，能够把 {A, B} 分成长度相等的两份，且一份中的元素全部大于等于另一份。那么，中位数就为 `(max(A1[:], B1[:]) + min(A2[:], B2[:])) / 2`。
- 怎么找到满足条件2的划分呢？选择较短的数组，假设长度为 m，对它可能的划分位置有 `m + 1` 种。我们可以进行二分搜索，那么时间复杂度能够进一步优化到 $O(log(min(m,n))$

算法流程

- 如果 A 长度大于 B，两者交换一下，保证 A 是更短的。
- 对 A 进行二分， low 和 high 初始化为 0 和 m，每次循环不断缩小二分区间 *对 A 的划分位置 `partition_x` 为区间中点 `low + (high - low) // 2`，根据条件1计算出 B 的划分位置 `partition_y`*。 我们在划分处的两端，可以得到四个值： A 左部分的最大值 `max_left_x`， A 右部分的最小值 `min_right_x`， B

左部分的最大值 max_left_y ， B 右部分的最小值 min_right_y 。如果某个值不存在，对于这种边界情况，我们把最大值设为无穷小，最小值设为无穷大，保证后一步的比较恒成立。 *如果此刻的划分满足了条件2，用上述四值来翻译一下就是 max_left_x <= min_right_y and max_left_y <= min_right_x:，那么我们就找到了中位数，是 max(max_left_x, max_left_y) + min(min_right_x, min_right_y)) / 2* 。 如果不满足，如果 max_left_x > min_right_y ，说明 partition_x 位置靠右了，令 high = partition_x - 1 ；反之，说明 partition_x 位置靠左了，令 low = partition_x + 1 。继续我们的循环。

复杂度分析：

- 时间复杂度：$O(log(min(m,n))$。 m 和 n 分别是两个数组的长度。
- 空间复杂度：$O(1)$。

代码

python | java | c++

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param: A: An integer array
    @param: B: An integer array
    @return: a double whose format is *.5 or *.0
    """
    def findMedianSortedArrays(self, A, B):
        # if input1 length is greater than switch them so that input1 is smaller than input2
        if len(A) > len(B):
            return self.findMedianSortedArrays(B, A)

        m, n = len(A), len(B)
        low, high = 0, m

        while low <= high:
            partition_x = low + (high - low) // 2
            partition_y = (m + n + 1)// 2 - partition_x

            # if partition_x is 0 it means nothing is there on left side. Use -INF for max_left_x
            if partition_x == 0:
                max_left_x = float('-inf')
            else:
                max_left_x = A[partition_x - 1]

            # if partition_x is length of input then there is nothing on right side. Use +INF for min_right_x
            if partition_x == m:
                min_right_x = float('inf')
            else:
                min_right_x = A[partition_x]

            if partition_y == 0:
                max_left_y = float('-inf')
            else:
                max_left_y = B[partition_y - 1]

            if partition_y == n:
                min_right_y = float('inf')
            else:
                min_right_y = B[partition_y]

            if max_left_x <= min_right_y and max_left_y <= min_right_x:
                # Now get max of left elements and min of right elements to get the median in case of even length combined array size
                if (m + n) % 2 == 0:
                    return (max(max_left_x, max_left_y) + min(min_right_x, min_right_y)) / 2
                # or get max of left for odd length combined array size.
                else:
                    return max(max_left_x, max_left_y)
            # we are too far on right side for partitionX. Go on left side.
            elif max_left_x > min_right_y:
                high = partition_x - 1
            # we are too far on left side for partitionX. Go on right side.
            else:
                low = partition_x + 1
        return 0
```

令狐冲
更新于 6/9/2020, 7:03:46 AM

每次移除较小的一部分数和较大的一部分数，直到其中一个数组为空 时间复杂度 $O(log(n+m))$ 可以拓展到 Median of K Sorted Arrays

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
public class Solution {
    /*
     * @param A: An integer array
     * @param B: An integer array
     * @return: a double whose format is *.5 or *.0
     */
    public double findMedianSortedArrays(int[] A, int[] B) {
        return findMedian(
            new PartialArray(A),
            new PartialArray(B)
        );
    }
    // 2020/2/28 add begin
    int findInThreeNumber(int a, int b, int c){
        if(b >= a && a >= c) {
            return a;
        }
        if(c >= b && b >= a) {
            return b;
        }
        return c;
    }

    int findInFourNumber(int a, int b, int c, int d) {
        if(c < d){
            return Math.max(a, c);
        }
        return Math.min(b, c);
    }

    double lengthIsOne(PartialArray A, PartialArray B) {
        if(B.size() % 2 == 1){
            int firstMedianIndex = B.getLowerMedianIndex();
            int firstMedian = B.getLowerMedian();
            int secondMedian = findInFourNumber(B.get(firstMedianIndex - 1), B.get(firstMedianIndex + 1), A.get(0), firstMedian);
            return (firstMedian + secondMedian) / 2.0;
        }
        else {
            return 1.0 * findInThreeNumber(A.get(0), B.getLowerMedian(), B.getUpperMedian());
        }
    }
    // 2020/2/28 add end
    private double findMedian(PartialArray A, PartialArray B) {
        while (!A.isEmpty() && !B.isEmpty()) {
            if (A.size() == 1 && B.size() == 1) {
                return (A.getMedian() + B.getMedian()) / 2.0;
            }
            if (A.size() == 1) { // 2020/2/28 add
                return lengthIsOne(A, B);
            }
```

```
            if (B.size() == 1) { // 2020/2/28 add
                return lengthIsOne(B, A);
            }
            PartialArray lowerArr = A;
            int lowerIndex = A.getLowerMedianIndex();
            if (A.getLowerMedian() > B.getLowerMedian()) {
                lowerArr = B;
                lowerIndex = B.getLowerMedianIndex();
            }

            PartialArray upperArr = A;
            int upperIndex = A.getUpperMedianIndex();
            if (A.getUpperMedian() < B.getUpperMedian()) {
                upperArr = B;
                upperIndex = B.getUpperMedianIndex();
            }

            int numOfRemoved = Math.min(
                lowerArr.getNumOfLower(lowerIndex),
                upperArr.getNumOfUpper(upperIndex)
            );


            if (lowerArr.get(lowerIndex) == upperArr.get(upperIndex)) {
                return lowerArr.get(lowerIndex);
            }
            lowerArr.removeLower(numOfRemoved);
            upperArr.removeUpper(numOfRemoved);
        }

        if (A.isEmpty()) {
            return B.getMedian();
        }

        return A.getMedian();
    }
}

class PartialArray {
    int[] arr;
    int start, end;

    PartialArray(int[] arr) {
        this.arr = arr;
        this.start = 0;
        this.end = arr.length - 1;
    }

    public int getLowerMedian() {
        return arr[(start + end) / 2];
    }

    public int getUpperMedian() {
        return arr[(start + end + 1) / 2];
    }

    public int getLowerMedianIndex() {
        return (start + end) / 2;
    }

    public int getUpperMedianIndex() {
        return (start + end + 1) / 2;
    }

    public int size() {
        return end - start + 1;
    }
```

```
    public double getMedian() {
        return (getUpperMedian() + getLowerMedian()) / 2.0;
    }

    public boolean isEmpty() {
        return size() == 0;
    }

    public int getNumOfUpper(int index) {
        if (index == end) {
            return 1;
        }
        return end - index;
    }

    public int getNumOfLower(int index) {
        if (index == start) {
            return 1;
        }
        return index - start;
    }

    public void removeLower(int numOfRemoved) {
        start += numOfRemoved;
    }

    public void removeUpper(int numOfRemoved) {
        end -= numOfRemoved;
    }

    public int get(int index) {
        return arr[index];
    }
}
```

获赞 5　　8 条评论

李助教
更新于 8/16/2020, 10:22:41 PM

<!-- 补充：二分答案的办法 -->

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
# 二分答案的办法

class Solution:
    """
    @param: A: An integer array
    @param: B: An integer array
    @return: a double whose format is *.5 or *.0
    """
    def findMedianSortedArrays(self, A, B):
        # write your code here
        len_a, len_b = len(A), len(B)
        if (len_a + len_b) % 2 == 1:
            return self.finf_kth(A, B, (len_a + len_b) // 2 + 1)
        else:
            left = self.finf_kth(A, B, (len_a + len_b) // 2 )
            right = self.finf_kth(A, B, (len_a + len_b) // 2 + 1)
            return (left + right) / 2

    def finf_kth(self, A, B, k):
        if len(A) == 0:
            left, right = B[0], B[-1]
        elif len(B) == 0:
            left, right = A[0], A[-1]
        else:
            left, right = min(A[0], B[0]), max(A[-1], B[-1])
        while left + 1 < right:
            mid = (left + right) // 2
            count1 = self.helper(A, mid)
            count2 = self.helper(B, mid)
            if count1 + count2 < k:
                left = mid
            else:
                right = mid
        count1 = self.helper(A, left)
        count2 = self.helper(B, left)
        if count1 + count2 >= k:
            return left
        else:
            return right

    def helper(self, array, flag):
        if len(array) == 0:
            return 0
        left, right = 0 ,len(array) - 1
        while left + 1 < right:
            mid = (left + right) // 2
            if array[mid] <= flag:
                left = mid
            else:
                right = mid
        if array[right] <= flag:
            return right + 1
        if array[left] <= flag:
            return left + 1
        return 0
```

获赞 3　　3 条评论

令狐冲
更新于 6/9/2020, 7:04:29 AM

There are two sorted arrays A and B 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)).

 详细题解请见九章算法微博: http://weibo.com/3948019741/Bz1uAswtv
 ```cpp class Solution { public: /* 对于一个长度为n的已排序数列a，若n为奇数，中位数为第(n/2+1)个数 , 若n为偶数，则中位数=[第(n/2)个数 + 第(n/2+1)个数] / 2 如果我们可以在两个数列中求出第K小的元素，便可以解决该问题 不妨设数列A元素个数为n，数列B元素个数为m，各自升序排序，求第k小元素 取A[k / 2] B[k / 2] 比较， 如果 A[k / 2] > B[k / 2] 那么，所求的元素必然不在B的前k / 2个元素中(证明反证法) 反之，必然不在A的前k / 2个元素中，于是我们可以将A或B数列的前k / 2元素删去，求剩下两个数列的 k - k / 2小元素，于是得到了数据规模变小的同类问题，递归解决 如果 k / 2 大于某数列个数，所求元素必然不在另一数列的前k / 2个元素中，同上操作就好。 */ double findKth(vector<int>& A, vector<int>& B, int A_st, int B_st, int k) { // 边界情况，任一数列为空 if (A_st >= A.size()) { return B[B_st + k - 1]; } if (B_st >= B.size()) { return A[A_st + k - 1]; } // k等于1时表示取最小值，直接返回min if (k == 1) return min(A[A_st], B[B_st]); int A_key = A_st + k / 2 - 1 >= A.size() ? INT_MAX : A[A_st + k / 2 - 1]; int B_key = B_st + k / 2 - 1 >= B.size() ? INT_MAX : B[B_st + k / 2 - 1]; if (A_key < B_key){ return findKth(A, B, A_st + k / 2, B_st, k - k / 2); } else { return findKth(A, B, A_st, B_st + k / 2, k - k / 2); } } double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) { int sum = nums1.size() + nums2.size(); double ret; if (sum & 1) { ret = findKth(nums1, nums2, 0, 0, sum / 2 + 1); } else { ret = ((findKth(nums1, nums2, 0, 0, sum / 2)) + findKth(nums1, nums2, 0, 0, sum / 2 + 1)) / 2.0; } return ret; } }; ```

获赞 0　　添加评论

cc189
更新于 12/25/2020, 7:07:37 AM

method1: sort

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    # method1: sort
    def findMedianSortedArrays(self, A, B):
        A = sorted(A + B)
        l = len(A)
        return [(A[l // 2] + A[l // 2 - 1]) / 2, A[l // 2]][l % 2 != 0]
```

获赞 5　　1 条评论

S同学
更新于 6/9/2020, 7:03:46 AM

binary search to solve it: O(log(min(x, y))) only partition on the short array and then base on the short array index, compute the corresponding partition index on the other array.

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
public class Solution {
    /*
     * @param A: An integer array
     * @param B: An integer array
     * @return: a double whose format is *.5 or *.0
     */
    public double findMedianSortedArrays(int[] A, int[] B) {
        if (A.length == 0 && B.length == 0) {
            return 0;
        }

        if (A.length > B.length) {
            return findMedianSortedArrays(B, A);
        }
        int start = 0;
        int end = A.length;
        int x = (A.length + B.length + 1) / 2;
        System.out.println(start + "," + end);
        while (start <= end) {
            int partitionX = (start + end) / 2;
            int partitionY = x - partitionX;

            double Amax = (partitionX == 0) ? Long.MIN_VALUE : A[partitionX - 1];
            double Amin = (partitionX == A.length) ? Long.MAX_VALUE : A[partitionX];
            double Bmax = (partitionY == 0) ? Long.MIN_VALUE : B[partitionY - 1];
            double Bmin = (partitionY == B.length) ? Long.MAX_VALUE : B[partitionY];

            if (Amax <= Bmin && Bmax <= Amin) {
                if ((A.length + B.length) % 2 == 1) {
                    return Math.max(Amax, Bmax);
                } else {
                    return (Math.max(Amax, Bmax) + Math.min(Amin, Bmin)) / 2.0;
                }
            } else if (Amax > Bmin) {
                end = partitionX - 1;
            } else {
                start = partitionX + 1;
            }
        }
        throw new IllegalArgumentException();
    }
}
```

获赞 5　　3 条评论

Yuchen
更新于 7/8/2020, 10:53:48 PM

时间复杂度为 O(min(log(n),log(m))) 二分法找出使得 (max_left_X <= min_right_Y) and (max_left_Y <= min_right_X) 的partition， 这样偶数情况下median就是 (max(max_left_X,max_left_Y) + min(min_right_X,min_right_Y))/2； 奇数情况下median就是 max(max_left_X,max_left_Y)

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
```

```
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
def findMedianSortedArrays(A, B):
    # write your code here


    ###################################################################

    if len(B) < len(A):
        x = len(B)
        y = len(A)
        X = B
        Y = A

    else:
        x = len(A)
        y = len(B)
        X = A
        Y = B

    ###################################################################

    low_X = 0
    high_X = x

    while low_X <= high_X:

        partition_X = (low_X + high_X) // 2
        partition_Y = (x + y + 1) // 2 - partition_X   # to in sure that numbers in left Y
                                                       # is more than right Y

        print(partition_X,partition_Y)

        if partition_X == 0:
            max_left_X = - float("inf")
        else:
            max_left_X = X[partition_X - 1]

        if partition_X == x:
            min_right_X = float("inf")
        else:
            min_right_X = X[partition_X]

        if partition_Y == 0:
            max_left_Y = - float("inf")
        else:
            max_left_Y = Y[partition_Y - 1]

        if partition_Y == y:
            min_right_Y = float("inf")
        else:
            min_right_Y = Y[partition_Y]

        print("X:",max_left_X,min_right_X)
        print("Y:",max_left_Y,min_right_Y)


        ###########################################################

        if (max_left_X <= min_right_Y) and (max_left_Y <= min_right_X):

            if (x + y) % 2 == 0:

                return (max(max_left_X,max_left_Y) + min(min_right_X,min_right_Y))/2
```

```
            else:
                return max(max_left_X,max_left_Y)

        elif max_left_X > min_right_Y:
            high_X = partition_X - 1 # it has to be -1 otherwise it wont move in edge case

        else: #max_left_Y > min_right_X
            low_X = partition_X + 1  # it has to be +1 otherwise it wont move in edge case
```

获赞 4　　添加评论

Wendy0601
更新于 6/9/2020, 7:03:48 AM

使用了二分法。 时间复杂度是O(range * log(m+n)), 其中range是两个数列中最大最小的差值。 其中找到第k个的算法：原理是利用快速排序的思想， 每次求出A,B中比中位数小的数目之和 （也用到了快速排序的方法），如果小于k， 坐标右移， 反之左移。 这个方法把快速排序用的非常熟练了。 谢谢令老师解释。

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param: A: An integer array
    @param: B: An integer array
    @return: a double whose format is *.5 or *.0
    """
    def findMedianSortedArrays(self, A, B):
        c = len(A) + len(B)
        k = int(c / 2)
        if c % 2 == 1:
            return self.findkth(A,B,k + 1)
        else:
            small = self.findkth(A,B,k)
            big = self.findkth(A,B,k + 1)
            return (small + big)/2.0

    def findkth(self, A, B, k):
        if len(A) == 0:
            return B[k - 1]
        if len(B) == 0:
            return A[k - 1]

        start = int (min (A[0], B[0]))
        end = max(A[len(A) - 1], B[len(B) - 1])

        while start + 1 < end:
            mid =  int((start + end) / 2 )
            k1 = self.countsmallerequ(A, mid)
            k2 = self.countsmallerequ(  B, mid)
            if k1 + k2 < k:
                start = mid
            else:
                end = mid
        if self.countsmallerequ(A,start) + self.countsmallerequ(B, start) >= k:
            return start
        return end


    def countsmallerequ(self, A, num):
        left, right = 0, len(A) - 1

        while left + 1 < right:
            mid = int((right + left)/2)
            if A[mid] <= num:
                left = mid
            else:
                right = mid

        if A[left] > num:
            return left
        if A[right] > num:
            return right

        return len(A)
```


获赞 3 1 条评论

cc189
更新于 6/9/2020, 7:03:48 AM

method2: binary search

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    # method2: binary search
    def findMedianSortedArrays(self, A, B):
        self.A, self.B = A, B
        l = len(A) + len(B)
        m = l // 2 + 1
        return [(self.kth(m - 1) + self.kth(m)) / 2, self.kth(m)][l % 2 != 0]

    def kth(self, k, i=0, j=0):
        A, B = self.A, self.B
        if i >= len(A): return B[j + k - 1]
        elif j >= len(B): return A[i + k - 1]
        elif k == 1: return min(A[i], B[j])

        m = k // 2
        k, ii, jj = k - m, i + m - 1, j + m - 1
        a = A[ii] if ii < len(A) else math.inf
        b = B[jj] if jj < len(B) else math.inf

        if a < b: i = ii + 1
        else: j = jj + 1

        return self.kth(k, i, j)
```

获赞 3　　添加评论

r同学
更新于 6/9/2020, 7:03:49 AM

思想类似于老师上课讲的方法：1）每次比index在k/2位置的数的大小，然后扔掉小的一半；2）每次扔完以后做recursion去寻找余下的数里第k-k/2个数； 比较容易写错的点：1）如果A或者B有一个越界，那接下来只在没越界的里面找；2）如果A和B同时越界，那说明必然是他们的最后一个数里大的那个数（k<=n+m) ; 3）递归的出口是k是0或者1（也就是目标数是前一个或者是当前这个的）

时间复杂度O(log(m+n)) 空间复杂度O(1)

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
public class Solution {
    /*
```

```
     * @param A: An integer array
     * @param B: An integer array
     * @return: a double whose format is *.5 or *.0
     */
    public double findMedianSortedArrays(int[] A, int[] B) {
        // write your code here
        int kth = (A.length + B.length + 1) / 2;
        if ((A.length + B.length) % 2 == 0) {
            int small = find(kth, A, 0, B, 0);
            int large = find(kth + 1, A, 0, B, 0);
            return small/2.0 + large/2.0;
        } else{
            return find(kth, A, 0, B, 0) / 1.0;
        }
    }

    private int find(int kth, int[] A, int indexA, int[] B, int indexB){
        if (kth == 0){
            if (indexA <= A.length && indexB <= B.length){
                return A[indexA - 1] > B[indexB - 1] ? A[indexA - 1] : B[indexB - 1];
            }
            return A[A.length - 1] > B[B.length - 1] ? A[A.length - 1] : B[B.length - 1];
        }

        if (kth == 1){
            if (indexA < A.length && indexB < B.length){
                return A[indexA] < B[indexB] ? A[indexA] : B[indexB];
            }
            if (indexA < A.length){
                return A[indexA];
            }

            if (indexB < B.length){
                return B[indexB];
            }
        }

        int kthhalf = kth / 2;
        if (indexA + kthhalf - 1< A.length && indexB + kthhalf - 1< B.length){
            if (A[indexA + kthhalf - 1] < B[indexB + kthhalf - 1]){
                return find(kth - kthhalf, A, indexA + kthhalf, B, indexB);
            }
            if (A[indexA + kthhalf - 1] > B[indexB + kthhalf - 1]){
                return find(kth - kthhalf, A, indexA, B, indexB + kthhalf);
            }

            return find(kth - kthhalf * 2, A, indexA + kthhalf, B, indexB + kthhalf);
        }

        if (indexA + kthhalf - 1< A.length){
            return find(kth - kthhalf, A, indexA + kthhalf, B, indexB);
        }

        if (indexB + kthhalf - 1< B.length){
            return find(kth - kthhalf, A, indexA, B, indexB + kthhalf);
        }
        return find(kth - kthhalf * 2, A, indexA + kthhalf, B, indexB + kthhalf);
    }
}
```

获赞 2　　3 条评论

Ling
更新于 6/9/2020, 7:03:49 AM

1.把AB比整体分成两半：开始分A的index是i =（0+len(A)//2， 分B的index必定就是j=(len(A)+ len(B)+1)//2-i , 这样A左边+ A右边 的个数= A右边 + B右边。总个数为奇数的时候，我们把多余那个放在左边那半。 2.保证了左半和右半个数相等的情况下, while min_index<= max_index: 如果左半所有数小于右半所有数(Bj-1 ()<=Ai () and Ai-1 ()<= Bj ())那么得到答案： 奇数就是max(Ai-1 (), Bj-1 ()) 偶：（max(Ai-1 (), Bj-1 ())+min（Ai ()+Bj ())/2 如果Bj-1 () > Ai (),说明Ai ()小，应该右移，从右边搜索（i移动j也会跟着移动保持左半个数等于右半个数）则min_index = i+1 如歌Ai-1 ()> Bj (), 说明Ai ()太大， 应该左移， 从左边搜索， 则max_index= i-1

时间复杂度 O(log(min(m,n)))

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param: A: An integer array
    @param: B: An integer array
    @return: a double whose format is *.5 or *.0
    """
    def findMedianSortedArrays(self, A, B):
        # write your code here
        median = 0
        m, n = len(A), len(B)
        if m ==0:
            return B[n//2] if n & 1 else (B[n//2 - 1] + B[n//2])/2.0
        if n == 0:
            return A[m//2] if m & 1 else (A[m//2 - 1] + A[m//2])/2.0

        min_index, max_index = 0, m
        while min_index <= max_index:
            i = (min_index  + max_index)//2
            j = (m + n + 1)//2 - i
            if i < m and  j > 0 and B[j-1] > A[i]:
                min_index = i + 1
            elif i > 0 and j < n and B[j] < A[i-1]:
                max_index = i - 1
            else:
                if i == 0:
                    median = B[j-1]
                elif j == 0:
                    median = A[i-1]
                else:
                    median = max(A[i-1], B[j-1])
                break
        if (m + n) % 2 == 1:
            return median
        else:
            if i == m:
                return (median + B[j])/2.0
            elif j == n:
                return (median + A[i])/2.0
            return (median + min(A[i], B[j]))/2.0
```

获赞 2　　添加评论

Adam57
更新于 8/19/2020, 10:24:53 PM

Based on quick select, O(log(a + b)), a, b is the length of A, B

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param: A: An integer array
    @param: B: An integer array
    @return: a double whose format is *.5 or *.0
    """
    def findMedianSortedArrays(self, A, B):
        # write your code here
        if A is None or B is None:
            return

        l = len(A) + len(B)
        m = l // 2

        if l % 2 == 0:
            return (self.findkth(A, B, 0, 0, m) + self.findkth(A, B, 0, 0, m + 1)) / 2
        else:
            return self.findkth(A, B, 0, 0, m + 1)
    # i_a, i_b mean how many elements already taken from a and b
    def findkth(self, A, B, i_a, i_b, k):
        if i_a >= len(A):
            return B[k - i_a - 1]
        elif i_b >= len(B):
            return A[k - i_b - 1]
        elif k - i_a - i_b == 1:
            return min(A[i_a], B[i_b])

        m = (k - i_a - i_b) // 2
        #if there are less element in A or A larger than B, then throw away all the element in B
        # here i_a + m > len(A) not >=  the reason is A still have m element to offer
        if i_a + m > len(A) or A[i_a + m - 1] > B[i_b + m - 1]:
            i_b += m
        elif i_b + m > len(B) or A[i_a + m - 1] <= B[i_b + m - 1]:
            i_a += m

        return self.findkth(A, B, i_a, i_b, k)
```

获赞 1　　添加评论

九章用户JE3A2D
更新于 6/9/2020, 7:03:57 AM

分类讨论，helper是找到合并后的数组index是k的那个元素的数值

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
public class Solution {
```

```
    /*
     * @param A: An integer array
     * @param B: An integer array
     * @return: a double whose format is *.5 or *.0
     */
    public double findMedianSortedArrays(int[] A, int[] B) {
        // write your code here
        int n = A.length;
        int m = B.length;
        if ((m + n) % 2 == 1) {
            return helper(A, 0, B, 0, (n + m) / 2);
        } else {
            return (helper(A, 0, B, 0, (n + m) / 2)
                    + helper(A, 0, B, 0, (n + m) / 2 - 1)) / 2;
        }
    }

    private double helper(int[] A, int startA, int[] B, int startB, int k) {
        //find the element with index k in the merged array
        if (k == 0) {
            if (startA == A.length) {
                return B[startB];
            }
            if (startB == B.length) {
                return A[startA];
            }
            if (A[startA] > B[startB]) {
                return B[startB];
            } else {
                return A[startA];
            }
        }

        if (k == 1) {
            if (startA == A.length) {
                return B[startB + 1];
            }
            if (startB == B.length) {
                return A[startA + 1];
            }
            if (A[startA] > B[startB]) {
                return helper(A, startA, B, startB + 1, 0);
            } else {
                return helper(A, startA + 1, B, startB, 0);
            }
        }

        int mid = k / 2;
        if (startA + mid  >= A.length) {
            return helper(A, startA, B, startB + mid, k - mid);
        }
        if (startB + mid  >= B.length) {
            return helper(A, startA + mid, B, startB, k - mid);
        }
        if (A[startA + mid] > B[startB + mid]) {
            return helper(A, startA, B, startB + mid, k - mid);
        } else {
            return helper(A, startA + mid, B, startB, k - mid);
        }
    }
}
```

九章用户WP7TG8
更新于 6/9/2020, 7:03:57 AM

归并排序的方法，但不需要递归，仅合并，也不需要再排序了 时间1209ms

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    def findMedianSortedArrays(self, left, right):
        result = []
        while len(left) > 0 and len(right) > 0:
            if (left[0] > right[0]):
                result.append(right.pop(0))
            else:
                result.append(left.pop(0))
        if (len(left) > 0):
            result.extend(left)
        else:
            result.extend(right)
        pp = int(len(result) / 2)
        if len(result) % 2 ==0:
            return (result[pp - 1] + result[pp]) / 2.0
        else:
            return result[pp]
```

获赞 1　　2 条评论

九章用户DN8ZSC
更新于 6/9/2020, 7:03:57 AM

二分法，时间复杂度貌似可以降到O(log(min(m,n))) 相当于在数组A和B中各切一刀，比如分别记为aCut和bCut，两把刀左边的数总共保持为(m+n)//2，A中的刀有m个位置，B中的刀有n个位置，但考虑到A和B的长度不一样，所以有min(m,n)个位置，然后再二分，看刀的位置对不对，不对再调整

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param: A: An integer array
    @param: B: An integer array
    @return: a double whose format is *.5 or *.0
    """
    def findMedianSortedArrays(self, A, B):
        self.totalLength = len(A) + len(B)
        if len(A) <= len(B):
            start = 0
            end = len(A)
        else:
```

```
            start = self.totalLength // 2 - len(B)
            end = self.totalLength // 2
        while start + 1 < end:
            mid = start + (end - start) // 2
            if self.compareOne(A, B, mid) and self.compareTwo(A, B, mid):
                return self.outputMedian(A, B, mid)
            if not self.compareOne(A, B, mid):
                end = mid
            else:
                start = mid
        if self.compareOne(A, B, start) and self.compareTwo(A, B, start):
            return self.outputMedian(A, B, start)
        return self.outputMedian(A, B, end)

    def compareOne(self, A, B, aCut):
        bCut = self.totalLength // 2 - aCut
        if aCut == 0 or bCut == len(B):
            return True
        if A[aCut - 1] <= B[bCut]:
            return True
        return False

    def compareTwo(self, A, B, aCut):
        bCut = self.totalLength // 2 - aCut
        if aCut == len(A) or bCut == 0:
            return True
        if A[aCut] >= B[bCut - 1]:
            return True
        return False

    def outputMedian(self, A, B, aCut):
        if self.totalLength % 2 == 1:
            return self.selectRight(A, B, aCut)
        return (self.selectRight(A, B, aCut) + self.selectLeft(A, B, aCut)) / 2

    def selectRight(self, A, B, aCut):
        bCut = self.totalLength // 2 - aCut
        if aCut == len(A):
            return B[bCut]
        if bCut == len(B):
            return A[aCut]
        return min(A[aCut], B[bCut])

    def selectLeft(self, A, B, aCut):
        bCut = self.totalLength // 2 - aCut
        if aCut == 0:
            return B[bCut - 1]
        if bCut == 0:
            return A[aCut - 1]
        return max(A[aCut - 1], B[bCut - 1])
```

获赞 1　　添加评论

Andy You
更新于 6/9/2020, 7:03:56 AM

发一份C++的题解，二分法，格式方面有什么问题希望大家帮忙指正，谢谢了。

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
```

```
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution {
public:
    /*
     * @param A: An integer array
     * @param B: An integer array
     * @return: a double whose format is *.5 or *.0
     */
    double findMedianSortedArrays(vector<int> &A, vector<int> &B) {

        int totalSize = A.size() + B.size();

        if (totalSize == 0) {
            return 0;
        }

        bool onlyOne = (totalSize % 2 == 1)? true : false;

        if (onlyOne) {
            return findKth(totalSize / 2, A, B);
        }

        return (findKth((totalSize - 1) / 2, A, B) +
                findKth((totalSize - 1) / 2 + 1, A, B)) / 2;
    }

    double findKth (int Kth, vector<int> &A, vector<int> &B) {
        int num = Kth + 1;

        int start, end;
        if (A.empty()) {
            start = B[0];
            end = B[B.size() - 1];
        }
        else if (B.empty()) {
            start = A[0];
            end = A[A.size() - 1];
        }
        else {
            start = min(A[0], B[0]);
            end = max(A[A.size() - 1], B[B.size() - 1]);
        }

        if (lessNumber(end, A) + lessNumber(end, B) < num) {
            return end;
        }

        while (start + 1 < end) {
            int ref = (start + end) / 2;
            if (num - lessNumber(ref, A) - lessNumber(ref, B) <= 0) {
                end = ref;
                continue;
            }
            start = ref;
        }

        return start;
    }

    int lessNumber (int ref, vector<int> &vec) {
        if (vec.empty()) {
            return 0;
        }

        int start = 0, end = vec.size() - 1;
```

```
        while (start + 1 < end) {
            int mid = (start + end) / 2;
            if (vec[mid] >= ref) {
                end = mid;
                continue;
            }
            start = mid;
        }

        if (vec[end] < ref) {
            return end + 1;
        }
        if (vec[start] < ref) {
            return start + 1;
        }
        return 0;
    }

};
```

获赞 1　　添加评论

Wendy0601
更新于 6/9/2020, 7:03:56 AM

使用了findkth 的方法。 时间复杂度是O(log(m + n)). 注意选择A,B的下标要转换成int型的， 否则报错。

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param: A: An integer array
    @param: B: An integer array
    @return: a double whose format is *.5 or *.0
    """
    def findMedianSortedArrays(self, A, B):
        # write your code here
        c = len(A) + len(B)
        if c % 2 == 1:
            k0 = int (c / 2) + 1
            return self.findkth(A, B, k0)
        else:
            k1 =   int(c / 2)
            k2 =   int(c / 2) + 1
            first = self.findkth(A,  B,  k1)
            second = self.findkth(A,  B, k2)
            return (first + second)/2.0

    def findkth (self, A,  B,  k):
        if len(A) == 0:
            return B[k-1]
        if len(B) == 0:
            return A[k-1]
        if k==1:
            return min(A[0], B[0])

        if len(A) >=  int(k/2):
            amid = A[int(k/2) -1]
        else:
            amid = float('Inf')

        if len(B) >=  int(k/2):
            bmid = B[ int(k/2)-1]
        else:
            bmid = float('Inf')

        if amid < bmid:
            return self.findkth(A[int(k/2):], B,  k -int(k/2))
        else:
            return self.findkth(A, B[int(k/2):],  k - int(k/2))
```

获赞 1　　1 条评论

九章用户XYWVD2
更新于 11/21/2020, 7:10:34 AM

基于find k-th

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution {
public:
    /*
     * @param A: An integer array
     * @param B: An integer array
     * @return: a double whose format is *.5 or *.0
     */
    double findMedianSortedArrays(vector<int> &A, vector<int> &B) {
        int m = A.size(), n = B.size();
        // 如果是奇数---只有一个中位数
        if ((m + n) % 2 == 1) {
            return getKth(A, 0, B, 0, (m + n) / 2  + 1);//数组中第(m + n) / 2  + 1 是中位数
        }
        // 如果是偶数 ---- 两个取平均值
        int left = getKth(A, 0, B, 0, (m + n) / 2);
        int right = getKth(A, 0, B, 0, (m + n) / 2 + 1);
        return (left + right) / 2.0;
    }

private:
    int getKth(vector<int>& A, int startA, vector<int>& B, int startB, int k) {
        //循环停止的三种情况
        if(startA >= A.size()) //A已经遍历
            return B[startB + k - 1];
        if(startB >= B.size()) //B已经遍历
            return A[startA + k - 1];

        // 已经找到第k小的数---循环结束
        if (k == 1) {
            return min(A[startA], B[startB]);
        }


        //比较两个数组中第k/2的元素，用来舍弃一部分元素
        int halfKthA = startA + k / 2 - 1 < A.size()
                       ? A[startA + k / 2 - 1] : INT_MAX;
        //如果这个index算出来不在A的index范围内，赋值无穷大
        int halfKthB = startB + k /2 - 1 < B.size()
                      ? B[startB + k / 2 - 1] :INT_MAX;


        if(halfKthA < halfKthB){//A的第K/2个元素小，删除A的
            return getKth(A, startA + k / 2, B, startB, k - k / 2);
        }
        return getKth(A, startA, B, startB + k / 2, k - k / 2);

    }
};
```

获赞 0 添加评论

HUE
更新于 8/14/2020, 4:43:38 AM

代碼簡潔 1. 每次捨去兩排序數組第 k/2 較小者的前 k/2 2. 兩盤續數組取 k/2 時，若超界，則認為其值為 INT_MAX

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution {
public:
    /*
     * @param A: An integer array
     * @param B: An integer array
     * @return: a double whose format is *.5 or *.0
     */
    double findMedianSortedArrays(vector<int> &A, vector<int> &B) {
        int m = A.size(), n = B.size();
        if ((m + n) % 2 == 1) {
            return findKTh(A, 0, B, 0, (m + n) / 2 + 1);
        }
        else {
            return (findKTh(A, 0, B, 0, (m + n) / 2)
                   + findKTh(A, 0, B, 0, (m + n) / 2 + 1)) / 2.0; // 要用 2.0 否則會捨去小數
        }
    }

    int findKTh(vector<int> &A, int aIndex, vector<int> &B, int bIndex, int k) {
        if (aIndex >= A.size()) {
            return B[bIndex + k - 1];
        }
        if (bIndex >= B.size()) {
            return A[aIndex + k - 1];
        }
        if (k == 1) {
            return min(A[aIndex], B[bIndex]);
        }

        int aTarget = aIndex + k / 2 - 1 >= A.size() ? INT_MAX : A[aIndex + k / 2 - 1];
        int bTarget = bIndex + k / 2 - 1 >= B.size() ? INT_MAX : B[bIndex + k / 2 - 1];
        if (aTarget < bTarget) {
            return findKTh(A, aIndex + k / 2, B, bIndex, k - k / 2);
        }
        else {
            return findKTh(A, aIndex, B, bIndex + k / 2, k - k / 2);
        }
    }
};
```

對 A、B 數組的數字範圍進行二分法，找出誰之前(包含他自己)有 k 個數

countSmallerOrEqual() 返回數組中小於等於 target(即 mid) 的數有幾個

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ /
Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution {
public:
    /*
```

```
     * @param A: An integer array
     * @param B: An integer array
     * @return: a double whose format is *.5 or *.0
     */
    double findMedianSortedArrays(vector<int> &A, vector<int> &B) {
        // write your code here
        int n = A.size() + B.size();

        if (n % 2 == 0) {
            return (findKth(A, B, n / 2) + findKth(A, B, n / 2 + 1)) / 2.0;
        }
        else {
                return findKth(A, B, n / 2 + 1);
        }
    }

    int findKth(vector<int> &A, vector<int> &B, int k) {
        if (B.empty()) {
                return A[k - 1];
        }
        if (A.empty()) {
                return B[k - 1];
        }

        int start = min(A[0], B[0]);
        int end = max(A.back(), B.back());

        while (start + 1 < end) {
                int mid = start + (end - start) / 2;
                if (countSmallerOrEqual(A, mid) + countSmallerOrEqual(B, mid) < k) {
                        start = mid;
                }
                else {
                        end = mid;
                }
        }

        if (countSmallerOrEqual(A, start) + countSmallerOrEqual(B, start) >= k) {
                return start;
        }

                return end;
    }

    int countSmallerOrEqual(vector<int> &nums, int target) {
        int start = 0, end = nums.size() - 1;

        while (start + 1 < end) {
                int mid = start + (end - start) / 2;
                if (nums[mid] <= target) {
                        start = mid;
                }
                else {
                        end = mid;
                }
        }

        if (nums[start] > target) {
                return start;
        }
        if (nums[end] > target) {
                return end;
        }

        return nums.size();
    }
};
```

获赞 0 添加评论

九章用户ZY953C
更新于 6/9/2020, 7:04:25 AM

翻译了一下C++的答案。python版本的通不过是因为python 3.x 不支持除号返回整数类型，所以提交会有一个typeError

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    def findKth(self, A, B, A_th, B_th, k):
        if (A_th >= len(A)):
            return B[B_th + k - 1]
        if (B_th >= len(B)):
            return A[A_th + k - 1]
        if (k == 1):
            return min(A[A_th], B[B_th])
        mid = int(k/2)
        A_tmp = sys.maxsize if A_th + mid - 1 >= len(A) else A[A_th + mid - 1]
        B_tmp = sys.maxsize if B_th + mid - 1 >= len(B) else B[B_th + mid - 1]

        if(A_tmp > B_tmp):
            return self.findKth(A, B, A_th, B_th + mid, k - mid)
        else:
            return self.findKth(A, B, A_th + mid, B_th, k - mid)


    def findMedianSortedArrays(self, nums1, nums2):
        """
        :type nums1: List[int]
        :type nums2: List[int]
        :rtype: float
        """
        size = len(nums1) + len(nums2)
        mid = int(size / 2)

        if (size & 1):
            return 1.0 * self.findKth(nums1, nums2, 0, 0, mid+1)
        else:
            left = self.findKth(nums1, nums2, 0, 0, mid)
            right = self.findKth(nums1, nums2, 0, 0, mid + 1)
            return 0.5 * (left + right)
```

获赞 0 1 条评论

Wendy0601
更新于 6/9/2020, 7:04:23 AM

python3 通过。 使用了findkth 的方法。 时间复杂度是O(log(m + n)). 注意选择A,B的下标要转换成int型的， 否则报错。

```
/**
* 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
* - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
* - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
* - Design类课程包括：系统设计 System Design，面向对象设计 OOD
* - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
* - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
*/
class Solution:
    """
    @param: A: An integer array
    @param: B: An integer array
    @return: a double whose format is *.5 or *.0
    """
    def findMedianSortedArrays(self, A, B):
        # write your code here
        c = len(A) + len(B)
        if c % 2 == 1:
            k0 = int (c / 2) + 1
            return self.findkth(A, B, k0)
        else:
            k1 =   int(c / 2)
            k2 =   int(c / 2) + 1
            first = self.findkth(A,  B,  k1)
            second = self.findkth(A,  B, k2)
            return (first + second)/2.0

    def findkth (self, A,  B,  k):
        if len(A) == 0:
            return B[k-1]
        if len(B) == 0:
            return A[k-1]
        if k==1:
            return min(A[0], B[0])

        if len(A) >=  int(k/2):
            amid = A[int(k/2) -1]
        else:
            amid = float('Inf')

        if len(B) >=  int(k/2):
            bmid = B[ int(k/2)-1]
        else:
            bmid = float('Inf')

        if amid < bmid:
            return self.findkth(A[int(k/2):], B,  k -int(k/2))
        else:
            return self.findkth(A, B[int(k/2):],  k - int(k/2))
```

获赞 0　　添加评论

加载更多题解

进阶课程