问题介绍

这是个超级超级经典的分治算法！！这个问题大致是说，如何在给定的两个有序数组里面找其中的中值，或者变形问题，如何在2个有序数组数组中查找Top K的值（Top K的问题可以转换成求第k个元素的问题）。这个算法在很多实际应用中都会用到，特别是在当前大数据的背景下。

我觉得下面的这个思路特别好，特别容易理解！！请按顺序看。是来自leetcode上的stellari英文答案，我整理并自己修改了一下。

预备知识

先解释下“割”

我们通过切一刀，能够把有序数组分成左右两个部分，切的那一刀就被称为割(Cut)，割的左右会有两个元素，分别是左边最大值和右边最小值。

我们定义L = Max(LeftPart)，R = Min(RightPart)

Ps. 割可以割在两个数中间，也可以割在1个数上，如果割在一个数上，那么这个数即属于左边，也属于右边。（后面讲单数组中值问题的时候会说）

比如说[2 3 5 7]这个序列，割就在3和5之间

[2 3 / 5 7]

中值就是（3+5）/2 = 4

如果[2 3 4 5 6]这个序列，割在4上，我们可以把4分成2个

[2 3 (4/4) 5 7]

中值就是（4+4）/2 = 4

这样可以保证不管中值是1个数还是2个数都能统一运算。

割和第k个元素

对于单数组，找其中的第k个元素特别好做，我们用割的思想就是：

常识1：如果在k的位置割一下，然后A[k]就是L。换言之，就是如果左侧有k个元素，A[k]属于左边部分的最大值。（都是明显的事情，这个不用解释吧！）

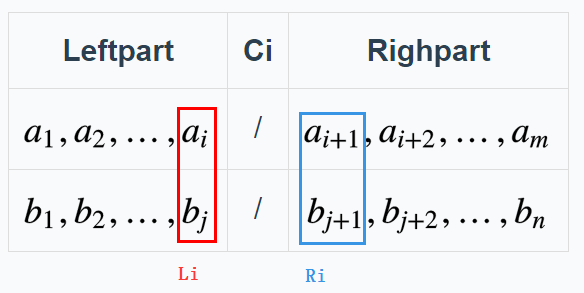
双数组

我们设:

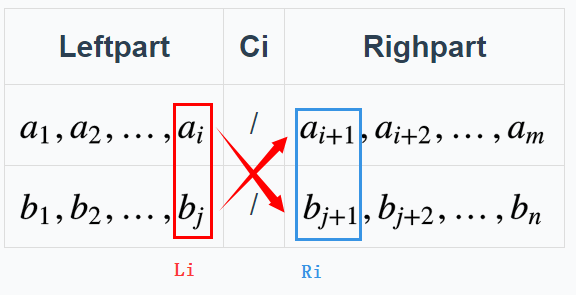
CiCi为第i个数组的割。

LiLi为第i个数组割后的左元素.

RiRi为第i个数组割后的右元素。



如何从双数组里取出第k个元素



首先Li<=RiLi<=Ri是肯定的（因为数组有序，左边肯定小于右边）

如果我们让L1<=R2L1<=R2 && L2<=R1L2<=R1

那么左半边 全小于右半边，如果左边的元素个数相加刚好等于k，那么第k个元素就是Max(L1,L2)，参考上面常识1。

如果 L1>R2，说明数组1的左边元素太大（多），我们把C1减小，把C2增大。L2>R1同理，把C1增大，C2减小。

假设k=3

对于

[1 4 7 9][1 4 7 9]

[2 3 5][2 3 5]

设C1 = 2，那么C2 = k-C1 = 1

[1 4/7 9][1 4/7 9]

[2/3 5][2/3 5]

这时候，L1(4)>R2(3)，说明C1要减小，C2要增大，C1 = 1，C2=k-C1 = 2

[1/4 7 9][1/4 7 9]

[2 3/5][2 3/5]

这时候，满足了L1<=R2L1<=R2 && L2<=R1L2<=R1，第3个元素就是Max(1,3) = 3。

如果对于上面的例子，把k改成4就恰好是中值。

下面具体来看特殊情况的中值问题。

双数组的奇偶

中值的关键在于，如何处理奇偶性，单数组的情况，我们已经讨论过了，那双数组的奇偶问题怎么办，m+n为奇偶处理方案都不同。

让数组恒为奇数

有没有办法让两个数组长度相加一定为奇数或偶数呢？

其实有的，虚拟加入‘#'(这个trick在manacher算法中也有应用)，让数组长度恒为奇数（2n+1恒为奇数）。

Ps.注意是虚拟加，其实根本没这一步，因为通过下面的转换，我们可以保证虚拟加后每个元素跟原来的元素一一对应



映射关系

这有什么好处呢，为什么这么加?因为这么加完之后，每个位置可以通过/2得到原来元素的位置。



在虚拟数组里表示“割”

不仅如此，割更容易，如果割在‘#'上等于割在2个元素之间，割在数字上等于把数字划到2个部分。

奇妙的是不管哪种情况：

Li = (Ci-1)/2

Ri = Ci/2

例：

割在4/7之间‘#'，C = 4，L=(4-1)/2=1 ，R=4/2=2

刚好是4和7的原来位置！

割在3上，C = 3，L=(3-1)/2=1，R=3/2 =1，刚好都是3的位置！

剩下的事情就好办了，把2个数组看做一个虚拟的数组A，目前有2m+2n+2个元素，割在m+n+1处，所以我们只需找到m+n+1位置的元素和m+n+2位置的元素就行了。

左边：A[m+n] = Max(L1+L2)

右边：A[m+n+1] = Min(R1+R2)

Mid = (A[m+n]+A[m+n+1])/2

= (Max(L1+L2) + Min(R1+R2) )/2

至于在两个数组里找割的方案，就是上面的方案。

分治的思路

有了上面的知识后，现在的问题就是如何利用分治的思想。

怎么分？

最快的分的方案是二分，有2个数组，我们对哪个做二分呢？

根据之前的分析，我们知道了，只要C1或C2确定，另外一个也就确定了。这里，为了效率，我们肯定是选长度较短的做二分，假设为C1。

怎么治？

也比较简单，我们之前分析了：就是比较L1,L2和R1,R2。

L1>R2，把C1减小，C2增大。—> C1向左二分

L2>R1，把C1增大，C2减小。—> C1向右二分

越界问题

如果C1或C2已经到头了怎么办？

这种情况出现在：如果有个数组完全小于或大于中值。可能有4种情况：

C1 = 0 —— 数组1整体都比中值大，L1 >R2，中值在2中

C2 = 0 —— 数组1整体都比中值小，L1 <R2，中值在1中

C1 = n\*2 —— 数组1整体都比中值小，L1 <R2，中位数在2中

C2 = m\*2 —— 数组1整体都比中值大，L1 >R2，中位数在1中

考虑下面两种情况了，解决方案：

如果C1 = 0 —> 那么我们缩小L1，L1 = INT\_MIN，保证判断正确。

如果C1 = n\*2 —> 那么我们增大R1，R1 = INT\_MAX，保证判断正确。

剩下两种情况解决方案类似。