

```

        mid1 = (start1 + end1) / 2;
        mid2 = (start2 + end2) / 2;
        // 元素个数为奇数，则 offset 为 0，元素个数为偶数，则 offset 为 1。
        offset = ((end1 - start1 + 1) & 1) ^ 1;
        if (arr1[mid1] > arr2[mid2]) {
            end1 = mid1;
            start2 = mid2 + offset;
        } else if (arr1[mid1] < arr2[mid2]) {
            start1 = mid1 + offset;
            end2 = mid2;
        } else {
            return arr1[mid1];
        }
    }
    return Math.min(arr1[start1], arr2[start2]);
}
}

```

## 在两个排序数组中找到第 $K$ 小的数

### 【题目】

给定两个有序数组  $arr1$  和  $arr2$ ，再给定一个整数  $k$ ，返回所有的数中第  $K$  小的数。

### 【举例】

$arr1=[1,2,3,4,5]$ ， $arr2=[3,4,5]$ ， $k=1$ 。

1 是所有数中第 1 小的数，所以返回 1。

$arr1=[1,2,3]$ ， $arr2=[3,4,5,6]$ ， $k=4$ 。

3 是所有数中第 4 小的数，所以返回 3。

### 【要求】

如果  $arr1$  的长度为  $N$ ， $arr2$  的长度为  $M$ ，时间复杂度请达到  $O(\log(\min\{M,N\}))$ ，额外空间复杂度为  $O(1)$ 。

### 【难度】

将 ★★★★★

### 【解答】

在了解本题的解法之前，请读者先阅读上一题“在两个长度相等的排序数组中找到上

中位数”这个问题的解答。本题也深度利用了这个问题的解法。以下的 `getUpMedian` 方法就是上中位数这个问题的代码，在 `a1[s1..e1]` 和 `a2[s2..e2]` 两段长度相等的范围上找上中位数。

```
public int getUpMedian(int[] a1, int s1, int e1, int[] a2, int s2, int e2) {
    int mid1 = 0;
    int mid2 = 0;
    int offset = 0;
    while (s1 < e1) {
        mid1 = (s1 + e1) / 2;
        mid2 = (s2 + e2) / 2;
        offset = ((e1 - s1 + 1) & 1) ^ 1;
        if (a1[mid1] > a2[mid2]) {
            e1 = mid1;
            s2 = mid2 + offset;
        } else if (a1[mid1] < a2[mid2]) {
            s1 = mid1 + offset;
            e2 = mid2;
        } else {
            return a1[mid1];
        }
    }
    return Math.min(a1[s1], a2[s2]);
}
```

下面开始求解本题，为了方便理解，我们用举例说明的方式。长度较短的数组为 `shortArr`，长度记为 `lenS`；长度较长的数组为 `longArr`，长度记为 `lenL`。假设 `shortArr` 长度为 10。`{1, 2, 3, ..., 10}` 依次表示 `shortArr` 的第 1 个数，第 2 个数……第 10 个数，注意，这个数字表示 `shortArr` 的第几个数的意思，并不代表值。假设 `longArr` 长度为 27。`{1', 2', ..., 27'}` 依次表示 `longArr` 的第 1 个数，第 2 个数……第 27 个数，注意，这个数字表示 `longArr` 的第几个数的意思，并不代表值。下面是找到整体第  $k$  个最小的数的过程：

情况 1，如果  $k < 1$  或者  $k > \text{lenS} + \text{lenL}$ ，那么  $k$  值是无效的。

情况 2，如果  $k \leq \text{lenS}$ 。那么在 `shortArr` 中选前面的  $k$  个数，在 `longArr` 中也选前面的  $k$  个数，这两段数组中的上中位数就是整体第  $k$  个最小的数。比如  $k=5$  时，那么 `{1...5}` 和 `{1'...5'}` 这两段数组整体的上中位数就是整体第 5 小的数。

情况 3，如果  $k > \text{lenL}$ 。举一个具体的例子来说，一共有 37 个数，求第 33 个最小的数 ( $33 > \text{lenL} = 27$ ) 就是这种情况。在 `{1...10}` 中，5 不可能成为第 33 个最小的数，因为即便是 5 比 27' 还要大。也就是说，即使 5 在 `longArr` 中把 27 个数全压在下面，5 在 `shortArr` 中也只把 4 个数压在下面，所以 5 最好的情况就是第 32 个最小的数。那么 `{1...4}` 就更不可能，所以 `{1...5}` 一律不可能。那么 6 可能是吗？可能。6 如果大于 27'，那么 6 就是第 33 个最小的数，直接返回，否则 6 也不是。同理，在 `{1'...27'}` 中，`{1'...22'}` 绝不可能是第 33 个最小的数。

23'如果大于 10, 那么 23'就是第 33 个最小的数, 直接返回, 否则 23'也不是。如果发现 6 和 23'有一个满足条件, 就可以直接返回。否则可以知道{1...6}和{1'...23'}这一共 29 个数都是不可能的, 那么{7...10}和{24'...27'}这两段数组整体的上中位数, 即这 8 个数里的第 4 小数, 就是整体第 33 个最小的数。

情况 4, 如果不是情况 1、情况 2 和情况 3, 说明  $\text{lenS} < k \leq \text{lenL}$ 。举一个具体的例子来说, 求第 17 个最小的数( $10 < 17 \leq 27$ )就是这种情况。在{1...10}中, 任何数都有可能是第 17 个最小的数。在{1'...27'}中, 6'不可能是第 17 个最小的数, 因为即使 6'在 shortArr 中把 10 个数全压在下面, 6'在 longArr 中也只把 5 个数压在下面, 所以 6'最好的情况就是第 16 个最小的数, 所以{1'...6'}一律不可能。在{1'...27'}中, 18'也不可能是第 17 个最小的数, 18'最好的情况也只能做第 18 个最小的数, 所以{18'...27'}一律不可能。只剩下{7'...17'}, 7'可能是吗? 可能。7'如果大于 10, 那么 7'就是第 17 个最小的数, 直接返回。否则 7'也是不可能的, 这时{1'...7'}这一共 7 个数都是不可能的, 那么{1...10}和{8'...17'}这两段数组整体的上中位数, 即这 20 个数里第 10 小的数, 就是整体第 17 个最小的数。

不管是以上 4 种情况的哪一种, 在求 arr1 和 arr2 长度相等的两个范围上的上中位数时, 范围最多也只是 shortArr 数组的长度, 所以时间复杂度为  $O(\log(\min\{M, N\}))$ 。具体过程请参看如下代码中的 findKthNum 方法。

```
public int findKthNum(int[] arr1, int[] arr2, int kth) {
    if (arr1 == null || arr2 == null) {
        throw new RuntimeException("Your arr is invalid!");
    }
    if (kth < 1 || kth > arr1.length + arr2.length) {
        throw new RuntimeException("K is invalid!");
    }
    int[] longs = arr1.length >= arr2.length ? arr1 : arr2;
    int[] shorts = arr1.length < arr2.length ? arr1 : arr2;
    int l = longs.length;
    int s = shorts.length;
    if (kth <= s) {
        return getUpMedian(shorts, 0, kth - 1, longs, 0, kth - 1);
    }
    if (kth > l) {
        if (shorts[kth - l - 1] >= longs[l - 1]) {
            return shorts[kth - l - 1];
        }
        if (longs[kth - s - 1] >= shorts[s - 1]) {
            return longs[kth - s - 1];
        }
        return getUpMedian(shorts, kth - l, s - 1, longs, kth - s, l - 1);
    }
    if (longs[kth - s - 1] >= shorts[s - 1]) {
        return longs[kth - s - 1];
    }
}
```

```

    return getUpMedian(shorts, 0, s - 1, longs, kth - s, kth - 1);
}

```

## 两个有序数组间相加和的 TOP $K$ 问题

### 【题目】

给定两个有序数组  $arr1$  和  $arr2$ ，再给定一个整数  $k$ ，返回来自  $arr1$  和  $arr2$  的两个数相加最大的前  $k$  个，两个数必须分别来自两个数组。

### 【举例】

$arr1=[1,2,3,4,5]$ ， $arr2=[3,5,7,9,11]$ ， $k=4$ 。

返回数组 $[16,15,14,14]$ 。

### 【要求】

时间复杂度达到  $O(k\log k)$ 。

### 【难度】

尉 ★★☆☆

### 【解答】

哪两个分别来自两个排序数组的数相加最大？自然是  $arr1$  的最后一个数和  $arr2$  的最后一个数，假设  $arr1$  长度为  $N$ ， $arr2$  长度为  $M$ ，如图 9-13 所示。

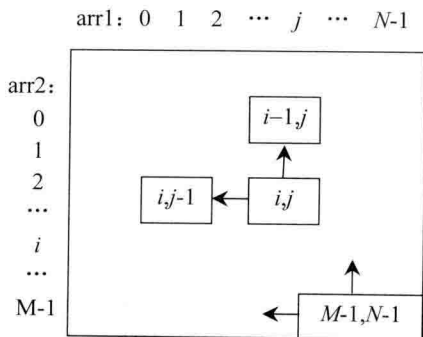


图 9-13