1. **比较两个有序数组的中位数**

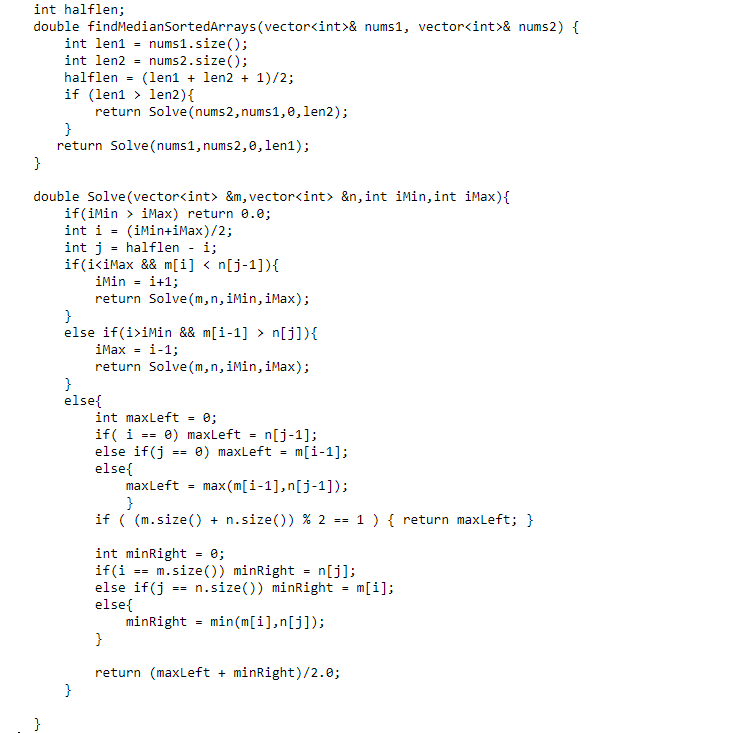
**题目描述**



**解法:**

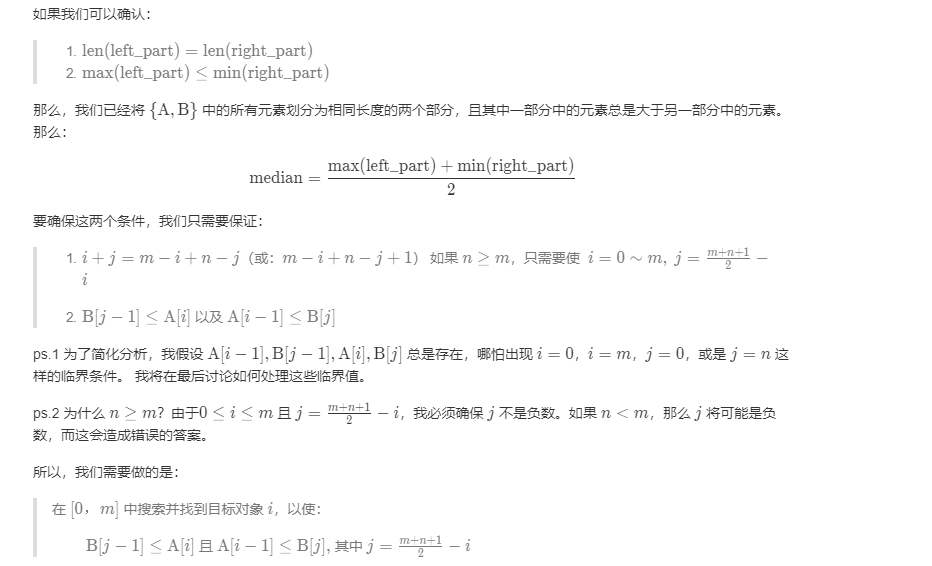
思考一下中位数的性质:如果一个数是某个数组的中位数，那么，在这个数组中，比这个中位数小的数和比这个中位数大的数，二者数目应该相等！

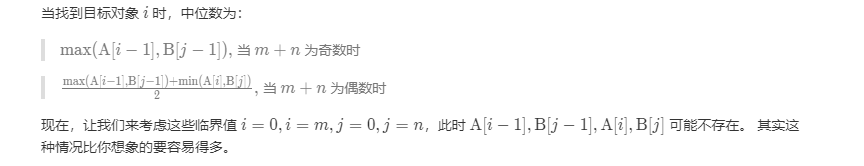
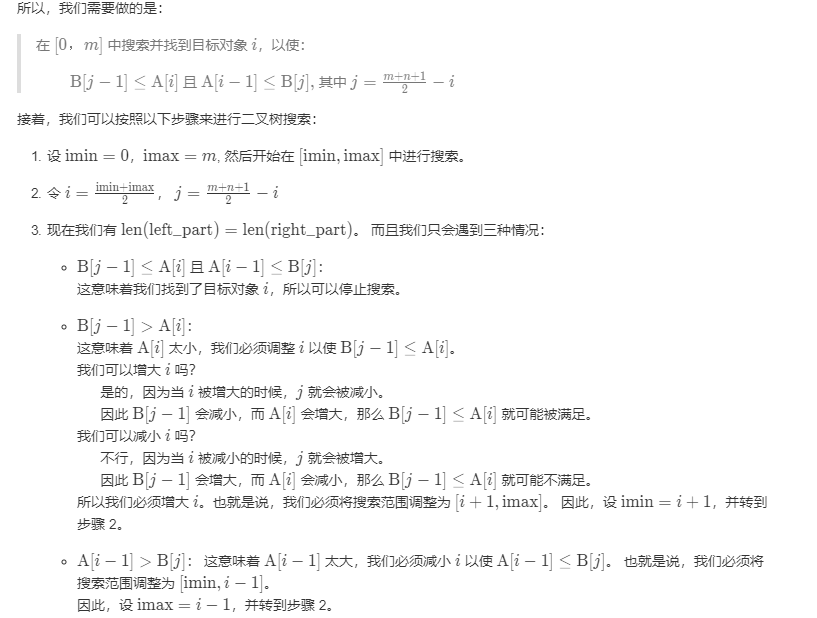
而题目中，两个数组已经排序，所以，可以用两个指针来分别指向两个数组。最后某个数是中位数的达成条件是:两个数组中，比某个指针小的数的数目和比某个指针大的数的数目相同，那么该指针所指即为两个数组的中位数。代码如下：



题解如下:

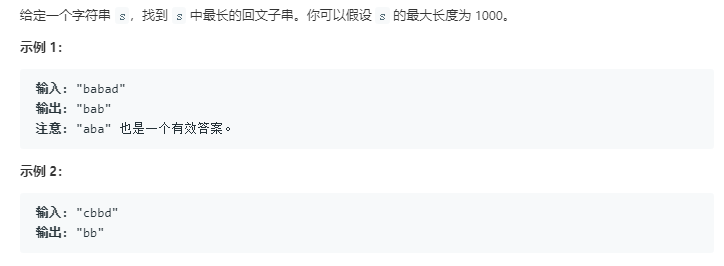






1. **最长回文字串**

**题目描述：**



**解法：**

利用动态规划的思想.设dp[i][j]表示下标i…j的字串是否是回文串。那么很容易可以得到一些结论:

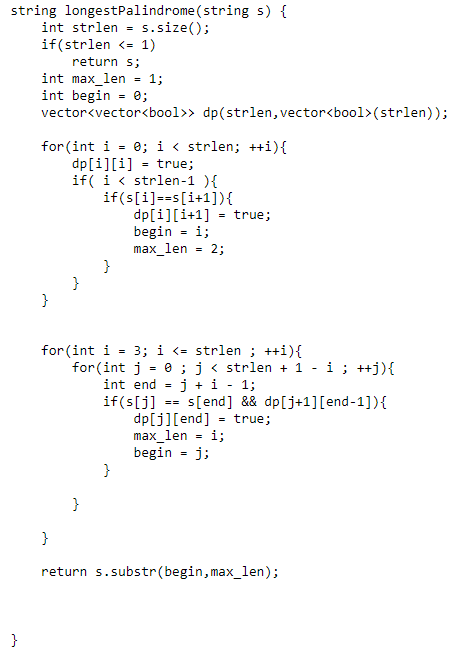
dp[i][i] = true;(for each i)

dp[i][i+1] = s[i] == s[i+1]? true : fasle;

以上是长度为1和2的字串的情况，那么从size = 3开始遍历其他字串，有如下转移方程：

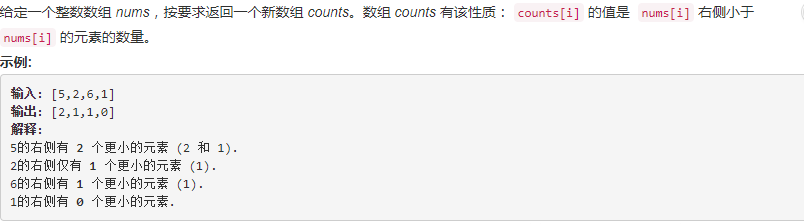
dp[i][j] = (s[i] == s[j] && dp[i+1][j-1]) ? true : false;

代码如下:



1. **计算右侧小于当前元素的个数**

**题目描述:**



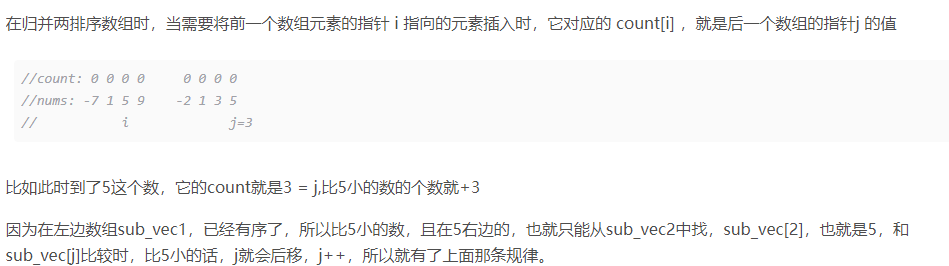
**解法1:利用二叉搜索树**

二叉搜索树具有左孩子比当前节点小，右孩子比当前节点大的特性，所以只需要在每个节点维护一个count，用于记录其左孩子的数目即可。

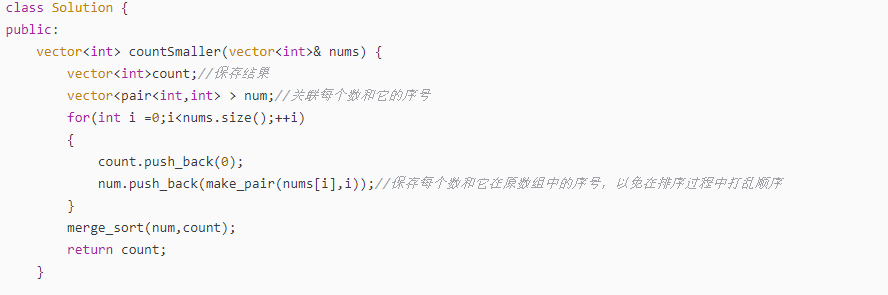
这样的话，可以从数组的尾部反向遍历，依次插入结点，如果需要将某个值插入到当前结点的左侧，就让本结点的count+1;如果插入到右结点，就让res = count + 1；

**解法2:利用归并排序**

在归并排序中，每次归并操作分为前后两个数组，那么显然，前数组中的数原本都是位于后数组中的数的右侧的。因此，在归并时，加入当前指针指向数组的下标为i，而后数组的下标指到了j处，说明，对于下标为i的元素来说，后数组指针j之前的所有数都比i对应的数来的小，所以对于i对应的数而言，其右侧比它小的数要再加上j个。如下所示：



**代码见下：**







**4、至少有K个重复字符的最长子串**

**题目描述**



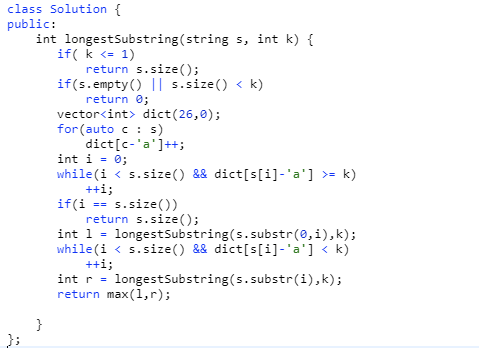
**解：**

不妨这么处理此问题:首先从字串起始位置开始遍历到其末尾，用字典(dict)记录每个字幕的出现次数。然后再遍历一次，找到第一个不满足”出现频率不少于k”的字符，令其位置为x。

显然任何包含x位置的字符串都不可能满足题目需求，所以将其拆分为两个部分即:

x左侧子串、x右侧字串。那么就相当于说，这个最长字串一定位于x左侧或者右侧，我们递归地求解问题，先对x的左侧字串进行递归。然后再对x的右侧进行递归。

值得注意的是，这里存在可剪枝的条件:即x右侧字串应当从第一个满足出现频率不小于k的字符处开始。而x左侧的字符显然都满足这个条件，所以左侧无需额外的剪枝操作。

代码如下:

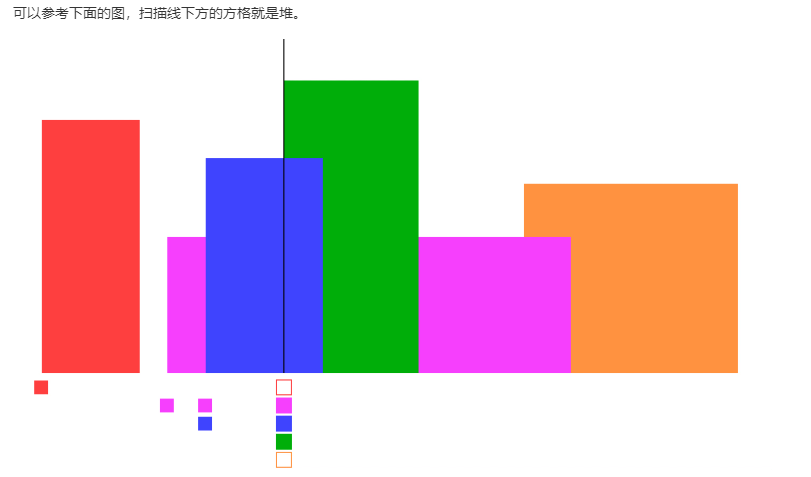
**5、天际线问题**

**题目描述**

见：<https://leetcode-cn.com/problems/the-skyline-problem/>

解:

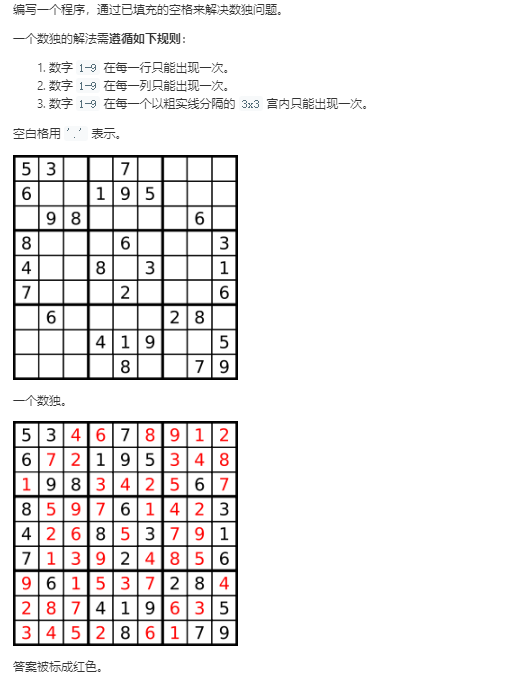
使用扫描线，从左至右扫过。如果遇到左端点，将高度入堆，如果遇到右端点，则将高度从堆中删除。使用 last 变量记录上一个转折点。如下图所示



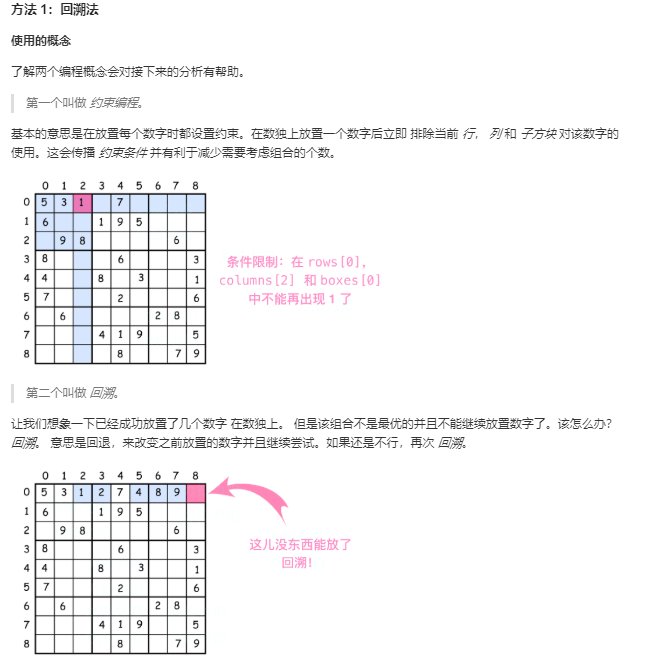
代码：

**6、解数独**

**题目描述**



**解法：本体解法和N皇后类似？**





***代码见下：***



