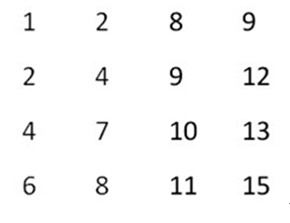
这几天在Iteye看到有个有奖问答的题目,二维数组的二分查找。想想最近也没什么事情，就做了一下。自己的算法基础很是薄弱，所以也参考了一些网上的东东。那个有奖什么就不参加了。哈哈。。。。   
  
    原题如下(估计很多人知道了)：   
    在一个二维数组中，每一行都按照从左到右递增的顺序排序，每一列都按照从上到下递增的顺序排序。请完成一个函数，输入这样的一个二维数组和一个整数，判断数组中是否含有该整数。   
例如下面的二维数组就是每行、每列都递增排序。如果在这个数组中查找数字7，则返回true；如果查找数字5，由于数组不含有该数字，则返回false。   
   
  
   我这里主要讲下我当时是怎么做的http://lsxjl.iteye.com/images/smiles/icon_redface.gif 。   
   我的第一印象如果想快速的查找的话应该要利用到对角线。因为对角线元素在他所在的矩阵中最大的，可以在一定程度缩小查找范围。但当时没有具体想清楚该怎么做。后来又想到题目中没有指定数组是N\*N的，所以先放弃了这个想法。然后继续想想，如果面试官问我这个题目的话该怎么做。   
  
   首先当然是保证不错,二重遍历是肯定不会错的。我都一个一个比较了还会有什么问题呢。当然这个就跟别人问你用什么方法排序的，你回答冒泡排序一样，没什么技术含量。不过至少说明你了解了题目的内容。   
   再深入一下，二维数组是数组的数组。一般对于有序数组可以采用二分法对数组进行一个lgn的查找。那这样可以对二维数组的每个数组进行一次二分查找。这样就比较不错了，知道了二分查找。估计可以打个及格的分数了。  
   继续深入，可能就会像我一样想到了利用对角线。但是这个要考虑到不是N\*N的情况。利用对角线对矩阵分块，然后就可以排除一些数据了。类如上图中，查找元素 7.跟对角线元素比较之后, 4 < 7 < 10, 那就说明，4左上角，10右下角的数据不用处理。就减少了一半的数据。这篇文章很清楚的说明了这种方法应该怎么做。<http://justjavac.iteye.com/blog/1310178>   
  
   做到这一步以后，我是没什么办法了。如果面试官问我，还有更好的办法没。我就只能拜拜了。不过，好的方法总是有的，只要好好的寻找。我在网上搜索的时候发现了杨氏矩阵查找的方法。<http://blog.csdn.net/michealmeng555/article/details/2489923>   
   杨氏矩阵查找跟对角线的方法差不多，但是不是用的主对角线\,而是用的副对角线/. 在副对角线中的元素有个性质。在以他为左右两个顶点的矩阵中，他是一个数组中的最大值，是另外一个数组中的最小值。那我们可以利用这个性质，假设我们选择最右上角的元素 a。那么对于需要查找的元素b，如 a > b 向左走，如果 a < b向下走。直到到达最左下角结束。真是太精妙了。