所谓偏序就是当你知道元素A,B,C时，元素A
在ACM中，这种类型的题目通常会告诉我们n个数组，每个元素的各属性对应于不同数组的同一位置的值
询问通常是回答比这元素小的有几个，或者是LIS这样的dp问题。

而解决偏序问题通常有以下方法：排序，数据结构(树状数组，线段树，平衡树)，cdq分治，分块。

接下来简单介绍关于维数不同的偏序该采用什么策略。
一维：这个其实不能叫做偏序，一维是全序的，这种情况只要直接排序就可以解决，当然使用数组结构也可以。
二维：先对第一维排序，然后第二维可以用cdq分治，也可以使用数据结构维护。
三维：同上，第一维要排序，然后可以两重cdq分治，cdq分治+数据结构，线段树或树状数组套平衡树
四维：一维排序，然后两重cdq分治+数据结构，或者cdq分治+线段树或树状数组套平衡树
五维：一维排序，然后两重cdq分治+线段树或树状数组套平衡树

以上方法的效率基本是每上一维多一个log,但是空间基本都是nlog（n）的，数据大的话到五维可能就超时了
关于不同方案间的优劣的话，排序是必须的，直接调用c++的快排即可，数据结构之间树状数组是最值得使用
的，常数小而且代码短，cdq分治在时间方面要慢于树状数组，但是数据结构通常只能维护一维，平衡树的效率
根据不同的写法不好直接评价，所以通常考虑使用排序+cdq分治+树状数组的方案，简单而且高效，必要的时候
可以考虑用平衡树。而关于树状数组套线段树的方法，其实那就是主席树的在线写法，但是这个方法容易mle

关于cdq分治，这里必须要讲讲我的理解了，研究cdq分治研究了好久，终于算是搞懂了其主要思想。
进行cdq分治之前必须要排序，因为分治的思想一定是有顺序的，也就是说右边的不能够影响到左边
这里以三维的偏序为例子，有n个元素，每个元素有x,y,z三种属性，只有当a.x<=b.x&&a.y<=b.y&&a.z<=b.z的时候
a

首先，分治的前提，先对x维度进行排序，同时要以y和z分别为第二三关键字，然后先把相等的情况处理掉，
然后进行分治，分治的过程就是先把一整段分成左右两部分，也就是二分，然后左右两部分内部的问题通过继续分治解决
当前分治要解决的就是左边全部元素对于右边的影响。而对于按x排序的元素来说，左边最大的x也一定小于右边最小的x
那这个时候，把左边按y排序，右边也按y排序，我们只要用两个指针，就可以确定对应于右边每个元素，左边
y元素比它小的有几个了，而这个时候只要对于z，用一个树状数组来统计，把左边元素的z值插入树状数组，然后在右边
的元素通过树状数组统计小于它的z值的有几个，那么问题就解决了。

然后是如何进行二重cdq分治，我觉得这才是cdq分治的核心，直接的cdq分治很好理解，但是再往后加一层的时候难度就加大了
为什么cdq分治可以反复嵌套，这个问题我想了很久，直到后来，我想通了，其实cdq的分治从某种意义上讲也是一种分块的思想
只不过分治每次分两块而已。而且cdq分治做的事情其实就是排序一维+分治+排序二维+分治+排序三维这样的循环过程。
也就是先对于x排序，然后分治，对于每个分治的过程，因为要计算的是左边对于右边的影响，那么要对于元素的所属打个标记来
区分，然后再把分治的区间按y排序，再进行第二次分治，第二次分治的过程中对区间进行z排序，然后按x的标记来统计产生的影
响就可以直接用排序来代替树状数组的对于z维的计算。也就是说对于四维

的状态只要再加上树状数组就ok了。

这里不得不说的是网上流传的cdq分治，多数是先左边然后第二次分治然后再右边的，然而其实并没有什么差别，分治本身不会
受到影响，cdq本身有常数问题，每次的排序其实是比较耗时的，但是网上流传的版本多数是直接调用sort，我觉得其实没必要
cdq分治本身就是二分的过程，那么直接在二分的时候进行归并排序不就好了吗，这样应该能减少一点时间的损耗。

然后是分块的做法，这种做法其实就是通过部分暴力来缩小数据范围然后来使用原数据范围不能用的方法来解决问题，
对于这题来说，同样对第一维排序，把y和z单独排序，然后从左到右分成k块，k通常选择sqrt(n)，有的时候可以根据题目
而改变，对于每个块内做法是直接暴力，而块与块之间考虑使用算法优化，对于当范围缩小以后，每个元素的y和z分别对应
与各自的块，然后就可以使用二维树状数组来优化这个前缀和，这样就可以快速统计整个问题了。
hihocoder上有一道五维的偏序，就是可以用分块然后bitset优化解决的。

上述方案的实现在另一篇hdu5618的文章里有完整的ac代码戳我看代码