**树状数组小结**

树状数组按照维数分为1维和2维，按照类型分可以分为插点问段和插段问点。这两个互逆，只是改变for循环的方向就可以了，但是内涵的内容真的是太多了。。。。

向上for循环，是为了访问比我大的元素，用于插点问段时的插入和插段问点时的查询

向下for循环，是为了访问比我小的元素，用于插点问段时的查询和插段问点时的插入

另外对于lowBit的应用及原理，我想用一个词可以比较准确的描述：管辖

树状数组用做统计很好用，假如y = f(x)，事先应该考虑好是把y作为下标，还是x作为下标。

下面转自：<http://hi.baidu.com/lilu03555/blog/item/4118f04429739580b3b7dc74.html>

hoj 1867 经理的烦恼  
此题是最基本的一维树状数组题目，直接进行简单的加一减一（通过判素）操作即可。

hoj 2430 Counting the algorithms  
从后往前不断删除（这样的话不存在区间包含问题），统计相同元素区间内数的个数。之后将这两个元素一起删除。（其实从前往后也行啊。。。）

hoj 1640 Mobile Phone  
该题目是典型的二维树状数组的题目。

hoj 2275 Number sequence  
题目就是统计序列中Ai < Aj > Ak（i < j < k）的个数，可以从前往后统计每个元素之前小于它的数的个数，在从后往前统计每个元素之后小于它的数的个数。然后乘积加和即可。注意树状数组统计是起始为1。

poj 2352 Stars  
相当经典的树状数组题目，一开始分析题目是第一感觉是二维的树状数组，不过数据范围显然不容许的，可先排序，然后再统计每个位置之前的星星的个数。

poj 3067 Japan  
与stars极其相似，唯一的不同是上题诗统计之前的个数，而这个统计之后的个数，当然我们可以用当前总数i减去之前的数即可得到。此题bt之处在于不能用long long 只能用\_\_int64。

poj 2155 Matrix  
此题是二维树状数组基本应用的变通，不像hoj1640简单的插入统计，而此题是对一个二维区间数值翻转（0变1，1变0），最后询问某处的值。所以可以使用奇偶性判断，这样我们可以使用树状数组在这个二维区间内加，就相当于翻转，最后统计某点翻转的次数。（插段问点）

poj 3321 Apple Tree  
首先进行DFS遍历一遍，对每个节点编号（进入递归时begin[i],出递归时end[i]，具体见《算法导论》），由于子树的编号是连在一起的，同时包含在子树的的根begin[root]和end[root]之间，所以以后可以直接利用树状数组对其内容修改和统计。（这个的思想很强。。。。遍历的时候可以将树的许转为线性序。。。）

hoj 2098 poj 2299 Ultra-QuickSort  
该题是求逆序数的题目，当然我们可以使用合并排序法，在此介绍一种用树状数组解决的方法，由于元素的范围巨大，可以先采用离散化（就是排序编号），然后依次查值，并统计当前比自己大的元素的个数（当前总数-小于等于自己），即可得到结果，和以上的Japan有着相似的处理方法。

补充：

POJ-2763

也是树状数组+欧拉序列（和3321一样），并且用到了LCA（一开始认为不用，但确实得用），不过AC的好辛苦。。。。。。。确实是一道很好的题！！！！

转自：http://hi.baidu.com/wegnahz/blog/item/3f6080c0082a59110ff47751.html

从上午11点一直wa到下午4点多。。。而且是因为一个bug。。。

这题就是给你一棵树，边上有权，然后维护两个操作，一个是问你从一个点走到另一个点的距离，另一个是改变一条边的权值。一开始想到很多 莫名其妙的算法，后来发现用dfs序列就很好。。。按照dfs序列构造一个树状数组，对于点i,begin[i]值为i到父节点的 权，end[i]=-begin[i]，这样任意一个点到根节点的路径长就是getsum(begin[i])。这样i，j路径长度就是 getsum(begin[i])+getsum(begin[j])-2\*getsum(begin[lca(i,j)])，求lca方法有rmq和 tarjan，因为我半秃只会用tarjan，所以写的非常麻烦（要离线做，开n个数组）。。。这同时导致了杯具的发生。tarjan中一个<=号 我打成了<，这样如果求i和i的lca，返回0.。。。。。。

另外，这题递归层数好像挺深。。。vc的栈还真挺大。还有就是只要能在线做的尽量在线做。。。离线要开的数组太多了。。。脑子都晕了

POJ-1656

水题，直接朴素就可以，二维树状数组可以加一些优化。。。不过是插段问段的，所以必然有一个（插入或者是查询）需要n^2效率

POJ-1990

很经典的一道题，数轴上有n个牛，每个牛在一个坐标x[i]上，并且有一个值v[i]，要求计算所有的1<=i<j<=n的abs(x[j]-x[i]) \* max(v[i], v[j])的值。。

先按照v进行升序排序，然后遍历，用两个树状数组统计出小于（大于）x[i]的牛个数and小于（大于）x[i]的牛的x之和，这样可以快速统计出来。。。

POJ-2481

数轴上n个牛，their favourite clover range is [Si, Ei] and [Sj, Ej]. If Si <= Sj and Ej <= Ei and Ei – Si > Ej – Sj, we say that cowi is stronger than cowj.  
统计出对于第i个牛，比他大的牛的个数。。。。

按照s排序，然后遍历（这时前面的x必然小于等于当前的x），然后统计E大于当前E的个数。。。边界情况特殊处理，

POJ-3378

在一个序列中统计：

1. 1 ≤ *i* < *j* < *k* < *l* < *m* ≤ *N*  
2. *Ai* < *Aj* < *Ak* < *Al* < *Am*

的个数。。

有点像hoj2275，以中间元素为轴，分别统计两边，但由于两边各有两个，因此需要用两个树状数组来统计。。。。

注意最后要用到高精度。。。囧。。

POJ-2182

1…N个数的某种序列，第i个数为arr[i]，前面比他小的数的个数为val[i]。已知val序列，求arr序列。。。。

从后向前遍历，用树状数组统计，中间加入二分查找。。。效率n\*lgn\*lgn。

POJ-3416

star的升级版。。。。两个树状数组就可以搞定了。。。