**一维树状数组**

这里只讨论一维情况  
\*对于序列a，我们设一个数组C定义C[t] = a[t – 2^k + 1] + … + a[t]，k为t在二进制下末尾0的个数。  
\*一般对出现0开始的数据都会全部往后推1  
\*求出2^p(其中p为x的二进制表示中最右边的那个1的位置)，如6的二进制表示为110，最右边的1为1  
\*一个强调：在此先声明 做树状数组的时候一定要切记避免0的出现去参与lowbit的操作 可以知道这个会造成add和calsum的死循环

[View Code](javascript:void(0);) C

|  |
| --- |
| int lim;*//上界*  LLD C[N];*//树状数组的存储结构*  int lowbit(int t)  {  return t&(-t);  } |

这样写都可以  
return t & ( t ^ ( t – 1 ) );  
return t-(t&(t-1));

[View Code](javascript:void(0);) C

|  |
| --- |
| *//对某位进行操作函数如下(以加法为例)*  void add(int a,int b)  {  while(a<=lim)  {  C[a]+=b;  a+=lowbit(a);  }  }  <!--more-->  *//求1 -- a和的函数代码如下:*  LLD calsum(int a)  {  LLD cnt=0;  while(a>0)  {  cnt+=C[a];  a-=lowbit(a);  }  return cnt;  } |

//读 取 某 个 位 置 的 值 f[i]：  
//一 般 我 们 可 以 用 sum[i]-sum[i-1]， 根 据 此 结 构 还 可 以 继 续 优 化 。 即 计 算 两 个 sum 的 时 候 ， 它 们 的 路 径 可 能 会 相 遇 。

[View Code](javascript:void(0);) C

|  |
| --- |
| int gsum(int t)  {  int sum=C[t];  if(t>0)  {  int z=t-(t&-t);  t--;  while(t!=z)  {  sum-=C[t];  t-=(t&-t);  }  }  return sum ;  } |

树 的 所 有 元 素 \* (乘)一 个 常 数 因 子 ， 可 以 通 过 两 种 方 式 实 现

[View Code](javascript:void(0);) C

|  |
| --- |
| void scale(int c)  {  for(int i=1;i<=MaxVal;i++)  add(-(c-1)\*gsum(i)/c,i);  }  void scale(int c)  {  for(int i=1;i<=MaxVal;i++)  C[i]=C[i]/c ;  } |

\*再次强调：在此先声明 做树状数组的时候一定要切记避免0的出现去参与lowbit的操作 可以知道这个会造成add和calsum的死循环

**\*逆序对的应用：  
\*快速统计区间和  
\*快速计算出一个序列比当前值小的数有几个  
(由此还可以延伸出比如还计算出前面比之小的数的总和poj 1190 MooFest等等)  
\*还要注意是求比当前小的还是比当前小于等于的数的个数 具体有hoj 2275 Number sequence**

**poj 2299 Ultra-QuickSort 树状数组求解逆序对数  
逆序对是指：  
设 A 为一个有 n 个数字的有序集 (n>1)，其中所有数字各不相同。  
如果存在正整数 i, j 使得 1 ≤ i < j ≤ n 而且 A[i] > A[j]，则 这个有序对称为 A 的一个逆序对。  
例如：数组 <2,3,8,6,1> 的逆序对为：<2,1> <3,1> <8,6> <8,1> <6,1> 总数共5个。**

**归并排序可以求解逆序对 树状数组也可以：  
方法：  
比如9 1 0 5 4  
那么离散化后就是5 2 1 4 3  
对这个数组进行如下操作**

[**View Code**](javascript:void(0);) **C**

|  |
| --- |
| for(i=1;i<=n;i++)  {  a=ans[i];*//获取离散后的值*  add(a,1);*//对该值的这个位置加1*  tmp=calsum(a);*//统计到这个位置的总和，正常没有出现逆序对的话那么加到第几个数的就是几*  cnt+=i-tmp;*//i表示第几个数 (由于逆序对产生的差)*  } |

[解题报告](http://blog.foreverzeus.com/?p=837)

poj 2352 starts 经典题目 题目给出星星的坐标 主要y是升序给出的

经典题目 题目给出星星的坐标 主要y是升序给出的  
等级的定义是x和y都比太小的星星的个数  
另外该题是输出每种星星等级的个数  
显然可以用树状数组做  
如果每次加的数都是1  
对于当前值进行一次calsum()就可以知道前面加了多少个1 显然也就是在此签前面有多少个数

[解题报告](http://blog.foreverzeus.com/?p=840)

poj 3067 Japan 树状数组  
日本为了举办ACM，新建了许多路，现在给你东西两个海岸城市相连的情况，  
求在这样全部的线连起来后有多少个交叉点。  
3 4 4  
1 4  
2 3  
3 2  
3 1  
该题是逆序对的另一种问法而已  
对于给出的对如果会发生相交肯定是由于存在  
a b  
c d  
a和d相连c和b相连 既是a<C&&B<D才会出现交叉  
可以把y坐标先排序 那么该题就可以变成对x坐标分析  
若x前面有m个与x成逆序的数就有m个交叉 题目就变成求排好y序后x里面的逆序对的个数(这样就跟poj2299一样了)

poj 2481 Cows树状数组  
给出n只牛的两个值[Si, Ei]；  
若存在这么一对直p1(a,b)和p2(c,d)  
若a<C&&D**(d-c)  
那么就说p1比p2stronger  
现在给出n对数输出第n只牛(比他自己stronger的牛的个数)  
对给出的区间  
x1 y1  
x2 y2  
..  
xn yn  
可以对y先降序排序故考虑牛i的时候y条件就可以省略(在其前面y都比他大)x按升序排列  
只需考虑x的值 若x与前面的相等那么那么当前的答案就和前面一个的一样(区间刚好相等不算)  
若不是只要算出该位置前有多少比他小的数就可以了（即满足了条件）  
这个过程可以用树状数组快速求解**[**解题报告**](http://blog.foreverzeus.com/?p=842)

**hoj 2275 Number sequence  
在一个序列中需找满足这样条件的数的个数  
for three number Ai, Aj, Ak, which satisfy that Ai < Aj > Ak and i < j < k.  
对于本题很明显是找出一个数计算左边比他小的数的个数resa和右边比他小的数的个数resb  
那么答案就是resa+resb 故其实用两个树状数组就能很快求出来  
注意一个细节问题就：Ai < Aj > Ak 是不能包括等于的情况一般直接用树状数组求出来的是带等于的情况  
这时候只要处理一下add对要加的值多加一个1就好了  
res=calsum(p[i]);  
add(pa[i]+1,1);**

[**解题报告**](http://blog.foreverzeus.com/?p=844)

**poj 1190 MooFest  
有n只牛 每只牛都有两个值 一个vol和一个pos  
每只牛与另一头牛交流所需要的最小值为max（vol1，vol2）\*之间的距离  
问每每两头牛交流时这个值最少是多少！  
用一个结构体存储数据 按vol排序  
用两个树状数组  
cnum[N]记录了比当前pos小的的牛的个数  
cloc[N]记录了比当前pos小的牛的pos之和**

[**解题报告**](http://blog.foreverzeus.com/?p=277)

**hdu 3015 Disharmony Trees  
用两个树状数组 根据poj1190 稍微该下就可以  
该题之间的距离是乘以两颗数中的高度的最小！！**[**解题报告**](http://blog.foreverzeus.com/?p=277)

***本文地址:*** [***http://blog.foreverzeus.com/?p=853***](http://blog.foreverzeus.com/?p=853)