**树状数组学习系列2 之 OJ题目大汇总——czyuan原创**

2009年08月13日 星期四 19:28

|  |
| --- |
| 上一篇我们对树状数组进行了一些分析(详见树状数组学习系列1 之 初步分析——czyuan原创<http://hi.baidu.com/czyuan_acm/blog/item/49f02acb487f06f452664fbc.html>），这篇主要是对各大OJ有关树状数组的题目进行汇总。        先提个注意点，由于Lowbit(0) = 0，这会导致x递增的那条路径发生死循环，所有当树状数组中可能出现0时，我们都全部加一，这样可以避免0带来的麻烦～～        简单：  **POJ 2299　Ultra-QuickSort**       <http://acm.pku.edu.cn/JudgeOnline/problem?id=2299>       求逆序数，可以用经典的归并排序做，也是基本的树状数组题目。  **POJ 2352 Stars**       <http://acm.pku.edu.cn/JudgeOnline/problem?id=2352>       题目意思就是求每个星星左下方的星星的个数，由于y轴已经排序好了，我们可以直接用按x轴建立一维树状数组，然后求相当于它前面比它小的个数，模板直接一套就搞定了～～  **POJ 1195 Mobile phones**       <http://acm.pku.edu.cn/JudgeOnline/problem?id=1195>       二维的树状数组，直接把Update()和Getsum()改为二维即可。       如Update()函数改为：       void Update(int x, int y, int d)       { // 注意：当i = 0，0 + Lowbit(0) = 0，会造成死循环!           int i, j;           for (i = x; i < maxn; i += Lowbit(i)) // 注意这里是maxn，是tree[]的大小.           {                 for (j = y; j < maxn; j += Lowbit(j))                 {                     tree[i][j] += d;                 }           }        }  **POJ 2481 Cows**        <http://acm.pku.edu.cn/JudgeOnline/problem?id=2481>        将E从打到小排序，如果E相等按S排序，然后就跟POJ 2352 Stars做法一样了～～  **POJ 3067 Japan**        <http://acm.pku.edu.cn/JudgeOnline/problem?id=3067>        先按第一个坐标排序从大到小排序，如果相等按第二个坐标从大到小排序，然后就又是跟Cows和Stars做法相同了...  **POJ 2029 Get Many Persimmon Trees**         <http://acm.pku.edu.cn/JudgeOnline/problem?id=2029>         O(n ^ 2)枚举起点，再用二维树状数组求其中的点数即可。  **HOJ 2275 Number sequence**        <http://202.118.224.210/judge/show.php?Contestid=0&Proid=2275>        两个一维树状数组，分别记录在它左边比它小的和在它右边比它大的即可～～  **HOJ 1867 经理的烦恼**        <http://202.118.224.210/judge/show.php?Contestid=0&Proid=1867>        先筛法求素数，然后如果从非素数改变成素数就Update(x, 1)，如果从素数改变成非素数就Update(x, -1)即可。  **Sgu 180 Inversions**        <http://acm.sgu.ru/problem.php?contest=0&problem=180>        经典树状数组 + 离散化。注意结果要用long long～～  **SPOJ 1029 Matrix Summation**        <https://www.spoj.pl/problems/MATSUM/>        基本的二维树状数组...       中等：  **POJ 2155 Matrix**       <http://acm.pku.edu.cn/JudgeOnline/problem?id=2155>       经典树状数组题目，分析见前一篇文章（树状数组学习系列1 之 初步分析——czyuan原创）～～  **POJ 3321 Apple Tree**      <http://acm.pku.edu.cn/JudgeOnline/problem?id=3321>       这题的难点不在于树状数组，而是如果将整棵树映射到数组中。我们可以用DFS()改时间戳的方法，用begin[i]表示以i为根的子树遍历的第一个点，end[i]表示以i为根的子树遍历的最后一个点。       比如数据为：       5       1 2       2 5       2 4       1 3       那么begin[] = {1, 2, 5, 4, 3}, end[] = {5, 4, 5, 4, 3}，下标从1开始。       对于每个点都对应一个区间(begin[i], end[i])，如果要改变点a的状态，只要Update(begin[a])，要求该子树的苹果树，即Getsum(begin[a] ) - Getsum(end[a] + 1)，(注：这里求和是求x递增的路径的和。)  **POJ 1990 MooFest**       <http://acm.pku.edu.cn/JudgeOnline/problem?id=1990>       这题的难点是要用两个一维的树状数组，分别记录在它前面横坐标比它小的牛的个数，和在它前面横坐标比它小的牛的横坐标之和。       按音量排个序，那么式子为：       ans += 1LL \* cow[i].volumn \* (count \* x - pre + total - pre - (i - count) \* x);       cow[i].volumn为该牛的能够听到的音量。       count为在第i只牛前面横坐标比它小的牛的个数。       pre为在第i只牛前面横坐标比它小的牛的横坐标之和。       total 表示前i - 1个点的x坐标之和。       分为横坐标比它小和横坐标比它大的两部分计算即可。  **Hdu 3015 Disharmony Trees**       <http://acm.hdu.edu.cn/showproblem.php?pid=3015>       跟上题方法相同，只要按它的要求离散化后，按高度降序排序，套用上题二个树状数组的方法即可。  **HOJ 2430 Counting the algorithms**       <http://202.118.224.210/judge/show.php?Contestid=0&Proid=2430>       这题其实是个贪心，从左往右或者从右往左，找与它相同的删去即可。先扫描一遍记录第一次出现和第二次出现的位置，然后我们从右到佐，每删去一对，只需要更改左边的位置的树状数组即可，因为右边的不会再用到了。  **tju 3243 Blocked Road**       <http://acm.tju.edu.cn/toj/showp3243.html>       这题主要在于如果判断是否连通，我们可以先用j = Getsum(b) – Getsum(a – 1)，如果j等于(b – a)或者Getsum(n) – j等于(n – (b – a))，那么点a, b联通。  **SPOJ 227 Ordering the Soldiers**      <http://www.spoj.pl/problems/ORDERS/> 这题与正常的树状数组题目正好想反，给定数组b[i]表示i前面比a[i]小的点的个数，求a[]数组。 我们可以先想想朴素的做法，比如b[] = {0, 1, 2, 0, 1}，我们用数组c[i]表示还存在的小于等于i的个数，一开始c[] = {1, 2, 3, 4, 5}，下标从1开始。      我们从右向左扫描b[]数组，b[5] = 1，说明该点的数是剩下的数中第4大的，也就是小于等于它的有4个，即我们要找最小的j符合c[j] = 4(这里可以想想为什么是最小的，不是最大的，挺好理解的)，而c[]是有序的，所以可以用二分来找j，复杂度为O(logn)，但现在问题是每次更新c[]要O(n)的复杂度，这里我们就想到树状数组，c[i]表示还存在的小于等于i的个数，这不正好是树状数组的看家本领吗～～所以处理每个位置的复杂度为O(logn \* logn)，总的复杂度为O(n \* logn \* logn)。  **hdu 2852 KiKi's K-Number**      <http://acm.hdu.edu.cn/showproblem.php?pid=2852>      这题与上面那题类似，只是要求比a大的第k大的数，那我们用Getsum(a)求出小于等于a的个数，那么就是要我们求第k + Getsum(a)大的数，而删除操作只要判断Getsum(a) – Getsum(a - 1)是否为0，为0则说明a不存在。       难题：  **POJ 2464 Brownie Points II**       <http://acm.pku.edu.cn/JudgeOnline/problem?id=2464>        http://hiphotos.baidu.com/czyuan_acm/pic/item/89093cdd649f431d5882dd54.jpg       这道题用二分也可以做的，这里介绍下树状数组的做法。首先有n个点，过每个点可以做x,y轴，把平面切成BL, TL, TR, BR四个部分，我们现在的问题是如果快速的计算这四个部分的点的个数。       这样我们可以先预处理，先按y坐标排序，求出每个点正左方和正右方的点的个数LeftPoint[], RightPoint[]，复杂度为O(n)，同样我们再以x坐标排序，求出每个点正上方和正下方点的个数UpPoint[], DownPoint[]。还要求出比点i y坐标大的点的个数 LageY[]。注意：这里要进行下标映射，因为两次排序点的下标是不相同的。       然后按x坐标从小到大排序，x坐标相等则y坐标从小到大排序。我们可以把y坐标放在一个树状数组中。 对于第i个点，求出Getsum(y[i])即为BL的个数，然后Update(y[i])。由于现在是第i点，说明前面有i – 1个点， 那么      TL = i - 1 - LeftPoint[i]- BL;      TR = LargeY[i] - TL – UpPoint[i] ;       BR = n - BL - TL - TR - LeftPoint[i] - RightPont[i] - UpPoint[i] – DownPont[i] - 1;      这样我们就求出四个部分的点的个数，然后判断有没有当前解优，有的话就更新即可～～  **UVA 11610 Reverse Prime**    <http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&page=show_problem&category=78&problem=2657&mosmsg=Submission+received+with+ID+7313177>      一道很综合的树状数组题，用到了树状数组中的很多知识点，包括离散化，二分查找等。      1. 先按题目要求筛法素数，找到所有的Reverse Prime。      2. 将这些Reverse Prime离散化，只有78500个左右。树状数组中tree[i]记录比i小的点的个数。 当执行q a操作时，二分查找最小的j, 使得Getsum(j) 等于 ++a(因为a可能为0，所以统一加一)。这步与上面SPOJ 227 Ordering the Soldiers 类似。       3. 当执行d a操作时，先找到a离散化后的值b，然后Update(b, -1)即可。      按这样做后，运行时间为: 0.3s多，感觉很诧异，因为都是0.1s以下的。这里特别感谢liuzhe大牛的指点，其实题目中的Reverse Prime是由10^6以下的素数倒置得到的，那么得到的要求是7位，最后一位一定是0，我们可以对每个除以10处理，那么数的范围就减小了10倍，速度就提高了不少。       最后自己又加了点优化，跑了0.056s，排在第3名～～      3 7313367 czyuan 0.056 C++  czyuan原创，转载请注明出处。 |