树状数组，小结\_（优雅的数据结构）

2009年09月26日 21:03

|  |
| --- |
| 又做了几道树状数组的题，决定放一块儿总结一下；恩，总结一下。。  （ps：大牛可以直接跳过。。。）  这得从一张图说起；  http://hiphotos.baidu.com/rain_bow_joy/pic/item/d0b159021e7cf35b3812bb25.jpg   树状数组中用的d【】，每个点都有一定的管辖范围；  如d[1]=a[1];  d[2]=a[1]+a[2];  d[3]=a[3];  d[4]=a[1]+a[2]+a[3]+a[4];  等等；  这样的结构关键是为了，对一个数组内部动态的删除，增加，来高效的求某个点或者某个区间的值；  比如说对数组a，改变某一位的值需O（1），求某个k区间值O（k）；  这样的m次操作是用O（m\*k）；  显然数据很大时，效率要差很多；  而树状数组，它改变某一位，或者求某个区间的和，都是O（logN）；效率大为改善；  对于上图怎么计算区间的和；关键是需要几个函数；  先说树状数组最简单的用法，修改上面图中的某个点，并求某段区间的和；  第一个函数；  **int** lowbit(**int** x) {     **return** x&(-x); } 这个函数主要是用来求的是某个点管辖范围；  x&(-x);这个东西的由来就不说了。。。灰常奇妙啊。。  如果是x+=x&(-x);就是得到的改点的父节点的值；比如x=4时；就能得到8；  而x-=x&(-x)的话，就是得到x这个点的管辖区间的下个区间的管辖点；  比如说，x=7,代入后6；不断循环到0能依次得到 6.。。4.；  把他们所有的管辖区间正好是1....7；  第二个函数 **void** update(**int** x,**int** num) {     **while**(x<=N)      {          d[x]+=num;          x+=lowbit(x);      } }  这个函数，是用来修改树状数组的；  如果是一般的算法只用修改改点就可；但是树状数组必须修改所有改点被管辖的区间；  比如把a数组的 a[2]减去1，（令N=16）；则所有2被管辖的点有4，8，16都应该减去1；  就是调用函数 update（2，-1）； 第三个函数 **int** getSum(**int** x) {     **int** s=0;     **while**(x>0)      {          s+=d[x];          x-=lowbit(x);      }     **return** s; }  这个函数就是求区间和了。。比如getSum（7）的话，就是求a[1]+a[2]+...a[7];  上面是最基本的用法；要学习树状数组必须把上面的过程原理搞明白；  搞明白d数组和原来a数组的区别，初学者应该自己画一画；  另外有好几种最原始树状数组的变形（一会儿讨论二维的）；  大体上可以分为两种；  一，每次修改的是一个点，所求的是关于某段区间；  这种情况最好办；比如说poj2352 stars；求每个点前面比他小的点的个数；  只用设置数组a[],先全是0，然后有某个点就依次修改，并以此统计；  这一种是最基本的向上修改，向下统计；（基本上都是。。。）  二，每次修改的是一个区间，所求的值是关于某个点的；  代表的典型题目是HOJ1556 color the ball；  这个题是每次修改了一整个区间，最后求的是每个点修改的次数；  这个需要将上面的函数，稍加修改；  **void** update(**int** x,**int** num) {     **while**(x>0)      {          d[x]+=num;          x-=lowbit(x);      } }  要向下修改，将它后面的区间都加一遍；  再向后修改，把不必要的修改区间再减去；  用他的父节点记录每个点的染色次数；  统计时要  **int** getSum(**int** x) {     **int** s=0;     **while**(x<=N)      {          s+=d[x];          x+=lowbit(x);      }     **return** s; }  这种修改一个区间，而求某个点的修改次数的，一般的都是向下修改，向上统计；  对于二维的情况，d[][]中的d[i][j]这个点也有他自己的管辖区间，从一维推广到二维；可以很容易理解；  一维情况是一段区间的管辖范围，二维就是一个矩形的管辖范围，而每一维可以独立考虑；  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  下面是几道题目的简单分析；  一，两种情况； 1，要向上统计，向下修改；一般是修改一段区间的值，查找的是某个位上的值；  HDU1556 color the ball；  这个题是这类题最基本的；关键是理解这个向上统计，向下修改时怎么操作的和原理；  poj2155 Matrix  这个是楼天成出的题目；是一道很典型的二维数状数组的应用；是hdu1556的二维版本；  记住要向上统计，向下修改；二维情况，最主要的是理解那个数组中的每个点保存的值的意义（一个矩形区域的总和）；最后记住取余；  poj2299 Ultra-QuickSort；（求逆序数）；  这个题可以用归并排序做出，这里就略过不说了。。  求逆序数，也是树状数组的典型应用，求一堆数中前面比他大的数的个数；也需要向上统计，向下修改；  可以与求一串数中前面比他小的数的个数这个题来比较思考。。。  poj3067 Japan  此题要先对x排序，再对y求那个前面比他大的数的个数；（理解下）；  也是基本应用；  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  二，2，要向上修改，向下统计；一般是修改某个位置上的值，查找的是一段区间的和；  poj2352 stars  这个题是最基本的一维情况，刚才已说过。。  poj1195 Mobile phone  是个二维的情况。改变的是某个位置上的数的大小；就将所有管辖这个点的 点修改；  统计的时候，就是向下统计；  poj2481 Cows  这个题要把y按不降排序，x不升排序，基本是就是poj2352了。。不过注意其中可以有完全重合区间；  poj3321 apple Tree;  这题要用到树状数组；但难度在那个dfs怎么求那个时间戳；求出后，在套入树状数组的那部分东东。就搞定了。。  poj1990 MooFest ;  这题O（n^2）算法必然超时；排序后要存储前面比他小的数的总和，和个数；故树状数组；  poj2309  树状数组的小试牛刀，一下搞定，想明白的话。。  可以用树状数组的地方，一定可以用线段树；反过来则不行；  但是，树状数组编码简单，对于一定区间修改，求值，很高效；  另外，一个讲树状数组灰常好的网址要吐血推荐。。。  <http://www.topcoder.com/tc?module=Static&d1=tutorials&d2=binaryIndexedTrees#prob>  恩，还是刷题有意思。。以后决定只有周末玩游戏。。呵呵。。。 |