**树状数组总结**

|  |
| --- |
| 问题：已知数组 a[],元素个数为n，现在更改数组中某些元素的值，求更改后a数组中i到j区间内元素的和（1<=i<=j<=n）。  对于这个问题我们当然可以用最朴素的方法来解决，从a[i]一直累加到a[j], 最坏的情况下复杂度为O(n)，对于m次更改和访问来说的复杂度就会为O(m\*n)，当m和n较大时，复杂度就太大导致我们提交题目时就会TLE了，囧！我们可以想一下，在元素发生改变的个数是比较少的，而我们却把所有的元素都进行了一遍计算，这样其实造成了很多无谓的运算，我们就可以把和分成一个个特定的小区间进行存储，从而减少再次运算时无谓的相加。但是如果我们存任意区间的和话，当n特别大的时候，存储量就会特别大，不易更改区间里的内容。我们整理一下思路，现在已经确定了要存储一些特定区间和的想法，但是不知具体该如何存储与操作。  如图  http://hiphotos.baidu.com/%D0%DC%C3%A8yingcai/pic/item/95c07089dbb66225c9fc7ad5.jpg  我们可以发现  http://hiphotos.baidu.com/%D0%DC%C3%A8yingcai/pic/item/ea66d2a04a53ea9acaefd0d6.jpg  首先观察c[k]中存储的是什么  例如 c[1]，把1转化为2进制后为1，右边第一个1代表的是1，从a[1] 向前数一个元素c[1]=a[1];  c[2],2转化为2进后为10，右边第一个1代表的是2，，从a[2]向前数两个元素   c[2]=a[1]+a[2];  c[3],3转化为2进制之后为101，右边第一个1代表的是1，则从a[3]向前数1个元素，c[3]=a[3]  总结一下， c[k]存的就是从a[k]开始向前数k的2进制表示中右边的第一个1代表的数字个数个元素之和。  这样做有什么好处呢，举例分析一下例如我们对元素a[2]进行了更改，它影响到的c[]数组中的元素只有c[2]c[4]c[8]我们只需一层层的向上修改就可以了。  例如我们求前4项的和s[4]，把4转化成2进制为 100，右边的第一个1出现在第2位上代表的是4，也就是说要从a[4]向前数4个元素，即 c[4],则s[4]=c[4]。  求前7项和s[7]的话，把7转化为2进制111，右边的第一个1出现在第0位上代表的是1，则从a[7]向前数一个元素，即c[7],c[7]=a[7].  去掉最右边的1后变为110，右边的第一个1出现在第2位上，代表的是2，则从a[6]向前数两个元素，即c[6],c[6]=a[5]+a[6].  再去掉最右边刚用过的那个1，变为100，右边的第一个1出现在第三位上，代表的是4，则从a[4]向前数4个元素，即c[4],c[4]=a[1]+a[2]+a[3]+a[4].  s[7]=c[4]+c[6]+c[7]，7的2进制表示为111，求和时只相加了3次.效率大大提高！应该看出来了吧，如果求s[k]的话，只需查找k的2进制表示中1出现的次数就能得到最终结果。不过千万注意进行更新时可千万不要传入的参数为0啊，因为下面i+=LowBit(i)会成为死循环哦！  int LowBit(int t)//计算t二进制中最右边的1所代表的数字  {  return t&(-t);  }  void Modify(int i,int val)//对更改的数值进行更新，同时相应的对上层的进行更新  {  while(i<=n)  {     c[i]+=val;     i+=LowBit(i);  }  }  int sum(int i)//对前i项进行求和  {  int sum=0;  while(i>0)  {     sum+=c[i];     i-=LowBit(i);  }  return sum;  }  具体题目分析  Poj2352是一道树状数组的入门题目，这道题的题意为我们省去了不少工作y坐标已经按升序，y相同时x按升序排列好了，是一道树状数组的裸题，直接用树状数组可得。源代码  [hi.baidu.com/%D0%DC%C3%A8yingcai/blog/item/89c271f2cc620ba3a50f525b.html](http://hi.baidu.com/%D0%DC%C3%A8yingcai/blog/item/89c271f2cc620ba3a50f525b.html)  Poj2481这道题就是比2352多加了排序，和查询。可以先按y从小到大，y相等时再按x从大到小排序！然后从后向前扫描，记录i之前所有的j区间Sj<Si的个数。源代码  [hi.baidu.com/%D0%DC%C3%A8yingcai/blog/item/96ca9f141b55f65df2de32eb.html](http://hi.baidu.com/%D0%DC%C3%A8yingcai/blog/item/96ca9f141b55f65df2de32eb.html)  Poj3321 首先建树(我采用的是邻接表), 然后DFS先根遍历该树, 对每个节点记录其最近序号(按照遍历顺序重新编号)和其子树的最小最大序号(也就是标记一棵树遍历的第一个和最后一个节点的遍历序号st, ed).剩下的就可以是典型的树状数组处理了.如:统计一个节点为根的树中apple数目时, 直接计算到st－, ed的和然后相减即可.源代码  <http://hi.baidu.com/%D0%DC%C3%A8yingcai/blog/item/88c70b26907ac327d5074225.html>  Poj1195前边都是一维树状数组，这道题变为了二维数组，其实本质并没有什么变化，直接变为二维即可源代码  [hi.baidu.com/%D0%DC%C3%A8yingcai/blog/item/902dc1cceba1af0f93457e5f.html](http://hi.baidu.com/%D0%DC%C3%A8yingcai/blog/item/902dc1cceba1af0f93457e5f.html)  Poj3067比1195更深了一步，需要一个思维的转化，将两个岛上的城市用一矩阵的行和列来表示，如果出现两个城市建桥的话，就标记该点为1，没有桥的标记为0  0 0 0 1 0 0 1 0 1 1 0 0  当某个点为1的时候，那么其右上方1的个数就是该点表示的线段交点数，代码中有一简单的转换，留给读者自己思考了啊，源代码  [hi.baidu.com/%D0%DC%C3%A8yingcai/blog/item/787544219f5fc65e9922edec.html](http://hi.baidu.com/%D0%DC%C3%A8yingcai/blog/item/787544219f5fc65e9922edec.html)  敬请大牛指点，感激不尽！ |
|  |