* **树状数组，线段树简介 - [**[**ACM**](http://kinslovertec.blogbus.com/c2708662/)**]**

2009-08-09

[版权声明](http://creativecommons.org/licenses/by/3.0/deed.zh)：转载时请以超链接形式标明文章原始出处和作者信息及[本声明](http://bangzhuzhongxin.blogbus.com/logs/11205960.html)  
<http://kinslovertec.blogbus.com/logs/43784719.html>

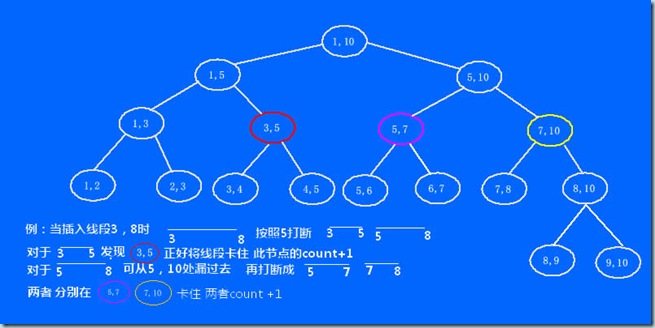
据weisili学长说，线段树是一个非常灵活，变种很多，应用很广的数据结构，实在是很感兴趣，比起DP，数据结构和图论方面的东西对我还是有更大吸引力的。

线段树实际上是一个特殊的二叉树，它不停的将线段二分，在操作前需按照线段长度先将线段树建好在进行操作，在插入线段时，即覆盖线段上某个子线段，同样递归操作，修改相应线段/线段组的count值,count代表此段区域被覆盖过的次数，插入的方式可以保证不会在计算覆盖次数时出错，查询某点被覆盖次数时，按照其所属范围递归下去，每次加上当前线段被覆盖的次数，最后得到的就是本点被覆盖的次数，时间复杂度O(LogN)。

总结起来基本的线段树操作有三个：

1建树 2插入线段 3查询

插入过程可以这么理解，整个树的节点使一些按depth划分优先级的通道，其长度何能通过的长度比他小的，且范围位于它之间的线段，当要插入一个线段时，相当于按照树中线段的划分将线段打碎，逐层下放，当正好卡在某一节点时，本节点count计数加一，

[](https://5lnpkw.bay.livefilestore.com/y1mbsunH7twoiil0Ex_3NzK1Njczd-Wzy4DzhHHXs0FMrodOm5lpHcaPWogL0wUNVkdW-4rbsBi8_3xZDrMKCdWnNBS9Qs892wvRozMPd4zYw2R9DCQXt_JorilaX6FHi8ety2Fp722a5Uk_-eU0w8m6g/%E7%BA%BF%E6%AE%B5%E6%A0%91%5b8%5d.jpg)

[**树状数组题目总结(转载)**](http://jinpingxp.spaces.live.com/blog/cns%2156C2A01D2CE133DC%21402.entry)

原出处：李璐 <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。

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。

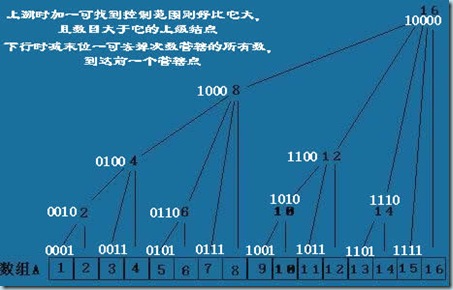
poj 2155 Matrix          
此题是二维树状数组基本应用的变通，不像hoj1640简单的插入统计，而此题是对一个二维区间数值翻转（0变1，1变0），最后询问某处的值。所以可以使用奇偶性判断，这样我们可以使用树状数组在这个二维区间内加，就相当于翻转，最后统计某点翻转的次数。

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。

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。

[**初学 树状数组的一点点想法**](http://jinpingxp.spaces.live.com/blog/cns%2156C2A01D2CE133DC%21400.entry)

树状数组是一个很有用的数据结构，数组内元素的修改和查询区间段元素合是此数据结构的两个基本操作，这里用到两个数组a[],C[]。a数组是原数组，仅用于理解，编码时不用有a数组，基本内容就是用C[]数组来对应原数组a[]，通过对C[] 数组的操作来实现它的两个基本操作：1。修改a中单个元素2。区间求和，的时间复杂度均为O(logN)（根据爱的猥琐大牛的说法，这是第一模式，第二模式为区间段修改，查询单个元素值）  
现讨论第一模式：  
树状数组用到了二进制的知识：（如图）

[](https://5lnpkw.bay.livefilestore.com/y1mDMM9040_lDNomBLd4xCuDmYMU4qoAHUo5VH2QDDMYZYuaRJqfhJMpFPEiET2JGHLomp1isLb_sSHIXUjRsdney5HgNHL0ySYplFNgO26-2kJgTuQdxyjpYvmY2cayh77YpZFEdgjDy1RnkjjEU7xdg/%E6%A0%91%E7%8A%B6%E6%95%B0%E7%BB%84%5b2%5d.jpg)

也就是说C[N]是a[i]到a[N]的和， N-i + 1= 2^k, k是N的二进制标识形式下末尾0的个数，也可以将2^k看成将N的二进制数减去末尾1时N所减少的数，例如对于12，其二进制1100，2^k = 100B = 2^2。

此时，对于求a到b区间上的和sum（a,b）就可求了，例如sum(3, 7) = getsum(7) – getsum(2);

上面让读者对树状数组可以有个总体直观的印象，下面讨论getsum的实现：

getsum()操作的实现，由前可知就是几个C[]的和，例如getsum(10) = C[10] + C[8] ,又如 getsum(14) = C[14] + C[12] + C[8]等等；下面是实现此操作的代码：

int Getsum(int k)  
{  
    int sum = 0;  
        while(k>0)  
       {  
        sum += c[k];  
        k -= Lowbit(k);  
        }  
    return sum;  
}

解释：Lowbit（）是用来求2^k的函数，读者参见上图及解释即可理解

Lowbit（）函数实现：

int Lowbit(int m)  
{  
    return m&(-m);  
}

解释：本处用到了位操作，可以自己举几个例子算算加深理解

最后是改变单值的操作了，即对单个元素值的修改，改数时，区间和数组C[]的某些元素就要发生改变，比如对a中第i个数修改，第一步修改C[i]，然后是包含C[i]的直属上级C[j],之后是C[j]的上级C[k]，知道下标超过了数组元素总个数n，就不用再改了，再改没有意义，超过n的范围不会去查询和。下面就是如何由i确定j,再由j确定k的问题了.  
对于一个数N，C[N]所包含的数是a[N,i],i = N - 2^k + 1,其上级的可容纳范围必然比它大，而由需要是它直属的上级，最简单的方法就是在N的二进制形式末尾1处再加一，这样这个数的二进制数尾零至少多了一个，而且是它增加尾0最少的方式，就是C[N]的直属上级,可参见上图理解，因此解决这个问题又可以使用到Lowbit函数来求2进制末尾增加1，十进制增加了几；

下面是Change（）函数的代码：

void Change(int k,int m,int n)  
{  
   while(k <= n)  
   {  
       c[k] += m;  
       k += Lowbit(k);  
   }  
}

注：参数是：数组总共有n个元素，将第K个元素加上m。