树状数组

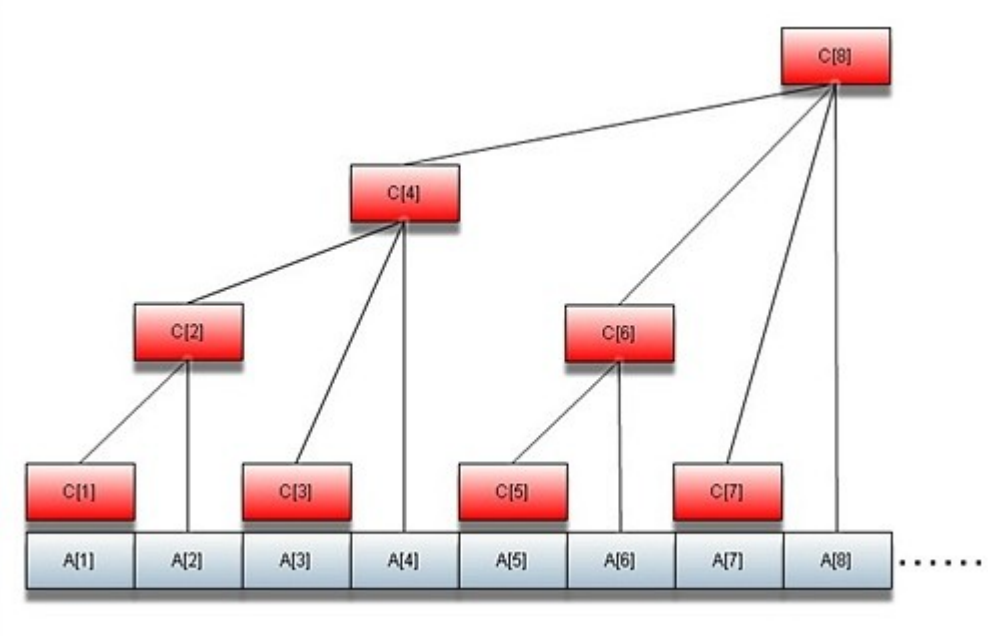
武钢三中 吴豪

【引言】

在解题过程中，我们有时需要维护一个数组的前缀和S[i]=A[1]+A[2]+...+A[i]。但是不难发现，如果我们修改了任意一个A[i],S[i]、S[i+1]...S[n]都会发生变化。可以说，每次修改A[i]后，调整前缀和S[]在最坏情况下会需要O(n)的时间。当n非常大时，程序会运行得非常缓慢。因此，这里我们引入“树状数组”，它的修改与求和都是O(logn)的，效率非常高。

【理论】

为了对树状数组有个形 象的认识，我们先看下面这张图。



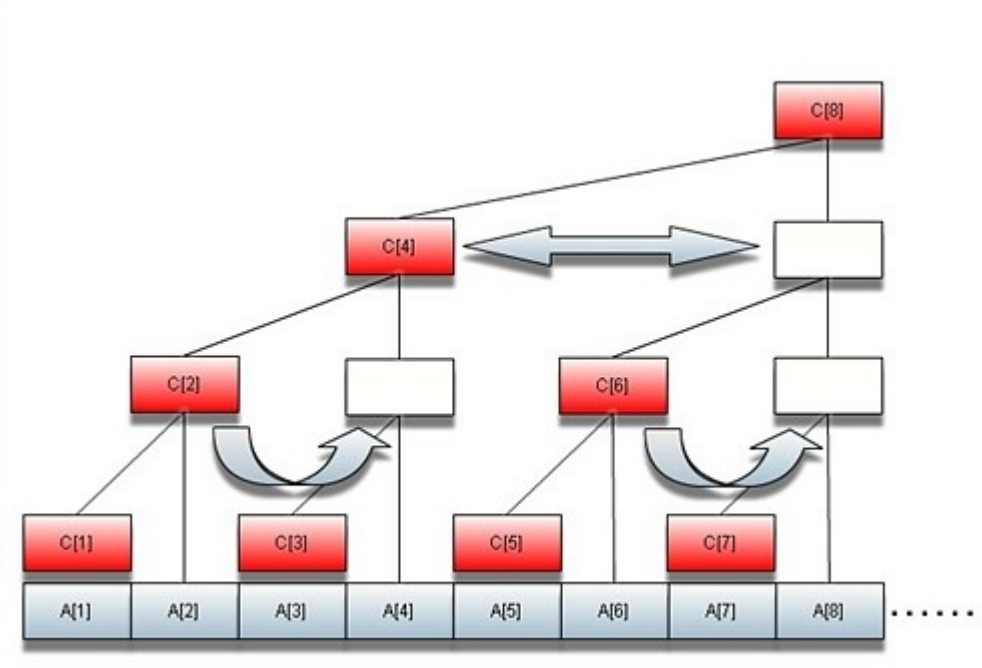
如图所示，红色矩形表示的数组C[]就是树状数组。

这里，C[i]表示A[i-2^k+1]到A[i]的和，而k则是i在二进制时末尾0的个数，或者说是i用2的幂方和表示时的最小指数。(当然，利用位运算，我们可以直接计算出2^k=i&(i^(i-1)) ）同时，我们也不难发现，这个k就是该节点在树中的高度，因而这个树的高度不会超过logn。所以,当我们修改A[i]的值时，可以从C[i]往根节点一路上溯，调整这条路上的所有C[]即可，这个操作的复杂度在最坏情况下就是树的高度即O(logn)。另外，对于求数列的前n项和，只需找到n以前的所有最大子树，把其根节点的C加起来即可。不难发现，这些子树的数目是n在二进制时1的个数，或者说是把n展开成2的幂方和时的项数,因此，求和操作的复杂度也是O(logn)。

接着，我们考察这两种操作下标变化的规律：

首先看修改操作：

已知下标i，求其父节点的下标。我们可以考虑对树从逻辑上转化：



如图，我们将子树向右对称翻折，虚拟出一些空白结点（图中白色），将原树转化成完全二叉树。

有图可知，对于节点 i，其父节点的下标与翻折出的空白节点下标相同。

因而父节点下标 p=i+2^k （2^k 是 i 用 2 的幂方和展开式中的最小幂，即 i 为根节点子树的规模）即 p = i + i&(i^(i-1)) 。

接着对于求和操作：

因为每棵子树覆盖的范围都是 2 的幂，所以我们要求子树 i 的前一棵树，只需让 i 减去 2 的最小幂即可。即 p = i - i&(i^(i-1)) 。

至此，我们已经比较详细的分析了树状数组的复杂度和原理。

在最后，我们将给出一些树状数组的实现代码，希望读者能够仔细体会其中的细节。

【代码】

求最小幂 2^k:

```
int Lowbit(int t)
{
    return t & ( t ^ ( t - 1 ) );
}
```

求前 n 项和：

```
int Sum(int end)
{
    int sum = 0;
    while(end > 0)
    {
```

```
        sum += in[end];
        end -= Lowbit(end);
    }
    return sum;
}
```

对某个元素进行加法操作：

```
void plus(int pos , int num)
{
    while(pos <= n)
    {
          in[pos] += num;
          pos += Lowbit(pos);
    }
}
```

以下内容转自OIBH

http://oibh.org/bbs/viewthread.php?tid=23806&highlight=%CA%F7%D7%B4%CA%FD%D7%E9

作者 sai901013

话说我学了这个**树状数组**N个日子了,还没有用过,今天一用,果然不凡...-_-只是PKUOJ的E文让我比较抓狂...看了半天才知道它讲什么.这个题目乃入门题目,不赘述了...下面说一下我用**树状数组**的体会(语言比较难理解，字体比较难忍受= =，请原谅)..

什么叫艺术

做了这个题目之后，我对算法的艺术性又有了进一步的认识。跟线段树的又长又臭相比，**树状数组**可谓娇小玲珑。那简洁的代码，与Treap、并查集等数据结构一样的美丽，清秀。不仅如此，它有着简单易用的特点，容易记忆。

最大的艺术就在于k+=(-k)k这个变化量的设计，我们可以看到，(-k)&k完全等价于k&(k^(k-1))，两个可以随便记一 个，不过显然第一个更容易记忆。虽然到现在我还不知道这两句是如何计算到“将k化为二进制数时末尾0的个数”(补充:可以参考本贴#17)=。 =！ ，但我很享受这一种奇特的美妙。

什么叫速度

除了insert()和get()时间复杂度都是O(log n)之外，其实这个很正常，**树状数组**特别的是它编程复杂度离奇的小-_-//... .據说它的时间复杂度常数还比线段树小很多。除此之外，它的空间复杂度也比线段树明显小约2/3。下面分析一下我写的Code。

```
1.
2.//先定义c[x]为树状数组,a[x]为原数组。
3. //其实这里叫remodify()函数比较合理,因为这里是修改某个元素而不是插入
4. //detal是将a[k]变化detal，而不是“变为detal”
5. //mix是要十分注意的，因为常常有人会被输入个数N所迷惑了
6. //因为修改c[k]的话对c[k+1]....c[mix]有影响，所以不难想到k的变化为增变化
7.
8. void insert(int *c,int k,int detal){
9.       for(; k<=mix ;k+=(-k)&k ) c[k] +=detal;
10.      }
11.
12.//顾名思义，getsum()就自然是某一段的和
13.//不过比较奇怪，返回的是a[1]...a[k]
14.//其实并不难为，如果你想要a[j]...a[k]，你可以getsum(k)-getsum(j)
15.
```

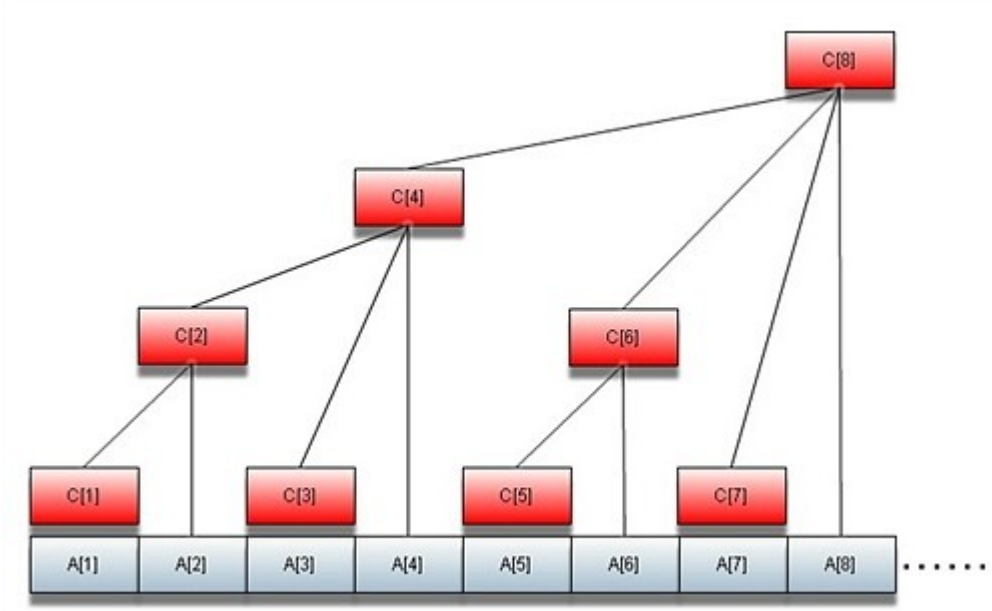
```
16. int getsum(int *c,int k){
17.       int t;
18.       for(t=0; k>0 ;k-=(-k)k ) t +=c[k];
19.       return t;
20.       }
```

复制代码

可能你看某幅图会容易搞清楚啥是**树状数组**，但是我不打算贴这一幅图片，因为百度一下就可以找到。

---//这个这个....绝对不是我懒！

算了，还是贴一下



：

什么要注意

世界上没有免费的午餐，你喜欢上**树状数组**，你就一定要承受它的缺点，-_-//貌似喜欢 MM 也一样。

不过有一点我还没有绝对地搞清楚，有人说**树状数组**可以解“最长不下降子序列”问题，但这个应该会涉及到求某一个区间范围内最值。但是貌似**树状数组**要符合减法原则，也就是说不能求某一个区间范围内最值。初步来讲，我认为“**树状数组**解最长不下降子序列”和“减法原则”可能并不矛盾，可能有一种符合“减法原则”解法，我不做深入讨论探究了。

下面引用一段话：

在很多的情况下，线段树都可以用**树状数组**实现．凡是能用**树状数组**的一定能用线段树．当题目不满足减法原则的时候，就只能用线段树，不能用**树状数组**．例如数列操作如果让我们求出一段数字中最大或者最小的数字，就不能用**树状数组**了。

除了上面的“减法原则”之外，还需要说明的是：**树状数组**由 1 开始。

习惯用 C/C++的同志们申请地址的时候必须从 0 开始，自然而然地，我们习惯了第一个元素是 0，但是**树状数组**必须由 1 开始。具体细节想一下就会知道，那是因为跟那个“将 k 化为二进制数时末尾 0 的个数”有关。

最后要说的是，干看是不行的，练习才是正道。 PKUOJ 2352 Stars 是一道不错的入门题目。

可以参考以下代码。不过最好还是自己做啦...-_-///..

本文个人原创,如有雷同,肯定抄我sai90.!

此外本人语言是比较难理解的，若各位高手们看得不顺眼，请见谅...

```
1.
2. #include <stdio.h>
3. int c[32003];
4. int a[32003];
5. int n,mix=32003;
6. void insert(int k,int detal)
7. {
8.       for(; k<=mix ;k+=(-k)k ) c[k] +=detal;
9. }
10.int getsum(int k)
11.{
12.      int t;
13.      for(t=0; k>0 ;k-=(-k)k ) t +=c[k];
14.      return t;
15.}
16.int main()
17.{
18.     memset(c,0,sizeof(c));
19.     memset(a,0,sizeof(a));
20.     int i,x,y;
21.     scanf("%d",n);
22.     for(i=0; i<n ;i++)
23.     {
24.             scanf("%d%d",x,&y);
25.             x++;
26.             a[getsum(x)]++;
27.             insert(x,1);
28.     }
29.     for(i=0; i<n ;i++) printf("%d\n",a);
30.     return 0;
31.}
```