

树状数组

数列操作 NK OJ1321

- 给出 n 个整数: a_1, a_2, \dots, a_n
- 反复进行下列两种操作:
 - 1. 修改某个数的值, 将第 i 个数增加 d
 - 2. 求出指定区间 $[L, R]$ 的数字和(如 $a_3 + a_4 + \dots + a_{10} + a_{11}$)
- $n \leq 100000$, 总操作数 ≤ 100000

可用**线段树**

也可用**树状数组**

树状数组用来干什么？

- 树状数组(binary indexed tree , 发明者Peter M. Fenwick 1994), 是一种设计新颖的数组结构, 它能够高效地获取数组中连续k个数的和。
- 概括说, 树状数组通常用于解决以下问题:
数组A中的元素可能不断地被修改, 怎样才能快速地获取连续几个数的和?

介绍

什么是树状数组?

给定一个数组 $A[]$,
我们设一个数组 $C[]$ 满足

$$C[1] = A[1]$$

$$C[2] = A[1] + A[2]$$

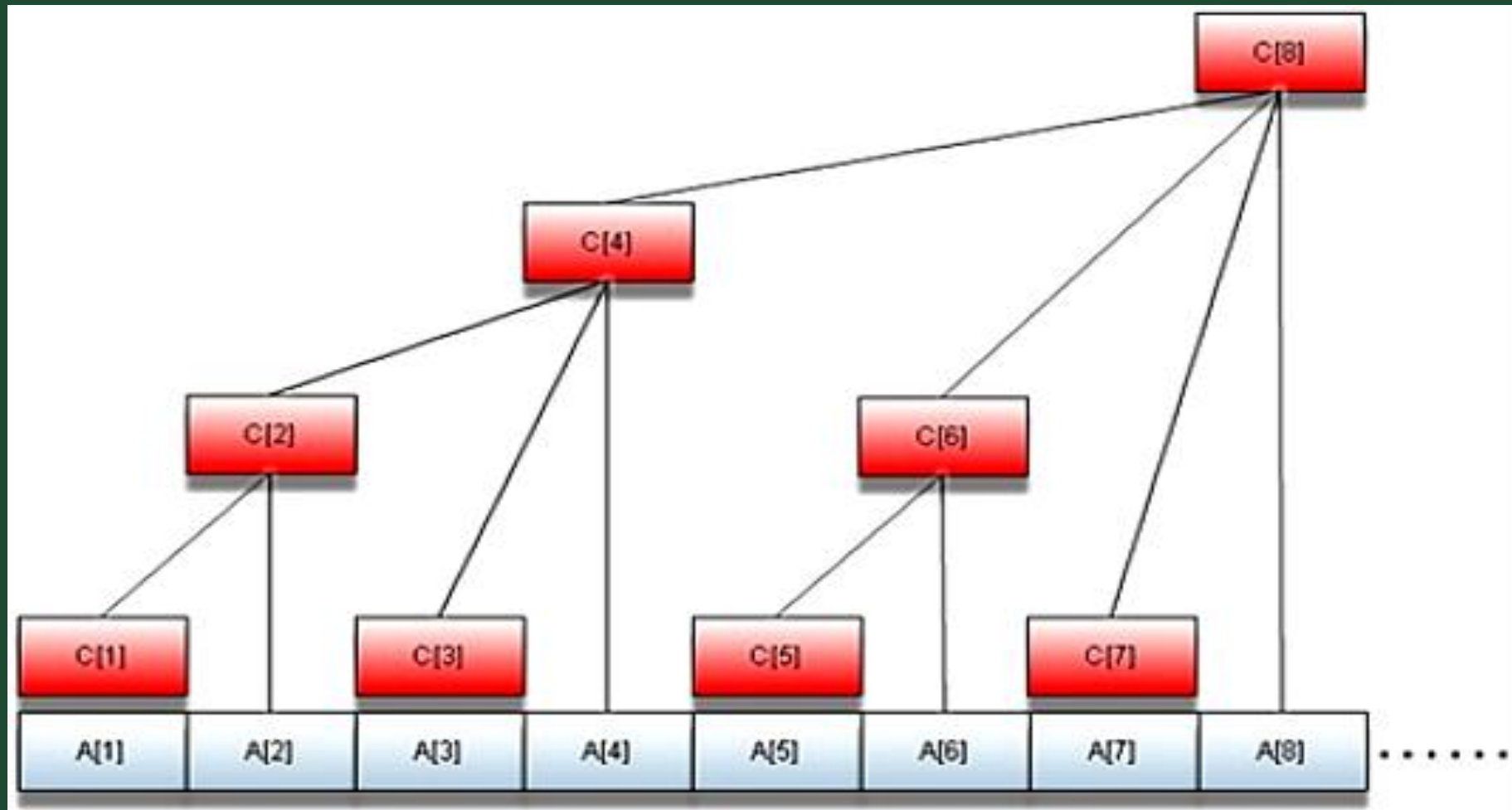
$$C[3] = A[3]$$

$$C[4] = A[1] + A[2] + A[3] + A[4]$$

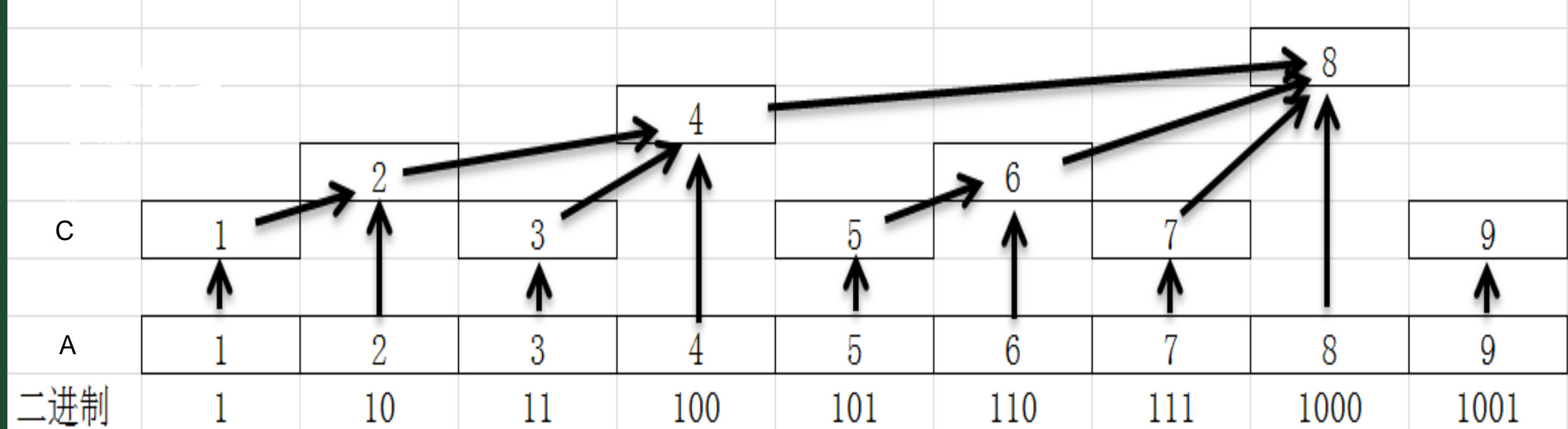
$$C[5] = A[5]$$

$$C[6] = A[5] + A[6]$$

.....



C数组就是树状数组, $C[i] = ?$



给定序列（数组） A ，我们设一个数组 C 满足

$$C[i] = A[i-2^k+1] + \dots + A[i]$$

其中， k 为 i 对应的二进制数末尾0的个数， i 从1开始算！

则我们称 C 为树状数组。

下面的问题是，给定 i ，如何求 2^k ？

答案很简单： $2^k = i \& (-i)$

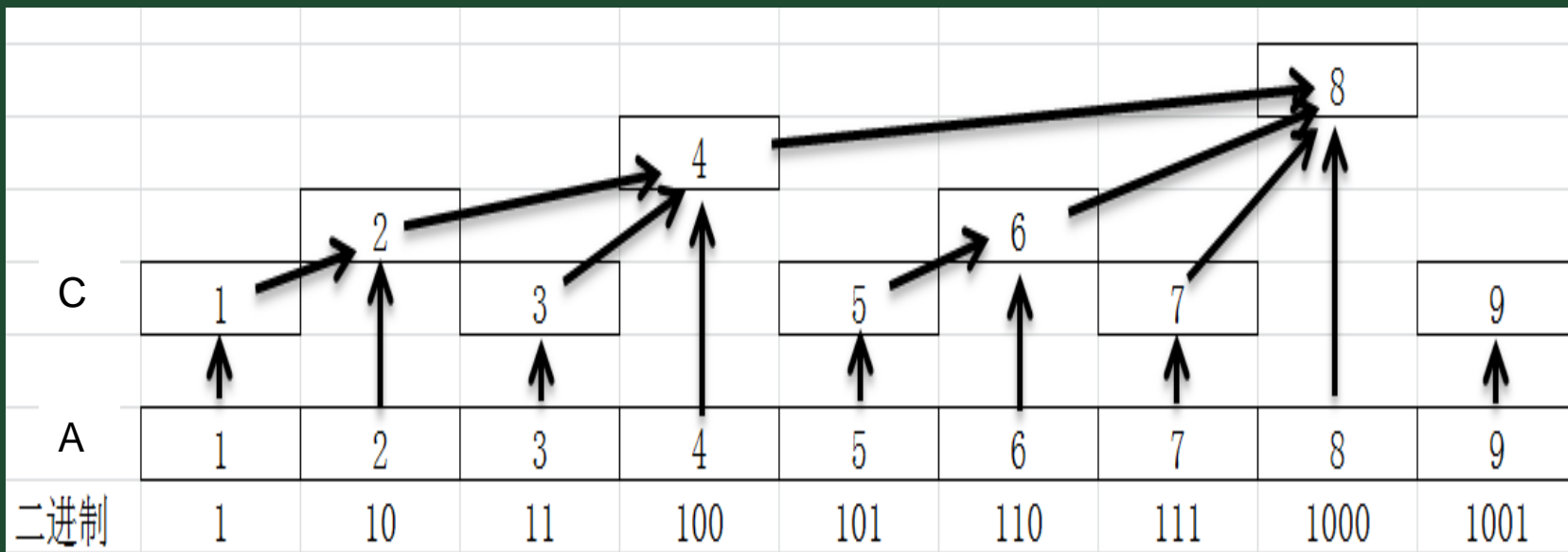
$i \& (-i)$

- 比如对于“010101000” 最末有3个0, k 的值应该是3, 算出的 2^k 应该为8
- 110101000 这是 $-i$ 的原码
- 101010111 这是 $-i$ 的反码
- 101011000 这是 $-i$ 的补码
- 010101000 这是 i 的补码
- 000001000 这是 $(i \& (-i))$
- $(00001000)_2 = (8)_{10}$

```
int lowbit(int x)
{
    return x & ( -x );
}
```

求和

重庆南开中学



当我们求 $A[1] + \dots + A[x]$ 的之和时,

$C[x]$ 如果包含的不一定是 $1 \dots x$ 的全部和, (比如 $C[6] = A[5] + A[6]$) 就需要再找一个 $C[k]$ (显然 $k < x$) 累加起来, 这个 k 我们称之为 x 的**前驱**, 举个例子:

$$A[1] + A[2] + \dots + A[6] = C[6] + C[4]$$

$$A[1] + A[2] + \dots + A[7] = C[7] + C[6] + C[4]$$

前驱的编号即为比自己小的, 最近的, 最末连续0比自己多的数

所以 x 的前驱 **$k = x - \text{lowbit}(x)$**

// 相当于 x 剪掉了自己最右边的1

求和GetSum()

```
int getSum(int x)    //求A[1]+A[2]+...+A[x]
{
    int Sum= 0;
    for ( int k = x; k > 0; k -= lowbit(k) ) Sum += C[k];
    return Sum;
}
```

求t的前驱

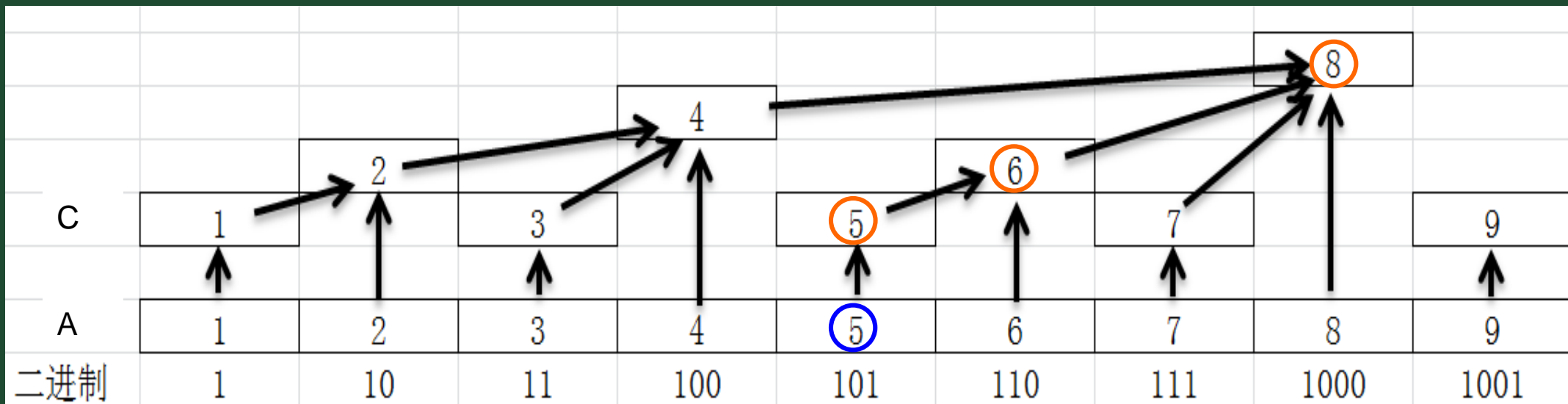
直接用一个循环求得Sum,时间复杂度为 $O(\log N)$

求区间 $[x,y]$ 之和怎么办? $A[x]+A[x+1]+\dots+A[y]$

$\text{getSum}(y) - \text{getSum}(x-1)$

修改

重庆南开中学



修改了某个A[i],就需改动所有包含A[i]的C[j];
从上图看就是要更改从叶子节点到根节点路径上的所有C[j]

怎么求一个节点的父节点?

经过观察和探究,前人们得出了这个规律:

父亲:比自己大的,最近的,末位连续0比自己多的数

x节点父亲的编号 = $x + \text{lowbit}(x)$ //相当于让x最右边的1变为0, 增加1个0

修改modify()

```
void modify(int x,int d) //将所有包含A[x]的C[ ]增加d
{
    for( int i = x; i<=n; i+=lowbit(i) )C[i] += d;
}
```

求i的父亲

/*其中n为数组的长度，时间复杂度同样是 $O(\log N)$ 的*/

数列操作 nkoj 1321

- 给出 n 个整数: a_1, a_2, \dots, a_n
- 反复进行下列两种操作:
 - 1. 修改某个数的值, 将第 i 个数增加 d
 - 2. 求出指定区间 $[L, R]$ 的数字和(如 $a_3 + a_4 + \dots + a_{10} + a_{11}$)
- $n \leq 100000$, 总操作数 ≤ 100000

数列操作 nkoj 1321

处理：给出 n 个整数： a_1, a_2, \dots, a_n

一开始树状数组 $C[]$ 全是0，一边输入初始数组，一边将数据加入树状数组

```
long long A, i, C[100005];  
for (i=1; i<=n; i++)  
{  
    cin>>A;  
    modify(i, A);  
} //时间 $O(\log n)$ 
```

```
void modify(int x, int d) //将所有包含A[x]的C[]增加d  
{  
    for( int i = x; i<=n; i+=lowbit(i) )C[i] += d;  
}
```

数列操作 nkoj 1321

处理操作： 1.修改某个数的值,将第i个数增加d
modify(i,d)

处理操作： 2.求出指定区间[L,R]的数字和
getSum(R) - getSum(L-1)

```
int getSum(int x)    //求A[1]+A[2]+...+A[x]
{
    int Sum= 0;
    for ( int k = x; k > 0; k -= lowbit(k) ) Sum += C[k];
    return Sum;
}
```

例1：求逆序对数

树状数组例题1：求逆序对数

一个长度为 n ($1 \leq n \leq 10000$)的数列由数字1到 n 构成，求该数列中逆序对的个数。

输入格式：

第一行，一个整数 n

第二行， n 个空格间隔的整数，表示数列

输出格式：

一行，一个整数，表示逆序对的个数。

样例输入：

5

3 2 5 1 4

样例输出：5

样例说明：逆序对分别是(3 2),(3 1),(2 1),(5 1),(5 4)

求逆序对可以用归并排序，也可以用树状数组

树状数组例题1：求逆序对数

1. 因为题目给出的数据范围是 $1 \leq n \leq 10000$, 所以我们相当于有了一个A数组, 下标范围是1到10000。其中A[i]记录的数字i是否出现过, 是A[i]=1, 否则A[i]=0。

那么, 对于树状数组而言, $C[i] = A[i - \text{lowbit}(i) + 1] + \dots + A[i]$, 相当于记录了在 $[i - \text{lowbit}(i) + 1, i]$ 这段区间中的数字出现的个数。

2. 一边读入数列一边讨论。当读入了数字x:

首先, 我们查询 $[x+1, n]$ 这段区间中的数字出现的个数, 因为它们是在x之前出现的, 又比x大, 所以都能与x构成逆序对。 $ANS += \text{getSum}(n) - \text{getSum}(x);$

然后, 我们再把x插入到树状数组中, 更新所有包含了x的C[]。

注: 因为树状数组的元素比如C[i]表示的是一段区间的数字和, 若C[i]对应的区间中包含了x, 我们就把C[i]++。

也就是 $\text{modify}(x, 1)$

树状数组例题1：参考代码

```
int main()
{
    int i,x,ans=0;
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        scanf("%d",&x);
        ans+=(getSum(n)-getSum(x));
        insert(x,1);
    }
    cout<<ans<<endl;
}
```

```
int lowbit(int x)
```

```
{
    return x&(-x);
}
```

```
int getSum(int x)
```

```
{
    int Sum=0;
    for( int i=x;i>0;i-=lowbit(i) )Sum+=c[i];
    return Sum;
}
```

```
void insert(int x,int d)
```

```
{
    for( int i=x;i<=n;i+=lowbit(i) ) c[i]+=d;
}
```

树状数组例题1：求逆序对数

如果题目改为：由 n ($n \leq 1,0000$) 个正整数构成的数列，数列中数字的数值范围是在1,000,000,000以内，求该数列中逆序对的个数。

这个问题还能用树状数组来处理吗？

我们无法开一个大小为 $C[1000000000]$ 的树状数组，直接用树状数组是不行的。
这样的问题应该先离散化,再用树状数组处理，就是所谓

离散化+树状数组

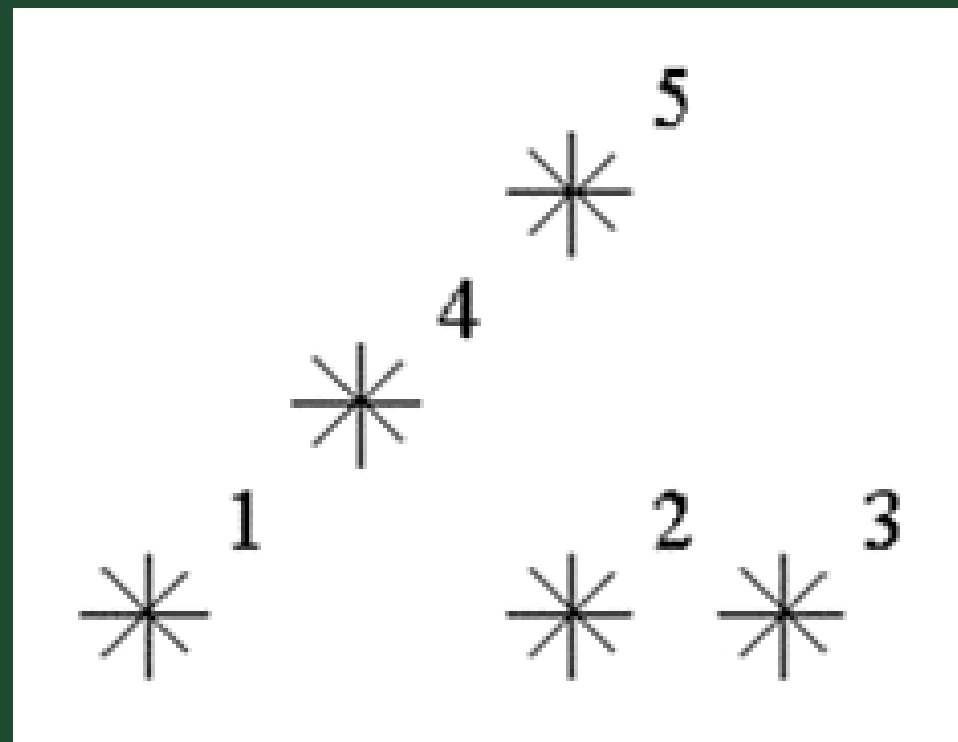
例2：数星星

树状数组例题2：数星星 nkoj1908 源自ural1028

宇航员经常检测星图,在星图上,星星由点表示而且每颗星星都有笛卡尔坐标。星星的等级表示左下方(包含正左和正下方)星星的数量。宇航员想知道星星等级的分布。

例如,如右面图形所示,第5号星等级是3(它由三个标记为1,2和4的星组成)。在此地图上,0等级的星星只有一个(1号),1等级的有两个(2和4号),2等级的有一个(3号),3等级的有一个(5号)。

请你设计一个程序,给出N ($N \leq 200,000$) 颗星星及坐标($0 \leq X, Y \leq 32000$), 在给定地图上计算出每个等级星星的数量。



树状数组例题2：数星星 树状数组解法

1. 我们可以把星星先按y坐标由小到大排序，若两颗星星y相同，则按x坐标由小到大排序。

2. 因为题目给出的数据范围是 $0 \leq x, y \leq 32000$ ，所以我们相当于有了一个A数组，下标范围是0到32000，其中A[i]记录的是x坐标为i的星星的颗数。

那么，对于树状数组而言， $C[i] = A[i-2^k+1] + \dots + A[i]$ ，相当于记录了x坐标在 $[i-2^k+1, i]$ 这段区间中的星星的颗数。

3. 按排序后的顺序，从左往右依次讨论每颗星星：

当讨论到坐标为(p,q)的星星时，只需查询区间 $[0, p]$ 中星星的颗数就行了。因为前面讨论过的所有星星的y坐标都不会大于q。

查询完成后，再把该星星插入到树状数组中，即更新C数组中，所有包含了A[p]的元素的值。

树状数组例题2：数星星 树状数组核心代码

```
void main()
```

```
{
```

```
    scanf("%d",&n);
```

```
    int i,x,y;
```

```
    for(i=1;i<=n;i++)
```

```
    {
```

```
        scanf("%d%d",&x,&y);
```

```
        ans[getSum(x)]++; //getSum(x)查询当前坐标在1到x间的星星颗数,ans记录结果
```

```
        Modify(x,1);
```

```
    }
```

```
    for(i=0;i<=32000;i++)printf("%d\n",ans[i]);
```

```
}
```

注意：

1.题目给出的数据是排好顺的；

2.树状数组只能表示从1开始的区间，而此题中x,y可能取到0，上面代码需适当修改。

树状数组例题2：数星星 效率对比

User	Memory	Time	Language	Code Length
ShadowIterator	1944K	124MS	G++	0.75K

← 树状数组

416162623	3216K	667MS	G++	1.33K
-----------	-------	-------	-----	-------

← 线段树

小结

- 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.
- 树状数组的每个操作都是 $O(\log(n))$ 的复杂度.

习题: NKOJ3697, 3702, 3703, 4406

思维: NKOJ 3709

难题: NKOJ 2033