树状数组

failedbamboo@gmail.com

维护前缀和

- 包含n个元素的整数数组A，每次可以
  - C(i, j): 修改一个元素A[i] = j
  - Q(i): 询问前缀$S_i=A_1+A_2+\ldots+A_i$的值
- 如何设计算法，使得修改和询问操作的时间复杂度尽量低？

LOWBIT

- 设**C[i]=a[i-2^k+1]+…+a[i],**其中k为i在二进制下末尾0的个数, 令LOWBIT(i)=2^k
  - 例如, i=1001010110010000, 则k=4
- 不难得到LOWBIT公式
  - LOWBIT(x) =x and (x xor (x-1))

```
inline int lowbit(int x)
{
        return x & (x ^ ( x - 1 ) );
}
```

修改A[x]的后果

- 修改A[x], 可能有很多C随之修改
- 例如x=76=$(1001010)_2$，可以得到：

p_1= 1001010
p_2= 1001100
p_3= 1010000
p_4= 1100000
p_5= 10000000

$P_1=x$
$P_{i+1}=P_i+LOWBIT(P_i)$

如何证明P_i和P_{i+1}之间的C值都不改变?

- 则需要依次修改$C[p_1],C[p_2],\ldots$

前缀和A[1]+…+A[x]的计算

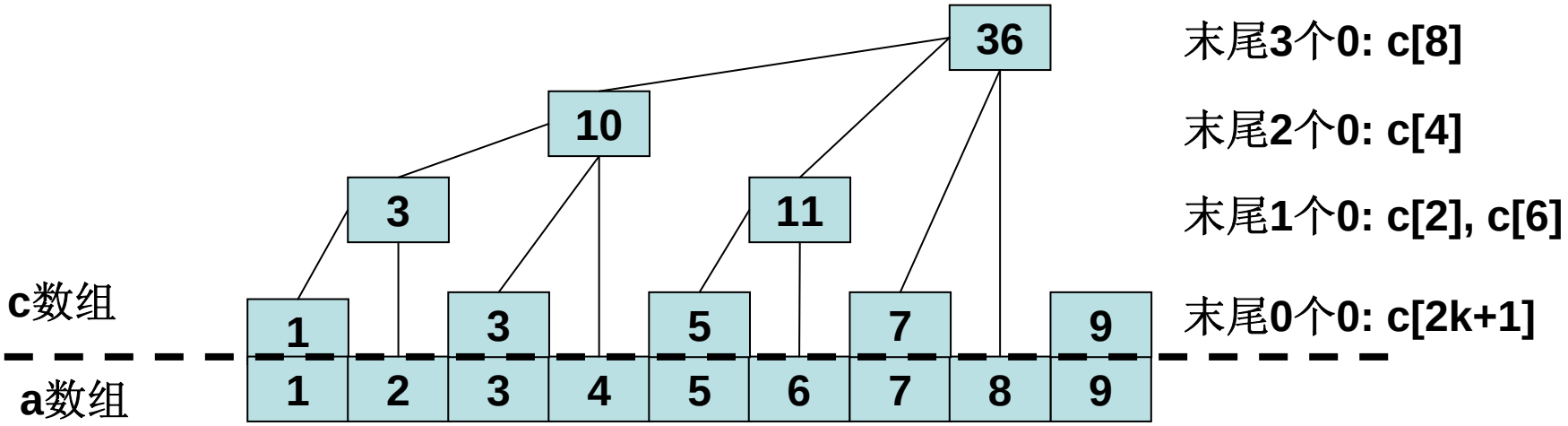
- 首先累加C[x], 因为它的定义是以x结尾的连续和. 它的连加起点是C[i-LOWBIT(i)+1], 因此问题转化为了求A[1]+…+A[i-LOWBIT(i)]
- 由此, 我们得到递推式

$$P_1=x$$
$$P_{i+1}=P_i-LOWBIT(P_i)$$

- 则只需要累加$C[p_1]$, $C[p_2]$, …

C的分层树状结构

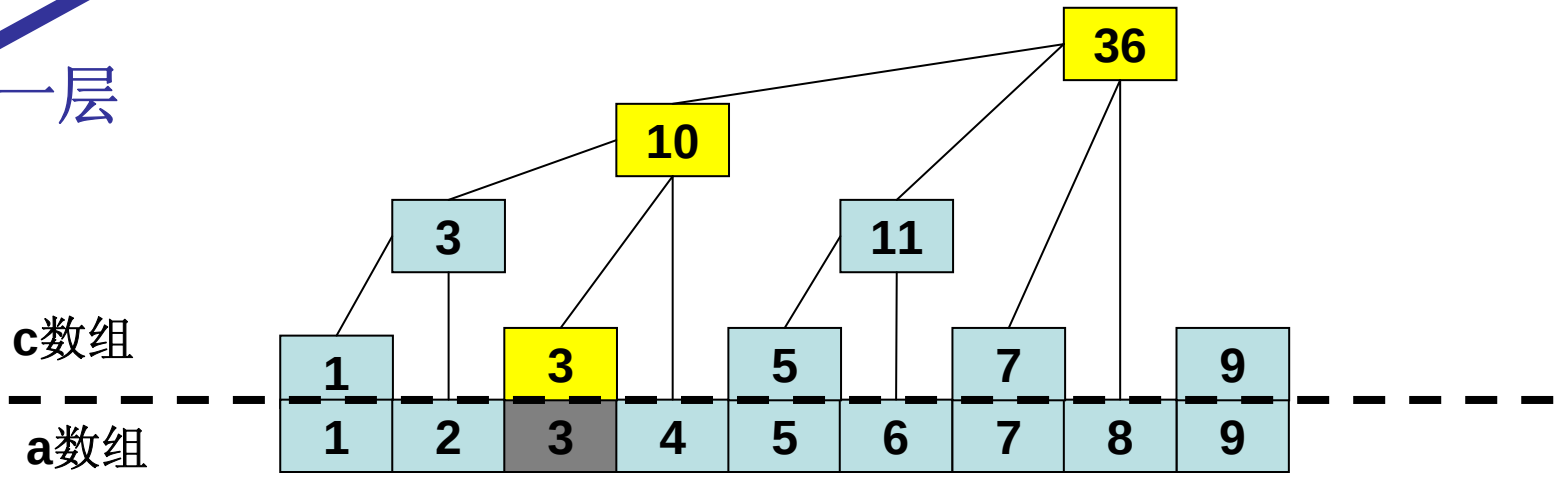
- 数组c[i] = a[i-2^k+1]+a[i-2^k+2]+…+a[i]
  - k为i在二进制形式下末尾0的个数
  - 起点是把i的最后一个**1**变为0再加1
- c数组的分层表示和递推关系如下图



第i层末尾有i个零，度数为i+1，定义式2^i项

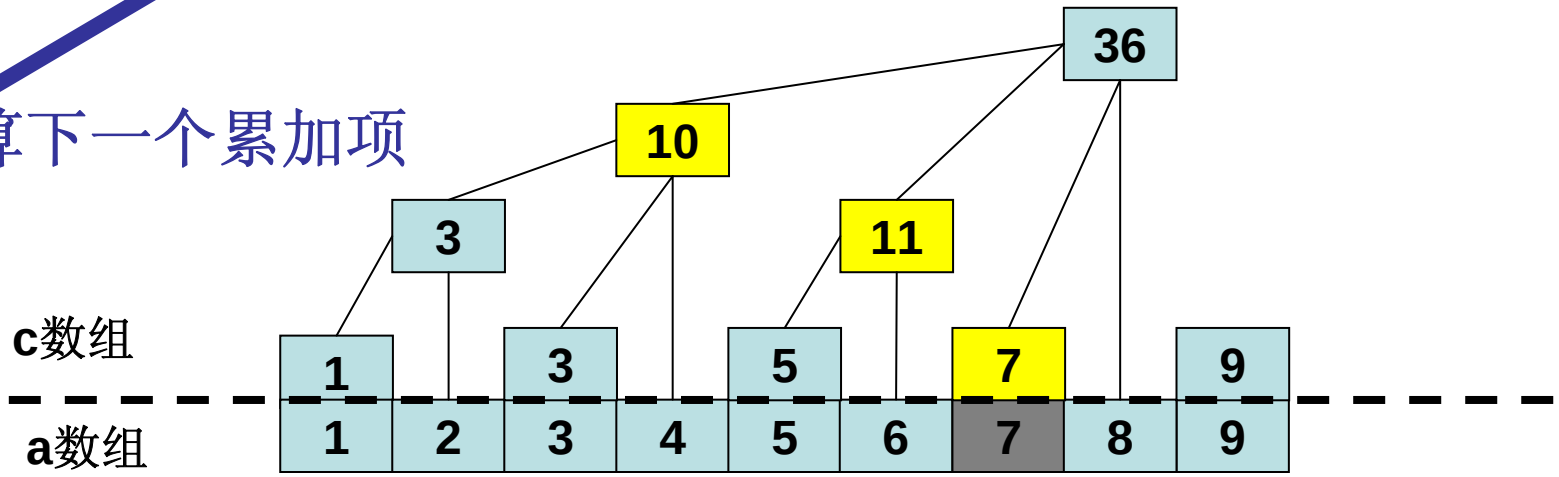
```
void Add(int p , int d){
        while(p <= n){
                C[p] += d;
                p += lowbit(p);
        }
}
```

提升一层



```
int Sum(int p){
        int ret = 0;
        while( p ){
                ret += C[p];
                p -= lowbit(p);
        }
        return ret;
}
```

计算下一个累加项

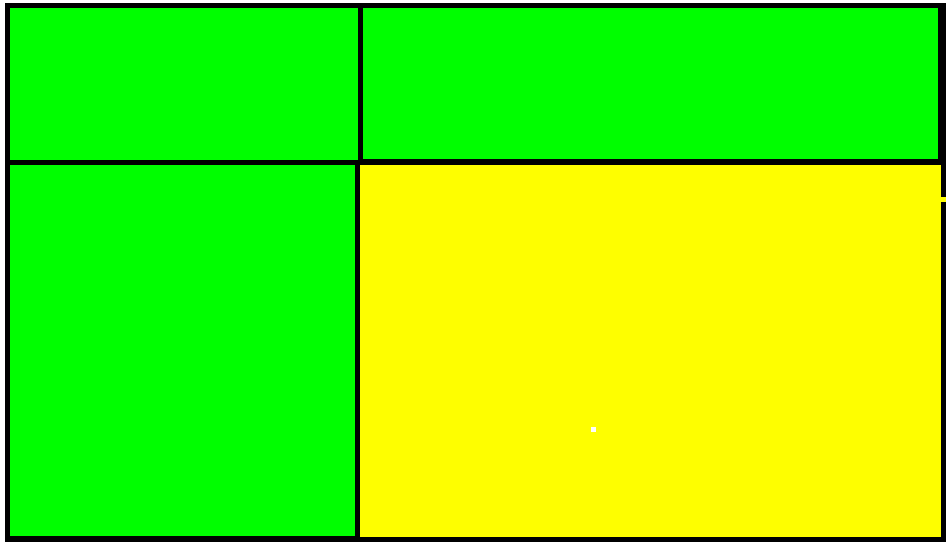


例1. 手机 (IOI2001)

- Tampere地区被划分成N*N个单元，形成一个N*N的表格，行列坐标均为0到N-1，每个单元中有一个移动电话信号发射基地
  - C(X,Y,A): 基地(X,Y)移动电话数的变化A
  - Q(L,T,R,B): 询问矩形区域内移动电话总数
- 注意C操作中A可正可负

分析

- 任意矩形转化为四个前缀矩形
- 转化为二维前缀和
- 二维独立, 分别处理, 每次操作$\log^2 n$



例2. 01矩阵

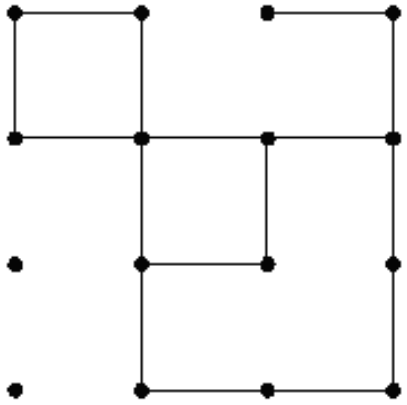
- 给n*n的01矩阵,支持
  - C(x_0,y_0,x_1,y_1)：改变矩形（每个元素取反）
  - Q(x,y)：查询(x,y)的值

分析

- 构造辅助01矩阵C’，初始为0
- 矩形分解: C(x_0,y_0,x_1,y_1)等价为改变以下4点的值
  - C’(x_0,y_0), C’(x_0,y_1), C’(x_1,y_0), C’(x_1,y_1)
- 元素(x,y)的最终值完全取决于在C’中(x,y)的右下方的元素和的奇偶性
- 维护二维前缀和→二维数状数组

例3. 方格问题

- 在一个N*N格点方阵的给了一些长度为1的线段, 每条线段的格式为(X , Y , D), 其中D可以是H（往右）或者V（往下）



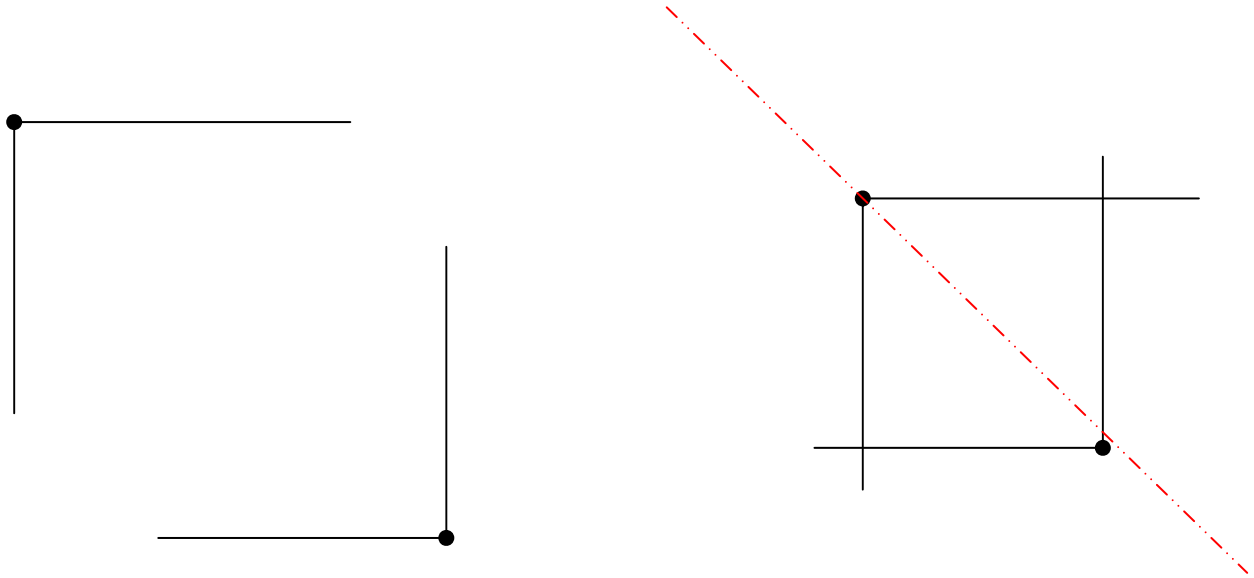
- 计算出一共有多少个正方形。上图共有3个正方形，两个边长为1，一个边长为2

分析

- 用一个01矩阵表示每条元线段是否存在
- **算法一:** 然后枚举左上角顶点和边长，检查每条元线段是否都存在。时间复杂度$O(n^4)$
- **算法二:** 预处理计算出每个点往下，往右可以延伸多长，判断整条边降为O(1)，总时间降为$O(n^3)$
- 设左上角为(x, y), 若存在边长为K的正方形
  - 不一定存在K-1的正方形
  - 但是我们至少可以确定，K-1正方形的其中两条边是必然存在的

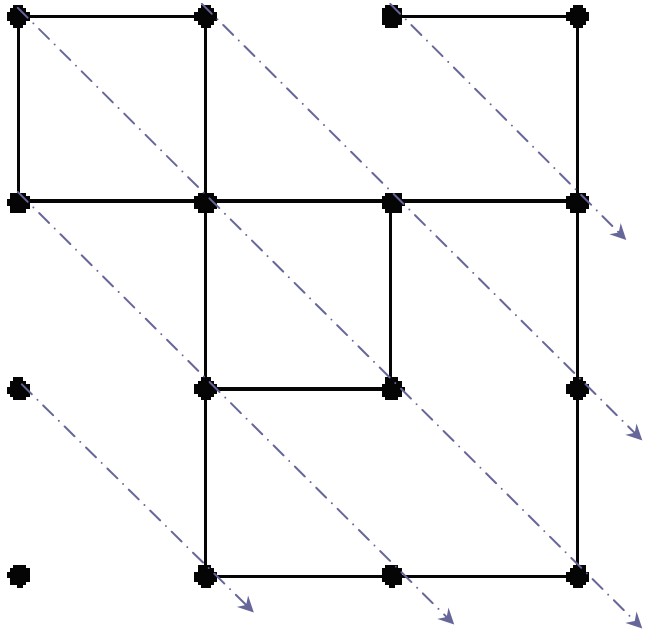
充要条件

- 把一个正方形拆成左上和右下两部分，则两个部分可以构成一个正方形的充要条件是
  - 两个顶点在同一条倾斜135度的直线上
  - 右下部分的两条边与左上部分两条边有交点



算法

- 依次处理各条斜线, 每条线自左上到右下依次考虑各个点
  - 累加新点往左上方向“作用范围”内的点数
  - 删除往右下方向已经延伸不到当前位置的点
- 用树状数组维护连续和，$O(n^2 \log n)$



例4. 队伍选择 (Balkan 2004)

- IOI要来了，BP队要选择最好的选手去参加。为了选出的选手是最好的，教练组织了三次竞赛并给出每次竞赛排名。所有N名选手都参加了每次竞赛并且每次竞赛都没有并列的。当A在所有竞赛中名次都比B前，我们就说A是比B更好。如果没有人比A 更好，我们就说A是优秀的。
- 求：优秀选手的个数

分析

- **朴素算法：**依次判断每个选手i是否优秀。判断的方法是枚举所有其他选手j，看是否j比i更好。时间复杂度为$O(n^2)$
- 三次比赛很容易降低为两次：按第一次比赛的排名重新给选手编号（第一名为1号…），则i比j好当且仅当i<j且a[i]<a[j]且b[i]<b[j]，其中a和b分别是第二次和第三次的比赛名次

分析

- 在平面中依次插入各个(a[i], b[i])，每次计算x<a[i]的点中y的最小值min，则i是优秀的当且仅当min比b[i]大
- 需要维护前缀最小值min(x[1…i]). 由于点只插入不删除，所以所有前缀最小值都不会变大，树状数组的框架仍然适用!
- 时间复杂度：O(nlogn)