二维树状数组

主讲人：邓哲也

二维树状数组

树状数组的一大优点就是它非常容易扩展到高维。

定义一个二维数组 A[1..n,1..n]，维护以下两种操作：

（1）给 A[i,j] 加上 d；

（2）查询 A[1..i, 1..j]的和。

二维树状数组

如同一维树状数组，把二维的sum数组定义如下：

$$\mathbf{sum}[\mathrm{x}][y]=\sum_{i=x-c(x)+1}^{x}\sum_{j=y-c(y)+1}^{y}A[i][j]$$

当sum数组第一维固定了之后，第二维记录的就是对应行的若干列合并之后的部分和。

二维树状数组

```
Query(int x, int y){
      ans = 0
      while(x > 0){
            ty = y
            while(ty > 0){
                  ans += sum[x][ty]
                  ty -= C(ty)
            }
            x -= C(x)
      }
      return ans
}
```

二维树状数组

```
Add(int x, int y, int d){
        while (x <= n){
                ty = y
                while (ty <= n){
                        sum[x][ty] += d
                        ty += C(ty)
                }
                x += C(x)
        }
}
```

二维树状数组

可以发现，每一重循环都只会执行 $\log_2 n$ 次。

因此二维情况下，单次查询和修改的时间复杂度都是

$O(\log^2 n)$

对于 k 维，每次的时间复杂度就是 $O(\log^k n)$

POJ 2155 Matrix

给一个 N*N 的二维数组 A，每个元素都是 0 或 1，一开始均为 0.

要求支持两种操作：

C x1, x2, y1, y2：对这个矩形内的所有元素做一次“非”操作，即 0 变 1，1 变 0

Q x y：查询 A[x][y] 的值

n <= 1000, T <= 50000

POJ 2155 Matrix

首先注意到 0 变 1，1 变 0 不太好维护。

实际上我们只需要把每次非运算看成+1，然后判断这个数模2的值，就可以知道它是否发生了改变。

问题变成了矩形加1+单点询问。

POJ 2155 Matrix

矩形+1 类比一维情形中的区间+1

我们可以维护差分序列

这样区间[x,y]+1变成了 [x]+1，[y+1]-1

同理，维护二维的差分序列

这样矩形+1变成了 两个点+1，一个点-1

POJ 2155 Matrix

对于左下角(x1,y1),右上角(x2,y2)的矩形来说。

(x1,y1) +1

(x2+1,y1) -1

(x1,y2+1) -1

(x2+1,y2+1) +1

这样就可以做到查询 (x,y) 的左下角和的时候，可以等价于单点询问。

时间复杂度 $O(T\ \log^2 n)$

下节课再见

万门中学 zx.wanmen.org