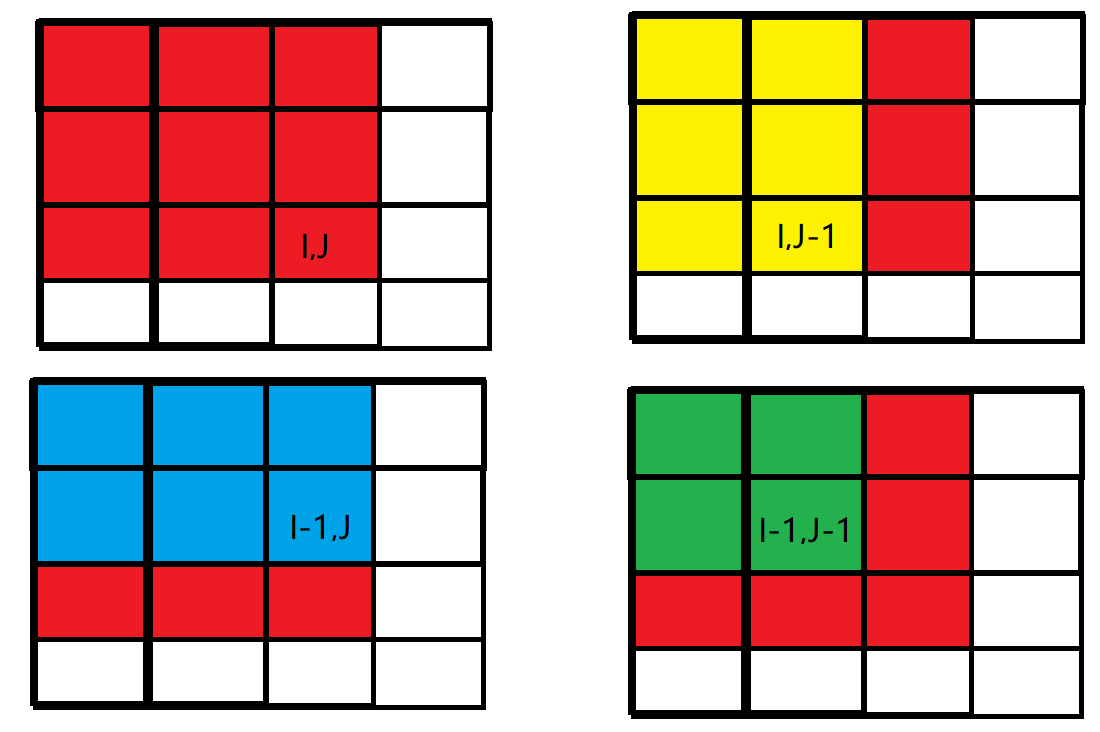
**二维差分**

**1. 二维前缀和**

首先我们学习二维差分的之前需要先了解二维前缀和

如图:



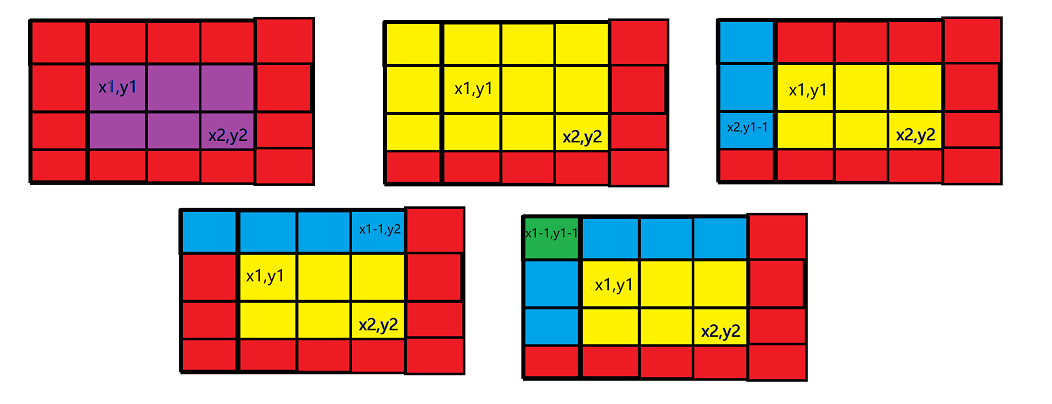
因为是从左到右，从上到下的遍历，当要求红色部分，(0,0) 到（i, j）处的前缀和时，我们黄色部分和蓝色部分已经是已知的了，而它们重叠的部分就是绿色部分，所以把黄色和蓝色部分的结果加起来，再减去绿色部分，最后加上 (i, j) 处的值就是（i, j）位置的前缀和了。

所以，二维前缀和就是:

sum[i][j]=a[i][j]+sum[i-1][j]+sum[i][j-1]-sum[i-1][j-1]

而我们要求左上角是(x1,y1),右下角是(x2,y2)的矩形区间内的值处理出前缀和后也可以O(1)时间内求出来。

如图:



我们要求紫色部分的值，我们已知的是黄色部分的值，但它多了两个蓝色部分的值，而两个蓝色部分有重叠了个绿色部分

所以要求的区间内的值就是:

sum[x2][y2]-sum[x2][y1-1]-sum[x1-1][y2]+sum[x1-1][x2-1]

**2. 二维差分**

我们以这题的样例来解释什么是二维差分

首先给出结论:

若想在点(x1,y1)到点(x2,y2)围成的矩形的每个位置增加1

则我们要进行以下操作:

设vis[i][j]:原点到点(i,j)围成的矩形的总和

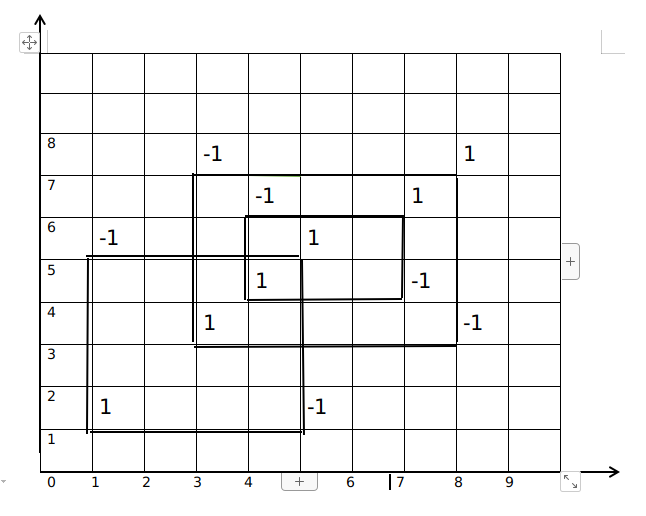
vis[x1 + 1][y1 + 1]++;

vis[x2 + 1][y2 + 1]++;

vis[x1 + 1][y2 + 1]--;

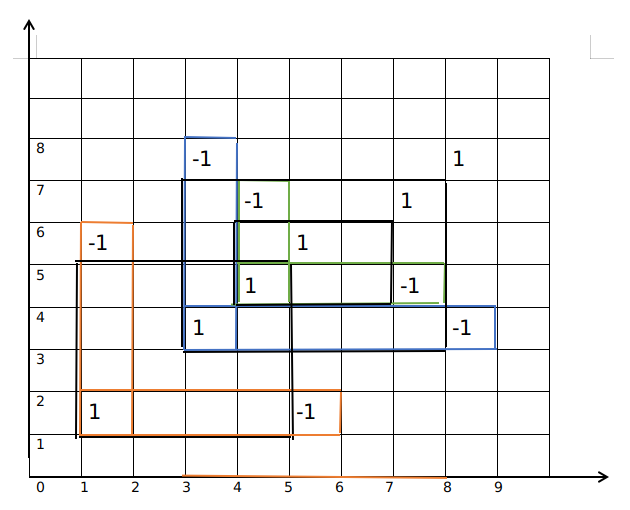
vis[x2 + 1][y1 + 1]--;

我们给出根据样例的三组数据实现后的坐标轴:



黑色线段围成的矩形就是每次要加1的矩形

下面我们解释为什么要这样做:



我们就以 点(1,1) 和 点(5,5)为例讲解

首先 vis[2][2]++;

而 vis[2][2] 的值改变,谁的值会受影响呢?

很明显: i >= 2 && j >= 2的vis[i][j]的值全部都会收到影响

我们先来看 i = 2,j >= 2的情况 也就是点(1,1)到点(2,6)橙色长方形

我们令

很简单,我们来证明一下: 令 j=7

想象一下点(0,0)和点(2,7)围成的矩形

我们会发现 点(2,2)加的1 和点(2,6)减的1 抵消了

故不会受影响了

证毕

这里明白后其他两处加1和减1就也能明白了

根据二维前缀和表示的是右上角矩形的和，由于差分只涉及前面相邻的数（由一维可以推出），并且由前面范围的数相加得到这个位置的数。那么类比二维前缀和和一维差分，可以简单推测出二维差分的公式

IMG_256

如何从差分矩阵得到原矩阵呢？可以参考下面公式

IMG_257