IOI2020 中国国家集训队第一阶段作业 解题报告

安徽师范大学附属中学 曾致远

1 Robots protection

1.1 题目来源

codeforces 575I[1]。

1.2 题目描述

Robots industries 公司生产的机器人可以用于保卫领土。

现在有一个小岛，我们把整个小岛放到二维平面直角坐标系上。小岛被分成了 $N \times N$ 个部分，每个部分是一个整点，可以用坐标 (x, y) 来描述，满足 $1 \leq x, y \leq N$ 且 x 和 y 是正整数。

每个机器人可以保卫小岛上的一个区域，这个区域是一个两条直角边分别平行于平面直角坐标系 x 轴和 y 轴的等腰直角三角形，一个点被保卫当且仅当它在这个三角形的内部或者边界上。

每一时刻，小岛的主人会购买一个机器人用于保卫小岛上一个区域的领土，或是查询某一个点被多少个机器人保卫了。你需要支持这些查询。

1.3 输入格式

第一行两个正整数 N, Q 描述小岛的大小和操作的个数。接下来 Q 行，每行描述一个操作，操作的格式有两种，如下所述：

(1)$dir\ x\ y\ len$

这个操作的含义是，小岛的主人购买了一个机器人，dir 描述三角形的方向，x, y 描述三角形的位置，len 表示三角形的大小，具体而言机器人保卫的等腰直角三角形区域如下所述：

$dir = 1$：三角形的顶点是 (x, y),$(x + len, y)$,$(x, y + len)$；

$dir = 2$：三角形的顶点是 (x, y),$(x + len, y)$,$(x, y - len)$；

[1]题目链接：https://codeforces.com/problemset/problem/575/I

$dir = 3$：三角形的顶点是 $(x, y), (x - len, y), (x, y + len)$；

$dir = 4$：三角形的顶点是 $(x, y), (x - len, y), (x, y - len)$。

(2)2 x y

这个操作的含义是，小岛的主人要查询点 (x, y) 被多少个机器人保卫了。

1.4 输出格式

对于每次询问，输出一行表示这次询问的答案。

1.5 数据规模和约定

$1 \le N \le 5000$

$1 \le Q \le 10^5$

$1 \le dir \le 4$

三角形的每个顶点都在小岛内部。

询问的每个点都在小岛内部。

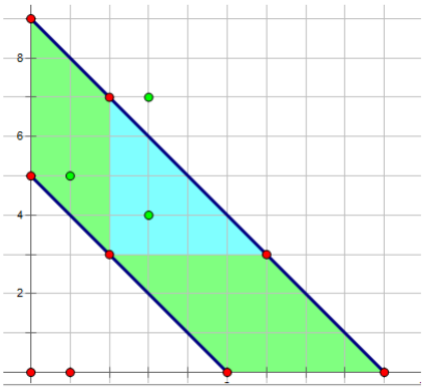
1.6 时空限制

时间限制：$1.5s$

空间限制：$512MB$

1.7 算法介绍

首先只需要考虑一个方向的等腰直角三角形对查询的贡献如何计算。对于剩下三个方向的三角形，可以考虑把整个坐标系旋转，都变成同一方向的，对查询的贡献计算方法就通用了。然后每次询问直接累加求和即可。下面只考虑如何处理 $dir = 1$ 的三角形对查询的贡献。



假如一个机器人保卫的区域是如图所示的蓝色区域。把其斜边延长交于两条坐标轴，同时过直角顶点做斜边的平行线也交于两条坐标轴，就是如图所示的两条蓝线，那么一个点在蓝色区域内的必要条件是该点被这两条蓝线夹住 (即在图中的蓝色区域和绿色区域的并)。这两条蓝线的斜率都为 1，点 (x, y) 被这两条蓝线夹住相当于对于 $x + y$ 有一个范围的限制；具体而言，假设这个三角形是题目中用 (X, Y, len) 描述的三角形，那么这个限制就是 $X + Y \le x + y \le X + Y + len$。

考虑那些被两条蓝线夹住但是并不在蓝色区域内的点，实际上就是绿色区域中的点，这些点 (x, y) 除了满足 $X + Y \le x + y \le X + Y + len$，还有一些其他限制。这两个绿色区域，显然被分成了两个不相交的部分，一个在左上方、另一个在右下方。对于左上方绿色区域中的点，还满足 $0 \le x \le X - 1$；对于右下方绿色区域中的点，还满足 $0 \le y \le Y - 1$。

综上所述，覆盖了一个点 (x, y) 的三角形区域个数，等于两条蓝线夹住了 (x, y) 的数目减去覆盖了 (x, y) 的绿色区域数目。根据前面的转化，现在要做的事可以描述成：支持插入一个矩形，和查询一个点此时被多少个矩形覆盖。这是一个经典的二维数点的问题，可以直接用二维线段树维护；由于本题不要求强制在线，使用分治算法可以得到更低的常数和空间复杂度。

时间复杂度是 $O(Q \log^2 N)$，如果使用分治空间复杂度是 $O(N + Q)$。事实上本题 N 比较小，所以可以使用二维树状数组维护，时间复杂度不变，空间复杂度是 $O(N^2)$，但是常数较小，实现也更为简单。

1.8　总结

本题难度较小。对于平面上两条直角边平行于坐标轴的直角三角形，考虑一个点 (x, y) 在其内部的条件，可以轻松发现是关于 x、y 和 $x + y$ 在一些区间上的限制。通过画图更进一步地思考，可以发现就是直接在对三个部分做一些加减，这三部分每一部分都只和两个信息相关，放到本题中也就是经典的数点问题了。关于实现，使用二维树状数组可以极大地简化代码，因此最终实现并不复杂。

2 Two Histograms

2.1 题目来源

Atcoder Grand Contest 35F[2]。

2.2 题目描述

给定一个 N 行 M 列的网格。网格的每个位置上都有一个非负整数，Takahashi 将会按照顺序做如下操作：

(1) 把每个位置上的数赋为 0；

(2) $\forall 1 \le i \le N$，选择一个整数 $k_i(0 \le k_i \le M)$，然后把第 i 行的前 k_i 个格子里面的数加上 1；

(3) $\forall 1 \le j \le M$，选择一个整数 $l_j(0 \le l_j \le N)$，然后把第 j 列的前 l_j 个格子里面的数加上 1。

经过这样的操作后，我们就得到了一个每个位置上的数均是 $0, 1$ 或 2 的网格。请你求出，在所有可能的操作下，可以得到多少种本质不同的网格，两个网格被称为本质不同的，当且仅当存在至少一个位置，两个网格对应位置上的数不相同。

因为答案可能很大，请求出其对 998244353 取模的结果。

2.3 输入格式

输入仅一行，两个正整数 N, M 表示网格的行数和列数。

2.4 输出格式

输出答案对 998244353 取模的结果。

2.5 数据规模和约定

$1 \le N, M \le 5 \times 10^5$

2.6 时空限制

时间限制：$2s$

空间限制：$1024MB$

[2]题目链接：https://atcoder.jp/contests/agc035/tasks/agc035_f

2.7 算法介绍

不妨先考虑这么一个问题。对于一对 (x, y) 满足 $1 \le x \le N$ 和 $1 \le y \le M$，假如先不考虑 k_x 和 l_y 是什么，而其他所有的 $k_i(i \neq x)$ 和 $l_j(j \neq y)$ 都已知，那么在什么情况下，两个不同的方案得到的网格是一样的。显然，这时候可以把网格看成全 0 的网格考虑。不难发现，只有 $k_x = y, l_y = x - 1$ 和 $k_x = y - 1, l_y = x$ 两种方案得到的网格是一样的，其他所有方案得到的网格互不相同。下面给出该性质的证明。

证明．考虑两次填数所涉及到的格子；如果没有形成一个四连通块，那么显然只有唯一的方案可以得到这样的网格；如果有相交即存在为 2 的格子，那么也显然只有唯一的方案可以得到这样的网格；除去这两种情况就只有一种形状了，即形成了一个 L 型，且这个形状上的格子都是 1，这时候有两种上面描述的方案可以填出这种形状。 □

所以不妨考虑把计数网格变成计数合法方案。对于一个方案，如果其满足 $\exists 1 \le x \le N, 1 \le y \le M$ 有 $k_x = y - 1, l_y = x$，那么这个方案就是不合法的，否则就是合法的。我们已知任意一个方案都唯一对应了一个网格；同时根据前面发现的性质，一个网格唯一对应了一个合法方案，因为可以对一个网格的某一个方案，不停把不合法的操作调整成合法的操作，从而得到一个合法方案。所以直接考虑如何计数合法方案即可。

有若干限制不能被违反的计数问题，很自然可以想到容斥。具体而言，枚举所有限制的集合，然后计数强制该集合中所有限制都被违反了的方案数，同时乘上容斥系数，求和就是答案。对应这个计数过程，枚举 R 表示有 R 对 (x, y) 满足 $k_x = y - 1, l_y = x$，这些二元组涉及到的行和列显然不会相交，所以方案数就是 $\binom{N}{R}\binom{M}{R}R!$。这时候剩下的 k_i 和 l_j 都可以随便选择，方案数就是 $(M+1)^{N-R}(N+1)^{M-R}$。于是，我们得到答案就是：

$$\sum_{R=0}^{min(N,M)} (-1)^R \binom{N}{R}\binom{M}{R} R!(M+1)^{N-R}(N+1)^{M-R}$$

这里涉及到一个容斥系数 $(-1)^R$，尽管这是一个经典的问题，但还是给出证明。

证明．考虑一个方案违反的限制集合 S，我们期望当 $S = \varnothing$ 时把方案统计到答案里面去恰好 1 次，否则就不统计。考虑 S 的所有子集，并乘上容斥系数即 -1 的子集大小次方，那么一个集合 S 被统计的次数是：

$$\sum_{T \subseteq S} (-1)^{|T|} = \sum_{i=0}^{|S|} \binom{|S|}{i} (-1)^i = (1-1)^{|S|} = 0^{|S|} = [|S| = 0] = [S = \varnothing]$$

这符合我们的要求，所以这样容斥是正确的。 □

预处理 $N+1$ 和 $M+1$ 的幂，即可做到 $O(N+M)$ 的时间复杂度，空间复杂度根据实现可以做到 $O(N+M)$ 或 $O(1)$。

2.8 总结

本题难度中等偏易。对于计数类问题，把对象转化成另一个与其一一对应的对象进行计数，是一个常用的思路。本题中就是通过观察性质，把要计数的网格找到了唯一的填数方案与其对应，从而把计数网格转化成计数满足一定条件的方案。正如题解中所说，“有若干限制不能被违反的计数问题，很自然可以想到容斥”，对满足条件方案的计数很明显可以发现容斥的模型，整个问题从而迎刃而解。在推导出答案后，实现基本就是进行简单预处理后直接复制，所以代码十分简短。

3 Reachable Cells

3.1 题目来源

AtCoder Grand Contest 028F2[3]。

3.2 题目描述

给定一个 N 行 N 列的网格，这个网格有 $N \times N$ 个方格。从上往下数第 i 行且从左往右数第 j 列的方格，可以用一个二元组 (i, j) 来描述它。每个方格要么是空的，要么被一个障碍物所占据。同时，每个空的方格上都写了一个数字。如果 $A_{i,j}$ 是 1 到 9 中某一个自然数，那么 (i, j) 是一个写了 $A_{i,j}$ 的空方格；否则 $A_{i,j} =' \#'$，(i, j) 是一个被障碍物所占据的方格。

我们称一个方格 Y 是一个方格 X 可达的 (或者说 X 可到达 Y)，当且仅当以下三个条件被满足：

(1) X 和 Y 不是同一个方格；

(2) X 和 Y 都是空的；

(3) 存在一条从 X 到 Y 的路径，满足在一个方格处时，接下来只往下或者往右走到一个相邻的空方格。

考虑所有的二元组 (X, Y) 满足 X 可到达 Y，定义一个这样的二元组价值为 X 和 Y 上数字的乘积，请求出所有满足条件二元组价值的和。

3.3 输入格式

第一行一个正整数 N，描述网格的大小。

接下来 N 行，每行一个长度为 N 的字符串。第 i 行的字符串的第 j 个元素即为 $A_{i,j}$。

3.4 输出格式

输出仅一行一个整数，表示答案即所有满足条件二元组价值的和。

3.5 数据规模和约定

$1 \leq N \leq 1500$

$A_{i,j}$ 是一个 1 到 9 中某一个自然数或者是 #。

[3]题目链接：https://atcoder.jp/contests/agc028/tasks/agc028_f2

3.6 时空限制

时间限制：$9s$

空间限制：$1024MB$

3.7 算法介绍

对于这样网格图上路径统计相关的问题，很自然可以想到用网格图分治的做法。具体而言，每次考虑一个子矩形内部对答案的贡献，就把这个子矩形分成两部分，分别计算两部分 (也分别是两个子矩形) 对答案的贡献求和，再加上跨越两个子矩形的贡献，就是整个子矩形对答案的贡献了。在接下来的一些描述中，我们会直接忽视那些有障碍物的点。

假设当前考虑的子矩形大小为 $H \times W$，即它有 H 行 W 列。我们假设 $W \leq H$。下面考虑把这个子矩形尽量均分成上下两个部分，设上面的部分为 U、下面的部分为 D，均分意味着 U 的行数是 $H_U = \lceil \frac{H}{2} \rceil$，$D$ 的行数为 $H_D = \lfloor \frac{H}{2} \rfloor$。根据前面说的分治做法，现在考虑计算 $X \in U$、$Y \in D$ 的 (X, Y) 对答案产生的贡献和。下面做出一些定义。

定义 3.7.1.

$Left(i, j)$ 表示最小的 x，满足 $D(1, x)$ 可以到达 $D(i, j)$。

$Right(i, j)$ 表示最大的 x，满足 $D(1, x)$ 可以到达 $D(i, j)$。

$Top(j)$ 表示 U 中可以到达 $D(1, j)$ 的点中，行标号的最小值，这里的标号指的是在 U 中的标号。

$Bot(j)$ 表示 $D(1, j)$ 可以到达的点中，行标号的最大值。

$Mpoint(a, b)$ 表示 $D(1, a)$ 和 $D(1, b)$ 能同时到达的点中，行标号的最小值。

$Brh(a, b, l)$ 表示 $D(1, a)$ 和 $D(1, b)$ 在 D 的前 l 行中可以同时达到点的点权和。

$Reachable(a)$ 表示 $D(1, a)$ 可以到达点的点权和。

这里的定义中有一些特殊情况。如果不存在这个函数要找的点，当这个函数是求“最大”时定义其值为 $-\infty$，是求“最小”时定义其值为 $+\infty$。

$\text{Left}(i, j), \text{Right}(i, j), \text{Top}(j)$ 和 $\text{Bot}(j)$ 这四个函数可以直接求。具体而言，可以把网格图按照能够直接到达的关系连有向边，看成一个有向无环图 DAG。在这个 DAG 上做一遍 dp 即可求出这四个函数在每个位置处的值。这个 dp 的时间复杂度是 $O(HW)$ 的。

考虑如何对于 $1 \leq a \leq b \leq W$ 求出 $\text{Mpoint}(a, b)$。首先，所有的 $\text{Mpoint}(a, a) = 1$。对于剩下的情况，考虑所有的 $D(i, j)$ 满足 $\text{Left}(i, j) \leq a < b \leq \text{Right}(i, j)$，若不存在这样的点则 $\text{Mpoint}(a, b) = +\infty$，否则找到令行标号即 i 最小的点，设其为 $D(p, q)$。下面分情况讨论：

$(1) p > \min\{\text{Bot}(a), \text{Bot}(b)\}$ 时，$\text{Mpoint}(a, b) = +\infty$。

这一点是显然的，因为在这样的条件下，找不到符合条件的点。

$(2) p \leq \min\{\text{Bot}(a), \text{Bot}(b)\}$ 时，$\text{Mpoint}(a, b) = p$。

首先 $\mathrm{Mpoint}(a,b) \ge p$ 是显然的，因为不满足所给条件的 $D(i,j)$ 一定无法被 $D(1,a)$ 和 $D(1,b)$ 同时到达。

下面证明 $\mathrm{Mpoint}(a,b) \le p$。

证明. 考虑从 $D(1,\mathrm{Left}(p,q))$ 走到 $D(p,q)$ 的任一路径 $Path_1$，和 $D(1,\mathrm{Right}(p,q))$ 走到 $D(p,q)$ 的任一路径 $Path_2$，此时有 $\mathrm{Left}(p,q) \le a < b \le \mathrm{Right}(p,q) \le q$。这时候从 $D(1,a)$ 走到第 $\mathrm{Bot}(a)(p \le \mathrm{Bot}(a))$ 行的路径 $Path$，必定会与 $Path_1$ 或 $Path_2$ 相交。具体而言，假设 $Path$ 经过了 $D(p,q)$，那么 $Path$ 与两条路径都有交点 $D(p,q)$；假设 $Path$ 经过了 $D(p,q_1)$ 满足 $q_1 < q$，那么 $Path$ 一定与 $Path_1$ 有交点；假设 $Path$ 经过了 $D(p,q_2)$ 满足 $q_2 > q$，那么 $Path$ 一定与 $Path_2$ 有交点。对于 $Path$ 从相交点开始，变换成与其相交路径后半部分走到 $D(p,q)$ 的部分，即就得到了 $D(1,a)$ 走到 $D(p,q)$ 的路径。同理，可以得到 $D(1,b)$ 走到 $D(p,q)$ 的路径。证毕。

□

综上所述，在这种情况下，$\mathrm{Mpoint}(a,b) = p$ 成立。上述的讨论过程，尤其是上面的证明部分中考虑相交路径的方法，在后文中多次用到，请读者引起注意。

直接做二维前缀最小值求出每个点对应的 (p,q)，就可以保证求所有 $\mathrm{Mpoint}(a,b)$ 的时间复杂度是 $O(HW + W^2)$ 即 $O(HW)$ 的了。

考虑如何 $\forall 1 \le a \le b \le W, 1 \le l \le H_D$ 快速查询 $\mathrm{Brh}(a,b,l)$。

显然，$\forall l > m = \min\{\mathrm{Bot}(a), \mathrm{Bot}(b)\}$，一定满足 $\mathrm{Brh}(a,b,l) = \mathrm{Brh}(a,b,m)$。所以下面我们只考虑 $l \le m$ 的情况。

$\mathrm{Mpoint}(a,b) > l$ 时，$\mathrm{Brh}(a,b,l) = 0$, 根据这两个函数的定义，这是显然的。

$\mathrm{Mpoint}(a,b) \le l \le m$ 时，就是在求满足 $x \le l$ 且 $\mathrm{Left}(x,y) \le a \le b \le \mathrm{Right}(x,y)$ 的 $D(x,y)$ 点权和，原因和求 $\mathrm{Mpoint}(a,b)$ 的讨论一样，在此不再赘述。这个东西的计算，考虑用容斥的思想，分成四个部分：

(1) 加上 $x \le l$ 且 $\mathrm{Left}(x,y) \le \mathrm{Right}(x,y)$ 的 $D(x,y)$ 点权和；

(2) 减去 $x \le l$ 且 $a < \mathrm{Left}(x,y) \le \mathrm{Right}(x,y)$ 的 $D(x,y)$ 点权和；

(3) 减去 $x \le l$ 且 $\mathrm{Left}(x,y) \le \mathrm{Right}(x,y) < b$ 的 $D(x,y)$ 点权和；

(4) 加上 $x \le l$ 且 $a < \mathrm{Left}(x,y) \le \mathrm{Right}(x,y) < b$ 的 $D(x,y)$ 点权和。

在从小到大枚举 l 的过程中，可以直接维护出每个 l 对应的 (1) 的结果；对 (2) 和 (3) 的查询用二维前缀和就可以做到 $O(1)$。对于 (4)，注意到满足 $a < \mathrm{Left}(x,y) \le \mathrm{Right}(x,y) < b$ 的 $D(x,y)$ 一定有 $x < \mathrm{Mpoint}(a,b) \le l$，这个性质像前面证明的时候一样直接考虑路径的相交就可以发现，所以就只是算 $a < \mathrm{Left}(x,y) \le \mathrm{Right}(x,y) < b$ 的 $D(x,y)$ 点权和了，条件和 l 无关，那么一开始也用二维前缀和预处理，就可以每次 $O(1)$ 查询了。所以在这些适当的预处理下，可以 $O(1)$ 查询 $\mathrm{Brh}(a,b,l)$。

根据定义可以得到，$\forall 1 \le a \le W, \mathrm{Reachable}(a) = \mathrm{Brh}(a,a,H_D)$。

在进行了大量预处理后，就可以进行对问题的求解了。

考虑用 $O(HW)$ 的 dp 对于每个 $U(i,j)$ 求出 $\text{Min}(i,j)$ 表示最小的 y，满足其可以到达 $D(1,y)$；$\text{Max}(i,j)$ 表示最大的 y，满足其可以到达 $D(1,y)$。忽视掉那些 $\text{Min}(i,j) > \text{Max}(i,j)$ 的点，即 $\text{Max}(i,j) = -\infty$ 且 $\text{Min}(i,j) = +\infty$、$U(i,j)$ 无法到达 D 中点的情况，那么将会有如下两个性质：

(1) 当固定一个 i_0 的时候，随着 j 的从小到大增加，$\text{Min}(i_0,j)$ 和 $\text{Max}(i_0,j)$ 均单调不降。即 $\text{Min}(i_0)$ 和 $\text{Max}(i_0)$ 有非严格单调性。

这里只提供证明的思路不再叙述具体的证明过程：考虑反证，假设一对违反条件的情况，以 $\text{Min}(i_0)$ 的一对逆序对为例，即 $j_1 < j_2$ 且 $\text{Min}(i_0,j_1) > \text{Min}(i_0,j_2)$，这时候考虑两条到对应点的路径必定相交，通过交换调整可以得到一条从 $U(i_0,j_1)$ 到 $D(1,\text{Min}(i_0,j_2))$ 的路径，显然与假设和 Min 函数的定义矛盾。按照这个思路同样可以证明 $\text{Max}(i_0)$ 的非严格单调性。

(2)$U(i,j)$ 可以到达 $D(1,y)$，当且仅当 $\text{Min}(i,j) \le y \le \text{Max}(i,j)$ 且 $\text{Top}(y) \le i$。原因和求 $\text{Mpoint}(a,b)$ 的讨论一样，在此不再赘述。

综上可以得到一个显然的做法，枚举 U 的每一行，然后从左到右扫描每一列，根据当前考虑点 $U(i,j)$，加入和删除一些 $D(1,y)$，维护当前考虑点可以到达的所有在 D 中点的点权和。

现在需要支持的事情是，维护一个类似队列的东西，每次在队列 Q 的后端加入一个 $D(1,y)$ 或是在前面删除一个 $D(1,y)$，同时求 Q 中可以到达点的点权和。定义 $\text{Only}(y)$ 表示：对于一个在 Q 中的 $D(1,y)$，其可以到达但是 Q 中所有列坐标大于 y 的点无法到达点的点权和。那么我们要查询的值就是 Q 中所有 $\text{Only}(y)$ 的和，下面考虑如何维护 $\text{Only}(y)$。前端删除是不会影响 $\text{Only}(y)$ 的，只有后端插入会产生影响。对于后端新插入的 $D(1,y_0)$，有 $\text{Only}(y_0) = \text{Reachable}(y_0)$；同时还有 Q 中的一些位置会发生更改。

假设 $y' < y'' < y_0$，那么所有 $D(1,y')$ 和 $D(1,y_0)$ 可以同时到达的前 $\min\{\text{Bot}(y'),\text{Bot}(y'')\}$ 行中的点，$D(1,y'')$ 都可以到达，这个性质也是像前面一样考虑路径的相交就可以证明。所以存在 y'' 满足 $y' < y'' < y_0$，同时 $\text{Bot}(y') \le \text{Bot}(y'')$ 的 $\text{Only}(y')$ 是不会发生改变的。所有可能发生改变的位置形成了一个序列 $J_1 < J_2 < \cdots < J_K$，根据前面的观察，它们一定满足 $\text{Bot}(J_1) > \text{Bot}(J_2) > \cdots > \text{Bot}(J_K)$，这实际上就是一个类似单调栈的结构，维护所有成为了严格后缀最大值的位置。对于所有的 $\text{Only}(J_i)$，有可能发生改变，从而变成新的 $\text{Only}'(J_i)$。

对于 J_K，有：

$$\text{Only}'(J_K) = \text{Only}(J_K) - \text{Brh}(J_K, y_0, \min\{\text{Bot}(J_K), \text{Bot}(y_0)\})$$

之后可以做一个向前递推的过程，当找到一个 $\text{Bot}(J_p) > \text{Bot}(y_0)$ 的 p 时，$\forall q < p, \text{Only}'(J_q) = \text{Only}(J_q)$，即都不会发生改变；

对于其它的 $p < K$，考虑简单的容斥可以得到，有：

$$\text{Only}'(J_p) = \text{Only}(J_p) - \text{Brh}(J_p, y_0, \min\{\text{Bot}(J_p), \text{Bot}(y_0)\}) + \text{Brh}(J_p, y_0, \min\{\text{Bot}(J_{p+1}), \text{Bot}(y_0)\})$$

注意到，当把 $D(1,y_0)$ 加入到 Q 末端时，J 这个序列会有一个后缀被删除，而这些点

和删除这些点后 J 的倒数第一个元素的 Only 才会发生改变，所以就像维护单调栈一样维护 J 即可，在维护的过程中顺便做出对 Only 的修改。注意对应到 Q 的前端删除，J 也会发生前端删除，所以具体实现的时候用双端队列来维护 J。

这样，每一行处理的时间复杂度都是 $O(W)$，总的时间复杂度是 $O(HW)$ 的。

综上所述，分治到每个大小为 $H \times W$ 的子矩形，进行处理的时间复杂度都是 $O(HW)$ 的，所以总的时间复杂度就是 $O(N^2 \log N)$ 的，可以通过本题。

3.8 总结

本题难度较大。对于一道背景是 DAG 上传递闭包这一经典问题的题目，如果不去深度挖掘与网格图有关的性质，考虑与网格图有关的做法，肯定是无从下手直到解决问题的。网格图分治的做法是显然的（这也并非本题的核心和难点），但是分治过程中的处理是比较难以思考的。本题中多次用到了关于路径相交后调整的思路，这使得我们在很多时候只需要考虑求出到达（或被到达）点的两个端点，再加上一些其它的必要条件，就可以得到一个点能到达另一个点的充分必要条件，而这样的条件只是在描述一些量的相对大小关系；而这样的性质还使得我们在有的时候可以简化问题，把一些复杂的条件简化成简单的条件，方便维护。通过观察和挖掘性质，DAG 上传递闭包这一相对困难的问题，放到网格图上就变得容易起来了。这样我们不仅可以预处理出很多求解问题所需要的信息，还可以用比较优秀的复杂度解决最后的维护问题。本题因为预处理很多所以代码显得比较冗长，实际上并没有什么细节，实现起来并不复杂。总之，本题是一道很优秀的好题，笔者十分喜欢。