IOI2018 国家集训队第一轮作业

杭州学军中学 毛嘉怡

October 30, 2017

Contents

1 Fountain Walk

1.1 试题来源

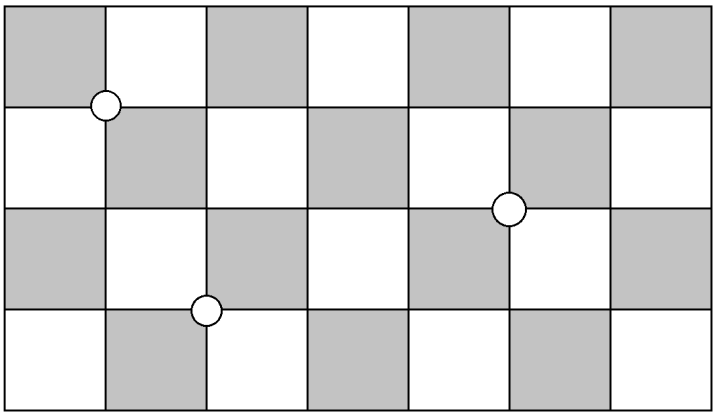
AtCoder Grand Contest 019 , Problem C

1.2 题目大意 & 数据范围

在一座城市当中，有 10^8 条东西方向的街道与 10^8 条南北方向的街道，都分别从 $0 \sim 10^8 - 1$ 编号。街道都是笔直的，且两条平行且相邻的街道之间的距离恰好为 100。

每条东西方向的街道与每条南北方向的街道都会相交，我们可以将一条编号为 x 的东西方向的街道与一条编号为 y 的南北方向的街道的交点上的路口表示为 (x, y)。

城市中一共有 N 个喷泉，第 i 个喷泉可以看做一个半径为 10、圆心位于 (X_i, Y_i) 的圆。当然，当一个位置有喷泉时，原本的道路就被覆盖而不能通行了，但你依旧可以沿着喷泉的边缘行走。下面是一个示例，可以行走的部分用实线描出。



[1]

每个路口上至多有一个喷泉，现在需要从 (x_1, y_1) 走到 (x_2, y_2)，你需要求出最短的路径长度。保证 (x_1, y_1) 与 (x_2, y_2) 不同，且都不是建有喷泉的路口。

- $0 \leq x_1, y_1, x_2, y_2 < 10^8$。
- $1 \leq N \leq 2 \times 10^5$。
- $0 \leq X_i, Y_i < 10^8$ 。
- 每条东西方向的街道上至多有一个喷泉，每条南北方向的街道上至多有一个喷泉。

[1] 图 1-1, 城市与喷泉的俯瞰结构示意图。

1.3 算法讨论

1.3.1 初步分析

组成最终答案的路径长度，由走过的东西、南北向街道段数、绕喷泉 180° 的次数、绕喷泉 90° 的次数共同决定。其中绕喷泉 180° 与无喷泉的情况相比，对路径长度贡献是正的，绕喷泉 90° 的贡献是负的。

首先不难发现，最终的答案路径当中不会出现环，即不会重复地经过某一个路口。我们可以考虑将连续两次经过某一个路口之间的路径删去，即使在这个路口处产生了最坏的影响——由绕 90° 转变为了绕 180°，这产生的代价 (5π) 也是远远小于删去的路径长度的。

通过观察和计算发现，刻意绕喷泉或避开单个喷泉产生的收益远远小于一段街道的长度，这促使我们产生一个猜想：

定理 1.1 街道在段数上取最小值，即 $|x_1 - x_2| + |y_1 - y_2|$ 时，总是能得到一个最优的路径方案。

证明 1.1 在此题中，旋转、翻转坐标系、交换起点与终点对答案显然都是不起影响的。方便起见，我们不妨只考虑从左下至右上的情况。当路径中只存在向右、向上走的边时，街道的段数只能取到 $|x_1 - x_2| + |y_1 - y_2|$。当在街道的段数上未取到最小值时，必然存在向左或向下走的情况。

首先我们证明从 (x_1, y) 至 (x_2, y) 的最短路径 $(x_1 \neq x_2)$ 无需变更方向。由于题中规定，一条东西向街道上至多有一个喷泉，因此直接走的距离

$d \leq 100|x_1 - x_2| + 10\pi - 20$。[2]

我们设一条从 (x_1, y) 经过若干次方向转变后到达 (x_2, y) 的路径所到达的 x 坐标极值分别为 x_{min}、x_{max}，其中 $x_{min} \leq \min(x_1, x_2) < \max(x_1, x_2) \leq x_{max}$。$y$ 坐标极值分别为 y_{min}、y_{max}，其中 $y_{min} \leq y \leq y_{max}$。这条路径需要经过的街道段数至少为 $2(x_{max} - \max(x_1, x_2) + \min(x_1, x_2) - x_{min} + y_{max} - y_{min}) + \max(x_1, x_2) - \min(x_1, x_2)$。绕过喷泉 180° 事件发生的最少次数为 0，绕过喷泉 90° 事件发生的最多次数为 $\min(y_{max} - y_{min}, x_{max} - x_{min})$。路径所要花费的距离

$d' \geq 200(x_{max} - \max(x_1, x_2) + \min(x_1, x_2) - x_{min} + y_{max} - y_{min}) + 100(\max(x_1, x_2) - \min(x_1, x_2)) + \min(y_{max} - y_{min}, x_{max} - x_{min})(5\pi - 20)$。[3]

我们令 $X = x_{max} - x_{min}$，$Y = y_{max} - y_{min}$。

$d' - d \geq 200(X + Y - |x_1 - x_2|) + \min(X, Y)(5\pi - 20) - 10\pi + 20$

- 当 $\min(X, Y) = X$ 时，

  $d' - d \geq 200Y + X(5\pi - 20) - 10\pi + 20 \geq (200 + 5\pi - 20)X - 10\pi + 20 = (180 + 5\pi)X - 10\pi + 20$。

  由于 $x_1 \neq x_2$，$X \geq 1$，有 $d' - d \geq 200 - 5\pi > 0$。

- 当 $\min(X, Y) = Y$ 且 $Y > 0$ 时，

  $d' - d \geq (180 + 5\pi)Y - 10\pi + 20 \geq 200 - 5\pi > 0$。

- 当 $\min(X, Y) = Y$ 且 $Y = 0$ 时，由于这条路径转变过方向，有 $X - |x_1 - x_2| > 0$。

[2] $10\pi - 20$ 是绕过一个喷泉 180° 对答案产生的贡献。

[3] $5\pi - 20$ 是绕过一个喷泉 90° 对答案产生的贡献。

$d' - d \geq 200(Y+1) + Y(5\pi - 20) - 10\pi + 20 = 220 - 10\pi > 0$。

这样我们证明了 $d' > d$ 恒成立，也就是 (x_1, y) 至 (x_2, y) 无需变更方向，类似地我们也可以证明 (x, y_1) 至 (x, y_2) 无需变更方向。

一条走过街道段数未取到最小值的路径，必定存在两个位于同一条街道的路口，且这两个街道间的路径并非笔直相连，此时我们将它们之间的路径修改为笔直相连一定不会使答案变劣（事实上我们在上面证明了，这样修改一定会使答案变优）。当任意两个相连且位于同一条街道的路口之间路径都不变更方向时，很容易发现路径上一定不存在向左或向下的移动，走过的街道段数于是必定取到最小值。

得到了定理 1.1 中的结论之后，问题转化为了给出起点、终点，终点位于起点右上方，规定只能往右、上两个方向走的最小路径长度。（当起点与终点位于同一条街道时，路径是唯一确定的，只需要简单地计算一下答案。）这样我们得到了一个以每个路口为状态、从两个方向转移的 DP 做法，时空复杂度均为 $O(|x_1 - x_2| \times |y_1 - y_2|)$。

我们要最大化绕喷泉 90° 事件的发生次数，同时最小化绕喷泉 180° 事件的发生次数。这很容易引导我们向最长上升子序列方向思考这道题。但是否转变方向、是否能够不经过任何喷泉地从一个路口到达另一个路口等问题使我们陷入了繁琐的细节。

1.3.2 发现重要边界：$\min(|x_1 - x_2|, |y_1 - y_2|) + 1$

继续沿着最小化绕喷泉 180° 次数的思路，我们通过观察一些简单的例子发现总是能够将它的次数控制在 ≤ 1 的范围内，那么这究竟是不是一个普遍适用的规律？在哪些情况下它的最小次数是 0、哪些情况下最小次数是 1 呢？

在我们接下来的讨论当中，约定 $x_1 \leq x_2$，$y_1 \leq y_2$（总是可以通过交换起点终点、翻转坐标系等方式等价地将原问题转化成这种情况）。

定理 1.2 我们定义一个喷泉的编号序列 $p_1, p_2 ... p_k$，序列中的元素均 $\in [1, N]$ 且两两不同。

一个序列满足条件，当且仅当 $X_{p_1} \geq x_1$、$Y_{p_1} \geq y_1$，$X_{p_k} \leq x_2$、$Y_{p_k} \leq y_2$，且对于任意 $i \in [1, k-1]$ 有 $X_{i+1} \geq X_i$、$Y_{i+1} \geq Y_i$。

起点到终点路径上的 LIS 定义为所有满足条件的序列当中长度 (k) 最大的一个序列 p。令 $len =$ 起点到终点路径上的 LIS 的长度。

当 $len = \min(|x_1 - x_2|, |y_1 - y_2|) + 1$ 时，对于任意一个从起点到终点的经过喷泉的合法序列 q，绕喷泉 180° 事件发生的次数至少为 1。

证明 1.2 当 $len = \min(|x_1 - x_2|, |y_1 - y_2|) + 1 = y_2 - y_1 + 1$ 时，起点到终点范围内每一条东西方向的街道上都有一个喷泉，如下图所示。

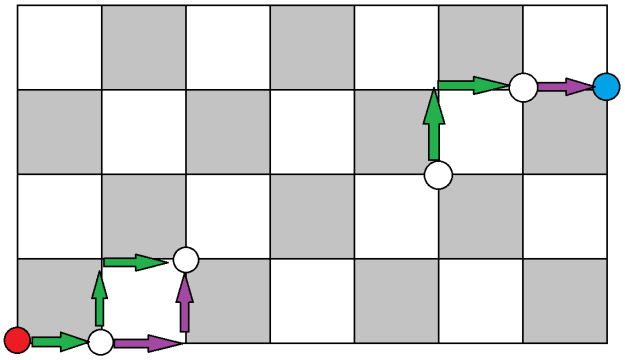
我们假设路径上经过的所有喷泉都是绕 90° 的方式。

我们在下面将绕喷泉 (X_i, Y_i) 90° 的方法分为两种：

- 左拐：$(X_i - 1, Y_i) \Rightarrow (X_i, Y_i) \Rightarrow (X_i, Y_i + 1)$。

- 右拐：$(X_i, Y_i - 1) \Rightarrow (X_i, Y_i) \Rightarrow (X_i + 1, Y_i)$

首先从起点到终点必定会经过至少一个喷泉，假设经过的第一个喷泉位于 (X_{q_1}, Y_{q_1})，要使经过的方式是绕 90°，只能通过左拐的方式。因为如果是通过右拐的方式，从 $(X_{q_1}, Y_{q_1} - 1)$ 走来，不难发现，从起点到 $(X_{q_1}, Y_{q_1} - 1)$ 之间的路径必然会经过至少一个喷泉，这与 (X_{q_1}, Y_{q_1}) 是经过的第一个喷泉矛盾。



假设我们经过了喷泉 (X_{q_i}, Y_{q_i}) 后，下一个经过的喷泉是 $(X_{q_{i+1}}, Y_{q_{i+1}})$。如果在喷泉 (X_{q_i}, Y_{q_i}) 处是左拐，那么在 $(X_{q_{i+1}}, Y_{q_{i+1}})$ 处也只能是左拐。

可以根据两个喷泉在 y 坐标上的差分两类讨论证明这一点：

- 当 $Y_{q_i} + 1 = Y_{q_{i+1}}$ 时，由于在 (X_{q_i}, Y_{q_i}) 处左拐，到达了 $(X_{q_i}, Y_{q_i} + 1) = (X_{q_i}, Y_{q_{i+1}})$，这时只能一直向右移动，$(X_{q_{i+1}}, Y_{q_{i+1}})$ 的上一个点只能是 $(X_{q_{i+1}} - 1, Y_{q_{i+1}})$，因此只能进行左拐。
- 当 $Y_{q_i} + 1 < Y_{q_{i+1}}$ 时，假设在 $(X_{q_{i+1}}, Y_{q_{i+1}})$ 处右拐，也就是从 $(X_{q_{i+1}}, Y_{q_{i+1}} - 1)$ 处走来，由于 $Y_{q_i} < Y_{q_{i+1}} - 1$，$(X_{q_i}, Y_{q_i} + 1)$ 与 $(X_{q_{i+1}}, Y_{q_{i+1}} - 1)$ 的路径上至少经过一个喷泉。这与经过 (X_{q_i}, Y_{q_i}) 后经过下一个的是 $(X_{q_{i+1}}, Y_{q_{i+1}})$ 矛盾。因此下一步只能进行左拐。

我们已经证明了第一步只能左拐，根据上面的证明运用数学归纳法可以得到路径上每一个经过的喷泉绕 90° 的方式只能是左拐。

假设经过的最后一个喷泉位于 (X_t, Y_t)，由于要进行左拐，满足 $Y_t < y_2$。左拐之后到达 $(X_t, Y_t + 1)$。而 $(X_t, Y_t + 1)$ 与 (x_2, y_2) 的路径上至少经过一个喷泉，这与 (X_t, Y_t) 是经过的最后一个喷泉矛盾。

因此在 $len = \min(|x_1 - x_2|, |y_1 - y_2|) + 1 = y_2 - y_1 + 1$ 的情况下，无法做到路径上经过的所有喷泉都是绕 90° 的。类似地我们也可以证明在 $len = \min(|x_1 - x_2|, |y_1 - y_2|) + 1 = x_2 - x_1 + 1$ 时也是如此。

证明了定理 1.2 后，我们很容易对于满足 $len = \min(|x_1 - x_2|, |y_1 - y_2|) + 1$ 条件的问题，构造出一个绕 180° 恰好一次的方案。即任意选择一个路径上的喷泉，在经过它时，不变更方向，这样在这个喷泉之前，所有经过的喷泉都可以左拐，在经过了这个喷泉之后，所有的喷泉都可以右拐。

结合定理 1.1 与定理 1.2。在 $len = \min(|x_1 - x_2|, |y_1 - y_2|) + 1$ 的情况下，我们确定了经过的街道的段数与绕喷泉 180° 的次数，只需要最大化绕喷泉 90° 的次数即可，这个值 = 经过的喷泉的总数 −1（以 180° 绕过的一个喷泉）。而最大化经过的喷泉总数，恰好满足 len 的定义。因此在这种情况的最优方案中，绕喷泉 90° 的次数为 $len - 1$。

至此，我们已经完全解决了 $len = \min(|x_1 - x_2|, |y_1 - y_2|) + 1$ 的情况。

定理 1.3 当 $len < \min(|x_1 - x_2|, |y_1 - y_2|) + 1$ 时，对于从起点到终点的任意一个合法的 LIS，都存在一种路径，符合 LIS 对路径上依次经过喷泉的描述，且绕喷泉 180° 的次数为 0。

证明 1.3 假设 $len < \min(|x_1 - x_2|, |y_1 - y_2|) + 1 = y_2 - y_1 + 1$

当 $len = 1$ 时，$x_2 - x_1 > 0, y_2 - y_1 > 0$。显然可以通过左拐或右拐的方式经过这个喷泉。

当 $len > 1$ 时，$len < y_2 - y_1 + 1$，因此必定存在一对在 LIS 序列中相邻的喷泉 (X_{LIS_i}, Y_{LIS_i})、$(X_{LIS_{i+1}}, Y_{LIS_{i+1}})$，满足 $Y_{LIS_i} + 1 < Y_{LIS_{i+1}}$。并且由 (X_{LIS_i}, Y_{LIS_i}) 作为左下角，$(X_{LIS_{i+1}}, Y_{LIS_{i+1}})$ 作为右上角的矩形中无除了这两个喷泉外的其他喷泉（否则与 LIS 的定义相矛盾）。我们可以令 $LIS_{1\sim i}$ 号喷泉都通过左拐的方式经过，在 LIS_i 号喷泉左拐后变更转弯方式：$(X_{LIS_i}, Y_{LIS_i}) \Rightarrow (X_{LIS_i}, Y_{LIS_i} + 1) \Rightarrow (X_{LIS_{i+1}}, Y_{LIS_i} + 1) \Rightarrow (X_{LIS_{i+1}}, Y_{LIS_{i+1}-1})$，并且途中不经过任何其他喷泉。然后 $LIS_{i+1\sim len}$ 号的喷泉都通过右拐的方式经过。

对于 $len < \min(|x_1 - x_2|, |y_1 - y_2|) + 1 = x_2 - x_1 + 1$ 的情况，我们也可以类似地构造出一个方案使得路径上经过每一个喷泉都是通过绕 90° 的方式。

证明了定理 1.3 后，对于 $len < \min(|x_1 - x_2|, |y_1 - y_2|) + 1$ 的情况，我们得到了一个同时最小化了绕喷泉 180° 的次数和最大化了绕喷泉 90° 的次数的方案。因此这种情况下的路径，绕 180° 的次数为 0，绕 90° 的次数为 len。

1.4 算法实现 & 题目总结

算法流程：

- 将读入的坐标通过交换起点终点、翻转坐标系等方法转化成能够让我们统一处理的某种位置关系。时间复杂度 $O(N)$。
- 求出 len。这里的求法可以看做是一个只考虑矩形内部的点的、按照 x 坐标排序后的 y 坐标序列的最长上升子序列。这个问题可以通过 $O(N \log_2 N)$ 的二分法求得，也可先离散坐标后使用数据结构在同样的时间复杂度内求得。
- 根据 len 与 $\min(|x_1 - x_2|, |y_1 - y_2| + 1)$ 的关系分类，并 O(1) 地计算出路径的长度。

总时间复杂度 $O(N \log_2 N)$，空间复杂度 $O(N)$。

这道题的主要难点在于需要充分利用题目中的性质得出结论、简化问题。从原来的一个结构复杂、较难进行筛选的答案集合不断地精简，直至最后：分类后的情况都能够直接得到一个确定的、形式简洁优美的答案。使用的算法简单好写，是一道仔细观察和分析后可以快速写出来的比较有趣的题。

2 Jigsaw

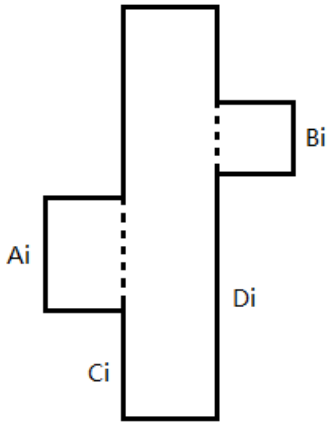
2.1 试题来源

AtCoder Grand Contest 017, Problem E

2.2 题目大意 & 数据范围

你得到了 N 块形状不规则的拼图，每一块拼图都是由三块宽度相同但高度可能不同的长方形拼图组合而成的。

更具体地说，第 i 块拼图由一块中间的高度为 H 的拼图，一块左边的高度为 A_i 的拼图和一块右边的高度为 B_i 的拼图组成。其中左边的拼图的底边比中间拼图的底边高 C_i，右边的拼图的底边比中间拼图的底边高 D_i。



我们现在要将这些拼图摆在一个可以视为无穷长的桌面上，同时需要满足下面的条件：

- 所有的拼图都需要放在桌面上。
- 所有拼图的中间部分的底边都需要完全贴在桌面上。所有拼图的左边、右边部分的底边都不能凌空。（也就是要求完全贴在桌面上或完全贴在别的拼图上）
- 拼图无法旋转或翻转。

你需要判断，是否存在满足条件的摆放拼图的方法。

- $1 \leq N \leq 10^5$，$1 \leq H \leq 200$。
- $1 \leq A_i, B_i \leq H$，$0 \leq C_i \leq H - A_i$，$0 \leq D_i \leq H - B_i$。
- 所有的输入数据均为整数。

2.3 算法讨论

2.3.1 初步分析

首先，我们整理一下题目中给出的条件，可以发现这道题是让我们判断是否存在这样一种方案：它通过编号序列的一个排列切成若干段后得到。并且每一段中需要满足:

- 设段首编号为 l，则 $C_l = 0$。
- 设段尾编号为 r，则 $D_r = 0$。
- 对于所有段内部相邻的两个编号 p 和 q，有 $D_p = A_q$ 且 $C_q = 0$，或 $B_p = C_q$ 且 $D_p = 0$。

我们发现两个拼图衔接的部分形式很整齐，考虑一个拼图 i 的左边部分，如果它的 $C_i = 0$，那么与它衔接的部分只需要满足一个与 A_i 的等量关系。如果它的 $C_i \neq 0$，那么与它衔接的部分的所有性质就与 A_i 没有任何关系了。不论是哪种情况，有用的实际上只有两个值中的一个。这使我们考虑精简信息:

我们为每个拼图的左边部分设计一个属性 P。当 $P_i = x(x > 0)$ 时，表示第 i 个拼图 $C_i = 0$，$A_i = x$。当 $P_i = x(x < 0)$ 时，表示第 i 个拼图 $C_i = -x$。数据范围保证了不存在 $P_i = 0$ 的情况。

类似地，我们为每个拼图的右边部分设计一个属性 Q。当 $Q_i = x(x < 0)$ 时，表示第 i 个拼图 $D_i = 0$，$C_i = -x$。当 $Q_i = x(x > 0)$ 时，表示第 i 个拼图 $D_i = x$。

接着我们把需要满足的性质转化到 P、Q 上。

- 设段首编号为 l，则 $P_l > 0$。
- 设段尾编号为 r，则 $Q_r < 0$。
- 对于所有段内部相邻的两个编号 i 和 j，有 $Q_i = P_j$。

我们惊喜地发现，原来较难处理的第三个条件借助 P、Q 写成了十分简单的形式。至此我们可以 $O(N^2)$ 地建出图，问题转化为了:

一张 N 个点 $O(N^2)$ 条边的有向图，判断是否能够用若干条起点和终点有一定限制的路径经过所有的顶点恰好一次。

但边数规模依然难以承受。

2.3.2 换一种建图思路

我们考虑建出序列 P、Q 值域大小个点，拼图 i 看作是一条从点 P_i 连向点 Q_i 的边。这时问题就巧妙地转化成了:

一张 $2H$ 个点 N 条边的有向图，判断是否能够用若干条起点和终点有一定限制的路径经过所有的边恰好一次。

由于点数与边数都在理想的范围内，经过每条边恰好一次的限制对我们来说也更为熟悉。我们不妨就使用这一种建图来继续下面的思考。

对于起点、终点的限制，更具体地来讲，是起点只能是权值 > 0 的点，终点只能是权值 < 0 的点。我们通过观察可以发现满足条件的图一定满足以下三条性质（在下面我们统一使用 ind_i 表示点 i 的入度，$outd_i$ 表示点 i 的出度）：

性质 2.1 任意正权点 i 满足 $ind_i \leq outd_i$。

性质 2.2 任意负权点 i 满足 $ind_i \geq outd_i$。

性质 2.3 在图的每一个连通分量当中，至少有一个点 i 满足 $ind_i \neq outd_i$。

性质 2.1、2.2 较为显然，因为正权点只能作为路径的起点或中间点，负权点只能作为路径的中间点或终点。点 i 每作为路径的起点一次，其 $ind_i - outd_i$ 减少 1；点 i 每作为路径的中间点一次，其 ind_i、$outd_i$ 各增加 1，因此它们的差不变；点 i 每作为路径的终点一次，其 $ind_i - outd_i$ 增加 1。当一个点作为路径的起点至少一次时，其出度一定大于入度。因此在一个所有点的入度与出度都相等的连通分量中，无法找到作为路径起点的点，因此一张满足条件的图一定满足性质 2.3。

定理 2.1 如果一张有向图 G 同时满足性质 *2.1*、*2.2*、*2.3*，则这张有向图能够用若干条满足条件的路径经过所有的边恰好一次。

证明 2.1 我们将一个同时满足性质 *2.1*、*2.2*、*2.3* 的连通分量称为 A 类图，将满足性质 *2.1*、*2.2* 但不满足性质 *2.3* 的连通分量称为 B 类图。

我们考虑从 A 类图中分离出一条路径或一个环，因为满足性质 *2.3* 的连通分量中至少有一条边，所以可以证明这是一定能做到的。由于满足性质 *2.3*，我们一定能从连通分量中找到至少一个点 s 满足 $outd_s > ind_s$，也一定能找到至少一个点 t 满足 $outd_t < ind_t$（一个连通分量内部的入度和与出度和相等）。我们选择其中一个出度大于入度的点作为起点进行 *DFS*。设当前点为 u 会出现以下几种情况：

- u 未被访问过且 $ind_u < outd_u$：这时有 $outd_u > 0$，任意选择一个出边继续 *DFS*。
- u 未被访问过且 $ind_u = outd_u$：这时有 $outd_u > 0$（当 $ind_u = outd_u = 0$ 时，该点为孤立点，不可能存在于一个至少有一条边的连通分量当中），任意选择一个出边继续 *DFS*。
- u 未被访问过且 $ind_u > outd_u$：这时我们已经找到了一个负权点，将其作为路径的终点并结束 *DFS*。成功分离出一条满足条件的路径。
- u 被访问过：结束 *DFS*。从上一次 u 被访问至当前所经过的边首尾相连，成功分离出一个环。

当我们从一个 A 类图中分离出一条路径或一个环后，分析顶点度数的变化不难发现，这个 A 类图可能会变成若干个（可能是 *0* 个）A 类图和若干个（可能是 *0* 个）B 类图。A 类图我们可以继续进行分离，直至最后原图变成了若干个 B 类图 + 若干条路径 + 若干个环。

我们接下来考虑 B 类图，当一个 B 类图中没有边时显然不需要考虑。当一个 B 类图中至少有一条边时，由于对于图中所有的顶点都有入度 = 出度，那么这个连通分量内部一定存在一个欧拉回路[4]。我们暂时将这条欧拉回路放在一边。

对于原图的一个连通分量，通过上面的操作，我们已经将它所有的边分离成了若干条路径和若干条欧拉回路（在最初分离出的环显然也可以看做是一条欧拉回路）。假设当前还有 x 条欧拉回路：

[4]有向图欧拉回路存在的条件：（1）图连通。（2）每个点的入度 = 出度。

- $x = 0$：该连通分量已经被完全分离成了若干条满足条件的路径，结束过程。
- $x > 0$：选择其中的一条欧拉回路，由于该连通分量属于 A 类图，这条欧拉回路上至少有一个点在原图的入度 $\neq$ 出度，也就是至少有一条连入（出）这条欧拉回路的边。找到那条边当前所在的路径（或欧拉回路），将当前这个欧拉回路加进那条路径（或欧拉回路）中，就会产生一条更长的路径（或一条更长的欧拉回路）。进行了这样一次合并操作后，x 减少 *1*。

上面的过程证明了对于一个 A 类图，我们总能将它分离成若干条满足条件的路径。由于原图 G 的每一个连通分量都是 A 类图，所以原图 G 也能被分离成若干条满足条件的路径。

至此，我们完全解决了这道题。

2.4 算法实现 & 题目总结

算法流程：

- 建出图，并统计每个点的入度与出度。时间复杂度 $O(N + 2H)$。
- 判断性质 2.1、2.2 是否被满足。时间复杂度 $O(2H)$。
- 建出有向图连通分量，并判断性质 2.3 是否被满足。这一步可以通过并查集来实现，时间复杂度为 $O(N\alpha(2H))$。也可以直接通过 DFS 或 BFS 来遍历连通分量，时间复杂度为 $O(N + 2H)$。

总时间复杂度为 $O(N + 2H)$。

这道题看上去是一道较为简单的结论题，但事实上它较好地考察了选手的图论功底和欧拉回路方面的知识。结论的形式是选手容易想到的方向，总体十分清新，比较巧妙。

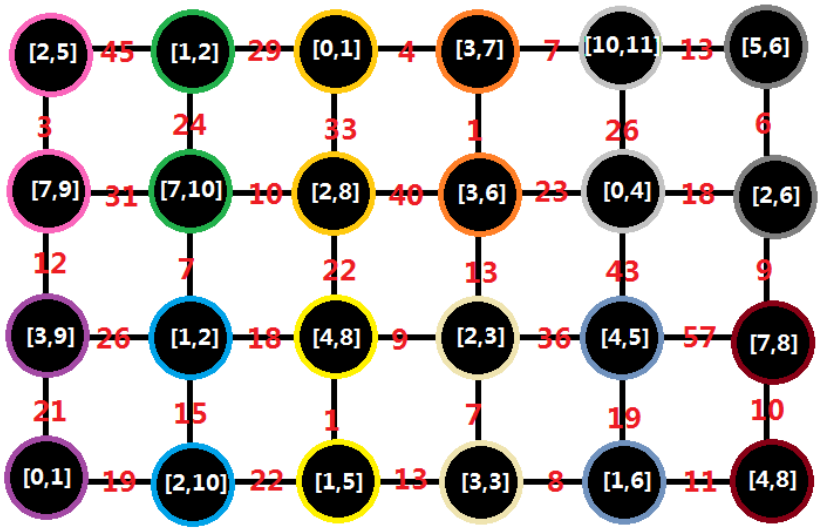
3 Bubble Gum

3.1 试题来源

Original problem by Yuhao Du [5]

3.2 题目大意 & 数据范围

给出一个有着 $m \times n$ 个点的网格图代表一座城市。一共有 m 行 n 列点，每个点表示一个路口。每两个在水平方向或竖直方向相邻的路口间都有一条街道相连，每条连接两个路口的街道之间都是平行或垂直的关系。每条街道上都有一个数字，每个路口都有一个数字区间。下面是一个有 4×6 个点的网格图示例。



对于每一个路口，你都可以在它的数字区间中选一个数作为权值。一条街道的权值计算方法为它所连接的两个路口上权值之差的绝对值乘上这条街道上的数字。

你需要求出所有街道的权值和的最小值是多少。

- $1 \leq n \leq 5 \times 10^4$，$1 \leq m \leq 5$。
- $0 \leq$ 路口的数字区间左端点，街道的权值。
- 令 $T = \max($ 路口的数字区间右端点，街道的权值 $)$，$T \leq 10^4$。

3.3 算法讨论

观察这道题，发现如果要枚举出所有的情况来更新答案，枚举规模达 $O(T^{nm})$，在 T 没有特殊限制的情况下，这样的复杂度都是无法承受的。

[5] Used in mjy0724's NOIp Training Contest

3.3.1 算法一

但注意到，m 的范围比较小，且对于一个路口而言只有相邻路口的权值需要记录。我们可以使用一个比较暴力的 DP 来降低复杂度：

$f_{i,S}$ 表示当前确定了 $1 \sim i$ 行路口的权值，其中第 i 行的路口的权值通过某种方式压起来的状态为 S 时，$1 \sim i$ 行的街道总权值最小是多少。

这个做法利用了题目中街道的权值计算方法的性质，时间复杂度为 $O(nT^{2m})$。

3.3.2 算法二

我们发现准确记录权值事实上还来自于绝对值计算的要求。因为如果没有绝对值符号，每个路口权值的贡献就是独立的了。这使我们想到了按照权值的大小顺序进行确定：

$f_{i,j}$ 表示当前确定了权值在 $[0,i]$ 区间内的所有点，其中每个点是否被选过的状态为 j 时，已经确定了权值的路口对于答案的贡献最小是多少。

这样一来，根据 DP 状态中的第二维可以直接得到当前确定的点在哪些街道上贡献是正的权值、哪些街道上贡献的是负的权值，每个路口对答案的贡献就可以独立地计算了。时间复杂度为 $O(2^{nm}Tnm)$。

3.3.3 算法三：$m = 1$ 时的做法

$m = 1$ 时，路口排列成一条直线，只有相邻的之间有影响。因此只需要记录单个点的权值即可方便地统计贡献。考虑：

$f_i(x)$ 表示第 i 个路口的权值确定为 x 时，路口 $1 \sim i$ 间街道的权值总和最小是多少。

直接计算的复杂度是 $O(nT^2)$。

我们考虑优化这个过程，假设当前路口 i 确定了一个权值 x，路口 i 与路口 $i-1$ 间的街道权值为 c_i，对于 $f_{i-1}(j)(j \in [0,x])$，转移到 $f_i(x)$ 是 $f_{i-1}(j) + c_i(x-j)$。因此我们只需要找到 $f_{i-1}(j) - c_i \times j$ 最小的答案转移过来即可，而这个值与 x 并无关系。我们对于每个路口维护这个 $f_{i-1}(j) - c_i \times j$ 的最小值，随着 x 的增大，用新满足条件的 j 来更新最小值。对于 $j \in [x+1,T]$ 的部分类似地倒过来做即可。这样，时间复杂度降为 $O(nT)$。

我们考虑继续优化这个过程：

$$f_i(x) = \min_{x'}\{f_{i-1}(x') + c_i|x - x'|\}$$

其中 $l_i \le x \le r_i$。

定理 3.1 $f_i(x)$ 是一个关于 x 的凸函数，其中 $x \in [l_i, r_i]$。

证明 3.1 $f_i(x)$ 显然关于 x 连续。令 M 为一个足够大的正实数。

令 $f_i'(x)$ 为 $f_i(x)$ 的右导数。当 $x \ge r_i$ 时，我们定义 $f_i'(x) = M$。当 $x < l_i$ 时，定义 $f_i'(x) = -M$。

我们只需要证明 $f_i'(x)$ 递增即可。

对 i 归纳，$i=1$ 的时候，$f_i(x)=0$，显然。

假设 $f_{i-1}'(x)$ 递增，我们需要推出 $f_i'(x)$ 的递增性。我们这里先假设 $f_i(x)$ 的定义域为 $\mathbb{R}$，然后将这个函数的定义域限制在 $[l_i,r_i]$ 上，并不改变 $f_i'(x)$ 的递增性。

记 $g(y)=f_{i-1}(y)+c_i|x-y|$，因为 f_{i-1} 与 $c_i|x-y|$ 都是凸函数，所以 $g(y)$ 也是个凸函数。同样定义 $g'(y)$ 为 $g(y)$ 的右导数，$g'(y)$ 递增。记 y_0 为满足 $g'(y)\geq 0$ 的最小的 y。由于 g 的凸性，那么 g 在 y_0 处取到最小值。

$$g_i'(y)=\begin{cases} f_{i-1}'(y)+c_i & y\geq x \\ f_{i-1}'(y)-c_i & y<x \end{cases}$$

所以当 $f_{i-1}'(x)\in[-c_i,c_i)$ 的时候，$g_i'(x)=f_{i-1}'(x)+c_i\geq 0$，且当 $y<x$ 时，$g_i'(y)=f_{i-1}'(y)-c_i\leq f_{i-1}'(x)-c_i<0$，所以 g 在 x 处取到最小值，即 $f_i(x)=f_{i-1}(x)$。

令 x_0 为满足 $f_{i-1}'(x)\geq c_i$ 的最小的 x，那么当 $f_{i-1}'(x)\geq c_i$ 时，当且仅当 $x\geq x_0$。$g_i'(x_0)=f_{i-1}'(x_0)-c_i\geq 0$，且 $y<x_0$ 时，$g_i'(y)\leq f_{i-1}'(y)-c_i\leq f_{i-1}'(x_0)-c_i<0$，所以 $f_i(x)=f_{i-1}(x_0)+c_i(x-x_0)$。

同理令 x_1 为满足 $f_{i-1}'(x)<c_i$ 的最大的 x，当 $x<x_1$ 时，$f_i(x)=f_{i-1}(x_1)+c_i(x_1-x)$。

求导得，

$$f_i'(x)=\begin{cases} -c_i & f_{i-1}'(x)\in(-\infty,-c_i) \\ f_{i-1}'(x) & f_{i-1}'(x)\in[-c_i,c_i) \\ c_i & f_{i-1}'(x)\in[c_i,+\infty) \end{cases}$$

所以 $f_i'(x)$ 也是递增的。

根据上面的证明，我们可以得出算法流程。$f_i'(x)$ 是一个分段函数，可以用一个双端队列维护。从 $f_{i-1}'(x)$ 转移到 $f_i'(x)$，将 $f_{i-1}'(x)$ 两端绝对值超过 c_i 的段改成 c_i 或者 $-c_i$，然后将这个函数的范围限制在 $[l_i,r_i]$ 之内。

这样，我们把时间复杂度降到了 $O(n)$。

3.3.4 算法四：路口权值 $\in[0,1]$ 时的做法

设第 i 行第 j 列路口的权值区间为 $[l_{i,j},r_{i,j}]$，我们在这里要讨论就是一类对任意 $i\in[1,m]$，$j\in[1,n]$，都有 $0\leq l_{i,j}\leq r_{i,j}\leq 1$ 的特殊情况的解法。

0 或 1 的取舍很容易令我们想到最小割，在下面的描述中，用 $(x\Rightarrow y,z)$ 来表示从点 x 向点 y 连一条容量为 z 的边。

- 我们建立一个源 s，汇 t 和 $m\times n$ 个代表每一个路口的点，我们用 $v_{i,j}$ 来表示代表第 i 行第 j 列路口的点。约定分到 S 集的点权值取 0，分到 T 集的点权值取 1。
- $l_{i,j}=0,r_{i,j}=0$：只能分到 S 集，$(s\Rightarrow v_{i,j},+\infty)$

- $l_{i,j}=1, r_{i,j}=1$：只能分到 T 集，$(v_{i,j} \Rightarrow t, +\infty)$
- 对于两个相邻的路口 $v_{i,j}$，$v_{i+1,j}$。设连接这两个路口的街道的权值为 val，可以连：$(v_{i,j} \Rightarrow v_{i+1,j}, val), (v_{i+1,j} \Rightarrow v_{i,j}, val)$。
- 对于两个相邻的路口 $v_{i,j}$，$v_{i,j+1}$。设连接这两个路口的街道的权值为 val，可以连：$(v_{i,j} \Rightarrow v_{i,j+1}, val), (v_{i,j+1} \Rightarrow v_{i,j}, val)$。

这样我们跑一遍最大流就可以计算出答案。时间复杂度 $O(MaxFlow(n \times m, n \times m))$。

我们发现，在这种特殊情况下，T 降到了 2。使用算法一，我们能得到一个 $O(n4^m)$ 的做法，使用算法二，我们能得到一个 $O(2^{nm+1}nm)$ 的做法。我们发现，在 n 比较大的情况下，算法二是很难进行优化的，因此我们不妨来考虑优化算法一。

算法一的劣势主要在于我们以一行作为了考虑对象，这导致我们在枚举该行状态的同时为了转移还需要枚举上一行的状态。如果我们一个个路口进行确定和转移，要保存和记录的状态大小是一样的，但枚举当前状态时的规模却大大地降了下来。更具体地说：

我们从左到右、从上到下地确定每一个路口的权值，确定一个权值时统计该路口与其上、左两个方向上相邻路口间街道的权值。这样，我们只需要记录该路口的前 m 个路口（按照确定的顺序）即可。令 $f_{i,S}$ 表示考虑了前 i 个路口，第 $i-m+1 \sim i$ 个路口的权值压成的状态为 S 时答案的最小值。

时间复杂度降到了 $O(nm2^m)$。

3.3.5 算法五

首先我们考虑任意图，记 x_u 表示 u 节点的标号，对一条边 (u,v) 用一个变量 $d_{(u,v)}$ 表示这两个点之间差的绝对值，那么有限制 $d_{(u,v)} \geq x_u - x_v, d_{(u,v)} \geq x_v - x_u$。不特殊说明，所有的变量都是非负的。

所以可以得到线性规划的式子

$$\begin{array}{lrll}
\textbf{minimize} & \sum c_{(u,v)} d_{(u,v)} & & \\
\textbf{subject to} & x_u & \geq l_u & \forall u \in V \\
& -x_u & \geq -r_u & \forall u \in V \\
& d_{(u,v)} - x_u + x_v & \geq 0 & \forall (u,v) \in E \\
& d_{(u,v)} - x_v + x_u & \geq 0 & \forall (u,v) \in E
\end{array}$$

对偶得

$$\begin{array}{lrll}
\textbf{maximize} & \sum l_u p_u - r_u q_u & & \\
\textbf{subject to} & p_u - q_u & \leq \sum_{(u,v)\in E} (y_{(u,v)} - y'_{(u,v)}) - \sum_{(v,u)\in E} (y_{(v,u)} - y'_{(v,u)}) & \forall u \in V \\
& y_{(u,v)} + y'_{(u,v)} & \leq c_{u,v} & \forall (u,v) \in E
\end{array}$$

记 $f_{(u,v)} = y_{(u,v)} - y'_{(u,v)}$，那么有 $-c_{(u,v)} \leq f_{(u,v)} \leq c_{(u,v)}$。

上式变成 $p_u - q_u \leq \sum_{(u,v)\in E} f_{(u,v)} - \sum_{(v,u)\in E} f_{(v,u)}$。

因为我们要最小化 $l_u p_u - r_u q_u$，所以如果上式的右端是负的，那么答案就是在 p_u 取到 0 的时候最小，如果是正的，那么就是 q_u 取到 0 的时候最小。

我们把 $f_{(u,v)}$ 当成从 u 点到 v 点的流量，那么右端为正等价于这个点的流出大于流入，那么从源点补流量，费用为 l_u，否则向汇点流流量，费用为 $-r_u$。

整体的建图如下。在下面的描述中，我们用 $(x \Rightarrow y, z, w)$ 表示由 x 向 y 连一条容量为 z，费用为 w 的边。

- 对于两个有街道相连的路口 u, v，$(u \Rightarrow v, c_{u,v}, 0)$，$(v \Rightarrow u, c_{u,v}, 0)$。
- 对于每个点 u，$(s \Rightarrow u, +\infty, l_u)$，$(u \Rightarrow t, +\infty, -r_u)$。

对上面的图跑最大费用流即可，时间复杂度 $O(MaxCostFlow(n \times m, n \times m))$。

3.3.6 算法六

回顾算法四的最小割思想，是将所有的路口分成了权值为 0 和权值为 1 的两边，相邻的路口之间的贡献也随着它们被分到了不同的集合里而体现在了被割掉的边上。那么对于更普通的情况，我们是否也可以使用这个方法呢?

首先不难发现，一定存在一种能达到最优解的答案满足，所有路口确定的权值都在读入的路口的左右端点处出现过（不一定是该路口的端点）。因此，在接下来的讨论中，我们只需要考虑将路口的权值左右端点离散后的的那 $O(n \times m)$ 个取值。

我们令离散后的权值序列为 $p_1, p_2 ... p_k$，一个直观的想法是：我们是否可以把每个 $p_i, p_{i+1} (i \in [1, k-1])$ 分别看作算法四中的 0 与 1，将权值区间整体 $\leq p_i$ 的点与源点连上容量为 $+\infty$ 的边，将权值区间整体 $> p_i$ 的点与汇点连上容量为 $+\infty$ 的边，各跑一遍最小割累计答案呢? 我们将一次统计 p_i, p_{i+1} 之间贡献的最小割称为 $Cut(i)$。在这次最小割中，被分到 S 集的可以看做是权值 $\leq p_i$ 的点，被分到 T 集的可以看做是权值 $> p_i$ 即 $\geq p_{i+1}$ 的点。两个被分到不同集合的点之间一条容量为 $p_{i+1} - p_i$ 的边被割开。

定理 3.2 *存在一种 $Cut(1 \sim k-1)$ 的最小割方案的组合，使得对于所有的点 u 一定满足下列两种情况中的一种：*

- *$\forall i \in [1, k-1]$，在 $Cut(i)$ 中，$u \in S$。*
- *$\exists i \in [1, k-1]$，在 $Cut(1 \sim i)$ 中，$u \in T$。在 $Cut(i+1 \sim k-1)$ 中，$u \in S$。*

证明 3.2 *换一种说法，即存在一种最小割方案，使 S 集元素只增不减。即对于任意 $i, j \in [1, k-1] (i < j)$，令 A 为 $Cut(i)$ 中的 S 集，B 为 $Cut(j)$ 中的 S 集，都有 $A \subset B$。*

我们假设在某两次相邻的最小割中，前一次最小割的 S 集为 S_1，T 集为 T_1；当前最小割的 S 集为 S_2，T 集为 T_2。令 $P_s = S_1 \cap S_2$，$P_t = T_1 \cap T_2$，$P_a = S_1 \setminus P_s$，$P_b = S_2 \setminus P_s$。则 $S_1 = P_s \cup P_a$，T 集 $T_1 = P_t \cup P_b$，当前最小割的 S 集 $S_2 = P_s \cup P_b$，T 集 $T_2 = P_t \cup P_a$。定义 $d'(U, V)$ 表示在前一次最小割图中由集合 U 中的点连向集合 V 中的点的边的容量和，$d(U, V)$ 表示在当前最小割图中由集合 U 中的点连向集合 V 中的点的边的容量和。$C'(U, V)$ 表示在前一次最小割图中以集合 U 为 S 集，集合 V

为 T 集时的最小割答案，$C(U,V)$ 表示在当前最小割图中以集合 U 为 S 集，集合 V 为 T 集时的最小割答案。

$\because C'(P_s \cup P_a, P_t \cup P_b) \le C'(P_s, P_a \cup P_b \cup P_t)$

$\therefore d'(P_s,P_t)+d'(P_s,P_b)+d'(P_a,P_t)+d'(P_a,P_b) \le d'(P_s,P_a)+d'(P_s,P_b)+d'(P_s,P_t)$

$\Rightarrow d'(P_a,P_t)+d'(P_a,P_b) \le d'(P_s,P_a)$

$\Rightarrow d'(P_a,P_t) \le d'(P_s,P_a)$

$d'(P_s,P_t)+d'(P_a,P_t) \le d'(P_s,P_t)+d'(P_s,P_a) \le d'(P_s,P_t)+d'(P_s,P_a)+d'(P_b,P_a)$

随着基准的权值的变化，按照我们连边的方式，相邻的两次最小割的图上只会发生以下两种变化：

- 删除点与汇点间容量为 $+\infty$ 的边。
- 添加源点与点间容量为 $+\infty$ 的边。

根据上面不难看出，$d(P_b,P_a)=d'(P_b,P_a)$。

由于 $P_a,P_s \subset S_1$，其中所有点的权值区间在前一次最小割时与汇点间容量为 $+\infty$ 的边已经不存在，所以当前最小割时 P_a,P_s 与汇点间不可能有容量为 $+\infty$ 的边。因此 $d(P_a,P_t)=d'(P_a,P_t), d(P_s,P_t)=d'(P_s,P_t)$。另外，由于 $S_2<+\infty$，在当前最小割中 $d(P_s,P_a)<+\infty$，因此在前一次最小割中 $d'(P_s,P_a)<+\infty$，也就是 $d(P_s,P_a)=d'(P_s,P_a)$。

因此在当前的最小割图中：

$d(P_s,P_t)+d(P_a,P_t) \le d(P_s,P_t)+d(P_s,P_a)+d(P_b,P_a)$

由于 P_b,P_t 在当前最小割中被割开，因此有 $d(P_b,P_t)<+\infty$

$\Rightarrow d(P_s,P_t)+d(P_a,P_t)+d(P_b,P_t) \le d(P_b,P_t)+d(P_s,P_t)+d(P_s,P_a)+d(P_b,P_a)$

$\Rightarrow C(P_s \cup P_a \cup P_b, P_t) \le C(P_s \cup P_b, P_t \cup P_a)$

因此，在当前最小割中，令 S 集为 $P_s \cup P_a \cup P_b$，T 集为 P_t 后总能在不改变最小割性质的前提下使 $S_1 \subset S_2$ 成立。

有了定理 3.2，就很容易证明按照上述方式跑 $k-1$ 次最小割能够得到正确答案了。我们设正确答案为 ans，上述最小割算法求出的答案是 $mincut$。

首先对于求出的一组 $Cut(1 \sim k-1)$ 的最小割方案，我们根据定理 3.2 总可以得到一组 S 集只增不减的方案。我们设一个点第一次被分到 S 集是在 $Cut(i)$ 中，我们就令这个点的权值为 p_i。对于这一种权值的分配方案，很显然上述最小割算法能够正确地得到答案，因此 $ans \le mincut$。

考虑最优解时每个点的权值分配方案，当一个点的权值是 p_x 时，我们令这个点在 $Cut(1 \sim x-1)$ 中都在 T 集，在 $Cut(x \sim k-1)$ 中都在 S 集，分析建图很容易发现这 $k-1$ 个割都是合法的。因此有 $mincut \le ans$。

综上，$ans = mincut$。我们按照这样的方式跑 $k-1$ 次最小割即可得到正确答案。时间复杂度为 $O(nmMaxFlow(n \times m, n \times m))$。

然后我们考虑优化这个算法。

类似算法四中的思路，我们也可以从 DP 入手。对于每一次最小割：

我们从左到右、从上到下地确定每一个路口的权值。定义 $f_{i,j}$ 表示考虑了前 i 个路口，第 $i-m+1 \sim i$ 个路口分别被分到 S 集或 T 集的信息压成的状态为 j 时答案的最小值。

时间复杂度降为 $O(n^2m^22^m)$。

3.3.7 算法七

我们已经能够较高效地求出单次的最小割了，接下来的优化方向很明显：快速地从前一个状态的最小割得到下一个状态的最小割。

在算法六中也有提到，当基准权值变成 p_x 时，最小割图会发生的变化有：

设点 u 的权值区间为 $[l_u, r_u]$，

- 当 $l_u = p_x$ 时，删除点 u 连向 t 的容量为 $+\infty$ 的边。
- 当 $r_u = p_{x-1}$ 时，添加点 s 连向 u 的容量为 $+\infty$ 的边。

不难发现，整个过程当中加/删边的操作总数是 $O(nm)$ 的，因为一个点至多只会加边一次、删边一次。另外，当一个点自己未加/删边以及与其相邻的点的位置都未发生变化时，它的位置显然也不会发生变化。这使我们想到了使用线段树来维护一个区间内的网格图的状态。

在一个代表区间 $[l, r]$ 线段树节点上，我们维护：

- $i \in [1, m]$，$j \in [l, r]$ 的路口间的最小割
- $i \in [1, m]$，$j = l$ 的路口的状态
- $i \in [1, m]$，$j = r$ 的路口的状态

进行合并操作时，只需要计算中间两排点之间产生的割。每次加/删边操作可看作一个单点修改，每次统计最小割可看作一个全局询问。时间复杂度为 $O(nm \log_2 n2^{4m})$。

3.3.8 算法八

在算法六中我们已经证明了存在最小割方案满足 S 集不减，这使我们想到了利用决策单调性维护 S 集中的点来解决这道题。

我们设计一个确定 $Cut(l+1 \sim r-1)$ 中的 S 集，其中待确定的点集为 P 的分治过程 $solve(P, l, r)$：待确定的点集意义为：在 $Cut(l)$ 中 P 集中的点都属于 T 集，在 $Cut(r)$ 中 P 集中的点都属于 S 集。

- 当 $l + 1 < r - 1$ 时，区间内没有需要确定的最小割，结束过程。
- 令 $mid = (l + r)/2$，算出 P 中的点在 $Cut(mid)$ 里被分到了哪一边。

- 我们令 S_{mid} 等于 $Cut(mid)$ 中 P 集被分到 S 集的点集，T_{mid} 等于 $Cut(mid)$ 中 P 集被分到 T 集的点集。那么在 $Cut(l+1, mid-1)$ 当中，待确定的点集就是 S_{mid}；在 $Cut(mid+1, r-1)$ 当中，待确定的点集就是 T_{mid}。
- $solve(S_{mid}, l, mid), solve(T_{mid}, mid, r)$。

在上面的过程中，只要记录下一个点最晚被分到 T 集的时刻，就能够方便地直接算出这个点的权值。得到了方案之后就可以直接根据定义计算答案了。

与之前的算法略有不同的是，我们为了保证复杂度，在 DP 最小割的时候不能够计算所有的点。由于我们只需要确定 P 集中的点在最小割中的状态，可以稍加修改设计下面一个 DP：

从上到下、从左到右地确定每一个 P 集中的点的状态，$f_{i,j}$ 表示考虑完了 P 集中的前 i 个点，其中 $i-m+1 \sim i$ 个点在最小割中的信息压成的状态为 j 时答案的最小值。

在确定一个点的状态时，先根据它的权值区间与基准权值 (p_{mid}) 的关系来判断它是否能被分配到 S、T 集。在对一个合法的分配计算增加的贡献时，我们对其上、左方向相邻的点是否在 P 集进行讨论，如果在 P 集当中，可以根据它们在 j 中的状态计算贡献；不在 P 中的点 u 一定满足下面情况中的一种：

- 在 $Cut(r)$ 中被分到了 T 集：这时一定有一种方案满足点 u 在 $Cut(1 \sim r)$ 中均在 T 集，因此 u 在 $Cut(mid)$ 中可以被确定为分在 T 集。
- 在 $Cut(l)$ 中被分到了 S 集：这时一定有一种方案满足点 u 在 $Cut(l \sim k-1)$ 中均在 S 集，因此 u 在 $Cut(mid)$ 中可以被确定为分在 S 集。

因此也可确定状态并计算贡献。值得注意的是，与前两次的 DP 不同，一个路口相邻的四个方向中，在之前由于每个点都会被计算一次，每次只需要计算两个方向的贡献即可。但当一个 P 集中的点，其右、下方向的点不在 P 集中时，这些方向的贡献就会被漏计。这种情况下，不要忘记加上这部分的贡献即可。

如果我们将 $solve(P, l, r)$ 之间调用的关系看成一棵树，那么每个点在同一层的节点当中显然至多只会出现一次。树高是 $O(\log_2 k)$ 的，因此 $\sum |P|$ 是 $O(nm \log_2 k)$ 的。对于一次 $solve(P, l, r)$ 操作，其时间复杂度瓶颈在 DP，是 $O(|P|2^m)$ 的，又因 k 是 $O(nm)$ 的，因此总复杂度为 $O(nm \log_2(nm) 2^m)$，完全可以通过这道题。

3.4 题目小结

这道题作为一道 OI 题，在部分分和特殊限制的设置上尽可能地鼓励了用多种做法来解决这道题的不同方面。典型的部分分做法有状压 DP、维护分段函数、最小割、线性规划、LP 对偶后跑费用流等，迭代、退火等近似算法在这道题中也能得到一定的分数。在标算与类似标算的解法中，需要选手通过观察和分析得到最小割方案间的性质，并以此为突破口，加速运算。优化的方向主要有维护修改和整体划分到部分确定两种，分别对应了算法七中的线段树做法与算法八中的分治做法。总的来说是一道想出标算的思维要求稍高，但整道题考察的广度很不错的题。