(LIS, LDS) - Coordinates 解题报告

宁波市镇海中学 刘恒熙

November 12, 2024

1 题目大意

对于一个排列 p，按照以下方式定义 $f(p)$：

- 对于排列中的一个元素 p_i，令 a_i 为以其结尾的最长上升子序列长度，b_i 为以其开头的最长下降子序列长度，定义其**坐标**为**有序**二元组 (a_i, b_i)。
- $f(p)$ 为 p 中所有元素的**坐标**构成的集合。

例如，$f(\{1,2,5,4,3,6\}) = \{(1,1),(2,1),(3,1),(3,2),(3,3),(4,1)\}$。

给定正整数 n 和 n 个互不相同的**有序**二元组 $(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)$，其中 x_i, y_i 均为不超过 n 的正整数。

构造一个长度最小的排列 p，使得 $f(p)$ 包含**所有** n 个给定的二元组，$f(p)$ 可以包含 n 个给定的二元组以外的其他元素。

若有多个长度最小的排列，输出任意一个。可以证明，在给定条件下，总是存在一个长度不超过 $3n$ 的排列。

若你输出的排列不是长度最小的排列，但是长度不超过给定参数 lim，也可获得部分分数。

2 数据范围

对于所有数据，保证 $1 \le n \le 5000, 3n \le \text{lim} \le 15000, 1 \le x_i, y_i \le n$。

本题共 6 个子任务：

- 子任务 1（5 分）：保证 $n \le 4$。
- 子任务 2（15 分）：保证 $n \le 100, \text{lim} \ge n^2$。
- 子任务 3（25 分）：保证 $n \le 300$。
- 子任务 4（25 分）：保证长度最小的排列的长度恰好为 n。
- 子任务 5（10 分）：保证 $(x_n, y_n) = (n, n)$。
- 子任务 6（20 分）：无额外限制。

对于每个测试点，若你成功构造了长度不超过 lim 的排列，可以获得 40% 的分数；若你成功构造了长度最小的排列，可以获得 100% 的分数。

对于每个子任务，你的得分为其所有测试点得分的最小值。

3 解题过程

以下用 LIS 表示最长上升子序列，用 LDS 表示最长下降子序列。

3.1 算法一

通过暴力枚举可以发现：当 $n \le 4$ 时，排列长度可以不超过 9。因此，只需要按照从小到大暴力枚举长度不超过 9 的排列即可。

时间复杂度 $\tilde{\mathcal{O}}((3n)!)$。可以通过子任务 1，期望得分 5。

3.2 算法二

构造长度为 n^2 的排列 $p = \{n, n-1, \ldots, 1, 2n, 2n-1, \ldots, n+1, \ldots, n^2, n^2-1, \ldots, n^2-n+1\}$。

可以发现，$p_{i \cdot n-j+1}$ 的**坐标**为 (i, j)，因此有 $f(p) = \{(i, j) \mid i, j \in [1, n]\}$，包含题目给定的所有二元组。

结合算法一，可以通过子任务 1，获得子任务 2 的部分分，期望得分 11。

3.3 算法三

考虑长度最小的排列的长度恰好为 n 的情况。

对于排列中的两个元素 p_i, p_j $(i < j)$，若 $p_i < p_j$，则 $a_i < a_j$；若 $p_i > p_j$，$b_i > b_j$。所以排列中所有元素的坐标**互不相同**。

注意到一个元素的坐标只与所有值不超过它的元素的相对位置有关，因此考虑按照值（即 p_i）从小到大的顺序加入元素。

观察加入元素的过程，可以发现当在排列 p $(|p| = n \ge 1)$ 中插入一个元素 $n+1$ 时，其坐标 (a', b') 满足以下条件：

1. **不被任何一个已有的坐标二维偏序，即不存在** $i \in [1, n]$**，使得** $a' \le a_i, b' \le b_i$。
   - 证明：对于一个 $i \in [1, n]$，若元素 $n+1$ 在 p_i 之后，则 $a' \ge a_i + 1$。（可以在以 p_i 结尾的 LIS 后拼接 $n+1$ 得到一个长度为 $a_i + 1$ 的上升子序列。）同理，若元素 $n+1$ 在 p_i 之前，则 $b' \ge b_i + 1$。
2. $(a'-1, b')$ **或** $(a', b'-1)$ **被某一个已有的坐标二维偏序，即存在** $i \in [1, n]$**，使得** $a' \le a_i + 1, b' \le b_i$ **或** $a' \le a_i, b' \le b_i + 1$。
   - 证明：设在以 $n+1$ 结尾的 LIS 中 $n+1$ 的前驱为 p_i（若不存在，设为 $p_{-\infty} = -\infty$），在以 $n+1$ 开头的 LDS 中 $n+1$ 的后继为 p_j（若不存在，设为 $p_{+\infty} = -\infty$）。如果 $p_i > p_j$，有 $a_i = a' - 1, b_i \ge b'$；否则 $p_i < p_j$，有 $a_j \ge a', b_j = b' - 1$。

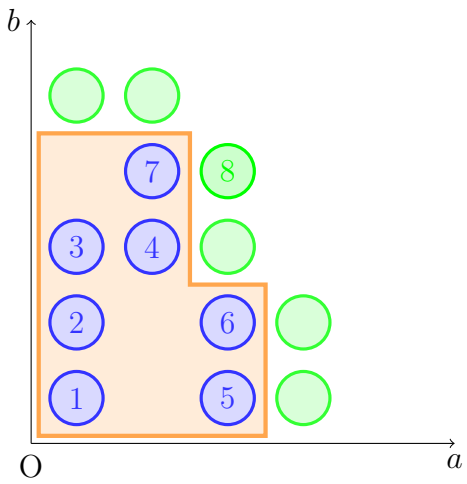
以上两个条件也是加入的坐标合法的充分条件。

充分性证明：

- 不失一般性地，设 $(a'-1, b')$ 被某一个已有的坐标 (a_i, b_i) 二维偏序，即 $a'-1 \le a_i, b' \le b_i$。
- 根据条件 1，$a' \le a_i$ 和 $b' \le b_i$ 不能同时成立。因此 $a' - 1 = a_i, b' \le b_i$。

- 设 $B_i = \max_{j=i}^{n} b_j, B_{n+1} = 0$。显然，$b_i \le B_{i+1} + 1$，所以 $B_{i+1} \le B_i \le B_{i+1} + 1$。
- 又因为 $b' \le b_i \le B_i$，一定存在 j，满足 $j > i, B_j = b' - 1$。取最小的 j 使得 $B_j = b' - 1$（一定有 $j > i$）。根据 j 的最小性，有 $b_{j-1} = b'$。
- 将 $n+1$ 插入 p_j **之前**（当 $j = n+1$ 时，插入到 p 的末尾），以 $n+1$ 开头的 LDS 长度为 $B_j + 1 = b'$。
- 将以 p_i 结尾的 LIS 拼接上 $n+1$ 可以得到一个长度为 a' 的上升子序列，以 $n+1$ 结尾的 LIS 长度至少为 a'。
- 如果以 $n+1$ 结尾的 LIS 长度超过了 a'，那么存在 $a_k = a'$ $(k < j)$。
  - 若 $p_k \le p_{j-1}$，则有 $a_{j-1} \ge a_k \ge a', b_{j-1} \ge b'$，违反了条件 1。
  - 若 $p_k > p_{j-1}$，则有 $a_k \ge a', b_k \ge b_{j-1} \ge b'$，违反了条件 1。
- 综上所述，以该方式插入 $n+1$，其坐标为 (a', b')。

因此，以上两个条件是加入的坐标合法的充要条件。
例如，在排列 $p = \{3, 7, 4, 2, 1, 6, 5\}$ 中插入元素 8，形成新排列 $p' = \{3, 7, 8, 4, 2, 1, 6, 5\}$：



图中，点的位置代表坐标，橙色区域为被二维偏序的区域，绿色点代表 p 中插入 8 可能的坐标。

称一个坐标集合 S **合法**当且仅当存在排列 p，使得 $f(p) = S$。
一个坐标集合 S **合法**的充要条件是：

- **不存在一个满足以下条件的格点路径** $(\mathrm{px}_1, \mathrm{py}_1), (\mathrm{px}_2, \mathrm{py}_2), \ldots, (\mathrm{px}_l, \mathrm{py}_l)$（以下称为**划分路径**）：
  - 格点路径只能向左，上，左上走，即 $\forall i \in [1, l-1], (\mathrm{px}_{i+1}, \mathrm{py}_{i+1}) - (\mathrm{px}_i, \mathrm{py}_i) \in \{(-1, 0), (0, 1), (-1, 1)\}$
  - 设 F 为被 S 中的某个坐标二维偏序的坐标的集合，即 $F = \{(x, y) | x, y \ge 1, \exists (x', y') \in S, x \le x', y \le y'\}$。格点路径只能包含 F 中的点，且起点和终点

可以再延伸一步离开 F，即 $\{(\mathrm{px}_i, \mathrm{py}_i) \mid i \in [1, l]\} \subseteq F$，
$(\mathrm{px}_1 + 1, \mathrm{py}_1) \notin F \vee (\mathrm{px}_1, \mathrm{py}_1 - 1) \notin F, (\mathrm{px}_l - 1, \mathrm{py}_l) \notin F \vee (\mathrm{px}_l, \mathrm{py}_l + 1) \notin F$。

- 格点路径上没有 S 中的坐标，即 $\{(\mathrm{px}_i, \mathrm{py}_i) \mid i \in [1, l]\} \cap S = \varnothing$。
- S 中有坐标在路径的右上方，即 $\exists i \in [1, l], (x, y) \in S, \mathrm{px}_i \le x, \mathrm{py}_i \le y$。

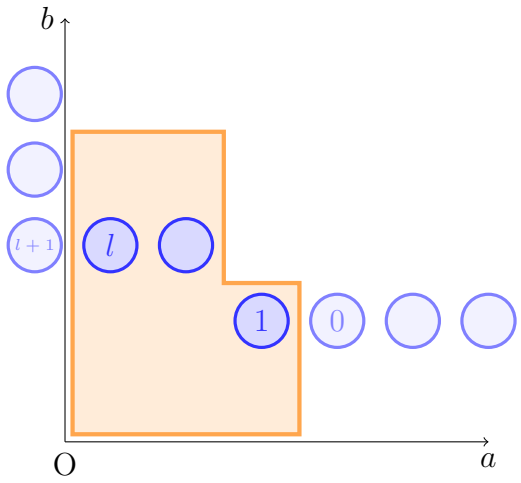
注意该条件蕴含了 $(1, 1) \in S$，因为当 $(1, 1) \notin S$ 时，$\{(1, 1)\}$ 是一条**划分路径**。
充分性：

- 当 S 满足条件时，**以下算法总能产生一个符合条件的排列**。
  - 首先令 $p = \{1\}$ $((1, 1) \in S)$。
  - 当 $1 \le |p| < |S|$ 时，设 T 为所有满足**加入坐标合法的条件 2**（$(a' - 1, b')$ 或 $(a', b' - 1)$ 被某一个已有的坐标二维偏序）的元素构成的集合。
  - 取出 T 中 $x + y$ 最小的元素，该元素同样满足条件 1，所以可以按照上文的方式加入排列。
- 当 $1 \le |p| < |S|$ 时，T 总是非空：假设 $1 \le |p| < |S|$，且 $T = \varnothing$，取 $F = \{(x, y) | x, y \ge 1, \exists (x', y') \in S, x \le x', y \le y'\}, F' = \{(x, y) | x, y \ge 1, \exists (x', y') \in f(p), x \le x', y \le y'\}, B = \{(x, y) \mid (x, y) \in F, (x, y) \notin F', (x - 1, y) \in F' \vee (x, y - 1) \in F'\}$，则 B 中的元素构成若干条路径，且每条路径都是一条**划分路径**，与不存在**划分路径**矛盾。
- 取出的元素同样满足条件 1：(x', y') $(x' \ge x, y' \ge y)$ 不可能在 (x, y) 之前加入，因为 (x, y) 会比 (x', y') 更早满足条件 2，并且 $x + y$ 更小。

必要性：

- 设存在一条**划分路径** $(\mathrm{px}_1, \mathrm{py}_1), (\mathrm{px}_2, \mathrm{py}_2), \ldots, (\mathrm{px}_l, \mathrm{py}_l)$，$F$ 为被 S 中的某个坐标二维偏序的坐标的集合。将**划分路径**的起点和终点各延伸一步离开 F，得到 $(\mathrm{px}_0, \mathrm{py}_0), (\mathrm{px}_1, \mathrm{py}_1), \ldots, (\mathrm{px}_l, \mathrm{py}_l), (\mathrm{px}_{l+1}, \mathrm{py}_{l+1})$。再将起点向右无限延伸，终点向上无限延伸，得到 $\ldots, (\mathrm{px}_0 + 1, \mathrm{py}_0), (\mathrm{px}_0, \mathrm{py}_0), \ldots, (\mathrm{px}_{l+1}, \mathrm{py}_{l+1}), (\mathrm{px}_{l+1}, \mathrm{py}_{l+1} + 1), \ldots$。被已加入坐标二维偏序的坐标的集合总是在该路径的左下方，且与该路径不交，故该路径右上方的坐标无法满足条件 2。

下图展示了如何延伸一条**划分路径**：



对于子任务 4，保证了长度最小的排列的长度恰好为 n。直接使用集合 S 合法条件充分性证明中的算法，即可构造一个排列 p。

时间复杂度 $\mathcal{O}(n^2)$，可以通过子任务 4，期望得分 25。

以下是将子任务 4 的时间复杂度降为 $\mathcal{O}(n\log n)$ 的方法（**在本题中不需要**）：

采用以下方式维护 T：

- 初始令 $T=\{(1,1)\}$。
- 对于 (x,y)，找到 y' 最小的 (x',y')，使得 $x'=x-1,y'\geq y$，若其存在，将 (x',y') 向 (x,y) 连边；找到 x' 最小的 (x',y')，使得 $x'\geq x,y'=y-1$，若其存在，将 (x',y') 向 (x,y) 连边。连边 $u\to v$ 的含义是，当 u 对应的元素被加入排列时，将 v 加入 T。

采用以下方式维护 p：

- 令 pa_i 为 p 中第一个满足以其结尾的 LIS 长度为 i 的元素，pb_i 为 p 中最后一个满足以其开头的 LDS 长度为 i 的元素。
- 插入元素时，可以根据 pa,pb 来 $\mathcal{O}(1)$ 定位，使用平衡树或离线后倒序用线段树插入元素。
- 插入元素后，只需对 pa,pb 单点改，区间 $+1$，可以使用线段树维护。

3.4 算法四

容易发现，$S'=S\cup\{(1,1),(2,1),\ldots,(n,1),(n,2),\ldots,(n,n)\}$ 总是**合法**的：

- 任意一条**划分路径**只能以 $\{(1,1),(2,1),\ldots,(n,1),(n,2),\ldots,(n,n)\}$ 为起点，而这些点都被 S' 包含了。

对 S' 构造对应的排列 p，有 $|p|=|S'|\leq|S|+|\{(1,1),(2,1),\ldots,(n,1),(n,2),\ldots,(n,n)\}|=3n-1\leq 3n$。

结合算法三，可以通过子任务 4，获得其他所有子任务的部分分，期望得分 55。

3.5 算法五

对于除了子任务 4 以外的子任务，需要最小化 $|p|$，等价于找到 S 最小的**合法**超集。

设 $F=\{(x,y)|x,y\geq 1,\exists(x',y')\in S,x\leq x',y\leq y'\}$。

把该问题转化成网络流的**最小割**问题，建图方式：

1. $s\xrightarrow{\infty}(x,y,0),\ (x,y)\in F,(x+1,y)\notin F\vee(x,y-1)\notin F.$
2. $(x,y,1)\xrightarrow{\infty}t,\ (x,y)\in F,(x-1,y)\notin F\vee(x,y+1)\notin F.$
3. $(x,y,0)\xrightarrow{1}(x,y,1),\ (x,y)\in F,(x,y)\notin S.$
4. $(x_1,y_1,1)\xrightarrow{\infty}(x_2,y_2,0),\ (x_1,y_1),(x_2,y_2)\in F,(x_2,y_2)-(x_1,y_1)\in\{(-1,0),(0,1),(-1,1)\}.$

其中，s 为源点，t 为汇点。

该图的结构与**划分路径**相对应，将该图的**最小割**（显然只包含 3 类边，即形如 $(x,y,0) \xrightarrow{1} (x,y,1)$ 的边）中的边对应的点加入 S，即可使 S **合法**。

根据**最大流最小割定理**，最大流的大小等于最小割的大小。残量网络上，起点从源点可达，终点不可达的边构成一个最小割。

该方式产生的排列长度也是最小的，因为每在 S 中加入一个元素，其最小割至多减小 1。

在 S 中加入一个元素可以看作先扩大 F，再割掉一条 3 类边。前者不会减小最小割，后者至多将最小割减小 1。

只需说明扩大 F 不会减小最大流，因为最大流的大小等于最小割的大小。

把扩大 F 拆分成若干次加入 (x,y) $(x=1 \vee (x-1,y) \in F, y=1 \vee (x,y-1) \in F)$，每一次都不会减小最大流。可以对原最大流做调整，使得流量不减小：

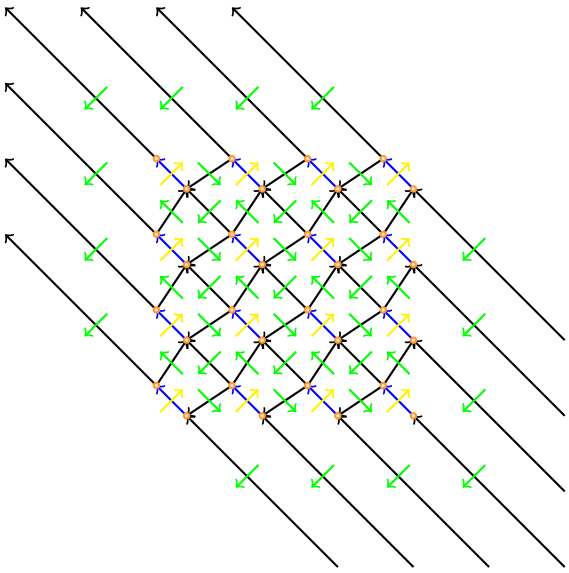
- 若原最大流中，$s \to (x-1,y,0),(x,y-1,1) \to t$ 均无流量，无需调整。
- 若原最大流中，仅 $s \to (x-1,y,0)$ 有流量，调整为 $s \to (x,y,0) \to (x,y,1) \to (x-1,y,0)$。
- 若原最大流中，仅 $(x,y-1,1) \to t$ 有流量，调整为 $(x,y-1,1) \to (x,y,0) \to (x,y,1) \to t$。
- 若原最大流中，$s \to (x-1,y,0),(x,y-1,1) \to t$ 均有流量，调整为 $s \to (x,y,0) \to (x,y,1) \to t, (x,y-1,1) \to (x-1,y,0)$。

使用 Dinic 算法求最大流，时间复杂度 $O(E^{3/2}) = O(n^3)$。其中 E 表示网络流中的边数，将流量为 ∞ 的边转化为 2 条流量为 1 的边即可套用单位网络上 Dinic 的时间复杂度分析。

结合算法三、四，可以通过子任务 1~4，获得子任务 5, 6 的部分分，期望得分 82。

3.6 算法六

当 $(n,n) \in S$ 时，可以对网络流的图进行对偶，求出对偶图的最短路：



图中，蓝色边表示原图中边权为 0 或 1 的边，黑色边表示原图中边权为 ∞ 的边，黄色边表示对偶图中边权为 0 或 1 的边，绿色边表示对偶图中边权为 0 的边。

将最短路中经过的边权为 1 的边对应的点加入 S，即可使 S **合法**。

求最短路时，可以直接在网格图上 01-BFS，也可以只对 S 中的点使用 Dijkstra 算法（(x_1, y_1) 到 (x_2, y_2) 的距离是 $\max\{\max\{x_2 - x_1, 0\} + \max\{y_2 - y_1, 0\} - 1, 0\}$）。

时间复杂度 $\mathcal{O}(n^2)$，结合算法三、四、五，可以通过子任务 1~5，获得子任务 6 的部分分，期望得分 88。

3.7 算法七

求最大流时，将**除了源点** s **以外的每个点**的可以到达的节点 (x, y, z) 按照 $x + y$ 排序，并优先遍历 $x + y$（称作**优先级**）较大的（特别地，汇点 t 优先级最高）。对于源点 s，将节点以 y 为第一关键字，以 x 为第二关键字从大到小排序，换句话说，就是按照顺时针的顺序扫描 F 右下的轮廓。这样进行一轮**不走反向边的** DFS 增广，就能求出最大流。

只需证明一次 DFS 后不存在增广路即可。假设一次 DFS 后仍存在一条增广路。

- 若该增广路不经过任何反向边，则 DFS 时一定会走这条增广路，所以这样的增广路不存在。
- 若该增广路经过了反向边，设其走的第一条反向边为 $v_1 \to v_2$，v_1 的前一个点为 v_0，在 v_2 之后走的第一条正向边的起点为 v_3，即该增广路为 $s \to \cdots \to v_0 \to v_1 \to v_2 \to \cdots \to v_3 \to \cdots \to t$，其中 $s \to v_1$ 都是正向边，$v_1 \to v_3$ 都是反向边。
  - 若 v_0 的**优先级**高于 v_2：由于 $v_1 \to v_3$ 都是反向边，v_1 到 t 有一条已经增广的路径，根据 DFS 及图的性质（一条路径不可能跨过另一条路径），DFS 时会先增广 $s \to \cdots \to v_1 \cdots \to t$，矛盾。
  - 若 v_2 的**优先级**高于 v_0：由于 $v_1 \to v_3$ 都是反向边，s 到 v_3 有一条已经增广的路径，根据 DFS 及图的性质，DFS 时会先增广 $s \to \cdots \to v_3 \cdots \to t$，矛盾。

因此，一次 DFS 后，残量网络上无增广路，也就是说，最大流已经求出。

实现时，无需建出网络流的图，只需在网格图上模拟 DFS 的过程。

时间复杂度 $\mathcal{O}(n^2)$，可以通过子任务 1~6，期望得分 100。

4 参考资料

[1] Wikipedia, Max-flow min-cut theorem

[2] Wikipedia, Dinic's algorithm