3 Two Bullets

3.1 题目来源

Petrozavodsk Winter 2022. Day 2. KAIST Contest + KOI TST 2021 Problem D[3].

3.2 题目大意

有一排建筑物从左往右排列，左边第 i 个建筑物的高度为 p_i。保证 $1 \le p_i \le n$ 且 p_i 两两不同，即 p_i 是一个 $1 \sim n$ 的排列。

你站在所有建筑物的左侧，你可以进行若干次操作，每次操作需要选择两个高度 $1 \le x, y \le n+1$（可以相同），两颗子弹将分别在 x 高度与 y 高度同时向右发射。

一颗高度为 x 的子弹会击中从左往右第一个高度 $\ge x$ 且认为被摧毁的建筑物，并将其摧毁。若不存在高度 $\ge x$ 且未被摧毁的建筑，则不会摧毁任何建筑。

特别的，若两颗子弹同时击中一个建筑，则只有这一个建筑会被摧毁。

你需要构造一种发射子弹的方案，经过最少次数的操作，让所有的建筑都被摧毁。

3.3 数据范围

对于所有数据，满足 $1 \le n \le 10^5$，$p_1, \ldots, p_n$ 构成一个 $1 \sim n$ 的排列。

3.4 解题过程

参考资料 1 的论文中给出了如下问题的解决方案与其最优性证明：

问题：给定一张有向无环图，每次可以选择至多两个入度为 0 的点删除，求最少删除多少次可以将所有点删空。

解决方案：

1. 求出原图的一种特殊拓扑序。方式与普通拓扑排序类似，维护当前入度为 0 的点集。令 $N(v)$ 表示 v 的所有入边被删除的时间集合。每次寻找将所有 $N(v)$ 中的元素降序排序后字典序最小的 v，选择该点 v 作为这一轮拓扑排序的点，并对应删除 v 的所有出边。

2. 求出拓扑序后，按照反着的顺序构造方案。每次维护所有出度为 0 的点中拓扑序最靠后的两个点（若只有一个出度为 0 的点则只选一个）删去，并将这两个点加入所构造方案的开头。

[3] https://qoj.ac/contest/820/problem/2555

原论文中证明了该解决方案所构造出的删点顺序即为操作次数最少的方案。

在该题中，对于所有 $i < j$ 且 $p_i > p_j$ 的点对 (i, j)，连接一条 $i \to j$ 的有向边，原问题即为在这张图上做上述问题。称该有向图为原排列的“排列图”。直接模拟该过程可以做到 $O(n^3)$。

令 $layer(i)$ 表示在排列图中以 i 结尾的最长链长度，即排列中以 i 结尾的最长下降子序列长度。有结论：对于任意两点 u, v 满足 $layer(u) < layer(v)$，必然有在拓扑排序中 u 在 v 前面。

证明：考虑归纳。$layer(u) = 1$ 的节点即为初始就没有入度的节点，其 $N(u) = \emptyset$，因此 $N(u)$ 一定是最小的，也就是说在所有 $layer(u) = 1$ 的节点被拓扑完之前不可能有 $layer(u) = 2$ 的节点被拓扑到。考虑任意一个 $layer(u) = x$ 的节点 u 必然有一个 $layar(v) = x - 1$ 的入边 $v \to u$，因此 $N(u)$ 中的最大值一定是某一个 $layer(v) = x - 1$ 的节点 v 的拓扑序位置。则对于任意两个 u, v 满足 $layer(u) < layer(v)$，必然有 $N(u)$ 中的最大值 $< N(v)$ 中的最大值，即 u 在拓扑排序中必然在 v 之前。

因此，我们可以对于 $layer(u) = 1, \ldots, n$ 的点依次做拓扑排序。并且注意到对于任意两个 $layer(u) = layer(v)$ 的点，u 和 v 之间不可能有连边，否则若有边 $u \to v$，则必有 $layer(v) > layer(u)$。这启示我们只需要对于一层 $layer(u)$ 相同的节点做按照 $N(u)$ 的排序即可。

排序过程中需要做 $O(n \log n)$ 次同层之间两个节点 u, v 的 $N(u), N(v)$ 降序排序后的字典序比较。容易发现这等价于比较 $N(u) \setminus (N(u) \cap N(v))$ 中的最大值和 $N(v) \setminus (N(u) \cap N(v))$ 中的最大值。

将所有的 (i, p_i) 看作一个二维平面上的点，$N(u)$ 即为一个左上角的区域，$N(u) \setminus (N(u) \cap N(v))$ 则为一个 3-side 的矩形区域。查询对应区域的最大值可以使用树套树维护，单次比较复杂度 $O(log^2 n)$，总复杂度为 $O(n \log^3 n)$。

注意到之前证明了对于任意两个 $layer(u) < layer(v)$ 的节点 u, v，必然有 u 的拓扑序在 v 之前，因此最大值一定在该 3-side 矩形区域内的 $layer(u)$ 最大的点之中，而 $layer(u)$ 在我们所有的排序之前已经被确定。静态查询 3-side 矩形最值可以使用持久化线段树做到 $O(n \log n)$ 预处理，$O(\log n)$ 查询。

由于一层的节点之间互相没有连边，因此一层节点可以被表示为 $u_1 < \ldots < u_k$，且 $p_{u_1} < \ldots < p_{u_k}$。因此在确定最大值 u 所在 $layer(u)$ 之后，只需要在这一层的点中做 3-side 查询，而这一层中点的横纵坐标分别递增，因此其实是一个区间最值查询，容易使用线段树或 ST 表做到 $O(n) - O(\log n)$ 或 $O(n \log n) - O(1)$。

至此，单次比较复杂度被优化至 $O(\log n)$，可以在 $O(n \log^2 n)$ 的复杂度内求出一组“特殊的”拓扑排序。

对于原算法的第二步，出度为 0 的节点等价于后缀最小值位置，可以直接用一个 set 动态维护当前所有的后缀最小值。每次删除一个数时，使用线段树二分操作不断找出新的后缀最小值位置即可，这一部分容易做到 $O(n \log n)$。

结合两个步骤，即可在 $O(n \log^2 n)$ 的时间复杂度，$O(n \log n)$ 的空间复杂度内解决问题。

3.5 参考资料

1. Coffman, Edward G., and Ronald L. Graham. ”Optimal scheduling for two-processor systems.” Acta informatica 1.3 (1972): 200-213.