2020 Multi-University Training Contest 4

Problem Analysis

杭州学军中学

July 30, 2020

A. Anti-AK Problem

Author: Zhou Renfei, Xu Zhean　　*Shortest Judge Solution: 5659 bytes*

首先，我们来考虑随机加入一个圆后，最后的答案如何计算。设球面总面积为 A，球面被覆盖的面积为 B，新增的圆的面积为 C。对于球面上的每一个点，计算它被覆盖的概率：已被原有的圆覆盖的点概率为 1，其余的点概率为 C/A。所以我们可以简单地写出答案，为 $B+\frac{(A-B)\cdot C}{A}$。问题转化为球面上圆的面积并问题。

原本这题是平面上的版本，如你所见，这就变成了圆的面积并模板题。后来，出题人翻出一本《高中数学 选修 *3-3*：球面上的几何》，这题就变为了现在球面上的版本……

有关球面几何的基础内容，《高中数学 选修 *3-3*：球面上的几何》有着较为全面的介绍。在这里，我们先简要介绍球面上的几何元素。圆是一个平面与球面相交的部分，它同时也是空间中的一个圆，分为大圆与小圆。大圆是最大的圆，即过球心的平面与球面相交的部分。除大圆外的所有圆都是小圆。类比平面的情况，球面上的“直线”就是大圆。对任意两个不“对立”的点 a,b，经过这两个点的（球面）直线较短的一段圆弧就是 a,b 之间的（球面）线段。

接着是球面上的一些公式，实际上，小圆以及球面三角形的面积都能简单计算。球冠的表面积公式为 $2\pi H$，其中 H 为球冠高度。对于球面上的三个点 A,B,C，其球面三角形的面积为 $R^2\cdot(\angle A+\angle B+\angle C-\pi)$，其中 $\angle A$ 是 A 处球面上的角，也是平面 OAB 与 OAC 的二面角，很容易将其推广到球面多边形的情况。

在介绍正式的解决方案之前，我们接着来了解一些球面几何实现上的细节。首先是圆上点的表示，用三维空间中的 (x,y,z) 能方便计算。某些情况下，我们需要将三维空间中的几何元素放在小圆对应的平面上进行操作，做法是任取平面上的一对基，将三维中的点坐标与基求点积，就能得到平面上的坐标。最后还有一个细节就是两圆求交，这可以转化为平面求交，相交得到的直线再与圆求交。利用之前的方法，把直线放在小圆的平面上处理即可。

模仿平面上的解决方案，计算球面上圆的面积并有多种方法。考虑将面积并拆成弓形和多边形的面积，将每个圆为中心，其它圆极角排序，删去没有被覆盖的圆弧对应的弓形，剩下了一些多边形。弓形部分容易计算，而多边形的内部可能存在一些“洞”，如何使得面积被正确地计算？其实解决方案也很简单，只需要计算边缘左侧的面积即可。如果球面多边形内存在 k 个洞，那么多边形的部分被计算了 $k+1$ 次面积，其余部分被计算了 k 次。由于球面的总面积为 $4\pi R^2$，只需要将最后的答案对 $4\pi R^2$ 取模即可。但是这种做法的数值稳定性较差。因为要计算多边形的面积，所以需要将点的坐标求出，并合并相同的坐标，抠出多边形的边缘。如果存在多个圆相切或相交在同一点，存在误差难以判断点是否需要合并，从而造成更大的误差。

更好做法是借鉴平面上的格林公式方法，即用上边缘减去下边缘。我们先随机旋转球面，避免圆经过南极或北极的极端情况。弓形部分用同样的方法计算，对于多边形的部分，我们将边缘 (A,B) 的两端向南极 S 连边，转化为计算球面三角形 SAB 的面积（注意这里的面积根据方向仍然有正负之分），而不需要求出多边形每个顶点的角度。处理面积并问题的格林公式方法的一般化及其证明将会在周任飞同学的集训队论文中作详细地论述。

实现上述算法后，精度误差较大的原因是三个点接近共线时，原有的球面三角形的面积计算公式有极大的误差。为了追求更高的精度，我们可以使用另一个公式来计算面积。假设球的半径为 1，我们已

经知道了两条边的长度 a, b 与它们之间的夹角 C，那么有：

$$\tan\frac{S_{\triangle ABC}}{2} = \frac{\sin\frac{1}{2}a\sin\frac{1}{2}b\sin C}{\cos\frac{1}{2}a\cos\frac{1}{2}b + \sin\frac{1}{2}a\sin\frac{1}{2}b\cos C}$$

这个公式的证明较为复杂，可以参考 *Spherical Trigonometry* 一书第 75 页之证明。

在不使用这个公式计算球面三角形之前，良好实现的代码的平均精度能够达到 10^{-9} 至 10^{-10} 左右，由于数据组数较少，有可能通过本题。在使用这个公式之后，平均精度能够达到 10^{-13} 左右，经过一万组数据的测试，仍能保证 10^{-10} 的精度。

时间复杂度为 $\mathcal{O}(n^2\log n)$。

B. Blow up the Enemy

Author: Zhou Renfei　　*Shortest Judge Solution: 430 bytes*

观察题目性质，由于两人的初始血量都为 100，所以对于每种武器 i，都能算出击杀对方的时间。具体来说，$\left\lceil\frac{100}{a_i}\right\rceil$ 就是需要的攻击次数，那么 $\left(\left\lceil\frac{100}{a_i}\right\rceil - 1\right)\cdot d_i$ 就是最早击杀对方的时间。

显然，你的最优决策是选择击杀时间最早的那一种武器，再枚举对方的选择计算胜利的概率即可。时间复杂度 $\mathcal{O}(n)$。若 $\mathcal{O}(n^2)$ 枚举每一对武器，也能通过本题。

C. Contest of Rope Pulling

Author: Xu Zhean　　*Shortest Judge Solution: 773 bytes*

先转化题意：有 $n+m$ 个物品，每个物品用一对数 (w_i, v_i) $(-10^3 \le w_i \le 10^3, -10^9 \le v_i \le 10^9)$ 描述。选择一个物品子集 S，满足 $\sum_{i\in S} w_i = 0$，最大化 $\sum_{i\in S} v_i$。这是一个经典的 01 背包问题，通过 DP 可以做到 $\mathcal{O}((n+m)^2\cdot\max\{w_i\})$ 的时间复杂度。

单个物品的 w_i 最大只有 10^3，但是 DP 过程中 $\sum w_i$ 的最大值可能达到 10^6。这无疑是极大的浪费。

如果我们以一种相对随机的方式安排物品的加入顺序，那么可以发现，最优解在每一阶段对应的状态都在 0 附近振荡。设最优解的子集为 S，在物品全集的加入顺序被等概率重排之后，子集 S 的加入顺序也被等概率重排。我们只需要设一个足够大的上界 T，只考虑 $[-T, T]$ 的 $\sum w_i$，做原来的 DP 即可。

接下来描述了在等概率重排序列之后，数列前缀和最大值足够小的原因。

新的问题

设有 $2n$ 个整数 $a_1, a_2, \cdots, a_n$，满足 $\sum_{i=1}^{2n} a_i = 0, |a_i| \le V$。将序列 $\{a_i\}$ 等概率随机排列得到序列 $\{b_i\}$。

计算 $A=\max\{\sum_{i=1}^{k} b_i | k\in\{1,2,\cdots,2n\}\}$ 期望的上界。

Lemma 1. *若存在 i,j 满足 $-V<a_i\le a_j<V$，令 a_i 减 1 且 a_j 加 1 后，目标期望增加。*

证明. 不妨设 i 经过重排后位置为 p，j 重排后位置为 q，有 $b_p\le b_q$。

设满足 $p<q$ 的排列集合为 S ，满足 $p>q$ 的排列集合为 T。对换 (p,q) 建立了 S 与 T 之间的一一映射。

考察最大前缀和的下标 k。对于 $p<q$ 的情况，若 $k<p$ 或者 $k\ge q$，则对目标期望没有影响。

显然 $p<q$ 时 $k\in[p,q)$ 的概率小于 $q>p$ 时 $k\in[q,p)$ 的概率，也就是增加的贡献大于减少的贡献，目标期望值增加。

□

经过不断操作之后，a_i 一定恰有 n 个 V 与 n 个 $-V$。可以只考虑 $a_i=\pm1$ 的情况。

考虑如何计算目标值为 m 的方案数。使用计算卡特兰数的对称法，可以得到目标值 $\le m$ 时的方案数为 $\binom{2n}{n}-\binom{2n}{n-m-1}$，目标值等于 m 的方案数为 $\binom{2n}{n-m}-\binom{2n}{n-m-1}$。

而总贡献为

$$\sum_{m=0}^{n} m\cdot\left(\binom{2n}{n-m}-\binom{2n}{n-m-1}\right)=\sum_{m=0}^{n-1}\binom{2n}{m}\approx 2^{2n-1}$$

使用斯特林公式来近似阶乘，总方案数为

$$\binom{2n}{n}=\frac{(2n)!}{n!n!}\approx\frac{\sqrt{4\pi n}\cdot 2^{2n}\cdot(n/e)^{2n}}{2\pi n\cdot(n/e)^{2n}}=\frac{2^{2n}}{\sqrt{\pi n}}$$

得到期望为 $\mathbb{E}(A)\approx\frac{\sqrt{\pi}}{2}\cdot\sqrt{n}$。

继续只考虑 $a_i=\pm1$ 的情况（需要注意此时 **Lemma 1** 不再成立），考虑计算 $A\ge m$ 的概率 $P(m)$。

$$\begin{aligned}
P(m)&=\frac{\binom{2n}{n-m}}{\binom{2n}{n}}=\frac{n!n!}{(n-m)!(n+m)!}=\frac{n(n-1)\cdots(n-m+1)}{(n+1)(n+2)\cdots(n+m)}\\
&\le\frac{n(n-1)\cdots(n-m+1)}{(n+1)^m}=\prod_{k=1}^{m}\left(1-\frac{k}{n+1}\right)\\
&\le\prod_{k=1}^{m}\left(1-\frac{1}{n+1}\right)^k=\left(1-\frac{1}{n+1}\right)^{(n+1)\cdot\frac{m(m+1)/2}{n+1}}\\
&\approx e^{-\frac{m(m+1)}{2(n+1)}}
\end{aligned}$$

若 $m=4\sqrt{2n}$，则超过上限的概率至多是 $e^{-16}\approx 10^{-7}$。

上界 T 取 $\mathcal{O}(\sqrt{n+m})$ 即可，时间复杂度 $\mathcal{O}((n+m)^{1.5}\cdot\max\{w_i\})$。

D. Deliver the Cake

Author: Zhou Renfei   Shortest Judge Solution: 1999 bytes

如果没有左右手的限制，本题就是一个经典的最短路问题。为了解决存在左右手限制的问题，我们可以将图中的每一个节点拆分为两个意义不同的节点。具体来说，对于任意节点 i，构造节点 i_1 与 i_2，其中 i_1 强制使用左手，而 i_2 强制使用右手。如果节点 i 的类型为 'L'，那么就不存在 i_2；如果节点 i 的类型为 'R'，那么就不存在 i_1。

接下来考虑连边，显然我们不会再一条道路上换两次以上的手。对于任意一条无向边 (a, b, d)，我们需要考虑节点 a_1, a_2 与 b_1, b_2 之间的连边（如果节点不存在就不连边），如果 a_p 与 b_q 连边且 $p \neq q$，就需要支付额外的 x 代价。由此，我们得到了一张新的图，s_1, s_2 到 t_1, t_2 的最短路长度即是答案。使用 Dijkstra 算法求出最短路，时间复杂度 $\mathcal{O}((n+m)\log n)$。

E. Equal Sentences

Author: Xu Zhean   Shortest Judge Solution: 579 bytes

尝试分析题目所述的条件，找到与之等价的条件。假设句子 S 与 T 几乎相等，取最大的 k 满足 $S[1:k] = T[1:k]$。如果 $k \neq n$，必然有 $S[k+1] \neq T[k+1]$。由于需要满足题目所述的条件，所以一定会有 $S[k+1] = T[k+2], S[k+2] = T[k+1]$。我们可以将 $T[k+1]$ 与 $T[k+2]$ 交换，继续执行操作。

由此可以发现，对于任意一个与 S 几乎相等的句子 T，都能唯一对应若干个不交的交换操作。同样地，任意一组不交的交换操作，也能唯一对应一个与 S 几乎相等的句子 T。需要注意，这里不交的交换操作 $(i, i+1)$ 应满足 $S[i] \neq S[i+1]$。

接下来可以用 DP 来解决这个问题。令 $f(i)$ 表示与 $S[1:i]$ 几乎相等的句子个数。转移到 i 的时候，考虑 i 与 $i-1$ 是否进行交换操作即可（交换需满足 $S[i-1] \neq S[i]$）。时间复杂度 $\mathcal{O}(n)$。

F. Fake Photo

Author: Zhou Renfei   Shortest Judge Solution: 1857 bytes

题目要求最大值最小，容易想到使用二分答案。判断是否存在一个时刻，时针和分针的角度之差都不超过 α。这一部分的实现方法很多，接下来介绍其中的一种。

对于每个手表，求出时针与分针的允许角度范围，并分别求交。如果时针或者分针范围的交集为空，那么答案 α 一定不可行。接下来考虑时针的范围 $[l, r]$，如果 $r - l \geq \frac{\pi}{6}$，那么任意的分针都可以构造，答案 α 一定可行。否则，根据时针的范围，我们可以求出其对应的分针范围，如果与之前求出的分针范围有交，答案 α 也是可行的。

视实现而定，时间复杂度为 $\mathcal{O}(n \log n \log^{-1} \epsilon)$ 或 $\mathcal{O}(12 \cdot n \cdot (\log n + \log^{-1} \epsilon))$。

G. Go Running

Author: Jiang Xunchi　　*Shortest Judge Solution: 2367 bytes*

建立一个二维坐标轴，若在第 a 秒，位置 b 上有至少一个人，就将坐标 (a,b) 染黑。

若第 i 个人出发时间为 l_i，结束时间为 r_i，出发坐标为 s_i，速度为 v_i，那么这个人的影响为：对于所有的 $0 \le t \le r_i - l_i + 1$，将 $(t, s_i + tv_i)$ 染黑。可以发现，被染黑的所有点都位于一条斜率为 v_i 的线段上。

则题意可以转化为：给定平面上 n 个点，选择尽可能少的斜率为 1 或 -1 的线段，使得这 n 个点被覆盖至少一次。

可以发现，将题意中的选择线段修改为选择直线，答案不会变化。因为选择的线段应该尽可能长。

接着将二维平面旋转 $45°$，这样选择的直线就变为平行于 x 轴或 y 轴的直线。

建立一张二分图，对于旋转后的一个点 (x_i, y_i)，连接一条从左侧的点 x_i 到右侧的点 y_i 的边。

建图之后，目标就转换为选择二分图的若干顶点。若选择了一个左侧的点 a，视作选择了直线 $x = a$；若选择了一个右侧的点 b，视作选择了直线 $y = b$。可以发现，对于每条边，都需要满足它的端点中存在至少一个被选择的点。

至此，本题已经转换为二分图的最小点覆盖问题。这是一个经典问题，有一个结论为：二分图的最小点覆盖等于二分图的最大匹配。之后求出这张二分图的最大匹配即可。

由于这张二分图的点数和边数均为 $\mathrm{O}(n)$ 级别，采用一些较快的匹配算法可以做到以 $\mathrm{O}(n\sqrt{n})$ 的复杂度解决此题。

H. Head Maker

Author: Zhang Zheyu　　*Shortest Judge Solution: 2702 bytes*

方法一

搜索。

对于一张网格，将每个单位格子视为一个点，每两个有公共边的格子连一条边，将这个网格转化成一张图 G。切开一条边可以视为在 G 上去掉一条边。那么，当 G 变成一棵树的时候，折叠的方案是唯一确定的。因此，枚举图 G 的生成树，即可枚举出所有的折叠方案。

猜想对于特别稠密的图（即边数减点数特别大的图），随机生成树能折成立方体的概率很大；而对于特别稀疏的图，可以很快地枚举所有的生成树。那么，以较为随机的方式枚举生成树，就能很快地解决这个问题。

另外，通过枚举所有的不可行的网格，可以发现，生成树之外的额外边数大于 4 的网格一定有解。如果特判这一类网格，那么搜索的时间将会得到一个可控的上界。

方法二

打表。

如果将对称、旋转后相同的连通块视为相同的话，那么在 6×6 的范围内，只有五千多种不同的连通块不可折成立方体。通过任意压缩方式将这五千多个连通块嵌入代码即可。

I. Imperative Meeting

Author: Xu Zhean, Zhou Renfei *Shortest Judge Solution: 1929 bytes*

考虑计算每条边的贡献：设两边分别有 i 和 $m-i$ 位同学，那么这条边至少需要贡献 $\min(i, m-i)$ 次。事实上，这个下界也是能够达到的，取这棵树的带权重心即可。设边 e 的两端的子树大小分别为 s 和 $m-s$，这条边的贡献为

$$f(s) = \sum_{i=1}^{m-1} \binom{s}{i}\binom{n-s}{m-i} \cdot \min(i, m-i)$$

其中 min 运算难以处理，将其拆成两部分：

$$f(s) = \sum_{i=1}^{\lfloor \frac{m-1}{2} \rfloor} \left(\binom{s}{i}\binom{n-s}{m-i} \cdot i + \binom{s}{m-i}\binom{n-s}{i} \cdot i \right) + [m \bmod 2 = 0] \cdot \binom{s}{m/2}\binom{n-s}{m/2} \cdot (m/2)$$

考虑对称性，我们这里只计算括号内的第一项之和 $g(s)$。记 $p = \lfloor \frac{m-1}{2} \rfloor$，下式 $g(s)$ 中的乘 i 可以通过吸收律消去。

$$g(s) = \sum_{i=1}^{p} \binom{s}{i}\binom{n-s}{m-i} \cdot i = s \cdot \sum_{i=1}^{p} \binom{s-1}{i-1}\binom{n-s}{m-i} = s \cdot h(s)$$

我们可以为 $h(s)$ 找到一个组合解释：共有 $n-1$ 个盒子，需要放 $m-1$ 个无法辨别的球，每个球放到一个盒子中，每个盒子至多放一个求，并且前 $s-1$ 个盒子里至多放 $p-1$ 个球，问有多少种放置方案。

首先必须要满足的是 $p \geq 1$，其次，当 $s = 1$ 时，所有 $\binom{n-1}{m-1}$ 种放置方案都是合法的。考虑 $s-1$ 变为 s 后，新增的不合法方案数。此时位置 $s-1$ 必然放置了一个球，前 $s-2$ 个位置恰好有 $p-1$ 个球，剩下的球就放在剩下的 $n-s$ 个盒子里。于是，我们能够 $\mathcal{O}(1)$ 地维护每个 $h(s)$，也就能分别 $\mathcal{O}(1)$ 地求出 $f(s)$。

总时间复杂度为 $\mathcal{O}(n)$。

J. Joyful Party

Author: Zhang Zheyu *Shortest Judge Solution: 3896 bytes*

建图

建图，每个谜题建立一个点。对于每一组 (x, l, r, c)，将 x 向所有 $[l, r]$ 中的点连一条边权为 c 的有向边。特别地，每个点都向 $[1, n]$ 中所有点连一条边权为 0 的有向边。

那么，答案即为弱连通块数量为 m 的最大内向森林。

我们给出两个非常显然的结论：

1. 对于任意一个弱连通块数量为 m 的内向森林，都能找到对应的 m 个序列的划分方案。
2. 对于任意一个 m 个序列的划分方案，都能找到对应的弱连通块数量为 m 的内向森林。

由此可见这样建图是正确的。

将边权取反后，我们就将原问题转化成了最小内向森林问题。

最小内向森林问题

方法一

采用凸优化转化成最小树形图问题。

在求最小树形图的时候，虽然边数非常地多，但如果不遍历所有的自环（包括缩点后形成的自环），只需要遍历 $\mathcal{O}(n)$ 条边就一定能结束算法。

具体来说，对于每一个缩成的点，用数据结构维护点集。这样就能很快地求出一个区间内最小的不在点集中的点。

时间复杂度：$\mathcal{O}\left(k \log k \log(nw)\right)$，其中 w 为值域。

方法二

套用最小内向森林算法。

假设现在已经得到了边数为 $i-1$ 的最小内向森林 H_{i-1}，想要求得边数为 i 的最小内向森林 H_i。

观察到 H_{i-1} 由很多个不连通的内向树形成，而对于每一棵内向树，都有一个对外扩张的代价。找到扩张代价最小的内向树，使其扩张即可得到 H_i。

为了方便求得一棵内向树的扩张代价，需要预先处理每棵内向树的根指向自己的缩环操作。这样的话，一棵内向树的根的最小出边加上之前的缩环代价，就是这棵内向树的扩张代价。

至于边数过大的问题，解决方法同上。

时间复杂度：$\mathcal{O}(k \log k)$。

K. Kindergarten Physics

Author: Zhou Renfei　　*Shortest Judge Solution: 223 bytes*

显然，两个质点移动的距离与质量、时间正相关，与距离负相关。自己估算或者观察样例，发现在数据范围可能的最坏情况下 $a = b = t_0 = 100, d = 1$，质点移动的距离远小于 10^{-6}（所容许的误差值），所以对于任意数据范围内的输入，都直接输出 d 即可。

L. Last Problem

Author: Zheng Luming　　*Shortest Judge Solution: 218 bytes*

这道题有很多做法，都能通过本题。

方法一

一种自然的想法是考虑画出颜色 n 之前需要进行什么准备。

如果 n 足够大，要想画颜色 n，一种方法是先画出以下形状 $G_{n,1}$：

	$n-1$	
$n-2$	n	$n-3$
	$n-4$	

要想画出 $G_{n,1}$，需要画出 $G_{n,2}$：

		$n-2$		
	$n-3$	$n-1$	$n-4$	
$n-4$	$n-2$	$n-5$	$n-3$	$n-6$
	$n-6$	$n-4$	$n-7$	
		$n-8$		

我们发现，这样操作可以共用数字，可以避免操作次数指数级增长。

不妨定义 $G_{n,l}$ 为以下操作序列：在 $(x, l+y)(x, y \in \mathbb{Z}, |x|+|y| \le l, (|x|+|y|) \bmod 2 = 0)$ 处绘制颜色 $n - l + \frac{3y-x}{2}$（如果不是正数就忽略）。容易看出，$G_{n,n}$ 是空操作序列，而 $G_{n,0}$ 就是在 $(0,0)$ 处绘制 n。

因为 $G_{n,l}$ 中的位置互不相同，奇偶性一致，所以 $G_{n,l}$ 中的操作顺序互不影响。

如果 $G_{n,l+1}$ 已经完成，那么对于 $G_{n,l}$ 中的每个操作 (x, y, c)，可以验证操作 $(x, y+1, c-1)$，$(x-1, y, c-2)$，$(x+1, y, c-3)$ 和 $(x, y-1, c-4)$ 分别要么颜色不是正数（因此不需要），要么存在于 $G_{n,l+1}$ 中。所以，在完成 $G_{n,l+1}$ 的所有操作后可以立即完成 $G_{n,l}$ 中的每个操作。

因此，依次进行 $G_{n,n-1}, G_{n,n-2}, \ldots, G_{n,1}, G_{n,0}$ 即可满足题目要求。$G_{n,l}$ 的操作次数是 $\mathcal{O}(l^2)$ 的，总操作次数是 $\mathcal{O}(n^3)$ 的。在 $n = 100$ 时，标算进行了 36839 步操作。

虽然这个做法是 $\mathcal{O}(n^3)$ 步的，但是常数很小（约为 $\frac{5}{144} \approx 0.03472$）。标算在 $n = 100$ 时用了 36839 步。几乎所有正确的算法都能在 10^5 步内通过本题。

方法二

按照 $f(x, y) : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}_5 = (3x + y) \bmod 5$ 将所有整点分为 5 类。

对于任意整点 (x, y)：

- $f(x, y+1) \equiv f(x, y) + 1 \equiv f(x, y) - 4$
- $f(x, y-1) \equiv f(x, y) - 1$
- $f(x+1, y) \equiv f(x, y) + 3 \equiv f(x, y) - 2$
- $f(x-1, y) \equiv f(x, y) - 3$

因此任意一类中的任意一个点都与另四类点直接相邻。

我们可以考虑按照 $f(x, y)$ 的值决定要对整点 (x, y) 染那些颜色。具体地，我们可以对于 1 到 n 中的每个颜色 c，将第 $c \bmod 5$ 类的所有整点染成颜色 c。这样，当绘制颜色 c 时，由于另四类点刚刚被染成颜色 $c-1$，$c-2$，$c-3$ 和 $c-4$（如果是正数），染色一定能成功。

虽然我们并不能进行无限步操作，但是我们只需要把一个点染成颜色 n，因此可以从它开始倒推依赖，同样只需要 $\mathcal{O}(n^3)$ 步就能完成，可以通过本题。