中国计算机学会
China Computer Federation

OpenCup 趣题选讲

王修涵

清华大学 交叉信息研究院

Email: wxh010910@gmail.com

2020 年 8 月 2 日

OpenCup 简介

OpenCup 是一个俄罗斯的 ICPC 训练赛，平均一两周一次比赛。

题目是各地的 camp 或者校赛题目，平均质量较高。

除了讨论两道 IOI 的试题外，我还选取了一些比较有趣的 OpenCup 的题目和大家分享。

Shoes

Statement[a]

[a]Source: IOI 2019 Day 1, Problem A

给定一个长度为 $2n$ 的非零整数序列 x ，你可以不断交换相邻位置的值使得对于任意 $0 \le i < n$ 都满足：

- $|x_{2i}| = |x_{2i+1}|$
- $x_{2i} < 0, x_{2i+1} > 0$

求最少交换次数。保证一定有解。

$n \le 10^5$ 。

Shoes

观察：对于绝对值相同的数，一定是第 i 个负数和第 i 个正数互相匹配。

这样我们就只需要解决这样一个问题：有 n 对数 (a_i, b_i) ，你需要给它们确定一个相对顺序，使得它们按顺序拼接起来之后逆序对个数最少。

Shoes

观察：对于绝对值相同的数，一定是第 i 个负数和第 i 个正数互相匹配。

这样我们就只需要解决这样一个问题：有 n 对数 (a_i, b_i) ，你需要给它们确定一个相对顺序，使得它们按顺序拼接起来之后逆序对个数最少。

由于 a_i 和 b_i 一定是相邻的，如果 $a_i > b_i$ ，我们可以先用一次次数将它们交换，不会对答案造成任何影响。

结论：按照 a_i 从小到大的顺序是最优的。可以使用调整法来证明。

最后用树状数组求逆序对统计答案即可。时间复杂度 $O(n \log n)$ 。

Split

Statement[a]

[a]Source: IOI 2019 Day 1, Problem B

给定一张 n 个点，m 条边的无向连通简单图和三个参数 a, b, c ，你需要将这个图分成三个大小分别为 a, b, c 的集合，使得至少两个集合是连通的。判断是否有解，如果有解需要给出一组方案。

$n \leq 10^5, m \leq 2 \times 10^5, a + b + c = n$ 。

部分分：$m = n - 1$ 。

Split

中国计算机学会 China Computer Federation

不妨设 $a \le b \le c$ ，只需要让大小为 a 和 b 的集合连通即可。

树的情况：注意到两个集合之间一定存在一条边将它们分开，不妨枚举这条边，那么这条边一侧的点数至少要是 a ，另一侧至少要是 b 。以 0 号点作为根，即询问是否存在一个子树，其子树大小在区间 $[a, n-b]$ 或 $[b, n-a]$ 内。一遍 DFS 求出子树大小即可。时间复杂度 $O(n+m)$ 。

Split

中国计算机学会 China Computer Federation

不妨设 $a \le b \le c$ ，只需要让大小为 a 和 b 的集合连通即可。

树的情况：注意到两个集合之间一定存在一条边将它们分开，不妨枚举这条边，那么这条边一侧的点数至少要是 a ，另一侧至少要是 b 。以 0 号点作为根，即询问是否存在一个子树，其子树大小在区间 $[a, n-b]$ 或 $[b, n-a]$ 内。一遍 DFS 求出子树大小即可。时间复杂度 $O(n+m)$ 。

考虑图上的情况。由于连通性只需要树就可以保持，所以如果有解，那么存在一棵生成树也有解。

假设子树大小区间为 $[l, r]$ ，由 $a+b+c \le n, a \le b \le c$ 可以立即推导出 $2l \le r, 2r > n$ 。

Split

中国计算机学会 China Computer Federation

任意找一棵 DFS 树，如果其存在一个子树大小在区间 $[l, r]$ 内，那么我们找到了一组解；否则由于 $2r > n$，一定存在一个唯一的点 v，其子树大小大于 r，且它的每个儿子子树大小都小于 l。

由于 DFS 树只有返祖边，非树边中只有 v 子树内连向 v 子树外的边的是有用的。假设是从子树内的 x 连向子树外的 y，x 到 v 路径上最后一个点是 z，那么我们可以删除 (v, z) 这条边，加上 (x, y) 这条边，这个操作相当于删掉了 z 这个子树。

假设当前子树大小为 $p > r$，删去一个大小为 $q < l$ 的子树，其大小 $p - q > r - l \geq l$，即删去一个子树后不会低于下界，所以能删多少就删多少是最优的。枚举所有非树边并删除对应的子树，直到 v 的子树大小在 $[l, r]$ 内时停止。如果到最后 v 的子树大小仍然大于 r，则无解，否则我们就找到了一组解。

较精细的实现可以做到 $O(n + m)$ 的时间复杂度。

Counting Cactus

Statement[a]

[a] Source: XX OpenCup GP of Kazan, Problem C

给定一张 n 个点的无向图，求图的生成仙人掌个数。

$n \leq 13$ 。

Counting Cactus

考虑对最终的仙人掌的圆方树进行树形 DP。

Counting Cactus

考虑对最终的仙人掌的圆方树进行树形 DP。

状态：

- $f(v, S)$ 表示以 v 为根，子树中的点集为 S 的方案数。
- $g(v, S)$ 表示以 v 为根，子树中的点集为 S，且 v 只跟一个点相连或只在一个环中的方案数。
- $h(v, u, S)$ 表示对一个环进行 DP，当前环走到了 v，终点是 u，子树中的点集为 S 的方案数。

简单讨论转移即可。时间复杂度 $O(3^n n^2)$。

Bonus：通过一些数学推导和子集卷积将复杂度优化到 $O(2^n poly(n))$。

Determinant

Statement[a]

[a]Source: XX OpenCup GP of Kazan, Problem D

给定一张 n 个点 m 条边的无向图，满足对于其中任意一个大小为 $k+1$ 的点集 A ，存在两个点 $a, b \in A$ 和一条边 e ，使得 a 到 b 的所有路径都经过 e 。求这个图的邻接矩阵的行列式。

$n \leq 2.5 \times 10^4, m \leq 5 \times 10^5, k \leq 25$ 。

Determinant

观察一：图的邻接矩阵的行列式等价于将它划分成若干个环，对于每种方案其权值为 -1 的偶环个数次方，对所有方案的权值求和。(考虑行列式的定义与线性求排列逆序对奇偶性的方法)

观察二：题目中的条件等价于每个边双的大小都不超过 k。

Determinant

观察一：图的邻接矩阵的行列式等价于将它划分成若干个环，对于每种方案其权值为 -1 的偶环个数次方，对所有方案的权值求和。(考虑行列式的定义与线性求排列逆序对奇偶性的方法)

观察二：题目中的条件等价于每个边双的大小都不超过 k。

将边双缩点建出对应的树形结构，在这个结构上进行树形 DP。一个子树有两种状态：根已经被子树内的环覆盖了；根被它与父亲的边连成的环覆盖。

为了区分两种状态，对于每个节点新建一个虚点，虚点与自己连自环表示第一种状态，虚点与实点成环表示第二种状态。合理设计边权，新的图的行列式即为要求的 DP 值。

时间复杂度 $O(nk^2)$ 。

Fast Spanning Tree

Statement[a]

[a] Source: XX OpenCup GP of Kazan, Problem F

有一个 n 个点的无向图，每个点有一个点权，图中初始没有任何边。给 m 个 (a_i, b_i, s_i) ，不断执行以下操作直到不能执行为止：

- 选取最小的 i ，使得 a_i 和 b_i 不连通，且 a_i 所在的连通块点权和加 b_i 所在的连通块点权和大于等于 s_i ，然后在 a_i 和 b_i 之间连一条边。

复原连边的过程。

$n, m \leq 3 \times 10^5, s_i \leq 10^6$ 。

Fast Spanning Tree

引理：如果 $a+b \geq s$，则 $a \geq \frac{s}{2}$ 或 $b \geq \frac{s}{2}$。

Fast Spanning Tree

引理：如果 $a+b \geq s$，则 $a \geq \frac{s}{2}$ 或 $b \geq \frac{s}{2}$。

对于每条边，在 a_i 或 b_i 的连通块点权和大于等于 $\frac{s_i}{2}$ 时检验是否能加入这条边。如果不能，去掉它们已有的点权和，新的 s_i 变成了至多原来 s_i 的一半。

加上启发式合并，时间复杂度 $O(n \log n \log V)$。

Honorable Mention

中国计算机学会 China Computer Federation

Statement[a]

[a] Source: XX OpenCup GP of Kazan, Problem H

给一个长度为 n 的序列，q 次询问某个区间的 k-最大子段和。

$n, q \leq 3.5 \times 10^4$ 。

Honorable Mention

引理：考虑对于一段区间，记 $f(k)$ 表示这个区间的 k-最大子段和，则 f 是凸函数。

Honorable Mention

中国计算机学会 China Computer Federation

引理：考虑对于一段区间，记 $f(k)$ 表示这个区间的 k-最大子段和，则 f 是凸函数。

建出线段树，对于每个区间 $[l, r]$ ，预处理出 $f(1), f(2), \cdots, f(r-l+1)$ 。在合并两个函数时，注意到函数是凸的，因此差分之后是单调的，可以贪心合并。

Honorable Mention

中国计算机学会 China Computer Federation

引理：考虑对于一段区间，记 $f(k)$ 表示这个区间的 k-最大子段和，则 f 是凸函数。

建出线段树，对于每个区间 $[l, r]$ ，预处理出 $f(1), f(2), \cdots, f(r-l+1)$ 。在合并两个函数时，注意到函数是凸的，因此差分之后是单调的，可以贪心合并。

对于答案是凸函数，要求选恰好 k 个的问题，有一个常见的技巧：因为一定存在 a 使得函数 $f(x)+ax$ 在 k 处取得极值，我们可以通过二分 a 找到恰好选 k 个的最优方案。在这个问题中，相当于每多选一个子段，会额外获得 a 的收益。

在线段树上定位出的区间二分出对应 a 的最优的 k ，并用一个 DP 合并这些信息即可。

Honorable Mention

引理：考虑对于一段区间，记 $f(k)$ 表示这个区间的 k-最大子段和，则 f 是凸函数。

建出线段树，对于每个区间 $[l, r]$ ，预处理出 $f(1), f(2), \cdots, f(r-l+1)$ 。在合并两个函数时，注意到函数是凸的，因此差分之后是单调的，可以贪心合并。

对于答案是凸函数，要求选恰好 k 个的问题，有一个常见的技巧：因为一定存在 a 使得函数 $f(x)+ax$ 在 k 处取得极值，我们可以通过二分 a 找到恰好选 k 个的最优方案。在这个问题中，相当于每多选一个子段，会额外获得 a 的收益。

在线段树上定位出的区间二分出对应 a 的最优的 k ，并用一个 DP 合并这些信息即可。

如果将二分改成整体二分，时间复杂度降低为 $O((n+q)\log n \log V)$ 。

Bulbasaur

中国计算机学会
China Computer Federation

Statement[a]

[a] Source: XX OpenCup GP of Warsaw, Problem B

有一个 nm 个点的有向分层图，共有 n 层，每层 m 个点，每条边一定是从第 i 层连向第 $i+1$ 层。
定义 $f(i,j)$ 表示选择若干条路径，每条路径从第 i 层出发，在第 j 层结束，且每条路径在顶点和边上都不交的情况下，最多选择的路径数。
求 $\sum_{i=1}^{n}\sum_{j=i+1}^{n} f(i,j)$ 。
$n \le 4\times 10^4, m \le 9$ 。

Bulbasaur

最大流等于最小割，问题可以转化为删除尽量少的点使得两边不连通。

Bulbasaur

中国计算机学会
China Computer Federation

最大流等于最小割，问题可以转化为删除尽量少的点使得两边不连通。

考虑第 l 层到第 r 层，记 $f(S)$ 表示使得仅有第 r 层的点集 S 与第 l 层的点连通，最少需要删掉的点数。不难发现，$f(S)$ 关于 l 是单调的，且 $f(S)$ 的值只有 m 种。

Bulbasaur

中国计算机学会 China Computer Federation

最大流等于最小割，问题可以转化为删除尽量少的点使得两边不连通。

考虑第 l 层到第 r 层，记 $f(S)$ 表示使得仅有第 r 层的点集 S 与第 l 层的点连通，最少需要删掉的点数。不难发现，$f(S)$ 关于 l 是单调的，且 $f(S)$ 的值只有 m 种。

从小到大枚举 r ，维护不同 DP 值的分界点转移即可。时间复杂度 $O(nm^2 2^m)$ 。

Eevee

Statement[a]

[a] Source: XX OpenCup GP of Warsaw, Problem E

定义对 k 个长度为 n 的序列的归并为，维护一个空序列 A ，执行下列操作 kn 次：

- 选择一个非空序列，将其第一个数取出，放入 A 的末尾。

两个归并不同，当且仅当存在某次操作，选取的序列标号不同。
一个归并被称为是合法的，当且仅当最终序列 A 中不存在一个长度为 k 的子串，使得这个子串中的数为同一个数。
给定 m 个长度为 n 的**随机生成的**排列，定义 $f(l, r)$ 表示只考虑 $[l, r]$ 中的排列不同的合法归并方法数，求 $\sum_{i=1}^{m} \sum_{j=i+1}^{m} f(i, j)$ 的值。
$n, m \leq 300$ 。

Eevee

考虑容斥，记 $f(x)$ 表示归并到了某个时刻，第一次出现连续的 k 个 x 的方案数。转移则是用总方案数减去在之前出现连续的 k 个 y 的方案数。

Eevee

考虑容斥，记 $f(x)$ 表示归并到了某个时刻，第一次出现连续的 k 个 x 的方案数。转移则是用总方案数减去在之前出现连续的 k 个 y 的方案数。

枚举区间左端点，每次新增一个排列，可以 $O(1)$ 计算出新的转移系数，这样的时间复杂度为 $O(n^2m^2)$ 。

注意到数据是随机的，考虑一对 (x, y) ，其转移系数非 0 的概率为 $\frac{1}{2^k}$ ，将 k 从 2 到 n 的求和的结果是 $O(1)$ 的。

精细的实现可以做到 $O(nm(n+m))$ 的时间复杂度。

Infernape

Statement[a]

[a]Source: XX OpenCup GP of Warsaw, Problem I

给一个 n 个点的树，q 次询问，每次给 k_i 个树上邻域（离 v_i 距离不超过 r_i 的点集），求至少属于 $k_i - 1$ 个点集的顶点个数。

$n \leq 10^5, \sum k_i \leq 3 \times 10^5$ 。

Infernape

中国计算机学会
China Computer Federation

定义广义树上邻域为树上邻域或者距离一条边距离不超过一定距离的点集，有结论：广义树上邻域的交仍然是广义树上邻域。

为了方便实现，将树上的每条边拆点，则只需考虑树上邻域。

Infernape

定义广义树上邻域为树上邻域或者距离一条边距离不超过一定距离的点集，有结论：广义树上邻域的交仍然是广义树上邻域。

为了方便实现，将树上的每条边拆点，则只需考虑树上邻域。

可以用一些讨论和树上倍增快速合并两个树上邻域，并通过预处理点分树快速求出一个树上邻域的点数。对于每个询问，枚举去掉一个树上邻域，将其他树上邻域的交的点数求和，再去掉算重的部分，即所有树上邻域的交即可。时间复杂度 $O((n+\sum k_i)\log n)$ 。

Container

China Computer Federation

Statement[a]

[a] Source: XX OpenCup GP of Korea, Problem D

给一个长度为 n 的只含有 1 和 2 的字符串，你可以执行以下操作：

- 选择一个长度不超过 3 的区间将它翻转，代价为区间内所有数的和加上一个给定的常数 c 。

求变成目标串的最小代价。

$n \le 500$ 。

Container

能进行的操作只有三种：$12 \leftrightarrow 21, 112 \leftrightarrow 211, 122 \leftrightarrow 221$ 。

Container

中国计算机学会 China Computer Federation

能进行的操作只有三种：$12 \leftrightarrow 21, 112 \leftrightarrow 211, 122 \leftrightarrow 221$ 。

将序列看成是一条路径，1 表示向上走，2 表示向右走，则一次操作对应一个 $1 \times 1, 1 \times 2, 2 \times 1$ 的矩形修改。

Container

能进行的操作只有三种：$12 \leftrightarrow 21, 112 \leftrightarrow 211, 122 \leftrightarrow 221$ 。

将序列看成是一条路径，1 表示向上走，2 表示向右走，则一次操作对应一个 $1 \times 1, 1 \times 2, 2 \times 1$ 的矩形修改。

先将初始解设为全部用 1×1 的覆盖，接下来我们可以合并两个 1×1 的方形得到一定的收益。注意到这是一个二分图，求出最小费用最大流即可。精细的实现可以做到 $O(n^3 \log n)$。

Bonus：用 DP 求解，时间复杂度 $O(n^2)$ 。

Angle Beats

中国计算机学会
China Computer Federation

Statement[a]

[a] Source: XIX OpenCup GP of Kazan, Problem A

给一个 $n \times m$ 的棋盘，每个位置是 ABC 三种状态，你要在上面放上尽量多的多米诺（每个多米诺覆盖三个格子），要求：

- 放的多米诺如果是 L 形，则其中心必须为 A 或 B，两边必须为 C。
- 放的多米诺如果是 I 形，则其中心必须为 A，两边必须为 C。

求最多能放的多米诺的个数。

$n, m \leq 100$ 。

Angle Beats

如果没有状态 B，问题等价于：给定一张二分图，左边每个点如果在匹配中，要匹配右边的两个点，求最大匹配。

Angle Beats

中国计算机学会 China Computer Federation

如果没有状态 B，问题等价于：给定一张二分图，左边每个点如果在匹配中，要匹配右边的两个点，求最大匹配。

将左边每个点拆成两个，它们之间连一条边。只有当两个点同时向外界连边时才会有收益。

Angle Beats

中国计算机学会 China Computer Federation

如果没有状态 B，问题等价于：给定一张二分图，左边每个点如果在匹配中，要匹配右边的两个点，求最大匹配。

将左边每个点拆成两个，它们之间连一条边。只有当两个点同时向外界连边时才会有收益。

对于状态 B，只需要将它拆成的两个点一个连横向边一个连纵向边即可。最后使用带花树求解最大匹配，时间复杂度 $O(n^2m^2)$ 。

Dates

中国计算机学会 China Computer Federation

Statement[a]

[a]Source: XIX OpenCup GP of Kazan, Problem D

有 m 个盒子，第 i 个盒子里可以放 a_i 个球。有 n 个球，第 i 个球如果被放在 $[l_i, r_i]$ 之间的某个盒子里，可以获得 p_i 的收益。求最大收益。
保证 $l_i \le l_{i+1}, r_i \le r_{i+1}$ 。
$n, m \le 3 \times 10^5$ 。

Dates

这是个经典的拟阵模型，可以使用贪心求解。

Dates

这是个经典的拟阵模型，可以使用贪心求解。

我们需要快速判断加入某个球之后是否还能将当前的所有球放入盒子里，这是一个二分图上的网络流问题，考虑使用 Hall 定理来加速判断。

Dates

这是个经典的拟阵模型，可以使用贪心求解。

我们需要快速判断加入某个球之后是否还能将当前的所有球放入盒子里，这是一个二分图上的网络流问题，考虑使用 Hall 定理来加速判断。

考虑 Hall 定理在区间上的版本，则对于任意 $x \le y$ 我们需要满足 $\sum_{i=l_x}^{r_y} a_i \ge w_y - w_x + 1$ ，其中 w_x 是当前所有球中 x 的排名。

用前缀和将式子分成只与 x 和 y 有关的两部分，分别维护它们的极值即可。时间复杂度 $O(m + n \log n)$ 。

Graph Counting

China Computer Federation 中国计算机学会

Statement[a]

[a] Source: XIX OpenCup GP of Kazan, Problem G

求 $2n$ 个点无标号图的个数，满足图中没有完美匹配，但加了任意一条边就有完美匹配。

$n \leq 5 \times 10^5$ 。

Graph Counting

根据 Tutte-Berge 公式，图 $G=(V,E)$ 的最大匹配为：

$$\frac{1}{2}\min_{U\subseteq V}(|U|-odd(G-U)+|V|)$$

其中 $odd(H)$ 表示 H 中大小为奇数的连通块个数。

Graph Counting

中国计算机学会
China Computer Federation

根据 Tutte-Berge 公式，图 $G=(V,E)$ 的最大匹配为：

$$\frac{1}{2}\min_{U\subseteq V}(|U|-odd(G-U)+|V|)$$

其中 $odd(H)$ 表示 H 中大小为奇数的连通块个数。

由于原图中最大匹配为 $n-1$，有 $\min_{U\subseteq V}(|U|-odd(G-U))=-2$。考虑这样的一个点集 U，如果 U 中某个节点度数不为 $|V|-1$，则添加一条连接 U 中某个点到其它点的边，这个值仍然不变，不合法；同理 $G-U$ 每个连通块也必须是团。

Graph Counting

中国计算机学会 China Computer Federation

设有 k 个度数为 $|V|-1$ 的点，那么删去它们后，应有 $k+2$ 个奇数大小的团，显然这样的图也满足加一条边后就有完美匹配的条件。对于每个团，我们将其点数加上 1 变成偶数，同时不管那些度数为 $|V|-1$ 的点 (这样的点数由团的个数唯一确定)，则现在总的点数为 $2n+2$ ，且每个团大小都是偶数。

答案为 $n+1$ 的整数拆分方案数减 1 （因为至少有两个团)，用多项式 exp 求解即可。时间复杂度 $O(n\log n)$ 。

Hall's Theorem

中国计算机学会 China Computer Federation

Statement[a]

[a]Source: XIX OpenCup GP of Kazan, Problem H

构造一个两边各 n 个点的二分图，记 $N(A)$ 表示与 A 中的点有直接连边的点集，你需要使得 $|N(A)| < |A|$ 的点集个数恰好为 k 。

$n \leq 20$ 。

Hall’s Theorem

将问题转化成构造恰好 k 个 A 使得 $|N(A)| \geq |A|$ 。

考虑这样一种图：左边第 i 个点连接右边的前 a_i 个点，且 a_i 递增，我们有：

$$k = \sum_{i=1}^{n} \sum_{j=1}^{a_i} \binom{i-1}{j-1}$$

这是一个类似组合数进制拆分的东西，贪心构造即可。时间复杂度 $O(n^2)$ 。

Jealous Split

Statement[a]

[a]Source: XIX OpenCup GP of Kazan, Problem J

给一个长度为 n 的序列，将其划分为 k 部分，记 s_i 表示第 i 部分的和，m_i 表示第 i 部分的最大值，你需要构造这样一个划分使得：
$|s_i - s_{i+1}| \leq \max(m_i, m_{i+1})$ 对所有 i 成立。
$n \leq 10^5$ 。

Jealous Split

考虑任意一种不合法的划分，我们移动划分点使得原来不合法的相邻两部分变得合法。

不难发现，调整之后每段和的平方的和变小了。

Jealous Split

中国计算机学会 China Computer Federation

考虑任意一种不合法的划分，我们移动划分点使得原来不合法的相邻两部分变得合法。

不难发现，调整之后每段和的平方的和变小了。

因此，我们只需要找到一种最小化每段和的平方的和的方案即可。答案是关于 k 的凸函数，二分斜率之后斜率优化。时间复杂度 $O(n \log V)$ 。

Busy Board

中国计算机学会 China Computer Federation

Statement[a]

[a] Source: XIX OpenCup GP of North America, Problem B

给一个 $n \times m$ 的黑白网格，你可以不断执行以下操作：

- 选择一个黑色格子，将它变白，并把同行同列的其他格子变成黑色。

求一个状态能否经过若干次操作到达另一个状态。

$n, m \leq 10^3$ 。

Busy Board

中国计算机学会 China Computer Federation

首先特判掉始末状态相同的情况。

Busy Board

首先特判掉始末状态相同的情况。

倒过来操作，每次操作是将一个白色格子变黑，要求它同行同列全是黑的，然后把它同行同列的格子改成任意的颜色。

Busy Board

首先特判掉始末状态相同的情况。

倒过来操作，每次操作是将一个白色格子变黑，要求它同行同列全是黑的，然后把它同行同列的格子改成任意的颜色。

执行一个类似于 BFS 的过程，可以进行操作的中心格子一定形如 $\{(r, c) \mid r \in R, c \in C\}$ 的形式，其中 R 和 C 分别是行和列的集合。

Busy Board

首先特判掉始末状态相同的情况。

倒过来操作，每次操作是将一个白色格子变黑，要求它同行同列全是黑的，然后把它同行同列的格子改成任意的颜色。

执行一个类似于 BFS 的过程，可以进行操作的中心格子一定形如 $\{(r, c) \mid r \in R, c \in C\}$ 的形式，其中 R 和 C 分别是行和列的集合。

合法状态需要满足：初始状态需要能进行操作，且对于其他不在 R 和 C 中的格子，它们的颜色保持不变。

精细的实现可以做到 $O(nm)$ 。

Hat With An Integer

Statement[a]

[a]Source: XIX OpenCup GP of Bytedance, Problem H

有 n 个人，第 i 个人手上有一个数 a_i ，每个人能看到别人的数，但不能看到自己的数。每个人还知道，在以下 $2(n-1)$ 个条件中，至少有一个是成立的：

- $a_{i+1} < a_i + b_i$
- $a_{i+1} > a_i + c_i$

每个人都是绝顶聪明的。如果有一天某个人知道自己的数不是 x ，那游戏就结束了。求游戏会在第几天结束。

$n \le 10^5$ 。

Hat With An Integer

结论：原问题等价于修改尽量少的 a_i ，使得这些条件都不成立。

Hat With An Integer

结论：原问题等价于修改尽量少的 a_i ，使得这些条件都不成立。

记 B 和 C 表示 b 和 c 的前缀和，相当于我们要保留尽量多的 a_i ，使得对于任意一对被保留的相邻的 (i, j) ，我们有：

- $a_i + (B_j - B_i) \leq a_j \Leftrightarrow a_i - B_i \leq a_j - B_j$
- $a_i + (C_j - C_i) \geq a_j \Leftrightarrow a_i - C_i \geq a_j - C_j$

问题转化成三维最长上升子序列问题，用二维数据结构或分治解决即可。
可以更优吗?

Hat With An Integer

结论：原问题等价于修改尽量少的 a_i ，使得这些条件都不成立。

记 B 和 C 表示 b 和 c 的前缀和，相当于我们要保留尽量多的 a_i ，使得对于任意一对被保留的相邻的 (i, j) ，我们有：

- $a_i + (B_j - B_i) \leq a_j \Leftrightarrow a_i - B_i \leq a_j - B_j$
- $a_i + (C_j - C_i) \geq a_j \Leftrightarrow a_i - C_i \geq a_j - C_j$

问题转化成三维最长上升子序列问题，用二维数据结构或分治解决即可。可以更优吗?

注意到如果 $a_i - B_i$ 和 $a_i - C_i$ 都是有序的话，i 也一定是有序的。时间复杂度 $O(n \log n)$ 。