7 月 4 日模拟赛题解

mcfx

2019 年 7 月 4 日

枚举每个排列，判断是否合法。

欢迎哪位同学上来讲一下。

请 sunshiness 来讲一下。

考虑计算有多少个排列 P，使得 $F(P) = Q$，其中 Q 是一个给定的排列。

考虑计算有多少个排列 P，使得 $F(P) = Q$，其中 Q 是一个给定的排列。
显然 $Q_N = N$，而 Q 中只有是前缀最大值的数是有可能是从前面换过来的。考虑每个前缀最大值是否是从前面换过来的，可以得到方案数为 2^k，其中 k 表示 $1 \sim N - 1$ 的前缀最大值个数。

考虑计算有多少个排列 P，使得 $F(P) = Q$，其中 Q 是一个给定的排列。
显然 $Q_N = N$，而 Q 中只有是前缀最大值的数是有可能是从前面换过来的。考虑每个前缀最大值是否是从前面换过来的，可以得到方案数为 2^k，其中 k 表示 $1 \sim N - 1$ 的前缀最大值个数。
考虑 dp，记 $dp(i, j)$ 表示当前在 i 位置，前 i 个位置的最大值是 j 的权值和。权值和是指所有满足要求的排列的 2^k 之和。当最大值变化时，dp 值需要 $*2$。

考虑计算有多少个排列 P，使得 $F(P) = Q$，其中 Q 是一个给定的排列。
显然 $Q_N = N$，而 Q 中只有是前缀最大值的数是有可能是从前面换过来的。考虑每个前缀最大值是否是从前面换过来的，可以得到方案数为 2^k，其中 k 表示 $1 \sim N-1$ 的前缀最大值个数。
考虑 dp，记 $dp(i,j)$ 表示当前在 i 位置，前 i 个位置的最大值是 j 的权值和。权值和是指所有满足要求的排列的 2^k 之和。当最大值变化时，dp 值需要 $*2$。
转移时，分两种情况考虑。

考虑计算有多少个排列 P，使得 $F(P) = Q$，其中 Q 是一个给定的排列。
显然 $Q_N = N$，而 Q 中只有是前缀最大值的数是有可能是从前面换过来的。考虑每个前缀最大值是否是从前面换过来的，可以得到方案数为 2^k，其中 k 表示 $1 \sim N-1$ 的前缀最大值个数。
考虑 dp，记 $dp(i,j)$ 表示当前在 i 位置，前 i 个位置的最大值是 j 的权值和。权值和是指所有满足要求的排列的 2^k 之和。当最大值变化时，dp 值需要 $*2$。
转移时，分两种情况考虑。
如果 $i+1$ 位置的数已经固定（设为 x），那么 $dp(i,j)$ 会转移到 $dp(i+1, max(j,x))$。

考虑计算有多少个排列 P，使得 $F(P) = Q$，其中 Q 是一个给定的排列。
显然 $Q_N = N$，而 Q 中只有是前缀最大值的数是有可能是从前面换过来的。考虑每个前缀最大值是否是从前面换过来的，可以得到方案数为 2^k，其中 k 表示 $1 \sim N-1$ 的前缀最大值个数。
考虑 dp，记 $dp(i,j)$ 表示当前在 i 位置，前 i 个位置的最大值是 j 的权值和。权值和是指所有满足要求的排列的 2^k 之和。当最大值变化时，dp 值需要 $*2$。
转移时，分两种情况考虑。
如果 $i+1$ 位置的数已经固定（设为 x），那么 $dp(i,j)$ 会转移到 $dp(i+1, max(j,x))$。
如果 $i+1$ 位置的数没有固定，可以枚举这个位置的值，然后进行转移。

考虑计算有多少个排列 P，使得 $F(P) = Q$，其中 Q 是一个给定的排列。
显然 $Q_N = N$，而 Q 中只有是前缀最大值的数是有可能是从前面换过来的。考虑每个前缀最大值是否是从前面换过来的，可以得到方案数为 2^k，其中 k 表示 $1 \sim N-1$ 的前缀最大值个数。
考虑 dp，记 $dp(i,j)$ 表示当前在 i 位置，前 i 个位置的最大值是 j 的权值和。权值和是指所有满足要求的排列的 2^k 之和。当最大值变化时，dp 值需要 $*2$。
转移时，分两种情况考虑。
如果 $i+1$ 位置的数已经固定（设为 x），那么 $dp(i,j)$ 会转移到 $dp(i+1, max(j,x))$。
如果 $i+1$ 位置的数没有固定，可以枚举这个位置的值，然后进行转移。
总复杂度 $O(n^3)$，根据实现不同可以得到 $40 \sim 60$ 分。

上面的 dp，当 $i+1$ 位置的数没有固定时，发现可以前缀和优化，那么总复杂度是 $O(n^2)$。

每一段连续的 1 单独是一个矩阵。

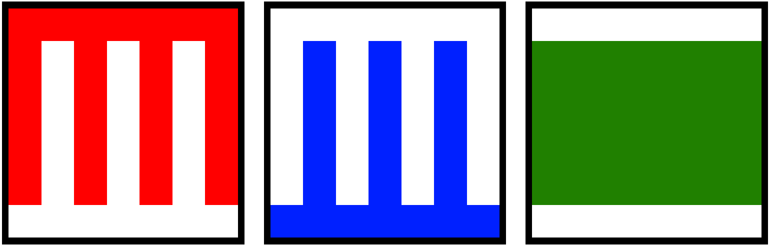
暴搜。

在暴搜过程中发现 $max(n, m) \geq 2$ 时，答案不超过 3。根据这个性质优化搜索过程。

当 $max(n, m) \geq 3$ 时，可以如下构造：

当 $max(n,m) \geq 3$ 时，可以如下构造：
考虑下面这样三个矩阵，红 $+$ 蓝 $-$ 绿得到的矩阵是一个第一行和最后一行全是 1，其他地方全是 0 的矩阵。

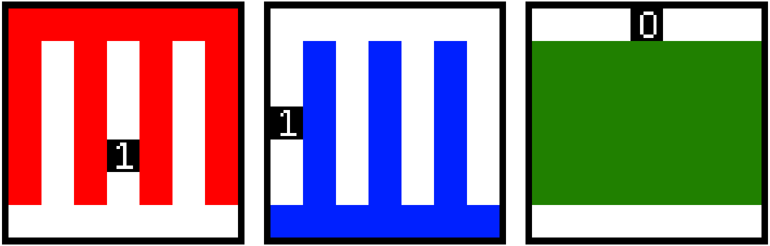
当 $max(n, m) \geq 3$ 时，可以如下构造：
考虑下面这样三个矩阵，红 $+$ 蓝 $-$ 绿得到的矩阵是一个第一行和最后一行全是 1，其他地方全是 0 的矩阵。



那么如果需要把中间某个位置变成 1，可以在红或蓝矩阵中的对应位置加一个 1。

那么如果需要把中间某个位置变成 1，可以在红或蓝矩阵中的对应位置加一个 1。
如果需要把第一行或最后一行某个位置变成 0，可以在绿矩阵中的对应位置加一个 1。

那么如果需要把中间某个位置变成 1，可以在红或蓝矩阵中的对应位置加一个 1。
如果需要把第一行或最后一行某个位置变成 0，可以在绿矩阵中的对应位置加一个 1。



首先特判 $min(n, m) = 1$。

首先特判 $min(n,m)=1$。
当 1 连通块个数为 1 时，答案为 1。

首先特判 $min(n, m) = 1$。
当 1 连通块个数为 1 时，答案为 1。
当 1 连通块个数为 2 时，答案为 2。

首先特判 $min(n, m) = 1$。
当 1 连通块个数为 1 时，答案为 1。
当 1 连通块个数为 2 时，答案为 2。
当有一个 0 连通块连接了所有 1 连通块时，答案也为 2。

首先特判 $min(n,m)=1$。
当 1 连通块个数为 1 时，答案为 1。
当 1 连通块个数为 2 时，答案为 2。
当有一个 0 连通块连接了所有 1 连通块时，答案也为 2。
这时 $max(n,m)\geq 3$，可以使用上面的构造。

可以预处理出每对点之间是否在 B 上是连通块。

可以预处理出每对点之间是否在 B 上是连通块。
然后可以暴力求出所有点对之间的答案。

把询问离线，然后考虑扫描线。

把询问离线，然后考虑扫描线。
一个区间 $[l, r]$ 在 B 中也是区间当且仅当这个区间在 B 中位置的 $\max - \min = r - l$。

把询问离线，然后考虑扫描线。
一个区间 $[l, r]$ 在 B 中也是区间当且仅当这个区间在 B 中位置的 $\max - \min = r - l$。
那么可以用两个单调栈维护 B 中位置的 $\max$ 和 $\min$，再用线段树维护 $\max - \min - (r - l)$。

把询问离线，然后考虑扫描线。
一个区间 $[l, r]$ 在 B 中也是区间当且仅当这个区间在 B 中位置的 $\max - \min = r - l$。
那么可以用两个单调栈维护 B 中位置的 $\max$ 和 $\min$，再用线段树维护 $\max - \min - (r - l)$。
这个线段树需要支持区间加，求历史最小值及其出现的次数。

把询问离线，然后考虑扫描线。
一个区间 $[l, r]$ 在 B 中也是区间当且仅当这个区间在 B 中位置的 $\max - \min = r - l$。
那么可以用两个单调栈维护 B 中位置的 $\max$ 和 $\min$，再用线段树维护 $\max - \min - (r - l)$。
这个线段树需要支持区间加，求历史最小值及其出现的次数。
可以在每个节点维护最小值、最小值个数、加标记、加标记的历史最小值、加标记的历史最小值个数、历史最小值、历史最小值个数。

把询问离线，然后考虑扫描线。
一个区间 $[l, r]$ 在 B 中也是区间当且仅当这个区间在 B 中位置的 $\max - \min = r - l$。
那么可以用两个单调栈维护 B 中位置的 $\max$ 和 $\min$，再用线段树维护 $\max - \min - (r - l)$。
这个线段树需要支持区间加，求历史最小值及其出现的次数。
可以在每个节点维护最小值、最小值个数、加标记、加标记的历史最小值、加标记的历史最小值个数、历史最小值、历史最小值个数。
pushdown 时考虑用加标记的历史最小值去更新历史最小值。

同样考虑离线扫描线。

同样考虑离线扫描线。
一个区间 $[l, r]$ 在 B 中是连通块当且仅当这个区间在 B 中的点数减边数 $= 1$。

同样考虑离线扫描线。
一个区间 $[l, r]$ 在 B 中是连通块当且仅当这个区间在 B 中的点数减边数 $= 1$。
每当当前考虑的 r 变大时，可以找出 A_{r+1} 在 B 中相连的点 x，然后对 $[1, pos\ of\ x\ in\ A]$ 打标记。

同样考虑离线扫描线。
一个区间 $[l, r]$ 在 B 中是连通块当且仅当这个区间在 B 中的点数减边数 $= 1$。
每当当前考虑的 r 变大时，可以找出 A_{r+1} 在 B 中相连的点 x，然后对 $[1, pos\ of\ x\ in\ A]$ 打标记。
同样用上面说的这么一棵线段树维护就可以了。

之前的做法中，线段树中一个位置 x 表示的是 x 到当前考虑的点的路径中，所有包含 x 的子路径的正确性之和。

之前的做法中，线段树中一个位置 x 表示的是 x 到当前考虑的点的路径中，所有包含 x 的子路径的正确性之和。
如果 dfs 一遍整棵树，dfs 过程中考虑每个节点到其子树中节点的边，那么可以让线段树中一个位置表示 x 到当前节点的这坨东西之和。

之前的做法中，线段树中一个位置 x 表示的是 x 到当前考虑的点的路径中，所有包含 x 的子路径的正确性之和。
如果 dfs 一遍整棵树，dfs 过程中考虑每个节点到其子树中节点的边，那么可以让线段树中一个位置表示 x 到当前节点的这坨东西之和。
那么对于到祖先的路径，可以直接在祖先处求出一条链上的这坨东西之和。可以树剖解决。

之前的做法中，线段树中一个位置 x 表示的是 x 到当前考虑的点的路径中，所有包含 x 的子路径的正确性之和。
如果 dfs 一遍整棵树，dfs 过程中考虑每个节点到其子树中节点的边，那么可以让线段树中一个位置表示 x 到当前节点的这坨东西之和。
那么对于到祖先的路径，可以直接在祖先处求出一条链上的这坨东西之和。可以树剖解决。
而对于其他路径，可以考虑 dsu on tree。在 lca 处考虑，两个端点中一定有一个不在重链上，那么可以暴力递归那边，然后统计另一端的贡献。

之前的做法中，线段树中一个位置 x 表示的是 x 到当前考虑的点的路径中，所有包含 x 的子路径的正确性之和。
如果 dfs 一遍整棵树，dfs 过程中考虑每个节点到其子树中节点的边，那么可以让线段树中一个位置表示 x 到当前节点的这坨东西之和。
那么对于到祖先的路径，可以直接在祖先处求出一条链上的这坨东西之和。可以树剖解决。
而对于其他路径，可以考虑 dsu on tree。在 lca 处考虑，两个端点中一定有一个不在重链上，那么可以暴力递归那边，然后统计另一端的贡献。
暴力递归时需要在线段树上撤销，可以直接记录下做出的修改，然后把原来的节点放回去。

之前的做法中，线段树中一个位置 x 表示的是 x 到当前考虑的点的路径中，所有包含 x 的子路径的正确性之和。

如果 dfs 一遍整棵树，dfs 过程中考虑每个节点到其子树中节点的边，那么可以让线段树中一个位置表示 x 到当前节点的这坨东西之和。

那么对于到祖先的路径，可以直接在祖先处求出一条链上的这坨东西之和。可以树剖解决。

而对于其他路径，可以考虑 dsu on tree。在 lca 处考虑，两个端点中一定有一个不在重链上，那么可以暴力递归那边，然后统计另一端的贡献。

暴力递归时需要在线段树上撤销，可以直接记录下做出的修改，然后把原来的节点放回去。

复杂度看起来是 $O((N+M)\log^2 N)$，但其实可以分析到 $O(N\log N+M\log^2 N)$。

假设当前考虑 x，要暴力递归的是 y 的子树。

假设当前考虑 x，要暴力递归的是 y 的子树。
一条两端都在 y 子树中的边，如果是返祖边，会产生 1 次全局修改；否则不会有影响。这样的边总共会在 $\log N$ 条重链上被暴力到，总复杂度 $O(N \log N)$。

假设当前考虑 x，要暴力递归的是 y 的子树。
一条两端都在 y 子树中的边，如果是返祖边，会产生 1 次全局修改；否则不会有影响。这样的边总共会在 $\log N$ 条重链上被暴力到，总复杂度 $O(N \log N)$。
一条一端在 y 子树中，另一端在 x 的其他子树中的边，会产生 1 次区间修改，但是只会在暴力处理 x 的儿子时会处理到，总复杂度 $O(N \log N)$。

假设当前考虑 x，要暴力递归的是 y 的子树。
一条两端都在 y 子树中的边，如果是返祖边，会产生 1 次全局修改；否则不会有影响。这样的边总共会在 $\log N$ 条重链上被暴力到，总复杂度 $O(N \log N)$。
一条一端在 y 子树中，另一端在 x 的其他子树中的边，会产生 1 次区间修改，但是只会在暴力处理 x 的儿子时会处理到，总复杂度 $O(N \log N)$。
查询复杂度应该也可以通过全局平衡二叉树优化到 $\log N$。