7 月 2 日模拟赛题解

mcfx

2019 年 7 月 2 日

状压 dp 模拟即可。

状压 dp 模拟即可。
如果在本地打表，实际上也可以通过 $N, K \leq 23$ 的测试点。

考虑对每个位置计算贡献。

考虑对每个位置计算贡献。
i 位置上最后剩下的人显然只可能来自 $i \sim i+K$ 这 $K+1$ 个位置。

考虑对每个位置计算贡献。
i 位置上最后剩下的人显然只可能来自 $i \sim i + K$ 这 $K + 1$ 个位置。
那么原问题可以简化为，有多少种方案，使得 $i \sim i + K$ 这 $K + 1$ 个人中，有人最后停留在 i。(对于最后面的 K 个人略有不同)

考虑对每个位置计算贡献。
i 位置上最后剩下的人显然只可能来自 $i \sim i+K$ 这 $K+1$ 个位置。
那么原问题可以简化为，有多少种方案，使得 $i \sim i+K$ 这 $K+1$ 个人中，有人最后停留在 i。(对于最后面的 K 个人略有不同)
为了方便计算，我们考虑统计停留在 i 的人中编号最小的。
那么要计算的也就是有多少种方案，使得 $x+i$ 位置的人能移动到 i 位置，并且每一步移动时，移动到的位置都没有人。

移动到的位置都没有人这个条件，实际上可以记录 $x-1$ 这个人的位置，然后只需要满足这个人的位置小于 x 的位置 -1。
那么可以如下 dp：
$dp(i,j,k)$ 表示：当前是第 i 个操作，x 这个人在 j，$x-1$ 这个人在 k。
转移时枚举三种可能：x 走一步，$x-1$ 走一步，或者其他人走一步。
这样可以做到 $O(n^4)$。

移动到的位置都没有人这个条件，实际上可以记录 $x-1$ 这个人的位置，然后只需要满足这个人的位置小于 x 的位置 -1。
那么可以如下 dp：
$dp(i,j,k)$ 表示：当前是第 i 个操作，x 这个人在 j，$x-1$ 这个人在 k。
转移时枚举三种可能：x 走一步，$x-1$ 走一步，或者其他人走一步。
这样可以做到 $O(n^4)$。
实际上倒着转移就可以做到 $O(n^3)$。

移动到的位置都没有人这个条件，实际上可以记录 $x-1$ 这个人的位置，然后只需要满足这个人的位置小于 x 的位置 -1。
那么可以如下 dp：
$dp(i,j,k)$ 表示：当前是第 i 个操作，x 这个人在 j，$x-1$ 这个人在 k。
转移时枚举三种可能：x 走一步，$x-1$ 走一步，或者其他人走一步。
这样可以做到 $O(n^4)$。
实际上倒着转移就可以做到 $O(n^3)$。
为了避免卡常，数据有一定梯度。
标程只有两个简单的优化。
首先发现 x 一定不超过 $\frac{K}{2}$，dp 状态可以直接删去 $\frac{3}{4}$。
其次 i 不需要记录，处理好转移顺序就可以了。

移动到的位置都没有人这个条件，实际上可以记录 $x-1$ 这个人的位置，然后只需要满足这个人的位置小于 x 的位置 -1。
那么可以如下 dp：
$dp(i,j,k)$ 表示：当前是第 i 个操作，x 这个人在 j，$x-1$ 这个人在 k。
转移时枚举三种可能：x 走一步，$x-1$ 走一步，或者其他人走一步。
这样可以做到 $O(n^4)$。
实际上倒着转移就可以做到 $O(n^3)$。
为了避免卡常，数据有一定梯度。
标程只有两个简单的优化。
首先发现 x 一定不超过 $\frac{K}{2}$，dp 状态可以直接删去 $\frac{3}{4}$。
其次 i 不需要记录，处理好转移顺序就可以了。
如果有其他 dp 做法（包括部分分），欢迎上来分享。

把 $1 \sim n^2$ 的数依次通过 (n, n) 中转移到对应位置即可。

思路同上，用若干直角三角形中转。

手动模拟一些三角形，和用他们怎么进行操作。

要达到 100 分，需要代价不超过 60.15。暴力跑出所有满足要求的三角形，发现共有 1316 个。

要达到 100 分，需要代价不超过 60.15。暴力跑出所有满足要求的三角形，发现共有 1316 个。
考虑一个性质：如果有两个三角形 $(1, 2, 3), (3, 4, 5)$，那么通过下面三次操作：
初始：$[1, 2, 3, 4, 5]$
$(3, 4, 5) \to [1, 2, 5, 3, 4]$
$(1, 2, 3) \to [5, 1, 2, 3, 4]$
$(5, 4, 3) \to [5, 1, 3, 4, 2]$
可以发现，实际上这相当于一次 $(1, 2, 5)$ 的操作。
那么对于一个三角形 (a, b, c)，可以通过若干次这样的中转，把 b 和 c 都替换掉。

这样可以直接套用上面的 5 分做法。

这样可以直接套用上面的 5 分做法。
经暴力验证，所有的三元组都是可以通过这些中转来实现的。具体实现可以每次 bfs，总复杂度 $O(n^6)$。
标程中为了方便，把这 1316 个三角形用表存了下来，不过直接生成也是可以的。

这样可以直接套用上面的 5 分做法。
经暴力验证，所有的三元组都是可以通过这些中转来实现的。具体实现可以每次 bfs，总复杂度 $O(n^6)$。
标程中为了方便，把这 1316 个三角形用表存了下来，不过直接生成也是可以的。
实际上在考场上可以只验证若干组随机数据而不需要在意正确性，即使有构造数据，也可以通过在最开始进行若干次随机操作打乱为近似随机数据。
另外如果把整个方阵写成排列 p，那么有解的条件是 $sgn(p) = 1$。

按题意模拟。
如果要得到 20 分，一种可能的方法是先把点重编号，每次加入链时 dfs 出经过的所有点，存进 bitset，然后 2 操作直接扫一遍。
(总之就是尽可能卡常，13s 时限还是很有希望的，不过我没写过不知道到底能不能过)

把每条链的 l, r 视为坐标，用 kd 树维护，每个节点维护最小值，当小于 0 时暴力删除。
复杂度 $O(n + m\sqrt{m})$，10^5 的部分分是给常数太大的实现准备的。

整体二分。每次可以用树剖维护，或者求出虚树再前缀和之类的。

整体二分。每次可以用树剖维护，或者求出虚树再前缀和之类的。求出每条链的删除时间之后，只需要维护每个点上经过多少条链就能维护权值和，不难用树剖实现。

整体二分。每次可以用树剖维护，或者求出虚树再前缀和之类的。求出每条链的删除时间之后，只需要维护每个点上经过多少条链就能维护权值和，不难用树剖实现。
复杂度 $O((n+m)\log^2 n)$ 或 $O((n+m)\log^3 n)$，根据树剖实现不同。
实际上数据好像没有专门造卡树剖的 $\sqrt{n}$ 条链的数据，所以应该都能过。

考虑点分，那么每条链只会在某个点分树中。（这个树应该叫啥）

考虑点分，那么每条链只会在某个点分树中。(这个树应该叫啥)
每次操作是把 $\log n$ 个点分树中，一端在 x 子树中的链的权值减掉。

考虑点分，那么每条链只会在某个点分树中。(这个树应该叫啥)
每次操作是把 $\log n$ 个点分树中，一端在 x 子树中的链的权值减掉。
但是减去一端的权值之后，另一端的权值会很难维护。

考虑点分，那么每条链只会在某个点分树中。(这个树应该叫啥)
每次操作是把 $\log n$ 个点分树中，一端在 x 子树中的链的权值减掉。
但是减去一端的权值之后，另一端的权值会很难维护。
那么实际上可以直接把权值 w 拆成两个 $\frac{w}{2}$，然后分给两端。当一端的权值小于 0 时，再求出这条链的实际权值，并进行重构。

考虑点分，那么每条链只会在某个点分树中。(这个树应该叫啥)
每次操作是把 $\log n$ 个点分树中，一端在 x 子树中的链的权值减掉。
但是减去一端的权值之后，另一端的权值会很难维护。
那么实际上可以直接把权值 w 拆成两个 $\frac{w}{2}$，然后分给两端。当一端的权值小于 0 时，再求出这条链的实际权值，并进行重构。
重构的总次数是 $O(m \log n)$ 的，每次插入链是 $O(\log n)$ 的。

考虑点分，那么每条链只会在某个点分树中。(这个树应该叫啥)
每次操作是把 $\log n$ 个点分树中，一端在 x 子树中的链的权值减掉。
但是减去一端的权值之后，另一端的权值会很难维护。
那么实际上可以直接把权值 w 拆成两个 $\frac{w}{2}$，然后分给两端。当一端的权值小于 0 时，再求出这条链的实际权值，并进行重构。
重构的总次数是 $O(m\log n)$ 的，每次插入链是 $O(\log n)$ 的。
每次 2 操作会处理 $O(\log n)$ 棵点分树，每次操作是 $O(\log n)$ 的。

考虑点分，那么每条链只会在某个点分树中。(这个树应该叫啥)
每次操作是把 $\log n$ 个点分树中，一端在 x 子树中的链的权值减掉。
但是减去一端的权值之后，另一端的权值会很难维护。
那么实际上可以直接把权值 w 拆成两个 $\frac{w}{2}$，然后分给两端。当一端的权值小于 0 时，再求出这条链的实际权值，并进行重构。
重构的总次数是 $O(m\log n)$ 的，每次插入链是 $O(\log n)$ 的。
每次 2 操作会处理 $O(\log n)$ 棵点分树，每次操作是 $O(\log n)$ 的。
那么总复杂度是 $O(m\log^2 n)$。空间复杂度是 $O(n\log n + m)$。

考虑点分，那么每条链只会在某个点分树中。(这个树应该叫啥)
每次操作是把 $\log n$ 个点分树中，一端在 x 子树中的链的权值减掉。
但是减去一端的权值之后，另一端的权值会很难维护。
那么实际上可以直接把权值 w 拆成两个 $\frac{w}{2}$，然后分给两端。当一端的权值小于 0 时，再求出这条链的实际权值，并进行重构。
重构的总次数是 $O(m\log n)$ 的，每次插入链是 $O(\log n)$ 的。
每次 2 操作会处理 $O(\log n)$ 棵点分树，每次操作是 $O(\log n)$ 的。
那么总复杂度是 $O(m\log^2 n)$。空间复杂度是 $O(n\log n + m)$。
实现上有一些细节，包括两个同一端的点需要用可删除堆维护，以及两端相同的链、递归到最底层需要特殊处理。
标程之前用 set 当可删除堆，在 OJ 上要跑 11s，后来改成了真的可删除堆，稍微快了一点。