刀等数学（ddsx）

假设 1 在 a 中的位置为 p，在 b 中位置为 q，由于 1 始终是堆中最后一个被取出的数，所以显然有 $q \ge p$，并且取出 1 后堆一定是空的，也就是说 a 的前 q 个数和 b 的前 q 个数的集合应该是一样的。

此时我们可以递归下去，因为 $[q+1,n]$ 和 $[1,q]$（但去掉 1）这两个 a 中的段是独立的，再分别考虑其中的最小值即可。

据此可以区间 DP。设 $dp(l,r,v)$ 表示只考虑下标区间 $[l,r]$ 中 $\ge v$ 的数时的答案，那么 $ans = dp(1,n,1)$，而转移就是枚举最小值的位置。

时间复杂度为 $O(n^4)$。

刀言刀语（dydy）

算法 1

每一次操作后花费 $O(n)$ 的时间计算一次满足条件的子串个数。

先处理比较麻烦的 3 操作，因为 3 操作还可以被另一个 3 操作取消，所以我们从后往前看（因为最后一个 3 操作肯定是不会被取消的），遇到一个当前还没有被取消的 3 操作，就将它的取消对象打一个取消标记，其实没有样例解释里看上去那么麻烦。

然后算一次匹配子串的个数，再 $O(n^2)$ 枚举就不行了，这时我们考虑“层”的次序：

我们发现，如果一个括号被另一个括号包起来，那么在这之后它就失效了，如 $(())$，对于中间处在里面的那一对括号来说，无论之后的串长什么样子，它都不能单独参与一个子串的匹配（永远是套在外面的那层括号里的），所以一旦出现了一个 2 操作，它里面包着的所有括号对就直接扔掉了。

记当前的有效括号对数量为 p，那么遇到一个 1 操作，就令 $p = p + 1$；如果遇到一个 2 操作就直接让 $p = 1$，然后更新答案 $ans = ans + p$，表示当前括号对可以和之前的任何有效括号对匹配，这样 $O(n)$ 扫描就可以求出答案。

时间复杂度为 $O(n^2)$，可以解决 $n \le 10^3$ 的部分分。

算法 2

算法 1 中，我们出现一个 2 之后就把前一层扔掉了，但是在有取消的情况下这样就会丢失一些信息。

首先做一点准备工作，取消一个取消操作，等于是恢复了某个操作，而再加一层，相当于是又取消了某个操作，所以我们可以解决掉 3 操作取消 3 操作的问题，转化成 3 操作取消 $1,2$ 操作的问题（对 3 操作记录 $canc(i)$ 表示取消了哪个操作，并额外记一个”取消“的标记；如果第 x 个操作取消这个 3 操作就相当于 $canc(x) = canc(i)$，不过再记一个“恢复“的标记，三层以上类似处理）。

然后，我们要处理的就是每一层信息的记录了，先考虑用某种数据结构维护每一层的 p_i 值（即有效括号对数），设层按照从里到外编号为 $1,\cdots,k$，每遇到一个 2 操作就增加了一层。

如果取消了一个 1 操作，它处在第 q 层，那么相当于 $p_q = p_q - 1$，因为这一层的一个有效括号消失了。

如果恢复了一个 1 操作，同理，$p_q = p_q + 1$。

如果取消了一个 2 操作，它处在第 q 层，那么相当于 $p_q = p_q - 1$，然后 $p_{q-1} = p_{q-1} + p_q$，而 p_q 不复存在。这意味着原本在 q 层的所有括号对被并入了 $q-1$ 层（因为没有外层括号框定了），-1 是因为本来这个括号对取消了。

如果恢复了一个 2 操作，它原本处在第 q 层，那么记 d_x 表示取消前的 p_{q-1}，那么相当于取消的逆操作，即增加一个 $p_q = p_{q-1} - d_x + 1$，然后 $p_{q-1} = d_x$。

我们并不需要真正删除 p_q，只需要打一个标记，表示它已经和 p_{q-1} 合并起来，成为了一段，于是我们要求的答案变成了与每一段的 p_q 之和有关。

在更新答案时，我们首先去除修改部分原先的答案，修改其实就是单点的加减，同时维护一下当前段的左右端点，然后计算一个区间和来更新答案即可。

可以用线段树维护单点修改，区间求和，左右端点的区间覆盖，时间复杂度为 $O(n \log n)$。

刀妙构造（dmgz）

显然所有 $a_i = i$ 是永远不会移动的不动点。相邻不动点之间如果不是一个错排，那么就是无解。

而如果所有段都是错排，就是有解的，那么只要证明所有错排一定有解就好了。不妨 $\{a_n\}$ 是一个 1 到 n 的错排。

考虑把 1 移动到 a_1。如果能保证后面是一个错排，归纳即可。那么我们把 1 不断往前移动，直到出现了 $a_{x-1} = x$ 这样有问题的局面。（如果一直没问题，就做完了）

这时，如果 $a_{x-2} \neq x-1$，我们只要把 a_{x-1} 和 a_{x-2} 先偷偷换一下，再把 1 往前推就行了。而如果 $a_{x-2} = x-1$，我们就再往前看 a_{x-3} 是否等于 $x-2$，以此类推。

到这里就很清楚了。我们往前找到第一个满足 $a_{x'-1} \neq x'$ 的位置，**若存在**，则把 $(x'-1, x'), (x', x'+1) \ldots (x-1, x)$ 按从左往右的顺序执行交换操作，就可以让 1 顺利往前走了（因为 $a_{x'-1} \neq x$ ）。**若不存在**，则前面一定形如 $2\ 3\ 4\ \ldots\ x-1\ x\ 1$，我们直接把 1 推到底，1 到 x 就排好了，对剩余的归纳即可。

至此，我们在证明了原排列有解的充要条件的同时，构造出了一个合法的操作序列。~~由于出题人才疏学浅，不知如何卡到理论上界 n^2，欢迎大家来交流。~~

Bonus：在 $\mathcal{O}(n \log n)$ 复杂度内求出有解的排列个数，对 998244353 取模。

刀压电线（dydx）

如果我们知道从房子 i 走到房子 j 的最短路，并以此为边权连边，容易想到求出最小生成树，询问时回答路径最大边权。

全部都连边显然太多了，需要只保留有可能出现在最小生成树上的边。注意到如果从 x 到 z 的最优路径经过了 y，那么只要连上 $(x,y)(y,z)$，是不需要连 (x,z) 的。因此，我们考虑跑一个多源 bfs，计算出距离每个空地最近的是哪个房子，即每个房子的管辖范围（显然是一个连通块）。如果两个这样的连通块不接壤，中间必然隔了至少一个 y ，连边是不优的。如果接壤了我们就连边。

注意到在 bfs 的过程中就可以找到所有接壤的地方，即拓展到一个已经更新过的位置，所属最近房子不同。那么这样连出来的边数就是 $\mathcal{O}(nm)$ 的了。总时间复杂度 $\mathcal{O}((nm+p+q)\log p)$。