题解

T1 雅

注意到当确定了这 k 次操作中操作行的次数 x ,那么操作列的次数即为 $k-x$ ，此时行和列交汇的部分因减去 p 而重合所多产生的代价是可以计算的。

因此考虑对于每一个 $0 \leq x \leq k$ ，计算单独对行和列进行 x 次操作后可以对计数器 c 产生的最大贡献，然后枚举 x ，将对行操作 x 次和对列操作 $k-x$ 次的最大贡献加和并取所有贡献的最大值即可。

考虑如何计算对行操作 x 次后的贡献，由于行与行之间的影响独立且操作的影响量相同，因此贪心，使用堆每次取出贡献最大的行操作即可，列类似的处理。时间复杂度 $O(k \log n)$ 。

T2 俗

本题存在多种优秀做法。

考虑使用类似颜色段均摊模型维护区间覆盖，因此可以使用平衡树维护所有加法标记相同的颜色段，每次单点修改可以将颜色段进行分裂，区间标记则可以将区间内的颜色段整体取出并标记，可以使用平衡树做到大常数 $O(n \log n)$ 。

注意到我们只会单点修改序列 a 的值，因此可以直接使用线段树维护加法标记，当涉及到单点修改的时候将标记下放，因此可以做到小常数 $O(n \log n)$ 。

另外一种做法是，考虑离线所有询问并从后往前扫描，可以发现此时问题的模型遍转化成了支持区间加和区间清空、查询区间和，使用线段树即可做到 $O(n \log n)$ 。

T3 共

注意到每次合并形如将两本书合并成一本书，我们将操作的整个过程构建操作二叉树，并按照深度从大到小考虑整棵操作树。

考虑在当前深度层合并两个点，其代价为这两个点对应的子树内的叶子节点的重量之和加上两个点子树最长链较小者对应的磨损值的代价。不妨更进一步的优化模型，我们默认当两个点向上合并出一个点时，我们砍掉磨损值较小的子树，则每次合并形如将两条链合并成一个新子树，代价和两条链中的较小者有关，然后我们砍掉这两条链的较小者，保留较大者继续向上合并。与此同时由于我们会希望磨损值之和尽量小，因此对于当前深度层，若有磨损值分别为 $a \leq b \leq c \leq d$ ，则合并 (a,b) 和 (c,d) 是最优选项，因为此时代价为 $a+c$ 且继续向上合并时两条链的磨损值分别为 $2b+1$ 和 $2d+1$ ，这是最优的选择。

并且我们注意到，由于初始每一本书在操作树上是一个叶子节点，因此考虑给每一本书赋予在操作树上的深度，则重量越大应当深度越小，否则我们可以调整得到更优方案。

因此考虑动态规划，设计 $dp\{t,i\}$ 表示当前已经有 t 个点出现在操作树中，此时处理的当前层还有 i 个点，转移有两种方式，一种是增加一个新的叶子节点对应按照重量从小往大排序后的第 $t+1$ 本书，另一种是将当前层相邻元素两两匹配并向上整体转移一层。时间复杂度 $O(\sum n^2)$，常数足够优秀。

T4 赏

首先由于两个集合没有交，因此我们枚举树上每一个点 x ，假设点 x 为其中一个集合中深度最小的点，并删去点 x 连向父节点的这条边。

然后我们在点 x 父节点所在连通块中枚举一个点 y ，并假设在点 x 在另外一个集合中是与点 y 匹配同构，然后我们在另外一个连通块内以点 y 为根。

设 $dp\{i,j\}$ 表示在点 x 所在连通块内的点 i 和在点 y 所在的连通块内的点 j 匹配同构的最大点集大小，计算 $dp\{i,j\}$ 时我们需要将点 i 的若干儿子节点和点 j 的若干儿子节点进行匹配。

可以发现这个匹配的过程实际上就是一个二分图最大权匹配，使用 KM 算法或者最大费用最大流计算即可。

时间复杂度 $O(n^5)$ 或 $O(n^6)$ ，实际效率非常高。