count

考虑 $x = a^b$ 的情况，其中 a 为质数。容易发现，如果 $b \bmod p = 0$，y 可以取 $a^{\frac{b}{p}}$ 和 a^{bp}，否则只能取 a^{bp}。

对于一般情况，对 x, y 做质因数分解变为 $p_1^{c_1} p_2^{c_2} \cdots p_k^{c_k}$。分开考虑所有质因数，条件等价于 $p \times \min\{c_{x,i}, c_{y,i}\} = \max\{c_{x,i}, c_{y,i}\}$。在这一个质因数上，$y$ 有两种取法当且仅当 $c_{x,i} \bmod p = 0$。答案为所有质因数分开计算取法后的乘积。

注意特判 $p = 1$ 的情况。

时间复杂度 $O(\sqrt{x})$。

seq

做法很多，简单介绍两种。

考虑原序列中相邻和最大的一对数，我们删去其中较大的那个数。这样一定是不劣的，因为只有操作这一对才会让答案变小，删去较大的数也显然是最优的。不断重复该过程即可，可以使用链表和堆进行维护。

考虑二分答案，注意到原序列中最小的数一定会被保留，这样可以从该位置断开视为链上的问题。二分答案后判定时，从前往后能选就选，不能选就尝试替换掉之前选的最后一个数即可。

还可以二分答案后按照数是否 $\le \lfloor \frac{k}{2} \rfloor$ 分开处理……

时间复杂度 $O(n \log n)$。

divide

记 $\operatorname{mex}(l, r)$ 表示序列中第 l 个数到第 r 个数的 mex。假设所有段的 mex 都等于 v，这说明所有段都不含 v 进而说明整个序列不含 v，于是 v 就一定等于整个序列的 mex。于是有 DP 方程：$f_i = \sum_{\operatorname{mex}(j,i)=v} f_{j-1}$。

根据 mex 的性质，满足 $\operatorname{mex}(j, i) = v$ 的所有 j 是一段前缀。维护每种数在 i 之前的第一次出现位置，其中的最小值及之前都满足 $\operatorname{mex}(j, i) = v$。可以使用 `std::set` 维护这些位置，之后使用前缀和优化 DP 即可。当然，这些位置是有单调性的，可以用一个指针维护。

时间复杂度 $O(n)$。

op

特判掉没有赋值操作的情况。

可以发现只有最后一次赋值操作以及之后的加法操作是有意义的，因此可以转化为任选一个赋值操作和若干个加法操作能拼出来的数，将赋值操作作为初始状态，对所有加法操作跑一遍背包 DP 就能求得可以凑出那些数了。可以利用 `std::bitset` 来加速，时间复杂度 $O\left(\frac{np}{w}\right)$。

思考这样一个背包的实质：对于每个权值 v，更新所有的 $i \to (i + v) \bmod p$。可以发现有效更新始终只有 $O(p)$ 次，尝试从此处切入。

尝试对于每个修改，分治出这样的更新。用一个类似线段树的结构，以及一个额外的树状数组维护出区间的 dp 数组哈希值。断环为链后，如果区间 $[l, r]$ 与 $[l + v, r + v]$ 不相同，则递归下去修改。

这样分析势能存在一个问题。对于 dp_i 为 1，但 $dp_{(i+v) \bmod p}$ 为 0 的情况，上述算法并不会更新，但却会递归下去。也就是说势能被利用，但未减少。

考虑本题特殊的性质：背包是在模意义下进行的。在模意义下，所有的 $i \to (i+v) \bmod p$ 必定会形成一个环。环上所有的 $(1,0)$ 都会对应一个 $(0,1)$，因此势能是正确的。视 n,p 同阶，时间复杂度 $O(n\log^2 n)$。