硬币机

对于题目给定的权值 a, b, 由于 b 必须在获得 a 后才能获得。我们可以发现设计 $a, a+b$ 两种权值可以分离“ b 必须在获得 a 后才能获得 ”的这种顺序，同时还能满足原题所需要的条件。

文章查重

测试点 $1 \sim 3$

任意的暴力应该都能过。

测试点 $4 \sim 5$

因为只有字符 0，所以出现次数只与 S 和 T_i 的长度有关，为 $\mid T_i \mid - \mid S \mid +1$，维护 T_i 的长度即可。

测试点 $6 \sim 10$

由于 $k_i = 1$ 考虑建出所有串的 $Trie$ 树。

在 $Trie$ 跑 S 的 KMP，暴力跑 KMP 匹配即可。

测试点 $11 \sim 14$

是否有优秀的 $O(nlogn)$ 做法?

测试点 $15 \sim 20$

和测试点 $6 \sim 10$ 的区别在于 k_i 不一定等于 1。

除了需要处理在后面接上字符串 D，还需要处理两个 T_i 串衔接部分造成的额外贡献。

由于 $\mid T_0 \mid > \mid S_0 \mid$，所以每一个衔接处的后半部分是确定的，所以它的前缀是 S_0 的后缀集合是固定的，可以通过找到 T_0 前缀的最长 S_0 后缀，来求出前半部分是 S_0 的某个前缀时对于答案的贡献。

实时处理 T_i 的最长的是 S_0 前缀的后缀，这个也就是 KMP 维护的信息。

时间复杂度 $O(\mid S \mid + \sum \mid T \mid + \sum \mid D_i \mid)$。

酒杯

算法一

设$f_{i,j}$表示前i层放了j个AC的方案数，转移枚举第$i+1$层放了$k \geqslant 1$个。复杂度$O\left(nm^2\right)$。

算法二

考虑子集反演。设f_S表示至少S对应的层为空的方案数，g_S表示恰好。那么有$f_S = (\text{所有层的大小之和} - \text{在集合}S\text{中的层的大小之和})^m$，我们发现它就等于$((2^n-1)-S)^m$。记$p(i)$为$i$二进制下1的个数，所求即为$g_\emptyset = \sum_S (-1)^{|S|} f_S = \sum_{i=0}^{2^n-1} (-1)^{n-p(i)} i^m$。复杂度$O\left(2^n \log m\right)$。

算法三

对上述式子求解。记$f_{i,j}$为$\sum_{x=0}^{2^i-1}(-1)^{n-p(x)}x^j$的值。讨论第$i$位为$0/1$，为1就乘上$-1$的系数，再根据二项式定理转移。复杂度$O\left(nm^2\right)$

算法四

我们发现每一位的取值只有$0/1$，并且互不影响，于是将逐位转移用倍增优化。复杂度$O\left(m^2\log n\right)$，可以通过。

第K大MEX

希望没有被爆标。

希望暴力没有拿不该拿的分。

测试点1

没有操作。

测试点$2\sim5$

离散化然后暴力。

测试点$6\sim7$

可以莫队，直接 $O(n\sqrt{n}\log n)$ 应该可以过；当然也可以值域分块做到 $O(n\sqrt{n})$。

更有启发性的做法是建立主席树，每个位置维护 $p_{i,v}$ 表示 $1\sim i$ 中 v 最后出现的位置，查询可以主席树上二分第一个 $< l$ 的 $p_{i,v}$。

测试点$8\sim11$

不会，但是莫队 $O(n\sqrt{n})/O(n\sqrt{n}\log n)$ 应该都能过。

测试点$12\sim21$

不会。

测试点$22\sim25$

考虑修改操作：分块，对于每一个块维护出值为 v 的任意一个位置 $pt_{i,v}$，全局维护一个并查集表示同一连通块中的权值相同，即 $a_i=a_{rt_i}$，这样对于一个整块的修改操作即为将 $pt_{i,x}$ 在并查集上的父亲设为 $pt_{i,y}$；对于散块可以直接暴力；这样修改操作就可以做到 $O(B+\frac{n}{B})$（忽略并查集）。

在 $6\sim7$ 中我们有一个 $O(\log n)$ 查询 mex 的方法，考虑扩展这个方法：首先，由于修改主席树是不可能的，而且难以扩展到 $k>1$ 的情况，所以考虑更暴力的维护方式。

注意到 mex 信息以及 $p_{i,v}$ 是难以合并的，所以舍弃线段树等需要信息合并的数据结构，考虑分块。

类似在线莫队，维护 $O(\frac{n}{B})$ 个关键点，每个关键点对应前缀信息 $p_{i,v}$，每次查询将 r 前面最后一个关键点到 r 之间的信息再加入 $p_{i,v}$，这样可以维护 $p_{r,v}$。

对于 $k>1$ 的 kthmex 查询，如果使用普通的二分答案，要解决的问题是 $p_{r,v}<l, v\le mid$ 二维偏序，基本不可能维护，所以选择更加契合分块结构的值域分块来做查询，即维护 $[kT+1,kT+T]$ 的 $p_{r,v}$ 的前/后缀信息，支持单点修改，可以用树状数组等结构维护，然后查询时时间复杂度为 $O(\frac{V}{T}\log n+T)$，离散化后为 $O(\frac{n}{T}\log n+T)$。

总结一下，修改操作使用分块维护 a_i 信息，同时处理出每一个关键点前缀对应 $p_{i,v}$ 的变化情况，只会变化 $p_{i,x}$ 与 $p_{i,y}$ 且是可维护的，用树状数组修改 $p_{i,x}/p_{i,y}$ 的值，时间复杂度为 $O(\frac{n}{B}\log n+B)$ 。

查询操作，找到 r 前最后一个关键点 p，如果 $p>l$，则直接暴力将 $[l,r]$ 中的点加入值域分块，这一部分的时间复杂度为 $O(B+T+\frac{n}{T})$；否则 $p\le l$，直接使用 p 所维护的 $p_{p,v}$ 信息，将 $[p+1,r]$ 中的权值设置为一定出现，然后遍历值域分块，对于每一个块查询 $\ge l$ 的 $p_{p,v}$ 数量，注意要加入 $[p+1,r]$ 的情况，这样可以得到答案所对应的值域块；遍历这个小值域块，再查询 $p_{p,v}$ 与 l 的大小关系，最终得到答案，这一部分时间复杂度为 $O(B+\frac{n}{T}\log n+T)$。

取 $B=T=\sqrt{n\log n}$，总的时间复杂度为 $O(n\sqrt{n\log n})$。

关于空间：如果要做到严格 $O(n\sqrt{\frac{n}{\log n}})$ 的空间复杂度，需要将树状数组改为平衡树，这样可以保证 c 个点的空间消耗为 $O(c)$，但是这样实现常数会非常大，而且查询的 $\frac{n}{T}\log n$ 与修改的 $\frac{n}{B}\log n$ ，都比另一部分查询 T 与修改 B 慢很多很多，出题人实现的这个做法跑的很慢，$n=m=10^5$，修改操作基本会修改的数据下大概要跑 $6s$（本机），洛谷 $5s$ 很勉强，详见 std1。

本来平衡树实现版本是以前的 std，但是由于跑的太慢了，所以当时把数据调小到了 8×10^4，仍然需要跑 $2\sim 3s$。

但是其实查询的 T 以及修改的 B 部分跑的很快，将 T/B 设置到 $30\sim 60$ 左右会变快，而且此时可以用树状数组替换平衡树维护 $p_{r,v}$，空间复杂度虽然是 $O(n\sqrt{n\log n})$ 但是实际空间消耗比原来小，因为树状数组只有一个数组，而且这个数组可以只开 `unsigned short`，详见 std2，所以可以跑 $n=m=10^5$ 的数据，本机 $2.5s$，洛谷上 $1.5s$，所以时限是 $3.5s$。