1 试题

1.1 问题描述

对于大小为 n 的可重集合 S，S 中每个元素都是 $[0,n]$ 中的整数。定义一次操作为：取出任一非空子集 X，将 X 从 S 中删去后，向 S 中加入 $\operatorname{mex} X$。

对于可重集合 S，令 $f(S)$ 表示，对 S 做任意多次操作后，S 中不同元素的个数的最大值。

现给出了 S 中的一个子集 T，并需要对于每个 k，求出满足 $T \subseteq S$，$0 \in S$ 且 $f(S) = k$ 的 S 的个数。答案需要对 $10^9 + 7$ 取模。

1.2 输入格式

第一行两个整数 n 和 $|T|$。

第二行 $|T|$ 个整数表示 T 中的元素。

1.3 输出格式

输出一行 n 个整数，分别表示对于每个 $k = 1, 2, \ldots, n$ 对应的答案。

1.4 数据范围与约定

对于所有数据，满足 $1 \le n \le 200$，$0 \le |T| \le n$，T 中元素皆为整数且 $\in [0,n]$。

Subtask 1 (5 points) $|T| = n$

Subtask 2 (10 points) $n \le 12$

Subtask 3 (20 points) $n \le 30$

Subtask 4 (20 points) $n \le 70$

Subtask 5 (20 points) $n \le 120$

Subtask 6 (25 points) $n \le 200$

2 解题过程

2.1 算法一

首先探讨在给定 S 的情况下，如何求出 $f(S)$。这将帮助我们通过 Subtask 1。我们令 S' 表示经过操作后得到的最优的可能的可重集合。若两个集合的不同的元素种数相同，那么我们取 0 尽可能多的集合。

那么为了得到 S'，显然我们在一次操作中，一定要么取 $X = \{x\}$ 来获得一个 0，要么取 $X = \{0, 1, \ldots, x-1\}$ 来获得一个 x。并且，S' 中不会存在出现次数 > 1 的非零元素（否则这些元素可以全部转化成 0）。

引理 2.1 *不存在 $x < y$ 满足 $y \notin S$ 且 $x \notin S'$，$y \in S'$.*

证明 2.1 由于 $x, y \notin S$，那么 y 必定是在一次操作中得到。那么设最后一次获得 y 的操作中取到的元素是 $X = \{0, 1, \ldots, y-1\}$，包含了 x，那么将这次操作转化为 $X = \{0, 1, \ldots, x-1\}$ 一定不劣。

引理 2.2 *若 $x \in S$，则 $x \in S'$.*

证明 2.2 对于 $x \neq 0$，若 $x \notin S'$，则根据上述引理可知不存在 $> x$ 的新元素。也就是说，x 被 S 中删去只有可能是用来获得一个 0。删去 x 以多获得一个 0 显然不会使得 S' 中不同元素变多；就算不变，由于 $0 \in S$，所以会因为要消耗这个 0 造新元素，导致 0 的数量也不会变多。对于 $x = 0$，证明也是类似且显然的。

据此，我们可以推得 S' 的形式。

定理 2.1 *一定存在正整数 m，使得 $S' = \{0, 1, \ldots, m-1\} \cup (S \cap [m+1, n])$，其中只有 0 会重复出现。*

于是，我们可以考虑枚举这样的 m，考虑如下的贪心策略来判断 S' 是否合法：

1. 将 $> m$ 的所有 S 中元素仅保留一份，其余全部转化为 0。
2. 从 $x = m$ 开始往前处理。令 e 表示目前 0 的数量，d 表示需要 $[0, x-1]$ 各几份。初始时 $d = 1$。
3. 设 x 的出现次数为 c，则若 $c > d$，那么 d 不需要变化，e 可以增加 $c - d$；否则 d 需要变化为 $2d - c$。
4. 直到 $x = 0$ 时，判断 e 和 d 的大小即可判断 S' 是否合法。

暴力枚举 S 的形态后用上述方法求出 $f(S)$ 即可通过 Subtask 1.

2.2 算法二

首先我们可以考虑做一步转化，转化为求解 $f(S) \geq k$ 的方案。

这样做带来的一个好处是：对于 S，判断 $f(S) \geq k$，只需要找到唯一的一个 m，使得答案为 k，再判断能否进行操作即可。

然后，对于每个 m，我们需要知道的关于 $S \cap [m+1, n]$ 的信息只有，其中有多少个元素，以及其中有多少种不同的元素。这个可以通过 DP 来解决。具体而言，$g_{i,j,k}$ 表示 $\geq i$ 的元素中，S 中有 j 个且其中不同元素有 k 种的方案数。

然后考虑再 DP 出 $f_{i,j,e,d}$ 表示扫到 $i+1$ 时，e, d 分别如状态所计，那么满足扫到最后能合法且 $|S \cap [0, m-1]| = j$ 的 $S \cap [0, m-1]$ 的数量。

设 h_k 为 $f(S) \geq k$ 的方案数。那么统计答案的时候，则只需要在 m 处将 $f_{m-1,j,x,1} \times g_{m+1,n-j,y}$ 贡献给 h_{m+y} 即可。

当然只是这么统计是不完全的，因为我们忽略了原本 S 的种类数 $= x$ 时对 $y < x$ 时的 h_y 的贡献。但处理这种贡献是平凡的，直接将 $g_{0,n,x}$ 贡献给 h_y 即可。

g 的转移是简单的。g 的状态数是 $O(n^3)$ 的，且转移需要枚举 i 元素的出现个数。暴力转移的复杂度是 $O(n^4)$，但是容易发现可以直接使用前缀和优化将复杂度降至 $O(n^3)$。

f 会少许繁琐一些。首先，f 的状态数实际上是 $O(n^3 \log n)$ 的。$e \leq n$，$j \leq n$，而对于 d 我们一定有 $d \times i \leq e + j$，所以状态数为 $O(n^3 \log n)$ 的。

考虑 f 的转移。我们枚举 i 元素的出现个数 c，那么根据算法一，我们需要讨论 c 和 d 的大小。$c \leq d$ 时，会从 $f_{i-1,j-c,e,2d-c}$ 转移；而 $c > d$ 时，会从 $f_{i-1,j-c,e+d-c,c}$ 转移。暴力处理复杂度是 $O(n^4 \log n)$ 的。但容易发现对于两种情况，我们也都可以使用前缀和优化，使得单次转移复杂度变为 $O(1)$。

综上，算法二的复杂度是 $O(n^3 \log n)$，可以通过本道题目。

若缺少了上述的优化，那么可以以更高的复杂度获得一些部分分，取决于实现。

3 参考资料

感谢金点，陈凯丰同学在本题出题过程中，与我关于本题的讨论。