【封印】题解

havzriu

2025.2

题意

简要题意

给定长度为 n，值域为 m 的正整数序列 A。你可以做若干次操作：
- 选择一个严格前缀最大值 A_s。删除这段前缀，如果 A_s 不为 1，将 $A_s - 1$ 插入序列的尾端。

问在若干操作后，你可以得到多少个不同序列。（包括空序列）

T 组数据。

$T \leq 10, 1 \leq n, m \leq 3000, 1 \leq A_i \leq m$

$4s, 512MB$

n,m ≤ 10

使用搜索，搜出所有可能的符合条件的序列。
可以证明，答案不超过 $n(m+1)2^n \leq 112640$。
搜索时，只需枚举 s，检查是否合法，然后递归。
朴素实现，总复杂度不超过 $O(Tn^4(m+1)2^n \log n)$。
实际完全无法跑满，可以轻松通过。
预计得分 8。

性质分析

我们把一次操作看成将一个前缀合并成一段，最后一个元素就是代表元，称其他元素被删去了。

考虑枚举原序列的一些元素，假设它们做代表元。现在考虑可行的必要条件：

对每个被删去的元素 A_x：

- 如果它后面有未被删除的元素，找到第一个，记为 A_y。则 $A_y > A_x$。
- 否则，找到第一个未被删除的元素，记为 A_z，则 $A_z - 1 > A_x$

性质分析

接下来，对一种分段方案，考虑它如何生成最终序列。

结论 1

我们只需每次取出第一个代表元 A_r，作为选择的 s。
这样即可不漏的生成所有对应可能的最终序列。

简要证明：
显然，如果每次选取的 s 都不应 r 之后。
然而为了尽快的删去应该被删去的元素，选择 r 作为 s 显然不劣。

性质分析

接下来考虑生成方案。
无非是选择前面的一些段减 $(k+1)$，后面的一些段减 k，最后把 ≤ 0 的段剔除。
最后把第一个减 k 的段转到最前面。称之关键段。
下面考虑 $k \geq 1$ 的情况。

性质分析

考虑删除的影响。
如果一段被删成了负数，那么总可以认为它被它后面的段删去。我们可以忽略这种情况。
于是我们可以做这样的限制：要求不能被删成负数，且如果某一段被删成 0，要求其最初与后面的一段值相等。
这也就是说只有接缝处才允许有 0。
例如
$2\ 5\ 4\ 4|4\ 4\ 3\ 5 \to 2\ 0\ 0|1\ 2$
(k=3)
显然可以合并到另一种情况：
$2\ 5|4\ 4\ 4\ 4\ 3\ 5 \to 2|1\ 2$
(k=3)

性质分析

容易在原序列中找到某些不会被其他元素删去的元素，称为极大元素。
显然，只要变成 0 的不是极大元素，就总能做好合并。
所以可以认为所有非极大元素都不能被删至 0。
再来考虑极大元素的情形。既然极大元素都被删成 0 了，那剩下的就也只剩 0/1 了。也可以方便的处理。
记极大元素的数目为 s，这部分的贡献无非是 $s-1$。
于是就不会产生重复了。下面我们考虑 $k=0$ 的情形，注意我们要规避 $k\geq 1$ 时已经计算过的部分。

性质分析

先考虑一种 corner，也就是存在极大元素是 1。这样原序列由 1/2 组成。
分类讨论最终答案是否有 2。如果最后有 2，那就只有一段前缀被操作了。
如果没有 2，那么最终序列全是 1。答案只和原序列 2 的数目相关。

性质分析

$k = 0$ 时有可能会出现 1 被删成 0 的情况。但是这里不能和之前一样做合并了。

考虑直接枚举关键段的位置，仍然只需考虑分段序列，这样唯一可能出问题的就是，删去一些原本的 1 后会导致重复。

不过被删去的 1 肯定原本紧靠着关键段。只要关键段不是尾端，就保证了不能像前面一样被合并。

关键段是尾端的情况比较特殊。如果尾部不是 1 自然没有问题。只需考虑尾部是 1 怎么处理。

性质分析

$k=0$ 时有可能会出现 1 被删成 0 的情况。但是这里不能和之前一样做合并了。
考虑直接枚举关键段的位置，仍然只需考虑分段序列，这样唯一可能出问题的就是，删去一些原本的 1 后会导致重复。
不过被删去的 1 肯定原本紧靠着关键段。只要关键段不是尾端，就保证了不能像前面一样被合并。
关键段是尾端的情况比较特殊。如果尾部不是 1 自然没有问题。只需考虑尾部是 1 怎么处理。
会导致重复无非是因为分段序列长成这样：

$$2\ 2\ 2\ [...]\ 1\ 1\ 1$$

而且我们让最后的 1 全都被删了。
既然我们在前面保留了 2，那么根据之前的贪心，这样显然是不对的。（即我们应该选 1 来替代 2）
特判掉结尾为 1 且开头为 2 的情况就好了。
这种情况下，直接规定这些 1 不能被删成 0。

算法设计

先考虑 $k=0$ 的情况如何快速计算答案：
某个点能作为关键段的必要条件是这个后缀要全选。
依据前面的判定条件做 dp 即可。

算法设计

对 $k \geq 1$ 的情况：
只需 dp 本质不同的分段序列，并统计贡献。
我们找到第一个全局 max 作为开头。接下来每次转移到当前的前缀 max。注意最后一段的限制。（不能比第一段的结尾大）
结构无非是前面一段限制是 k，后面一段限制是 $k+1$。因此 dp 的时候记录下目前可选的 k 即可。
由于总状态数是 $O(n^2)$，每次需要枚举转移到哪，因此每次 dp 的复杂度为 $O(n^3)$。
预计得分 68。

算法设计

对 $k \geq 1$ 的情况：
只需 dp 本质不同的分段序列，并统计贡献。
我们找到第一个全局 max 作为开头。接下来每次转移到当前的前缀 max。注意最后一段的限制。（不能比第一段的结尾大）
结构无非是前面一段限制是 k，后面一段限制是 $k+1$。因此 dp 的时候记录下目前可选的 k 即可。
由于总状态数是 $O(n^2)$，每次需要枚举转移到哪，因此每次 dp 的复杂度为 $O(n^3)$。
预计得分 68。
在外层枚举 k，这样状态转为一维。
转移时依然是可以转移到单调栈上任意的位置，这样就能做到线性。因此总复杂度 $O(n^2)$。
预计得分 100。

特殊性质 B

选好被保留的元素，答案只和 (第一个选择的元素) 和 (选择的总元素相关)。
枚举第一个选择的元素 A_t，枚举共选了 l 个元素，贡献为：

$$\binom{n-t-1}{l-2}((A_t-1)\times l+1)$$

这里注意最后一个元素必选，且 $t \neq n$。
总复杂度 $O(n^2)$，预期得分 12。

$n \leq 18, m \leq 70$

直接 $O(2^n)$ 枚举哪些元素被保留了，扫一遍数出贡献。
总复杂度 $O(2^n \text{poly}(n))$，预期得分 12。

特殊性质 A

这一部分特殊性质并没有针对性的解法。但是可以节约大量的分类讨论，也能加速 dp。

例如计数时，至多只有两种转移，简化了转移流程。

End.