APIO 2026 中国区《上升》试题讲解

APIO 2026 中国区命题组

2026 年 5 月 9 日

算法 1

状压 dp，设 $f(S)$ 表示，填了前 $|S|$ 个位置，出现数的集合为 S，的方案数之和。

算法 1

状压 dp，设 $f(S)$ 表示，填了前 $|S|$ 个位置，出现数的集合为 S，的方案数之和。

转移时直接枚举后一个元素，时间复杂度 $O(n2^n)$。可以通过测试点 $1 \sim 4$，期望得分 20 分。

算法 2

考虑测试点 $5 \sim 7$。

算法 2

考虑测试点 $5 \sim 7$。

记 $f(i,j)$ 表示确定好前 i 个数之间的大小关系，第 i 个数的 rank 是 j，的权值积之和。

算法 2

考虑测试点 $5 \sim 7$。

记 $f(i,j)$ 表示确定好前 i 个数之间的大小关系，第 i 个数的 rank 是 j，的权值积之和。

转移我们可以暴力一点，直接枚举第 $i+1$ 个数的 rank，这样就是 $O(n^3)$ 的，期望得分 15 分。

结合算法 1，总共可以获得 35 分。

算法 3.1

我们发现这个 S 在 dp 中刻画起来不是特别方便，这是由于同时存在上升和下降的要求。

算法 3.1

我们发现这个 S 在 dp 中刻画起来不是特别方便，这是由于同时存在上升和下降的要求。

所以我们需要做一个转化，记 $w'_i = w_i - 1$，这样

$$\prod_{i \in S} w_i = \prod_{i \in S} (w'_i + 1)$$

这个的组合意义就是，从 S 中再选择一个集合 T，T 中所有位置的权值之积的和。这样转换后，我们就不需要处理下降的限制了。

算法 3.1

假定我们此时已经选择好这个 T，这时原序列会被划分成若干个（不一定极长的）上升段。接着如果一个上升段存在某个位置是已经被确定的，这个限制分裂成可以变成“左边到这个位置”和“这个位置到右边”都是上升的。

算法 3.1

假定我们此时已经选择好这个 T，这时原序列会被划分成若干个（不一定极长的）上升段。接着如果一个上升段存在某个位置是已经被确定的，这个限制分裂成可以变成“左边到这个位置”和“这个位置到右边”都是上升的。

所以现在需要考虑的有四类：

- 一个限制中不存在任何位置被确定。
- 一个限制的首尾都被确定。
- 一个限制只有尾被确定。
- 一个限制只有首被确定。

算法 3.1

我们首先不考虑第一类，看看只有后三类的时候怎么做。

这时假设我们从左到右考虑到以 i 结尾的上升段，那么只有最后一类限制会出现 $> premax := \max_{1 \le j \le i} p_j$ 的数。

算法 3.1

我们首先不考虑第一类，看看只有后三类的时候怎么做。

这时假设我们从左到右考虑到以 i 结尾的上升段，那么只有最后一类限制会出现 $> premax := \max_{1 \le j \le i} p_j$ 的数。

于是我们记 $f(i, j)$ 表示：从左往右填到以 i 结尾的上升段，$[1, i]$ 中有 j 个位置的数 $> premax$，我们尚未知道这些数具体填什么，但我们已经知道他们之间的大小关系，所有情况权值积的和。

算法 3.1

考虑主动转移，从 $f(i,j)$ 贡献到 $f(k,j')$。从左往右不断加整个限制，每当我们加入一个限制，首先需要考虑 $(premax, premax')$ 中哪些数出现在前面。

算法 3.1

考虑主动转移，从 $f(i,j)$ 贡献到 $f(k,j')$。从左往右不断加整个限制，每当我们加入一个限制，首先需要考虑 $(premax, premax')$ 中哪些数出现在前面。

设 $\Delta = premax' - premax - 1$。记 $g(j)$ 表示我们加入当前的限制后，存在 j 个大于 $premax$。枚举 d，这就是从 $g(j)$ 转移到 $g'(j-d)$，系数为 $\binom{\Delta}{d}$。

算法 3.1

然后考虑新增的这一段限制，设 $zeros$ 表示一段未填数的个数。

那么分类讨论每种情况的转移：

算法 3.1

然后考虑新增的这一段限制，设 $zeros$ 表示一段未填数的个数。

那么分类讨论每种情况的转移：

- 如果是第二类，则从 $g(j)$ 转移到 $g'(j)$，系数为 $\binom{\Delta-d}{zeros}$。
- 如果是第三类，则从 $g(j)$ 转移到 $g'(j)$，系数为 $\binom{premax'-1-i+j}{zeros}$。
- 如果是第四类，从 $g(j)$ 转移到 $g'(j+zeros)$，系数为 $\binom{j+zeros}{j}$

算法 3.1

然后考虑一下现在的复杂度，状态数为 n^2。转移时我们要枚举 k，考虑到最后加段的时候是 $O(1)$ 的，瓶颈在于 $premax$ 的增加。

考虑到我们是一个一个加限制，d 的上界是 Δ，而 $\sum \Delta = O(n)$。所以迄今为止都是 $O(n^3)$ 的。

算法 3.1

然后考虑一下现在的复杂度，状态数为 n^2。转移时我们要枚举 k，考虑到最后加段的时候是 $O(1)$ 的，瓶颈在于 $premax$ 的增加。

考虑到我们是一个一个加限制，d 的上界是 Δ，而 $\sum \Delta = O(n)$。所以迄今为止都是 $O(n^3)$ 的。

问题在于第一类转移，这时我们需要额外枚举一个 l，表示在这一段有多少 $> premax$ 的。则转移从 $g(j)$ 到 $g(j+l)$，系数是 $\binom{premax-1-i}{zeros-l} \times \binom{j+l}{l}$。

这是 $O(n^4)$ 的，结合前面的部分分期望得分 65 分。

算法 3.2

优化也是简单的。枚举 $t := zeros - l$，我们只需保证 $t + l \leq \max zeros$。

算法 3.2

优化也是简单的。枚举 $t := zeros - l$，我们只需保证 $t + l \leq \max zeros$。

观察转移里每一项，要么和 l 相关，要么和 t 相关。不存在与二者同时相关的项。

算法 3.2

优化也是简单的。枚举 $t := zeros - l$，我们只需保证 $t + l \leq \max zeros$。

观察转移里每一项，要么和 l 相关，要么和 t 相关。不存在与二者同时相关的项。

所以可以直接分步转移，时间复杂度 $O(n^3)$，期望得分 100 分。

算法 4

上面是使用一次容斥的做法，接下来我来介绍一下验题人的使用零次容斥的做法。

算法 4

上面是使用一次容斥的做法，接下来我来介绍一下验题人的使用零次容斥的做法。

考虑直接 dp。由于已经填好的值是单调递增的，相邻的值之间会具有相似的结构，所以可以对每一对相邻的值之间的部分进行 dp。

这里不妨设 $p_0 = 0, p_{n+1} = n + 1$，则我们依次考虑所以极长的未填值的区间 (l, r)，即 p_l, p_r 已经固定，$\forall l < i < r, p_i$ 还未确定。

算法 4

由于我们只关心相对大小关系而非 p_i 的具体值，将所有值分为 $[0, p_l), (p_l, p_r), (p_r, n+1]$ 这三个部分，只需要记录每个部分填了多少个数。

算法 4

由于我们只关心相对大小关系而非 p_i 的具体值，将所有值分为 $[0, p_l), (p_l, p_r), (p_r, n+1]$ 这三个部分，只需要记录每个部分填了多少个数。

进一步注意到，在 l 之前的部分没有任何 $> p_l$ 的元素已经被确定，所以可以只确定 $[0, p_l)$ 部分的值，对于 $(p_l, n+1]$ 这个部分，我们只确定元素之间的相对排名关系，而暂时不赋具体的值。

这样可以记 $f_{i,j}$ 表示考虑到第 i 个极长连续段，前面有 j 个元素的值 $< p_l$ 且已经被确定。

算法 4

转移时，可以对 (l, r) 内的元素做最朴素的 dp（$dp_{i,j}$ 表示考虑到 i，最后一个元素的排名为 j 的答案）。需要额外记录 $[0, p_l)$ 和 (p_l, p_r) 这两个部分分别填了多少个元素。

算法 4

转移时，可以对 (l, r) 内的元素做最朴素的 dp（$dp_{i,j}$ 表示考虑到 i，最后一个元素的排名为 j 的答案）。需要额外记录 $[0, p_l)$ 和 (p_l, p_r) 这两个部分分别填了多少个元素。

在区间内转移结束后，再枚举 l 之前的部分有多少个元素填在了 (p_l, p_r) 这个区间内。这些元素的相对排名已经被确定，但它们与当前这一段的元素之间没有关系，所以可以自由组合，乘一个组合数的系数即可完成转移。

算法 4

转移时，可以对 (l, r) 内的元素做最朴素的 dp（$dp_{i,j}$ 表示考虑到 i，最后一个元素的排名为 j 的答案）。需要额外记录 $[0, p_l)$ 和 (p_l, p_r) 这两个部分分别填了多少个元素。

在区间内转移结束后，再枚举 l 之前的部分有多少个元素填在了 (p_l, p_r) 这个区间内。这些元素的相对排名已经被确定，但它们与当前这一段的元素之间没有关系，所以可以自由组合，乘一个组合数的系数即可完成转移。

直接实现可以得到一个多项式复杂度的做法，视实现不同可以做到 $O(n^6) \sim O(n^4)$ 不等。注意到最后一个数是 p_r，所以它的排名结合 p_l 的排名可以直接推断出三个区间内各有多少个元素，因此只需要在记录最后一个元素排名的同时记录 p_l 的排名即可。通过前缀和优化转移可以做到段内转移 $O(n^3)$，段之间合并 $O(n^4)$。期望得分为 $50 \sim 65$（根据不同的实现）。

算法 4

这个做法的瓶颈在于合并一个连续段的结果与之前部分的信息。考虑将这一段的信息融合进之前的信息进行 dp。在段开始的时候，可以看作 p_l 下面有若干个空位，在填这些空位的时候必须直接填入具体的值而不能只确定排名。

在填 $> p_l$ 的部分时，只关心相对排名而不填具体的值。这样在最后一个元素 $< p_l$ 时，我们记录它的值（有多少空位），在最后一个元素 $> p_l$ 时，我们记录它的排名。

算法 4

这个做法的瓶颈在于合并一个连续段的结果与之前部分的信息。考虑将这一段的信息融合进之前的信息进行 dp。在段开始的时候，可以看作 p_l 下面有若干个空位，在填这些空位的时候必须直接填入具体的值而不能只确定排名。

在填 $> p_l$ 的部分时，只关心相对排名而不填具体的值。这样在最后一个元素 $< p_l$ 时，我们记录它的值（有多少空位），在最后一个元素 $> p_l$ 时，我们记录它的排名。

即令 $f_{i,j}$ 表示 p_l 下面还有 i 个空位没有填，最后一个元素 $< p_l$ 且下面还有 j 个空位没有填。令 $g_{i,j}$ 表示 p_l 下面还有 i 个空位没有填，最后一个元素与 p_l 之间已经填了 j 个元素。

算法 4

这部分的转移容易使用前缀和优化做到 $O(n^3)$。考虑怎么在 p_r 处更新整个前缀的信息。

算法 4

这部分的转移容易使用前缀和优化做到 $O(n^3)$。考虑怎么在 p_r 处更新整个前缀的信息。

由于 $p_r > p_l$，只有 g 才是有用的。

对于 $g_{i,j}$，枚举 l 之前有多少个在 (p_l, p_r) 之间的元素，与当前的 j 个元素用组合数合并，然后加上 i 就是现在 p_r 之下空位的数量，而这恰好就是整段 dp 需要记录的信息。

所以枚举一个值即可，复杂度 $O(n^3)$。期望得分 100 分。

可能存在的算法 5

出题人有一个使用两次容斥的做法。

可能存在的算法 5

出题人有一个使用两次容斥的做法。

但由于他最开始是 $O(n^4)$ 的，而且声称自己懒得想怎么任意模数。

所以这里不讲这个做法了。