IOI2021 集训队作业试题交流

西南大学附属中学校 蒋凌宇

2020 年 11 月 27 日

目录

1 试题《Yet Another Permutation Problem》

来源：原创

1.1 题目描述

给定一个长度为 n 的排列 $[1, 2, \ldots, n]$，你可以进行以下操作：

- 选择一个数，将其取出然后放到排列的开头或末尾。

对每个 $k = 0, 1, \ldots, n-1$，求出进行至多 k 次操作可能得到的排列个数。由于这些数可能非常大，你只需要回答它们除以 m 的余数即可。

1.2 输入

一行两个整数 n, m ($1 \le n \le 1\,000$, $10^8 \le m \le 10^9 + 9$, m 是素数)。

1.3 输出

n 行，第 i 行一个整数表示 $k = i - 1$ 时的答案。

1.4 子任务

Subtask 1 (10 points): $n \le 10$。
Subtask 2 (10 points): $n \le 18$。
Subtask 3 (10 points): $n \le 50$。
Subtask 4 (30 points): $n \le 300$。
Subtask 5 (40 points): 无额外限制。

1.5 时空限制

1 s, 1024 MiB。

2 解题过程

2.1 暴力算法

2.1.1 算法一

模拟题目中的过程，用 BFS 求出每个排列最少需要的操作次数，然后统计每种操作次数的排列个数即可。时空复杂度为 $O(n! \cdot poly(n))$，其中 $poly(n)$ 是关于 n 的多项式，视实现而定，通常为 $O(n^2)$ 或 $O(n^3)$。

这个算法可能无法在规定的时间限制内求出 $n = 10$ 的答案，但由于答案都不超过 $10! = 3\,628\,800$，选手可以在本地求出所有可能的输入的答案后，将其写入代码。

期望得分：10。

2.2 初步分析

性质 1. *假设排列 p 的最长上升子段长度为 l，则得到 p 需要的最少操作次数为 $n - l$。*

证明. 考虑从 p 开始，每次将排列的第一个或最后一个元素移动到任意位置，直到排列变成 $[1, 2, \ldots, n]$。这个过程是原过程的逆，因此它们的最少步数相等。考虑 p 中没有操作过的元素，它们应当是递增的。并且每次操作都只能操作第一个或最后一个元素，因此这些元素应该是 p 的一个连续上升子段。因此最少步数不会少于 $n - l$。设 p 的最长上升子段为 S，每次取 p 第一个或最后一个元素 (不能在 S 中)，将其插入 S 中的对应位置，使得 S 仍然有序，就能使得排列的最长上升子段长度增加 1。因此，从 $[1, 2 \ldots, n]$ 得到 p 的最少步数即为 $n - l$。

□

运用性质 1，不超过 k 步能得到的排列数就等于最长上升子段至少为 $n - k$ 的排列数。

2.2.1 算法二

注意到排列的最长上升子段长度只和所有相邻元素的大小关系有关。枚举 2^{n-1} 种大小关系，则需要对每种大小关系求出对应的排列个数。可以通过简单的 DP 在 $O(n^3)$ 时间内求出，也可以用前缀和优化至 $O(n^2)$，总复

杂度为 $O(2^n n^3)$ 或 $O(2^n n^2)$。另一种方法是枚举不满足的 $p_i < p_{i+1}$ 的关系集合容斥，时间复杂度为 $O(3^n)$；枚举 $<$ 和 $>$ 中较少的一种进行容斥，可以做到 $O(\sum_{i=0}^{n} \binom{n}{i} 2^{\min\{i,n-i\}}) = O(\frac{2\frac{3}{2}^n}{\sqrt{n}})$。

同样的，这些算法并不都能在规定时限内求出 $n = 18$ 的答案，但选手可以在本地求出答案后写入代码。

期望得分：20。

2.2.2 算法三

考虑计算所有上升子段长度都 $< n - k$ 的排列数，用 $n!$ 减去这个数就是对应 k 的答案。

设 $dp(n, x, l)$ 表示长度为 n，最后一个数为 x，结尾的上升子段长度为 l 的排列数。转移时枚举下一个数的相对大小即可。对每个 k 的复杂度为 $O(n^4)$，可以用前缀和优化至 $O(n^3)$。总复杂度 $O(n^5)$ 或 $O(n^4)$。

期望得分：30。

2.3 最终算法

不妨直接考虑排列的结构，即若干个极长上升子段的拼接，每一个长度为 l 的段的权值为 $[l < n - k]$，排列的权值为所有极长上升子段权值的乘积。但是由于存在“极长”的限制，难以用生成函数的运算来描述。考虑去掉极长的限制，设一个上升子段的 EGF 为 $G(x)$，则若干个上升子段的拼接的 EGF 即为 $\frac{1}{1-G(x)}$。容易发现，设 $H(x)$ 为 $G(x)$ 对应的 OGF，则每个极长上升子段的权值实际上被计算为了 $[x^l]\frac{1}{1-H(x)}$。换句话说，如果我们希望得到的极长上升子段权值的 OGF 为 $F(x)$，只需令 $H(x) = 1 - \frac{1}{F(x)}$，$G(x)$ 为 $H(x)$ 对应的 EGF，则所有排列的权值和的 EGF 即为 $\frac{1}{1-G(x)}$。

2.3.1 算法四

极长上升子段权值的 OGF 为 $F(x)\sum_{i=0}^{n-k-1} x^i = \frac{1-x^{n-k}}{1-x}$，令 $H(x) = 1 - \frac{1}{F(x)} = \frac{x-x^{n-k}}{1-x^{n-k}}$，$G(x)$ 为 $H(x)$ 对应的 EGF，则上升子段长度都不超过 $n-k$ 的排列的 EGF 即为 $\frac{1}{1-G(x)}$。在 $O(n^2)$ 的时间内可以求出一个 k 的答案，总时间复杂度 $O(n^3)$。

值得一提的是，可以通过对长度为 $n-k$ 的子段容斥的方式推导出同样的公式，细节留作练习。

期望得分：60。

2.3.2 算法五

用 FFT 优化算法四中的求逆。时间复杂度 $O(n^2 \log n)$。

期望得分：100。

2.3.3 算法六

注意到 $G(x)$ 系数非零的项只有 $O(\frac{n}{n-k})$ 项，因此朴素求逆的复杂度为 $O(\frac{n^2}{n-k})$，总复杂度为 $O(\sum_{k=0}^{n-1} \frac{n^2}{n-k}) = O(n^2 \log n)$。

期望得分：100。

3 总结

本题是一道组合计数问题，考察了容斥、动态规划、生成函数、FFT 等多个知识点。选手需要首先注意到题目的性质，然后可以通过传统容斥推出 DP 方程，也存在利用生成函数的便捷推导方式。在最终的实现上，选手可以选择用 FFT 来优化生成函数运算，如果进一步分析还能得到同样复杂度且不需要 FFT 的巧妙做法，算得上是一道不错的题目。