《春天，在积雪下结一成形，抽枝发芽》命题报告

马耀华

2020 年 12 月

目　录

1 题目简介

1.1 题目来源

原创

1.2 题目大意

分别求长度为 $1 \sim n$ 的不存在非平凡（长度在 $[2, n-1]$ 之间）连续段的排列数目 mod 998 244 353 的值。

一个序列是连续的，当且仅当把这个序列的值由小到大排序后，第 i 个数的值是第 1 个数的值加上 i−1。

1.3 限制与约定

对于 $type = 0$ 的数据，你仅需要输出长度为 n 的答案；对于 $type = 1$ 的数据，你需要分别输出长度为 $1 \sim n$ 的答案。

对于所有数据，$type \in \{0, 1\}$，$1 \leq n \leq 10^5$。

子任务编号	$n \leq$	$type \in$	分值
1	8	$\{0, 1\}$	10
2	1 000	$\{0, 1\}$	20
3	10^5	$\{0\}$	30
4	10^5	$\{0, 1\}$	40

时间限制：2s

空间限制：512MB

2 部分分算法

2.1 算法一

对于 $n \leq 8$ 的数据，我们可以分别枚举长度 $len = 1 \sim n$ 的全排列并暴力验证是否不存在非平凡的连续段，进而求出答案。

验证过程最暴力的实现是枚举 $\mathcal{O}(len^2)$ 个区间，单次 $\mathcal{O}(len)$ 验证，因而单次验证时间复杂度为 $\mathcal{O}(len^3)$。

总时间复杂度为 $\mathcal{O}(n! \cdot n^3)$，可以通过子任务 1，期望得分 10 分。

2.2 算法二

对于 $n \leq 1\,000$ 的数据，我们希望得到一个 $\mathcal{O}(n^2)$ 的算法。考虑使用动态规划。

令 f_n 表示长度为 n 的**不存在非平凡连续段**的排列个数，则答案即为 $f_1 \sim f_n$。

直接求解 f_n 比较困难，我们先考虑求解另一个相似的问题：令 g_n 表示长度为 n，且**所有非平凡连续段必定包含最后一个数**的排列个数 [1]。

定理 2.2.1 $g_1 = 1, g_2 = 2, g_n = (n-2)g_{n-1} + \sum_{i=3}^{n-2}(i-2)g_i g_{n-i+1}, \forall n \geq 3$。

证明 对于任意长度为 $n \geq 3$ 的排列 a，考虑 $b = a^{-1}$，即 $b_{a_i} = i$。显然 a 与 b 之间一一对应，于是 g_n 也可以理解为长度为 n 且**所有非平凡连续段包含最大值**的排列个数，对称地，g_n 也可以理解为长度为 n 且**所有非平凡连续段包含最小值**的排列个数。

考虑在不包含最小值的长度为 $n-1$ 的 $[2, n]$ 的排列 P 中插入 1，来得到长度为 n 的 $[1, n]$ 的且所有非平凡连续段包含最大值 n 的排列 Q，分类讨论：

1. P 中所有非平凡连续段包含 n：这样的 P 有 g_{n-1} 个，插入 1 只需不与 2 相邻即可，共有 $n-2$ 个插入位置，方案数为 $(n-2)g_{n-1}$。
2. P 中存在不包含 n 的非平凡连续段：为使 Q 中所有非平凡连续段均包含 n，那么 P 中不包含 n 的极长非平凡连续段必须唯一，且不包含 2。枚举该极长非平凡连续段 S 的长度 $i \in [2, n-3]$，则 1 会插入在 S 内部，组成一个长度为 $i+1$ 的子序列 S'。我们将 S' 保持大小顺序对应到 $[1, i+1]$ 的排列 L，那么插入后 S' 不存在任何非平凡连续段，等价于 L 中所有非平凡连续段包含最小值 1，于

是有 g_{i+1} 种不同的 L。在 P 中将 S 缩起来后，我们可以得到一个长度为 $n-i$ 的排列 R，R 中任何非平凡连续段都包含最大值，于是有 g_{n-i} 种不同的 R。且给定一个 R，我们可以将它某个非最小值又非最大值的位置（有 $n-i-2$ 个）在保持大小顺序的情况下换成一个 S，得到一个 Q，且这样的替换是一一对应，于是总方案数为 $\sum_{i=2}^{n-3}(n-i-2)g_{i+1}g_{n-i}=\sum_{i=3}^{n-2}(i-2)g_ig_{n-i+1}$。

加上初值，我们有:

$$g_1=1,g_2=2,g_n=(n-2)g_{n-1}+\sum_{i=3}^{n-2}(i-2)g_ig_{n-i+1},\forall n\geq 3$$

. □

这样，我们容易在 $\mathcal{O}(n^2)$ 的时间复杂度内求出 $g_1\sim g_n$。接下来考虑如何利用 $g_1\sim g_n$ 求出 $f_1\sim f_n$。

定理 2.2.2 $f_1=1,f_2=2,f_n=g_n+2h_{n-1}-\sum_{i=2}^{n-1}g_if_{n-i+1},\forall n\geq 3$。其中 $h_1=1,h_2=1,h_n=g_{n-1}-h_{n-1},\forall n\geq 3$。

证明 显然所有不存在非平凡连续段的排列 a 都满足所有非平凡连续段必定包含最后一个数 a_n，于是我们考虑从 g_n 中扣除不满足条件的排列来得到 f_n。

我们考虑枚举不满足条件的排列中最长的包含最后一个数的连续段长度 $i\in[2,n-1]$，则该连续段有 g_i 种方案，将这一部分缩起来后得到的新排列是一个长度为 $n-i+1$ 且不存在任何非平凡连续段的排列，有 f_{n-i+1} 种方案。

看起来递推式为 $f_n=g_n-\sum_{i=2}^{n-1}g_if_{n-i+1}$，$\forall n\geq 3$?

很容易发现这样的推理是有漏洞的：若我们直接把最长的包含最后一个数的长为 i 的连续段与任意一个长为 $n-i+1$ 且不存在任何非平凡连续段的排列合并起来，得到的新排列未必满足它所有的非平凡连续段都包含最后一个数 a_n。也就是说，我们可能从 g_n 中扣的太多了，有一些不属于 g_n 中的排列也被我们减去了。

考虑什么情况下会发生上面的情况：如果出现这种情况，则任何一个这样的非平凡连续段 $[l,r]$ 一定满足 $l\in[1,n-i]$，$r\in[n-i+1,n-1]$。

若 $i<n-1$，我们分类讨论 l：

1. $l>1$：此时 $[l,r]$，$[n-i+1,n]$ 均是连续段，且 $[l,r]\cap[n-i+1,n]\neq\varnothing$。于是 $[l,r]\cup[n-i+1,n]=[l,n]$ 也是非平凡连续段，且它比 $[n-i+1,n]$ 更长，矛盾。
2. $l=1$：同样有 $[l,r]$，$[n-i+1,n]$ 均是连续段，且 $[l,r]\cap[n-i+1,n]\neq\varnothing$。于是 $[l,r]\setminus[n-i+1,n]=[l,n-i]$ 也是连续段，且 $[l,n-i]$ 长度至少为 2，也是非平凡连续段，于是缩起来后的新排列仍有非平凡连续段，矛盾。

综上，当 $i < n-1$ 时，不可能出现上述情况，也即我们减去的排列个数正确。

若 $i = n-1$，$n-i+1 = 2$，我们分类讨论 r：

1. $r > 2$：此时 $[l, r]$，$[n-i+1, n]$ 均是连续段，且 $[l, r] \cap [n-i+1, n] \neq \varnothing$。于是 $[l, r] \cap [n-i+1, n] = [2, r]$ 也是非平凡连续段，且 $r < n$ 告诉我们这是一个不包含 a_n 的在 $[n-i+1, n]$ 中的非平凡连续段，仍然矛盾。
2. $r = 2$：此时 $[l, r] = [1, 2]$。又由于 $[n-i+1, n] = [2, n]$ 是连续段，于是 $[n-i+1, n]$ 对应的权值区间是 $[1, n-1]$ 或 $[2, n]$。故 $[a_1, a_2] = [1, 2]$ 或 $[n, n-1]$。容易发现这种情况确实不在 g_n 中被统计。

综上，我们漏掉的情况就是 $[a_1, a_2] = [1, 2]$ 或 $[n, n-1]$ 的，两种情况对称，我们只考虑前者。这其实等价于我们枚举的长度为 $n-1$ 的连续段是一个以最小值开头的任意非平凡连续段都包含 a_n 的排列。

令这样的排列个数为 h_{n-1}，加上初值，我们就可以得到正确的递推式:

$$f_1 = 1, f_2 = 2, f_n = g_n + 2h_{n-1} - \sum_{i=2}^{n-1} g_i f_{n-i+1}, \forall n \geq 3$$

现在还有一个问题是找出 h_n 的递推式，这是容易的：当 $n \geq 3$ 时，我们考虑删去一个这样的长为 n 的排列 P 的开头 1，剩余的排列是一个长度为 $n-1$ 的 $[2, n]$ 的任意非平凡连续段都包含最后一个数的排列 Q，总共有 g_{n-1} 个，且按我们上面的讨论，使 P 不合法（存在不包含最后一个数的非平凡连续段）的 Q 必定是以 2 开头的，这部分显然有 h_{n-1} 个，于是 $h_n = g_{n-1} - h_{n-1}$。

加上初值，我们有：

$$h_1 = 1, h_2 = 1, h_n = g_{n-1} - h_{n-1}, \forall n \geq 3$$

. □

于是，我们在求出 $g_1 \sim g_n$ 后，也容易在 $\mathcal{O}(n)$ 的时间复杂度内先递推出 $h_1 \sim h_n$，再在 $\mathcal{O}(n^2)$ 的时间复杂度内递推出 $f_1 \sim f_n$。

总时间复杂度为 $\mathcal{O}(n^2)$，可以通过子任务 1, 2，期望得分 30 分。

2.3 算法三

考虑使用描述排列连续段的数据结构：析合树 [2] 来解决问题。这里为了方便，我们认为叶节点既不是析点也不是合点。

当 $n = 1$ 时答案为 1，$n = 2$ 时答案为 2，$n = 3$ 时答案为 0。

当 $n \geq 4$ 时，可以发现一个不存在任何非平凡连续段的排列，恰好与一棵根节点为析点，且有 n 个叶节点作为它的儿子的析合树一一对应。因此我们的问题可以从排列计数变为对析合树的计数了。

根据析合树的定义，容易得到下述引理：

引理 2.3.1 如果一个点是某个析点的儿子，它自己没有任何限制，即它自己可以是任意的析点、合点或叶节点；如果一个点是某个合点的儿子，而它父亲合点的儿子对应连续段单调上升/下降，它自己是析点或叶节点的话仍然没有限制，但是如果是合点，自己儿子对应连续段只能单调下降/上升。

与算法二中一样，我们令 f_n 表示长度为 n 的**不存在非平凡连续段**的排列个数。再令 g_n 表示叶结点个数为 n 的**根节点是合点，且根节点儿子对应连续段单调上升**的析合树个数，对称地，g_n 也表示叶结点个数为 n 的**根节点是合点，且根节点儿子对应连续段单调下降**的析合树个数。

令 $F(x) = \sum_{n\geq 4} f_n x^n$（$f_n$ 对应的 OGF），$G(x) = \sum_{n\geq 2} g_n x^n$（$g_n$ 对应的 OGF），$H(x) = \sum_{n\geq 1} n! x^n$（所有长度为 n 的排列（叶结点个数为 n 的析合树）的个数对应的 OGF）。

引理 2.3.2 $G(x) = \sum_{n\geq 2} (H(x) - G(x))^n$。

证明 考虑每棵根节点是合点，且根节点儿子对应连续段单调上升的析合树 T。

T 的根节点每个儿子所在的子树都是一棵析合树，且根节点要么是析点或叶节点，要么是儿子对应连续段单调下降的合点。这类析合树对应的 OGF 显然就是所有析合树中除去根节点儿子对应连续段单调上升的合点对应的析合树，即 $H(x) - G(x)$。

同时 T 的根节点儿子对应连续段单调上升，根据 OGF 的运算，枚举 T 的根节点儿子个数，即有 $G(x) = \sum_{n\geq 2} (H(x) - G(x))^n$。 □

这样，我们有：

$$\begin{aligned} G(x) &= \sum_{n\geq 2} (H(x) - G(x))^n \\ &= \frac{(H(x) - G(x))^2}{1 - (H(x) - G(x))} \\ \Rightarrow G(x) &= \frac{H^2(x)}{H(x) + 1} \end{aligned}$$

引理 2.3.3 $F(H(x)) = H(x) - x - 2G(x)$。

证明 令叶节点个数为 n 的根节点为析点的析合树个数为 w_n，令 $W(x)=\sum_{n\geq 4} w_n x^n$（$w_n$ 对应的 OGF）。这样的析合树是所有析合树中除去了单个叶节点或根节点为合点的析合树所剩的，那么我们显然有 $W(x)=H(x)-x-2G(x)$。

考虑这样的某棵析合树 T，按定义 T 的根节点至少有 4 个儿子，那么 T 的根节点的儿子对应连续段缩起来后，对应一个不存在任何非平凡连续段且长度 ≥ 4 的排列，且它每个儿子所在的子树是一棵没有限制的析合树。

于是，根据 OGF 的运算，枚举 T 的根节点的儿子个数，即有 $W(x)=\sum_{n\geq 4} f_n \cdot H(x)^n = F(H(x))$，也即 $F(H(x))=H(x)-x-2G(x)$。 □

注意到这里 $H(x)$ 没有常数项且一次项为 1，于是 $H(x)$ 有复合逆 $I(x)=H^{-1}(x)$，满足 $H(I(x))=I(H(x))=x$。

那么：

$$\begin{aligned} F(H(x)) &= H(x)-x-2G(x) \\ \Rightarrow F(x) &= H(I(x))-I(x)-2G(I(x)) \\ &= x-I(x)-\frac{2H^2(I(x))}{H(I(x))+1} \\ &= x-\frac{2x^2}{x+1}-I(x) \end{aligned}$$

比较对应项系数，可以得到 $f_n=[x^n]F(x)=2\cdot(-1)^n-[x^n]I(x)$，$\forall n\geq 4$。

这样，我们发现求出 $f_1\sim f_n$ 就等价于求出 $I(x) \mod x^{n+1}$ 了。

对于 $type=0$ 的数据，只需要求出 f_n，也即求出 $[x^n]I(x)$，我们可以利用拉格朗日反演快速完成：

$$[x^n]I(x)=\frac{1}{n}[x^{n-1}](\frac{x}{H(x)})^n$$

这里只需要使用 $\mod x^n$ 意义下的多项式求逆和多项式求幂，均可以在 $\mathcal{O}(n\log n)$ 的时间复杂度内完成。

总时间复杂度为 $\mathcal{O}(n\log n)$，可以通过子任务 3，期望得分 30 分。

2.4 算法四

考虑在算法三的基础上稍加改动来通过子任务 $1,2$。

一个简单的做法是分别对 $len=1\sim n$ 跑一遍算法三，这样总时间复杂度为 $\mathcal{O}(n^2\log n)$，常数较大。

下面我们均在 $\mod x^{n+1}$ 意义下讨论。

考虑直接求出 $I(x)$，这样就能同时求出 $f_1 \sim f_n$ 了。回到定义 $I(x) = H^{-1}(x)$，这相当于求出一个多项式复合逆的前 $n+1$ 项系数。

一个比较简便的方法是利用分块思想：取 $M = \lceil \sqrt{n} \rceil$。先算出 $J(x) = \frac{x}{H(x)}$，然后我们依次算出 $J^0(x), J^1(x), \ldots, J^{M-1}(x)$ 和 $J^M(x), J^{2M}(x), \ldots, J^{M^2}(x)$。这部分可以依次在上一项基础上乘一个 $J(x)$ 或 $J^M(x)$ 得到下一项，总时间复杂度为 $\mathcal{O}(n\sqrt{n}\log n)$。然后我们要求出 $[x^k]I(x)$，等价于求出 $[x^{k-1}]J^k(x)$。令 $k = pM + q$（$0 \le p \le M, 0 \le q < M$），那么我们即要算 $[x^{k-1}]J^{pM}(x) \cdot J^q(x)$，利用前面结果单次可以 $\mathcal{O}(n)$ 完成，于是这部分总时间复杂度为 $\mathcal{O}(n^2)$。

总时间复杂度为 $\mathcal{O}(n^2)$，可以通过子任务 $1, 2$，结合算法三，期望得分 60 分。

值得一提的是，这里即使我们使用了一些复杂度较为优秀的求多项式复合逆算法，由于常数过大可能也难以通过子任务 4 得到满分。

3 标准算法

3.1 算法五

考虑在算法二的基础上进行优化。

我们回顾一下算法二的两个递推式：

$$g_1 = 1, g_2 = 2, g_n = (n-2)g_{n-1} + \sum_{i=3}^{n-2}(i-2)g_i g_{n-i+1}, \forall n \geq 3$$

$$f_1 = 1, f_2 = 2, f_n = g_n + 2h_{n-1} - \sum_{i=2}^{n-1} g_i f_{n-i+1}, \forall n \geq 3$$

我们可以发现一个这是一个半在线卷积的形式：f_n, g_n 只依赖于更低的项的值，且转移是一个卷积的形式。

使用分治 FFT 算法求解半在线卷积的话，时间复杂度为 $\mathcal{O}(n \log^2 n)$，可以通过子任务 $1 \sim 4$，期望得分 100 分。

值得一提的是，如果使用 [3] 中的算法，可以做到 $\mathcal{O}(n(\log n)^{1+\epsilon})$ 的时间复杂度。

3.2 算法六

考虑在算法四的基础上进行优化。

可以发现，这里需要求复合逆的多项式比较特殊，是 $H(x) = \sum_{n\geq 1} n! x^n$。

根据 [4] 中的结果，我们知道 $H(x)$ 满足下面 ODE：

$$\begin{aligned}
&H(x) = H'(x) \cdot x^2 + H(x) \cdot x + x \\
\Rightarrow &H(I(x)) = \frac{1}{I'(x)}(\frac{d}{dx}H(I(x)))I^2(x) + H(I(x))I(x) + I(x) \\
\Rightarrow &x = \frac{1}{I'(x)}I^2(x) + I(x) \cdot x + I(x) \\
\Rightarrow &I^2(x) - I'(x) \cdot x + (x+1)I(x)I'(x) = 0
\end{aligned}$$

这是一个 $I(x)$ 满足的 ODE。令 $c_n = [x^n]I(x)$，比较对应项系数，我们可以得到一个 c_n 的递推式：

$$c_0 = 0, c_1 = 1, c_n = -\sum_{i=1}^{n-1}(i+1)c_i c_{n-i} - \sum_{i=2}^{n-1} i c_i c_{n-i+1}, \forall n \geq 2$$

可以发现这同样也是一个半在线卷积的形式，使用分治 FFT 算法求解半在线卷积的话，时间复杂度为 $\mathcal{O}(n\log^2 n)$，可以通过子任务 $1\sim4$，期望得分 100 分。

使用 [3] 中的算法，同样可以做到 $\mathcal{O}(n(\log n)^{1+\epsilon})$ 的时间复杂度。

4 总结

满足本题要求的排列称为 simple permutation[5]。本文从两个角度考虑了 simple permutation 的计数问题：通过传统的动态规划角度，得到一个答案的递推式并使用半在线卷积优化；通过析合树来描述排列，利用生成函数推导转化为求多项式的复合逆问题，并挖掘问题的特殊性，利用 ODE 得到答案的递推式并使用半在线卷积优化。两个算法的出发点不同，但最后都转化为了对答案递推式做半在线卷积，体现了计数问题的殊途同归。

本题一开始是作者为 Comet OJ 比赛[1]命制的，当时的版本是本题的子任务 3。后面经过研究，发现可以同时得到 $1 \sim n$ 的答案，于是命制了现在的版本。

本题综合考察了选手的动态规划，数据结构，生成函数等知识，有较高的思维难度，但代码实现不算复杂，是一道作者认为命制不错的题目。

此外，在算法六中，我们得到了 $I(x)$ 满足的 ODE，但这里比较特殊，如果直接利用 [6] 中算法求解，会遇到 $e^{\frac{1}{x}}$ 这种项。限于作者水平，无法给出 $\mathcal{O}(n \log n)$ 算法，欢迎有想法的同学与作者交流。

[1] Comet OJ-Contest 6 F

参考文献

[1] anonymous author.OEIS A090753.

[2] OI-wiki contributors. 析合树.

[3] Joris van der Hoeven.Faster relaxed multiplication.ISSAC 14,Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation.

[4] Richard P. Stanley.*Enumerative Combinatorics Volume 2*.

[5] anonymous author.OEIS A111111.

[6] Brent R. P. , Kung H. T. .Fast Algorithms for Manipulating Formal Power Series.*Journal of the ACM*.