欧拉数问题选讲

北京大学 陈昕阳

2026.2.11

欧拉数的定义

一个长度为 n 的排列定义为长度为 n 的序列，满足 $1 \sim n$ 各出现一次。

对于一个长度为 n 的排列 p，$\forall 1 \le i \le n$，记 p_i 是 p 的第 i 个元素。

一个长度为 n 的排列 p 的上升数 $\operatorname{asc}(p)$ 定义为有多少下标 $1 \le i < n$ 满足 $p_i < p_{i+1}$，即 $\operatorname{asc}(p) = \sum_{i=1}^{n-1} [p_i < p_{i+1}]$。

欧拉数 $\left\langle {n \atop k} \right\rangle$ 定义为有多少长度为 n 的排列有恰好 k 个上升。

今天讲什么？

今天会从四个角度介绍欧拉数的四种求法，但这样也至多讲 1h 就讲完了。

所以还会考虑一些与之相关的问题。总的来说，今天讲的内容大致是：

- 排列计数问题。
- 关心（简单来说是）排列中一些（一般是相邻的）项的相对大小关系。
- **今天考虑的所有问题答案均对某个固定的大质数取模。**

  例题的题解写得都比较简略。

基本想法

现在考虑如何计算 $\left\langle {n \atop k} \right\rangle$，即求有多少 n 阶排列 p 满足 $\mathrm{asc}(p) = k$。

$\mathrm{asc}(p)$ 的定义是 $\sum_{i=1}^{n-1} [p_i < p_{i+1}]$，只关心相对大小关系。这意味着排列中的最大值 n 和最小值 1 天然就是特殊的：假设 $p_k = n$，那么只要 $k > 1$，$p_{k-1} < p_k$ 就必然成立，对于为 1 的位置也一样。

如果要尝试通过 DP 来计算 $\left\langle {n \atop k} \right\rangle$，应该尝试通过某种方式将问题分解为子问题。

对于一个长度为 n 的排列 p，设 $p_k = n$，不妨考虑把 p_k 这一项删掉得到一个长度为 $n-1$ 的排列 q，此时 $\mathrm{asc}(q)$ 相比 $\mathrm{asc}(p)$ 的变化。

讨论变化

变化是非常局部的，区别只有：

- 如果 $k > 1$，本来 p 中 p_{k-1} 和 p_k 相邻且 $p_{k-1} < p_k = n$，从而这个上升消失了。
- 如果 $k < n$，本来 p 中 p_k 和 p_{k+1} 相邻，但 $n = p_k > p_{k+1}$，无事发生。
- 如果 $1 < k < n$，现在 q 中 p_{k-1} 和 p_{k+1} 两项相邻，但不知道 p_{k-1} 和 p_{k+1} 的大小关系。

进一步分类讨论：

- 如果 $k = 1$，$\mathrm{asc}(q) = \mathrm{asc}(p)$。
- 如果 $k = n$，$\mathrm{asc}(q) = \mathrm{asc}(p) - 1$。
- 否则，如果 $p_{k-1} < p_{k+1}$，则 $\mathrm{asc}(q) = \mathrm{asc}(p)$（消失了一个又新增了一个）；如果 $p_{k-1} > p_{k+1}$，则 $\mathrm{asc}(q) = \mathrm{asc}(p) - 1$。

反过来想想

站在 p 的角度看，看上去事情还是不太简单。

但可以反过来想想，对于一个确定的 q，有 n 种不同的 p 可以生成它：因为 p 中删去最大值 n 得到 q，那么已知 q 的时候可以任意选 n 个“空位”之一插入 n。

n 个“空位”指的是插到两个数之间有 $n-2$ 个，插到最前面最后面各一个。

反过来想想

之前的讨论是：

- 如果 $k=1$，$\operatorname{asc}(q)=\operatorname{asc}(p)$。
- 如果 $k=n$，$\operatorname{asc}(q)=\operatorname{asc}(p)-1$。
- 否则，如果 $p_{k-1}<p_{k+1}$，则 $\operatorname{asc}(q)=\operatorname{asc}(p)$（消失了一个又新增了一个）；如果 $p_{k-1}>p_{k+1}$，则 $\operatorname{asc}(q)=\operatorname{asc}(p)-1$。

这相当于说插到最前面得到的 p 满足 $\operatorname{asc}(p)=\operatorname{asc}(q)$，插到最后面得到的 p 满足 $\operatorname{asc}(p)=\operatorname{asc}(q)+1$。而插到一个 $q_i<q_{i+1}$ 的 q_i 和 q_{i+1} 之间，$\operatorname{asc}(q)=\operatorname{asc}(p)$；插到 $q_i>q_{i+1}$ 之间，$\operatorname{asc}(q)=\operatorname{asc}(p)+1$。

q 总能生成 $\operatorname{asc}(q)+1$ 个 $\operatorname{asc}(p)=\operatorname{asc}(q)$ 的不同的 p，以及 $n-1-\operatorname{asc}(q)$ 个 $\operatorname{asc}(p)=\operatorname{asc}(q)+1$ 的不同的 p。

最终解法

q 总能生成 $\operatorname{asc}(q)+1$ 个 $\operatorname{asc}(p)=\operatorname{asc}(q)$ 的不同的 p，以及 $n-1-\operatorname{asc}(q)$ 个 $\operatorname{asc}(p)=\operatorname{asc}(q)+1$ 的不同的 p。

这表明有递推关系：$\left\langle {n \atop k} \right\rangle = \left\langle {n-1 \atop k} \right\rangle (k+1) + \left\langle {n-1 \atop k-1} \right\rangle (n-k)$。

原因是：对于 k 个上升的长度为 n 的排列 p，其要么由 $\operatorname{asc}(q)=k$ 的排列 q 生成而来，而每个 $\operatorname{asc}(q)=k$ 的排列 q 可以生成 $k+1$ 个不同的 $\operatorname{asc}(p)=k$ 的排列 p。

要么由 $\operatorname{asc}(q)=k-1$ 的排列 q 生成而来，每个可以生成 $n-1-(k-1)=n-k$ 个不同的排列 p。

为了避免边界条件，我们认为这个递推关系对 $n \geq 2$ 成立，而 $k=0$ 的 $\left\langle {n \atop k} \right\rangle$ 没有后面那一项。

显然，$\left\langle {1 \atop 0} \right\rangle = 1$，而对于 $k>0$ 的 $\left\langle {n \atop k} \right\rangle = 0$。

根据这个递推关系，可以 $\Theta(n^2)$ 算出所有 $0 \leq k < n$ 的 $\left\langle {n \atop k} \right\rangle$。

试试看！

现在你已经完全学会欧拉数了，来试试看这道题目吧！

定义 (n, α) 阶排列为，全体长度为 αn 的序列 a 满足以下两条限制：

- $\forall 1 \le k \le n$，a 中恰好有 α 个 k。
- 对于 $1 \le i < k < j \le \alpha n$，如果 $a_i = a_j$，那么 $a_k \ge a_i$。

给定 α, n, m, n_0 以及一个 (n_0, α) 阶排列 p，计算有多少 (n, α) 阶排列 q 既包含 p 作为子序列，又满足 $\mathrm{asc}(q) = m$。

$1 \le n, n_0\alpha \le 2 \times 10^3$。

source：CTS2023 Day1 另一个欧拉数问题。

提示：题目并不难。（因为这个 source 标记的并不完全正确）

题解

对一个 (n,α) 阶排列 a 并选定 m，只保留 a 中 $\leq m$ 的数后，得到一个 (m,α) 阶排列。

向一个 (n,α) 阶排列中插入 α 个 $n+1$ 希望得到一个 $(n+1,\alpha)$ 阶排列，充要的插入方式是选中 $n\alpha+1$ 个空隙之一把 α 个 $n+1$ 全部插进去。

设出 $dp_{i,j}$ 表示有多少 (i,α) 阶排列 q 包含 p 作为子序列，同时 $\operatorname{asc}(q)=j$。转移分向 q 的上升位置插入 α 个 $i+1$，或者向一个非上升位置插入，特殊考虑插入到开头或结尾。

$\Theta(n^2)$。

试试看！

给定 n, k 以及长度为 n 的序列 a，求有多少长度为 n 的排列 p 满足：设长度为 n 的序列 b 满足 $b_i = a_{p_i}$，则 $\mathrm{asc}(b) = k$。

$1 \le n \le 5 \times 10^3$。

source：ABC267G。

我们稍后会讨论这题如何做到更快。

题解

相当于问 a 的所有重排，有多少有 k 个上升。下面视作值一样就是相同的重排，最后答案乘上每种颜色出现次数阶乘之积即可。

仍然从小到大插入值，考虑对于一个长度为 m 有 k 个上升（且其中值都 $\leq v$）的序列，向里面插入 q 个 $v+1$ 的所有方案，得到的序列上升数的分布。

一组插入方案就是将 q 分为 $m+1$ 个自然数之和，表示给 $m+1$ 个空隙中的每一个插入多少个 v。

向中间 $m-1-k$ 个非上升空隙或末尾插入至少一个 v，每一个这样的空隙会导致上升数增加 1。

相当于说所有将 q 分为 $m+1$ 个自然数之和的方式，问前 $m-k$ 个中有 l 个非零的方案数，以这么多不同的方案数，转移到长为 $m+q$ 的序列有 $k+l$ 个上升。

这个转移系数是 $\binom{m-k}{l}\binom{q+k}{l+k}$。

题解

于是 $dp_{v,k}$ 设 a 中值 $\leq v$ 的数的所有重排，有多少上升为 k。枚举 v, k 后转移需要再枚举 l。

从组合意义或者 $\binom{q+k}{l+k}$ 这一项系数都可看出，只有 $l \leq q$ 才有意义，而 q 是 a 中值 $v+1$ 的出现次数，而所有颜色出现次数总和为 n，故也是 $\Theta(n^2)$。

回顾

这一节介绍了从插入极值（并将插入之后的部分平移下标）的角度考虑的第一种插入 DP，可以注意到在欧拉数问题中从“由子问题添加一个元素到达更大的问题”角度考虑会比“从原问题删去一个元素递归到子问题”的角度考虑更加便利。

疑惑

但一个疑惑是，这种方法的能力限度是什么？为什么本节考虑的几个问题都可以这样解决？

一个解释是：本节考虑的问题所有下标的地位都是均等的，这为平移下标提供了可能。反映在 $\mathrm{asc}(p)$ 的表达式上就是其平等地考虑了所有 $1 \le i < n$，只关心是否 $[p_i < p_{i+1}]$。

很容易举出下标地位并非“均等”的例子：

考虑额外输入一个长度为 $n-1$ 的 01 序列 w，把 $\mathrm{asc}(p)$ 的定义改成 $\sum_{i=1}^{n-1} w_i[p_i < p_{i+1}]$，前面的做法会遇到一个本质的困难：如果还是删去最大的元素，会遇到一次下标的整体平移，而移动前后大量位置对应的 w_i 都变化了。

不过这个问题也可以用插入 DP 解决，不过方式略有区别。

重新计算欧拉数

现在我们还是想计算 $\left\langle {n \atop k} \right\rangle$，但是每次插入最大值的方式太难理解了，能不能使用每次删去末尾的元素的方式来递归到子问题呢？

首先的问题是如果删掉的末尾元素是 x，前 $n-1$ 个元素就是包含 $1 \sim x-1$ 和 $x+1 \sim n$ 各一个的长度为 $n-1$ 的序列了，并没有严格地递归到“长为 $n-1$ 的排列”的子问题。

但注意到前 $n-1$ 个元素也只关心相对顺序，所以可以作离散化，即将 $x+1 \sim n$ 分别视为它们减 1。

反过来从添加元素的角度考虑，就是对于给定的长为 $n-1$ 的排列 q，其有 n 种扩展成为一个长度为 n 排列 p 的方法：任选一个 $1 \le x \le n$，然后构造 p 是 $\forall 1 \le i < n, p_i = q_i + [q_i \ge x]$，而 $p_n = x$。

不幸的是：这样得到的 $\operatorname{asc}(p) = \operatorname{asc}(q) + [q_{n-1} < x]$，这意味着这样考虑只知道 $\operatorname{asc}(q)$ 是不够的。

重新计算欧拉数

所幸如果将状态定义为：$dp_{i,j,k}$ 表示有多少长度为 i 的排列有 j 个上升，而末尾元素是 k，转移确实是封闭的。

具体来说，转移是枚举 $1 \le x \le i+1$ 然后将 $dp_{i+1,j+[k<x],x}$ 加上 $dp_{i,j,k}$。

转移边界是 $dp_{1,0,1} = 1$，而 $\left\langle {n \atop k} \right\rangle = \sum_{i=1}^{n} dp_{n,k,i}$。

然而这个方法直接实现是 $\Theta(n^4)$，前缀和优化转移后是 $\Theta(n^3)$ 的，反而更慢了。

不过它还是稍微有点用的，考虑以下两道例题：

试试看！

给定 n 和长度为 $n-1$ 的 01 串 s，计算有多少长度为 n 的排列 p 满足：

- $\forall 1 \le i < n,\ [p_i < p_{i+1}] = s_i$。

$1 \le n \le 2000$。

source：LOJ575 不等关系弱化版。

我们稍后会讨论这题如何做到更快。

题解

设 $dp_{i,j}$ 表示有多少长度为 i 的排列 p，使得前 i 项相邻的相对大小关系符合 s 的描述，而 $p_i = j$。

转移所有从前 i 项扩展到前 $i+1$ 项的扩展方案，和前面一样相当于说枚举 p_{i+1} 的取值即可，同样是前缀和优化 $\Theta(n^2)$。

试试看！

对于给定的 k 以及系数序列 $w_0 \sim w_{2^{k-1}-1}$，定义一个 $n \geq k$ 阶排列 p 的权值 $val(p) = \prod_{i=1}^{n-k+1} w_{f(p_i,p_{i+1}\dots p_{i+k-1})}$，其中 $f(a_1, a_2 \dots a_k) = \sum_{i=1}^{k-1} 2^{i-1}[a_i < a_{i+1}]$。

给定 n，计算所有 n 阶排列的权值和。

$1 \leq n \leq 2 \times 10^3, 2 \leq k \leq 4$。

source：QOJ9651 又一个欧拉数问题弱化版。

我们稍后会讨论这题如何做到更快。

题解

现在需要知道连续 k 项之间，每相邻两项的相对大小关系。

直接状压设 $dp_{i,j,S}$ 表示长度为 i 的排列，$p_i = j$，而末尾 $k-1$ 项之间相邻的相对大小关系是二进制数 S 即可。

转移和前面一样，前缀和优化一下是 $\Theta(n^2 2^k)$。

回顾

这一节介绍了从插入最后一个元素（并将值域作平移）的角度考虑的第二种插入 DP，它提供了一种处理一些下标地位不均等的问题的方式，缺点是要记录最后一个元素的大小，使得其有时（其实我认为几乎总）不能达到最优复杂度。

一道经典题

事实上还有第三种插入 DP 的方式，也可以用它来计算欧拉数，然而用这种方式计算欧拉数会让其看上去没有任何优点，所以我们介绍这种方法的经典例题：

- 定义长度为 n 的排列 p 是 zig-zag 的，当且仅当 $\forall 1 < i < n$，$(p_{i+1} - p_i)(p_i - p_{i-1}) < 0$。
- 如果对于 $1 < i < n$，$p_i > \max(p_{i-1}, p_{i+1})$ 时称 p_i 是峰；而 $p_i < \min(p_{i-1}, p_{i+1})$ 时称 p_i 是谷的话。这个条件相当于说所有 $1 < i < n$ 的 p_i 必须是峰或谷。
- 给定 n, s, t，计算有多少长为 n 且是 zig-zag 的排列 p 满足 $p_1 = s, p_n = t$。
- $n \leq 2000$。
- source：QOJ5532 CEOI 2016 kangaroo。

这道题也可以做到更快，但不在今天的讨论范围内。更快的做法简而言之：代数推导又赢了。

困境

尝试使用第二种插入 DP，其可以描述相邻元素的大小关系，也可以满足 $p_n = t$ 的限制，但是 p_1 不知道怎么办，事实上直接设 $dp_{i,s,t,0/1}$ 表示长度为 i 的 zig-zag 排列，且首元素是 s 尾元素是 t，倒数两个元素的关系是上升/下降可以完成转移，不幸的是这是 $\Theta(n^3)$ 的。

使用第一种插入 DP 的困难之处在于，如果在 q_i 和 q_{i+1} 之间插入 n，会导致 q_i 和 q_{i+1} 接下来都必须作为谷（只要不是边界元素），但 q_{i-1}, q_i 以及 q_{i+1}, q_{i+2} 的相对大小关系是未知的。

换句话说，第一种插入 DP 遇到的本质困难是：q_i 和 q_{i+1} 与其邻居的大小关系发生了变化。

所以如果邻居关系一经确定就不再更改，就可以解决这个问题。

分析

设想你已经确定了排列 p，现在一开始所有位置都是未被点亮的。按照 $v = 1 \sim n$ 的顺序，你找到唯一的 v 的出现位置 i 即 $p_i = v$，然后点亮位置 i，条件相当于说什么？

- $i = 1$ 必须在第 s 轮被点亮，$i = n$ 必须在第 t 轮被点亮。
- 对于 $1 < i < n$，其被点亮时要么 $i - 1$ 和 $i + 1$ 都被点亮，要么都没被点亮。

如果考虑极长的被点亮的连续段，这相当于说 $1 < i < n$ 的 i 被点亮时，其要么新建了一个全新的被点亮的连续段（周围都没被点亮），要么其合并了两个被点亮的连续段（中间只隔着未被点亮的格子 i）。

简化情形

先不管 s 和 t 的限制，并要求 $n \geq 3$，那么无论何时都不能点亮一个有恰好一个被点亮邻居的格子（否则它们至少一个不是开头或结尾，从而违反限制）。

对于 n 个格子，进行前 i 轮点亮过程（且均符合上述限制）后，设现在被点亮的格子构成 j 个连续段，发现这样的局面仍然难以转移：接下来需要点亮原来未被点亮的格子，但未被点亮的格子构成的集合不同，转移的方案也不同。

这是因为这是站在从 n 个格子按顺序点亮 i 个的角度来考虑的，如果仿照先前的思路，应该只对已经“存在”的 i 个被点亮的格子考虑。

简化情形

$dp_{i,j}$ 表示，一开始将 $1 \sim i$ 分为 j 段（每段非空），跨越段的相邻不算作相邻（因为之后中间必须插入至少一个元素），相同段内部的相邻才视作相邻（之后中间不能插入元素），所有这样的划分方案对应的子问题中，zig-zag 排列（定义为每个段非首尾元素必须是峰或谷）的数量之和。

对于一种 $1 \sim i$ 划分为 j 段的方案，再搭配上这种划分方案中的一个 zig-zag 排列。其有两种插入 $i+1$ 并重新诠释段划分的方式：

简化情形

- 声称 $i+1$ 对应的格子被点亮时，其两个邻居都被点亮了，从而它会合并两个连续段，这样即可保证 $i+1$ 对应的格子之后一定是峰。有 $j-1$ 种合并两个连续段的方式，于是将 $dp_{i+1,j-1}$ 加上 $(j-1)dp_{i,j}$
- 声称 $i+1$ 对应的格子被点亮时，其两个邻居都未被点亮，从而它会创建新的连续段，这样即可保证 $i+1$ 对应的格子之后一定是谷。有 $j+1$ 种创建一个连续段的方式，于是将 $dp_{i+1,j+1}$ 加上 $(j+1)dp_{i,j}$。

最终答案是 $dp_{n,1}$，这样就解决了没有 s,t 限制的情形，时间复杂度为 $\Theta(n^2)$。

再说两句

（对于初学者来说）状态的定义看上去有些奇怪：为什么要对所有划分方式将对应的 zig-zag 排列数量求和？

事实上这可以认为是：如果只保留所有 $\leq i$ 的数以及之间的相邻关系，长度为 n 的 zig-zag 排列会根据此时的形态被划分为若干等价类。对于不同等价类，再加入 $> i$ 的数它们一定还是不一样；而对于同一等价类，它们之间的差异是插入 $> i$ 的数时的操作不同所造成的。

所以 $dp_{i,j}$ 描述的是保留所有 $\leq i$ 的数以及之间的相邻关系，构成 j 个连续段的等价类数量。插入 $i+1$ 时，每个等价类会根据插入方式各自分裂，这一步的分裂方案数就是转移系数 $j-1$ 和 $j+1$。

对于有 s, t 限制的情况，留作习题（因为后面这道习题不是严格意义上的欧拉数问题）。

试试看！

给定长度为 n 的正整数序列 a，设 $m = \sum_{i=1}^{n} a_i$，问有多少长度为 m 的序列 b 满足：

- $\forall 1 \le i \le n$，i 在 b 中恰好出现 a_i 次。
- $\forall 1 \le i < m, |b_i - b_{i+1}| = 1$。

$1 \le n, a_i \le 2 \times 10^5$。

source：ARC146E Simple Speed。

容斥原理是什么？

这个问题没法回答，因为“容斥”这个词的语义事实上被滥用了，现在任何是“数方案数，但是加加减减算一下”的方法都可能被称为容斥。

不过针对常规的使用所谓“容斥原理”解决的问题，从以下视角看待问题可能更为简便：

容斥原理

- 现在要计数一些对象，设计数对象构成的全集是 A，某个具体的对象是 a。
- 对于 $a \in A$，我们会关心它是否满足指定的 n 种性质上的表现。设 f_i 是 $A \to \{0, 1\}$ 的映射，$f_i(a) = 1$ 当且仅当 a 满足第 i 种性质。
- 有两个预先指定好的长度为 n 的权值序列 p, q，$a \in A$ 的权重 $w(a) = \prod_{i=1}^{n} (p_i[f_i(a) = 1] + q_i[f_i(a) = 0])$。
- 目标是计算 $\sum_{a \in A} w(a)$。

容斥原理

- $w(a)=\prod_{i=1}^{n}(p_i[f_i(a)=1]+q_i[f_i(a)=0])$。
- **观察是**：$w(a)=\prod_{i=1}^{n}(p_i+(q_i-p_i)[f_i(a)=0])$。
- **所以**，$\sum_{a\in A}w(a)=\sum_{S\subseteq\{1,2\ldots n\}}(\prod_{i\notin S}p_i\prod_{i\in S}(q_i-p_i))cnt(S)$，**其中** $cnt(S)$ **表示有多少** $a\in A$ **满足** $\forall i\in S,f_i(a)=0$。

  这就是容斥原理。

容斥原理

容斥原理的经典例题是，给出长度为 n 的序列 b，求 $[1,m]$ 中有多少个数不是任何一个 a_i 的倍数。

以这个视角看待，$A=\{1,2...m\}$。$f_i(a)=[b_i \nmid a]$，而 $p_i=1,q_i=0$。

代入上面的公式，$p_i=1,p_i-q_i=-1$，从而答案是 $\sum_{S\subseteq\{1,2...n\}}(-1)^{|S|}cnt(S)$，其中 $cnt(S)$ 表示 $[1,m]$ 中有多少个数是 $\forall i\in S$ 的 b_i 的倍数，这与我们的认识相符。

再次计算欧拉数

可以这样看待欧拉数问题：A 是全体 n 阶排列构成的集合，有 $n-1$ 个性质 $f_1 \sim f_{n-1}$，对于排列 $a \in A$，$f_i(a)=1$ 当且仅当 $[a_i < a_{i+1}]$。

不同的是，现在 p_i, q_i 不再是一个数，我们采用以 x 为变元的多项式，让 $p_i = x, q_i = 1$。那么对于一个排列 $a \in A$，

$$\prod_{i=1}^{n-1} (p_i[f_i(a)=1] + q_i[f_i(a)=0]) = x^{\mathrm{asc}(a)}。$$

即 $\left\langle {n \atop k} \right\rangle$ 就是 $\sum_{a \in A} w(a)$ 的 $[x^k]$ 次项系数。

再次计算欧拉数

也可以把 $w(a)$ 写成 $w(a)=\prod_{i=1}^{n}(q_i+(p_i-q_i)[f_i(a)=1])$，求和公式会变为 $\sum_{a\in A}w(a)=\sum_{S\subseteq\{1,2\dots n-1\}}(\prod_{i\notin S}q_i\prod_{i\in S}(p_i-q_i))cnt(S)$，其中 $cnt'(S)$ 表示有多少 $a\in A$ 满足 $\forall i\in S,f_i(a)=1$。

代入 $p_i=x,q_i=1$，$\sum_{a\in A}w(a)=\sum_{S\subseteq\{1,2\dots n-1\}}(x-1)^{|S|}cnt(S)$，现在的问题无非是如何计算 $cnt'(S)$。

注意 $|S|$ 确定后，$(x-1)^{|S|}$ 也确定了。从而只需要对所有 $0\le k<n$，计算选定 $1\sim n-1$ 中 k 个位置，在这些位置上上升的排列数量之和。

选定 k 个位置必须上升相当于将 n 个格子分为 $n-k$ 段，段之间没有限制，而段内部必须上升。

接下来的推导可以用生成函数技巧，也可以不用。

方案 A：我不会生成函数！

注意到其实只需要确定 $1 \sim n$ 每个数被划分到 $n-k$ 个段中的哪个，因为段内部必须是升序，排列方式是唯一的。方案数似乎是 $(n-k)^n$？

但其实这样会计入某个段内部不包含任何值的划分方案，而这是不合法的。

所以现在要计数将 $1 \sim n$ 分到 $n-k$ 个有标号的盒子中，要求每个盒子都非空的方案数。

其实是斯特林数，不过不知道也无所谓。

根据容斥原理，答案是 $\sum\limits_{S \subseteq \{1,2 \ldots n-k\}} (-1)^{|S|}(n-k-|S|)^n$。

化简一下就是 $\sum\limits_{i=0}^{n-k}(-1)^i(n-k-i)^n\binom{n-k}{i}$，设为 ans_k。

而刚才已经得到答案是 $\left\langle {n \atop m} \right\rangle = [x^m]\sum\limits_{k=0}^{n}(x-1)^k ans_k$，推到这里已经可以 $\Theta(n^2)$ 计算一行欧拉数。

方案 A：我不会生成函数！

$$\left\langle {n \atop m} \right\rangle = \sum_{k\geq 0} \binom{k}{m} (-1)^{k-m} \sum_{i=0}^{n-k} i^n (-1)^{n-k-i} \binom{n-k}{i}。$$

$$= \sum_{k\geq 0} \sum_{i\geq 0} i^n \binom{k}{m} \binom{n-k}{i} (-1)^{n-i-m}。$$

$$= \sum_{i\geq 0} i^n (-1)^{n-i-m} (\sum_{k\geq 0} \binom{k}{m} \binom{n-k}{i})。$$

固定 i 后对 k 的求和是范德蒙德卷积，答案是 $\binom{n+1}{m+i+1}$。

最终结果是 $\left\langle {n \atop m} \right\rangle = \sum_{i=0}^{n-m} i^n \binom{n+1}{m+i+1} (-1)^{n-i+m}$。

方案 A：我不会生成函数！

$$\left\langle {n \atop m} \right\rangle = \sum_{i=0}^{n-m} i^n \binom{n+1}{m+i+1} (-1)^{n-i+m}。$$

对 $i \in [0, n-m]$ 计算 i^n 可以 $\Theta(n)$，据此可以 $\Theta(n)$ 计算欧拉数单项。

同时注意到设 $a_i = i^n, b_{i+m} = \binom{n+1}{m+i+1}(-1)^{n+i+m}$，则 $\left\langle {n \atop m} \right\rangle = \sum_{0 \le i,j \le n, j-i=m} a_i b_j$ 是标准的差卷积，于是进行一次卷积可以 $\Theta(n \log n)$ 计算一行欧拉数。

方案 B：我会生成函数！

n 个有标号的球分到 $n-k$ 个有标号的盒子要求都非空，因为不同盒子之间是归并标号所以可以设往一个盒子里放 k 个球的方案数序列 $0,1,1...$ 的 EGF 是 $\sum_{i\geq 1}\frac{y^i}{i!}=e^y-1$。

所以 $n-k$ 个盒子最后要求一共 n 个球的方案数就是 $[y^n](e^y-1)^{n-k}$。

$\left\langle {n \atop m} \right\rangle = n![x^m][y^n]\sum_{k\geq 0}(x-1)^k(e^y-1)^{n-k}$。

设 $F=\sum_{k\geq 0}(x-1)^k(e^y-1)^{n-k}$ 是以 x,y 为变元的二元生成函数，现在的目标是提取 F 的某一项系数。

$F=(x-1)^n\sum_{k=0}^{n}(\frac{e^y-1}{x-1})^k$。

$F=(x-1)^n\frac{1}{1-\frac{e^y-1}{x-1}}$，等比数列求和。

方案 B：我会生成函数！

$$F=(x-1)^n\frac{1}{1-\frac{e^y-1}{x-1}}。$$

$$F=(x-1)^n\frac{x-1}{(x-1)-(e^y-1)}，\text{分子分母同乘 }(x-1)。$$

$$F=-(x-1)^{n+1}e^{-y}\frac{1}{1-xe^{-y}}，\text{分子分母同乘 }-e^{-y}。$$

$$F=-\sum_{k\geq 0}x^ke^{-y(k+1)}(x-1)^{n+1}，\text{这里是将满足 } A \text{ 无常数项的 } \frac{1}{1-A} \text{ 展开为 } \sum_{k\geq 0}A^k。$$

提取 $[x^m][y^n]$ 结果是 $-\sum_{k\geq 0}\frac{(-(k+1))^n}{n!}[x^{m-k}](x-1)^{n+1}$。

进一步展开是 $-\sum_{k\geq 0}\frac{(-(k+1))^n}{n!}\binom{n+1}{m-k}(-1)^{n+1-(m-k)}$。

方案 B：我会生成函数！

$$-\sum_{k\geq 0}\frac{(-(k+1))^n}{n!}\binom{n+1}{m-k}(-1)^{n+1-(m-k)}。$$

乘上 $n!$ 并化简，答案是

$$\left\langle {n \atop m} \right\rangle = \sum_{k=0}^{m}(k+1)^n\binom{n+1}{m-k}(-1)^{m-k}。$$

和方案 A 一样，算出所有 k^n 后可以线性求一项。设 $a_k=(k+1)^n, b_{m-k}=\binom{n+1}{m-k}(-1)^{m-k}$，可以得到和卷积的形式 $\left\langle {n \atop m} \right\rangle = \sum_{0\leq i,j\leq n, i+j=m} a_i b_j$。一次 $\Theta(n\log n)$ 的卷积可以求出一行欧拉数。

再作一些变形可以得到方案 A 中的表达式，但不重要。

试试看！

给定 n 和长度为 $n-1$ 的 01 串 s，计算有多少长度为 n 的排列 p 满足：

- $\forall 1 \le i < n,\ [p_i < p_{i+1}] = s_i$。

  $1 \le n \le 10^5$。

  source：LOJ575 不等关系。

  从这里往后的题很多都需要用到卷积。

题解

沿用计算欧拉数时的建模，对于第 i 个条件，若 $s_i = 1$ 则让 $p_i = 1, q_i = 0$，否则让 $p_i = 0, q_i = 1$ 即可。

答案是 $\sum_{S \subseteq \{1,2 \ldots n-1\}} (\prod_{i \notin S} [s_i = 0])(\prod_{i \in S} (-1)^{[s_i=0]}) cnt(S)$。

当然这个时候相较于公式，视作对每条 $i, i+1$ 之间的边，如果方向是 $p_i < p_{i+1}$ 则不变；否则容斥成无限制减去 $p_i > p_{i+1}$ 更为直观。

设 dp_i 表示所有对 $1 \sim i$ 之间的边选定是否容斥的方案，容斥系数乘上所有段长度的阶乘的倒数之积的和。

转移是枚举 i 所在的段是 $[j, i]$，则 $j \sim i$ 之间的边，如果本来是小于则不变，是大于则必须容斥反向乘上容斥系数 -1，且 $j-1, j$ 之间的边必须容斥成无限制，而 $1 \sim j-1$ 这部分是子问题。

题解

认为 $s_0 = 0$，设 $c_i = \prod_{j=1}^{i} (-1)^{[s_i=0]}$。

转移是 $dp_i = \sum_{j=1}^{i} \frac{1}{(i-j+1)!} dp_{j-1} [s_{j-1} = 0] c_{i-1} c_{j-1}$。

整理后是半在线卷积的形式，CDQ 分治用 NTT 计算左边到右边的贡献即可 $\Theta(n \log^2 n)$。

试试看！

对于给定的 k 以及系数序列 $w_0 \sim w_{2^{k-1}-1}$，定义一个 $n \geq k$ 阶排列 p 的权值 $val(p) = \prod_{i=1}^{n-k+1} w_{f(p_i, p_{i+1} \dots p_{i+k-1})}$，其中 $f(a_1, a_2 \dots a_k) = \sum_{i=1}^{k-1} 2^{i-1} [a_i < a_{i+1}]$。

给定 n，计算所有 n 阶排列的权值和。

$1 \leq n \leq 10^5, 2 \leq k \leq 4$。

source：QOJ9651 又一个欧拉数问题。

题解

确定一个排列 p 的 $n-1$ 对相邻值 p_i, p_{i+1} 的大小关系之后，$val(p)$ 就确定了。

记这 $n-1$ 个相对大小关系信息是 01 序列 a，满足 $\forall 1 \le i < n, a_i = [p_i < p_{i+1}]$。

那么所有对应 a 的排列 p 的权值都是对所有 a 长度为 $k-1$ 的子区间，读出这个 $k-1$ 位二进制数的值作为下标，将对应的 w 全部乘起来。

原问题的答案就是对所有 a，将 a 对应的权值乘上符合 a 的 n 阶排列数量再求和。

上一题已经分析得到了，符合 a 的 n 阶排列数量就等于 $\sum_{b_1 \sim b_{n-1}, \forall b_i \ge a_i} (\prod_{i=1}^{n-1} (-1)^{a_i+b_i}) cof(b)$，其中 $cof(b)$ 是 $n!$ 除以 b 所有极长 1 连续段的长度 $+1$ 的阶乘之积。

题解

于是原问题的答案就是，枚举两个长为 $n-1$ 的 01 序列 a, b，要求 $a_i \le b_i$ 恒成立，将容斥系数 $\prod_{i=1}^{n-1}(-1)^{a_i+b_i}$ 乘上第一部分权值 $\prod_{i=1}^{n-k} w_{g(a_i,a_{i+1}\dots a_{i+k-2})}$ 再乘上第二部分权值 $cof(b)$ 加和，其中 $g(a_i, a_{i+1}\dots a_{i+k-2})$ 表示视作二进制数。

这样构造所有合法的 (a, b) 对：一开始选择 $k \ge 0$ 然后将 b 末尾接 k 个 1，而 a 末尾接 k 个 0/1；接下来重复若干次选一个 $k \ge 0$，向 b 末尾先接一个 0 再接 k 个 1，向 a 末尾先接一个 0 再接 k 个 0/1。

可以发现合法的 (a, b) 对恰好有一种构造方式，并且构造的每一步对应 b 的一个极长 1 连续段。

题解

可以计算 $cof_{k,S,T}$ 表示假设 a 末尾已经至少有 $k-2$ 个元素，且这些元素对应 S，接下来进行一次向 b 末尾添加 1 个 0 和 k 个 1 的拼接，对 a 的所有选择使得这次添加完成后 a 的末尾 $k-2$ 个元素对应 T 的方案的权值和，这个很容易 $\Theta(4^k n)$ 计算。

这里权值指的是新拼上去的段的容斥系数、a 部分的权值、b 部分的权值之积。

设 $2^{k-2} \times 2^{k-2}$ 矩阵 A 满足 $A_{S,T} = \sum_{k=0}^{n} x^{k+1} cof_{k,S,T}$。

假设构造过程中得到了一个长度为 $i \geq k-2$ 的 (a,b) 对，且 a 的尾 $k-2$ 个字符对应 S，那么以它为基础接下来进行若干次拼接最终得到长度为 $n-1$ 合法 (a,b) 对的所有方案的权值（新乘上去的部分）之和就是 $\sum_{T=0}^{2^{k-2}-1} [x^{n-1-i}](\sum_{s\geq 0} A^s)_{S,T}$。

题解

所以问题的关键就是计算 $\sum_{s=0}^{\infty} A^s \bmod x^n = \frac{1}{I-A} \bmod x^n$，即对于一个系数是多项式的矩阵在模 x^n 意义下求逆。

使用牛顿迭代可以转化为进行模 $x^1, x^2, x^4 \ldots x^n$ 意义下各 $O(1)$ 次系数为多项式的矩阵乘法，对每个位置分别 DFT 然后对每个单位根下的点值分别进行矩阵乘法加起来再 IDFT 回来就只需要做总长 $\Theta(4^k n)$ 的 DFT/IDFT 以及 $\Theta(8^k n)$ 时间的点乘。

接下来再递推计算 $st_{i,S}$ 表示进行若干次拼接得到长度为 i 的 (a, b) 对，且 a 末尾 $k-2$ 个字符（不存在的就不管）对应 S，**且一旦有至少 $k-2$ 个字符就立即停止**的所有方案的权值和即可。

“一旦有 $k-2$ 个字符就立即停止”使得这可以在 $\Theta(4^k n)$ 时间内计算。

总复杂度 $\Theta(4^k n \log n + 8^k n)$。

二项式反演

网上流传的 $\Theta(n)$ 或 $\Theta(n\log n)$ 求一项欧拉数的讲稿基本都会公式化地设 f_k 表示有多少排列恰好有 k 个上升，g_k 表示对所有 fix k 个位置必须是上升的方案（$\binom{n-1}{k}$ 种），求出有多少排列符合这 k 个位置都是上升的限制并加和。然后由组合意义可见 $g_i=\sum\limits_{j\geq i}\binom{j}{i}f_j$，然后使用“二项式反演”结论，$\sum\limits_{j\geq i}(-1)^{j-i}\binom{j}{i}g_j$ 经过一番推导后恰好等于 f_i，于是说一句只需求出 $g_0\sim g_{n-1}$ 后利用 $f_i=\sum\limits_{j\geq i}(-1)^{j-i}\binom{j}{i}g_j$ 就可以求出答案了。

可以发现前面提到的 ans_k 就是 g_k，而 $f_k=\left\langle {n \atop k}\right\rangle$，但我们并不是使用二项式反演将它们联系起来的。

与二项式反演的联系

前面容斥原理的模型给出 $\sum_{a\in A} w(a) = \sum_{k=0}^{n-1} (x-1)^k ans_k$，另一方面由定义 $\sum_{k=0}^{n-1} \left\langle {n \atop k} \right\rangle x^k = \sum_{a\in A} w(a)$。

代入 $\left\langle {n \atop k} \right\rangle = f_k, ans_k = g_k$，变为 $\sum_{k=0}^{n-1} f_k x^k = \sum_{k=0}^{n-1} g_k (x-1)^k$。

对两边提取 $[x^i]$ 项系数得到 $f_i = \sum_{j\geq i} (-1)^{j-i} \binom{j}{i} g_j$ 正是二项式反演的结论。

这表明两种方法其实是殊途同归的……吗？

二项式反演无用论

二项式反演确实是一个足够简便也能解决很多问题的结论，但它当然无法解决所有问题，它的能力范围甚至不如前面提到的容斥模型。

回到插入 DP 一节最后引入的问题，将问题稍作修改为：

- 给定 n, k 以及长度为 $n-1$ 值域为 012 的序列 w，对于一个长度为 n 的排列 p，定义 $\operatorname{asc}(p) = \sum_{i=1}^{n-1} w_i[p_i < p_{i+1}]$。
- 求出有多少长度为 n 的排列 p 满足 $\operatorname{asc}(p) = k$。

如果用此前的容斥模型，无非是 p_i 原本必然为 x，现在变为了 p^{w_i}。虽然输入的 w 序列导致求 $\sum_{S\subseteq\{1,2\ldots n-1\}} (\prod_{i\in S}(x^{w_i}-1))cnt(S)$ 难以高效完成，但容斥模型终究可以给出一个 $\Theta(n^3)$ 的解法。而机械化地代入二项式反演的式子就难以解决该问题了。

试试看！

给定 n, m 以及长度为 n 的序列 a，求有多少长度为 n 的排列 p 满足：设长度为 n 的序列 b 满足 $b_i = a_{p_i}$，则 $\mathrm{asc}(b) = m$。

$1 \le n \le 5 \times 10^5$。

source：ABC267G 加强版。

题解

还是视作值一样就是相同的重排，最后答案要乘上某个常数。

回顾本节是如何用方案 A 计算欧拉数的，根据相同的分析，关键在于计算将序列 a 的所有元素划分到 k 个上升段之一的方案数。

显然对每种颜色的划分仍然是独立的，而如果颜色 c 出现 occ_c 次，只需考虑从 k 个上升段中选出 occ_c 个包含颜色 c 的，方案数是 $\binom{k}{occ_c}$。

所以这个问题的答案是 $\prod_{c=1}^{n}\binom{k}{occ_c}$，然后把方案 A 中所有 k^n 换成这一项即可知道原问题的答案是：

$$\sum_{i=0}^{n-m}\binom{n+1}{m+i+1}(-1)^{n-i+m}(\prod_{c=1}^{n}\binom{i}{occ_c})。$$

题解

瓶颈是对 $\forall 1 \le i \le n$ 计算 $\prod_{c=1}^{n} \binom{i}{occ_c}$。

注意到 $\sum_{c=1}^{n} occ_c = n$，所以 occ_c 只有至多 $\sqrt{n}$ 种，对每种计算出现次数变为有 $a_1 \sim a_m, b_1 \sim b_m$，表示 $occ_c = a_i$ 的有 b_i 个，然后每次算 $\prod_{j=1}^{m} (\binom{i}{a_j})^{b_j}$ 即可。

看似复杂度是 $\Theta(n\sqrt{n}\log n)$，但其实对于递增正整数序列 a 以及正整数序列 b 且满足 $\sum_{i=1}^{m} a_i b_i \le n$，有 $\sum_{i=1}^{m} \log b_i$ 不超过常数倍 $\sqrt{n}$，证明略。

故时间复杂度为 $\Theta(n\sqrt{n})$。

取离散对数后做任意模数 NTT 也可以做到 $\Theta(n \log n)$（不考虑离散对数的时间），不过不重要。

试试看！

给定 n 以及一个长度为 n 的 01 序列 c，记一个长度为 n 的排列 a 的权值 $f(a)=\sum_{i=1}^{n} s(a_i, a_{(i \bmod n)+1})$，其中 $s(x,y)$ 定义为：

- 若 $c_x = c_y$ 则为 0。
- 若 $x < y$ 且 $c_x = 1, c_y = 0$，则为 0。
- 否则 $x > y$ 时为 $(-1)^{c_x}$，否则为 $(-1)^{c_y}$。

对于 $\forall -n \le k \le n$，计算有多少不同的长度为 n 的排列 a 满足 $a_1 = 1$ 且 $f(a) = k$。

$1 \le n \le 3 \times 10^3$。

source：CTT2024 Day3 比赛。

参考原题面可能更直观一点。

题解

如果觉得 $s(x,y)$ 的贡献直接是 $s'(x,y)=(-1)^{c_x}$，那么 $\sum_{i=1}^{n} s'(a_i, a_{i \bmod n+1})=0$ 恒成立。

接下来考察一下 $s(x,y)-s'(x,y)$ 的值是什么：

- 若 $x<y$ 且 $c_x=1, c_y=0$ 则 $s(x,y)=0, s'(x,y)=-1$，从而差为 1。
- 若 $x<y$ 且 $c_x=0, c_y=1$ 则 $s(x,y)=-1, s'(x,y)=1$ 从而差为 -2。

设 n 个性质 $f_1 \sim f_n$，设在 a 中 $a_j=i$，当且仅当 $a_{j-1}<i$（注意是循环的）时 $f_i(a)=1$。

让 $c_i=1$ 时 $p_i=x, q_i=1$；$c_i=0$ 时 $p_i=x^{-2}, q_i=1$，就可以将问题转化为求 $\sum_{a \in A} w(a)$ 的形式。

题解

答案是 $\sum_{S\subseteq\{1,2\ldots n\}}(\prod_{i\in S,c_i=0}(x-1))(\prod_{i\in S,c_i=1}(x^{-2}-1))cnt(S)$，其中 $cnt(S)$ 表示有多少 $a_1=1$ 的排列满足 S 内所有性质。

问题落到如何计算 $cnt(S)$，可以视作对每个 $i\in S$ 要选定一个 $j<i$ 且 $c_j\neq c_i$ 表示 j 必须恰好排在 i 的前面，当然每个 j 至多只能被选中一次。

已经选定的部分会组成若干条链，设是 k 条，注意 1 一定是某条链的开头，所以剩下的将 k 条链拼在一起且 $a_1=1$ 的方案数就是 $(k-1)!$ 种。

题解

而对 $c_i = 0$ 的 i 选 $j < i$ 的 $c_j = 1$ 的过程，与对 $c_i = 1$ 的 i 选 $j < i$ 的 $c_j = 0$ 的过程，是独立的。

选出 s 个 $c_i = 0$ 的位置，并为其中每一个选定一个编号比它小的 $c_j = 1$ 的位置的方案数之和，容易 DP $\Theta(n^2)$ 计算，设答案是 $ans_{0,s}$；同理另一边的答案是 $ans_{1,t}$。

那么最终答案就是

$$\sum_{s,t\geq 0} ans_{0,s} ans_{1,t} (n-1-s-t)!(x-1)^s (x^{-2}-1)^t。$$

先对每个 t 做一次 NTT 变为 $\sum_{s,t\geq 0} a_{s,t} x^s (x^{-2}-1)^t$ 的形式，再对每个 s 做一次 NTT 即可求出最终答案多项式。

总复杂度 $\Theta(n^2 \log n)$。

试试看！

给定 n，对 $\forall 0 \le x, y < n$，计算有多少长度为 n 的排列 p 满足 $\mathrm{asc}(p) = x, \mathrm{asc}(p^{-1}) = y$，其中 p^{-1} 表示 p 的逆排列。

$1 \le n \le 500$。

source：QOJ7759 Permutation Counting 2。

试试看！

有 n 张编号为 $1 \sim n$ 的牌，正面写着其编号，反面写着一个数对，记编号为 i 的牌写的数对是 (a_i, b_i)。

n 张牌排成一个序列，初始从左到右第 i 张编号为 i。

有两个按钮，按下第一个按钮后所有牌会按照 a_i 从小到大排序，如果 a_i 相同则原来的位置靠前的牌现在靠前；按下第二个按钮则是按照 b_i 排序，同理如果 b_i 相同按原来的位置排。

按按钮的操作是不可逆的，且任意时刻你只能观测到从左到右每张牌的编号。

目标是：猜出所有 (a_i, b_i)。

如果所有 a_i, b_i 是从 $[1, n]$ 之间独立均匀随机选取的，求在最优策略下你的期望胜率是多少。

$n \leq 5 \times 10^5$。

source：QOJ14023 International Olympiad in Fishing。

之前的建模还能做些什么

回顾之前的建模：

- $a \in A$ 的权重 $w(a) = \prod_{i=1}^{n} (p_i[f_i(a) = 1] + q_i[f_i(a) = 0])$。
- 可以改为：$w(a) = \prod_{i=1}^{n} (p_i + (q_i - p_i)[f_i(a) = 0])$。
- 也可以改为 $w(a) = \prod_{i=1}^{n} (q_i + (p_i - q_i)[f_i(a) = 1])$。

还有更多变形吗？

每个位置是独立的，i 相关的项是 $(p_i + (q_i - p_i)[f_i(a) = 0])$ 还是 $(q_i + (p_i - q_i)[f_i(a) = 1])$ 或者 $(p_i[f_i(a) = 1] + q_i[f_i(a) = 0])$ 即保持原来的不变都是可以的。

看上去很显然。

逐步容斥

$$(p_1+(q_1-p_1)[f_1(a)=0])\prod_{i=2}^{n}(p_i[f_i(a)=1]+q_i[f_i(a)=0])。$$

可以把前面的括号拆开，变为算 p_1 倍的，接下来只管 $2\sim n$ 这些性质的所有方案权值和；再加上 (q_1-p_1) 倍的，必须不满足性质 1 的所有方案，接下来只管 $2\sim n$ 这些性质对应的权值和。

观察：前后两个问题之间没有任何关系，这意味着如果接下来还想进一步容斥，对它们的容斥可以独立进行，即可以选定不同的容斥方向，简单来说就是选择 $(p_i+(q_i-p_i)[f_i(a)=0])$ 还是 $(q_i+(p_i-q_i)[f_i(a)=1])$。

看上去并没有什么用？注意必须不满足限制 1 与不管限制 1 可能是两个差别很大的问题！如果是这种情况，这样的精细处理带来的收益是巨大的。

例题

给定 k 以及长度分别为 $2^k-1, 2^k, 2^k-1$ 的 01 序列 a, b, c。称一个 2×2^k 的矩阵 A 合法，当且仅当：

- $\forall 1 \le i \le 2^{k+1}$，$i$ 在 A 中出现恰好一次。
- $\forall 1 \le i < 2^k$，$a_i = [A_{1,i} < A_{1,i+1}]$。
- $\forall 1 \le i \le 2^k$，$b_i = [A_{1,i} < A_{2,i}]$。
- $\forall 1 \le i < 2^k$，$c_i = [A_{2,i} < A_{2,i+1}]$。

求合法的矩阵 A 数量模 2。

T 组多测，每组的 k 一样但 a, b, c 序列可能不同。

$T \le 32, k \le 18$。

source：CTS2026 Day1 谜题 2

题解

相当于是对一个特殊的网格图做拓扑序计数：这个网格图 2 行 2^k 列，所有相邻的点之间都连了有向边。

根据容斥原理，对一张任意的图 G，可以任选 G 中有向边 $u \to v$，那么图 G 的拓扑序数量（下面简称答案）就等于从 G 中删去 $u \to v$ 得到图的答案，减去从 G 中反转 $u \to v$ 这条边方向得到图的答案。

考虑 2^k 列以及模 2 到底有什么用：总点数是 2^{k+1}，如果图不弱连通，答案一定是 0，原因是合并各个弱连通块答案会乘二项式系数，但这个系数模 2 一定是 0。

推论：如果视为无向图后一条边是这张无向图的割边，这条边的方向不影响答案，可以随意反向。

此外，由于模 2，容斥系数 -1 也可以直接不管。

题解

通过“逐步容斥”也就是每次选定一个子问题，选定其中一条边对它进行容斥从而该子问题分裂成两个，最终可以使得所有子问题满足：

- 网格图中所有横向的边一定指向右边。
- 所有点至多有一条出边。
- 图弱连通。

第三条性质不是很重要，因为如果不满足就直接丢弃这个子问题就可以了。

这样得到的子问题是每个点儿子数量不超过 2 的根向树，答案是 1 当且仅当对所有儿子数为 2 的点，其两个儿子的子树大小写作二进制无交。

接下来我们给出一个达成该目的的算法。方便起见我们先考虑原题满足 $\forall 1 \leq i < 2^k$ 满足 $b_i \neq b_{i+1}$，也就是所有竖着的边方向交错的子任务。

题解

对 p 进行归纳，作出以下归纳假设：

- 只对前 p 列内部的边进行容斥，就已经使得网格图前 p 列内部已经满足上述性质。
- 第 p 列的竖边没有被改变方向。

$p = 1$ 是归纳起点，考虑 $p \to p + 1$ 该如何做。

最简单的想法是直接容斥让 $p, p + 1$ 列之间的两条横边要么向右要么不存在，这样第一条性质就直接满足了。

第二条呢？好消息是 $p - 1, p$ 列之间的横边一定向右，从而 $(1, p), (2, p)$ 两个点的出边只有可能是 $p, p + 1$ 列之间的横边，或者连接 $(1, p), (2, p)$ 之间的边。

但这里还需要进一步讨论。

题解

现在的情况如图 1，虚线边表示这条边可能存在也可能不存在（对 $p, p+1$ 列之间横向边的容斥创造了一些子问题，在各个子问题中边的存在性可能不同），实线边表示这条边必然存在。

先对 $(1,p),(1,p+1)$ 之间的边是否存在分类讨论：

Case 1，$(1,p) \to (1,p+1)$ 的边不存在，如图 2。

此时 $(2,p) \to (2,p+1)$ 若不存在则图直接不连通，所以只需考虑其存在的情况。

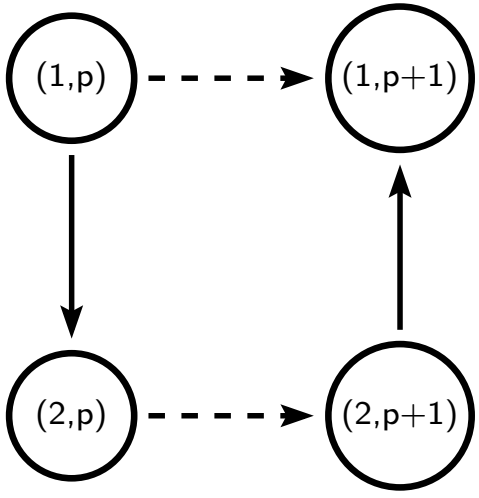

Figure: 1

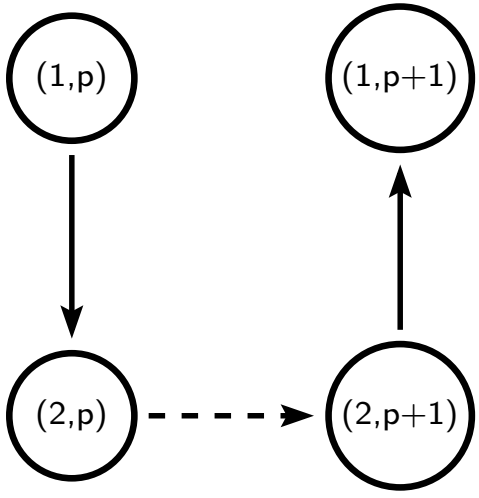

Figure: 2

题解

Case 2，$(1,p) \to (1,p+1)$ 的边存在，如图 3。

对 $(2,p),(2,p+1)$ 之间的边是否存在进一步讨论：

Case 2.1，$(2,p) \to (2,p+1)$ 的边存在如图 3.1。

可达性相同的 DAG 的答案一样，删去边 $(1,p) \to (1,p+1)$ 不改变可达性，所以可变为图 3.2 答案不变。

可以发现图 3.2 和图 2 长得完全一样。

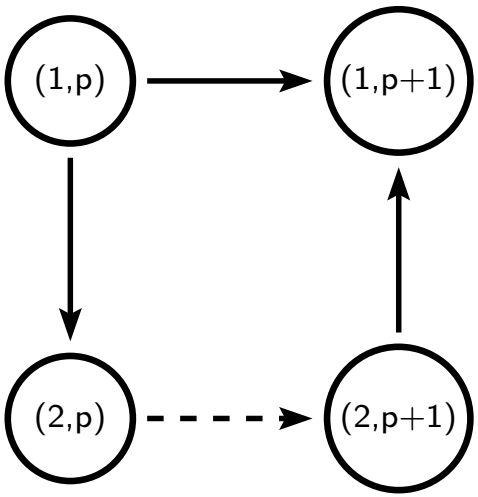

Figure: 3

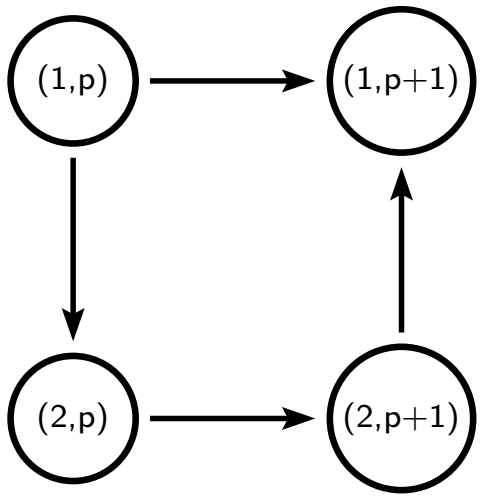

Figure: 3.1

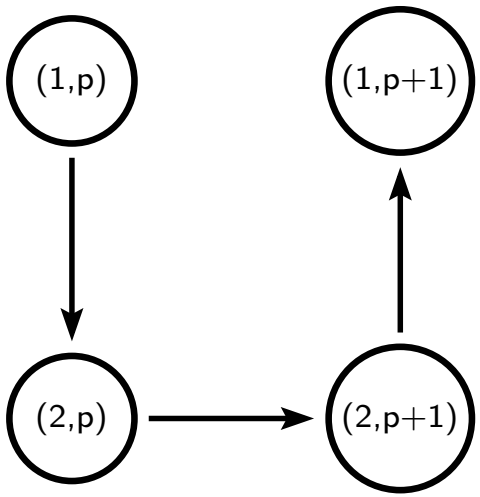

Figure: 3.2

题解

Case 2.2，$(2,p) \to (2,p+1)$ 的边不存在，如图 3.3。

此时 $(1,p)$ 和 $(2,p)$ 之间这条边是割边，可以总可以认为它向上如图 3.4。

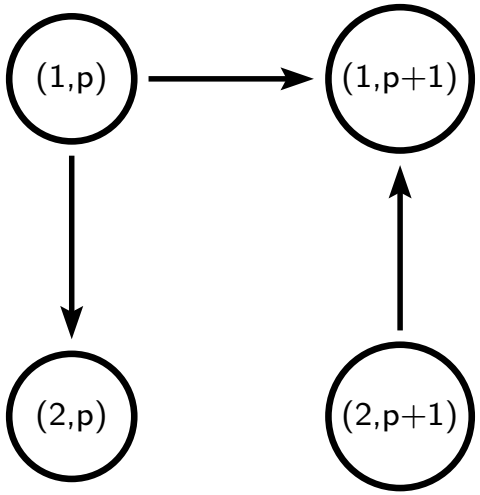

Figure: 3.3

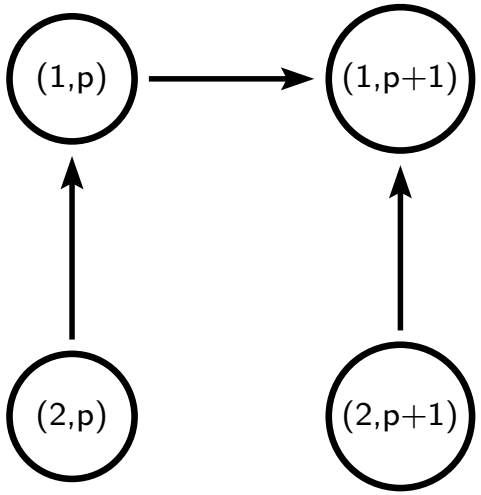

Figure: 3.4

题解

对于原题意下的限制 $A_{x_1,y_1} < A_{x_2,y_2}$ 连接有向边 $(x_1, y_1) \to (x_2, y_2)$。

上述分析表明当 $b_p = 1, b_{p+1} = 0$ 时，对于原先的子问题会分裂为：

- Case 1 要求第一步容斥后 $(1, p) \to (1, p + 1)$ 的边不存在，即原先方向是 $(1, p + 1) \to (1, p)$，条件是 $a_p = 0$。还要求 $(2, p) \to (2, p + 1)$ 的边存在，但不论这条边原先的方向，第一步容斥后总会得到一个向右的子问题，这里没有限制。
- Case 2.1 要求第一步容斥后两条边均存在，同上这里没有限制。
- Case 2.2 要求第一步容斥后 $(1, p) \to (1, p + 1)$ 存在而 $(2, p) \to (2, p + 1)$ 不存在，条件是 $c_p = 0$。

题解

简单来说：

- $a_p = 0$ 时得到一个如图 3.2 的子问题。
- 无论如何，得到一个如图 3.2 的子问题。
- $c_p = 0$ 时得到一个如图 3.4 的子问题。

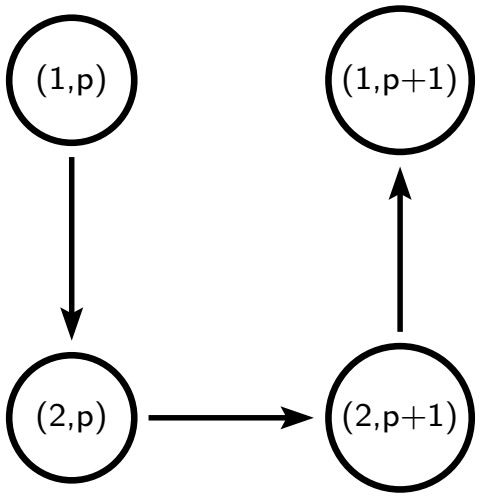

Figure: 3.2

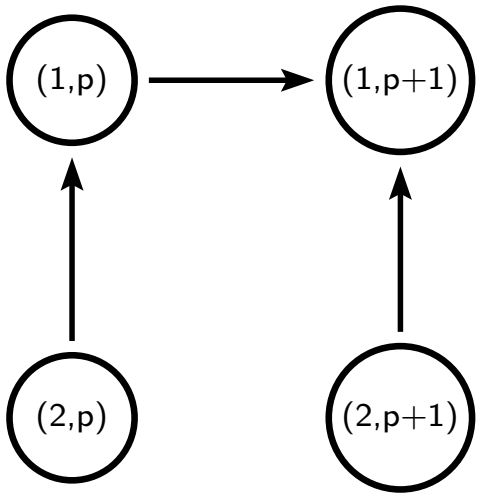

Figure: 3.4

题解

现在已经容斥得到期望的根向树的结构，假设对于一个已有的子问题，其二元组信息是 (x, y)，表示该子问题中 $1 \sim p-1$ 列的点一直沿着出边走，首个走到 $(1, p)$ 的点有 x 个，$(2, p)$ 的则是 y 个，在该容斥过程中：

- $a_p = 1$ 且 $valid(x+1, y)$ 时得到一个 $(0, x+y+2)$ 的子问题。
- $c_p = 0$ 且 $valid(x, y+1)$ 时得到一个 $(x+y+2, 0)$ 的子问题。

其中 $valid(s, t) = 1$ 当且仅当 s, t 二进制下无交。

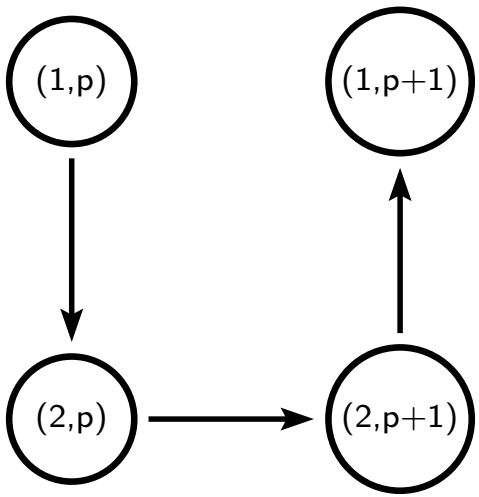

Figure: 3.2

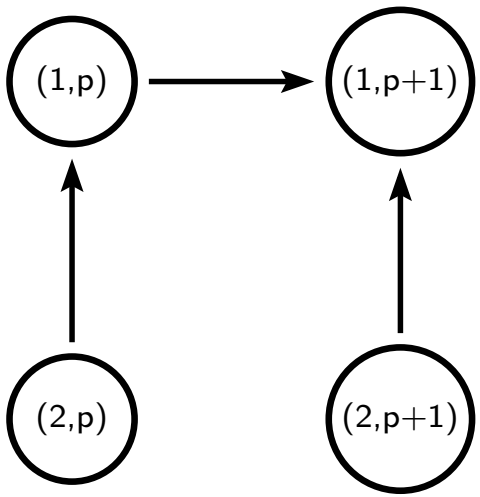

Figure: 3.4

题解

$b_p = 0, b_{p+1} = 1$ 的转移是对称的，注意到容斥进行到第 p 列时二元组必然形如 $(2(p-1), 0)$ 或 $(0, 2(p-1))$，记录有多少子问题对应这两种二元组 DP 转移即可。

但上述分析都基于 $\forall 1 \le i < 2^k, b_i \neq b_{i+1}$ 的假设。

题解

若其实 $b_p = b_{p+1}$，如图 4。可以先对 $(2,p) \to (1,p)$ 这条边容斥，得到的子问题中边方向变为 $(1,p) \to (2,p)$ 的（如图 4.1）是前面讨论的情形。

否则 $(1,p),(2,p)$ 间边不存在，由于原本前 p 列是树，现在切开 $(1,p),(2,p)$ 间边后 $(1,p),(2,p)$ 两点必然不连通，从而横跨 $p, p+1$ 列的边都是割边，都转成向右如图 4.2 即可。

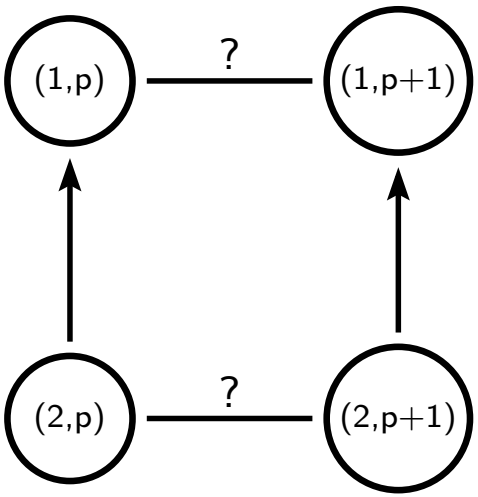

Figure: 4

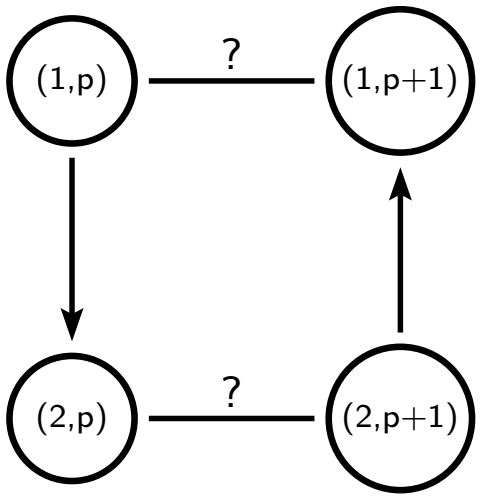

Figure: 4.1

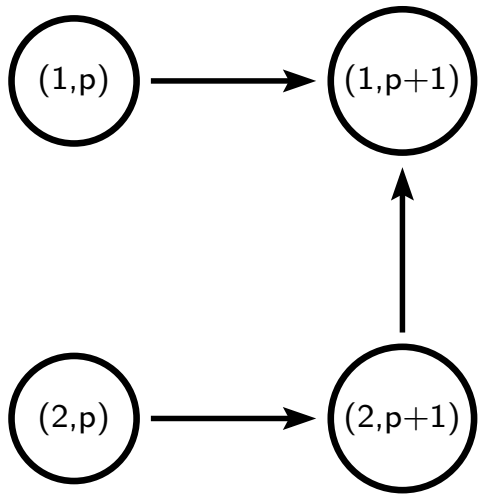

Figure: 4.2

题解

所以 $b_p = b_{p+1}$ 的情况只需要添加 $(x, y) \to (x+1, y+1)$ 的转移即可。

注意到 $x + y = 2(p-2)$ 恒成立，直接对每种二元组记录对应方案数转移是 $\Theta(4^k)$ 的。

进一步的优化是考虑二元组之间的转移要么是一直都 $+1$，要么是直接到达某一项为零的状态。

可以只记录所有某一项为零的状态的真实 DP 值，然后直接考虑这些状态间的转移，具体来说直接考虑进行多少次都 $+1$ 操作后汇聚到一个 $(0, *)$ 的状态或 $(*, 0)$ 的状态。

“汇聚”转移时有转移前两个二进制数无交的限制，这保证了有效的转移只有 $\Theta(3^k)$ 个。

题解

最后观察到 a, b, c 只控制一些潜在可行的转移是否真的进行，因此可以将代码写成“永远执行所有 $\Theta(3^k)$ 种潜在转移，而在转移时与上这个转移的条件”的形式。

DP 状态只有 1bit 信息 0/1，且每组数据中转移结构完全相同，因此可以压位在一轮计算内同时维护 w 组数据的 DP 值，其中 w 是计算机字长。

精细实现后，时间复杂度为 $\Theta(\left\lceil \frac{T}{w} \right\rceil 3^k + T2^k)$。

试试看！

定义一个 n 阶排列的超过数 $\mathrm{surp}(p) = \sum_{i=1}^{n} [p_i > i]$。

给定 n，对 $\forall 0 \le x, y \le n$，计算有多少长度为 n 的排列 p 满足 $\mathrm{surp}(p) = x, \mathrm{asc}(p) = y$。

$1 \le n \le 60$。

source：CTT2022 Day4 欧拉？欧拉！

注：使用刚才提到的逐步容斥的方法有一个分析上很简明的解法（但实现可能仍然比较痛苦），可以参考本题官方题解。

广告

使用逐步容斥，可以做到对任意有向广义串并联图（即对一张给定的广义串并联图每条边定向变为有向图）在 $\Theta(n^4)$ 时间内计算其拓扑序数量，如果允许用 NTT 就是 $\Theta(n^3 \log n)$。

如果是更为特殊一些的有向仙人掌，可以专门设计算法做到 $\Theta(n^3)$。

再特殊一点的有向树大家就很熟悉了，有经典的 $\Theta(n^2)$ 的算法。

具体可以参考我 24 年集训队论文《浅谈树拓扑序计数相关问题的一些方法》中的相关内容，文中大致梳理了有向树 -> 有向仙人掌 -> 有向广义串并联图这一脉络下每个情形的算法。

广告

论文还讨论了如果对以上三种情况予以特化，例如叶向树、叶向边仙人掌（这个定义可以参考论文）的情形如何加速计算。特殊的有向广义串并联图我在当时没有仔细考虑，论文发表到今天的两年内慢慢弥补了这个遗憾，我出的 CF2164F2 和刚才提到的谜题 2 相当于是两个特殊的广义串并联图的情形。

论文讨论的内容与今天讲课的内容关联很大但主题上略有差异，今天主要关注的是链上的情况且侧重研究上升数，而论文主要探讨在更复杂的情况下做拓扑序计数。

当然前面的讨论也表明研究上升数和计数拓扑序基本上是一回事，其中微小的差异具体是什么，大家了解足够多这方面的知识后应该可以自行体会到。

未知

最后列举几个未知的问题：

- 对于比广义串并联图更复杂的结构，该如何在多项式时间内统计其拓扑序数量？例如可以考虑 Apollonian Network。
- 此前提到的有向树、有向仙人掌、有向广义串并联图，虽然已经有多项式复杂度做法，能否做得更快？比方说有向树能做到低于 n^2 吗？有向仙人掌做到了 $\Theta(n^3)$ 而有向广义串并联图是 $\Theta(n^3 \log n)$，这两个结果并列在一起显得并不自然。
- 对于一些已经被解决的问题（尤其是 EI 解决的一些复杂的欧拉数问题），它们的做法太过复杂、非常难以理解，因此被束之高阁，能否简化这些问题的求解？

扩展阅读

讲课的核心内容已经结束了，最后简单提两个相关的方向，是概率、测度方法以及求解迭代列。

重访欧拉数

考虑从 $(0,1)$ 独立均匀随机采样 $a_1 \sim a_n$，则出现重复值的概率是 0，而 $a_1 \sim a_n$ 的相对顺序有 $n!$ 种，每种出现的概率都是 $\frac{1}{n!}$。

所以，$\frac{1}{n!}\left\langle {n \atop k} \right\rangle$ 就等于 $\operatorname{asc}(a) = k$ 的概率。

根据 a 序列，构造 b 序列满足 $b_i = (a_i - a_{i-1}) \bmod 1$（认为 $a_0 = 0$）长度为 n，一个有趣的观察是 $n - 1 - \operatorname{asc}(a) = \left\lfloor \sum_{i=1}^{n} b_i \right\rfloor$。

原因：相当于初始站在 0，$\forall 1 \le i \le n$，向右走直到走到 a_i，如果碰到 1 就回到 0（即是循环的），$\left\lfloor \sum_{i=1}^{n} b_i \right\rfloor$ 描述了碰到 1 了多少次，而每个 $a_i > a_{i+1}$ 的 i 会导致必须碰到 1 一次。

重访欧拉数

可以想象，$(0,m)^n$ 上所有坐标之和不超过 m 的点构成的图形的测度应该是 $m=1$ 时的 m^n 倍，所以是 $\frac{m^n}{n!}$。

但现在是 $(0,1)^n$，考虑测度是可减的，可以容斥枚举一个子集 $S\subseteq\{1,2...n\}$ 计算所有 $(a_1,a_2...a_n)\in(0,m)^n$ 且 $\forall i\in S$ 满足 $a_i\geq 1$ 并 $\sum_{i=1}^{n}a_i\leq m$ 的点 $(a_1,a_2...a_n)$ 构成的测度，乘上 $(-1)^{|S|}$ 加和即可。

所以答案是 $\sum_{i=0}^{n}(-1)^i\binom{n}{i}\frac{(m-i)^n}{n!}$，设这是 $f(m)$，那么 $\left\langle {n \atop k}\right\rangle=n!(f(k+1)-f(k))$。

进一步化简可以说明这个式子和先前得到的等价，就不再进行了。

重访欧拉数

可以想象，$(0,m)^n$ 上所有坐标之和不超过 m 的点构成的图形的测度应该是 $m=1$ 时的 m^n 倍，所以是 $\frac{m^n}{n!}$。

但现在是 $(0,1)^n$，考虑测度是可减的，可以容斥枚举一个子集 $S\subseteq\{1,2...n\}$ 计算所有 $(a_1,a_2...a_n)\in(0,m)^n$ 且 $\forall i\in S$ 满足 $a_i\geq 1$ 并 $\sum_{i=1}^{n}a_i\leq m$ 的点 $(a_1,a_2...a_n)$ 构成的测度，乘上 $(-1)^{|S|}$ 加和即可。

所以答案是 $\sum_{i=0}^{n}(-1)^i\binom{n}{i}\frac{(m-i)^n}{n!}$，设这是 $f(m)$，那么 $\left\langle {n \atop k}\right\rangle=n!(f(k+1)-f(k))$。

进一步化简可以说明这个式子和先前得到的等价，就不再进行了。

试试看！

给定 n 和长度为 n 的序列 a，现在从 $[0,1]$ 中独立等概率随机得到 $b_1 \sim b_n$，问满足以下条件的概率：

- $\forall 1 \le i \le n, \sum_{j=1}^{n} b_j \le \frac{a_i}{10^6}$。

$n \le 30$。

source：CF913H Don't Exceed。

试试看！

给定 n, m, k，对 $\forall 1 \leq i \leq n$，求有多少长度为 n 的排列 p 满足 $\operatorname{asc}(p) = m$ 且 $p_k = i$。

$n \leq 5 \times 10^5$。

source：UOJ593 新年的军队。

提示：要解决这道题，今天讲的内容远远不够，感兴趣的同学可以自行研究。

求解迭代列

回顾一开始我们得到的递推关系 $\left\langle {n \atop k} \right\rangle = \left\langle {n-1 \atop k} \right\rangle (k+1) + \left\langle {n-1 \atop k-1} \right\rangle (n-k)$，如果设欧拉数的第 n 行的生成函数是 $F_n(x)$ 的话，这个递推关系给出了 $F_{n+1}(x) = (nx+1)F_n(x) + x(1-x)F'_n(x)$，边界是 $F_0(x) = 1$。

那么 $F_{n+1}(x) = (nx+1)F_n(x) + x(1-x)F'_n(x)$ 这一迭代列应该蕴含了欧拉数的全部信息，从这个角度推导出后面几个彼此等价的线性求欧拉数单项的式子应该是可行的。

但不幸的是，求解带微分的迭代列是困难的，即使我们只允许递推关系是常数阶微分并乘上常数次多项式的线性组合（而且这个递推关系不含与 n 相关的内容），一般来说这个迭代列也无法求解。而这里递推式的 $nxF_n(x)$ 这一与 n 关联的项会增加更多的困难。

求解迭代列

不过如果设 $G_n(x) = \dfrac{F_n(x)}{(1-x)^{n+1}}$，则：

$$G'_n(x) = \frac{F'_n(x)}{(1-x)^{n+1}} + (n+1)\frac{F_n(x)}{(1-x)^{n+2}}。$$

$$G_n(x) + xG'_n(x) = \frac{(1-x)xF'_n(x) + (n+1)xF_n + (1-x)F_n}{(1-x)^{n+2}}。$$

$$= \frac{(1-x)xF'_n(x) + (nx+1)xF_n}{(1-x)^{n+2}} = G_{n+1}(x)。$$

$G_{n+1}(x) = G_n(x) + xG'_n(x)$ 这个迭代关系是很好处理的，因为它的效果是给 $[x^k]$ 系数点乘 $(k+1)$，而递推起点是 $G_0(x) = 1 + x + x^2 + \ldots$，从而 $[x^k]G_n(x) = (k+1)^n$。那么 $F_n(x) = G_n(x)(1+x)^{n+1}$，$F_n(x)$ 的系数也容易提取了。

求解迭代列

疑惑当然是 $G_n(x)$ 是怎么凑出来的，事实上可以待定一列多项式 $Q_0, Q_1 \ldots$ 要求 $G_n(x) = F_n(x)Q_n(x)$ 且 $G_n(x)$ 满足较为简单的迭代关系（这一步需要枚举），并尝试解出 Q_i（当然，不总能做到）。

如果你想对这个问题有更深入的了解的话，可以阅读一开始提到的另一个欧拉数问题的题解，这道题的迭代关系是 $F_{n+1}(x) = (\alpha n x + 1)F_n(x) + x(1-x)F'_n(x)$，会更难以处理。

或者你也可以参考 求解迭代列问题的一些手段。

致谢

谢谢大家！