浅谈树拓扑序计数相关问题的一些方法

杭州第二中学 陈昕阳

摘 要

树拓扑序计数问题是一般图拓扑序计数问题的一个特殊情形。相较于一般图拓扑序计数问题目前不存在多项式复杂度的做法，由于树的结构较为特殊简单，树拓扑序计数问题存在优秀的性质，利用这些性质可以在线性时间内求解该问题。本文从树拓扑序计数问题出发，主要介绍解决树拓扑序计数问题及其拓展问题的几种常用方法，并通过一些例题展示了这些方法在信息学竞赛中的应用。

1 前言与约定

图拓扑序计数问题是图论中的经典问题：给定一张 n 个点 m 条边的有向图，请问其有多少不同的拓扑序？虽然找到有向图的任意一种拓扑序或报告无解是简单的，目前对拓扑序计数却不存在多项式复杂度的做法。

但如果对图这一结构进行弱化变为有向树（将所有有向边视为无向边后形成一颗树的有向图），并且保证存在一个点 r 使得图是以 r 为根的叶向树时（所有有向边方向是从深度较小点指向深度较大点的有向树），这个问题则存在非常优美的解法与结论，有时被称为树拓扑序计数问题。事实上这一问题在信息竞赛中经常作为转化后的题意出现，并且也已经有了许多方面的拓展。

本文将由浅入深地介绍解决这一类问题的各种方法，以及用这些方法该如何解决各种拓展问题。这些方法与问题一部分是已有的，一部分则是本文原创。

1.1 约定

一张图的拓扑序数量最多可能达到 $n!$，在 n 较大时会难以储存，所以我们约定只需计算答案取模大质数的结果。这同时意味着我们认为对较小的数（比如 $[1,n]$ 内的）求逆元总是可行。

事实上预处理 $1 \sim n$ 的逆元还需要做一次 $O(\log mod)$ 的快速幂，下文也会忽略这部分的复杂度。

本文中提到的各种概念，如果没有特殊说明，一般与大众所了解的定义一致。

2 从经典问题引入

本节会介绍树拓扑序计数问题的经典做法，以及从叶向树到叶向森林（每个连通块都是叶向树的有向图）的拓展。

2.1 树拓扑序计数问题的经典解法

先明确一下问题：

例题一[1]：给定一张 n 个点以 r 为根的叶向树，请问其有多少种不同的拓扑序？$n \le 10^6$。

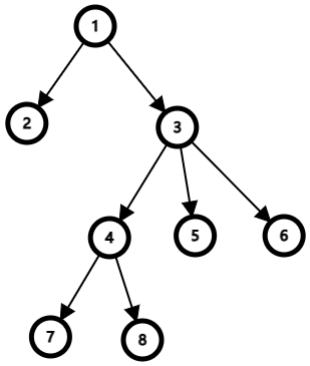

图 1: 一个叶向树的例子

该问题为人熟知的做法是使用动态规划，设 dp_u 表示仅考虑 u 子树内的点和边构成的子图的拓扑序数量，显然最终答案就是 dp_r。自下至上 DP，假设对于点 u，其所有孩子 $v_1, v_2 ... v_k$ 的 DP 值都已经计算完成，考虑如何求出 dp_u。首先因为 u 直接或间接地可以到达其子树内的所有点，所以其必须排在最前面，此后我们假设 $v_1, v_2 ... v_k$ 每个子树内所有节点的排列顺序已经确定，那么根据树的结构，$v_1, v_2 ... v_k$ 这些子树之间是没有先后限制的（之间没有边），问题等价于归并长度依次为 $siz_{v_1}, siz_{v_2} ... siz_{v_k}$（$siz_p$ 表示 p 的子树大小）的序列，确定 $v_1, v_2 .. v_k$ 子树内所有节点排列顺序有 $dp_{v_1} \times dp_{v_2} \times ... dp_{v_k}$ 种方案，而归并这些序列有 $\binom{\sum_{i=1}^{k} siz_{v_i}}{v_1, v_2 ... v_k}$ 种方案，相乘即可。

该做法还可以进一步化简，可以归纳证明 dp_u 事实上等于 $\frac{(siz_u)!}{\prod_{\text{u is ancestor of p}} siz_p}$，即 dp_u 是 u 子树大小的阶乘除以其所有后代的子树大小之积。当 u 是叶子时这一点显然成立，而 u 并非叶子时，将多重组合数拆开，等于要乘上 $(\sum_{i=1}^{k} siz_{v_i})! = (siz_u - 1)! = \frac{(siz_u)!}{siz_u}$，除以所有 $(siz_{v_i})!$，

[1]题目来源：经典问题

而根据归纳假设 $dp_{v_i} = \dfrac{(siz_{v_i})!}{\prod\limits_{v_i \text{ is ancestor of p}} siz_p}$，整理后即可得到结论。

时间复杂度为 $O(n)$。

2.2 拓展到叶向森林

叶向森林即每个弱连通块都是一颗叶向树。

上文的经典做法其实基于两点：

- 如果一张图可以被划分为若干部分，不同的部分之间没有限制（连边），那么我们只需将各部分答案相乘，再乘上一个多重组合数即可得到整体的答案。
- 如果图中是存在一个点必须先于剩下的所有点，那么因为这个点必须占据拓扑序的第一个位置，可以将其直接删去。

  那么叶向森林的情况根据第一点，只需分别求出每颗叶向树的答案，然后用多重组合数合并即可。

  类似地，可以得到答案是 $\dfrac{n!}{\prod\limits_{i=1}^{n} siz_i}$，求解的时间复杂度仍为 $O(n)$。

  此处特别指出图为叶向森林情况的答案公式，是因为后文会试图把更复杂的问题规约到叶向森林的情况，并使用该公式求解。

3 经典问题的新解法

树拓扑序计数问题的答案公式非常简洁优美，笔者最近发现该公式存在组合意义下的解释，对比之前的数学归纳法，利用该组合解释还可以立刻解决一个较为复杂的拓展问题。

3.1 组合解释

考虑以下针对给定叶向树的过程：

- 每个点有黑或白的颜色，初始所有点都是白色，另有一个长度为 n 的辅助序列 a。
- 进行 n 轮操作，每轮选一个白色点（之前选过的点也可以再选），找到其最浅的白色点祖先 p，将其涂黑。假设这是第 i 轮，记录 $a_i = p$。

可以发现，无论怎么选择每一轮的白色点，最终得到的序列 a 都是原叶向树的一个拓扑序。但一个拓扑序可能对应多种 n 轮每轮选择一个白色点的方案。

不过可以观察到，当 a 序列是一个拓扑序时，会生成其的选择方案永远是 $\prod_{i=1}^{n} siz_i$ 种。这是因为，如果拓扑序第 i 位是 u，这代表在过程中第 i 轮被涂黑的点是 u，而在这一轮 u 被涂黑的充要条件是，选择的白色点 v 在 u 子树内，所以涂黑 u 的一轮选择白色点的方案数是 siz_u，而不同轮之间选择白色点的方案数是独立的，由此得到结论。

而可以注意到，n 轮每轮选择一个白色点的方案数是 $n!$。因为该过程必定生成一个拓扑序，且对于任意拓扑序，都存在恰 $\prod_{i=1}^{n} siz_i$ 种生成其的方案。所以拓扑序的数量是 $\frac{n!}{\prod_{i=1}^{n} siz_i}$，得证。

3.2 一个拓展问题

树拓扑序计数问题事实上是排列计数问题。排列计数问题的经典拓展方式是固定其中一项 $p_x = y$，统计给剩下的项填上值并且符合限制的方案数。

那么如果对树拓扑序计数问题进行该拓展，该如何解决呢？

例题二[2]：**给定一颗 n 个点，以 1 为根的叶向树和长度为 n 的权值序列 b。定义 $f(u)$ 为，所有该图的拓扑序中 b_{idx_u} 之和（其中 idx_u 表示在拓扑序上 u 出现的位置）。对 $1 \le u \le n$，求 $f(u)$。$n \le 5000$。**

事实上我们可以解决更一般的问题：计算所有 $ans(x,y)$ 表示 x 出现于第 y 位的拓扑序数量。

考虑组合解释中描述的过程，限制转化为 x 在第 y 轮被涂黑。可以发现要知道 x 点在哪一轮被涂黑，只需知道 x 的祖先中最浅的白点 p 是谁即可。每次从剩余白点中选一个也可分为两种情况，第一种是选中了 p 子树内的点，那么 p 被涂黑，最浅的白点变为 p 到 x 简单路径上的第二个点（或者如果 $p = x$ 那么我们知道这一轮 x 被涂黑）；或者选中的白点不在 p 子树内，最浅的白点仍是 p。

于是可以这样设计 DP：设 $dp_{i,p}$ 为已经涂完 i 轮，x 祖先中最浅的白点是 p 的涂色方案数。边界是 $dp_{0,1} = 1$，转移是枚举第 i 轮的涂色情况：要么在涂完 $i-1$ 轮后最浅的白点就是 p，第 i 轮选的点不在 p 子树内，那么 $dp_{i,p}$ 加上 $(n+1-i-siz_p)dp_{i-1,p}$；要么是涂完 $i-1$ 轮后最浅的白点是 fa_p，第 i 轮选的点在 fa_p 子树内，那么 $dp_{i,p}$ 加上 $siz_{fa_p}dp_{i-1,fa_p}$。

最后 $ans'(x,y) = dp_{y-1,x}(n-y)!$，因为前 $y-1$ 轮涂色一定要涂黑 x 的所有祖先，并且 x 仍为白色，这样的方案数是 $dp_{y-1,x}$，第 y 轮一定要涂 x，而后 $[y+1,n]$ 这些轮则任意涂。

这里是 $ans'(x,y)$ 是因为我们现在统计的是 x 在第 y 轮被涂黑的不同选择方案数，但每 $\prod_{i=1}^{n} siz_i$ 种选择方案对应一种拓扑序，所以真正的 $ans(x,y) = \frac{ans'(x,y)}{\prod_{i=1}^{n} siz_i}$。

对于原问题则有 $f(u) = \sum_{i=1}^{n} ans(u,i)b_i$。

[2]题目来源：https://qoj.ac/problem/7792

时间复杂度 $O(n^2)$，对 i 一维使用滚动数组则能轻易做到线性空间。

3.3 拓展问题的进一步拓展

拓展问题中我们 $O(n^2)$ 算出了所有带单项钦定的答案。事实上使用类似的方法，可以 $O(n^{2k})$ 算出对于所有钦定 k 元组（这有 $\binom{n}{k}n^k$ 种）的答案，即所有从 n 个点中选 k 个点，分别钦定这 k 个点必须填 k 种对应值的答案。这里我们认为 k 是小常数。

但稍有不同的是，在单项钦定的情况中，一旦目标点 x 被涂黑，接下来的涂色过程就不需考虑了，可以直接乘上任意涂色的方案数 $(n-y)!$。在多项钦定的情况下，事实上要记录 $dp_{i,S,T}$ 其中 T 是一些二元组构成的集合，S 是 $\{1,2,3...n\}$ 的子集，表示已经涂了 i 轮，在这 i 轮中作出限制 $T=\{(a_j,b_j)\}$，点 a_j 必须在第 b_j 轮被涂黑，S 集合是所有尚未被涂黑的带有钦定的点祖先中首个白点构成的集合。

转移同样考虑第 i 轮涂黑了哪个点，可以发现被涂黑的点要么是 S 集合中某个点 p 的父亲，那么前驱状态就是向 S 集合加入 fa_p 并删去 fa_p 所有孩子；要么被涂黑的点在 T 中被记录；要么第 i 轮操作前 S 集合和第 i 轮操作后一模一样。

只保留 $|T|+|S|\le k$ 的 DP 状态，并且对于 $|T|=k$ 的状态直接结算答案不再继续转移，即可做到 $O(n^{2k})$ 的复杂度。

4 容斥原理与枚举值域线

叶向森林的情况答案有非常简洁的公式，但使用条件稍显苛刻：必须满足每个点入度不超过 1，并且图无环。

使用容斥原理可以将不满足条件的图拆分为很多满足该条件的子问题，只需将子问题的答案带权求和（乘上对应容斥系数），即可得到原问题的答案。

枚举值域线则是一种拆贡献的技巧，如果要计算 $|u-v|$，其中 $u,v\in\mathbb{Z}$，只需计算 $\sum_{k\in\mathbb{Z}}[u\le k][v>k]+[v\le k][u>k]$。

本节将介绍如何使用容斥原理解决两个经典问题，并给出维护值域线技巧的一道例题。

4.1 有向树拓扑序计数

例题三[3]：给定一张 n 个点的有向图，保证将所有有向边视为无向边后图是一颗树，请问其有多少种不同的拓扑序？$n\le 5000$。

任意定根 r，可以将限制边分为两类，根向的和叶向的。如果所有边均为叶向，那么显然这是叶向森林拓扑序计数。

[3]题目来源：https://loj.ac/p/3802

但事实上可能还有根向边，使用容斥原理枚举一个子集的根向边违反限制变为叶向，其余根向边忽略不管。这样得到了 $2^{\text{根向边}}$ 个子问题，每个子问题都是叶向森林拓扑序计数，将每个子问题的答案乘上容斥系数加和即可得到原问题答案。

具体地，我们设 $dp_{u,i}$ 表示确定了 u 子树内所有根向边是否违反限制，u 所在叶向树有 i 个点的 $\dfrac{(-1)^{|S|}}{\prod\limits_{\text{u is ancestor of v}} siz_v}$ 之和。转移时先递归到 u 的各个孩子去计算 DP 值，然后每次合并上一个孩子 v_i。如果 u 和 v_i 的边是叶向的，那么直接进行背包合并；如果 u 和 v_i 的边是根向的，那么有两种情况，第一种这条边变为无限制，那么合并后 u 所在叶向树点数不变；第二种这条边容斥变为叶向，那么系数要乘 -1，也是进行背包合并。

合并完各个孩子的信息后，对于 $dp_{u,i}$，将值乘上 $\dfrac{1}{i}$。

最后答案是 $n!\sum\limits_{i=1}^{n} dp_{r,i}$。

时间复杂度根据树形背包合并的复杂度分析，是 $O(n^2)$ 的。

4.2 有向链的特殊情况

如果对有向树进一步特化，求解拓扑序数量的复杂度可以进一步降低。

例题四[4]：给定一张 n 个点的有向图，保证将所有有向边视为无向边后图形成一条链，请问其有多少种不同的拓扑序？$n \le 10^5$。

不妨认为链上点的编号从左到右分别是 $1 \sim n$。

如果图由许多条链组成，而每条链的所有边方向都一致，那么利用叶向森林的拓扑序计数公式可知，设这些链的长度分别为 $a_1, a_2 ... a_k$ 答案是 $\dfrac{n!}{\prod\limits_{i=1}^{k} a_i!}$。

比如说我们希望所有边的方向都是从左到右，那么遇到从右到左的边就需要容斥，要么让这条边违反限制，要么不管这条边。设 dp_i 为已经确定了前 $i+1$ 个点之间边的容斥方向，i 作为一条链的右端点的所有方案的 $\dfrac{(-1)^{|S|}}{\prod\limits_{j=1}^{k} a_j!}$ 之和，这里 $a_1 \sim a_k$ 是前 i 个点构成的链长度的集合。

可以发现 dp_i 不为 0，当且仅当要么 i 和 $i+1$ 之间的边初始是从右到左的，要么 $i = n$。转移是枚举 i 所在链的左端点 j，将 dp_i 加上 $dp_{j-1}\dfrac{(-1)^{sum_i - sum_j}}{(i-j+1)!}$，其中 sum_i 表示 $1 \sim i$ 这些点之间边从右到左的数量。

可以发现这是卷积形式，进行半在线卷积即可 $O(n \log^2 n)$ 算出所有 $dp_1 \sim dp_n$。最后答案为 $n!dp_n$。

[4]题目来源：https://loj.ac/p/575

4.3 更复杂的答案形式

有一些树拓扑序计数的拓展问题不仅要求数出方案数，还针对拓扑序的具体内容赋予了对应权值，要求求出所有合法拓扑序的权值之和。

例题五[5]：给定一颗 n 个点，以 1 为根的叶向树。定义一个合法拓扑序 a 的权值 $f(a)$ 为 $\sum_{i=1}^{n-1}|a_i-a_{i+1}|$，对所有不同的 a，求 $f(a)$ 之和。合法拓扑序 a 的定义是长度为 n 的排列满足 $\forall 2\le u\le n, a_u>a_{fa_u}$。$n\le 500$。

考虑先固定 i，试图算出所有合法拓扑序的 $|a_i-a_{i+1}|$ 之和。枚举值域线 k，不失一般性变为求 $a_i\le k$ 且 $a_{i+1}>k$ 的拓扑序数量。

将每个点 u 按照其填上的值 a_u 涂色，如果 $a_u\le k$ 将其涂为白色，否则将其涂为黑色。那么限制是 i 点必须为白色，$i+1$ 点必须为黑色。

考虑如果一种涂色方案已经被确定，如何计算有多少拓扑序对应这种涂色方案。首先要保证对于所有边 $u\to v$，不存在 v 是白色而 u 是黑色，否则不可能有拓扑序对应该涂色方案。接下来，只要白色点填的值在 $[1,k]$ 内，黑色点填的值在 $(k,n]$ 内，所有两侧点颜色不同的边的拓扑序限制一定是自然得到满足的。

接下来只需考虑两侧点颜色相同的边给予的拓扑序限制，那么首先每种颜色独立。只考虑某种确定的颜色的话，可以发现问题等价于叶向森林拓扑序计数，可以直接套用公式。我们记 siz'_u 为 u 子树内有多少点与 u 同色，那么答案就是 $k!(n-k)!\prod_{i=1}^{n}\frac{1}{siz'_i}$，当然我们需要保证白色点有 k 个，黑色点有 $n-k$ 个。不过换个角度不直接枚举 k 的话，可以看作是该方案带有两种颜色出现次数阶乘之积的权值。

直接 DP，设 $dp_{u,k}$ 为确定了 u 子树内所有点的颜色，u 为白色，子树内共有 k 个黑色点，并且若 $i,i+1$ 在子树内的话，i 必须为白色，$i+1$ 必须为黑色的限制得到满足的所有涂色方案的 $\frac{1}{siz'_i}$ 之和。转移是先对所有孩子的信息背包合并（当然孩子是可以为黑色的，此时对其子树涂色只有一种方案，对应权值也容易预处理），再考虑自己的 $\frac{1}{siz'_u}$ 权值以及如果 $u=i$ 或 $u=i+1$ 的话这个点的黑白限制。最后算答案时方案数是 $\sum_{k=1}^{n}dp_{1,k}k!(n-k)!$。

一轮这样的 DP 是 $O(n^2)$ 的，要进行 $n-1$ 轮，故总复杂度为 $O(n^3)$。

5 优化时间复杂度的两个技巧

上文提到的主要是整体上如何解决树拓扑序计数相关问题的方法，关系的是是否能给出一个多项式复杂度的做法。但有时可能得到的做法时间复杂度过高，比如多一个 n，此时可以用本节介绍的两个技巧降低时间复杂度。

[5]题目来源：原创

5.1 拉格朗日插值法

如果有一系列多项式 $f_1 \sim f_k$，现在希望计算 $\prod_{i=1}^{k} f_i$ 的各项系数，但直接乘复杂度太高。如果已知 $\prod_{i=1}^{k} f_i$ 的次数不超过 n，可以算出 $f_1 \sim f_k$ 在 $x = 0 \sim n$ 时的点值，将点值相乘后得到 $\prod_{i=1}^{k} f_i$ 在 $x = 0 \sim n$ 时的点值，再使用拉格朗日插值法从点值插回多项式。

在此给出插值公式：已知 n 次多项式的 $n+1$ 个点值 (x_i, y_i)，那么 $f(x) = \sum_{i=0}^{n} y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$。

本文中用到的插值法均为 $O(n^2)$ 的暴力插值，即先算出 $\prod_{i=0}^{n}(x - x_j)$，然后对所有 i 将总乘积 $O(n)$ 除以一次多项式 $(x - x_i)$，得到 $\prod_{j \neq i}(x - x_j)$ 各项系数后乘上 $\frac{y_i}{\prod_{j \neq i} x_i - x_j}$ 后加到答案多项式上。

5.2 允许不满足所有限制

先前我们研究的都是只要存在一条边 $u \to v$，那么 u 就必须出现于 v 之前的情况。如果我们改为求有多少排列 p 满足，删去原图的不超过 k 条边后其可以成为一个合法拓扑序呢？

可以做出以下两个观察：

- 对于任意排列 p，最少删边数量使其可以变为合法拓扑序，等于最少反转边数量（即选择最少的原图中的边将其反向），使其可以变成合法拓扑序。
- 对于任意排列 p，恰好存在一个边集 $E' \in E$，使得反转 E' 中所有边方向后，p 变为一个合法拓扑序。

  所以我们只需枚举原图所有大小不超过 k 的边子集 E'，反转 E' 中所有边方向后求拓扑序数量并加和即可得到答案。

例题六[6]：给定一张 n 个点的有向图，保证将其所有边视为无向边后形成一颗树。对 $k \in [0, n]$ 计算有多少排列 p 满足，删去原图不超过 k 条边后排列 p 可以变为一个合法拓扑序。$n \leq 500$。

考虑使用元 x 的指数记录反转了多少条边，即设出答案的 GF。

根据上面的分析，我们可以认为原图的每条边还有一条反向边，权值为 x，而原边权值为 1。我们需要对每对重边选出一条，总共选出 $n-1$ 条边，定义这种选择方案的权值是所有选出边权值的乘积再乘上被选出所有边构成图的拓扑序数量，只需算出所有选择方案权值之和即可得到答案（最后还要做个前缀和）。

[6]题目来源：原创

任意定根 r，在从一对重边中选出一条后，如果选出的边的方向并非叶向，就使用容斥原理将这条边拆开。类似有向树的拓扑序计数的方法进行 DP 即可，但此时 DP 数组的每一位都是多项式，计算多项式乘法需要消耗额外的时间，即使允许 NTT 时间复杂度仍为 $O(n^3 \log n)$。

注意到这些多项式次数都不超过 n，于是使用拉格朗日插值法维护 $n+1$ 个点值最后插值出答案多项式即可做到 $O(n^3)$。

5.3 更复杂的值域线与插值法

事实上在枚举值域线时可能需要枚举不止一条，插值法也可以应用得更灵活。

例题七[7]：**给定一颗 n 个点，以 1 为根的叶向树，和三个长度均为 n 的权值序列 b,c,d。定义一个合法拓扑序 a 的权值 $f(a)$ 为 $\sum_{i=1}^{n}\sum_{j=1}^{n} c_i d_j b_{|a_i-a_j|}$，对所有不同的 a，求 $f(a)$ 之和。合法拓扑序的定义是长度为 n 的排列满足 $\forall 2 \le u \le n, a_u > a_{fa_u}$，并且认为 $b_0 = 0$。$n \le 500$。**

首先略微调整一下 f，记 $F(a) = \sum_{i=1}^{n}\sum_{j=1}^{n} [a_i < a_j] c_i d_j b_{a_j - a_i}$，那么只要能计算 $F(a)$ 之和，交换 c,d 序列再求一次 $F(a)$ 之和即可得到原问题的答案。

这里的贡献比例题五的 $[a_i < a_j](a_j - a_i)$ 更加复杂，是 $[a_i < a_j] b_{a_j - a_i}$，枚举一层值域线难以处理，所以考虑枚举两层值域线 V_1, V_2 其中 $V_1 \le V_2$，限制 $a_i \le V_1, a_j > V_2$。还是将每个点 u 按照 a_u 涂色，如果 $a_u \le V_1$ 将其涂为白色，如果 $V_1 < a_u \le V_2$ 将其涂为灰色，如果 $a_u > V_2$ 将其涂为黑色。那么限制是 i 点必须为白色，j 点必须为黑色。

类似地，可以得到当一种合法涂色方案（即对于每条边，父亲的颜色总是不深于孩子的）被确定时，对应的拓扑序数量是 $(V_1!)(V_2 - V_1)!(n - V_2)! \prod_{i=1}^{n} \frac{1}{siz'_i}$，$V_1, V_2$ 可以看作是对白色、灰色点的计数。

仍然 DP，设 f_u 为 u 为黑色，u 子树内所有点 v 的 $\frac{1}{siz_v}$ 乘积；$g_{u,i}$ 为 u 为灰色，对 u 子树内所有点涂色，恰有 i 个灰点的所有合法涂色方案的 $\prod_{\text{u is ancestor of v}} \frac{1}{siz'_v}$ 之和；$h_{u,i,j}$ 为 u 为白色，对 u 子树内所有点涂色，恰有 i 个白点，j 个灰点的所有合法涂色方案的 $\prod_{\text{u is ancestor of v}} \frac{1}{siz'_v}$ 之和。这三个东西的转移都是先背包合并，再考虑自身的 $\frac{1}{siz'_u}$ 的影响。

不过我们还要从黑点中选出一个点作为 j，权值乘上 d_j；从白点中选出一个点作为 i，权值乘上 c_i。所以对 f,h 要额外加一维表示是否已经选出了对应的 j/i。注意这里的 i,j 的定义不是上一段 DP 数组中的，而是被限制涂为白色的点和被限制涂为黑色的点。

转移要做二维背包合并，时间复杂度为 $O(n^4)$，不可接受。可以发现对于 h，j 这一维的值是多少不影响转移过程。所以我们可以先 $O(n^2)$ 算出 f,g。然后对于 h，将答案视作关于 j 的 n 次多项式，只维护 $j \in [0,n]$ 时多项式的点值，最后使用拉格朗日插值法插出多项

[7]题目来源：原创

式。这样 DP 部分和插值部分时间复杂度均为 $O(n^3)$。

这样就对所有 $V_1 \le V_2$ 算出了对所有拓扑序 a，$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i d_j [a_i \le V_1][a_j > V_2]$ 之和。进行一次二维差分即可对所有 $V_1 \le V_2$ 得到 $\sum_{i=1}^{n}\sum_{j=1}^{n} c_i d_j [a_i = V_1][a_j = V_2 + 1]$ 之和，从而可以计算答案。并且可以发现我们事实上求出了比要求的答案更多的信息，即我们对所有值的二元组 (i, j) 算出了，对所有合法拓扑序 a，$a_p = i$ 的点的权值 c_p 乘上 $a_q = j$ 的点的权值 d_q 之积的和。

时间复杂度为 $O(n^3)$，空间复杂度为 $O(n^2)$。

事实上使用 3.3 的做法取 $k = 2$ 并抛弃掉一些无用信息也可以做到 $O(n^3)$ 时间，但实现起来可能更为复杂。

5.4 信息共用

在进行多次 DP 时，一部分信息可能被重复使用，那么事实上可以只计算这些信息一次从而减少消耗的时间。

事实上在 3.2 的解法中就用到了这种思想：在 $dp_{i,p}$ 的定义中，无论 x 是 p 子树内的哪个点，$dp_{i,p}$ 都是一致的，不必重复计算。这是非常自然的，因为无论 x 是哪个点，要将它涂黑都必须将从根到 p 这条链上的所有点涂黑，这一公共部分没有差别。

在此我们给出另一使用这一思想可以降低时间复杂度的例题：

例题八[8]：给定一颗 n 个点，以 1 为根的叶向树。对所有 $1 \le u < v \le n$ 求出 $f(u,v)$ 表示所有合法拓扑序 a 的 $|a_u - a_v|$ 之和。合法拓扑序的定义是所有是长度为 n 的排列满足 $\forall 2 \le u \le n, a_u > a_{fa_u}$。$n \le 500$。

还是像例题五一样枚举值域线，那么要对所有 $p \ne q$ 算出钦定 p 必须是白色点，q 必须是黑色点的所有染色方案权值和。

直接枚举 q，$O(n^2)$ DP 算出所有 $g_{u,i}$ 表示对 u 子树内所有点染色，构成合法染色方案，一共有 i 个白色点，并且 q 点如果在子树内则必须为黑色的所有染色方案权值和（权值是子树内所有 $\frac{1}{siz'_u}$ 的乘积）。这样事实上压缩表示了 u 为黑色或白色的两种信息，$i > 0$ 时 u 为白色，$i = 0$ 时 u 为黑色。

接着可以考虑自上至下地 DP，设 $dp_{u,i}$ 表示对 u 子树**外**所有点染色，构成合法染色方案，如果 q 在子树外则必须为黑色，已知 u 子树内有 i 个白色点的所有染色方案权值和（这里权值是子树外所有 $\frac{1}{siz'_u}$ 的乘积再乘上两种颜色点数量阶乘之积），这里只需考虑 $i > 0$ 的情况。

这样如果能 $O(n^2)$ 求出 dp，钦定 p 白色，q 黑色的所有方案权值和就是 $\sum_{i=1}^{siz_p} dp_{p,i} g_{p,i}$。

考虑如何求出 dp，首先 $\forall 1 \le i \le n, dp_{1,i} = i!(n-i)!$。对于 $u > 1$，$dp_{u,*}$ 应该从 $dp_{fa_u,*}$

[8]题目来源：原创

先在第二维上执行 $\frac{dp_{fa_u,i}}{i} \to dp'_{fa_u,i-1}$，这代表考虑 fa_u 的贡献并去掉其的影响。接着应该是将 $dp'_{fa_u,*}$ 在第二维上与 fa_u 除 u 以外的其他孩子 $v_1 \sim v_k$ 的 $g_{v_i,*}$ 逐个进行减法卷积，因为从 fa_u 换到 u，u 的其他孩子也由子树内变为子树外，需要考虑它们的贡献。

从 $dp \to dp'$ 是简单的，对于剩下的部分，相当于给出 k 个多项式 $f_1 \sim f_k$ 以及大多项式 F，$\deg(F) = \sum_{i=1}^{k} \deg(f_i)$，要对所有 $1 \le i \le k$ 算出 $subconv(F, \prod_{j \ne i} f_j)$ 的 $0 \sim \deg(f_i)$ 次项系数，$subconv(P, Q)$ 表示对 P 和 Q 执行减法卷积。

考虑对 $1 \le i \le k$ 算出 $prd_i = \prod_{j=i}^{k} f_j$，那么对于 $i = 1$ 结果就是 $subconv(F, prd_2)$。接下来要计算的所有信息都需要除去 f_1，所以可以直接令 $F \leftarrow subconv(F, f_1)$ 然后不管 f_1 这一项，这时只需保留 F 的 $0 \sim \sum_{j=2}^{k} \deg(f_j)$ 项系数即可。接下来 $i = 2$ 的结果就是 $subconv(F, prd_3)$，以此类推。

这样做的时间复杂度为 $O(\deg^2(F) - \sum_{i=1}^{k} \deg^2(f_i))$，根据树形背包的时间复杂度分析整体是 $O(n^2)$ 的。

当 q 固定时计算一轮的复杂度为 $O(n^2)$，故总时间复杂度为 $O(n^3)$。

例题八如果直接应用例题五的解法，时间复杂度为 $O(n^4)$。为什么我们能降低一个 n 的复杂度？用例题五的做法相当于要进行 n^2 轮 DP，但每轮 DP 的不同点其实只在于有着不同的 p, q，要限定 p 为白色点，q 为黑色点。所以每轮 DP 应当是 “比较相似” 的，可以公用一些信息。于是我们考虑固定一个 q，当 q 固定时不同点就只在于 p 了。但如果自下至上地 DP，信息仍然有很大差异。转换思路考虑从上至下地 DP，只要 p 在 u 子树内，那么对 u 子树外所有点染色的 DP 信息就应是完全一样的，可以共用这部分信息从而降低时间复杂度。

当然从上至下 DP 又会引出要除去其他信息可能时间复杂度不正确的问题。为此我们将一部分要除去的信息合并，然后枚举各个孩子，一边计算转移给这个孩子的信息，一边除去这个孩子的信息为接下来的计算作准备。

事实上从转置原理的角度来看这样的方法是非常自然的，因为从上至下 DP 可以看作是对经典的从下至上 DP 进行了转置，这里我们描述的过程就是转置了从下至上 DP 时每次考虑合并一个孩子信息的过程。

6 利用优秀限制方向解决问题

即使有向图的结构比有向树更复杂，如果限制方向具有优秀的性质，使得可以以较为明确的顺序计算各个部分的拓扑序信息再对信息进行合并，那么也可以在较为优秀的复杂度内解决问题。

本节将给出一道利用优秀限制方向解决问题的例题，并将一道原本标算为 $O(n^3)$ 的题目通过转化到该例题将时间复杂度优化到 $O(n^2)$。

6.1 叶向边仙人掌拓扑序计数

边仙人掌的定义是无重边无自环且每条边至多属于一个简单环（不经过重复点的环）的连通图。而叶向边仙人掌类似叶向树，代表对边仙人掌的每条边定向，变为有向边仙人掌，但存在一个点 r 使得边的方向总是从 dis 较小的点指向 dis 较大的点的情况，请看例题：

例题九[9]：给定一个 n 个点 m 条边的边仙人掌以及根 r。认为图是单位边权的，计算出图上每个点 u 到 r 的最短路 dis_u。接下来将每条边 (u,v) 进行定向，由 dis 小的一方指向 dis 大的一方。保证不存在一条边连接的两个点 dis 相同。请问定向后的图有多少拓扑序？ $n \le 5000, m \le 2(n-1)$。

这个问题类似叶向树，考虑改进叶向树的做法，但如果我们仍然设 dp_u 表示 u 直接或间接可达的所有点以及这些点之间连边构成的子图的拓扑序数量的话，会遇到一个问题：设想一个大小为 4 的环，假设 $r=1$，$1 \to 2, 1 \to 3, 2 \to 4, 3 \to 4$ 有边，并且 4 还指向了图的另外一些部分（如下图），此时 dp_3 的信息包含了 $3,4,5,6,7$ 两个点，dp_2 的信息包含了 $2,4,5,6,7$ 两个点，如果希望通过合并 dp_2, dp_3 来得到 dp_1 的话，点 $4,5,6,7$ 同时在两边出现，不能再用先前归并序列的角度来考虑问题了。

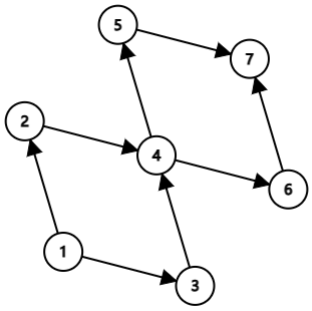

图 2: 一个上述想法不可行的例子

不过可以注意到，事实上这样做重复出现的点一定是一个环上唯一没有出度的点及其可达的点，所以对这个情况特殊处理应该还是比较简单的。

处理仙人掌的另一个问题是转移顺序并不明确，一种方法是对仙人掌建圆方树从而变成顺序更明确的树，不过这里我们会介绍另一种更直观的 “收缩” 方法。

我们注意到，对于一个点集 S，如果对于任意 $u \in S$，其所有出边 $u \to v$ 都有 $v \in S$，且存在唯一的一个点 p 满足如果存在边 $q \to u$，要么 $q \in S$，要么 $q = p$，并且 p 直接或间接可达 S 集合所有点。那么可以删去 S 集合内所有点，记录这些点都 “跟随” 于 p 之后，在此后的计数过程中时，一旦考虑到 p，就立即考虑如何排列 “跟随” 其的这些点，将方案数乘

[9]题目来源：校内模拟赛题

上点集 S 的拓扑序数量以及从此后拓扑序列选一些空位给这些点的方案数。我们称一次这样删去点集 S 的操作为针对点集 S 的一次 “收缩”。

注意收缩掉点集 S 时，$u \in S$ 的点 u 可能也被一些点 “跟随”，我们认为 “跟随” 其的点此后转而 “跟随” p。

事实上 2.1 介绍的树拓扑序计数问题的经典做法，就可以看作是在不断收缩叶子，直到最后只剩下根。

对于叶向边仙人掌，一定可以通过以下两种操作 “收缩” 到只剩根：

- 选一个点 u 以及其一条出边 $u \to v$，满足 v 只与 u 相连，收缩 v。称这种操作是收缩叶子。
- 选一个点 u，u 是某个包含其的简单环 C 中 dis 最小的点，$\forall v \in C \backslash \{u\}$，所有与 v 相连的边都在环 C 上，那么收缩环 C 除 u 以外的所有点。称这种操作是收缩环。

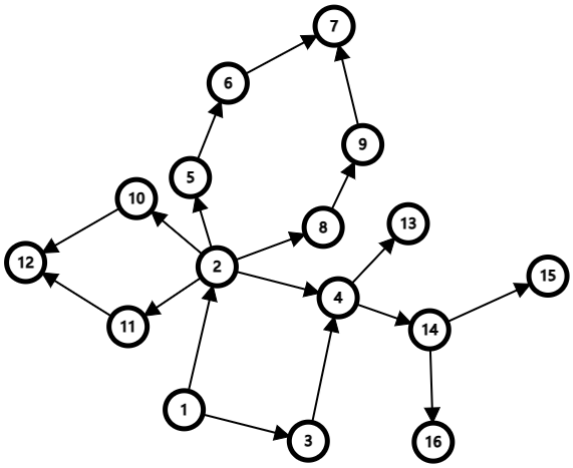

图 3: 一棵叶向边仙人掌

对于上图，一个可能的收缩过程是：选 $u = 14$，进行两次收缩叶子，分别收缩掉 15 和 16 两个点；选 $u = 4$，进行两次收缩叶子，分别收缩掉 13, 14 两个点；选 $u = 2$，进行收缩环，收缩掉 5, 6, 7, 8, 9 五个点；选 $u = 2$ 进行收缩环，收缩掉 10, 11, 12 三个点；选 $u = 1$ 进行收缩环，收缩掉 2, 3, 4 三个点。此时只剩下 1 这个根。

在收缩的过程中，我们维护 siz_u 表示 u 以及 “跟随” u 的点的数量，ans_u 表示 “跟随” u 的点以及 u 构成的点集的导出子图的拓扑序数量。初始显然所有 $siz_u = ans_u = 1$。

考察两种操作结束后对 siz_u 和 ans_u 的影响，可以发现 siz_u 总是会加上 S 集合内所有 siz 之和，主要需要考察的是对 ans 的影响。

- 对于收缩叶子操作，可以发现和叶向树等价，就相当于归并各个孩子对应子图的拓扑序列，ans_u 乘上 ans_v 以及 $\binom{siz_v + siz_u - 1}{siz_v}$。
- 对于收缩环操作，我们需要特殊处理。记环 C 的两端弧分别包含 $a_1, a_2...a_k$ 以及 $b_1, b_2...b_k$ 这些点，满足 $a_1 = b_1 = u, a_k = b_k$，设 $a_k = b_k = v$。考虑将拓扑序反过来，一开始只考察 v 以及" 跟随 " 于 v 的点怎么排，维护指针 $i = j = k$。每次从 i, j 中选一个，要么将 a_{i-1} 加在拓扑序最前面，然后在拓扑序上插入所有跟随其的点，要么选 b_{j-1}，选完后对应指针减 1。注意只有 $i = j = 2$ 时才能加入 u，并且加入 u 后立即停止。这里也需要一个 DP，记 $dp_{i,j}$ 为 $a_i \sim a_k, b_j \sim b_k$ 这些点，以及跟随这些点的点形成点集的导出子图的拓扑序数量，每次要么加入 a_{i-1} 以及" 跟随 " 其的点转移到 $dp_{i-1,j}$，要么加入 b_{j-1} 转移到 $dp_{i,j-1}$。转移时也需要乘分配位置的组合数。进行一轮这样 DP 的复杂度是 $O(k^2)$ 的。

收缩到只剩根 r 时结束，复杂度实际上是该边仙人掌所有简单环长的平方和，而因为每条边至多属于一个简单环，这不超过 m^2。而边仙人掌事实上必定满足 $m \le 2(n-1)$，所以复杂度事实上是 $O(n^2)$ 的。

6.2 利用叶向边仙人掌的做法解决另一问题

例题十[10]：给定一颗 n 个点以 1 为根的叶向树。定义一个排列 a 的权值 $f(a)$ 为 $\sum_{i=1}^{n-1} |a_i^{-1} - a_{i+1}^{-1}|$，求原图所有合法拓扑序的权值之和。合法拓扑序的定义是所有是长度为 n 的排列 a 满足 $\forall 2 \le u \le n, a_u > a_{fa_u}$。$n \le 5000$。

这是例题五定义权值时对排列求逆的形式，标算做法类似例题五的做法，是枚举值域线，最终时间复杂度为 $O(n^3)$，这里我们给出一种截然不同的 $O(n^2)$ 做法。

权值是 $|a_i^{-1} - a_{i+1}^{-1}|$，即对所有 $i \in [1, n-1]$，计算值 i 所在点的编号到值 $i+1$ 所在点的编号之差，然后取绝对值再求和。

如果我们能对所有点对 (u, v)，算出 u 拓扑序上是 v 前驱的方案数 $res(u, v)$，那么原问题就是求 $\sum_{u \ne v} res(u, v)|u - v|$。

我们根据 u, v 有无祖先后代关系分类讨论：

较为简单的情形是 u, v 有祖先后代关系，那么当且仅当 u 是 v 父亲时 $res(u, v)$ 才不为 0。此时因为拓扑序上遇到 u 就必须立即选 v，所以可以删去 v，将所有 v 的孩子接到 u 上。考虑这样改造后的 $n-1$ 个点的树的合法拓扑序，显然通过将 u 替换为 u, v 可以映射到原树的 u 必须是 v 前驱的合法拓扑序，并且构成双射。改造后的树的拓扑序是易于求得的，因为对 siz 序列的影响仅仅只有 u 到根的链上每个点 siz 减小 1，以及删去了 siz_v。

[10]题目来源：https://qoj.ac/problem/3843

更为复杂的情形是 u,v 没有祖先后代关系。我们连接边 $fa_v \to u$，并删去 v 点，将 v 的所有孩子接到 u 上。这样改造后的 $n-1$ 个点的图的合法拓扑序仍然与原树的 u 必须是 v 前驱的合法拓扑序构成双射，但问题在于因为多连了一条边，此时得到的不是树了。

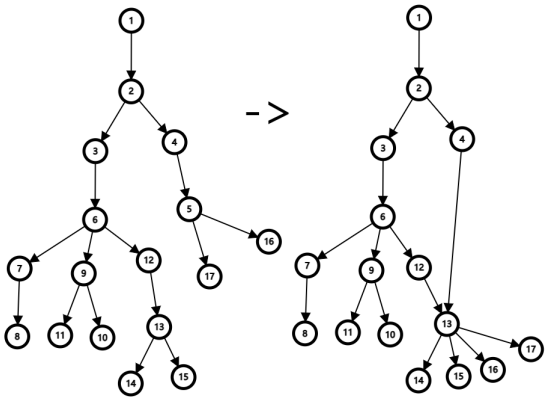

图 4: 当 $u=13, v=5$ 时对图进行的调整

不过可以注意到此时图的形态可以视为上文中提到的叶向边仙人掌，因为虽然此时环上边的方向不一定是按照到根的距离定向，但仍然保持是以环上距离根最近的点作为分界点，一段弧上的边为顺时针方向，一段弧上的边是逆时针方向，并且最近的点可以到达环上所有点。而且还具有更好的性质：一定只有一个环。

设 $w=\mathrm{lca}(u,v)$，环距离根最近的点一定是 w，从 w 向上的部分一定只有树边。所以如果能算出 w 下面部分的拓扑序数量，调整到整张图的拓扑序数量是简单的，只需不断进行归并标号即可，可知整张图的拓扑序数量一定是 w 下面部分的乘上只与 w 子树外部分相关的一个系数 prd_w。

一个问题在于，在叶向边仙人掌一节中我们的做法是从环上距离根 dis 较大的部分向 dis 较小的部分 DP 的（也就是环上节点是按照拓扑序从大往小加入的），而现在如果我们想 $O(n^2)$ 求出所有 $res(u,v)$，显然需要设计一个从环上 dis 较小部分向 dis 较大部分 DP 的算法。

而之所以之前我们选择了按照拓扑序从大到小加入的顺序，是因为加入一个点时需要考虑“跟随”其的所有点，按照拓扑序从大到小的顺序便于我们将这些“跟随”点与已被考虑的点进行归并。

那么反转这个过程，按照拓扑序从小到大的顺序加入环上所有节点，就应该加入一个环上节点时从已经归并好的序列中拆分出所有这个点的“跟随”节点。即假设拓扑序有 s 个空位，初始 $s=siz_w-1$，每次加入环上一个节点 u，并且“跟随”其的有 k 个节点的话，首先 u 一定占据最前面的空位，然后要从剩下 $s-1$ 个空位中选出 k 个分配给所有“跟随”其的节点，然后空位减少到 $s-1-k$ 个。

具体来说，我们这样进行 DP：设 $dp_{p,q}$ 表示两段环上可以加入的下一个点分别是 p,q

的方案数，当然只考虑 w 向下的部分。两种方案被认为是不同的当且仅当 p,q 直接或间接可达的点在拓扑序上占据位置构成的集合不同，或者存在 p,q 均不可达的点在拓扑序上占据的位置不同。

边界是 p 为 $w \to u$ 在原树路径上第二个点，q 为 $w \to v$ 在原树路径上第二个点，此时需要给 w 除 p,q 以外所有孩子分配空位。设 ord_u 为原树 u 子树拓扑序数量，w 除 p,q 有孩子 $v_1 \sim v_k$，那么 $dp_{p,q} = \prod_{i=1}^{k} ord_{v_i} \binom{siz_w - 2}{siz_{v_1}, siz_{v_2} \ldots siz_{v_k}, siz_p + siz_q - 1}$。

对于一般的转移，如果 $fa_p \neq w$，那么 $dp_{p,q}$ 可能由 $dp_{fa_p,q}$ 通过在环上加入 fa_p 来得到，设 fa_p 除了 p 有孩子 $v_1 \sim v_k$，那么 $dp_{p,q}$ 加上 $dp_{fa_p,q} \prod_{i=1}^{k} ord_{v_i} \binom{siz_{fa_p} + siz_q - 2}{siz_{v_1}, siz_{v_2} \ldots siz_{v_k}, siz_p + siz_q - 1}$，即给孩子 $v_1 \sim v_k$ 分配空位。如果 $fa_q \neq w$，也有对称的转移。

考虑最后如何计算答案，显然只需用 $dp_{u,u}$ 的值，再给 u,v 所有孩子分配空位即可（因为 v 的孩子被转接给 u 了），最后要乘上系数 prd_w 。

这样就顺利得到了环上节点按照拓扑序从小到大顺序加入的 DP 做法。虽然对于不同的 u,v，环的形态有所不同。但当 p,q 确定时，无论怎么选 u,v，只要 p 是 u 的祖先，q 是 v 的祖先，$dp_{p,q}$ 的值都一致，所以可以共用 DP 信息。唯一需要稍作修改的是这样 $dp_{u,u}$ 信息可能不存在，要用 $dp_{u,v}$。

进行一些预处理后转移都可以做到 $O(1)$，因而时间复杂度为 $O(n^2)$。

7 更为精巧地应用容斥原理

本节内容是第四节关于容斥原理部分的推进，通过更为精巧地应用容斥原理，我们可以对一般有向边仙人掌以及有向广义串并联图进行拓扑序计数。

7.1 一般有向边仙人掌

即只保证将所有有向边变为无向边后，图是一个边仙人掌。

例题十一[11]：给定一张 n 个点 m 条边的有向图，保证将所有有向边视为无向边后，图是一个边仙人掌。请问原图有多少拓扑序？$n \leq 500$，$m \leq 2(n-1)$。

叶向边仙人掌中，边的方向有很好的性质，便于我们进行 “收缩” 操作。一般有向边仙人掌则失去了这个性质，不过类比叶向树到一般有向树的拓展，我们应该仍能用容斥原理来解决该问题。

借助容斥原理，我们可以对每条边的方向设定一个预期，如果真实方向符合预期，那么就什么都不干。否则，可以把这条方向不符预期的边，拆成一条无限制边和符合预期的边（并带有一定容斥系数）。

[11]题目来源：原创

任意定根 r，我们可以把仙人掌上的边分为两类：在环上的边，称为环边；不在任意环上的边，称为树边。

还是求出 r 到每个点 u 的最短路 dis_u，我们这样设定每条边的预期：

- 对于树边，预期方向是两个点中 dis 较小的指向较大的。
- 对于环边，我们依次考虑每个简单环 C，显然这样每条环边会被考虑恰好一次。记该环上 dis 最小的点是 u，从 u 开始顺时针考虑环上的每条边，一开始我们设定预期方向是顺时针方向。每考虑一条边，一定会拆出一条顺时针方向的边，可能还有一条无限制边，我们从这两条拆出的边（可能只有一条）中选择一条作为本组方案内这条边的限制类型。**一旦我们选出了一条无限制边，从环上的下一条边开始，以预期方向为逆时针方向拆边，**直到所有边都被拆开。

拆完边后，最后将所有拆边方案得到的图的拓扑序数乘上对应容斥系数加和即可。

对于环上的边，我们实际上动态调整了我们的预期方向。这样做有什么好处呢？注意到如果一组方案中存在一个简单环，环上的所有边均为顺时针方向，那么显然对应图拓扑序数为 0。否则这组方案中所有环都存在一条分界边 E，E 本身是无限制边，在 E 前面的边均为顺时针限制边，在 E 后面的边要么是无限制边，要么是逆时针限制边。

可以发现，这样一来，只考虑环上边的话，环上所有点的入度都不超过 1，且 u 入度必定为 0。并且根据对树边的容斥，一个点至多被一条树边指向。再根据仙人掌图的结构，一个点 p 如果可能被树边指向（即从 r 到 p 的所有最短路最后一条边都是树边），那么包含 p 的所有环，p 都是 dis 最小的点，从而不可能被环边指向。

综合以上几点，可以发现这样容斥后每个点入度不超过 1，并且不存在环，于是我们将问题转化为了叶向森林拓扑序计数。

考虑如何实现该容斥，我们仍然可以借助叶向边仙人掌的 “收缩” 顺序来进行转移，为此我们稍微修改一下 “收缩” 的定义：

对于一个点集 S，如果存在一个唯一的点 p，使得对于任意 $u \in S$，所有与其相关的边 $u \to v$ 或 $v \to u$，满足要么 $v \in S$，要么 $v = p$。那么可以应用容斥原理，将所有与任意 $u \in S$ 相关的边 $u \to v$ 或 $v \to u$ 最终容斥为 p 是一个根节点的叶向森林。删去 S 集合内所有点，记录这些点都 “跟随” 于 p 之后。我们称一次这样删去点集 S 的操作为针对点集 S 的一次 “收缩”。

仍需动态维护 siz_u，修改 $ans_{u,i}$ 的定义为所有容斥方案，使得 “跟随” 于 u 的点集 S_u 构成的叶向森林中，u 所在树有 i 个点的 $\dfrac{\text{该方案容斥系数}}{\prod\limits_{v \in S_u} siz'_v}$ 之和，siz'_v 代表该容斥方案中 v 的子树大小。

仍然考察收缩叶子操作与收缩环操作对 ans 的影响：

- 对于收缩叶子操作，首先此时 v 在容斥方案中的子树大小一定已经确定，于是给 $ans_{v,i}$ 先乘上 $\frac{1}{i}$。然后考察 u,v 之间连边的方向，如果本就是 $u \to v$ 那直接背包合并即可；否则要拆成容斥系数为 -1 的 $u \to v$ 和一条无限制。这一步的复杂度是 $O(siz_u siz_v)$。

- 对于收缩环操作，在这一轮操作中所有 $\forall v \in C \backslash \{u\}$，$siz'_v$ 被确定，要乘上这些 $\frac{1}{siz'_v}$。记这个简单环上的点按顺序是 $a_1, a_2 ... a_k, a_{k+1}$，其中 $a_1 = a_{k+1} = u$。枚举分界边 E 是 (a_i, a_{i+1}) 间的连边，显然只有这条边方向是 $a_{i+1} \to a_i$ 才可行。枚举 j 从 $i+1$ 到 k，维护 f_s 表示所有容斥方案（只考虑 a_{i+1} 到 a_j 间所有边如何容斥），a_j 子树大小为 s 的权值和，每次先合并上 a_j 的其他孩子，即与 $ans_{a_j,*}$ 背包合并，然后将 f_s 乘上 $\frac{1}{s}$。接着考虑 a_j, a_{j+1} 间的连边在容斥方案中选择哪种方向，对 f 进行对应修改。同理再枚举 j 从 i 到 1，维护 g_s 表示所有容斥方案（只考虑所有 a_j 到 a_i 间所有边如何容斥），a_j 子树大小为 s 的权值和，转移也是类似的，但注意边必须选顺时针方向，并且最后 $\frac{1}{siz_u}$ 不用乘。最后对 f, g 背包合并得到 u 真实的子树大小对应的 DP 值，对所有 i 加和得到新的 ans_u。

最后收缩到只剩点 r 时停止，最终答案是所有 $\frac{ans_{r,i}}{i}$ 之和乘上 $n!$。

分析时间复杂度，定义一个局面的势能是所有点 siz_u^2 之和，也就是对每个点，“跟随”其的点数量 +1 的平方和。

初始局面势能为 n，终止局面势能为 n^2。收缩叶子操作消耗 $O(siz_u siz_v)$ 时间且使势能上升 $2siz_u siz_v$，收缩环操作消耗 $O(|S|((\sum_{p\in S} siz_p)^2 - \sum_{p\in S} siz_p^2))$ 时间，且使势能上升 $(\sum_{p\in S} siz_p)^2 - \sum_{p\in S} siz_p^2$，因此时间复杂度为 $O(n^3)$。

7.2 有向广义串并联图

广义串并联图是对于任意四个节点都不存在 6 条两两没有公共边的路径连接这 4 个节点中的每一对节点的无向连通图，不过在本文中我们只需要知道这样的图一定可以通过删一度点操作、缩二度点操作、叠合重边操作转化为只剩一个点的图即可。这一点的证明及更多关于广义串并联图的论文可见参考文献 [1]。

具体描述一下三种操作，分别是：

- 删一度点操作：如果存在点 u 度数恰为 1，删去它以及与它相邻的边。

- 缩二度点操作：如果存在点 u 度数恰为 2，设它的两个邻居是 v, w，删去边 $(u,v),(u,w)$，加入边 (v,w)，然后删去点 u。

- 叠合重边操作：如果存在两条边 (u,v)，将这对重边叠成一条边。

可以发现前文提到的树、边仙人掌都是广义串并联图，但存在广义串并联图既不是树也不是仙人掌，所以这个问题显然更复杂。

接下来我们研究一下如果有向图在将所有边视为无向边后是广义串并联图，该如何高效地拓扑序计数。

例题十二[12]：给定一张 n 个点 m 条边的有向图，保证将所有有向边视为无向边后，图是广义串并联图。请问原图有多少拓扑序？ $n \le 100, m \le 2(n-1)$。

方便起见，我们认为视为无向边后图是无重边无自环的。因为方向相同的重边只需保留一条，方向相反的重边和自环会导致答案直接为 0。

这样一来，叠合重边操作一定立即发生于缩二度点操作之后，只有在 v,w 之间本来就有边时需要进行一次叠合重边操作。

我们还是希望应用容斥原理将问题转化到叶向森林拓扑序计数，也即需要通过容斥保证每个点的入度不超过 1，并且要确保不存在环。考虑如何设定容斥的预期方向，一个简单的想法是当要删去点 u 时，确定其所有邻边的预期方向为指向 u。这样每个点只有在被删去时才有可能增加入度，所以入度总是 ≤ 2，但我们希望的是入度 ≤ 1。

显然只有缩二度点时才会出现入度 $= 2$ 的情况，此时是容斥后两个邻居 v,w 均存在 $v \to u, w \to u$ 的边。但我们注意到如果此时 v,w 之间存在边，比方说存在 $v \to w$ 的边，那么就可以删去 $v \to u$ 的边，因为只要有 $v \to w, w \to u$ 的边，$v \to u$ 的限制是自然成立的，这样一来 u 的入度就变为 1 了。

但本来 v,w 之间可能不存在边，为了解决这一情况，我们可以创造两个子问题，这两个子问题中分别有 $v \to w$ 的边和 $w \to v$ 的边，只要分别解决两个子问题然后将答案加起来就得到了 v,w 之间没有边的初始情况的答案了。

考虑如何将上述分析写成 DP 的形式：首先任何时刻有 m 条边的图事实上都存在 2^m 个子问题，即 m 条边每条的定向选择哪一种导出的 2^m 种情况。对每条边维护 $dp_{id,0/1,i,j}$ 表示若将第 id 条边 $= (u,v)$ 定向为正/反向，在容斥拆边过程中，使得最终叶向森林 u 子树大小恰增加 i，v 子树大小恰增加 j 的所有方案的权值和。注意这里的 i,j 都是这条边带来的影响，对应的点事实上是在先前的三种操作过程中被收缩到这条边上的点。再维护 $val_{u,i}$ 表示不考虑所有边带来的子树大小增量，只考虑删一度点操作对 u 子树大小影响的话，容斥拆边过程中 u 子树大小为 i 的所有方案的权值和。

当然也需要维护 dp_{id} 的 $i+j$ 最大值 sz_{id}，可以发现这个值就是 (u,v) 这条边上收缩了多少点；以及 val_u 的 i 最大值 Sz_u。

初始所有 $val_{u,1} = 1$，代表每个点自身，$Sz_u = 1$；所有边都是 $dp_{i,0,0,0} = 1$，即边的初始方向有 1 的方案，$sz_i = 0$。我们永远能删去任何一个度数 ≤ 2 的点：

- 删孤立点：只需将 $\sum_{i=1}^{Sz_u} \frac{val_{u,i}}{i}$ 直接乘入答案即可。

[12]题目来源：原创

- 删一度点操作：设 u 唯一的邻居是 v，(u,v) 的编号是 id。首先应该在删去点 u 时结算 $\frac{1}{siz_u}$ 的权值，u 的子树大小 siz_u 应当是 (v,u) 这条边的影响加上 val_u 本身的值，这里需要将 $dp_{id,0/1}$ 在 i 这一维上与 val_u 做背包合并。然后考虑删去 u 可能使 v 的子树大小增加对 val_v 产生影响，对于边方向是 $v \to u$ 的子问题无需容斥，是 v 的子树大小增加 u 的子树大小，而对于边方向是 $u \to v$ 的子问题需要拆成一个无限制（那对 v 的子树大小就没有影响了）和一个 $v \to u$。不过即使是拆成无限制，边对于 v 的子树大小增量此时也是要叠加到 val_v 上去的。这里先是要 $O(sz_{id}^2 Sz_u)$ 计算边和 val 对 u 子树大小增量的总和，然后要 $O(sz_{id}(sz_{id}+Sz_u)Sz_v)$ 将边对 v 的贡献，v 本来的贡献，以及可能的 u 在叶向树上是 v 儿子的贡献三者合并。最后 Sz_v 增加 $Sz_u + sz_{id}$。

- 缩二度点操作：设 u 的两个邻居是 v,w，(u,v) 的编号是 id_1，(u,w) 的编号是 id_2，(v,w) 的编号为 id_3。(v,w) 边此时可能不存在，需要额外新建。如果要新建，$dp_{id_3,0,0,0} = dp_{id_3,1,0,0} = 1$，且 $sz_{id_3} = 0$。类似地也需要在此时结算 $\frac{1}{siz_u}$ 的权值，u 的子树大小此时要考虑 $(u,v),(u,w)$ 两条边以及 val_u 本身。考虑先将 $dp_{id_1,0/1}$ 在 i 这一维上与 val_u 背包合并将 val_u 的信息合并到 dp_{id_1} 上，这消耗 $O(Sz_u sz_{id_1}^2)$ 时间，注意 0/1 两种方向都要合并，此时我们可以认为 sz_{id_1} 增加了 Sz_u。这样就只需考虑 $(u,v),(u,w)$ 两条边，u 的子树大小直接是两条边对 u 子树大小的增量之和。还需要考虑删去 id_1, id_2 两条边后它们的信息需要传递给 id_3，先固定 id_3 的方向，在这种方向前提下枚举 id_1, id_2 的 $2^2 = 4$ 种方向，先结算 $\frac{1}{siz_u}$ 的贡献，然后根据各自的方向进行容斥，得到容斥后这两条边总共给 v 子树大小增加 i'，w 子树大小增加 j' 的所有容斥方案的权值和，这一步消耗 $O(sz_{id_1}^2 sz_{id_2}^2)$ 时间。最后将这部分信息与 id_3 原本的信息背包合并，这一步消耗 $O((sz_{id_1}+sz_{id_2})^2 sz_{id_3}^2)$ 时间。最终 sz_{id_3} 增加 $sz_{id_1} + sz_{id_2}$。

分析时间复杂度，还是设势能为 $\sum_{u=1}^{n} Sz_u^2 + \sum_{i=1}^{m} sz_i^2$，初始势能为 0，最终势能不超过 n^2。可以发现合并过程中消耗的时间要么是 $O(ne)$ 的，要么是 $O(e^2)$ 的，其中 e 是一轮合并上升的势能，所以时间复杂度是 $O(n^4)$ 的。

如果允许使用 NTT，可以将缩二度点时的复杂度降为 $O(sz_{id_1} sz_{id_2} n \log n)$ 和 $O((sz_{id_1} + sz_{id_2}) sz_{id_3} n \log n)$，从而做到 $O(n^3 \log n)$ 的时间复杂度。

8 结语

本文由浅至深地介绍了解决树拓扑序计数相关问题的各种方法，并通过十二道例题展现了这些方法的具体应用。这些方法与例题一部分是对前人的结果进行相对系统性的总结，一部分是笔者做出的全新的探索。但本文给出的做法与思路未必是完美的，可能仍有改进的空间；各种扩展例题之间可能也可以相互组合，创造出全新的题目。这些可能性还要等待后人来探索！

9 致谢

感谢中国计算机学会提供学习和交流的平台。
感谢国家集训队教练彭思进、杨耀良的指导。
感谢父母对我的培养和教育。
感谢学校的栽培，李建老师的教导和同学们的帮助。
感谢章绍嘉同学、孙恒喆同学与我进行交流讨论，并为本文审稿。
感谢吴泽岳同学为本文的编写提供技术支持。
感谢其它所有本文所参考过的资料的提供者。
感谢各位百忙之中抽出宝贵的时间来阅读本文。

10 参考文献

[1] 吴作同. 《《公园》命题报告》，IOI2019 中国国家队候选队员论文集。