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

2023.2

前言

叶向树拓扑序计数问题是算法竞赛中的经典问题，在此之上也已有许多拓展问题与方法，我在我的集训队论文中对此进行了一些总结和研究。

前言

叶向树拓扑序计数问题是算法竞赛中的经典问题，在此之上也已有许多拓展问题与方法，我在我的集训队论文中对此进行了一些总结和研究。

接下来将介绍三种拓展问题以及解决的方法，由于时间限制，默认大家都对叶向树拓扑序计数问题有基本了解。

问题

给定一棵 n 个点以 1 为根的叶向树和一个常数 k。你需要对所有长度为 k 的序列对 a, b 满足 a 递增，b 互不相同且 a, b 值域均为 $[1, n]$，算出有多少长度为 n 排列 p 满足 $\forall 1 \leq i \leq k, p_{a_i} = b_i$，且对叶向树的每条边 $u \to v$，$p_u < p_v$。答案模 998244353。

问题

给定一棵 n 个点以 1 为根的叶向树和一个常数 k。你需要对所有长度为 k 的序列对 a, b 满足 a 递增，b 互不相同且 a, b 值域均为 $[1, n]$，算出有多少长度为 n 排列 p 满足 $\forall 1 \leq i \leq k, p_{a_i} = b_i$，且对叶向树的每条边 $u \to v$，$p_u < p_v$。答案模 998244353。

$n^{2k} \leq 3 \times 10^7, k \leq 4$。

双射转化

考虑以下针对给定叶向树的过程：

双射转化

考虑以下针对给定叶向树的过程：

1. 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。

双射转化

考虑以下针对给定叶向树的过程：

1. 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。
2. 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 $a_i = p$。

双射转化

考虑以下针对给定叶向树的过程：

1. 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。
2. 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。

双射转化

看上去算重了，但其实可以观察到，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。因为拓扑序第 i 位是 u，代表过程中第 i 轮被涂黑的点是 u，在 u 父亲已经被涂黑的前提下，这充要于选择的白色点 v 在 u 子树内，这有 siz_u 种方案，而不同轮独立可以直接相乘。

双射转化

看上去算重了，但其实可以观察到，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。因为拓扑序第 i 位是 u，代表过程中第 i 轮被涂黑的点是 u，在 u 父亲已经被涂黑的前提下，这充要于选择的白色点 v 在 u 子树内，这有 siz_u 种方案，而不同轮独立可以直接相乘。

所以钦点 k 个点在拓扑序上的位置可以转化为在染色问题中钦点它们分别应当在哪一轮被涂黑。

双射转化

看上去算重了，但其实可以观察到，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。因为拓扑序第 i 位是 u，代表过程中第 i 轮被涂黑的点是 u，在 u 父亲已经被涂黑的前提下，这充要于选择的白色点 v 在 u 子树内，这有 siz_u 种方案，而不同轮独立可以直接相乘。

所以钦点 k 个点在拓扑序上的位置可以转化为在染色问题中钦点它们分别应当在哪一轮被涂黑。

这样转化的好处在于如果只关心 k 个点在哪一轮被涂黑，只需要知道这 k 个点中每个点，最浅的白色祖先是哪个即可。

DP 实现

设 $dp_{i,S,T}$ 表示已经完成了前 i 轮涂色，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，而 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，这样的涂色方案数。

DP 实现

设 $dp_{i,S,T}$ 表示已经完成了前 i 轮涂色，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，而 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，这样的涂色方案数。

转移考虑第 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。

DP 实现

设 $dp_{i,S,T}$ 表示已经完成了前 i 轮涂色，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，而 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，这样的涂色方案数。

转移考虑第 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 $O(n^{2k}poly(k))$ 的复杂度。

总结

事实上在 k 远小于 n 时这个做法差不多是达到了复杂度下界的，因为要计算 $\binom{n}{k} n^{\underline{k}}$ 个信息，而消耗的复杂度则是 $O(n^{2k}poly(k))$

总结

事实上在 k 远小于 n 时这个做法差不多是达到了复杂度下界的，因为要计算 $\binom{n}{k} n^{\underline{k}}$ 个信息，而消耗的复杂度则是 $O(n^{2k} poly(k))$

不过这个做法在 $k > 1$ 时实现起来可能比较痛苦。

方法的引入

树上问题，自下至上 DP，进行背包合并，这是非常常见的做法。往往能说明暴力背包合并的复杂度整体就是 $O(n^2)$ 的，与只合并一次背包的复杂度相同因此不会成为瓶颈。

方法的引入

树上问题，自下至上 DP，进行背包合并，这是非常常见的做法。往往能说明暴力背包合并的复杂度整体就是 $O(n^2)$ 的，与只合并一次背包的复杂度相同因此不会成为瓶颈。

但有时不得不自上至下 DP，然后需要做逆向树形背包合并（背包分裂？），此时复杂度就可能出现一些问题。

例题

给定一棵 n 个点，以 1 为根的叶向树。对所有 $1 \leq u < v \leq n$ 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 $\forall 2 \leq u \leq n, a_u > a_{fa_u}$。答案模 998244353。

例题

给定一棵 n 个点，以 1 为根的叶向树。对所有 $1 \leq u < v \leq n$ 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 $\forall 2 \leq u \leq n, a_u > a_{fa_u}$。答案模 998244353。

$n \leq 500$。

初步考虑解法

首先 $|x-y|$ 类贡献考虑枚举值域线拆成 $(\sum_V [x \le V][V < y]) + (\sum_V [y \le V][V < x])$。把所有 $a_u \le V$ 的点涂成白色，剩下的涂成黑色。考虑枚举一种黑白染色方案，那么不能有一条边 $u \to v$ 满足 u 黑色而 v 白色。只要满足这些边的限制，设一共涂了 k 个黑色点，该染色方案对应合法排列 a 的数量就是 $\frac{k!(n-k)!}{\prod_{i=1}^{n} siz'_i}$，其中 siz'_u 是 u 子树内与 u 同色点的数量。设这个对应排列数为该染色方案权值。

初步考虑解法

首先 $|x-y|$ 类贡献考虑枚举值域线拆成 $(\sum_V [x \le V][V < y]) + (\sum_V [y \le V][V < x])$。把所有 $a_u \le V$ 的点涂成白色，剩下的涂成黑色。考虑枚举一种黑白染色方案，那么不能有一条边 $u \to v$ 满足 u 黑色而 v 白色。只要满足这些边的限制，设一共涂了 k 个黑色点，该染色方案对应合法排列 a 的数量就是 $\frac{k!(n-k)!}{\prod_{i=1}^{n} siz'_i}$，其中 siz'_u 是 u 子树内与 u 同色点的数量。设这个对应排列数为该染色方案权值。

那么相当于需要对所有 $p \neq q$ 算出 p 为白色 q 为黑色的所有染色方案的权值和。

初步考虑解法

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，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。

初步考虑解法

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，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。

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，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。

初步考虑解法

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，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。

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，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 $\sum_{i=1}^{siz_p} dp_{p,i} g_{p,i}$。

初步考虑解法

考虑如何求出 dp，首先 $\forall 1 \le i \le n, dp_{1,i} = i!(n-i)!$。对于 $u > 1$，首先要去掉 fa_u 的影响，会导致 i 的项以 $\frac{1}{i}$ 的系数移到 $i-1$，这样得到 $dp'_{fa_u,*}$。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。

初步考虑解法

考虑如何求出 dp，首先 $\forall 1 \le i \le n, dp_{1,i} = i!(n-i)!$。对于 $u > 1$，首先要去掉 fa_u 的影响，会导致 i 的项以 $\frac{1}{i}$ 的系数移到 $i-1$，这样得到 $dp'_{fa_u,*}$。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。

此时就需要进行上文提到的逆向树形背包合并，怎么做复杂度才正确呢？

逆向树形背包合并的正确做法

这相当于给出 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。

逆向树形背包合并的正确做法

这相当于给出 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 $subconv(F, \prod_{j \neq i} f_j)$ 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。

例题的收尾与总结

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。

例题的收尾与总结

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。

所以例题可以 $O(n^3)$ 解决。

例题的收尾与总结

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。

所以例题可以 $O(n^3)$ 解决。

事实上从转置原理的角度来看，逆向树形背包合并这样的解决方法是非常自然的，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。

对更一般的图拓扑序计数

有向树拓扑序计数问题的做法是 well-known 的——通过容斥原理转化到一些叶向森林拓扑序计数的子问题，然后带上容斥系数把答案加起来。

对更一般的图拓扑序计数

有向树拓扑序计数问题的做法是 well-known 的——通过容斥原理转化到一些叶向森林拓扑序计数的子问题，然后带上容斥系数把答案加起来。

事实上只要更精巧地使用容斥原理，可以把有向边仙人掌拓扑序计数问题、甚至有向广义串并联图拓扑序计数问题也转化到一些叶向森林拓扑序计数的子问题。

有向边仙人掌拓扑序计数

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。求原图有多少拓扑序，答案模 998244353。

有向边仙人掌拓扑序计数

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。求原图有多少拓扑序，答案模 998244353。

$n \leq 500$，$m \leq 2(n-1)$。

容斥分析

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。

容斥分析

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。

通过对一条边的方向设定一个预期，可以保证在子问题中要么这条边不存在，那么该边的方向与预期一致。

容斥分析

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。

通过对一条边的方向设定一个预期，可以保证在子问题中要么这条边不存在，那么该边的方向与预期一致。

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：

1. 对于树边，预期方向是两个点中 dis 较小的指向较大的。

容斥分析

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。

通过对一条边的方向设定一个预期，可以保证在子问题中要么这条边不存在，那么该边的方向与预期一致。

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：

1. 对于树边，预期方向是两个点中 dis 较小的指向较大的。
2. 对于环边，依次考虑每个简单环 C，确定这个环上所有边在子问题中的方向。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。如果在新子问题中边是顺时针方向，预期方向不变；否则，从环上的下一条边开始，预期方向变为逆时针方向。

容斥分析

这里边的预期方向事实上与前面分解出的多个子问题中选择了哪个有关。可以看作是一开始只有一个子问题，所有边均未被拆开。每次新考虑一条边，如果是顺时针方向那该子问题不分裂；如果是逆时针方向，分裂到一个顺时针方向的子问题以及一个无限制的子问题，在无限制的子问题中预期方向被反转了。

容斥分析

这里边的预期方向事实上与前面分解出的多个子问题中选择了哪个有关。可以看作是一开始只有一个子问题，所有边均未被拆开。每次新考虑一条边，如果是顺时针方向那该子问题不分裂；如果是逆时针方向，分裂到一个顺时针方向的子问题以及一个无限制的子问题，在无限制的子问题中预期方向被反转了。

这样容斥的好处在于最终每个子问题每个点入度都 ≤ 1，这个通过对点和边进行一些分类即可证明。那么排除掉有环的情况，剩下的子问题就都是叶向森林了。

DP 实现

子问题中所有边当然都在原图中出现过。原图中所有树边在子问题中要么不存在要么是由距 r 较近的点指向较远的；所有环边在子问题中，与 dis 最小点 u 弱连通的部分也一定是由距 u 较近的点指向较远的（这里的距离是在子问题中的）。

DP 实现

子问题中所有边当然都在原图中出现过。原图中所有树边在子问题中要么不存在要么是由距 r 较近的点指向较远的；所有环边在子问题中，与 dis 最小点 u 弱连通的部分也一定是由距 u 较近的点指向较远的（这里的距离是在子问题中的）。

所以转移顺序也是比较明确的，对仙人掌建圆方树后自下至上 DP 即可，DP 的状态设计与叶向树问题是类似的。

DP 实现

子问题中所有边当然都在原图中出现过。原图中所有树边在子问题中要么不存在要么是由距 r 较近的点指向较远的；所有环边在子问题中，与 dis 最小点 u 弱连通的部分也一定是由距 u 较近的点指向较远的（这里的距离是在子问题中的）。

所以转移顺序也是比较明确的，对仙人掌建圆方树后自下至上 DP 即可，DP 的状态设计与叶向树问题是类似的。

更多 DP 细节在此略去，最终可以得到一个 $O(n^3)$ 的做法。

有向广义串并联图计数

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。求原图有多少拓扑序，答案模 998244353。

$n \leq 100$，$m \leq 2(n-1)$。

容斥分析

广义串并联图肯定要考虑其对应的三种操作了，不过事实上如果一开始图是简单图，叠合重边操作总是发生于一次缩二度点操作之后，可以一起考虑。

容斥分析

广义串并联图肯定要考虑其对应的三种操作了，不过事实上如果一开始图是简单图，叠合重边操作总是发生于一次缩二度点操作之后，可以一起考虑。

这样一来可以看作是要不断进行删 $0,1,2$ 度点的操作。

容斥分析

广义串并联图肯定要考虑其对应的三种操作了，不过事实上如果一开始图是简单图，叠合重边操作总是发生于一次缩二度点操作之后，可以一起考虑。

这样一来可以看作是要不断进行删 $0,1,2$ 度点的操作。

还是希望通过容斥分解到很多每个点入度 ≤ 1 的子问题，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。

容斥分析

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。

容斥分析

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，通过把两个子问题答案加和得到原问题答案。

容斥分析

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，通过把两个子问题答案加和得到原问题答案。

注意：该做法中容斥会创造子问题，拆 v, w 间的边也会创造子问题。

DP 实现

删 $0,1,2$ 度点会导致一些点获得子树大小增量，但增量对应的点已经不在当前的图上，需要记录这个增量信息。

DP 实现

删 $0,1,2$ 度点会导致一些点获得子树大小增量，但增量对应的点已经不在当前的图上，需要记录这个增量信息。

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。

DP 实现

删 $0,1,2$ 度点会导致一些点获得子树大小增量，但增量对应的点已经不在当前的图上，需要记录这个增量信息。

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。

记录 DP 信息 $dp_{id,0/1,i,j}$ 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。

DP 实现

删 $0,1,2$ 度点会导致一些点获得子树大小增量，但增量对应的点已经不在当前的图上，需要记录这个增量信息。

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。

记录 DP 信息 $dp_{id,0/1,i,j}$ 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。

还需记录 $val_{u,i}$ 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。

DP 实现

希望删除一个度数 ≤ 2 的点 u 时，要对涉及边选取的方向进行讨论，考虑涉及的点的 val 信息以及涉及边的 dp 信息，并结算这个点的 $\frac{1}{siz_u}$ 贡献。u 的子树大小增量也可能需要传递到某个与它有边的点上，这个增量可能记在边上，即更新 dp（当增量与某条边方向有关时），也可能记在点上，即更新 val（该增量与任何边方向都无关）。

DP 实现

希望删除一个度数 ≤ 2 的点 u 时，要对涉及边选取的方向进行讨论，考虑涉及的点的 val 信息以及涉及边的 dp 信息，并结算这个点的 $\frac{1}{siz_u}$ 贡献。u 的子树大小增量也可能需要传递到某个与它有边的点上，这个增量可能记在边上，即更新 dp（当增量与某条边方向有关时），也可能记在点上，即更新 val（该增量与任何边方向都无关）。

在转移过程中需要进行二维背包合并，对两维均暴力卷积最终做法是 $O(n^4)$ 的，对一维暴力卷积一维 FFT 可以做到 $O(n^3 \log n)$。

谢谢大家！