试题精讲及拓展

TQX

2023 年 11 月 6 日

吐槽

本场难度基本上对标 NOIP2021。
原因是在参与 NOIP2021 中深刻感受到了在思维题与大数据结构题之间取舍的痛苦。
T1 很简单，有一点易错点。
T2 很一眼，有一点细节。
T3 是大数据结构，但是容易想到，对标 NOIP2021 T4，但应该简单很多。
T4 比较思维，对标 NOIP2021 T3，但应该困难一点。

得分情况与吐槽

题解

subtask 1: 保证输入的字符串没有 ？

```
puts("1");
```

题解

subtask 1: 保证输入的字符串没有 ?

puts("1");

subtask 2:$n \leq 4, d \leq 4$

最直观的想法：大力 2^{nd} 枚举所有 ? 然后计算答案。最后去重。
大部分题目都会给这种部分分，写着写着就熟练了，不怕麻烦。

题解

subtask 1: 保证输入的字符串没有 ?

puts("1");

subtask 2:$n \leq 4, d \leq 4$

最直观的想法：大力 2^{nd} 枚举所有 ? 然后计算答案。最后去重。
大部分题目都会给这种部分分，写着写着就熟练了，不怕麻烦。

subtask 3:$d \leq 8$

不需要先枚举完再计算算式，可以边计算算式边枚举。
计算到 $c_{i-1}a_i$，不需要关心前面的 $i-1$ 个 a 具体是什么，只需要知道他们的结果就行了。
对于 $i = 1 \to n$，依次计算出前 i 个数字能得出的所有结果，从 $i \to i+1$ 只需要 2^{2d} 的枚举。
当然如果你知道动态规划的话，这就相当于设 $dp_{i,s}$ 表示前 i 个数能否得出 s，复杂度 $\mathcal{O}(n2^{2d})$。

题解

subtask 4:$d \leq 16$

理论上是轮廓线 DP。但轮廓线 DP 的思考难度严格大于这题正解的思考难度。
将上面 dp 每次考虑一个数改为按二进制位逐位考虑，同时维护每一位的最新结果。这样复杂度就是 $\mathcal{O}(nd2^d)$ 了。

题解

subtask 4:$d \leq 16$

理论上是轮廓线 DP。但轮廓线 DP 的思考难度严格大于这题正解的思考难度。
将上面 dp 每次考虑一个数改为按二进制位逐位考虑，同时维护每一位的最新结果。这样复杂度就是 $\mathcal{O}(nd2^d)$ 了。

subtask 5:$d \leq 32$

位运算的问题的最经典性质就是位运算可以拆开对每一个二进制位分别计算。想到这一点后，自然就能发现本题的答案对每一位是独立的。如果你打表发现答案总是 2^n，或许能帮助你想到这一点?
于是对每一位进行上面的 dp，设 $dp_{i,s}$ 表示前 i 个数能否得出 s，其中 $s \in [0,1]$。转移分情况讨论即可。

题解

subtask 5:$d \leq 32$

```
if(s[j-1]=='0'){
    if(a[j][i]!='1') dp[j][0]|=dp[j-1][0]|dp[j-1][1];
    if(a[j][i]!='0') dp[j][1]|=dp[j-1][1],dp[j][0]|=dp[j-1][0];
}
if(s[j-1]=='1'){
    if(a[j][i]!='0') dp[j][1]|=dp[j-1][0]|dp[j-1][1];
    if(a[j][i]!='1') dp[j][1]|=dp[j-1][1],dp[j][0]|=dp[j-1][0];
}
```

最终答案就是每一位的可能性数乘起来。

题解

subtask 6:$d \leq 64$

没有什么特殊的，注意一下答案可能是 2^{64}。
这是一个经典易错点了，它甚至会爆掉 $unsigned\ long\ long$，如果你不想写高精，可以直接在每一位都有 2 种可能性时手动输出这个数值。
这个易错点继承自 CSP-S2020 T2，经典永流传。

bonus

关于位运算的另一个经典性质：
考虑每个二进制位，注意到 $\wedge 1$ 和 $\vee 0$ 实际上对答案没有任何影响。而 $\wedge 0$ 和 $\vee 1$ 一旦出现就会将答案强制变为 $0/1$，与前面的算式无关。

取出算式中的所有 $\wedge 0$ 和 $\vee 1$（第一位可以看作是一个 $\wedge 0$ 或 $\vee 1$）。

因此实际上每一位只有一种可能性当且仅当最后一个是 $\wedge 0$ 且之后全是 $\wedge$ 或 $\vee 0$；或者最后一个是 $\vee 1$ 且之后全是 $\vee$ 或 $\wedge 1$。基于这个性质可以拓展对更强的情况进行计数，不过没有放到这道题上。
在这道题上可能会让你像验题人一样因为没判第一位挂掉。

得分技巧与考试策略

总而言之 NOIP T1 肯定是首先要攻破的目标吧，即使不会也一定会有非常送分的部分分，通常就是非常特殊的性质或者直接爆搜。
在本道题上，就是出现位运算可以考虑拆开每一位。以及打表很多时候非常有用。

得分情况与吐槽

本题在考前难度连续降了两次。

题解

以下记 V 表示坐标的范围。

subtask 1:$n, q \leq 1000, V \leq 10$

暴力枚举矩形内所有点，复杂度 $\mathcal{O}(qV^2)$。这也是经典部分分，千万别忘了写。

题解

以下记 V 表示坐标的范围。

subtask 1:$n, q \leq 1000, V \leq 10$

暴力枚举矩形内所有点，复杂度 $\mathcal{O}(qV^2)$。这也是经典部分分，千万别忘了写。

subtask 2:$q \leq 10000, V \leq 1000$

之前的复杂度是单次询问 V^2，我们自然想到将 V^2 变成 V。
优化二维枚举的方法自然是一维枚举：考虑枚举每一行，同时计算这一行的答案。
不妨假设特殊点 u 在 v 的左侧。可以发现在这一行中从左到右，与 u 的距离逐渐变大，与 v 的距离逐渐变小。因此这一行要么全部 $a_u > a_v$，要么全部 $a_u < a_v$，要么存在某个分界点，左侧 $>$，右侧 $<$，中间可能 $=$（每次增加 2，所以不一定有 $=$）。
分界点可以二分，也可以直接计算。复杂度 $\mathcal{O}(qV)$。

题解

subtask 3:$n \leq 100$, $V \leq 1000$

注意到 $q > n^2$，因此考虑对每对特殊点进行预处理。
同样地，可以枚举这一对特殊点围成的矩形中的每一行计算分界点，不妨假设这对特殊点是 u, v,u 在 v 的左侧。
依然有两种情况；u 在 v 的左下方或者 u 在 v 的左上方。如果是后者，可以将 u, v 以及询问矩形都关于 y 轴对称变成第一种情况。

题解

subtask 3:$n \leq 100$, $V \leq 1000$

注意到 $q > n^2$，因此考虑对每对特殊点进行预处理。
同样地，可以枚举这一对特殊点围成的矩形中的每一行计算分界点，不妨假设这对特殊点是 u, v,u 在 v 的左侧。
依然有两种情况；u 在 v 的左下方或者 u 在 v 的左上方。如果是后者，可以将 u, v 以及询问矩形都关于 y 轴对称变成第一种情况。

此时在同一列从下往上，同样也是每移动一格，$a_u - a_v$ 增加 2。因此每一行的分界点组合起来会形成一条斜率为 1 的直线。预处理这条分界线，每组询问，用上边与下边和分界线求交即可。

题解

subtask 3:$n \leq 100,\ V \leq 1000$

注意到 $q > n^2$，因此考虑对每对特殊点进行预处理。
同样地，可以枚举这一对特殊点围成的矩形中的每一行计算分界点，不妨假设这对特殊点是 u, v,u 在 v 的左侧。
依然有两种情况；u 在 v 的左下方或者 u 在 v 的左上方。如果是后者，可以将 u, v 以及询问矩形都关于 y 轴对称变成第一种情况。

此时在同一列从下往上，同样也是每移动一格，$a_u - a_v$ 增加 2。因此每一行的分界点组合起来会形成一条斜率为 1 的直线。预处理这条分界线，每组询问，用上边与下边和分界线求交即可。

subtask 4:$q \leq 1000$

给离散化或其他神秘做法。

题解

subtask 5: 询问矩形是一行或一列

使用 subtask 2 的做法即可，不过这次需要行列都写。

题解

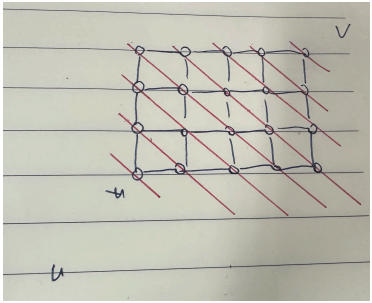
subtask 5: 询问矩形是一行或一列

使用 subtask 2 的做法即可，不过这次需要行列都写。

subtask 6: 无特殊限制

对每组询问使用 subtask 3 的做法，可以发现分界线可以直接计算得出，不需要对每行分别计算。

一个更好写的做法，可以直接通过以下方式求询问矩形与分界线的交线：



按上图方式分层，每层内部 $a_u - a_v$ 相同，层与层之间相差 2，可以 $\mathcal{O}(1)$ 找到分界层。复杂度 $\mathcal{O}(q)$。

代码实现

代码实现

很多同学估计是真的分了若干种情况讨论。
对于这类细节题目，在写之前一定要想好，要分哪些情况？每种情况怎么做，最好画画图。

得分情况与吐槽

题解

以下均假设 n, q 同阶。

subtask 1:$n \leq 100, q \leq 100$

枚举集结点 rt，则一条边的流量就是以 rt 为根时该边连接的子树的权值和。对每个 rt 都 $\mathcal{O}(n)$ 遍历一遍整棵树求流量，复杂度 $\mathcal{O}(qn^2)$。

题解

以下均假设 n, q 同阶。

subtask 1:$n \leq 100, q \leq 100$

枚举集结点 rt，则一条边的流量就是以 rt 为根时该边连接的子树的权值和。对每个 rt 都 $\mathcal{O}(n)$ 遍历一遍整棵树求流量，复杂度 $\mathcal{O}(qn^2)$。

subtask 2:$n \leq 1000, q \leq 1000$

可以发现只有与 rt 直接相连的边的流量可能会成为答案，可以直接枚举，枚举复杂度不超过 $\sum_{i=1}^{n} deg(i) = n$。
计算某点 u 的子树权值和也不需要暴力枚举，可以分三种情况：
1. 在原树上 u 在 rt 的子树内：则答案就是原树上 u 的子树权值和。
2. 在原树上 u 在 rt 的某个祖先的另一个儿子的子树内：答案也是原树上 u 的子树权值和。
3. 在原树上 u 是 rt 的祖先：答案为全树总权值和减去 u 最靠近 rt 的儿子的子树权值和。
这个最近的儿子可以倍增找到，预处理原树上的子树权值和即可做到复杂度 $\mathcal{O}(nq \log n)$。

题解

subtask 5: 没有操作 3

实际上是最好做的一个特殊性质。没有操作 3 就意味着在任何一个节点集结的流量永远不会改变，因此只需要在一开始的时候 $\mathcal{O}(n)$ dfs 一遍算出每个节点的流量，剩下的就是支持集合中加入某个数、删除某个数、询问最小值。使用 set 、线段树或者任意算法都可以做到 $\mathcal{O}(n \log n)$。

题解

subtask 5: 没有操作 3

实际上是最好做的一个特殊性质。没有操作 3 就意味着在任何一个节点集结的流量永远不会改变，因此只需要在一开始的时候 $\mathcal{O}(n)$ dfs 一遍算出每个节点的流量，剩下的就是支持集合中加入某个数、删除某个数、询问最小值。使用 set 、线段树或者任意算法都可以做到 $\mathcal{O}(n \log n)$。

subtask 3: 所有点都是特殊点

此时最优的集结点就是树的**带权重心**。
记 all 为全树所有点权值和，以下记 sum_u 表示 u 在原树上的子树权值和。
一个点是带权重心当且仅当其任意一个儿子的子树的权值和都 $\leq \frac{all}{2}$。证明考虑移动。
这也意味着带权重心在原树上的子树权值和 $\geq \frac{all}{2}$。并且可以证明一定存在一个带权重心的子树权值和严格 $> \frac{all}{2}$。

题解

subtask 3: 所有点都是特殊点

这就产生了一个精妙的计算带权重心的算法。考虑树的 dfs 序，每个点的子树都是 dfs 序的一个区间。而带权重心的子树覆盖了至少一半的权值，因此若取出 dfs 序上第一个前缀权值和 $> \frac{all}{2}$ 的位置，它一定在带权重心的子树中。
找出该点祖先中深度最大的满足 $sum_u \geq \frac{all}{2}$ 的 u，它一定是其中一个带权重心。

接下来就是要算 u 的权值，u 到父亲那条边可以直接计算，到儿子的边可以通过计算子树内（不包括 u 自己）的 sum 最大值得出。
使用线段树实现上述操作，复杂度 $\mathcal{O}(n \log^2 n)$，瓶颈在于倍增 $+$ 线段树判断。

题解

subtask 4: 没有 12 操作

找出带权重心 rt 后，考虑以 rt 为根。此时重心外每个点到父亲这条边的流量一定 $\geq \frac{all}{2}$，因此就是它的权值。也就是说它的权值就是 all 减去以 rt 为根时它子树的大小。

题解

subtask 4: 没有 12 操作

找出带权重心 rt 后，考虑以 rt 为根。此时重心外每个点到父亲这条边的流量一定 $\geq \frac{all}{2}$，因此就是它的权值。也就是说它的权值就是 all 减去以 rt 为根时它子树的大小。

之前 subtask 2 我们已经分析过了：对于不为 rt 祖先的点，其子树的大小与原树相同，很好维护。对于 rt 的祖先，显然权值最大的一定是深度最大的一个，我们只需要能找出这个深度最大的就行。最后别忘了 rt 自己的贡献。
非祖先的部分：rt 的祖先可以用树链剖分转化为 dfs 序上的 $\log$ 段区间，因此将所有特殊点按照 dfs 序排序，用线段树维护它们 sum 的最大值，并强制 rt 的祖先不向上贡献。
那么 3 操作对 c_x 的修改首先会导致其到祖先链上 sum 的链加，可以树剖拆分为线段树上的 $\log$ 个区间加。然后修改 rt，先将原先 rt 的祖先恢复贡献，再将新 rt 的祖先取消贡献。这些都可以通过线段树维护，需要维护区间内所有能贡献的点的 sum 最大值 mx_1，与所有点的 sum 最大值 mx_2。

题解

subtask 4: 没有 12 操作

对于祖先的部分，同样树剖拆分为求区间内最靠右的特殊点。随便怎么维护都行，甚至可以直接 lower_bound，当然为了后续带修，还是使用线段树吧。
总复杂度 $\mathcal{O}(n \log^2 n)$。

题解

subtask 4: 没有 12 操作

对于祖先的部分，同样树剖拆分为求区间内最靠右的特殊点。随便怎么维护都行，甚至可以直接 lower_bound，当然为了后续带修，还是使用线段树吧。
总复杂度 $\mathcal{O}(n\log^2 n)$。

subtask 6: 树是一条链

此时带权重心很好求只需要二分。权值最小的点则可以分别找左右最靠近 rt 的特殊点，使用线段树维护。

题解

subtask 7: 无特殊限制

在 subtask 5 的基础上增加了对特殊点的修改：
祖先部分：带修询问区间内最靠右的特殊点，线段树维护。
非祖先部分：对所有点建线段树，改为维护区间内所有有贡献的特殊点的 sum 最大值，和所有特殊点的 sum 最大值即可。

代码实现

我的代码实现非常的菜，有很多不必要部分。

对于这种数据结构题，在写之前一定要想好到底要维护什么，支持什么操作，不然就会像我的 NOIP2021 T4 一样遭重。代码最好框架化，有注释。
部分分通常对正解有很大帮助，想不出的时候可以思考，大胆地写。

得分情况与吐槽

题解

subtask 2:$n \leq 10, r = 1$

首先 $\prod r \leq 2^n$ 暴力枚举初始状态。
博弈问题不会怎么办？暴力维护整个局面的状态进行记忆化搜索/DP！
记 $max(S), min(S)$ 表示当前状态为 S，Alice 与 Bob 分别先手时的答案。暴力枚举所有能转移到的状态转移。因为所有边的总移动次数一定在减少，所以转移没有环。
复杂度 $\mathcal{O}(n2^{2n})$。

题解

subtask 2:$n \leq 10, r = 1$

首先 $\prod r \leq 2^n$ 暴力枚举初始状态。
博弈问题不会怎么办？暴力维护整个局面的状态进行记忆化搜索/DP！
记 $max(S), min(S)$ 表示当前状态为 S，Alice 与 Bob 分别先手时的答案。暴力枚举所有能转移到的状态转移。因为所有边的总移动次数一定在减少，所以转移没有环。
复杂度 $\mathcal{O}(n2^{2n})$。

subtask 3:$n \leq 10, r \leq 2$

初始状态是什么并不影响 DP 的过程，只需要进行一次 DP，复杂度 $\mathcal{O}(n3^n)$。

特殊性质 B：$r_i = 1$

首先考虑计算所有边权确定时的答案，再计数。
特殊性质 B 下，每条边要么只能经过一次，要么已经断掉。只需要维护剩下的边构成的结构，一定是多棵树，对每棵树分别处理。
在每棵树中，枚举每个出发点作为根，那么这次的博弈就简单了，只是从根出发走到某个叶子，不能走回头路。
记 $max(u), min(u)$ 表示当前在 u，Alice 与 Bo 分别先手时的答案，有转移：

$$max(u) = \max_{v \in son_u} min(v)$$

$$min(u) = \min_{u \in son_v} max(v)$$

叶子的 max, min 就是其权值。

特殊性质 B：$r_i = 1$

subtask 5:$n \leq 1500$

考虑计数。题目的一个关键性质是点权 ≤ 10。
因此考虑设 $cmax(d, u)$ 表示仅考虑 u 子树内边权的选择，$max(u) \geq d$ 的情况有多少种，$cmin(d, u)$ 为 $min(u) < d$ 的情况，有转移：

$$cmax(d, u) = \prod_{v \in son_u} (all_v - cmin(d, v))$$

$$cmin(d, u) = \prod_{v \in son_u} (all_v - cmax(d, v))$$

其中 all_u 时 u 子树内边权任取，一共有多少种情况。特判叶子结点。
最终答案就是 $\sum_d d(cmax(d, u) - cmax(d, u + 1))$。由于一开始要枚举根节点，复杂度 $\mathcal{O}(n^2 a)$。

特殊性质 B：$r_i = 1$

subtask 6:$n \leq 200000$

接下来自然就是经典的换根 dp 了。
先从 1 出发 dfs 一遍。再 dfs 一遍，求出 $DP_{fa}(u)$ 表示 u 父亲方向的子树的贡献。这里 DP_fa 是上面的 $cmax, cmin, all$ 等多种东西合并到一起形成的。在换根 dp 中，把所有元素合并到一起成为一个类并写一个比较好看的合并能够使代码简单很多。

第二遍 dfs 到点 u 时，将 u 的所有儿子取出来形成一个序列，由于上面的转移是 $\prod$，每个儿子 v 的 DP_{fa} 都可以通过一段前缀与一段后缀合并起来得到。最后再将每个点的 DP_fa 与它原本的子树权值合并起来即可得到以它为出发点的答案。这也是换根 DP 通用的一种写法。
复杂度 $\mathcal{O}(na)$。

特殊性质 C：$r_i \leq 2$

边 (u, v) 可以被走两次，意味着当 Alice 从 u 走到 v 时，Bob 可以立即走回来使得 Alice 无法再走到 v。
还是考虑权值确定时计算答案，此时可以认为只有权值为 $1, 2$ 的边。
沿用之前的 max, min 的 dp，假设当前棋子在 u，Alice 先手。如果 u 是叶子，游戏结束。

特殊性质 C：$r_i \leq 2$

边 (u, v) 可以被走两次，意味着当 Alice 从 u 走到 v 时，Bob 可以立即走回来使得 Alice 无法再走到 v。
还是考虑权值确定时计算答案，此时可以认为只有权值为 $1, 2$ 的边。
沿用之前的 max, min 的 dp，假设当前棋子在 u，Alice 先手。如果 u 是叶子，游戏结束。

如果 u 有权值为 1 的边，Alice 选择这条边，答案直接可以从 $min(v)$ 转移过来。因此 $max(u) \geq \max_{v \in son_u, w(u,v)=1} min(v)$。
如果 Alice 选择走其它边，Bob 一定会选择沿这条边走回来。这是因为 Alice 无论如何都有 $\max_{v \in son_u, w(u,v)=1} min(v)$ 保底了，再走其他边一定是因为走了更优，那 Bob 当然选择直接 ban 掉。
因此 $max(u) = \max_{v \in son_u, w(u,v)=1} min(v)$。

特殊性质 C：$r_i \leq 2$

但如果 u 的所有连边权值都为 2，情况就不同了。如果 Bob 选择全部 ban 掉，Alice 就会停在 u，但 Bob 实际上有可能在 Alice 移动后将棋子困在某棵子树内，因此上面的理论就不成立了。
这就出现了困住这个概念，记 $mxs(u)$ 表示 Alice 先手，棋子在 u，u 到父亲有边时 Alice 能否将棋子困在 u 子树内。如果不能则 $mxs = -INF$，否则 $mxs =$ Alice 在困住的基础上，能做到的最大权值。

这时实际上是 Bob 在做选择，因为只要 Bob 一直撤回，Alice 就必须遍历选择直到 Bob 满意，最后 Bob 还能停在 u。如果 Bob 不能停在 v 子树，那么 Alice 可以回到 u，那么选择 v 就不如选择停在 u 了。
于是有转移：

$$max(u) = \min(a_x, \min_{v \in son(u)} mns(v))$$

min 的转移时类似的。

特殊性质 C：$r_i \leq 2$

考虑 mxs 的转移。如果 u 没有权值为 1 的边，那么只要 Bob 依次撤回每个操作，Alice 只能回到父亲。所有 $mxs_u = -INF$。
否则，Alice 可以停留在这些权值为 1 的边的子树中，于是：

$$mxs(u) = \max_{v \in son(u), w(u,v)=1} mn(v)$$

mns 的转移是类似的，至此我们就得到了一个 $\mathcal{O}(n)$ 计算单次权值的算法。
再扩展到计数前，我们先扩展到 $r_i > 2$ 的情况。

特殊性质 A：$l_i = r_i$

一条边权为 $k > 2$ 的边会有什么影响？根据上面的讨论，你想到了：边权为 k 是否与边权为 $k-2$ 等价？
答案是正确的。因为只要 Bob 第一次走这条边时，Alice 立即撤回，此后 Bob 的选择空间没有发生变化，Alice 可以保证答案不小于 $k-2$ 时的答案。Bob 也可以做到类似的事。因此边权为 k 的答案与边权为 $k-2$ 的答案确实相同。

由此，对于一般的 $l_i = r_i$ 的情况，我们可以将边权转化为 $0, 1, 2$ 后直接用上面的 dp 完成。再加上换根 dp 即可通过所有特殊性质 A 的点。

计数

使用类似 subtask 5 的做法，记设 $cmax(d,u)$ 表示仅考虑 u 子树内边权的选择，$max(u)\geq d$ 的情况有多少种，$cmin(d,u)$ 为 $min(u)<d$ 的情况，$cmxs(d,u),cmns(d,u)$ 类似。
为了方便记 $w_{i,j,0/1/2}$ 表示 (i,j) 这条边分别有多少种情况可以转化为边权为 $0/1/2$。

对于 $cmax(d,u)$ 的转移，也要分两种情况：
1.u 没有边权为 1 的儿子，此时要求 $a_x\geq d$ 且对所有 $w(u,v)=2$ 的 v，$mns(v)$ 必须 $\geq d$。因此对于每个儿子 v，要么选择 $w(u,v)=0$，然后子树内可以任选；要么选择 $w(u,v)=2$，然后子树内 mns 必须 $\geq d$，有转移：

$$cmax(d,u)\leftarrow[a_u\geq d]\prod_{v\in son_u}(w_{u,v,0}all_v+w_{u,v,2}(all_v-cmns(v,d)))$$

2. u 有边权为 1 的儿子，且这些儿子的 mn 的最大值 $\geq d$。这里可以取补集，则所有儿子要么 $w=0/2$，要么 $w=1$ 且 $mn<d$，于是：

$$cmax(d,u)\leftarrow all_u-\prod_{v\in son_u}(w_{u,v,0}all_u+w_{u,v,2}all_u+w_{u,v,1}cmin(v,d))$$

计数

其余的转移是类似的：

$$cmin(d,u) \leftarrow [a_u < d] \prod_{v \in son_u} (c_{u,v,0}\, all(v) + c_{u,v,2}(all(v) - cmns(d,v))$$

$$cmin(d,u) \leftarrow all_u - \prod_{v \in son_u} (w_{u,v,0}\, all_u + w_{u,v,2}\, all_u + w_{u,v,1}\, cmin(d,v))$$

$$cmxs(d,u) = all_u - \prod_{v \in son_u} (w_{u,v,0}\, all_u + w_{u,v,2}\, all_u + w_{u,v,1}\, cmin(d,v))$$

$$cmns(d,u) = all_u - \prod_{v \in son_u} (w_{u,v,0}\, all_u + w_{u,v,2}\, all_u + w_{u,v,1}\, cmin(d,v))$$

使用换根 DP 优化，复杂度 $\mathcal{O}(na)$。