题目大意

给定一颗 n 个点的有根树，1 是根，记 u 的父亲是 fa_u。另给出一长度为 n 的权值序列 b。

称一个长度为 n 的排列 a 为这颗树的合法拓扑序，当且仅当 $\forall 2 \leq u \leq n, a_u > a_{fa_u}$。

对每个点 u，定义 $f(u)$ 为，在所有这颗树的合法拓扑序中，b_{a_u} 之和。

现在对 $1 \leq u \leq n$，求 $f(u) \bmod 10^9 + 7$。

输入格式

第一行一个整数 n 表示树的点数。

第二行 $n-1$ 个数，第 i 个表示 fa_{i+1}，描述树的结构。

第三行 n 个整数，第 i 个表示 b_i，描述权值序列。

输出格式

一行 n 个整数，第 i 个表示 $f(i) \bmod 10^9 + 7$。

数据范围

Subtask	$n \leq$	特殊限制	分值
1	10	无	5
2	20	无	10
3	100	无	15
4	350	无	15
5	3000	A	15
6	3000	无	20
7	5000	无	20

特殊限制 A：$\forall 1 \leq i \leq n, b_i = i$。

对于所有数据：$2 \leq n \leq 5000$，$1 \leq fa_i < i$，$0 \leq b_i < 10^9 + 7$。

解题过程

记 u 的子树大小是 siz_u，每个算法中 dp 数组的定义基本都是独立的。

- 算法一

  直接暴力枚举所有 $n!$ 种排列 a 并判断是否是合法拓扑序。

  时间复杂度 $O(n!n)$。

- 算法二

  本算法记 $ans_{x,y}$ 为 $a_x = y$ 的合法拓扑序数量，且填的数和点都从 0 开始标号。

原题意可以看作从小到大填数，每个位置只能填一个数，对位置 u 填数时，其父亲必须已经被填上数，数有多少填数方案。$ans_{x,y}$ 则相当于额外增加了数 y 必须被填在点 x 的限制 。

建一张 2^n 个点的图，代表填数过程中每个点是否已经被填上数的 $0/1$ 状态，点 S 向点 T 有有向边，当且仅当作为二进制数 $S\in T$，$\text{popcnt}(S)+1=\text{popcnt}(T)$，且设 T 相比 S 多出的一位是 u，S 必须包含 fa_u 这一位（如果 u 不是根）。

原题意可以看作数有多少 0 到 2^n-1 的路径，$ans_{x,y}$ 则是额外增加了走的第 y 步必须增加 2^x 的限制。

对每个点算出 f_S 表示 0 走到 S 方案数，g_S 表示 S 走到 2^n-1 方案数。枚举所有 S 并枚举其所有出边 $S\to T$，$S+2^p=T$，给 $ans_{p,\text{popcnt}(S)}$ 加上 f_Sg_T 即可。

时间复杂度 $O(2^nn)$。

- 算法三

  考虑 DP，我们希望以 $O(n^2)$ 时间对于一个 p 算出 $f(p)$。

  设 ord_u 为 u 子树的合法拓扑序数量（“子树内的” 即只对 u 子树内的点填 a，值域是 $[1,siz_u]$），$dp_{u,i}$ 为 u 子树内的合法拓扑序，且 $a_p=i$ 的数量，$dp_{u,i}$ 只在 u 为 p 祖先时有定义。

  ord_u 非常经典，就是 $siz_u!\prod\limits_{v\in subtree(u)}\dfrac{1}{siz_v}$。

  $dp_{u,i}$ 则考虑按照 u 的深度从大到小求出。$u=p$ 时 $dp_{p,1}=ord_p$，而对于 $k>1$，$dp_{p,k}=0$。

  对于剩余的 $dp_{u,i}$，先让 u 继承 u 向 p 方向孩子的 dp 信息。接下来依次考虑 u 其他孩子 $v_1\sim v_k$ 的影响，加入孩子 v_i' 时，考虑 v_i' 子树内有 j 个点 $a<a_p$，那么将 $dp_{u,i}$ 以 $ord_{v_{i'}}\dbinom{i-1+j}{j}\dbinom{siz_u-i+siz_{v_i'}-j}{siz_u-i}$ 倍的权值转移到 $dp'_{u,i+j}$，dp' 是新考虑孩子 $v_{i'}$ 后的 dp 值。siz_u 是当前 u 的子树大小，会随着不断合并孩子不断增大。合并完所有孩子后，a_u 一定只能填 1，所以还要对 dp 数组向右平移 1。

  最后 $dp_{1,i}$ 的值就是原问题中 $a_p=i$ 的合法拓扑序数量，与权值向量做点乘即可求出 $f(p)$。

  时间复杂度根据树形背包的复杂度分析可得是单次 $O(n^2)$ 的，因此总复杂度是 $O(n^3)$。

- 算法四

  如果 $b_i=i$，那么拆贡献是容易的，记 $g(u,v)$ 为满足 $a_u<a_v$ 的合法拓扑序数量与总拓扑序数量 ord_1 之比，那么 $f(v)=(\sum\limits_{u\neq v}g(u,v)+1)ord_1$。

  考虑对于固定的一对点 u,v，$g(u,v)$ 怎么算，这其实是 AGC060C。如果 u,v 有祖先后代关系，答案是显然的。若 u,v 没有祖先后代关系，显然只保留 u,v LCA 子树内的所有点，得到的占比与原问题一致。考虑算法二中提及的填数转化，我们其实只关心 u,v 到 LCA 的链上一共有多少个点被填上了数，并且根据那题的结论，假设 u 到 LCA 链上第一个没被填数的点是 p，v 是 q，那么先填 p 的排列占比是 $\dfrac{siz_p}{siz_p+siz_q}$，据此可以 $O(n^2)$ DP 算出一组 $g(u,v)$，具体是设 $dp_{i,j}$ 表示 u 到 LCA 链上下一个填 i，v 到 LCA 链上下一个填 j 的概率和。

  回到原问题，注意到只要 u 是 i 的后代，v 是 j 的后代，$dp_{i,j}$ 总是一致，求出这些一致的值。该 DP 对答案产生贡献当且仅当 $u=i$，v 是 j 的后代，此时会给 v 加上 $dp_{i,j}\dfrac{siz_i}{siz_i+siz_j}$ 的概率。

  dp 的具体转移：初始值是如果 i,j 是兄弟节点那么 $dp_{i,j}=1$，转移是如果 fa_i 不是 j 的祖先，那么 $dp_{i,j}$ 加上 $\dfrac{siz_{fa_i}}{siz_{fa_i}+siz_j}dp_{fa_i,j}$，同理如果不是 fa_j 不是 i 的祖先，那么 $dp_{i,j}$ 加上 $\dfrac{siz_{fa_j}}{siz_i+siz_{fa_j}}dp_{i,fa_j}$。

  时间复杂度 $O(n^2)$。

- 算法五

考虑把算法三反转过来，设 $dp_{u,i}$ 表示 u 子树内所有点的 a 内部顺序尚未确定，但 u 子树外所有点 a 相对顺序已经确定，且也已经确定了将 u 子树内所有点的 a 与外部归并起来时，哪些值属于子树内 a，哪些属于子树外 a，并且我们关心的关键点 p 在 u 子树内排名为 i 的所有方案的 b_{a_p} 之和。

这样定义的好处在于 p 是 u 子树内的具体哪个点并不重要，从而只有 $O(n^2)$ 种信息。

对于根，$dp_{1,i}=b_i$。对于其他点 u，考虑先让 dp_u 继承其父亲 dp_{fa_u} 的信息（当然，要向左平移 1 因为 fa_u 一定在 fa_u 子树内排名为 1），然后依次去掉 fa_u 其他孩子 $v_1\sim v_k$ 的信息，去掉孩子 $v_{i'}$ 信息时，考虑 $v_{i'}$ 子树内有 j 个 $<a_p$ 的点，那么以 $\binom{i-1}{j}\binom{siz_{fa_u}-i}{siz_{v_{i'}}-j}ord_{v_{i'}}$ 的权值从 $dp_{u,i}$ 转移到 $dp'_{u,i-j}$，同样地 siz_{fa_u} 是当前 fa_u 的子树大小，会随着不断去掉孩子不断减小

结算答案时，发现 $f(u)$ 实际上就是 $dp_{u,1}ord_u$。

可以发现无论是定义还是转移，这个算法确实就是算法三的反转，但问题在于，直接暴力去除信息复杂度显然不正确。

仔细分析这个减法卷积，假设去除前 siz_{fa_u} 是 sz，复杂度实际上是 $O(siz_{v_{i'}}(sz-siz_{v_{i'}}))$。我们实际上希望对 fa_u 的每个孩子，算出从 dp_u 去除除了它以外所有孩子信息的结果，直接像洛谷 P4095 一样分治去除的复杂度是 $O(n^2\log n)$，下面给出证明：

- 考虑节点 u 分治去除的代价，记其有孩子 $v_1\sim v_k$，当分治到 $[l,r]$ 时会分别需要从 $[l,r]$ 的信息去除 $[l,mid]$，以及从 $[l,r]$ 的信息去除 $[mid+1,r]$，假设均按照编号从小到大去除（不是其实也一样），代价是 $\sum\limits_{i=l}^{mid}siz_{v_i}\sum\limits_{j=i+1}^{r}siz_{v_j}+\sum\limits_{i=mid+1}^{r}siz_{v_i}(\sum\limits_{j=l}^{mid}siz_{v_j}+\sum\limits_{j=i+1}^{r}siz_{v_j})$，整理后是 $2(\sum\limits_{i=l}^{mid}siz_{v_i})(\sum\limits_{i=mid+1}^{r}siz_{v_i})+\sum\limits_{l\le i<j\le mid}siz_{v_i}siz_{v_j}+\sum\limits_{mid<i<j\le r}siz_{v_i}siz_{v_j}$。
- 那么对于一对 $i<j$，$siz_{v_i}siz_{v_j}$ 被计入代价的次数不超过 $2+\log k$。于是总复杂度不超过 $O(n^2\log n)$，可能实际上是 $O(n^2)$ 的，不太确定。

- 算法六

考虑改进算法五，其实并不一定要按顺序去掉其他孩子的信息，我们也可以把其他孩子的信息一起去掉，考虑如果要一起去掉孩子 $v_{i'},v_{j'}$ 的信息，假设这两个子树内一共有 j 个 $<a_p$ 的点，那么转移系数应该是 $\binom{i-1}{j}\binom{sz-i}{siz_{v_{i'}}+siz_{v_{j'}}-j}ord_{v_{i'}}ord_{v_{j'}}\binom{siz_{v_{i'}}+siz_{v_{j'}}}{siz_{v_{i'}}}$，于是现在可以在 $O(sumv(sz-sumv))$ 的时间内去掉子树大小和为 $sumv$ 的很多孩子的信息了。

再使用分治去除信息，此时从 $[l,r]$ 分别去除 $[l,mid]$，$[mid+1,r]$ 的代价是 $2(\sum\limits_{i=l}^{mid}siz_{v_i})(\sum\limits_{i=mid+1}^{r}siz_{v_i})$，总复杂度是 $O(n^2)$ 的

可以发现此时实际上取 $mid=\dfrac{l+r}{2}$ 是不必要的，随便取 mid 复杂度都是 $O(n^2)$。

不过其实此时都不需要分治了，直接对每个孩子，以 $O((siz_u-siz_v)siz_v)$ 的时间去掉其他孩子的信息时间复杂度也是 $O(n^2)$。

- 算法七

设 $ans_{x,y}$ 为 $a_x=y$ 的合法拓扑序数量，无论是算法五还是算法六，都只能对所有 x 求出 $\sum\limits_{y=1}^{n}ans_{x,y}b_y$ 的值，但我们有一个可以在 $O(n^2)$ 时间内求出所有 $ans_{x,y}$ 的算法。

考虑以下过程：

- 一开始所有点都是白点，进行 n 轮染色操作，每次选一个白点，染黑其祖先中深度最浅的白点 p，假设这是第 i 轮染色，那么记录 $a_p=i$。

显然有 $n!$ 种染色方式，且无论怎么染色，最后得到的 a 都是一个合法拓扑序。于是该过程与合法拓扑序建立了对应关系，但一个合法拓扑序可能对应多种染色方案。

- 结论：任意一个拓扑序恰对应 $\prod_{i=1}^{n} siz_i$ 种染色方案。

这事实上只需考虑每个点被染黑时，导致它染黑选的是哪个白点即可，a 指出了每轮操作被染黑的是哪个点，而对于点 u，当且仅当选到了它子树内的任意白点时，它被染黑，这有 siz_u 种方案。

于是要算 $ans_{x,y}$，只需算出第 y 轮点 x 被染黑的染色方案即可，记这个值是 $ans'_{x,y}$。

考虑 DP，假设固定 x，那么抽出 x 到 1 的链，设 $dp_{i,j}$ 为第 i 轮链上的第 $[1,j]$ 个点被染黑的染色方案数，记 s_j 为链上第 j 个点的子树大小，链上一共有 k 个点，那么转移是：

边界：$dp_{0,0}=1$。

对于 $i\geq 1,1\leq j<k$，$dp_{i,j}=(n-(i-1)-s_{j+1})dp_{i-1,j}+s_jdp_{i-1,j-1}$。

$ans'_{x,y}=s_kdp_{y-1,k-1}(n-y)!$。

当然你直接转移还是 $O(n^3)$ 的，不过类似算法四，注意到大量信息是可以公用的，修改定义为 $dp_{i,u}$ 表示染色 i 轮，当前链上 u 是首个尚未被染黑的元素，那么边界是 $dp_{0,1}=1$，转移是 $dp_{i,u}=dp_{i-1,u}(n-(i-1)-siz_u)+dp_{i-1,fa_u}siz_{fa_u}$，当 $u=1$ 时没有后一项的值。

算答案的时候 $ans'_{u,y}=dp_{u,y-1}siz_u(n-y)!$。

参考资料

无。