A 随机游走

设 f_u 为 u 子树的答案，T_u 为 u 子树内的所有边权之和，W_u 为 u 子树内所有点权之和，则假如已经确定了遍历儿子的顺序 $v_1, v_2, \ldots, v_k$。

那么转移式为 $f_u = \sum_{i=1}^{k} \left(f_{v_i} + W_{v_i} \times \sum_{j=1}^{i-1} T_{v_j} \right)$。

考虑交换相邻两个儿子 v_i, v_{i+1} 发生的变化，显然 v_i, v_{i+1} 之前以及 v_i, v_{i+1} 之后的贡献分不会变化，变化的是 v_i, v_{i+1} 带来的贡献：

1. 当遍历顺序 v_i 先于 v_{i+1} 时，这部分贡献为 $f_{v_i} + W_{v_i} \times \sum_{j=1}^{i-1} T_{v_j} + f_{v_{i+1}} + W_{v_{i+1}} \times \sum_{j=1}^{i} T_{v_j}$
2. 当遍历顺序 v_{i+1} 先于 v_i 时，这部分贡献为
   $f_{v_{i+1}} + W_{v_{i+1}} \times \sum_{j=1}^{i-1} T_{v_j} + f_{v_i} + W_{v_i} \times \left(\left(\sum_{j=1}^{i-1} T_{v_j} \right) + T_{v_{i+1}} \right)$

则遍历顺序 $v_i v_{i+1}$ 先于 $v_{i+1} v_i$ 时，需要满足的条件为

$$f_{v_i} + W_{v_i} \times \sum_{j=1}^{i-1} T_{v_j} + f_{v_{i+1}} + W_{v_{i+1}} \times \sum_{j=1}^{i} T_{v_j} < f_{v_{i+1}} + W_{v_{i+1}} \times \sum_{j=1}^{i-1} T_{v_j} + f_{v_i} + W_{v_i} \times \left(\left(\sum_{j=1}^{i-1} T_{v_j} \right) + T_{v_{i+1}} \right) \tag{1}$$

$$f_{v_i} + f_{v_{i+1}} + W_{v_i} \times \sum_{j=1}^{i-1} T_{v_j} + W_{v_{i+1}} \times \left(\left(\sum_{j=1}^{i+1} T_{v_j} \right) + T_{v_i} \right) < f_{v_i} + f_{v_{i+1}} + W_{v_i} \times \left(\left(\sum_{j=1}^{i+1} T_{v_j} \right) + T_{v_i+1} \right) + W_{v_{i+1}} \times \sum_{j=1}^{i-1} T_{v_j}$$

$$W_{v_{i+1}} T_{v_i} < W_{v_i} T_{v_{i+1}}$$

然后就将 sort 里面的 cmp 函数填上这个就可以了。

为什么将儿子序列排一遍序是正确的？我们考虑任一儿子序列 $v_1, v_2, \ldots, v_k$，根据冒泡排序的原理，每次选出两个相邻的逆序对并交换，可以使逆序对刚好 -1，直至没有相邻的逆序对，说明序列已排序，而在本题中将两个相邻的不优对进行交换，每次交换都会使答案变得更优。直至没有相邻的不优对，说明儿子序列也已按照某种特定的顺序进行了排序，而对于任意一种序列，都可以做若干次使答案更优的交换操作使其变成这种特定顺序的序列，说明这种序列的顺序就使所有排列方式中最优的，故直接排序是完全正确的。

时间复杂度：$O(n \log n)$。

B 分发奖励

有一万种做法可以做这道题，这里讲一下符合 NOIP T2 定位的简单做法。

首先求出树的其中一个 dfs 序，然后变成了每次给两个区间塞一个相同的数。

虽然是两个区间，你可能会因此转化成高维的问题然后大力套数据结构维护了，但是因为是对每个点求出有多少个和它相同的，对于序列来讲，我们按顺序遍历每个点，进入到一个区间的时候就将它和它对应着的区间加进线段树里，出来的时候就删去，然后要询问的是被区间覆盖的点数数量，这是个经典的扫描线问题。

树上也可以这么做，不过有一个更简便的做法，还是先求出树的 dfs 序，不过枚举的时候不用先转成序列，直接遍历一遍树就可以了，进入这个节点的时候就把该加的所有区间都加进线段树里，出来的时候就删掉，然后每次再在线段树里查询一遍即可。

时间复杂度：$O((n+m) \log n)$

C 卡路里

首先最优情况是选择一段区间 $[l,r]$ 然后从 l 一直向右走走到 r（反过来也行）然后每款奶茶选择区间最大卡路里的店（如果有多个相同的则认为位置靠左的最大）。

考虑对于第 i 个店的第 j 款奶茶，什么时候会成为区间 $[l,r]$ 内的所有第 j 款奶茶中卡路里量最大的那个，当且仅当 $l'\le l\le i\le r\le r'$，我们只需要求出 l',r' 即可。

显然 l' 要满足 $l'\le i$ 中距离 i 最近且满足 $a_{l'-1}\ge a_i$ 的，r' 则需要满足 $r'\ge i$ 中距离 i 最近且满足 $a_{r'+1}>a_i$（这里认为 $a_0=a_{n+1}=+\infty$），可以用单调栈轻松求出。

所以我们可以得出，如果 $l\in[l',i]$ 且 $r\in[i,r']$，那么区间 $[l,r]$ 的代价就会因此增加第 i 个店中第 j 款奶茶的卡路里数，容易发现这是一个二维平面加的形式，可以通过二维差分做到 $O(1)$ 加。

这样的话对于每个 $[l,r]$，先求出选择的奶茶的最大卡路里之和，这其实就是单点查的形式，在查之前，先做一遍二维前缀和把差分转化成原来的形式，接着再求出区间要走的里程数，二者意见就是该区间的答案了，最后的答案就是这 $O(m^2)$ 个区间的最大答案。

时间复杂度：$O(m^2+nm)$

D 传话游戏

$$n,m\le 30$$

考虑对每个元素 $(S_1)_i$ 设一个存活时间 t_i 代表 $S_1\sim S_{t_i}$ 中 $(S_1)_i$ 始终未被删去直到 S_{t_i+1} 开始元素 $(S_1)_i$ 才被删去（$t_i=n$ 代表 $(S_1)_i$ 永远不会被删去）。

考虑什么时候才会出现本质相同的两种子序列，就是有同种元素的情况下才可能会发生，因此对同种元素考虑是值当的。

先找一些必要条件：若 $(S_1)_i=(S_1)_j$，且当前来到 $S_{\min(t_i,t_j)}$，如果 $S_{\min(t_i,t_j)}$ 的 $(S_1)_i=(S_1)_j$ 中间没有不同的元素要保留到下一轮，那么删 $(S_1)_i$ 还是 $(S_1)_j$ 那么本质都相同，所以当 $t_i>t_j$ 和 $t_i<t_j$ 时都只能保留一种计数，这里只保留 $t_i>t_j$。

根据这个规定，可以得到一个很重要的必要条件：若 $(S_1)_i=(S_1)_j$ 且 $t_i<t_j$，那么必须保证存在 $i<k<j$ 且 $(S_1)_k\ne(S_1)_i=(S_1)_j$ 满足 $t_k>\min(t_i,t_j)$ 相当于 $t_k>t_i$。

考虑这个条件的充分性，考虑对每个 S_t 分开考虑，此时所有的 $t_i\ge t$ 可以看成 1，所有的 $t_i<t$ 可以看成 0，那么其实就是在证明每个 S_1 的子序列都存在唯一的 $[t_1,t_2,\dots,t_m]$ 表示方法。

这个其实是简单的，将所有 t_i 转化为 $0/1$ 之后，唯一不合法的条件是存在 $(S_1)_i=(S_1)_j$ 使得 $t_i=0,t_j=1$ 且对于所有的 $i<k<j$ 且 $(S_1)_k\ne(S_1)_i=(S_1)_j$ 都满足 $t_k=0$，但其实 $(S_1)_k\ne(S_1)_i=(S_1)_j$ 这个条件可以去掉，因为就算存在 $(S_1)_k=(S_1)_i=(S_1)_j$ 且 $t_k=1$，那么不合法的条件可以缩小到区间 $[i,k]$，这样一直缩小总能找到最开始不合法的 i,j。

于是不合法的条件就变成了：存在 $(S_1)_i=(S_1)_j$ 使得 $t_i=0,t_j=1$ 且对于所有的 $i<k<j$ 都满足 $t_k=0$，

那这个其实就简单了，将 t_i 设为 1 的过程其实就是按顺序枚举序列并贪心构造子序列的过程，因为假如当前要找 x 来构造子序列，那么如果枚举到了一个 x 却不选，那么这个位置的 $t=0$，那么在之后也一定要选择 x，这个位置的 t 一定等于 1，且这中间的元素也一定没法选，所以这个选法一定是不合法的，这也就证明了 S_1 的每个子序列对应着唯一的 01 序列 t。

有了充分性之后，就可以得出刚才的那个必要条件其实是**充要条件**，我们直接对合法的 t 计数即可。

但是直接对这个条件计数还是难做，但可以发现该条件可以转化成对于每个 i，满足 $t_j>t_i$ 且 $j>i$ 的最小的 j 都要保证 $(S_1)_i\ne(S_i)_j$。

转化成这个可以看成是类似于在笛卡尔树上进行的区间 dp 问题，直接设 $f_{l,r,x,i}$ 为 $t_{l\sim r}$ 内满足区间 $[l,r]$ 的最大 t 值来自于位置 x，且值为 i 的方案数（如果有多个最大值则认为**位置最小的最大**），如果预处理出 $f_{l,r,x,*}$ 的前缀和可以做到 $O(nm^5)$。

$n \leq 200$

但是这个 x 根本无需存，因为无论如何满足 $t_j > t_x$ 且 $j > x$ 的最小的 j **一定是** $r+1$（如果 $r = m$ 直接令 $(S_1)_{m+1} = 0$ 即可，这样 $i \in [1, m]$ 内没有 $(S_1)_i$ 等于 $(S_1)_{m+1}$），所以只需要保证 $(S_1)_x \neq (S_1)_{r+1}$ 即可。

于是只需要记 $f_{l,r,i}$ 为 $t_{l\sim r}$ 内满足区间 $[l, r]$ 的 t 的最大值为 i，且该最大值位置的 S_1 与 $(S_1)_{r+1}$ 不同的方案数即可，同样预处理出 $f_{l,r,*}$ 的前缀和可以做到 $O(nm^3)$。

$m \leq 70$

由于 n 特别大，但注意到 $f_{l,r}$ 可以看成关于 i 的 $r-l$ 次多项式，那么 $f_{l,r,*}$ 的前缀和数组 $sum_{l,r}$ 就可以看成关于 i 的 $r-l+1$ 次多项式。

最后答案就是 $sum_{1,m,n}$。

证明考虑归纳，当 $l = r$ 时 $f_{l,r,i} = 1$，所以 $f_{l,r}$ 可以看成关于 i 的 0 次多项式，假如区间长度 $len < r-l+1$ 的 $f_{l',r'}$ 都可以看成关于 i 的 $len-1$ 次多项式，而 $sum_{l',r'}$ 可以看成关于 i 的 len 次多项式。因为 $f_{l,r}$ 的转移过程就是枚举最大值位置 mid，然后分成两个子区间 $[l, mid-1]$ 和 $[mid+1, r]$，每个子区间都是两个 sum 数组相乘，分别相当于关于 i 的 $mid-l$ 次和 $r-mid$ 次多项式，所以卷起来之后就是关于 i 的 $r-l$ 次多项式，$sum_{l,r}$ 也就是关于 i 的 $r-l+1$ 次多项式。

有了这个条件之后，你可以直接把 $f_{l,r}, sum_{l,r}$ 都改成多项式的形式，这样 dp 复杂度是 $O(m^5)$ 的。

正解

但其实不用这么麻烦，由于一个 len 次多项式可以用 $len+1$ 个不同点值然后用**拉格朗日插值**插出来，而由于 $sum_{1,m}$ 在前文分析完是 $m+1$ 次多项式，所以你只需要求出 $sum_{1,m,0\sim(m+1)}$ 就可以通过拉格朗日插值插出 $sum_{1,m,n}$，从而将时间复杂度优化到 $O(m^4)$，**常数很小**，可以通过。