生成树

由于 i 与 $i+1$ 最大公约数一定为 1，因此最小生成树一定是 $n-1$。

对于最大生成树，从大到小枚举边权 w_i，不难发现只需要尝试连接 w_i 与 $2\times w_i$，$2\times w_i$ 与 $3\times w_i$，$\ldots$ 即可。

时间复杂度 $O(n\log n)$

序列

我们先做一些初步分析，设每个点往后给了 c_i，若 $\forall c_i>0$，那么我们可以全部 -1。

于是我们考虑枚举一个 0，这样这个点是没有往后放的，就可以破坏为链。

为了避免重复计数，我们枚举第一个 0，强制前面没有 0。此时，我们计算所有情况的贡献和。此时，我们暴力设 $dp_{i,j}$ 表示到 i，给后面 j 个的贡献，那么有 $dp_{i,j}=(a_i+k-j)dp_{i-1,k}$。

我们注意到只需要维护 $\sum dp_{i,j},\sum dp_{i,j}\times j$ 就可以完成转移。此时复杂度 $O(n^2)$。注意到转移可以写成一个 2×2 的矩阵乘法，我们维护一个前后缀就可以了。复杂度 $O(n)$。

魔力

20 Pts

可以 2^n 枚举所有可能的状态，对每个状态求出权值，然后枚举全排列按题意所述求解即可。

若树成一条链

此时桶子喜欢所有点对。

若将一个点 x 从 A 集合挪到 B 集合，考虑此时不满值的变化量：

$\Delta=\sum_{i\in B}[anc(x,i)\wedge w_i>w_x]+[anc(i,x)\wedge w_i<w_x]-\sum_{i\in A}[anc(x,i)\wedge w_i<w_x]+[anc(i,x)\wedge w_i>w_x]-d_x$

把后半段化为容斥处理，后半段可以写成：

$-(\sum_{i\in A}[anc(x,i)\lor anc(i,x)]-\sum_{i\in A}[anc(x,i)\land w_i>w_x]+[anc(i,x)\land w_i<w_x]-\sum_{i\in A}[(anc(i,x)\lor anc(x,i))\land w_i=w_x])$

把括号拆开与前半段合并，有：

$\Delta=\sum_i[anc(x,i)\land w_i>w_x]+[anc(i,x)\land w_i<w_x]-d_x-\sum_{i\in A}[anc(x,i)\lor anc(i,x)]+\sum_{i\in A}[(anc(i,x)\lor anc(x,i))\land w_i=w_x])$

前三项对每个点来说是定值，可以用一遍预处理计算出来。

对一条链而言，第四项对每一个 A 中的点都相同，可以忽略不计。

只需要考虑第五项。

考虑两个 w 相同的节点 x,y，设 x 是 y 的祖先；

只考虑前半部分而言，对两个点贡献相同的部分即为 x 的祖先部分和 y 的子树部分。

由于 x 与 y 之间的节点数一定小于 d_y-d_x，就算这些节点都对 y 有贡献而对 x 无贡献，也一定满足 $\Delta_x>\Delta_y$。

因此先操作 y 一定是更优秀的，对于节点 x 我们都已经知道子树中其余权值为 w_x 的节点都已经被操作过，而所有祖先中权值为 w_x 的节点都没有被操作过，也就是第五项的值是确切已知的。

于是乎，每个点的 Δ 除去第四项其实是定值，而第四项与选中哪个点无关，我们可以直接将 Δ 预处理出来然后对所有点排序，从小到大加入即可。

时间复杂度 $O(n\log n)$。

若所有节点权值相同

设 $anc(x,i)$ 表示 x 是 i 的祖先。

此时冈伦、红莉栖喜欢所有点对。

首先对于点对 (i,j)，显然只有形如 $i<j$ 的点对会产生不满值，因此我们只计算 $i<j$ 的点对。

考虑一开始的不满值计算，使用容斥：

$\frac{n\times(n-1)}{2}-\sum_i\sum_{j>i}[anc(i,j)\lor anc(j,i)]$

若将点 x 移动至 B 集合，忽略 $\frac{n\times(n-1)}{2}$ 这一坨，不满值的变化量 Δ 可以用如下式子计算：

$\Delta_x=\sum_{i\in A}[anc(i,x)\lor anc(x,i)]-d_x$

考虑应该按什么顺序加入这些点。

与树成一条链的情况类似，稍加讨论可以发现儿子一定比父亲更优秀，因此在插入一个点之前只需要默认子树都已经被加入即可。

时间复杂度 $O(n\log n)$。

若所有节点权值互不相同

发现冈伦和桶子的不满值计算非常讨厌，仍然可以转化成容斥处理，

也就是将

$\sum_{i\in A}\sum_{j\in A}[\text{冈伦不喜欢点对 }(i,j)\text{ 或桶子不喜欢点对 }(i,j)]$

变为

$\text{点对总数}-\sum_{i\in A}\sum_{j\in A}[\text{冈伦喜欢点对 }(i,j)\text{ 且桶子喜欢点对 }(i,j)]$

类似所有节点权值相同的情况，我们仍然可以发现，一对点 (i,j) (j,i)只会产生一次不满值，所以我们又能把问题转化为基于无序点对的。

考虑什么样的点对是两人同时喜欢的，在无序点对的前提下，是 i **为** j **祖先，且** $w_i\le w_j$。

也就是初始二人不满值的计算方式是：

$\frac{n(n-1)}{2}-\sum_{i\in A}\sum_{j\in A}[anc(i,j)=1\land w_i\le w_j]$

仍然设出 Δ，仍然不考虑 $\frac{n(n-1)}{2}$ 这一坨的变化（那一坨只与 A 中点数有关）：

$\Delta_x=\sum_{i\in B}[anc(x,i)\land w_x<w_i]+[anc(i,x)\land w_i<w_x]+\sum_{i\in A}[anc(x,i)\land w_x\le w_i]+[anc(i,x)\land w_i\le w_x]-d_x$

由于没有节点权值相同，所以后面的 $\le$ 可以变为 $<$；

类似链的情况，我们合并同类项：

$\Delta_x=\sum_i[anc(x,i)\land w_x<w_i]+[anc(i,x)\land w_i<w_x]-d_x$

对于每个 x 为定值，直接算出来后排序即可。

时间复杂度 $O(n\log n)$。

100 Pts

继续所有节点点权互不相同的情况：

将 $=$ 的部分拆开为：

$\Delta_x=\sum_i[anc(x,i)\land w_x<w_i]+[anc(i,x)\land w_i<w_x]-d_x+\sum_{i\in A}[(anc(x,i)\lor anc(i,x))\land w_i=w_x]$

对于最后一项，我们在链时讨论过，可以类似地得出对于 w 相同的点儿子一定会被先删掉；

所以对于一个点我们只需要默认子树中的 w 相同的点都删了而祖先的没删即可，Δ_x 又变成了定值。

直接排序即可。计算固定值的过程可以使用树状数组。

40 Pts 的部分分是为暴力计算每次哪个节点 Δ 最小设计的。

时间复杂度为 $O(n\log n)$。

时空结构

Task 1

$n!\times n$ 暴力即可。

Task 2

不难发现答案就是所有情况的不稳定度（下称点权）和。

不妨对于每一个点单独计算该点在所有情况下的点权和（下称对答案的贡献）。

不难发现当 $p\neq 1$，除了自身以外会影响 p 号点点权的只有 1 号节点，直接使用组合数计算出有多少种情况满足 1 号点在排列中在 p 之前即可。

时间复杂度为 $O(n)$

Task 3

令根节点深度为 0，对深度为 1 和 2 的点分别讨论它们对答案的贡献。

对于深度为 1 的那些点，只有 1 号点能影响它们，故讨论与 Task 2 相同；

对于深度为 2 的那些点，设当前要计算答案的点为点 p，点 p 的父亲为点 q，点 q 的父亲为 1。

那么对于点 p，有如下这些情况：

1. 点 p 在给自己子树添加点权之前没有受到来自点 q 或 1 的点权贡献；
2. 点 p 在给自己子树添加点权之前受到了来自点 q 的点权贡献，但没有受到来自 1 号点的点权贡献；
3. 点 p 在给自己的子树添加点权之前受到了来自点 1 的点权贡献，但没有受到来自 q 号点的点权贡献；
4. 点 p 在给自己子树添加点权之前受到了来自点 q 和 1 号点的点权贡献。

对于情况 1,2,3，方案数和该点最终点权的计算方式是简单的；

对于情况 4，我们还需要分两类讨论：

1. 点 1 在点 q 之前
2. 点 1 在点 q 之后

讨论之后对于这两种情况的方案数计算也是简单的。

时间复杂度为 $O(n)$

Task 4

仍然考虑沿用 Task 3 的想法，将所有点按照深度分类，由于树为随机，所以树高期望情况下不超过 $\sqrt{n}$。

对于某一个节点 p，不难设从根到 p 的路径上的点**一开始的**点权分别为 $s_1,s_2,s_3,\cdots,s_k$，不难发现**最终** p 的点权一定可以表示为 $\sum_{i=1}^{k}s_ix_i$，其中 x_i 表示从根出发的第 i 个点对点 p 的贡献次数。

考虑如何计算出这个贡献次数。

对于 p 的某个祖先 q，我们忽略所有除去 $p-q$ 路径上的节点，只考虑 $p-q$ 路径上的点，设这条路径长度为 d，并从 $q-p$ 依次将经过的点**重新标号**为 $1,2,\cdots,d$（即链顶为 1，链底为 d），现在的问题变成了给定一条长度为 d 的链，你需要计算出链顶最终对链底作了几次贡献。

由于只有这条链内部自身的点会收到来自链顶的贡献并对 d 造成贡献，不妨将先将除去这条链的其他节点删去，仅考虑长度为 d 的所有排列。考虑某一个排列 P 我们如何计算方案，我们设 dp_i 表示到达排列中的第 i 个元素时，1 号点对**当前元素**（注意，不是 d 号元素）做了几次贡献，不难发现对于这个特定的排列 P，1 号点对 d 号点的贡献就是 $\sum dp_i$（因为 d 是链底，所以每个点在向下贡献点权的时候都会将自身的点权贡献给 d 号点）。

dp 的转移方程是比较显然的：$dp_i=\sum_{j<i\text{且}P_j<P_i}dp_j$，初值为$dp_i=[P_i=1]$

再观察一下，可以发现 dp 的本质就是在计算这个排列中有多少个以 1 开头的递增子序列。

不妨对于**每一种**递增子序列，我们都考虑它在多少个排列中出现了。

令所有可能的递增子序列构成的集合为 S，那么最后答案的计算方式是

$$\sum_{i\in S}\binom{d}{len(i)}(d-len(i))!$$

其中，$len(i)$ 表示 i 这个递增子序列的长度。

不难发现相同长度的递增子序列可以合并起来一起计算，我们容易得到：

$$\sum_{i=1}^{d}\binom{d-1}{i-1}\binom{d}{i}(d-len(i))!$$

其中，$\binom{d-1}{i-1}$ 是在枚举当前的递增子序列选中了哪些元素，由于 1 必须在递增子序列中出现，因此方案数就是 $\binom{d-1}{i-1}$。

这样，设 H 为树高，我们可以 $O(H^2)$ 地计算出每种贡献，令 $g_d=\sum_{i=1}^{d}\binom{d-1}{i-1}\binom{d}{i}(d-len(i))!$，最后答案的计算方式就是：

$$\sum_{i=1}^{n}\sum_{j\in P(i)}g_{dist(i,j)}s_j\binom{n}{dist(i,j)}(n-dist(i,j))!$$

其中，$dist(i,j)$ 表示从 i 到 j 需要经过的点数，$P(i)$ 表示 i 的祖先节点(包括自己) 集合，$\binom{n}{dist(i,j)}(n-dist(i,j))!$ 就是在排列中选出 d 个位置放置 i 到 j 的 d 个节点再任意排列其他节点的方案数。

注意到这里只计算了每个节点在为子树加的时候对自身/其他节点的贡献，我们并没有计算节点一开始的点权，因此最终的答案还应加上 $\sum_{i=1}^{n}s_i\times n!$。

树在随机意义下的最大深度为 $\sqrt{n}$，因此时间复杂度为 $O(n\sqrt{n})$