枇杷树（loquat）

算法零

直接按题意模拟即可。

期望得分 10 分。

算法一

考虑按题意建出每棵树。

对于一棵树中的一条边，它对当前树的答案的贡献即为它两边的子树大小相乘再乘上边权。直接枚举计算每棵树中每条边的贡献即可。

期望得分 30 分。

算法二

仍然考虑用之前的答案推出当前答案。

由于两棵子树中的答案已经统计了，我们只需考虑经过了新连的边 l 的答案即可。首先考虑 l 的贡献，显然为 $w \times siz_x \times siz_y$。然后考虑剩余答案，显然为

$$\begin{aligned}&\sum_{i=0}^{siz_x-1}\sum_{j=0}^{siz_y-1} dis(i,u)+dis(j,v)\\ =&\sum_{i=0}^{siz_x-1} dis(i,u)\times siz_y+\sum_{i=0}^{siz_y-1} dis(i,v)\times siz_x.\end{aligned}$$

记 $f(u,x)=\sum_{i=0}^{siz_x-1} dis(i,u)$，则剩余答案为 $siz_y \times f(x,u)+siz_x \times f(y,v)$，于是

$$ans[i]=ans[x]+ans[y]+w\times siz_x\times siz_y+siz_y\times f(x,u)+siz_x\times f(y,v).$$

考虑求如何快速求出 $f(i,j)$？我们可以使用递归。分类讨论 j 在组成 i 的哪棵子树中，原本就在 T_x 中的点直接递归计算，在 T_y 中的点需要先到 v_i 处，再从 v_i 经过新边到达 j 点，即有：

$$f(i,j)=f(x[i],j)+f(y[i],v[i])+(w[i]+dis(x[i],u[i],j))\times siz_{y[i]}.$$

现在考虑求出在 i 树内的 j,k 点之间的距离 $dis(i,j,k)$。仿照之前 f 的求法，分别讨论 j 和 k 在 $x[i]$ 中还是 $y[i]$ 中即可以求出。

使用记忆化，由于 dis 的状态数只有 m^3 ，故复杂度为 $\tilde{O}(m^3)$。

期望得分 100 分。

上古遗迹（relics）

Subtask 1 & 2

我们记一个 a_i 和 b_i，表示第 i 块石板最多在 $[a_i, b_i]$ 这个区间是最小值，然后在 $[a_i - 1, b_i]$ 和 $[a_i, b_i + 1]$ 都不是最小值了（如果有这两个区间的话）。这个可以用单调栈解决。然后一次询问 (l, r) 的答案就是：

$$ans = \max_{i=l}^{r} \{(\min\{b_i, r\} - \max\{a_i, l\} + 1) \cdot h_i\}.$$

满分做法

离线询问，然后考虑每块石板对询问的贡献。我们发现这里这个 min 和 max 很烦人，于是把它们分类讨论。

首先 $l \le a_i \le b_i \le r$ 和 $a_i \le l \le r \le b_i$ 的情况可以很轻松地用二维偏序（树状数组）处理，接着我们来看剩下两种情况。注意到我们翻转石板以及所有的询问区间后 $l \le a_i \le r \le b_i$ 和 $a_i \le l \le b_i \le r$ 这两种情况是等价的，所以我们现在来讨论 $a_i \le l \le b_i \le r$ 这种情况。

对于这种情况，我们的式子变成了：

$$(b_i - l + 1) \cdot h_i = (b_i + 1) \cdot h_i - h_i l$$

发现了什么？这是一个一次函数的形式，其中 l 是询问区间的一个变量，而 $(b_i + 1) \cdot h_i$ 是一块石板对应的一个常数。也就是说，我们把每一块石板转化为一条形如 $f(x) = (b_i + 1) \cdot h_i - h_i x$ 的直线，那么对于每一个询问 (l, r) 我们就是要求所有 $a_i \le l$ 且 $b_i \le r$ 的石板对应的直线中 $f(l)$ 的最大值。

我们先来看这个子问题：维护一个一次函数的集合，支持插入直线和对于 x 查询所有直线中 $f(x)$ 的最大值。使用李超线段树时间复杂度是 $O(\log n)$ 的。

然后我们有了这个工具后来看原问题就很简单了，我们只需要在 cdq 分治的过程中动态插入直线并求出 $f(x)$ 最大值更新答案即可。时间复杂度 $O(n \log^2 n)$。

吞天得手（ttds）

Subtask 1

可以直接枚举出所有可能的队列情况，再进行比较和计算。时间复杂度 $O(n2^n)$。

Subtask 3

k 相当小，可以直接考虑暴力找到最小的几个数将他们暴力合并在一起凑够 k 个即可。

Subtask 2&4

我们考虑当前已经有了一部分最优秀（即字典序最小）的字符串，由于字典序的性质，我们肯定考虑往这些字符串的后面加一些字符让他们继承父亲的优秀。于是我们可以发现他们的继承方式大概可以构成一个 DAG，而我们的转移是在 DAG 上进行 bfs。

具体来说，我们现在有一些原序列上位置的**有序集合**，这个集合中的位置表示了一些以这些位置结尾的相当优秀的前缀，而且这些前缀是相同的（即这些集合中的位置的权值也是相同的）。于是我们考虑往这个前缀中加一个原序列中在集合第一个元素之后（即最靠前的位置）的最小值（这个最小值可能会有多个，它位置可能会在集合中）, 我们把这些最小值的位置称为决策集合，然后得到由往前缀上加上决策集合中的一个值的新前缀。这个新前缀无疑是当前最优的（因为字符串的大小完美地满足 bfs 性质），然后我们再把决策集合作为新的有序集合（它承载了所有的新前缀），递归地进行下去直到找满了 k。如果没有找够 k 个，我们就回溯到这一层，用现有的有序集合找到原序列后缀的次小值，第三小值……直到找满 k 个为止。如果原序列后缀已经用完了，就返回。

于是实现分为两部分，第一部分处理出原序列中一个位置的下一个和自己值相同的位置，时间复杂度 $O(n)$。

第二部分考虑对一个有序集合找到原序列中后缀的第 k 大值。

在 Subtask 2 中，可以直接暴力预处理，时间复杂度 $O(na_i + n\log k)$。

在 Subtask 4 中，主席树套上就行了，时间复杂度 $O(n\log k + k\log a_i)$。