题解

T1 不

题目中给出的匹配贡献的 $\max$ 形式似乎不太方便直接维护的，考虑对其强制约束一个偏序关系，即分类讨论拆掉贡献中的 $\max$ 。

对于一组匹配的物品 x 和 y ，其贡献为 $\max\{a_x+a_y,b_x+b_y\}$ ，因此当 $a_x+a_y\geq b_x+b_y$ 的时候，贡献为 a_x+a_y ，否则贡献为 b_x+b_y 。

由于这两种情况是对称的，因此考虑 $a_x+a_y\geq b_x+b_y$ 的处理方式，另一种情况只需要类似的处理即可。移项得到当 $a_x-b_x\geq b_y-a_y$ 时贡献为 a_x+a_y ，此时物品 x 和物品 y 分别来自集合 L 和 C ，因此对于来自集合 L 的物品 x ，将其属性 c_x 定义为 a_x-b_x ，对于来自物品集合 C 的物品 y ，将其属性 c_y 定义为 b_y-a_y 。

考虑使用线段树维护这个过程，类似最大子段和的维护方式，即在线段树节点合并左右儿子的信息时，首先将左右儿子的最优贡献合并到该节点，然后分别考虑 集合 L 的物品在左儿子且集合 C 的物品在右儿子 和 集合 L 的物品在右儿子且集合 C 的物品在左儿子 的情况，因此分别维护线段树子树中集合 L 和集合 C 中的最小的 a 和 b ，按照上述过程进行合并即可。

需要注意的是可能会有来自不同集合的物品属性值相同，此时任意给其钦定一个顺序即可。时间复杂度 $O(n\log n)$ 。

T2 远

首先当 $a_i=1$ 时将其放置在划分所在的段的末尾即可，因此只需考虑 $a_i>1$ 时的序列。

当一个段的长度 $>\log v$ 时，由于 $a_i>1$ ，因此该段按照任意顺序重排进行函数复合的结果一定会 $>c$ ，因此每一个段的长度是 $O(\log v)$ 级别的，因此设 $dp\{i,j\}$ 表示当前处理了前 i 个函数，划分了 j 个段，转移时枚举一段的长度，并在该段内考虑贪心计算代价，时间复杂度 $O(n^2\log n\log v)$ 。

转移可以预处理每个段的代价，因此时间复杂度为 $O(n\log^2 n\log v+n^2\log n)$ ，可以使用凸优化技巧做到更优时间复杂度。

T3 万

首先考虑当排列 p 固定时如何计算方案数，不妨将长度为 $n-1$ 的操作序列单独考虑，当有 $i<p_i$ 时，说明原本在位置 i 上的元素会通过交换操作一步一步的抵达 p_i 这个位置，也就是说这要求对于交换 $(i,i+1),(i+1,i+2),\dots,(p_i-1,p_i)$ 时对于其中的每一对相邻的 $(j,j+1)$ 和 $(j+1,j+2)$ ，我们均要求操作 $(j,j+1)$ 的时间早于操作 $(j+1,j+2)$ 的时间。同时，还要求满足 $(i,i+1)$ 的操作时间早于 $(i-1,i)$ 的操作时间，以及 (p_i-1,p_i) 的操作时间早于 (p_i,p_i+1) 的操作时间。当 $p_i<i$ 时，类似的处理。

可以发现此时对于相邻的操作 $(i,i+1)$ 和 $(i+1,i+2)$ ，我们可以得知两个操作的相对先后顺序，且仅有相邻操作会有先后限制，考虑使用动态规划，设 $dp\{t,i\}$ 表示这 $n-1$ 个操作的前 t 个操作已经安排了一个长度为 t 的先后顺序，此时第 t 个操作在这前面的 t 个操作中相对操作位置为 i ，则若第 t 个操作早于第 $t+1$ 个操作则 dp 转移等价于前缀和，若第 t 个操作晚于第 $t+1$ 个操作则 dp 转移等价于后缀和。因此排列固定时可以做到 $O(n^2)$ 。

接下来考虑排列不固定的情况，先将所有能够确定的相邻操作相对顺序进行确定，对于不确定先后相对顺序的相邻操作，可以发现我们任意钦定其先后顺序均可，即我们在该处转移时既做前缀和也做后缀和，即所有的 dp 的值在当前不受限制的 t 处均变为全局的整体的 dp 值的和。因此按照上述方式动态规划转移，时间复杂度 $O(n^2)$ 。

T4 里

题目要求维护基环树 (i,a_i) 的邻域相关信息，定义点 i 的子邻域为满足 $a_j=i$ 的 j 构成的集合，父邻域为 a_i 单独构成的集合。

注意到如果将一个点的权值理解为其子邻域的贡献和加上其父邻域的贡献和，则题中修改邻域贡献，查询单点贡献可以转化为修改父邻域贡献和查询邻域贡献和，由于修改时只有 $O(1)$ 个点的子邻域贡献会发生改变，因此修改 $a_i\leftarrow j$ 时，取出 i,a_i,a_{a_i},j,a_j ，分析其新加入的边和新删除的边的对应贡献，需要注意加边和删边的顺序。

查询时查询该点的子邻域贡献和，对于父邻域单独计算即可。

查询全局权值的 $\max$ 和 $\min$ 可以将点 i 的权值放在 a_i 处考虑，对于每一个 i ，将其子邻域中权值的 $\max$ 和 $\min$ 加上来自 i 自身的贡献后，加入数据结构中维护即可。时间复杂度 $O(n\log n)$ 。