array

注意到 $a_i \le 10$。

枚举每类数，强制这类数在区间 $[L,R]$ 中出现次数大于一半。

假设当前枚举的数为 c 。

那么把 $a_i=c$ 看成 $+1$ ，$a_i \neq c$ 看成 -1，满足条件的区间等价于区间和大于 0 。

设前缀和数组为 b_i ，那么有 $b_R > b_{L-1}$，使用归并排序，线段树或者树状数组求出这样的逆序对个数即可。

时间复杂度 $\mathcal{O}((\max a_i) n \log n)$。

fac

由于每个元素的答案是独立的，所以只需要对于每个 A_i 求解答案然后求和即可。

设 $f(x,k)$ 表示 x 经过操作 k 次后的答案，那么有：

$$f(x,k)=\sum_{d|x} f(d,k-1)$$

直接递推即可做到 $\mathcal{O}(Vk\log V)$。其中 $V=\max a_i$。

对于 A_i 质因数只有一种，直接递推即可。

$$f(p^x,k)=\begin{cases}1 & \text{if}(x=0)\\ p\times f(p^{x-1},k)+\binom{k-1+x}{x} & \text{otherwise}\end{cases}$$

对于 $n=1$ 的问题，可以把所有转移建边。得到的图除去自环就是一张 DAG 。那么问题变成从 A_1 出发，走 k 步到达的点权和。

设 $g(x,i)$ 表示没有走过自环，走 i 步到达 x 的方案数。然后把剩下 $k-i$ 步分配到每个点走自环数量上，插板法即可。

最后发现这个函数是积性函数，即对于 $\gcd(x,y)=1$, 有 $f(x,k)\times f(y,k)=f(xy,k)$。

所以预处理所有 p^x 处的值即可。

per

设 m 表示最大的删除元素，则删除的数字必须 $\le m$。

把 $>m$ 的元素在 p 中标记出来，设为 $q_1,q_2,\dots,q_{n-m}$，特殊的，令 $q_0=0, q_{n-m+1}=n+1$ 。

则任何被删除的子段都必须完全位于 q_i+1 和 $q_{i+1}-1$ 之间。

并且从某个 $[q_i+1,q_{i+1}-1]$ 中最多可以选择一个子段进行删除。

因此，我们最多可以选择 $n-m+1$ 个需要删除的子段（加上前后缀的部分），但由于 $>m$ 的元素只有 $n-m$ 个，所以删除 $n-m+1$ 个子段是不可能的。

另一方面，如果选择删除 $k\le n-m$ 个子段，那么总是有可能找到一种方案完全删除它们。

因此，我们只需要找到选择 $k \leq n-m$ 个子段的方案数，满足每个 $[q_i+1, q_{i+1}-1]$ 之间只有一个子段，且 m 被包含在某一个子段中。

考虑容斥，每一个 $[q_i+1, q_{i+1}-1]$ 选一段的方案数（可以不选）乘起来，然后减去 $n-m+1$ 个非空子段的方案。

此时复杂度可以做到 $\mathcal{O}(n^2)$ 。

最后用 set 或者并查集动态维护这个过程中的答案即可，时间复杂度 $\mathcal{O}(n\log n)$ 或者 $\mathcal{O}(n\alpha(n))$ 。

game

设 T_i 表示第 i 关的树。

首先需要明白如何计算单局游戏的最大得分。

由于距离 u 最远距离的点一定是直径端点之一，所以直径是非常重要的。令 $d_i=(x_i,y_i)$ 表示 T_i 的直径两个端点。（有多组任取合法的即可）

那么方案就是先把除了 $path(x_i,y_i)$ 上的点全部删完，然后再删这个直径。

由于 T_{i+1} 是由 T_i 加一个顶点 u 得到，所以 D_{i+1} 将是 $\{x_i,y_i,u\}$ 其中两个。为了方便，除非以 u 为端点的直径严格更大，否则不会选择 u 为端点。

假设已知 T_i 的分数，考虑如何计算 T_{i+1} 的分数。若 $D_{i+1}=D_i$，那么 u 将贡献 $\max(dis(x_i,u),dis(y_i,u)$，然后被删掉。

否则 u 是直径的一个端点。假设 p_u 是 u 所连的结点，说明 p_u 可以成为 T_i 的一个直径端点。设 x 为 p_u 对应的另一个端点，那么 D_i 可以为 (x,p_u) 。（选择任何一条直径，最大分数都不会变）

说明在 T_i 时，不在 $path(x,p_u)$ 上的结点 v 贡献了 $\max(dis(v,x),dis(v,p_u))$，最后移除整条直径。重要的是对于每个 v ，到底是 x 更远还是 p_u 更远。

考虑动态维护直径中点（可能在边上，也可能在点上）。那么一次长度增加会从边到点或者从点到边。

先考虑从边到点，假设两个集合分别是 S_1,S_2，那么会有一个集合距离突然变远，只需要加上这个集合大小即可。

再考虑从点到边，假设当前中点为 m ，往子树 m' 走半步，那么只需要除了子树 m' 的所有点贡献加一。

这些点数都可以用数据结构动态查找子树和解决。时间复杂度 $\mathcal{O}(n\log n)$。