A

搜索题，dfs 中记录当前变换层数与需要变换的初始元素，递归输出直到达到 m 个数。

优化的入手观察是 S_k 的前 k 个数一定是 1，因此 k 很大是没用的。

std 使用的优化：如果初始元素为 1 则无论变换多少次都只有一个 1；输出的所有数都是初始的 x 的因子。

B

首先尝试解决单个询问。也就是对于特定元素 x 的答案。

假设最终搜索中在节点 i 终止。那么这条路径是可以唯一确定的，长度为 $\mathcal{O}(\log n)$。

那么就可以在这条路径上有 L 个元素小于 x ，R 个元素大于 x ，E 个元素等于 x 。（E 始终都是 0 或 1 ，这取决于终止节点是否为叶节点。）

接下来考虑一个 $1 \le y < x$ 对 x 的贡献。那么有 L 种方式放置在路径上。

接下来有 $x-2$ 个小于 x 的数，需要选 $L-1$ 个填进来，方案数为 $\binom{x-2}{L-1} \cdot (L-1)!$。

大于 x 的地方的贡献为 $\binom{n-x}{R} \cdot R!$。

若 $E=1$ ，则要求 x 必须放在节点 i ，否则没有要求。

然后其他的元素都可以随机排列，即 $(n-L-R-E)!$ 种方案。

所以贡献为：

$$y \cdot L\binom{x-2}{L-1}\binom{n-x}{R}(L-1)!R!(n-L-R-E)!$$

而 $y > x$ 是类似的。所以能在 $\mathcal{O}(n)$ 的时间复杂度内解决单个问题。

但是实际的 i 是哪个节点根本无所谓，我们只关心 L, R, E 这三个数。于是首先把每个节点的三元组预处理出来，最多只有 $\mathcal{O}(\log^2 n)$ 个。

所以最后复杂度为 $\mathcal{O}(n\log n + m\log^2 n)$。

C

考虑 n 个点的无向图，若 $A_i \& A_j = 0$ 则添加一条边 (i, j)。

对于某个连通块包含了顶点 $\{v_1, v_2, \ldots, v_k\}$ ，那么无论如何交换值都只会在内部置换。

显然希望这个连通块内部能按照下标大小排序，事实证明这也是可以做到的。

现在的问题是可能有 $\mathcal{O}(n^2)$ 条边，这个复杂度太高了。

但是由于只关心连通性，不关心具体有哪些边，如果能找到一个边更少的图就可以解决这个问题。

我们用 x' 表示 $(2^{20}-1) \oplus x$ ，那么 $x \& y = 0$ 等价于 y 是 x' 的子集。

于是 x 和 x' 间连双向边，然后 x' 向子集内连单向边（只需要向某一位 1 变成 0 的所有情况连边，即最多 20 个）

这样的话边数只有 $2^{20} \times 10 + 2n$ 条边左右。对这个图跑 Tarjan 求 scc 即可。

D

$k_i=r_i-l_i+1,\prod k_i=K$。同时将树上边权改为 1 。

考虑枚举产生贡献的点对 (i,j) 以及它们此时的 c，那么贡献为：（下面的式子把 $i=j$ 的减去再除以 2 就是答案）

$$ans=\sum_c\sum_{c\in[l_i,r_i]}\sum_{c\in[l_j,r_j]}d(i,j)\frac{K}{k_ik_j}$$

把树上距离拆成和 LCA 有关的形式，用 d_i 表示点 i 的深度，即：

$$ans=\sum_c\sum_{c\in[l_i,r_i]}\sum_{c\in[l_j,r_j]}(d_i+d_j-2d_{LCA})\frac{K}{k_ik_j}$$

接下来就是一堆式子化简：

$$ans=K\sum_c\sum_{c\in[l_i,r_i]}\frac{1}{k_i}\sum_{c\in[l_j,r_j]}(d_i+d_j-2d_{LCA})\frac{1}{k_j}$$

把括号拆开分成三部分，分别是：

$$\sum_{c\in[l_i,r_i]}\frac{d_i}{k_i}\sum_{c\in[l_j,r_j]}\frac{1}{k_j}$$
$$\sum_{c\in[l_i,r_i]}\frac{1}{k_i}\sum_{c\in[l_j,r_j]}\frac{d_j}{k_j}$$
$$-\sum_{c\in[l_i,r_i]}\frac{1}{k_i}\sum_{c\in[l_j,r_j]}\frac{1}{k_j}2d_{LCA}$$

前两部分是简单的，用扫描线即可解决。枚举 c 即可。

至于最后部分，求两个点 x,y 的 LCA 深度相当于求有多少点满足是 x,y 的公共祖先。

那么对 x 到根节点路径上的所有点 $+1$ ，查询 y 到根节点路径上所有点权值即可。

那么还是扫描线枚举 c ，当有新的点 x 加入集合或者从集合里被删除时，就涉及到计算 x 与其他所有点的 d_{LCA} 之和了。用上述描述的方式路径加、路径求和即可。

具体的，若新加入一个点 x ，那么把 x 到根节点路径所有节点权值之和乘 $\frac{2}{k_i}$ 加入答案中，然后让这条路径上所有点权值加上 $\frac{1}{k_i}$。（注意第三种计算的是 $(i<j)$ 的所有点对答案，但是前两种视你的实现方式可能计算 $(i\ge j)$ 的答案，实现的时候注意一下细节即可）。

删除同理。

时间复杂度 $\mathcal{O}(n\log^2 n)$。