A

对于一个节点，假设它在先序遍历中第 i 个出现，后序遍历中第 j 个出现，那么它取 0 带来的收益是 0，它取 1 带来的收益是 $2^{n-i}-2^{n-j}$。

所以采用贪心的策略，对于每一个节点，如果在先序遍历中的出现时间早于在后序遍历中的出现时间，那么就取 1，否则就取 0。把每一个节点的权值加起来即可。

时间复杂度 $O(n)$。

B

设第 i 个叶子节点在最后的取值是 x_i，则我们可以把根节点的取值表示出来，它一定是一个关于 $x_1,\dots,x_n$ 的不超过 n 次的多项式 F。其中每一项一定是 $\{x_1,\dots,x_n\}$ 中若干个不同的变量的乘积。

注意到最后填入数字时是完全对称的，因此对于 F 中包含了 k 个变量的一项，无论它是哪些变量的乘积，它对总和的贡献总是一样的。

因此做法分成两步，第一步计算出从 a 中选出 k 个数字乘起来的所有方案的权值和，状态 $f[i][j]$ 表示前 i 个数选了 j 个出来相乘的所有方案的权值和。第二步是计算出 F 中次数为 i 的项有多少个，$g[i][j]$ 表示以 i 为根的子树中次数为 j 的项有多少个，加法对应的就是向量加，乘法对应卷积。

两者都是经典的 $O(n^2)$ 的动态规划。

C

即对每一个 $[1,r]$ 中的质数 p，计算它在 $[l,r]$ 内 p 的最高次数。

当 $p\le 10^7$ 时，这样的质数数量很少，可以暴力。

当 $p>10^7$ 时，显然 p 的最高次数是 1，因此只要判断 $[l,r]$ 中有没有 p 的倍数就可以了，这个的条件是 $\lfloor\frac{l-1}{p}\rfloor\neq\lfloor\frac{r}{p}\rfloor$。不难发现 $(\lfloor\frac{l-1}{p}\rfloor,\lfloor\frac{r}{p}\rfloor)$ 的取值一共有 $O(\sqrt{r})$ 对，因此只需要对每一个合法区间计算里面的所有质数的乘积。因为模数给定，所以可以分段打表。

D

不难发现，对于同一棵树和相同的操作集合，任意交换操作的顺序树的最终形态都不会发生改变。如果对点集 S 进行压缩， 那么最终的结果是 $\{S,root\}$ 虚树经过的所有点的父亲都变成了根节点，其他节点的父亲节点不变。

令 $size[u]$ 表示 u 的子树内有多少个节点，设集合 T 的所有点的父亲都变成了根节点，那么对总深度的影响是减去了 $\sum_{u\in T}size[u]$。

设在结束时，第 i 棵树被压缩的节点集合是 S_i，对于一个在区间 $[l,r]$ 上的压缩操作，它只会在 (S_{l-1},S_l) 和 (S_r,S_{r+1}) 间产生区别。因此我们依次考虑每棵树，问题就变成了加入、删除节点，询问虚树经过的所有节点的权值和。

这是一个比较传统的数据结构问题，按照 DFS 序用 set 维护当前的点集，设插入的点是 k，他的前驱是 l，后继是 r，$\mathrm{LCA}(l,k),\mathrm{LCA}(r,k)$ 中深度较大的点是 f，那么这次插入产生的新路径是 (k,f) 间的路径。对于删除操作，则是删除抹去的路径是 (k,f) 之间的路径。

总时间复杂度为 $O(n\log n)$（假设 n,m,q 同阶）。