

## A

记大小为  $n$  的深度自同构的树数量为  $f_n$ ，则答案为

$$\sum_{d|n} f_d$$

每一棵树的每个子树也应该是相同的，故

$$f_n = \sum_{d|(n-1)} f_d$$

枚举倍数即可  $O(n \log n)$  计算。

## B

需要维护每个点到当前根的距离  $f_u$  以及每个根节点到叶子的最长距离  $g_u$ ，则每次合并的时候更新  $u$  的根节点的最长距离

$$g_{root} = \max(g_{root}, f_u + g_v + 1)$$

同时更新  $v$  子树里面每个节点到新根的距离，即加上  $f_u$ 。

寻找根节点的操作与更新操作都可以使用带权并查集完成，单次复杂度即为并查集复杂度。

## C

集合里面加入数字后 gcd 若改变则至少除 2，所以不同的 gcd 数量最多  $O(n \log a)$  级别。

固定左端点，通过倍增预处理区间 gcd 可以在  $O(n \log n \log a)$  的时间复杂度内找出每一个 gcd 发生变化的右端点，记录每一个这样的极大区间。

枚举 gcd，考虑以  $f_i$  代表  $1 \sim i$  中 gcd 等于枚举的数的不交区间集的方案数，这是个简单的 DP，可以将区间按左端点排序后利用线段树优化。复杂度与区间个数有关，总复杂度均摊下来也是  $O(n \log n \log a)$ 。

对后缀按相同方式做一遍，则包含  $i$  位置的方案数就是总数减去只在  $[1, i-1], [i+1, n]$  中选区间的方案数，后者可以直接通过前后缀计算得到的方案数乘算获得。

## D

对  $n$  个串建立 Trie 树，每次打字相当于在 Trie 上走一步，不能走出去，走到单词的结尾就删去对应单词并回到起点。

假设一个点的左儿子子树大小为  $x$ ，右儿子子树大小为  $y$ ，则共需经过此节点  $x+y$  次，并且之后的一步需要有  $x$  次打出 0， $y$  次打出 1。

容易发现每个节点的决策是独立的，仅跟之前经过此节点时的后续情况有关，记  $F(x, y)$  为一个节点在还需要打出  $x$  次 0 和  $y$  次 1 时的成功概率，转移如下：

$$F(x, y) = \max\{pF(x-1, y) + (1-p)F(x, y-1), pF(x, y-1) + (1-p)F(x-1, y)\}$$

最后的概率就是每个节点在初始条件下的成功概率之积。