Problem A. Common Edges

对原图求边双连通分量并缩点，可以证明两条路径的最少公共边数量为边双树上两条路径的最少公共边数量。显然两条路径都必须经过边双树上的所有边，因此只需证明边双连通分量内部任意四点（记为 u,v,x,y）可以构造出两条没有公共边的路径：

由于 u,x 在同一边双内，可以设 $u \overset{P_1}{\rightsquigarrow} x, u \overset{P_2}{\rightsquigarrow} x$ $(P_1 \cap P_2 = \varnothing)$，$v \overset{Q}{\rightsquigarrow} y$。分以下几种情况讨论：

- $P_1 \cap Q = \varnothing$：取 $u \overset{P_1}{\rightsquigarrow} x,\ v \overset{Q}{\rightsquigarrow} y$；
- $P_1 \cap Q \neq \varnothing$：设 Q 与 $P_1 \cup P_2$ 第一次交与点 p，最后一次交于点 q，设 $v \overset{Q_s}{\rightsquigarrow} p \overset{Q_m}{\rightsquigarrow} q \overset{Q_e}{\rightsquigarrow} y$。
  - p,q 同时属于 P_1 或 P_2，不妨设 $p,q \in P_1$：取 $u \overset{P_2}{\rightsquigarrow} x,\ v \overset{Q_s}{\rightsquigarrow} p \overset{P_1}{\rightsquigarrow} q \overset{Q_e}{\rightsquigarrow} y$；
  - p,q 分别属于 P_1 和 P_2，不妨设 $p \in P_1, q \in P_2$：取 $u \overset{P_2}{\rightsquigarrow} q \overset{Q_e}{\rightsquigarrow} y,\ v \overset{Q_s}{\rightsquigarrow} p \overset{P_1}{\rightsquigarrow} x$；

计算缩点后边双树上 u 到 x 和 v 到 y 的公共边数量。记 $\mathrm{lca}(u,v), \mathrm{lca}(u,y), \mathrm{lca}(v,x), \mathrm{lca}(x,y)$ 中较深的两个点为 a,b，答案为 $\mathrm{dep}_a + \mathrm{dep}_b - 2\,\mathrm{dep}_{\mathrm{lca}(a,b)}$。再计算 u 到 y 和 v 到 x 的公共边数量取最小值即可。

Problem B. Contest Preparation

签到。分类讨论：

- 当 $n = 0$，答案显然为 0；
- 当 $0 < n \le m$，答案为 2，因为要先造完题才能验题；
- 当 $n > m$，需要的总时长为 $2n$。尽量将所有任务均分给 m 个人，答案为 $\lceil 2n/m \rceil$。

Problem C. Diameter

从 i 到 j 的距离等于从 0 到 $(j-i) \bmod 2^n$ 的距离，因此我们只需要计算从 0 出发最远的距离与距离 0 最远的点数。

求 0 到 x 的距离等价于对 0 在 $\bmod 2^n$ 下进行 $\pm 2^s$ $(s \in S)$ 使之变为 x 的最少操作次数。很明显交换两个操作对操作次数没有影响，因此可以假设所有操作按 s 从小到大进行。

考虑对 y 进行一系列操作使其变为目标数 x。由于进行 $\pm 2^s$ 操作之后不会改变第 0 到 $s-1$ 二进制位的值，因此在进行 $\pm 2^s$ 前必须保证操作数的前 $s-1$ 位与目标数相同，即 $y_{0:s} = x_{0:s}$（$x_{a:b}$ 表示 x 的二进制表示中第 a 到 $b-1$ 位）。

同时为保证操作次数最少，操作 $\pm 2^{S_i}$ 进行的次数一定小于 $2^{S_{i+1}-S_i}$，否则该操作可以用一次 $\pm 2^{S_i+1}$ 替代。因此在进行 $\pm 2^{S_i}$ 操作时，操作数的第 S_{i+1} 位之后一定全为 0 或 1，即 $y_{S_{i+1}:n} \in \{0, 2^{n-S_{i+1}} - 1\}$。

假设 S 内元素从小到大排序，可以证明 $S_0 = 0$，否则图不连通。令 $S_m = n$。考虑一个确定的数 x，根据上面得到的结论，可以通过 dp 计算 0 到 x 的距离：

用 $f_{h,i}$ $(h \in \{0,1\}, 0 \le i \le m)$ 表示使得二进制表示下第 S_i 位之前的部分与 x 相同，且第 S_i 位及之后的数全为 h 所需要的最少操作次数。有下面几种转移：

- $f_{0,i+1} = f_{0,i} + k$，其中 $k = x_{S_i:S_{i+1}}$：进行 k 次 $+2^{S_i}$ 操作；

- $f_{0,i+1}=f_{1,i}+k$：进行 $k+1$ 次 $+2^{S_i}$ 操作（第一次操作后 $x_{S_i:n}$ 向上溢出变为全 0）；
- $f_{1,i+1}=f_{1,i}+2^{S_{i+1}-S_i}-1-k$：进行 $2^{S_{i+1}-S_i}-1-k$ 次 -2^{S_i} 操作；
- $f_{1,i+1}=f_{0,i}+2^{S_{i+1}-S_i}-k$：进行 $2^{S_{i+1}-S_i}-k$ 次 -2^{S_i} 操作（第一次操作后 $x_{S_i:n}$ 向下溢出变为全 1）；

综上有

$$\begin{cases} f_{0,i+1}=\min(f_{0,i},f_{1,i}+1)+k \\ f_{1,i+1}=\min(f_{0,i}+1,f_{1,i})+2^{S_{i+1}-S_i}-1-k \end{cases}$$

初始状态为 $f_{0,0}=0$，$f_{1,0}=1$；从 0 到 x 的最短距离为 $\text{ans}_1=\min(f_{0,m},f_{1,m})$。

接下来计算 ans_1 的最大值以及使得 ans_1 最大的方案数。令

$$\begin{cases} a_i=f_{1,i}-f_{0,i} \\ b_i=1-(f_{1,i}+f_{0,i})+\sum\limits_{0\le j<i}(2^{S_{j+1}-S_j}-1) \end{cases}$$

初始状态为 $a_0=1,b_0=0$。可以计算得到

$$\begin{cases} a_{i+1}=2^{S_{i+1}-S_i}-1-2k+\min(a_i,1)-\min(a_i+1,0) \\ b_{i+1}=b_i+\max(a_i-1,0)+\max(-a_i-1,0) \end{cases}$$

可以证明对于任意 i 存在 k 使得 $|a_i|\le 1$，即使得 $b_{i+1}=b_i$。又由于 $b_{i+1}\ge b_i$，故 $\min_x b_m=0$，即 $\max_x(f_{0,m}+f_{1,m})=-m+1+\sum_{0\le i<m}2^{S_{i+1}-S_i}$，因此

$$\text{ans}_1=-\left\lfloor\frac{m}{2}\right\rfloor+\sum_{0\le i<m}2^{S_{i+1}-S_i-1}$$

注意到对于任意 i，当 $|a_i|>3$ 或 $b_i>1$ 时 ans_1 一定无法取到最大值，因此可以用 $g_{i,a,b}$ $(|a|\le 3,0\le b\le 1)$ 记录使得 $a_i=a,b_i=b$ 的方案数，最后答案为

$$\text{ans}_2=2^{n-1}\cdot(g_{m,0,1}+g_{m,-1,0}+g_{m,0,0}+g_{m,1,0})$$

也可以把 dp 式手动化简，得到

$$\text{ans}_2=2^{n+\lceil m/2\rceil-2}\cdot\begin{cases}\dfrac{m}{2}+2+\sum_{0\le 2i<m}[S_{2i+1}\ne S_{2i}+1], & 2\mid m \\ 1, & 2\nmid m\end{cases}$$

不包括排序复杂度 $\mathcal{O}(m)$。

Problem D. Difference

正解要求复杂度为 $\mathcal{O}(n\log V)$，原则上不允许 $\mathcal{O}(n\log V\log n)$ 通过。

显然要求第 k 大通常转化为二分函数值 x 然后求有多少个区间函数值比 x 大。

当 x 已知时，对于一个区间左端点 l，容易发现满足 $f(l,r-1)<x$ 而 $f(l,r)\geq x$ 的 r 是单调的：即 l_1 对应的右端点为 r_1，l_2 为 r_2，则 $l_1<l_2$ 必有 $r_1<r_2$。因为随着区间左端点的右移，即使 $\max\{a_j\}-\min\{a_j\}$ 不变，$(r-l+1)$ 项也会减少，因而 $f(l,r)>f(l+1,r)$。

因而基于以上的单调性，可以使用双指针（尺取法）来 $\mathcal{O}(n)$ 的找到每个 l_i 对应的 r_i，那么大于等于 x 的个数即为 $\sum_{1\leq i\leq n}(n-r_i)$。总复杂度 $\mathcal{O}(n\log V)$，V 为值域 1×10^{14}。

Problem E. Ellipse

在复平面内考虑所有点。不妨令 $M=0$（否则令所有 $P_i\leftarrow P_i+M$）。对 P_i 进行 DFT，即令 $Z_s=\sum_i P_i\omega_n^{-si}$ $(0\leq s<n)$，则对于每次操作有

$$\begin{aligned}
P_i &\leftarrow \frac{1}{3}\left(P_i+\frac{P_{i-1}+P_{i+1}}{\cos(2\pi/n)}\right)\\
Z_s &\leftarrow \sum_i \frac{\omega_n^{-si}}{3}\left(P_i+\frac{P_{i-1}+P_{i+1}}{\cos(2\pi/n)}\right)\\
&= \frac{\sum_i P_i\omega_n^{-si}}{3}+\frac{\sum_i P_{i-1}\omega_n^{-s(i-1)}\omega_n^{-s}+\sum_i P_{i+1}\omega_n^{-s(i+1)}\omega_n^{s}}{3\cos(2\pi/n)}\\
&= \left(\frac{1}{3}+\frac{\omega_n^s+\omega_n^{-s}}{3\cos(2\pi/n)}\right)Z_s
\end{aligned}$$

进行 k 次操作后有

$$Z_s\leftarrow\left(\frac{1}{3}+\frac{2\cos(2\pi s/n)}{3\cos(2\pi/n)}\right)^k Z_s$$

对 Z_s 进行 IDFT 后有

$$P_i\leftarrow\frac{1}{n}\sum_s\left(\frac{1}{3}+\frac{2\cos(2\pi s/n)}{3\cos(2\pi/n)}\right)^k Z_s\omega_n^{si}$$

由于 $n\geq 7$，$\cos(2\pi/n)>1/2$。因此当 $1<s<n-1$ 时有

$$\left|\frac{1}{3}+\frac{2\cos(2\pi s/n)}{3\cos(2\pi/n)}\right|<1$$

故当 $k\to\infty$ 时，$P_i'=(Z_1\omega_n^i+Z_{n-1}\omega_n^{-i})/n$，求出所有点的坐标即可。复杂度 $\mathcal{O}(n)$。或者计算出 P_i' 所在椭圆的面积为 $\pi(|Z_1|^2-|Z_{n-1}|^2)/n^2$，根据圆与内接正多边形的面积比可以计算出答案为

$$\frac{|Z_1|^2-|Z_{n-1}|^2}{n}\cdot\sin\frac{2\pi}{n}$$

Problem F. K-th Power

首先将问题转化为求解 $[1,x]$ 范围内不具有 p^k 因子的数字个数。

考虑容斥，使用莫比乌斯函数。首先将全集纳入考虑范围（即 $\mu(1)=1$），若 a 为质数，则需要从集合中挖去所有它的倍数 ka^i，即减去，乘以系数 $\mu(p)=-1$；若 $a=\prod_{i=1}^k p_i$，那么它在被考虑之前会被它的所有因子容斥过一次。由于最终这个数字不能在最终集合中，因而 $\sum_{d|a}\mu(d)=0$。若 a 有平方因子 p 则它已经被 a/p 容斥过了这个数字就不考虑，因而 $\mu(a)=0$ — 与莫比乌斯函数定义相同。因而最终答案为：

$$\sum_{i=1}^{\sqrt[k]{x}}\mu(i)\left\lfloor\frac{x}{i^k}\right\rfloor$$

对于 $k=2$ 的情况，乘方的计算可以认为是 $\mathcal{O}(1)$ 的；对于 $k\geq 3$，$\sqrt[k]{x}\leq 1\times 10^5$，因快速幂的 $\mathcal{O}(\log k)$ 可以轻松通过。最坏复杂度出现在 $k=2$，为 $\mathcal{O}(\sqrt{n})$。

Problem G. Nerdle

由于合法的表达式数量不多，可以直接把所有合法表达式求出来之后根据模式串筛选（当然也可以边构造边筛不过难写一些）。

分别用 N_l, M_l, A_l 表示长度为 l 的数字、乘法表达式、加法表达式的 (表达式字符串 s, 值 v) 集合。N_l 可以直接暴力构造，M_l 和 A_l 可以通过下面的转移构造：

$$\begin{aligned}M_{l_1+l_2+1}=&\{(s_1\texttt{*}s_2,v_1v_2)\mid(s_1,v_1)\in M_{l_1},(s_2,v_2)\in N_{l_2}\}\\&\cup\{(s_1\texttt{/}s_2,v_1/v_2)\mid(s_1,v_1)\in M_{l_1},(s_2,v_2)\in N_{l_2},v_2\mid v_1\}\end{aligned}$$

$$\begin{aligned}A_{l_1+l_2+1}=&\{(s_1\texttt{+}s_2,v_1+v_2)\mid(s_1,v_1)\in A_{l_1},(s_2,v_2)\in M_{l_2}\}\\&\cup\{(s_1\texttt{-}s_2,v_1-v_2)\mid(s_1,v_1)\in A_{l_1},(s_2,v_2)\in M_{l_2}\}\end{aligned}$$

所有合法的等式集合 E 为

$$E=\{s_1\texttt{=}s_2\mid(s_1,v_1)\in A_i,(s_2,v_2)\in A_{8-i},v_1=v_2\}$$

复杂度 $\mathcal{O}(|E|)$。

Problem H. Permutation Counting

考虑数组 b 表示数 j 在排列中的位置，即 $p_{b_j}=j$ ，则 b 也是一个排列。根据题意，若 $b_{j_1}=x_i,\ b_{j_2}=y_i$ ，则有 $j_1<j_2$ ，即在排列 b 中，数 x_i 需要比数 y_i 更早出现。

按照输入所给 m 对点，从 x_i 到 y_i 连有向边，可以发现若存在环则无解，若不存在环则形成由若干棵从根节点往叶子节点连边的树组成的森林。

转换题意，可以发现，只有当前入度为 0 的点才可以作为当前排列的下一个数，即答案所求为整个图的拓扑序计数。

为了简化计算，森林可以看作由 $n+1$ 向所有数的根节点连边的一棵树，做树上 DP 。设 $f[x]$ 表示 x 所在子树的答案，对 $f[x]$ 的求解是一个排列组合问题。伪代码如下：

```
size = x 所在子树大小
f[x] = 1
FOR x 的所有孩子 y
        f[x] = f[x] * f[y] * C(size, y 所在子树大小)
        size = size - y 所在子树大小
```

其中 $C(n,\ m)$ 表示组合数，$C(n,\ m)=\frac{n!}{m!(n-m)!}$ 。

Problem I. Repetition

本题的想法为找一个哈希好做而使用 SA 或者 SAM 不太好写的题。

哈希做法较为显然：先预处理出每个串的哈希，然后使用 KMP 把 T 串出现的位置在 S_i 中打上标记（可以是中止节点），之后二分答案 l，将每个串中长度为 l 并且不含 T 的子串（即 $l\geq|T|$ 且从当前位置开

始不会覆盖到任何一个标记，可以使用前缀和维护标记的区间和）出现次数记下来，若每个串中均有同一个哈希值出现了 k 次或更多即可，使用 `gp_hash_table` 复杂度为 $\mathcal{O}(n \log n)$。

SA 的做法：还是首先使用 KMP 将 T 出现的位置在 S_i 中打上标记（记录结束位置），然后使用后缀数组排序。对于同一个子串，其在 SA 上出现位置必然连续，因而双指针扫一遍，维护在每个串上均出现 k 次的区间，在 height 数组上使用 RMQ 查询区间的 lcp，若此时 lcp 长度满足不会覆盖到区间任意一个 T 标记，则为一个合法答案，复杂度受制于 SA 的排序，为 $\mathcal{O}(n \log n)$。

广义 SAM 做法：首先建立这 n 个串的广义 SAM 并染色，根据 link 树和字符转移出边建立 DAWG，然后把 T 串拉上去跑，那么 T 的终止节点及其能走到的节点均为禁止节点。然后再在 DAWG 去进行启发式合并，合并子节点的次数和颜色信息。

Problem J. Sequence

首先我们进行逐位考虑，经过观察之后不难发现，对于第 i 个数，当且仅当它前面（包括它自己）第 j 位出现了奇数次，在它后面（不包括它自己）第 j 位也出现了奇数次，它覆盖值的第 j 位才为 1。因此如果第 j 位在当前序列中总共出现了奇数次，则任何一个数覆盖值的第 j 位都为 0；如果第 j 位在当前数列中总共出现了偶数次，则任何一个数覆盖值的第 j 位只有在它前面（包括它自己）出现了奇数次才为 1。同样的，若第 i 个数的第 j 位在它之前（包括它自己）出现了偶数次，则无论如何该数的第 j 位都为 0。

因此我们可以记 a_i 表示第 i 个数哪些位上可能为 1。再令 f_i 表示此时是否存在一个位置 k 使得 i 是 a_k 的子集，显然每次在序列最后新增一个数 num_k 时，需要令 $f_i = 1, i \in a_k$ ，查询时则自高而低逐位枚举每一位，若加上这一位后 f 值为 1，则将这一位记入答案。

Problem K. Triangles

观察样例一给出的图，可以发现图中所有的三角形斜边均由红线组成，直角边均由黑线组成，并且每条红线不重复地对应左上和右下两个三角形，因此只需统计有多少条红线即可。一条长度为 l 的线段包含 $l(l+1)/2$ 条小线段，不重叠的线段数量可以直接相加。复杂度 $\mathcal{O}(n)$。

Problem L. WA Sorting

这个算法第一步会将最小值 a_m 和 a_1 交换，这部分的贡献为前缀最小值改变的次数；然后对于 i ，此时前面的序列为 $a_2, ..., a_{m-1}, a_1, a_{m+1}, ..., a_{i-1}, a_m$ ，求出这个序列的递减的单调栈 s ，则 a_i 的贡献为 s 中小于 a_i 的数的个数，这部分可以二分求出。维护这个单调栈，然后二分更新答案即可在 $\mathcal{O}(n \log n)$ 的时间内求出一个序列的答案。

对于序列 $a_1, ..., a_n$ ，最小值 a_m ，在序列末尾插入一个数 $a_{n+1} = x$ 。

- 如果 $x > a_m$ ，则 x 对前面的贡献为 0 ，x 自身的贡献在单调栈中二分即可得出。
- 如果 $x < a_m$ ，则算法第一步的贡献会多 1 ；

  对于 i ，前面的序列变为 $a_2, ..., a_{m-1}, a_m, a_{m+1}, ..., a_{i-1}$ 。

  - 如果 $i < m$ ，贡献不变。
  - 如果 $i = m$ ，贡献为 1 。

- 如果 $i > m$ ，此时单调栈为 $s' = \text{stack}(a_2, ..., a_{m-1}) + \{a_m\}$ （$\text{stack}(A)$ 表示序列 A 生成的单调栈），这部分答案无法快速统计，只能在遇到 a_m 时，把 s' 也保存下来，每次插入时也统计 s' 的答案，最后还要加上 a_1 在 s' 中的贡献。

此时，序列的单调栈变为 $\text{stack}(a_2, ..., a_{m-1}) + \{a_m\} + \{x\}$ ，即 $s' + \{x\}$ ，所以同时维护 s 和 s' 以及它们对应的答案即可。注意当 $m = 1$ 时需要特判一下。

Problem M. XOR Almost Everything

对于所有 $1 \le i \le n$，对除第 i 个之外的数异或 a_i，可以使得所有数均变为 $\oplus_i a_i$（记为 S）。因此当 $S = 0$ 时一定合法。当 $S \neq 0$ 时，分两种情况讨论：

- 当 n 为偶数时，对每个位置进行异或 S 操作，可以使得所有数均为 0，因为每个位置均被异或了 $n - 1$ 次 S；
- 当 n 为奇数时，注意到进行任意操作均不会改变 $\oplus_i a_i$ 的值。由于 $S \neq 0$，一定无法使所有数变为 0。

综上，当且仅当 n 为偶数或 $\oplus_i a_i = 0$ 时输出 `YES`。