Problem A. Xcellent Tree Query Problem

可以将启发式合并的过程倒过来。每次我们将大小较小的一边暴力分离出去。同时用线段树或者平衡树维护每个连通块内部的情况。均摊或者直接维护，复杂度都能做到 $O(n\log^2 n + q\log n)$。

Problem B. Random Nim Game

考虑 $n = 1$ 的情况。容易发现当 $a_1 > 1$ 时先手获胜的概率为 $\frac{1}{2}$，可以归纳证明。

当 $n > 1$ 时，若存在 $a_i > 1$，则类似地，先手操作这一堆时获胜的概率为 $\frac{1}{2}$。于是先手获胜的概率与选择的石子堆无关，任取一堆均为 $\frac{1}{2}$。

即，当且仅当 $a_1 = a_2 = \cdots = a_n = 1$ 时答案为 0 或 1，否则答案为 $\frac{1}{2}$。

时间复杂度 $O(\sum n)$。

Problem C. Rotation

由于 $(1234),(1235)$ 生成 S_5，$(1234),(4567)$ 生成 S_7，$(1234),(3456),(5678)$ 生成 S_8，所以对任意可以旋转的 u,v,w,x，将其用并查集合并。每个连通块大小如果不是 4 或 6，那么就可以达到任意置换。如果是 4，就是 (1234) 生成的群；如果是 6，就是 $(1234),(3456)$ 生成的群。直接判断即可。

时间复杂度 $O(n)$。

Problem D. Medians Strike Back

构造是这样的：

若答案是 B，则构造为 $\{1,3,1,3,\ldots,1,3,2,2,1,3,\ldots\}$，即 B 个 1 3 之后 2 2 构成一循环。

证明：任何一个长度为 $2B+2$ 的区间一定至少有 2 个 2，上述构造同时满足 2 最少。

Problem E. Subsequence Not Substring

解法一

假设出现的最小的字符是 a，次小的是 b，以此类推。

1 字符串里仅有不超过两个极长相等连续段

容易发现此时无解，其余情况均有解。

2 有解

2.1 a 的出现位置不构成一段连续的区间

容易发现此时答案一定是一个全 a 串，具体的长度是 a 出现位置构成的最长连续端长度 $+1$。

2.2 a 的出现位置构成一个区间

2.2.1 这个区间的右端点为 n，即之后不存在其他字符

1. b 的出现位置是一个区间，且之后全为 a：此时找到第 3 小的字符 c，答案为 ca；
2. b 的出现位置不是一个区间，或是一个区间且之后不全为 a：此时答案一定形如 $\texttt{b}^t$ 或 $\texttt{b}^t\texttt{a}$，只需要找到与 a 相邻的一段 b 的长度，令其为 k，若 k 是所有 b 连续段中最长的一个则答案为 $\texttt{b}^{k+1}$，否则答案为 $\texttt{b}^{k+1}\texttt{a}$。

2.2.2 在这个区间之后的字符不全相同

此时我们可以贪心，每次取当前最小的可以选的字符（即后缀最小值），若某一时刻不为子序列则一定合法，或者某一时刻剩余后缀仅有一种字符，此时删去原有子序列的最后一个位置，并添加一个后缀仅有的字符。

2.2.3 在这个区间之后的字符全相同

此时找到在这段 a 之前最小的字符 x。

1. 这段 a 的前一个字符不为 x：直接取 xa 作为答案；
2. a 的个数不为 1 且 x 不为 b：这说明 a 之后所有的字符都是 b，此时取 $\texttt{xa}^{count(\texttt{a})+1}\texttt{b}$ 最优；

3. x 在 a 之前的出现情况构成一个区间，且该区间的右端点与 a 出现区间的左端点相邻：若 x 与 a 之后的字符不同，说明 x 是 b，此时取 a 前所有的 x，以及一个 a 之后的字符；否则说明都是 b，找到两个区间中较长的一个，输出长度 +1 个 b；

4. 其余情况：此时处理方式与 2.2.1.2 一致。

解法二

考虑怎么刻画 S 的一个子串：由 SAM 的定义，SAM 是最小的刻画了 S 所有子串的自动机，所以先建出 SAM。

考虑怎么刻画 S 的一个子序列：每次贪心跳到下一个最近的字符 c，即子序列自动机。

在 SAM 上每个点 u 预处理出 dp_u，表示贪心的起点最多是 dp_u，才存在一个字符串满足从 u 开始走会失配，但是贪心能匹配上。

转移是 $dp_u = \max pre[dp_{trans(u,c)} - 1][c]$，其中 $pre[i][j]$ 表示 i 前面最后一个字符 j 的位置。

有了 dp_u 之后，考虑一个贪心，初始答案字符串为空。

每次尝试加如一个字符 a，贪心会得到一个新的位置，SAM 会得到一个新的节点，如果位置小于等于新节点的 dp 值即可跳过去。如果不存在这个节点，但可以贪心那么就结束循环，否则继续。

Problem F. Product of Sorting Powers

使用不删除莫队（回滚莫队）配合链表解决区间询问，如果对任意 i, j 都能 $O(1)$ 计算 $a_i^{a_j}$，就可以 $O(n\sqrt{q})$ 回答询问。

考虑使用离散对数，问题转化为计算 $a_j \log a_i$。于是我们只需对所有 a_i 预处理出 $\log a_i$ 即可。

考虑使用如下方法计算 x 的离散对数：作带余除法 $p = qx + r$（$q, r \in \mathbb{Z}$，$0 \le r < x$），同时

$$p = (q+1)x + (r - x)$$

于是可以得到：

$$\log x \equiv \log(-r) - \log q \equiv \log(x-r) - \log(q+1) \pmod{\varphi(p)}$$

若我们已经预处理出 $\le \sqrt{p}+1$ 范围内的所有数的离散对数，那么只需考虑 $x > \sqrt{p}+1$ 的情况。此时 $q = \lfloor p/x \rfloor < \sqrt{p}$，因此 $q, q+1$ 的离散对数也是已知的。于是，我们可以根据上面两条式子，将计算 $\log x$ 的问题递归到计算 $\log r$ 或 $\log(x-r)$ 的问题。由于 $\min(r, x-r) \le x/2$，我们可以在一次递归中将问题规模缩小一半，在 $O(\log x)$ 的时间内计算 x 的离散对数。

至于处理出 $\le \sqrt{p}+1$ 的所有数的离散对数，可以使用 BSGS 在 $O(\sqrt{\pi(\sqrt{p})p})$ 的复杂度内解决。于是，预处理后可以在 $O(n \log a_i)$ 的复杂度内计算所有 $\log a_i$，足以通过本题。

Problem G. Sum of Binomial Coefficients

将 $\binom{n}{b_i}$ 看成关于 n 的 b_i 次多项式。同时 $\binom{n}{b_{i+1}}/\binom{n}{b_i}$ 是一个关于 n 的 $b_{i+1}-b_i$ 次多项式。于是，记 $f_i(x)=\binom{x}{b_{i+1}}/\binom{x}{b_i}$，一次询问相当于计算 $\sum_{i=1}^{R}\prod_{j=1}^{i}f_j(n)$，也即计算 $(\sum_{i=1}^{R}\prod_{j=1}^{i}f_j(x)) \bmod (x-n)$。

上述操作相当于，对多项式序列 $f_i(x)$ 先做前缀积，再做前缀和，最后询问序列的某个位置，对 $(x-n)$ 取模的结果。对多项式序列 $f_i(x)$ 建立线段树，并在上面做分治以解决该问题。

下面具体地描述该分治算法。首先将所有询问存入线段树的对应叶子节点，并对每个线段树节点 k，计算出其子树内所有询问的取模多项式乘积 $s_k(x)=\prod_i(x-n_i)$。显然 $s_k(x)=s_{lson}(x)s_{rson}(x)$。

考虑使用分治 `solve`(k,l,r,F) 计算前缀积。分治到节点 k（对应区间为 $[l,r]$）时，`solve` 中应传入 $F=\prod_{i=1}^{l-1}f_i(x)$，然后递归到 `solve`$(lson,l,mid,F)$ 和 `solve`$(rson,mid+1,r,F\times\prod_{i=l}^{mid}f_i(x))$。

注意到我们可以将 F 对 $s_k(x)$ 取模而不影响答案。因为 $(F \bmod s_k(x)) \bmod s_{lson}(x)=F \bmod s_{lson}(x)$，而最后我们只需在叶子节点得到 $F \bmod s_{leaf}(x)$ 的结果，因此事先对 $s_k(x)$ 取模是不影响答案的。

而这样在分治过程中取模，可以将 `solve` 中传入的多项式的次数控制在 $s_k(x)$ 的次数以下。因此，线段树上每一层所涉及的多项式次数之和不超过 $O(q+\max b)$。由于多项式乘法、取模等运算复杂度为 $O(len\log len)$，故这个分治的总复杂度为 $O(q\log^2 q)$（$q,\max b$ 同阶）。

原问题是先前缀积、再前缀和，但是分治的过程是基本一样的，具体细节可以参考 std 或留给读者自行思考。

Problem H. HEX-A-GONE Trails

考虑链 $x \to y$，首先看先手的第一步操作，如果不是沿着链走，那么相当于将除了 x 子树（链外的部分）之外的部分拱手相让给后手，我们可以算出 x 子树内的最大深度以及子树外从 y 出发可以走的最大深度，如果前者更大，那么先手就找到了必胜策略，否则先手这么走一定会输，所以他会沿着链走。

如果先手沿着链走，那么就轮到后手，后手会做个类似的判断，此后的每一步都是这样，即决定是要沿着链走还是走到链外的子树中。

深度的计算可以用各种方法做，例如事先预处理链上第 i 个点往链外延申的最长距离 d_i，然后用个 ST 表维护 $d_i + i$ 和 $d_i - i$ 的区间最大值。

Problem I. Colorings Counting

考虑 Burnside 引理，枚举旋转的次数 i、颜色平移的次数 j 与是否翻转计算不动点个数。

首先计算不翻转的情况：

n 个点被划分为 $\gcd(i,n)$ 个环，第 l 个环为 $(l+ki) \bmod n$。

每个环上的关系为 $a_l \equiv a_{(l+i) \bmod n} - j \equiv \dots \pmod m$。

直接处理上述条件是比较困难的，但注意到实际上只需要考虑 $[1, \gcd(i,n)]$ 的染色方案，这与 1 和 $\gcd(i,n)+1$ 是否同色有关。

注意到 $\gcd(i,n)+1$ 与 1 颜色相同当且仅当每一段颜色都相同，也即 $j = 0$，其余情况颜色均不同。$[1, \gcd(i,n)]$ 的染色方案容易用数学方法或矩阵快速幂优化 DP 快速计算。

注意到对于 j 一维的限制，形如 $aj \equiv 0 \pmod b$，这可以直接化简为 $j \equiv 0 \pmod{\frac{b}{\gcd(a,b)}}$，可以直接计数。

枚举 $\gcd(i,n)$ 可以做到 $O(d(n))$ 或 $O(d(n)\log n)$ 的时间复杂度，其中 $d(n)$ 表示 n 的因子个数。

然后计算翻转的情况：

此时的轮换大小一定不超过 2，且均形如 $\{0,1,2,\dots,n-1\} \to \{i-1, i-2, \dots, 0, n-1, n-2, \dots, i\}$。

注意到两部分中间的位置均为大小为 1 的轮换或大小为 2 且相邻两个数的轮换，为了保证相邻颜色不同与颜色平移后相同的限制，大小为 1 的轮换需要 $j=0$，大小为 2 的轮换需要 $j = \frac{m}{2}$。于是 n 为奇数时一定不存在翻转后的不动点，m 为奇数时不存在 i 为奇数的不动点。

若 n 为偶数，且 i 为偶数，则由 $0, 1, \dots, \frac{i}{2}-1, n-1, n-2, \dots, \frac{n+i}{2}$ 共 $\frac{n}{2}$ 个位置的颜色即可确定其余位置的颜色，不动点个数即为对长度为 $\frac{n}{2}$ 的链 m 染色，满足相邻点不同色的方案数。m 为偶数且 i 为奇数的情况类似。

综上，在对 n 分解素因数后，可以在 $\tilde{O}(d(n))$ 的时间复杂度内完成计算。分解素因数可以使用 Pollard-Rho 算法完成。

Problem J. Widely Known Problem

给所有串赋一个权值 v_t，其中 t 为一个 s 的子串，初始均为 0。我们直接将所有模式串对应的 v_t 加 1，查询的答案即为查询串 $s[l,r]$ 所有本质不同子串中 v 的和。

考虑所有串形成的 trie（这里 trie 字符反着加，为了匹配正串 SAM）。记 f_u 为 u 到根的所有串的 v 之和。答案为 trie 上所有是 $s[l,r]$ 子串的点的 v 之和。显然构成一个 trie 上包含根的连通块，答案即为：

$$\sum_{u\text{是叶子}} f_u - \sum_u (u\text{的儿子个数} - 1) f_u$$

首先考虑后者。按 r 扫描线，考虑 LCT，维护 SAM 上连续上一次出现位置相同的节点。每次 access 时遇到一个连续段时，考虑段尾的点何时儿子个数 $+1$，应为 $l \le \mathrm{last} - \mathrm{len}$ 且 $s[1,r]$ 所在子树第一次出现节点（即上一个连续段中的 last 不会在这里贡献）。所以对 l 是一个区间加。查询即为单点查询。该部分时间复杂度 $O(n\log^2 n + q\log n)$。

再考虑前者。首先叶子一定左端点为 l。其次若 $s[l,t]$ 非叶子，说明存在 $s[l',r'] = s[l,t]$ 且 $t < r' \le r$。那么对所有 $l \le t' < t$，$s[l,t']$ 均非叶子。所以一定存在一个 x，使得 $s[l,t]$ 是叶子当且仅当 $t \ge x$。考虑如何找出该 x。按 l 扫描线，考虑 LCT，维护 SAM 上连续上一次出现位置相同的节点。每次查询一个 $s[l,r]$ 对应的 x 时，找到所有连续段，对于这些连续段，其对应 r 单调，且每个连续段内对应 r 连续。容易离线后 $O(n\log n + q\log n)$ 内解决。

最后考虑如何求出 $\sum_{i=x}^{r} f_{s[l,i]}$。差分后变为若干个 $\sum_{i=l}^{r} f_{s[l,i]}$。这等价于广为人知题的查询，可以用基本子串结构在 $O((n+q)\log n)$ 内解决。

总时间复杂度为 $O(n\log^2 n + q\log n)$。若 n,q 同阶，可优化至 $O\left(\frac{n\log^2 n}{\log\log n}\right)$。

Problem K. Three Operations

可以发现每次贪心地选三个操作中使得 x 最小的，这样选择后两种操作的次数一定不超过 $O(\log x)$ 次，直接暴力枚举足够多轮后，剩下均用 -1 操作解决即可。

Problem L. Landmine

考虑 BFS，每次要找所有未被扫过的点中能够炸到当前点 i 的所有点，相当于找所有未被标记的点 j 使得 $dist(i,j) \leq r_j$。建立点分树，那么就相当于对所有 i 在点分树上的祖先 u，找 u 点分树子树中所有未被标记的点 j，使得 $dist(i,u) \leq r_j - dist(j,u)$。直接对每个点 u 按照 $r_j - dist(j,u)$ 将所有 j 排序即可。每次扫指针找到所有需要求的点。

时间复杂度 $O(n \log^2 n)$。

Problem M. Minimal and Maximal XOR Sum

任选一个长度为 k 的区间，将其翻转，然后可以再利用 $f(k)=\frac{k(k-1)}{2}$ 次交换相邻两个数的操作再将这个区间翻转回去，相当于什么操作都没有做，而却多了一个权值为 k 的操作和 $f(k)$ 个权值为 2 的操作，所以：

- 如果 $k\equiv 0 \pmod 4$ 或 $k\equiv 1 \pmod 4$，则我们可以任何时候将答案（权值异或和）异或 k；
- 如果 $k\equiv 0 \pmod 4$ 或 $k\equiv 1 \pmod 4$，则我们可以任何时候将答案异或 $k\oplus 2$。

设 n 的二进制最高位为 2^k 位（$k>1$），根据上面两条，除了 2^1 这一位之外，我们可以在其他位任意异或任意一个小于 n 的数，所以最小值就是其他位都是 0，最大值就是其他位都是 1。

再考虑 2^1 这一位，可以发现每进行一个区间长度位 $k\equiv 2 \pmod 4$ 或 $k\equiv 3 \pmod 4$ 的区间翻转操作时，排列的奇偶性就会改变，所以 2^1 这一位的值只和排列奇偶性有关，奇排列则为 1 偶排列则为 0。所以只要求一下排列奇偶性就行了。

时间复杂度 $O(n)$。