对于 D 题的原题意，出题人和验题人赛前都没有发现标算存在的问题，导致了许多选手的疑惑和时间的浪费，在此表示真诚的歉意！

预计难度分布：

Easy - DJKL, Medium - ABCEG, Hard - FHI

A. Integers Exhibition

不难发现非 K-magic 数是非常少的，考虑先预处理出来，二分回答询问。

以下我们讨论如何求出非 K-magic 数，为方便描述，我们称一个正整数是良好的当且仅当其是非 K-magic 的。

对于一个质数 p，我们考虑所有仅包含小于 p 的质因子的正整数集 G。不难发现：

- 若 $x \in G$，且在 G 中已经有超过 K 个小于 x 的整数约数个数多于 x，即 x 一定不是良好的，则 xp^c $(c \geq 0)$ 也一定不可能是良好的。

这样我们就可以得到一个初步的想法。开始我们认为仅有 1 是良好的，枚举质因子 p，对于每一个原来认为是良好的数 x，将 xp^c $(c \geq 0)$ 加入候选列表，接着将候选列表排序，除去已经可以确定不是良好的数，进入下一轮迭代。容易证明，在这个算法中，筛去一个不是良好的数 x，是不会在后续过程中令一个原本不是良好的数，变成一个良好的数的，故筛去良好的数的过程是合法的剪枝。

然而枚举的质因子的范围有多大呢？联想 $K = 0$ 这一经典问题，我们知道对于 10^{18} 的范围，考虑前 20 个质因子都绰绰有余了，因为将更大的质因子加入是非常不优的。在 K 更大的时候，我们采用“迭代至稳定”的思想，每一轮迭代后检查答案是否变化，如果在较长一段迭代后答案无任何变化，我们就认为质因子 p 的上界已经达到。经过实践，在 $K = 233$ 时，p 的最大值取到 293 即可。

我们考虑如何在一轮迭代中除去确定不是良好的数。考虑维护前 $K+1$ 大值，从小到大枚举候选列表中的数 x，若 x 小于第 $K+1$ 大值，我们就把这个数除去。否则更新前 $K+1$ 大值。根据上述描述可以大致估算复杂度。设 $K = 233$ 时，10^{18} 内良好的数的数量为 N，经过实践，可以知道 N 约为 50000。每次扩展最多把一个数扩展成 $\log M$ 个数，在剪枝完毕后，列表大小又回归到 N 以下，故时间复杂度可以估算为 $O(NKMax(p)\log M)$，常数较小。

B. Harvest of Apples

定义 $S(n,m) = \sum_{i=0}^{m}\binom{n}{i}$，不难发现 $S(n,m) = S(n,m-1) + \binom{n}{m}, S(n,m) = 2S(n-1,m) - \binom{n-1}{m}$。也就是说，如果我们知道 $S(n,m)$，就能以 $O(1)$ 的代价计算出 $S(n-1,m), S(n,m-1), S(n+1,m), S(n,m+1)$，可以采用莫队算法。

时间复杂度 $O(T\sqrt{MAX})$。

C. Problems on a Tree

用并查集维护两种连通块 —— Easy + Medium 题的连通块，维护大小；Easy 题的连通块，维护大小以及与此连通块只隔一个 Hard 题的 Easy + Medium 连通块大小之和即可。

D. Nothing is Impossible

如果仅有 1 道题，至少有一个人做对这题需要有 **错误答案个数 + 1** 个人。

那么容易发现在每道题正确答案只有一个的情况下，如果 n 道题中存在 s 道题，使得学生人数 m 不少于每道题 **错误答案个数 + 1** 相乘的结果，那么一定有人能够得到 s 分。故我们将题目按**错误答案个数**从小到大排序，找到最大的 p 满足 $\prod_{i\leq p}(b_i+1) \leq m$ 就是答案。

E. Matrix from Arrays

简单推导可得

$$M[i][j] = A[(\frac{(i+j)(i+j+1)}{2} + i) \mod L] = A[(\frac{3i}{2} + \frac{j}{2} + \frac{i^2}{2} + \frac{j^2}{2} + ij) \mod L] = M[i+2L][j] = M[i][j+2L]$$

预处理左上角 $2L \times 2L$ 的矩阵的二维前缀和，$O(1)$ 回答询问。时间复杂度 $O(L^2+Q)$。

F. Travel Through Time

由于可持久化的存在，直接维护哪些位置有棋子十分困难。考虑维护一些全是棋子的线段，这些线段可以有重叠部分，但是需要保证这些线段覆盖了所有的棋子。

注意到如果我们只维护了线段的左右边界，甚至不用知道某个左边界具体对应的是哪个右边界，就可以知道某个位置上有没有棋子。因此只维护左右边界，把左边界看成左括号，右边界看成右括号，那么这就是一个括号序列。

比如说对于 0111011110 这样一串格子（`1`表示有棋子，`0`表示没有），我们可以用这样的括号序列来维护：0(111)0(1111)0。由于一个局面并不对应了唯一的括号序列，因此这些括号序列也是可以的：0(111)0(11)(11)0，0(1(1(1)))0(((11(11))))0。

对于每一个操作，都可以用括号序列维护：

- 操作一：在 x 前加入一对括号。
- 操作二：将所有左括号向左移动 x，将所有右括号向右移动 x。
- 操作三：在 l 的前面与 r 的后面加入形如 `... )))((( ...` 的括号使得没有线段穿过 l 和 r。然后将 l,r之间的括号直接翻转并反转。比如说对于 0(111)0(11(1)1)0，如果要翻转 $[3,8]$，首先补充括号，变成：0(1] [11)0(11(1]][[1)1)0（为了区分,`[`与`]`是新加入的括号），然后翻转，得到：0(1] [[1)11)0(11] [[)1)0。

对于左括号与右括号，分别开一棵可持久化 Treap 维护即可。时间复杂度 $O(n\log n)$。

G. Depth-First Search

由于题目和字典序有关，不妨运用逐位确定的思想。

首先，我们要求第一位小于 B_1 的序列总数，即我们需要快速求出以每个点为根时的DFS序列总数。对于有根树，设 $f(i)$ 为以 i 为根的子树的DFS序列总数，有

$f(u)=|son(u)|!\prod_{v\in son(u)} f(v)$

我们可以先任选一个根 DFS 求出 f，在第二遍 DFS 考虑祖先的贡献即可将所有点的答案求出。

接着我们以 B_1 为根，逐位确定求出所有的答案。和上述方法类似，即如果当前在 B_i 点，要走到 B_{i+1} 点，需要求出所有第 $i+1$ 位小于 B_{i+1} 的方案数，简单计算即可。

需要注意的是，由于我们可能需要快速计算某个点下某个子树的名次，所以需要用树状数组或线段树来优化这个过程。

时间复杂度 $O(n\log n)$。

H. Eat Cards, Have Fun

考虑如何计算某个特定排列 A 的 Value.

$Value(A)=\sum_{i=1}^n (n-i)!\sum_{j>i}[A_j<A_i]$

这启发我们对于每个 i 分别计算贡献。考虑当第 i 张卡片被吃掉的时候，我们需要知道这张卡片**左边、右边**分别已有多少卡片被吃掉（记为 l,r），才能确定第 i 张卡片在 A 中的位置；我们还需要知道这张卡片**左边、右边**分别已有多少**卡面数字小于 a_i 的卡片**被吃掉（记为 $\hat{l},\hat{r}$），才能确定第 i 张卡片对答案的贡献，即 $\sum_{j>i}[A_j<A_i]$。如果知道了 $l,r,\hat{l},\hat{r}$，那么答案就是

$Ans=\sum_{i=1}^n\sum_{l=0}^{i-1}\sum_{r=0}^{n-i}\sum_{\hat{l}=0}^{l}\sum_{\hat{r}=0}^{r}(n-l-r-1)!(a_i-1-\hat{l}-\hat{r})P(i,l,r,\hat{l},\hat{r})$

其中 $P(i,l,r,\hat{l},\hat{r})$ 是达到对应情况的概率。我们可以枚举第 i 张卡片是在第 k 轮 $(k>0)$ 被吃掉的来计算概率：

$P(i,l,r,\hat{l},\hat{r})=\binom{b_i}{\hat{l}}\binom{i-b_i-1}{l-\hat{l}}\binom{a_i-b_i-1}{\hat{r}}\binom{n-i-a_i+b_i+1}{r-\hat{r}}\sum_{k=1}^{\infty}(1-(1-p)^k)^l((1-p)^k)^{i-1-l}p(1-p)^{k-1}(1-(1-p)^{k-1})^r((1-p)^{k-1})^{n-i-r}$

其中 $b_i=\sum_{j<i}[a_j<a_i]$.

观察式子 $(1-(1-p)^x)^y$，可以用二项式定理展开：

$(1-(1-p)^x)^y=\sum_{i=0}^y\binom{y}{i}(-1)^i(1-p)^{xi}$

利用上述结论，进一步化简：

$$
\begin{aligned}
&\sum_{k=1}^{\infty}(1-(1-p)^k)^l((1-p)^k)^{i-1-l}p(1-p)^{k-1}(1-(1-p)^{k-1})^r((1-p)^{k-1})^{n-i-r}\\
=&p\sum_{k=1}^{\infty}(1-p)^{k(i-1-l)+k-1+(k-1)(n-i-r)}\sum_{x=0}^{l}\binom{l}{x}(-1)^x(1-p)^{xk}\sum_{y=0}^{r}\binom{r}{y}(-1)^y(1-p)^{y(k-1)}\\
=&p\sum_{x=0}^{l}\sum_{y=0}^{r}(-1)^{x+y}\binom{l}{x}\binom{r}{y}\sum_{k=1}^{\infty}(1-p)^{k(i-1-l)+k-1+(k-1)(n-i-r)+xk+y(k-1)}\\
=&p\sum_{x=0}^{l}\sum_{y=0}^{r}(-1)^{x+y}\binom{l}{x}\binom{r}{y}\sum_{k=0}^{\infty}(1-p)^{k(n-l-r+x+y)+(i-1-l+x)}\\
=&p\sum_{x=0}^{l}\sum_{y=0}^{r}(-1)^{x+y}\binom{l}{x}\binom{r}{y}\frac{(1-p)^{i-1-l+x}}{1-(1-p)^{n-l-r+x+y}}
\end{aligned}
$$

至此，我们获得了一个时间复杂度为 $O(n^5)$ 的算法。上述公式显然有不少冗余，可以进一步优化。

回顾原式：

$Ans = p\sum_{i=1}^{n}\sum_{l=0}^{i-1}\sum_{r=0}^{n-i}(n-l-r-1)!(\sum_{x=0}^{l}\sum_{y=0}^{r}(-1)^{x+y}\binom{l}{x}\binom{r}{y}\frac{(1-p)^{i-1-l+x}}{1-(1-p)^{n-l-r+x+y}})(\sum_{\hat{l}=0}^{l}\sum_{\hat{r}=0}^{r}(a_i-1-\hat{l}-\hat{r})\binom{b_i}{\hat{l}}\binom{i-b_i-1}{l-\hat{l}}\binom{a_i-b_i-1}{\hat{r}}\binom{n-i-a_i+b_i}{r-\hat{r}}$

以下将定义若干辅助函数加速计算答案。

定义 F： $F(i,l,r)=\sum_{x=0}^{l}\sum_{y=0}^{r}(-1)^{x+y}\binom{l}{x}\binom{r}{y}\frac{(1-p)^{i-1-l+x}}{1-(1-p)^{n-l-r+x+y}}$

考虑如何快速计算 F。不妨定义 F_n：

$F_n(l,r)=\sum_{x=0}^{l}\sum_{y=0}^{r}(-1)^{x+y}\binom{l}{x}\binom{r}{y}\frac{(1-p)^{n-1-l+x}}{1-(1-p)^{n-l-r+x+y}}$

显然 $F(i,l,r)=F_n(l,r)(1-p)^{i-n}$。

$F_n(l,r)=l!r!\sum_{x=0}^{l}\sum_{y=0}^{r}\frac{(-1)^x}{x!}\frac{(-1)^y}{y!}\frac{(1-p)^{n-1-(l-x)}}{(l-x)!(r-y)!(1-(1-p)^{n-(l-x)-(r-y)})}$

令 $G(x,y)=\frac{(1-p)^{n-1-x}}{x!y!(1-(1-p)^{n-x-y})}$：

$F_n(l,r)=l!r!\sum_{x=0}^{l}\frac{(-1)^x}{x!}\sum_{y=0}^{r}\frac{(-1)^y}{y!}G(l-x,r-y)$

可以在 $O(n^3)$ 的时间计算出 F_n。

定义 L,R,L_+,R_+,H：

$L(i,l)=\sum_{\hat{l}=0}^{l}\binom{b_i}{\hat{l}}\binom{i-b_i-1}{l-\hat{l}}, R(i,r)=\sum_{\hat{r}=0}^{r}\binom{a_i-b_i-1}{\hat{r}}\binom{n-i-a_i+b_i+1}{r-\hat{r}}$

$L_+(i,l)=\sum_{\hat{l}=0}^{l}\hat{l}\binom{b_i}{\hat{l}}\binom{i-b_i-1}{l-\hat{l}}, R_+(i,r)=\sum_{\hat{r}=0}^{r}\hat{r}\binom{a_i-b_i-1}{\hat{r}}\binom{n-i-a_i+b_i+1}{r-\hat{r}}$

$H(i,l,r)=\sum_{\hat{l}=0}^{l}\sum_{\hat{r}=0}^{r}(a_i-1-\hat{l}-\hat{r})\binom{b_i}{\hat{l}}\binom{i-b_i-1}{l-\hat{l}}\binom{a_i-b_i-1}{\hat{r}}\binom{n-i-a_i+b_i+1}{r-\hat{r}}=(a_i-1)L(i,l)R(i,r)-L_+(i,l)R(i,r)-L(i,l)R_+(i,r)$

以 $O(n^3)$ 的代价预处理 L,R,L_+,R_+, 可以在 $O(n^3)$ 的时间计算出 H。

现在 Ans 就可以在 $O(n^3)$ 的时间计算出来啦。

$Ans=p\sum_{i=1}^{n}\sum_{l=0}^{i-1}\sum_{r=0}^{n-i}(n-l-r-1)!F_n(l,r)(1-p)^{i-n}H(i,l,r)$

I. Delighful Formulas

根据题意列出式子：

$Ans=\sum_{i=1}^{N}[\gcd(i,N)=1]\sum_{j=1}^{i}j^K$

莫比乌斯反演：

$$
\begin{aligned}
Ans &= \sum_{d|N}\mu(d)\sum_{i=1}^{N}[d \mid i]\sum_{j=1}^{i} j^K \\
&= \sum_{d|N}\mu(d)\sum_{i=1}^{\frac{N}{d}}\sum_{j=1}^{id} j^K
\end{aligned}
$$

定义 F:

$F_p(N) = \sum_{i=1}^{N} i^p$

显然 F_p 是 $p+1$ 阶多项式:

$F_p(N) = \sum_{i=0}^{p+1} a_{p,i}N^i$

利用 F 化简原式:

$$
\begin{aligned}
Ans &= \sum_{d|N}\mu(d)\sum_{i=1}^{\frac{N}{d}} F_K(id) \\
&= \sum_{d|N}\mu(d)\sum_{i=1}^{\frac{N}{d}}\sum_{j=0}^{K+1} a_{K,j}(id)^j \\
&= \sum_{d|N}\mu(d)\sum_{j=0}^{K+1} a_{K,j}d^j\sum_{i=1}^{\frac{N}{d}} i^j \\
&= \sum_{d|N}\mu(d)\sum_{j=0}^{K+1} a_{K,j}d^j F_j(\frac{N}{d}) \\
&= \sum_{d|N}\mu(d)\sum_{j=0}^{K+1} a_{K,j}d^j \sum_{k=0}^{j+1} a_{j,k}(\frac{N}{d})^k \\
&= \sum_{d|N}\mu(d)\sum_{i=-1}^{K+1} d^i \sum_{j=0}^{K+1}\sum_{k=0}^{j+1}[j-k=i]a_{K,j}a_{j,k}N^k
\end{aligned}
$$

定义 G:

$G_i = \sum_{j=0}^{K+1}\sum_{k=0}^{j+1}[j-k=i]a_{K,j}a_{j,k}N^k$

利用 G 化简原式:

$$
\begin{aligned}
Ans &= \sum_{d|N}\mu(d)\sum_{i=-1}^{K+1} d^i G_i \\
&= \sum_{i=-1}^{K+1} G_i \sum_{d|N}\mu(d)d^i \\
&= \sum_{i=-1}^{K+1} G_i \prod_{p|N}(1-p^i)
\end{aligned}
$$

如果我们能快速计算出 G，就可以在 $O(MK)$ 的时间计算答案，其中 M 为质因子个数。

将 G 用伯努利数展开，可以发现是卷积的形式，直接 NTT，时间复杂度 $O(K\log K)$。

J. Let Sudoku Rotate

搜索加可行性剪枝即可通过。由于数独限制较强，剪枝效果良好。

K. Expression in Memories

注意在类似 `+0?` 的情况下，`?` 须被替换为 `+` 或 `*`，其余情况直接将 `?` 替换为非零数字就好。替换完成后判断一下是否合法。

L. Graph Theory Homework

容易证明 $\lfloor\sqrt{a}\rfloor + \lfloor\sqrt{b}\rfloor \ge \lfloor\sqrt{a+b}\rfloor$，进而可以证明边权满足三角不等式，故直接从 1 走到 n 就是最优的。