本次题目的难度分布如下：

- Iron：A, F, L
- Bronze：E, I, J
- Silver：B, G, H
- Gold：C, D, K

出题人名单（洛谷用户名）：

- C, H 是 _jimmywang_ 出的。
- E 是 djwj223 出的。
- J 是 Gorenstein 出的。当然，在 ICPC 区域赛中，这种长度的题目描述并不罕见，甚至还有可能是英文。
- 其他题都是 yummy 出的。

致歉：由于出题团队的同学这个暑假要参与军训（以及西湖大学特有的假期科研实训），所以比赛准备得较为仓促。然后，我们备赛的重心又放在了各种题目的数据强度上（虽然单个测试点的形式又让我们的数据强度仍然难以保证），导致题面中出现了比较多的 typo，我在此（yummy）代表出题组对各位选手道歉。

关于卡常：

- 在 ICPC 中，使用快读是常规操作。离散化不要用 `map` 也是常规操作。
- 1012 的时间限制是**没有快速读入**的 std 用时的两倍。
- 1007 的 std 严格 $O(n)$，因为我没有刻意卡带 log，所以数据范围 5×10^5。如果你的 log 比较大，那么可能过不去。

Iron

这一部分题主要考察大家的基本功，难度相当于区域赛送分题。

A. 数字卡片

注意到 n 是 4 的倍数的条件为下面两条之一：

- n 的个位是 $0, 4, 8$，且十位不存在或为偶数。
- n 的个位是 $2, 6$，且十位是奇数。

为了尽量节省卡片，$0, 4, 8$ 要单独当成一个数，剩下的数尽量配成“奇数 - $2, 6$” 的对子。

分别计算奇数的个数 c_{1357} 和 2 的个数 c_2。如果 $6,9$ 的个数 c_{69} 满足 $c_{69} \geq |c_{1357} - c_2|$，那么能凑成 $\lfloor \frac{c_{69} + c_{1357} + c_2}{2} \rfloor$ 个对子，否则只能凑出 $\min(c_{1357}, c_2) + c_{69}$ 个。

F. 括号匹配

注意到"存在一个子串是括号串"当且仅当存在一个子串是 `()`。

分类讨论奇偶回文串。以奇回文串为例：

枚举每个对称中心 c，尝试从半径 $r = 0$ 开始向外扩展，并同时维护"是否包含 `()`"以及"当前是否回文"两个 tag，对于所有合法的 r，让答案增加 1 即可。

L. 测试若歪

按题意模拟即可。维护一个 cnt 数组表示每道题的 AC 数以及 fir 数组表示每道题的首 A。

注意到没保证 $\sum n$，所以我们只能枚举所有**有 AC** 的题目计算答案，以及多测清空时也只清空**有 AC** 的题目。

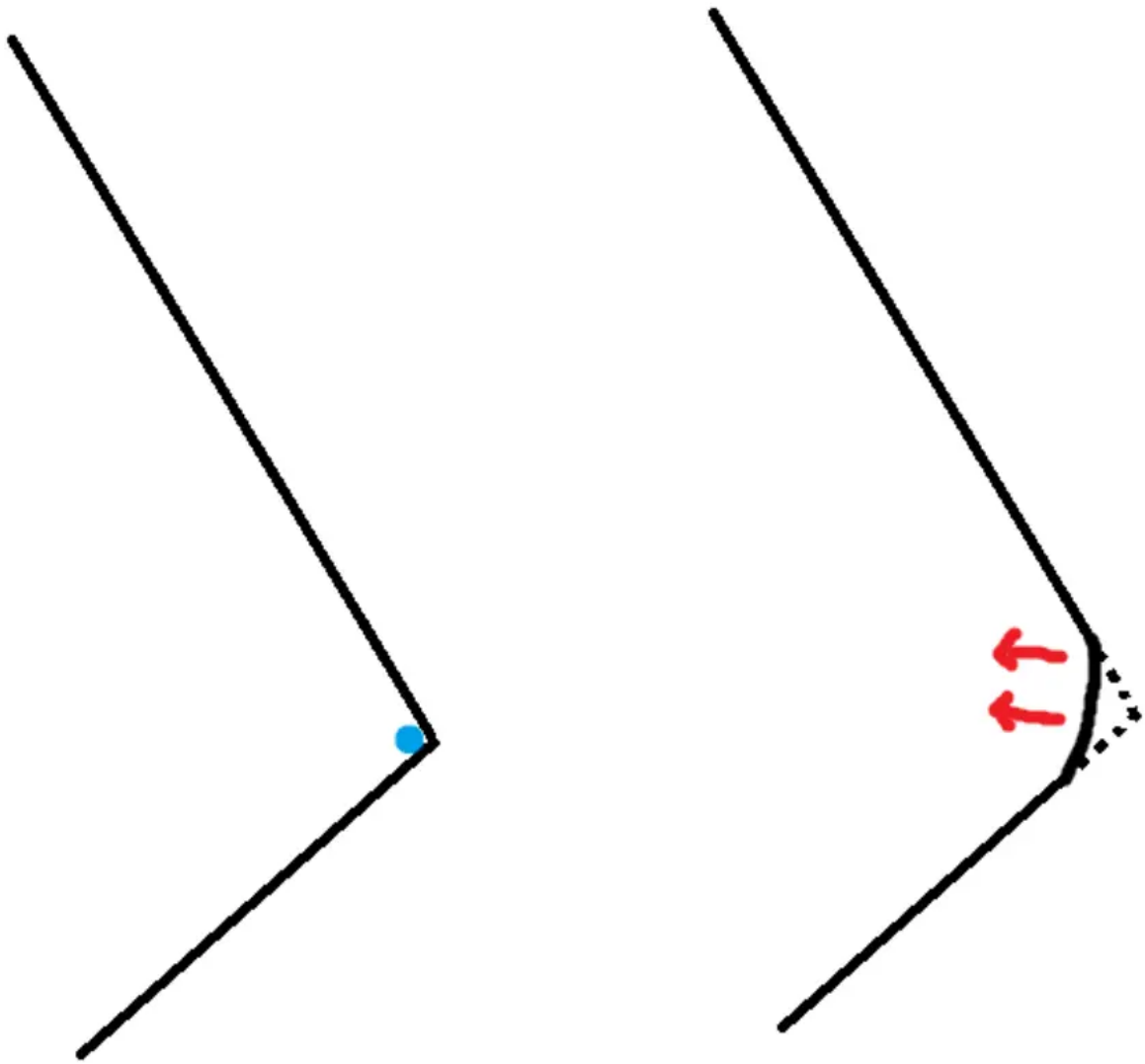
实现精细的话，时间复杂度可做到 $O(n + \sum m)$（虽然也未必更快），当然 $O(n + \sum m \log n)$ 也放过去了。

Bronze

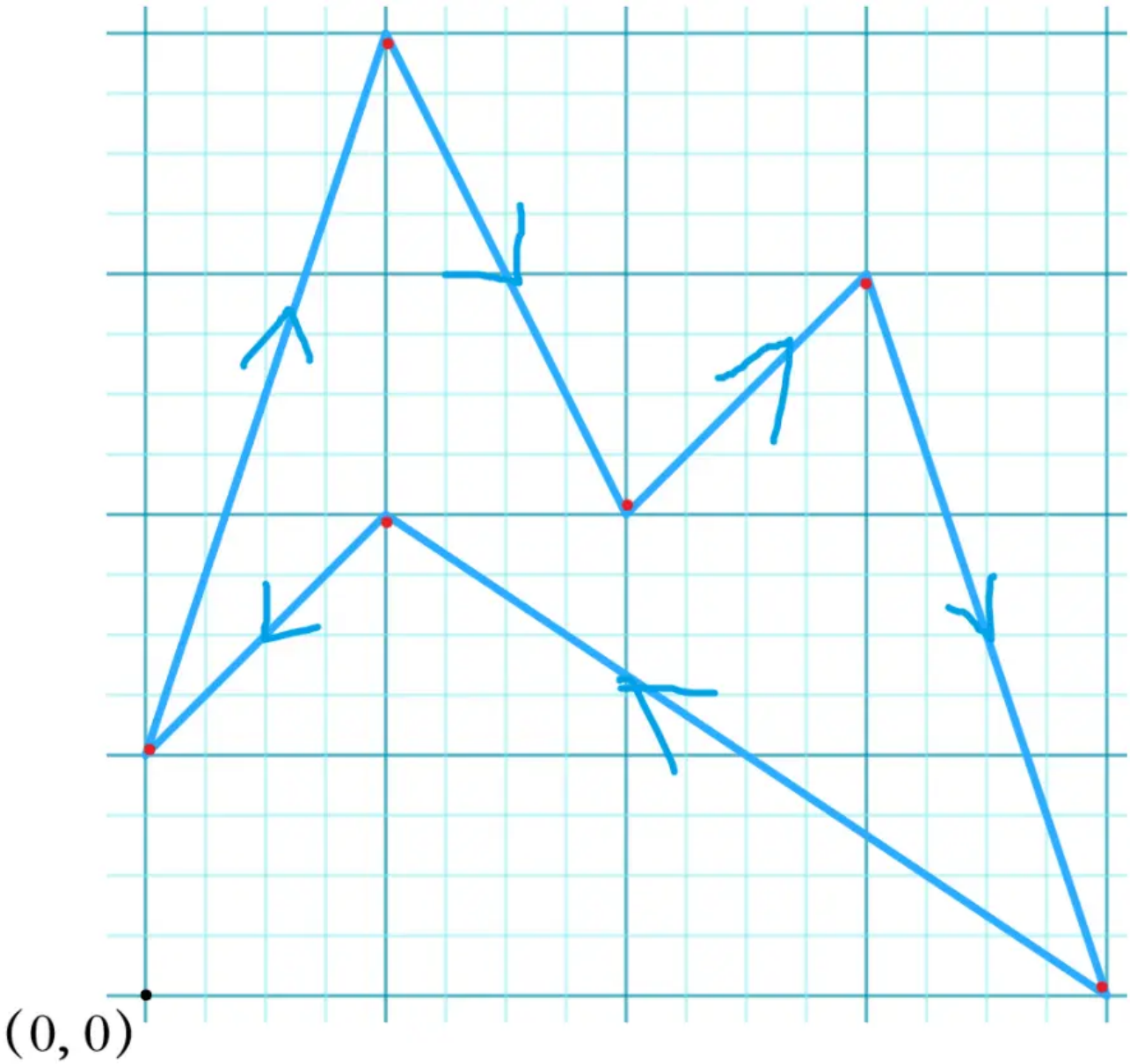
这一部分题在代码或思维方面比较简单，难度相当于区域赛铜牌题。

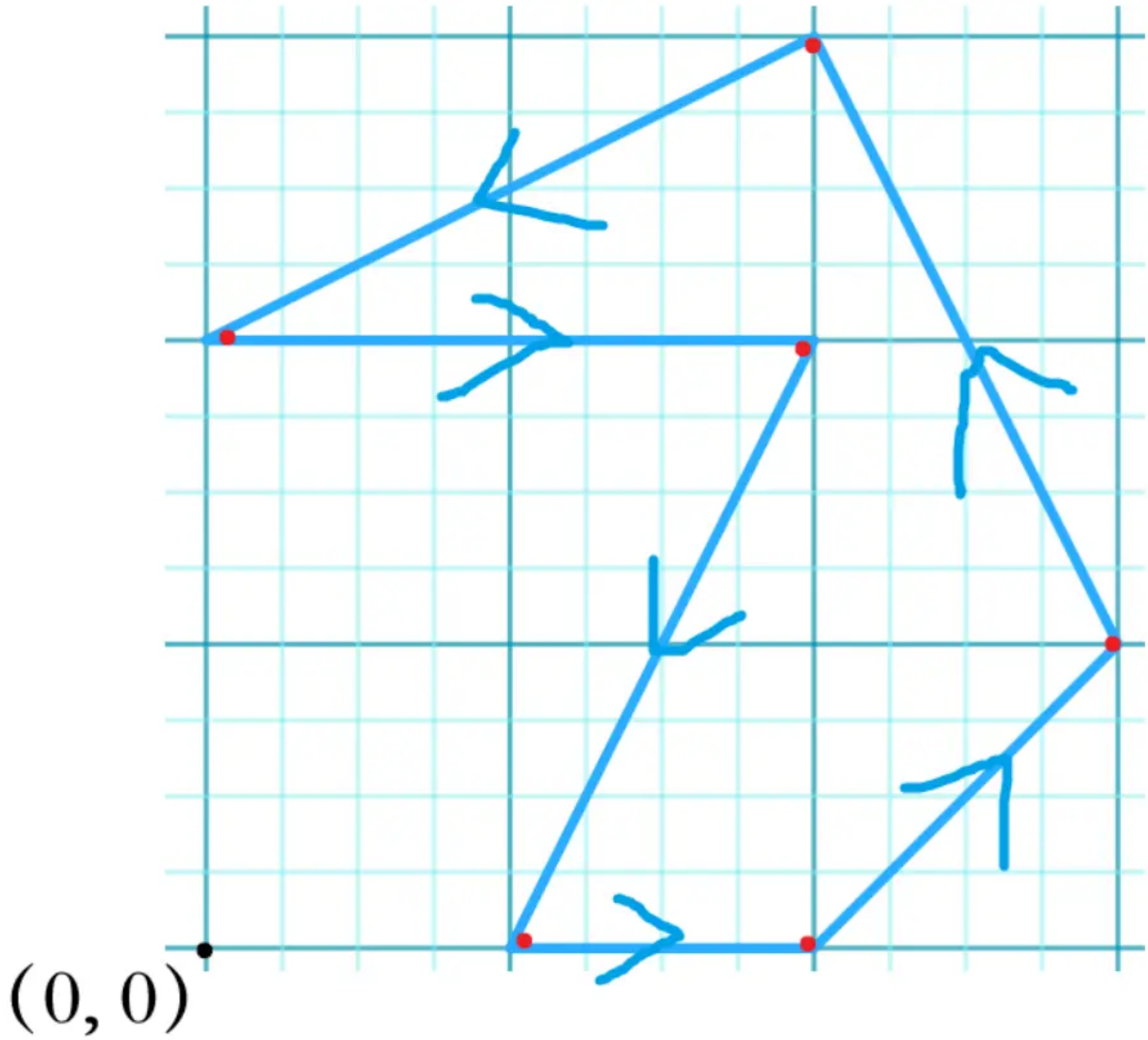
E. 计算几何

大家首先需要意识到，在一条折线的每个拐点处，必须在突出侧内部放置一个钉子，不然的话弹性绳就会收缩使得最终形状不符合预期。



但是由于题目问的是每个点在折线包成的多边形内部还是外部，所以我们还需要判断这些线段的左边是多边形内部还是右边是多边形内部。判断方法可以是去判断折线从开始到最后是顺时针转了一圈（每条线段右边是多边形内部）还是逆时针转了一圈（每条线段左边是多边形内部）


(0, 0)



我们可以看到，样例一折线是顺时针转了一圈，每条线段右边是多边形内部；样例二折线是逆时针转了一圈，每条线段左边是多边形内部。

到这里，题目就做完了。

I. 乘法逆元

当 p 足够大（$k \geq 23$）时，$2^k > (119k)^2 > p^2$，我们有下列做法：

$LHS = \bigoplus_{i=1}^{p-1} 8^k + \text{inv}(i) \cdot 4^k + i \cdot 2^k + i \cdot \text{inv}(i)$ 中的四项互不干扰。

- 8^k 出现了偶数次，异或和为 0。
- 因为 inv 是个排列，有 $\bigoplus_{i=1}^{p-1} \text{inv}(i) = \bigoplus_{i=1}^{p-1} i$。

- 因为 $\mathrm{inv}(\mathrm{inv}(i))=i$，所以除了 $1,p-1$ 以外其他的 $i\cdot\mathrm{inv}(i)$ 都两两抵消了，$\bigoplus_{i=1}^{p-1} i\cdot\mathrm{inv}(i)=1^2\oplus(p-1)^2$。

所以接下来问题就是 $\bigoplus_{i=1}^{p-1} i$ 怎么求。注意到 $(4k)\oplus(4k+1)\oplus(4k+2)\oplus(4k+3)=0$，我们有 $\bigoplus_{i=1}^{p-1} i=\bigoplus_{i=(p-1)-(p-1)\bmod 4}^{p-1} i$，直接 $O(1)$ 求出来就行了。

不难验证 $k\le 21$ 时，直接计算并不会爆 `long long`，而当 $k=22$ 时，也仅有 8^k 这一项爆 `long long`，而这一项本来就会抵消。

因此直接暴力算然后 `unsigned long long` 自然溢出啥事没有。

J. 阿斯蒂芬

将每个魔法水晶视作一个点，则诸个 A_j 按照 $j\to k\in A_j$ 诱导出一张有向图 G。设 $G=(V,E)$。下文称包含至少两个点的 SCC 为 **真 SCC**。

对于 G 中的任意一个真 SCC $S\subset V$，一旦任一 $x\in S$ 在某时刻 $a_x>0$，则整个 SCC 的过程不能停止，并且还会为其出边不断贡献能量。我们称这种真 SCC 为**可激活的**。于是问题转化为：找所有这样的点 x，使得对所有可激活的真 SCC S，不存在 S 到 x 的路径。我们记这样的 x 构成集合 $U\subset V$；所有 $V\setminus U$ 中的点皆要被设置为“沉寂”。

容易发现，一个真 SCC S 是可激活的，如果它本身包含 $a_x>0$ 的点 x，或者有任何一个点 y 满足 $a_y>0$ 且存在 y 到 S 的路径。对 G 缩点得到 G' 后，对每个点 $x\in G'$，按照拓扑排序的顺序维护是否有在前的点有能量能流到 x；若能，且 x 在原图是一个真 SCC，则它就是可激活的，所有以它出发能到达的点皆要被设置为“沉寂”。

Silver

这一部分题需要稍高级的知识点或较充分的思考，难度约等于区域赛银牌题。

B. 预处理器

笑点解析：本题 idea 来源没了。

这题的 idea 来源是 yummy 一个经典之作：

```
#define S(x) x*x
cout<<S(3<<1)<<endl;
```

但是后来他发现，我们完全没有办法维护模意义下的十进制拼接，于是只好在最外层加一个括号。

先考虑只有加法和乘法怎么做。设 $P = 10^9 + 7$。

每个算式的运算结果一定只有 a 或 $a + x + c$ 两种形式，其中 a, c 是纯乘法算式，x 可以是加乘混合算式。下文中，后者被记为 (a, x, c)。

我们不难写出上述算式的合并：

- $a_1 \times a_2 = a_1 a_2$，$a_1 + a_2 = (a_1, 0, a_2)$。
- $a_1 \times (a_2, x_2, c_2) = (a_1 a_2, x_2, c_2)$，$a_1 + (a_2, x_2, c_2) = (a_1, a_2 + x_2, c_2)$。
- $(a_1, x_1, c_1) + (a_2, x_2, c_2) = (a_1, x_1 + c_1 + a_2 + x_2, c_2)$。
- $(a_1, x_1, c_1) \times (a_2, x_2, c_2) = (a_1, x_1 + c_1 a_2 + x_2, c_2)$。

我们把这个“a 或 (a, x, c)”的结构定义为**加乘结果**结构体。

接下来考虑左移。

请读者注意 `a<<b<<c` 等于 `a<<b+c`，以及 `a<<b` 等于 `a<<b%(P-1)`（费马小定理）。

因此每个算式的运算结果一定只有 u 或者 `u<<v` 两种形式，其中 u, v 都是加乘结果。

注意到 `u<<v` 中的 u 可能会放在第一个 `<<` 左边，也可能不在，因此我们要同时维护 $u \bmod P$ 和 $u \bmod (P - 1)$，分别记作 l, r，则可以把含左移的运算结果写作 (l, r, v)；单独的 u 同理写成 (l, r)。

然后同样地根据语义归并即可。读者可自行完成或参考 std。

G. 三角剖分

先不考虑删边。

考虑把三角形作为结点，相邻的三角形之间连边，以某个贴边三角形为根，你就得到了一棵二叉树。

规定用一个三角形和父结点三角形的公共边 (u, v) 来表示三角形，且 $u < v$。

设 $f(u, v)$ 表示 (u, v) 为根的子树中 u, v 已经连通的方案数，$g(u, v)$ 表示 u, v 不连通的方案数。

设 (u, w) 的两个孩子是 (u, v) 和 (v, w)，那么：

- $f(u, w) = f(u, v)g(v, w) + g(u, v)f(v, w) + f(u, v)f(v, w)$

- $g(u,w) = f(u,v)g(u,w) + g(u,v)f(u,w)$

接下来考虑删边，定义 $f(v,u), g(v,u)$ 表示 (u,v) **子树外**的三角形生成树数量。

- $f(v,u) = f(v,w)g(w,u) + g(v,w)f(w,u) + f(v,w)f(w,u)$
- $g(v,u) = f(v,w)g(w,u) + g(v,w)f(w,u)$。

（Hint：只需要旋转草稿纸就可以从刚才的转移方程得到现在的。）

如果令 $d(u,v) = f(u,v) - g(u,v)$ 表示"不通过 (u,v) 就连通的方案数"，那么对于每个对角线 (u,v)，只要计算

$d(u,v)g(v,u) + d(v,u)g(u,v)$。

若实现精细，时间复杂度可以 $O(n)$，粗糙的话则是 $O(n \log n)$。

稍微解释一下 std：首先用类似计数排序的方法给所有**单向**边排序。

然后对转移方程进行了改动：$dp(u,j)$ 表示 u 为起点的第 j 条边的 dp 值。

如果令 y 为 u 起点的第 $j-1$ 条边终点，那么 u 为起点的第 j 条边终点，一定也是 y 为起点的最后一条边（否则边会相交）。因此我们不需要 `map` 或哈希，就可以找到三角形正确的三边。

H. 老猫下山

> Fun fact：这题由 ZJCPC 2025 热身赛 C 题改编，它是本题在 $k=2, a_{i,j}=0$ 下的构造方案版本。

$n=1$ 或 $m=1$ 的情况十分平凡。以下我们仅讨论 $n,m \geq 2$ 的情况。

我们先不管这个成本，先来看有解性问题。假设我们最终想要所有数都等于 x，由此我们可以直接求出每个位置（在模 k 意义下）需要被经过的次数 $t_{i,j} = x - s_{i,j}$。我们接下来证明一个引理。

引理 1：在所有 $x \in [0, k-1]$ 中，最多只有一个 x 能够导出一个解。

证明：我们考虑 $t_{1,1}, t_{1,2}, t_{2,1}$ 这三个数。

显然，格子 $(1,2)$ 和 $(2,1)$ 被经过的唯一来源是从 $(1,1)$ 走向它们。因此经过 $(1,1)$ 的次数一定等于 经过 $(1,2)$ 和 $(2,1)$ 的次数只和。换言之，$t_{1,1} \equiv t_{1,2} + t_{2,1} \pmod{k}$。由于我们刚刚是在模 k 意义下定义的 t，我们最好还是用同余表示这个关系。

也就是说，有解的一个必要条件是 $t_{1,1} \equiv t_{1,2} + t_{2,1} \pmod k$，也即 $x - s_{1,1} \equiv x - s_{1,2} + x - s_{2,1} \pmod k$。由此可以解出一个唯一的 x。

此时我们获得了模意义下每个格子应该被经过多少次，现在考虑如何判断这个 x 是否合法。把网格图转化为一张 DAG，其中边只能向下或向右走。我们现在开始关注一条往右或往下的**边**被经过的次数。以下的计算均在模 k 意义下进行。

首先观察第一行。每个格子如果被经过，一定只能从其左边一个格子过来。因此第一行所有向右的边需要被经过的次数是能够直接确定的。又由于第一行的所有格子除了向右走（次数已经确定）只能向下走了，因此向下走的次数也已经能够确定。

向下走的次数又间接确定了第二行中所有向右的边的次数，因为这些格子被经过的来源要么是从上面来（次数已知），要么从左边来。重复这个过程我们就可以地推确定所有向右走和向下走的边所需要经过的次数。

递推的过程中检查是否合法即可。

现在我们已经能够确定每条边（在模意义下）会被经过多少次，接下来求解答案。走一条路径这种操作很像带费用的流量行进，我们建立费用流模型，尝试使用最小费用流解决。

整理一下我们现在有的限制：

- 每条边被经过的次数是 $pk + q$，其中 q 已经由递推确定。
- 除了 $(1, 1), (n, m)$，其余点需满足流量平衡。

如何处理呢？对于一个点，假设它的入边（从左边和上面过来的边）的次数分别是 $ak + p, bk + q$，出边（向右和向下走的边）的次数是 $ck + r, dk + s$，那么此时流量平衡就能被写成：

$(a + b)k + p + q = (c + d)k + r + s$ 其中 p, q, r, s 应该满足 $p + q \equiv r + s \pmod k$，这是由递推决定的。将上面的等式右侧的 $r + s$ 移到左边并除整体以 k，我们能够重写流量平衡为 $a + b + \delta = c + d$，其中 $\delta = (p + q - r - s)/k \in \{-1, 0, 1\}$。

建立源汇点 S, T，S 连向 $(1, 1)$，流量无限，费用为 $a_{1,1}$；(n, m) 连向 T，流量无限，费用为 0。网格图中的所有边流量无限，费用为这条边指向的格子的成本。这张图足以处理所有 $\delta = 0$ 的情况。如果某个点 $\delta > 0$，说明这个点会凭空吃掉 1 的流量，在这个点和 T 之间连边并强制流量必须为 δ，费用为 0；如果某个点 $\delta < 0$，说明这个点会凭空产生 1 的流量，在 S 和这个点之间连边并强制流量必须为 $-\delta$，费用为 0。这样就能描述每个点的流量平衡了。

“强制”这一条件可以看作上下界。跑上下界有源汇最小费用可行流即可。注意这一遍跑出来的费用需要乘以 k。这里其实只处理了 $pk + q$ 中的 p 的平衡问题，还需要加上所有的 q 带来的费用，这是平凡的。

Gold

C. 碗窑尽空

> Fun fact：jimmyywang 没学 SAM，但是在出题时基于 SA 的 height 数组分治发明了 parent 树，但是一直找不到只能用 SA 做不能用 parent 树做的题，于是放弃了，出了这个题。

我们要求这个东西。

$s_k = \sum_{1\le i<j\le n-k+1}[S_{[i,i+k-1]} = S_{[j,j+k-1]}]f_{j-i}$

观察：$S_{[i,i+k-1]} = S_{[j,j+k-1]}$ 等价于 $\mathrm{LCP}(S_{[i,n]}, S_{[j,n]}) \ge k$。其中 LCP 表示最长公共前缀。

因此原式等价于：

$s_k = \sum_{1\le i<j\le n}[\mathrm{LCP}(S_{[i,n]}, S_{[j,n]}) \ge k]f_{j-i}$ 此处不再对 i,j 有上限限制是因为如果 $j > n-k+1$，$|S_{[j,n]}| < k$，也就自然不可能有 $\mathrm{LCP}(S_{[i,n]}, S_{[j,n]}) \ge k$。

现在我们把问题转化为求 $res_k = \sum_{1\le i<j\le n}[\mathrm{LCP}(S_{[i,n]}, S_{[j,n]}) = k]f_{j-i}$，那么 s_k 就是 res_k 的后缀和，容易处理。

现在问题转化为要求的是：

$res_k = \sum_{1\le i<j\le n} f_{j-i} \times [\mathrm{LCP}(S_{[i,n]}, S_{[j,n]}) = k]$

注意到我们其实是在给后缀求 LCP，这提示我们使用一些后缀结构（SA、SAM 的 parent 树、后缀树等）处理。这里用 parent 树举例。

众所周知 parent 树上前缀前缀节点的 lca 就是前缀的 LCS（最长公共后缀），因此反转整个串后就是后缀的公共前缀了；同时在反转后 i,j 之间距离不变，因此 f_{j-i} 不变，直接反转处理是不改变答案的。

（其实刚刚推导中可以直接使用 LCS 推导，结果是一样的，可以不用反转。）

假设我们在反转后的串上处理，此时答案变为：

$res_k = \sum_{1\le i<j\le n} f_{j-i} \times [\mathrm{LCS}(S_{[1,i]}, S_{[1,j]}) = k]$ 设前缀 $S_{[1,i]}$ 在 parent 树上的节点是 v_i，那么原式等价于：

$res_k = \sum_{1\le i<j\le n} f_{j-i} \times [len(\mathrm{lca}(v_i, v_j)) = k]$

那么显然的，我们转换成枚举 lca：

$\sum_u [len(u)=k] \sum_{v_i,v_j\in \text{subtree}(u),v_i<v_j} f_{v_j-v_i}$ 这里要求 v_i, v_j 来自不同儿子的子树（或者其一是 u 自己）。

其实这个 $[len(u)=k]$ 完全就是在说，我们求出每个 u 的 $\sum_{v_i,v_j\in \text{subtree}(u),v_i<v_j} f_{v_j-v_i}$ 后，把这个值加到 $res_{len(u)}$ 上就行。parent 树节点数是 $O(n)$ 的，因此枚举 u 已经可以接受。现在的问题就在于对每棵子树求出 $\sum_{v_i,v_j\in \text{subtree}(u),v_i<v_j} f_{v_j-v_i}$。

考虑自底向上计算。对于 u，我们计算的是子树之间的交叉贡献，这可以树上启发式合并。对每个节点 u 维护一个 u 的子树内的前缀节点集合，集合之间的启发式合并是简单的，现在考虑在这个过程中维护交叉贡献。

首先我们有 $f_x=(A^x)_{0,1}$，其中 $A=\begin{bmatrix}0&1\\3&1\end{bmatrix}$，$A_{0,1}$ 代表 A 右上角位置的值。那么 $f_{x-y}=(A^{x-y})_{0,1}=(A^x(A^{-1})^y)_{0,1}$。用一点数学知识我们知道 $A^{-1}=\begin{bmatrix}-\frac{1}{3}&\frac{1}{3}\\1&0\end{bmatrix}$。记 $B=A^{-1}=\begin{bmatrix}-\frac{1}{3}&\frac{1}{3}\\1&0\end{bmatrix}$。

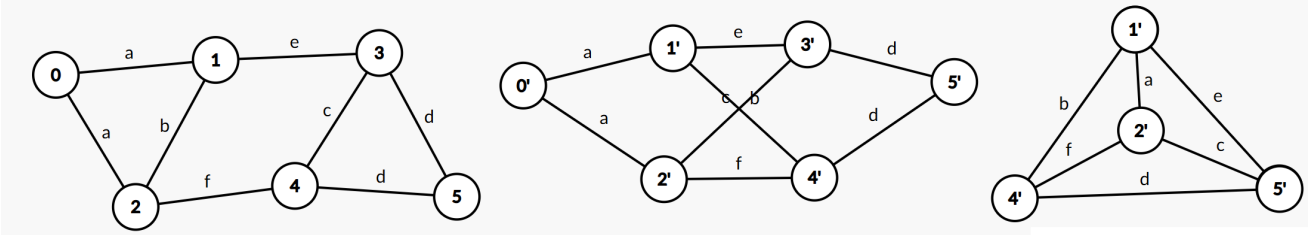
我们现在考虑在维护集合的同时维护交叉贡献。回忆集合的维护方法是找到重儿子并把其他儿子子树中的元素逐个插入重儿子的集合。那么我们就可以在插入一个元素 x 的过程中计算它与已经在集合里的元素之间产生的贡献。具体的，设当前集合为 P，加入 x 的贡献就是：

$$\sum_{y\in P,y<x} f_{x-y}+\sum_{y\in P,y>x} f_{y-x}$$
$$=\left(A^x(\sum_{y\in P,y<x} B^y)+(\sum_{y\in P,y>x} A^y)B^x\right)_{0,1}$$

开一个位置 x 上有权值 A^x 或 B^x 的两棵树状数组即可。加入 x 后也容易带修维护。要减小常数可以把 A^x 的树状数组倒序建，因为只会查后缀和。

预处理 A 和 B 的 1 到 n 次幂是 $O(n)$ 的；建立 parent 树是 $O(n)$ 的；启发式合并和树状数组各一只 $\log$，总复杂度为 $O(n\log^2 n)$。

D. 简单的图



如上图，考虑同一个环内有两条对角线的情形。

- 若对角线不相交（左图），则 $a+f+c+e=b+f+d+e=a+b=c+d$，从而 $e=f=0$。
- 若对角线相交（中图），则把它变成右图，我们有 $c+e+b+f=a+f+d+e=a+b+c+d$，同时也等于 $a+b+f=a+c+e=c+d+f$，作差可得 $b=d=e=0$。

因此一个环内所有对角线必须共用两个端点（而不能是类似“目”等）。

加上 Hikari 的条件，我们知道所有点双都必须是“口”或者“日”。

而“日”的两个端点之前恰有 3 条路径，不符合 Tairitsu 的条件，因此所有点双都是环。

现在问题变成了，给你一棵仙人掌，每次询问问从 u 到 v 的边权异或和最大值。

先随便选一棵 dfs 树，规定 $u\to v$ 的边权异或和的初始值，然后把路径上所有环的异或和加入线性基，选出最大异或和即可。

具体可以使用树剖转化成链，然后用 b_i 表示“重链头到结点 i 的所有环构成的线性基”。

特别地，线性基要记录每个数的加入时间，并且若某个位置被占用，应该让更老的数字接着往下高斯消元，这样我们就能 $O(w)$ 地查询一段重链上数字的最大异或和。

查询时，把查询路径拆成 $O(\log n)$ 条重链，先暴力合并重链上的线性基，然后再查询异或和最大值即可。

时间复杂度 $O(nw+qw^2\log n)$。

题外话：由于出题人能力有限，没有尽可能把 $O(nw+qw^2\log^2 n)$ 的树剖 + 线段树做法卡掉，导致出题人自认为更有趣的线性基部分没有得到很好的呈现。

或许出题人其实一开始就应该去掉树剖的部分，直接变成多次询问“$a_l,\ldots,a_r$ 的子序列最大异或和是多少”，就能最大限度地突出这个 idea。

K. 抽象代数

令 $U=\{1,\ldots,n\}$，定义如下二元关系：

- $a\sim_1 b$ 表示 $a\oplus b=a, b\oplus a=b$。
- $a\sim_2 b$ 表示 $a\oplus b=b, b\oplus a=a$。
- $a\sim b$ 表示 $a\sim_1 b$ 或 $a\sim_2 b$。
- $a\preceq b$ 表示 $a\oplus b=a$ 或 $b\oplus a=a$。

Lemma 1 $\sim$ 是等价关系。进一步地，$\sim$ 划分出的每个等价类 A 中，要么 $\forall a,b\in A, a\sim_1 b$，要么 $\forall a,b\in A, a\sim_2 b$。

自反性和对称性显然，下面验证传递性。设 $a \sim b, b \sim c$。不妨设 $b \notin \{a, c\}$。

- 若 $a \sim_1 b, b \sim_1 c$，则 $a \oplus c = (a \oplus b) \oplus c = a \oplus (b \oplus c) = a$，同理 $c \oplus a = c$，因此 $a \sim_1 c$。
- 若 $a \sim_2 b, b \sim_2 c$，类似地可以证明 $a \sim_2 c$。
- 若 $a \sim_1 b, b \sim_2 c$，则 $(b \oplus c) \oplus (a \oplus b) = c \oplus a \in \{a, c\}$，但另一方面，$b \oplus (c \oplus a) \oplus b$ 无论 $c \oplus a$ 是 c 还是 a，最终都得到 b，矛盾。
- 若 $a \sim_2 b, b \sim_1 c$，则 $c \sim_1 b, b \sim_2 c$，矛盾。

Remark $a \sim b \iff a \preceq b \text{ and } b \preceq a$。因此 $\preceq$ 可以被投影到 $U/\sim$ 上。

Lemma 2 $\preceq$ 是预序，且 $U/\sim$ 上，$\preceq$ 定义了一个全序。

先证明预序。自反性显然，考虑传递性。设 $a \preceq b, b \preceq c$。

- 若 $a \sim b, b \sim c$，则在 Lemma 1 中已被证明。
- 若 $a \sim b, b \nsim c$（换言之，$b \oplus c = c \oplus b = b$）：
  - 若 $a \sim_1 b$，则 $a \oplus c = (a \oplus b) \oplus c = a \oplus (b \oplus c) = a$。
  - 若 $a \sim_2 b$，则 $c \oplus a = c \oplus (b \oplus a) = (c \oplus b) \oplus a = a$。
- 若 $a \nsim b, b \sim c$：
  - 若 $b \sim_1 c$，则 $a \oplus c = (a \oplus b) \oplus c = a \oplus (b \oplus c) = a$。
  - 若 $b \sim_2 c$，则 $c \oplus a = c \oplus (b \oplus a) = (c \oplus b) \oplus a = a$。
- 若 $a \nsim b, b \nsim c$，则 $a \oplus c = (a \oplus b) \oplus c = a \oplus (b \oplus c) = a$。

证明 $\preceq$ 是序关系后，全序是显然的。

称前 K 个数是关键的，后面的数是散的。分两部分考虑：先算 $f_k(S)$ 表示当前考虑了的关键数 S 分成 k 个等价类的答案，再计算 $g_k(n)$ 表示考虑完 k 个关键等价类和 n 个散点的答案。输出方案时，计算 $\sum_{k=1}^{K} f_k(U) g_k(n-K)$ 即可。

对于每个 S 预处理好其作为前缀的合法性 $v(S)$（如果 $a \oplus b = b \oplus a = a$，那么 $b \in S, a \notin S$ 就不合法），以及其作为等价类的方案数 $c(S)$。特别地，$|S| = 1$ 时 $c(S) = 2$，因为它给答案带来 $\times 2$ 的贡献。

$v(S) = 0$ 时不产生贡献，于是我们直接把对应的 $f_k(S)$ 定义成 0，于是我们有 $f_k(S) = v(S) \sum_{T \subseteq S} c(T) f_{k-1}(S \setminus T)$。

对于 $g_k(n)$，我们有 $g_k(n) = \sum_{i=1}^{n} 2\binom{n}{i} g_k(n-i) + \sum_{i=0}^{n} \binom{n}{i} g_{k-1}(n-i)$。

其中第一个 $\sum$ 表示最后一个等价类不含当前等价类，第二个表示包含。

接下来优化 $f_k(S)$ 的计算。我们发现这玩意除了乘了个 $v(S)$ 外就是子集卷积，形式化地，$f_k = v \cdot (c * f_{k-1})$，其中 $\cdot$ 表示逐点相乘，$*$ 表示子集卷积。

直接暴力做 K 次子集卷积，时间复杂度 $O(K^3 2^K)$。

最后优化 $g_k(n)$ 的计算。

规定首字母大写后的函数时原来函数的 EGF，例如 $G_0(x) = \sum\limits_{k=0}^{n} g(k) \dfrac{x^k}{k!}$。

那么枚举 n 元集中的等价类数量 j，我们有 $G_0(x) = \sum\limits_{j \in \mathbb{N}} (2\exp(x) - 2)^j = \dfrac{1}{3 - 2\exp(x)}$。

类似地，$G_k(x) = \exp(x) \sum\limits_{j \in \mathbb{N}} G_{k-1} (2\exp(x) - 2)^j = G_{k-1} \dfrac{\exp(x)}{3 - 2\exp(x)}$。

这个直接 NTT 算就行了。

时间复杂度 $O(K^3 2^K + nK \log n)$。我特地选了一个高度封装、效率相对低下的 poly 板子，但愿没有人被卡常。