信息学竞赛中行列式的相关问题

绍兴市第一中学 王展鹏

摘 要

求解矩阵行列式是信息学竞赛中的一类经典问题，本文从今年的一道统一省选题出发，提出了两种思路，转化成了两种问题。对于求解任意矩阵的伴随矩阵的问题，本文详细介绍了基于分治以及基于递推的两种算法在不同模数情况下的应用。对于求解特殊代数结构下的矩阵行列式问题，本文讨论了元素是一次多项式时不同模数下的解法、元素是低次多项式时矩阵行列式在特殊模数下的解法、元素是低次多项式时矩阵行列式的解法。在最后，本文介绍了一个求解在任意交换环上的矩阵行列式的多项式算法及其优化。

1 引言

行列式是线性代数中的经典概念，有着优秀的性质。无论是在线性代数、多项式理论，还是在微积分学中，行列式作为基本的数学工具，都有着重要的应用。行列式在信息学竞赛中也时有考察。但在现有大多数问题中，题目往往仅仅计算要求对一个大素数取模的结果，而不涉及更深的知识与技巧。因此本文对求解行列式相关问题进行了探究，对于不同的代数系统，讨论了多种不同的思路与解法，希望能帮助填补这一方面的空白，同时也期待选手对这方面进行更深入的研究。

2 预备知识

为方便文章讨论与读者阅读，本节中将给出本文中用到的抽象代数，线性代数以及图论中的一些概念及其简单的性质。已经具备相关知识的同学可以跳过对应的部分。

2.1 二元运算与代数系统

定义 2.1.1 (二元运算)**.** 设 X 为一给定集合，则称映射 $f : X^2 \to X$ 为集合 X 上的一个二元运算。即对于任意有序对 $(a, b) \in X^2$，存在一个确定的 $h \in X$ 与之对应，记作

$$h = f(a, b)$$

或

$$h = a \circ b$$

定义 2.1.2 (交换律与结合律)**.** 对于定义在集合 X 上的二元运算 $\circ$，如果对任意的 $a, b \in X$ 都有

$$a \circ b = b \circ a$$

则称这个二元运算是交换的。若对任意的 $a, b, c \in X$ 都有

$$(a \circ b) \circ c = a \circ (b \circ c) \tag{1}$$

则称这个二元运算是结合的。

定义 2.1.3. 设集合 $S \subset X$, $\circ$ 是 X 上的二元运算，且对任意的 $a, b \in S$ 都有

$$a \circ b \in S$$

则称集合 S 对运算 $\circ$ 封闭。

定义 2.1.4 (代数系统)**.** 在其上定义了若干二元运算 $\circ, \star, \cdots$ 的集合 X 称为一个代数系统。例如上述代数系统可记作 $(X, \circ, \star, \cdots)$。

2.2 半群与群

定义 2.2.1 (半群与交换半群)**.** 称代数系统 $(X, \circ)$ 是一个半群，当且仅当二元运算 $\circ$ 是结合的。特别地，如果半群中的二元运算 $\circ$ 是交换的，则称 $(X, \circ)$ 为交换半群。

定义 2.2.2 (半群的幺元素、幺半群)**.** 半群 $(X, \circ)$ 中的元素 e 如果满足：对任意的 $a \in X$，必有

$$a \circ e = e \circ a = a \tag{2}$$

则称 e 为半群 $(X, \circ)$ 的幺元素。有幺元素的半群称为幺半群。

定理 2.2.1. 半群中的幺元素要么不存在，要么有且仅有一个。

证明. 不妨设 $e \neq e'$ 均为半群 $(X, \circ)$ 的幺元素，则由式 (2) 可知

$$e = e \circ e' = e'$$

这与 $e \neq e'$ 矛盾。因此幺元素如果存在必然唯一。 □

定义 2.2.3 (可逆元素)**.** 设幺半群 $(X, \circ)$ 的幺元素为 e，对于元素 $a \in X$，若存在 $b \in X$ 使得

$$a \circ b = b \circ a = e \tag{3}$$

则称 a 为可逆元素（或单位元素），并把 b 称作 a 的逆元素，简称逆元，记作 $b = a^{-1}$。

定理 2.2.2. 对于半群中的每一个可逆元素，其逆元素唯一。

证明. 不妨设 $b \neq b'$ 同时为半群 $(X, \circ)$ 中元素 a 的逆元素，则由式 (1) (2) (3) 可知

$$b' = (b \circ a) \circ b' = b \circ (a \circ b') = b$$

这与 $b \neq b'$ 矛盾。因此逆元素如果存在必然唯一。 □

定义 2.2.4 (群与交换群)**.** 设 $(X, \circ)$ 是一个幺半群。如果它的每个元素都有逆元，则称 $(X, \circ)$ 是群。特别地，如果二元运算 $\circ$ 是交换的，则称之为交换群。

2.3 环

定义 2.3.1 (环与交换环)**.** 设 X 是一个给定的集合，在其上定义了两种二元运算 $\oplus$ 和 $\odot$。代数系统 $(X, \oplus, \odot)$ 称为环，如果它满足以下条件：

(1) $(X, \oplus)$ 是一个交换群，通常称作环的加法群；

(2) $(X, \odot)$ 是一个半群，通常称作环的乘法半群；

(3) 对于任意的 $a, b, c \in X$，有

$$a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c)$$
$$(b \oplus c) \odot a = (b \odot a) \oplus (c \odot a)$$

特别地，如果运算 $\odot$ 也是交换的，则称这个代数系统为交换环。

定义 2.3.2 (零元素)**.** 环 $(X, \oplus, \odot)$ 的加法群 $(X, \oplus)$ 含有幺元素 o，称 o 为环 X 的零元素，本文中用 0 表示。

定义 2.3.3 (幺元素)**.** 如果环 $(X, \oplus, \odot)$ 的乘法半群 $(X, \odot)$ 含有幺元素 e，则称 e 为环 X 的幺元素，本文中用 1 表示。

定义 2.3.4 (域)**.** 如果交换环 $(X, \oplus, \odot)$ 的乘法半群 $(X, \odot)$ 中所有非零元素都有逆元素，则称 A 是域。

2.4 矩阵与行列式

定义 2.4.1 (矩阵)**.** 由 $n \times m$ 个数 $a_{i,j}$ 排成的 n 行 m 列的数表称为 n 行 m 列的矩阵，简称 $n \times m$ 矩阵。这 $n \times m$ 个数称为矩阵的元素。

对于一个 n 行 m 列的矩阵 A，设它的行号集合为 $\{1, ..., n\}$，列号集合为 $\{1, ..., m\}$。矩阵 A 第 i 行第 j 列的元素记作 $A_{i,j}$。

定义 2.4.2 (上海森堡矩阵)**.** 对于 $n \times n$ 的矩阵 A，如果对于 $\forall i > j+1$ 都有

$$A_{i,j} = 0$$

则称 A 是上海森堡矩阵。

定义 2.4.3 (行列式)**.** 对于一个 $n \times n$ 的方阵 A，将它的行列式定义为

$$\det(A) = \sum_{\sigma} (-1)^{sgn(\sigma)} \prod A_{i,\sigma_i}$$

σ 是任意一个 $1,...,n$ 的排列，$sgn(\sigma)$ 表示 σ 的逆序对数。

行列式有如下性质：

1. 在行列式中，一行（列）元素全为 0，则此行列式的值为 0；
2. 行列式中的两行（列）互换，改变行列式正负符号；
3. 在行列式中，某一行（列）有公因子 k，则可以提出 k；
4. 将一行（列）的 k 倍加进另一行（列）里，行列式的值不变。

证明直接考虑行列式的展开式即可，这里略去。

利用这些性质，可以用经典的高斯消元法在 $O(n^3)$ 级别的运算次数来计算一个定义在域上的矩阵的行列式。事实上，后三条性质对应的变换等价于左乘一个矩阵，称为初等行变换矩阵，这个变换称为初等行变换。

行列式还有另外一条经典性质：

定理 2.4.1 (行列式的乘法定理)**.** 对于任意两个 $n \times n$ 矩阵 A, B，有

$$\det(AB) = \det(A)\det(B)$$

证明. 按照高斯消元法，将矩阵 A 分解为一个上三角矩阵 A' 左乘一些初等行变换矩阵的乘积，即 $A = P_1P_2...P_kA'$，矩阵 B 分解为一个上三角矩阵 B' 右乘一些初等列变换矩阵的乘积，即 $B = B'Q_1Q_2...Q_k$。

由于两个上三角矩阵相乘，对角线上的元素也相乘，因此 $\det(A'B') = \det(A')\det(B')$，之后分类讨论证明左乘初等行变换矩阵和右乘初等列变换矩阵满足上述定理即证毕。

□

定义 2.4.4 (余子式和代数余子式)**.** 将方阵 A 去掉 $i_1, i_2, \ldots, i_k$ 行 $j_1, j_2, \ldots, j_k$ 列的矩阵的行列式称为这 k 行 k 列的余子式，记为 $A\begin{pmatrix} i_1, i_2, \ldots, i_k \\ j_1, j_2, \ldots, j_k \end{pmatrix}$，如果不特别强调，默认为去掉一行一列。将 $(-1)^{i_1+i_2+...+i_k+j_1+j_2+...+j_k} A\begin{pmatrix} i_1, i_2, \ldots, i_k \\ j_1, j_2, \ldots, j_k \end{pmatrix}$ 称为代数余子式。

定义 2.4.5 (伴随矩阵)**.** 定义 A 的伴随矩阵是一个 $n \times n$ 的矩阵，使得其第 i 行第 j 列的元素是 A 关于第 j 行第 i 列的代数余子式，记作 A^*。

定义 2.4.6 (分块矩阵与乘法)**.** 对于一个矩阵，以水平线和垂直线将其划分为若干更小的矩阵称为分块矩阵。分块矩阵中，位在同一行（列）的每一个子矩阵，都拥有相同的列数（行数）。左矩阵的列分块完全和右矩阵的行分块相同时，可以做分块乘法。规则类似于普通的矩阵乘法，相当于将每一块矩阵看作一个元素，元素乘法对应矩阵乘法。

2.5 图论

定义 2.5.1 (基尔霍夫矩阵)**.** 令 A 是图的邻接矩阵，D 是图的度数矩阵，则基尔霍夫矩阵 $L = D - A$。

定理 2.5.1 (矩阵树定理)**.** 一个图的生成树个数等于它的基尔霍夫矩阵的任意一行一列的代数余子式。

3 问题引入

3.1 题目描述

给定一个 n 个顶点 m 条边（点和边都从 1 开始编号）的无向图 G，保证图中无重边和无自环。每一条边有一个正整数边权 w_i，对于一棵 G 的生成树 T，定义 T 的价值为：T 所包含的边的边权的最大公约数乘以边权之和，即：

$$val(T) = \left(\sum_{i=1}^{n-1} w_{e_i}\right) \times \gcd(w_{e_1}, w_{e_2}, \ldots, w_{e_{n-1}})$$

其中 $e_1, e_2, \ldots, e_{n-1}$ 为 T 包含的边的编号。

询问 G 的所有生成树 T 的价值之和对 $P = 998244353$ 取模后的结果。

$2 \le n \le 30, 1 \le m \le \frac{n(n-1)}{2}, 1 \le w_i \le 152501$

3.2 约定

在下文中，我们认为 $O(\log P)$ 是远小于 $O(n)$ 的复杂度，并且令 $W = \max(w_i)$。

3.3 题目讨论

该题来自于 CCF 统一省选二试 A 卷第三题。

定义 S 表示所有生成树构成的集合。先做一些简单的转化：

$$\begin{aligned}
&\sum_{T\in S}\left(\sum_{i=1}^{n-1} w_{e_i}\right)\times \gcd(w_{e_1}, w_{e_2}, \ldots, w_{e_{n-1}}) \\
=&\sum_{T\in S}\left(\sum_{i=1}^{n-1} w_{e_i}\right)\times\left(\sum_{d\mid \gcd(w_{e_1},w_{e_2},\ldots,w_{e_{n-1}})} \varphi(d)\right) \\
=&\sum_{d=1}^{W}\varphi(d)\times\left(\sum_{T\in S, d\mid w_{e_1},\ldots,d\mid w_{e_{n-1}}}\sum_{i=1}^{n-1} w_{e_i}\right)
\end{aligned}$$

也就是说，对于所有 d，只要计算当只考虑权值是 d 倍数的边时，所有生成树边权和的和。

自然的想法是，枚举所有边 (a,b)，计算它的出现次数，显然等于总生成树个数减去去掉这条边以后的个数。生成树计数可以利用矩阵树定理转化为求行列式，时间复杂度 $O(mn^3)$，难以接受。

注意到减去一条边会导致基尔霍夫矩阵 $O(1)$ 个元素值变化。如下图所示矩阵所示。

$$\begin{bmatrix}
\ddots & \vdots & \cdots & \vdots & \iddots \\
\cdots & A_{u,u}\to A_{u,u}-1 & \cdots & A_{u,v}\to A_{u,v}+1 & \cdots \\
\cdots & \vdots & \ddots & \vdots & \cdots \\
\cdots & A_{v,u}\to A_{v,u}+1 & \cdots & A_{v,v}\to A_{v,v}-1 & \cdots \\
\iddots & \cdots & \cdots & \vdots & \ddots
\end{bmatrix}$$

因此可以简单讨论一下这四个位置的权值修改对行列式产生的影响。

考虑依次修改 $A_{u,u}, A_{u,v}, A_{v,u}, A_{v,v}$ 位置的值，实时计算新矩阵的行列式，容易发现行列式变化值依次是

$$-(-1)^{u+u}A\begin{pmatrix}u\\u\end{pmatrix},\ (-1)^{u+v}A\begin{pmatrix}u\\v\end{pmatrix},\ (-1)^{v+u}A\begin{pmatrix}v\\u\end{pmatrix}-A\begin{pmatrix}u,v\\u,v\end{pmatrix},\ -\left((-1)^{v+v}A\begin{pmatrix}v\\v\end{pmatrix}-A\begin{pmatrix}u,v\\u,v\end{pmatrix}\right)$$

注意到 $A\begin{pmatrix}u,v\\u,v\end{pmatrix}$ 项正负抵消，因此总的变化量恰好就是四个位置的代数余子式乘上对应的变化量的和。至此，问题转化为计算出所有位置的代数余子式。

另一个想法考虑问题的组合意义，最终的生成树恰好由一条固定计算权值的边和其余 $n-2$ 条边构成，那么可以将一条边的权值看作 $w_i x+1$，计算基尔霍夫矩阵任意代数余子式的一次项系数就是问题的答案。也就是说，问题变为求元素是一次多项式的矩阵的行列式的一次项系数。

因此，我们将原问题转化成了上述两个问题，接下来，本文将详细讨论这两个问题并加以推广。

4 取模意义下求解伴随矩阵

4.1 拆分问题

考虑如何计算删除一行一列的余子式，我们将算法分成两部分：

1. 枚举删除哪一行，计算剩余矩阵消元后的形式，类似下图。

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n-2} & a_{1,n-1} & a_{1,n} \\ 0 & a_{2,2} & a_{2,3} & \cdots & a_{2,n-2} & a_{2,n-1} & a_{2,n} \\ 0 & 0 & a_{3,3} & \cdots & a_{3,n-2} & a_{3,n-1} & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{n-2,n-2} & a_{n-2,n-1} & a_{n-2,n} \\ 0 & 0 & 0 & \cdots & 0 & a_{n-1,n-1} & a_{n-1,n} \end{bmatrix}$$

2. 在枚举行的基础上，枚举删除哪一列，可以发现剩余矩阵的行列式等于一个上三角矩阵的行列式乘上一个上海森堡矩阵的行列式，也就是说，需要计算对于任意右下角 $i \times i$ 子矩阵的行列式。

由于本算法扩展性比较强，因此下文认为模数可以是任意数。

4.2 第一部分

考虑使用分治做法来解决这个问题。使用 $solve(l,r)$ 表示计算当删除的行处于区间 $[l,r]$ 内时，每行消元后的矩阵。

考虑每次递归到 $solve(l,mid)$ 和 $solve(mid+1,r)$，如果递归到 $solve(mid+1,r)$，那么可以将 $l \sim mid$ 这些行消成类似于上三角矩阵的形式，递归另一部分；如果递归到 $solve(l,mid)$，可以将 $l \sim mid$ 和 $mid+1 \sim r$ 整体交换后采用类似于上面的做法解决。

实现的时候，可以将下面的行换到上面，乘上对应的系数就可以了。

同时，由于模数可能是合数，因此消元过程需要使用辗转相除法，即将消元的那一列辗转相减。

4.3 第二部分

下图就是一个上海森堡矩阵：

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n-1} & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n-1} & a_{2,n} \\ 0 & a_{3,2} & a_{3,3} & \cdots & a_{3,n-1} & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{n,n-1} & a_{n,n} \end{bmatrix}$$

对于第二部分，相当于要求一个上海森堡矩阵任意右下角 $i \times i$ 子矩阵的行列式，这是一个经典问题。

令 f_i 表示右下角 $i \times i$ 子矩阵的行列式，考虑将行列式展开，可以发现一旦选择了第一行，就会变成一个子问题，因此

$$f_i = a_{i,i} f_{i+1} - a_{i+1,i} a_{i,i+1} f_{i+2} + a_{i+1,i} a_{i+2,i+1} a_{i,i+2} f_{i+3} + \ldots$$

4.4 复杂度分析

对于第一部分，要消元的次数是递归到的区间长度和加上辗转相除时交换行的次数，容易发现区间长度和是 $O(n \log n)$。辗转相除部分每当一行确定不删以后（即当 $solve(l, r)$ 时，不在这个区间内的行都确定不删），每次用辗转相除法消元时，类似于求最大公约数，除了第一次交换以外，每被交换一次，第一个非零位置至少除以二，因此额外的消元次数 $O(n \log n \log P)$。

每次消元复杂度是 $O(n)$，因此第一部分复杂度是 $O(n^3 \log n + n^2 \log n \log P)$。

对于第二部分，需要执行 n 次，每次是一个 $O(n^2)$ 的 DP，因此复杂度为 $O(n^3)$ 。

忽略不是瓶颈的部分，总的复杂度 $O(n^3 \log n)$。

4.5 从整体来看

注意到，我们是要计算 $n \times n$ 个答案，能不能用一些位置的答案来推出另一些位置呢？有定理：

定理 4.5.1. 令 I 表示单位矩阵，那么对于任意方阵 A，满足：$AA^* = \det(A)I$。

证明. 将上式展开，得到

$$\forall 1 \le i, j \le n, \sum_{k=1}^{n} A_{i,k} \times a_{j,k} (-1)^{j+k} = [i = j] \times \det(A)$$

。

考虑将行列式按第 j 行展开的形式：$\sum_{k=1}^{n} A_{j,k} \times a_{j,k} (-1)^{j+k}$。

比较两个式子，左式结果实际相当于用第 i 行覆盖第 j 行后的行列式，因此显然等于右式。 □

4.6 回到质数

不妨重新思考模数是质数的情况，分类讨论矩阵的秩（令 n 为矩阵阶数）：

$rank(A) = n$：矩阵有逆，由 $AA^* = \det(A)I$ 可以得到 $A^* = \det(A)A^{-1}$，直接使用高斯消元算法计算逆矩阵和行列式即可；

$rank(A) \le n-2$：容易发现，无论如何去掉一行一列，剩余矩阵的秩都不会变大，因此任意位置的余子式一定是 0；

$rank(A) = n-1$：仍然考虑 $AA^* = \det(A)I$ 这一等式，因此可以对 A 进行初等行变换以实现高斯消元。我们知道，对矩阵进行初等行变换等价于左乘初等行变换矩阵。也就是说，左乘一个行变换矩阵 C，使得最后等式形式变为：$A'A^* = \det(A)C$，其中 A' 是消元后的矩阵。

具体实现时，假设处理到第 k 行第 k 列时，如果 $\forall i \ge k, A_{i,k} = 0$，那么最终必然 $A_{k,k} = 0$。由等式可以得出对于 $\forall i < k, j$，满足

$$A^*_{i,j} = \frac{0 - \sum_{i<l\le n} A'_{i,l}A^*_{l,j}}{A'_{i,i}}$$

由于秩是 $n-1$，因此之后还没消过的大小为 $(n-k+1)\times(n-k)$矩阵上必然列满秩，容易发现 $\forall i > k, j$，满足 $A^*_{i,j} = 0$。所以最后只需暴力计算第 k 行的余子式，就可以直接递推出其余所有行了。

朴素地计算一行余子式是 $O(n^4)$ 的，令人难以接受。不过容易发现，由于是同一行的余子式，可以先删掉这一行，之后利用之前的做法，先消元以后删掉任意一列就又满足上海森堡矩阵的性质了，复杂度仍然是 $O(n^3)$。

但事实上可以用 $A^*A = \det(A)I$ 类似地通过一列来推每一列，由于本题矩阵的对称性，并不需要再进行一次高斯消元，因为一行推其他行和一列推其他列的系数显然是相同的。因此，只需要暴力计算一个位置的余子式就可以了，时间复杂度同样是 $O(n^3)$。

4.7 递推方法

可以发现，不管模数是素数还是合数，我们的目标都是得到 $A'A^* = \det(A)C$，其中 A' 是上三角矩阵，对于$\forall i, j$，可以得到

$$\sum_{k\ge i} A'_{i,k}A^*_{k,j} = \det(A)C_{i,j}$$

容易发现，能不能推出 $A^*_{i,j}$，取决于 $A'_{i,i}$ 是否存在逆元。如果存在，就可以通过已经算过的位置来递推此位置，同样地，如果用递推列的方法得到 $A'_{j,j}$ 存在逆元，也可以推出此位置。因此，只要算出所有 i, j 满足 $A'_{i,i}, A'_{j,j}$ 都没有逆元的位置的余子式，就可以推出所有的余子式了。用 s 表示这样的列个数，自然，我们希望这样的 s 不太大。

4.8 最终算法

考虑能不能将之前的算法扩展到合数的情况。

由于模合数意义下的运算不是整环，因此不能用传统的方法讨论矩阵的秩。

但我们仍然可以用高斯消元来处理 $AA^* = \det(A)I$，用辗转相除法来消元，如果用辗转相除法求得的 gcd 与模数不互质，因为没有逆元，我们就不能用之前的递推方法来推这一行的伴随矩阵了。为了使得结果是上三角矩阵，可以将没有逆元的列换到最后，同时这也保证了那些没有逆元的列都在最后。为了方便，之后就假设 A' 满足这样的性质。

定理 4.8.1. *对 P 质因数分解后，对于它的因子 p^k，如果按上述求得的 $\gcd \bmod p = 0$ 列的数量大于 k，那么所有要求的余子式模 $p^k = 0$。*

证明. 考虑把模数当成 p，如果满足上述条件，求得矩阵的秩显然 $< n - k$，同样地，去掉一行一列不会使矩阵的秩变大，因此最后消元以后的主对角线上至少有 k 个 p 的倍数，即这个位置的余子式 $\bmod p^k = 0$。 □

由上述定理，可以发现如果所有余子式模 $p^k = 0$，那么我们就可以去掉这个因子。设简化过后的模数 $P = \prod p_i^{k_i}$，那么 $s \le \sum k_i \le \log_2 P$。

朴素地求 s^2 个位置的余子式，时间复杂度 $O(n^3 s^2)$。还是不太能令人满意。为了方便讨论与实现，可以将这些位置交换到矩阵的右下角，只要求出这些位置的余子式就可以了。我们可以先将左上角消成上三角形式，如下图（为了方便，令 $t = n - s$）：

$$\left[\begin{array}{cccc|ccc} a_{1,1} & a_{1,2} & \cdots & a_{1,t} & a_{1,t+1} & \cdots & a_{1,n} \\ 0 & a_{2,2} & \cdots & a_{2,t} & a_{2,t+1} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{t,t} & a_{t,t+1} & \cdots & a_{t,n} \\ \hline a_{t+1,1} & a_{t+1,2} & \cdots & a_{t+1,t} & a_{t+1,t+1} & \cdots & a_{t+1,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,t} & a_{n,t+1} & \cdots & a_{n,n} \end{array}\right]$$

之后枚举删除哪一行，可以将其余行消掉

$$\left[\begin{array}{cccc|ccc} a_{1,1} & a_{1,2} & \cdots & a_{1,t} & a_{1,t+1} & \cdots & a_{1,n} \\ 0 & a_{2,2} & \cdots & a_{2,t} & a_{2,t+1} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{t,t} & a_{t,t+1} & \cdots & a_{t,n} \\ \hline 0 & 0 & \cdots & 0 & a_{t+1,t+1} & \cdots & a_{t+1,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & a_{i-1,t+1} & \cdots & a_{i-1,n} \\ 0 & 0 & \cdots & 0 & a_{i+1,t+1} & \cdots & a_{i+1,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & a_{n,t+1} & \cdots & a_{n,n} \end{array}\right]$$

最后再枚举删除哪一列，朴素使用 s^3 的高斯消元计算右下角的行列式再乘上 $\prod_{1\le i\le t} a_{i,i}$ 就行了。总的复杂度 $O(n^3 + n^2 \log^2 P + \log^5 P)$。

当然，我们还能继续优化。仍然采用之前 $O(n^3 \log n)$ 算法的思想，把左上角消完以后，用分治算法消最后的 s 行，最后会剩下 $s \times s$ 的类似上海森堡矩阵的矩阵，依然可以使用消上海森堡矩阵的方法求出每个位置的余子式，总的复杂度是 $O(n^3 + n^2 \log P \log\log P)$。

这样的复杂度已经几乎不比朴素求行列式的 $O(n^3)$ 复杂度劣了。至此，在任意模数下，对于求解任意矩阵的伴随矩阵的问题，我们得到了一个复杂度令人满意的做法。

4.9 其他思路

可以发现，到最后一步以后，问题转化为求右下角一个方阵所有位置的代数余子式。考虑给矩阵加一行，加一列，那么对于 (x, y) 位置的代数余子式，等于形如下面矩阵的行列式的相反数。

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,y} & \cdots & a_{1,n-1} & a_{1,n} & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,y} & \cdots & a_{2,n-1} & a_{2,n} & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ a_{x,1} & a_{x,2} & a_{x,3} & \cdots & a_{x,y} & \cdots & a_{x,n-1} & a_{x,n} & 1 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & a_{n,3} & \cdots & a_{n,y} & \cdots & a_{n,n-1} & a_{n,n} & 0 \\ 0 & 0 & 0 & \cdots & 1 & \cdots & 0 & 0 & 0 \end{bmatrix}$$

这是因为根据定义计算行列式时，每行每列需要恰好选择一个位置，那么为了最后乘积非零，显然必须得选这两个 1，剩余自然就是 (x, y) 的余子式了。

由于左上角 $n \times n$ 的矩阵不随着 x, y 的变化而变化，因此可以对左上角进行高斯消元，并记录消元的结果，当固定 (x, y) 以后矩阵除了 $a_{n+1,v} = 1$ 以外，一定是一个上三角矩阵。因此只需利用高斯消元使得最后一行也满足上三角矩阵性质就可以计算行列式了。由于 $v > n - s$，考虑到极端情况辗转相除每次可能会进行 $\log P$ 次，时间复杂度 $O(n^3 + s^4 \log P) = O(n^3 + log^5 P)$，代码实现比起之前的算法更简洁。如果不结合之前的递推算法，直接计算每个位置的余子式，通过一些技巧也容易做到 $O(n^3 + n^2 \log^3 P)$

5 取模意义下计算元素是多项式的矩阵行列式的一次项

5.1 朴素实现

当模数是质数时，可以使用经典的高斯消元算法计算行列式。

在高斯消元过程中，可以用最低非零次项最小的多项式所在的行来消其余行，将矩阵消成上三角形式就可以直接计算行列式了。

由于最后只需求一次项，全过程可以在 mod x^2 过程下进行，总的复杂度 $O(n^3)$。

5.2 扩展到合数

如果模数是合数，可以发现很难将之前的算法扩展。

例如在 mod 8 意义下，如下图所示矩阵。

$$\begin{bmatrix} 4 & a \\ x+2 & b \end{bmatrix}$$

可以注意到，我们很难使用 4 消掉 $x + 2$。

不过根据上一个问题 $O(n^3 \log n)$ 复杂度算法给我们的启示：如果开始将矩阵消成较简的形式，最后的求行列式部分可能就会简单得多。

可以发现，在模合数意义下，虽然不能使用高斯消元将以一次函数为元素的矩阵消成上三角，但可以仅仅将常数项消成上三角。

如下图

$$\begin{bmatrix} a_{1,1}x + b_{1,1} & a_{1,2}x + b_{1,2} & a_{1,3}x + b_{1,3} & \cdots & a_{1,n}x + b_{1,n} \\ a_{2,1}x & a_{2,2}x + b_{2,2} & a_{2,3}x + b_{2,3} & \cdots & a_{2,n}x + b_{2,n} \\ a_{3,1}x & a_{3,2}x & a_{3,3}x + b_{3,3} & \cdots & a_{3,n}x + b_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1}x & a_{n,2}x & a_{n,3}x & \cdots & a_{n,n}x + b_{n,n} \end{bmatrix}$$

仍然可以枚举选择哪个 x，贡献就是该位置的代数余子式。容易注意到，对于主对角线上的位置，余子式就是主对角线上其余常数项的乘积，如果是没有常数项的位置，那么余子式就是两条对角线常数项和一个只考虑常数项的上海森堡矩阵行列式的乘积，对于右上角的位置，余子式则显然是 0。

对于一个上海森堡矩阵，对于任意 s,t，需要求左上角为 (s,s) 右下角为 (t,t) 的子矩阵行列式。那么可以枚举任意左上角 $t\times t$ 的子矩阵，这显然仍然是一个上海森堡矩阵，然后计算所有右下角 $(t-s+1)\times(t-s+1)$ 子矩阵行列式，这与要求的所有行列式一一对应。综上，我们仍然在 $O(n^3)$ 的复杂度内解决了这个问题。

6 取模意义下计算元素是多项式的矩阵的行列式

6.1 问题介绍

回顾之前做法，可以发现：我们虽然得到了较为优秀的复杂度，但仍然使用了很取巧的思路：枚举选择的一次项位置。实际问题中，更常见的是需要求出整个行列式，而上述算法就难以再次扩展。考虑尝试其它的思路。

为了方便，本节认为矩阵元素的多项式次数是 $O(1)$ 的。事实上，如果设多项式最高次数为 d，复杂度往往只要再乘一个 d 就好了。

6.2 基于拉格朗日插值的做法

考虑最终行列式的形式，显然是一个次数 $O(n)$ 的多项式。因此考虑使用经典的拉格朗日插值算法还原出这个行列式。

将模数质因数分解，设 $P=\prod p_i^{k_i}$，考虑分别计算出对 $p_i^{k_i}$ 取模后的结果后再用中国剩余定理合并。

考虑拉格朗日插值的形式：

$$f(x)=\sum y_i\prod_{i\neq j}\frac{(x-x_i)}{x_j-x_i}$$

容易注意到，当 $\forall j<k, x_j\not\equiv x_k(\mathrm{mod}\,p_i)$ 时，由于除法有逆元，因此只要点值比次数大，就能插出多项式。

沿用之前的证明思路，假设模数是素数 p_i，如果存在 $x_j\equiv x_k(\mathrm{mod}\,p_i)$，这说明在模 p_i 意义下，有一个点值是没有意义的。也就是说，想要解出系数对 p_i 取模的结果，也需要这么多点值，这也就说明了这是充分条件。

令答案多项式次数为 $k-1$，显然当 $k\leq\min(p_i)$ 时，只需选取 $0,...,k-1$ 这些点值暴力计算行列式就可以算出答案；但当 $k>\min(p_i)$ 时，即使选取了所有 $0,...,P-1$ 的点值也不足以解出行列式。

要解决这个问题，必须要扩大模数，在 $P = 2^k$ 的最坏情况下，模数的位数会达到 $O(n)$ 级别，需要维护高精度数，总的复杂度高达 $O(n^4 \times f(n))$，$f(n)$ 表示 n 位高精度数运算的复杂度。

综上，拉格朗日插值在模数最小质因子较大时比较优秀，而在其余情况下不那么令人满意。

6.3 基于中国剩余定理的做法

上一节中，在模数比较小的情况下，由于没有逆元除法很难进行。

那么不妨考虑一个更暴力的想法，去掉取模，直接计算答案。

由于现在没有取模，我们可以方便地进行插值，因此只需要计算元素是整数的矩阵行列式就可以了。

既然模数是素数的情况能够方便的计算，要将不取模的情况转化为素数，让人联想到中国剩余定理。

令最终行列式位数级别为 $g(n)$，根据定义，容易发现 $g(n) \le n \log n$，实际上这个上界并不紧，更紧的上界可以参见参考文献 [2]，由于篇幅有限，这里略去。

我们需要用 $O(g(n))$ 个大质数来还原答案，每次复杂度 $O(n^3)$，再结合插值，总的复杂度 $O(n^4 g(n))$。

7 计算定义在任意交换环上的矩阵的行列式

7.1 算法思路

总结以上做法，发现上述算法都要用到高精度数，在信息学竞赛中难以在考场上实现。并且，两个算法都需要用到多项式运算的性质，在最后，我们更想要一个能在任意交换环上运行的算法。

定义新的矩阵 $B(z) = I + z(A - I)$，容易注意到 $B(0) = I, B(1) = A$。

也就是说，如果计算出 $B(z)$ 的行列式，直接计算系数和就是答案了。根据定义，矩阵 $B(z)$ 的形式如下图。

$$\begin{bmatrix} (a_{1,1}-1)z+1 & a_{1,2}z & a_{1,3}z & \cdots & a_{1,n}z \\ a_{2,1}z & (a_{2,2}-1)z+1 & a_{2,3}z & \cdots & a_{2,n}z \\ a_{3,1}z & a_{3,2}z & (a_{3,3}-1)z+1 & \cdots & a_{3,n}z \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1}z & a_{n,2}z & a_{n,3}z & \cdots & (a_{n,n}-1)z+1 \end{bmatrix}$$

考虑直接实现高斯消元法，由于矩阵只有主对角线上元素有常数项 1，在算法执行过程中，我们可以保证矩阵一直满足这个性质。因此这些元素总是有逆元，且求逆只需用到幺元、乘法和加法，符合交换环的性质，困扰我们的问题就被解决了。

观察答案形式容易注意到最终次数不超过 n，因此全过程可以对 z^{n+1} 取模，时间复杂度瓶颈在于 $O(n^3)$ 次两个交换环上的多项式卷积，朴素实现需要 $O(n^5)$ 次交换环上的运算。

事实上，交换环卷积算法可以在 $O(n\log n\log\log n)$ 内完成，具体算法可以见参考文献[1]，由于该算法跟本文关系不大，这里略去。

综上，我们得到了一个 $O(n^4\log n\log\log n)$ 的优秀算法。

7.2 矩阵乘法

众所周知，矩阵乘法可以做到 $O(n^\omega)$ 的复杂度，其中 $\omega\approx 2.373$，这个算法可以见参考文献 [4]，由于其过于复杂且和本文关系不大，这里不作展开。在实际应用中，往往使用相对好写且常数优秀的 $O(n^{2.807})$ 算法作为代替，下文对此做简单介绍。

为了方便，将矩阵扩充到 $2^k\times 2^k$ 大小，多的元素用零填充。设求 $C=A\times B$。

将每个矩阵分成规模为 $2^{k-1}\times 2^{k-1}$ 的四块：

$$A=\begin{bmatrix}A_{1,1} & A_{1,2}\\ A_{2,1} & A_{2,2}\end{bmatrix},\quad B=\begin{bmatrix}B_{1,1} & B_{1,2}\\ B_{2,1} & B_{2,2}\end{bmatrix},\quad C=\begin{bmatrix}C_{1,1} & C_{1,2}\\ C_{2,1} & C_{2,2}\end{bmatrix}$$

我们有：

$$\begin{aligned}C_{1,1}&=A_{1,1}B_{1,1}+A_{1,2}B_{2,1}\\ C_{1,2}&=A_{1,1}B_{1,2}+A_{1,2}B_{2,2}\\ C_{2,1}&=A_{2,1}B_{1,1}+A_{2,2}B_{2,1}\\ C_{2,2}&=A_{2,1}B_{1,2}+A_{2,2}B_{2,2}\end{aligned}$$

引入新矩阵：

$$\begin{aligned}M_1&:=(A_{1,1}+A_{2,2})(B_{1,1}+B_{2,2})\\ M_2&:=(A_{2,1}+A_{2,2})B_{1,1}\\ M_3&:=A_{1,1}(B_{1,2}-B_{2,2})\\ M_4&:=A_{2,2}(B_{2,1}-B_{1,1})\\ M_5&:=(A_{1,1}+A_{1,2})B_{2,2}\\ M_6&:=(A_{2,1}-A_{1,1})(B_{1,1}+B_{1,2})\\ M_7&:=(A_{1,2}-A_{2,2})(B_{2,1}+B_{2,2})\end{aligned}$$

可得：

$$C_{1,1} = M_1 + M_4 - M_5 + M_7$$
$$C_{1,2} = M_3 + M_5$$
$$C_{2,1} = M_2 + M_4$$
$$C_{2,2} = M_1 - M_2 + M_3 + M_6$$

根据主定理，总共需要 $T(n) = 7T(n/2) + O(n^2) \approx O(n^{2.807})$ 元素运算。

7.3 矩阵求逆

类似地，将矩阵扩充到 $2^k \times 2^k$ 大小，多出来的右下角部分可以用单位矩阵填充，显然新矩阵逆的左上角部分就是原矩阵的逆。

考虑仍然将矩阵分块，即将矩阵表示为 $M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ 的形式，定义其转换矩阵：

$$L = \begin{bmatrix} I_p & 0 \\ -D^{-1}C & D^{-1} \end{bmatrix}$$

其中 I_p 表示一个 $p \times p$ 的单位矩阵。矩阵 M 右乘转换矩阵 L，具体的形式是：

$$M \cdot L = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} I_p & 0 \\ -D^{-1}C & D^{-1} \end{bmatrix} = \begin{bmatrix} A - BD^{-1}C & BD^{-1} \\ 0 & I_q \end{bmatrix}$$

事实上，左上角的 $A - BD^{-1}C$ 称为原矩阵的舒尔补。因此，矩阵M的逆，如果存在的话，可以用 D^{-1} 以及其舒尔补（如果存在的话）来表示：

$$\begin{aligned} \begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} &= \begin{bmatrix} I & 0 \\ -D^{-1}C & D^{-1} \end{bmatrix} \begin{bmatrix} (A - BD^{-1}C)^{-1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & -BD^{-1} \\ 0 & I \end{bmatrix} \\ &= \begin{bmatrix} A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1} & -A^{-1}B(D - CA^{-1}B)^{-1} \\ -(D - CA^{-1}B)^{-1}CA^{-1} & (D - CA^{-1}B)^{-1} \end{bmatrix} \end{aligned}$$

由本算法的性质，容易注意到 D 和 $D - CA^{-1}B$ 永远可逆，这是因为无论何时，矩阵都只有在主对角线上有常数项 1。也就是说，我们将问题转化为了规模更小的矩阵求逆和矩阵乘法。一个 n 阶矩阵的逆，可以转化为两个 $n/2$ 阶矩阵求逆和多次矩阵乘法和加法，总计需要 $T(n) = 2T(n/2) + O(n^\omega) = O(n^\omega)$ 元素运算。

7.4 矩阵的行列式

同样地，将矩阵扩充到 $2^k \times 2^k$ 大小，多出来的右下角部分可以用单位矩阵填充，显然行列式不变。对于矩阵 $\begin{bmatrix} A & B \\ C & D \end{bmatrix}$，我们还有：

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} I & -A^{-1}B \\ 0 & I \end{bmatrix} = \begin{bmatrix} A & 0 \\ C & D - C \times A^{-1}B \end{bmatrix}$$

所以

$$\begin{aligned} \det\left(\begin{bmatrix} A & B \\ C & D \end{bmatrix}\right) &= \det\left(\begin{bmatrix} A & 0 \\ C & D - C \times A^{-1}B \end{bmatrix}\right) \\ &= \det(A)\det(D - C \times A^{-1}B) \end{aligned}$$

同样由于本算法的性质，A 和 $D - CA^{-1}B$ 都一定可逆，我们将行列式转化为了规模更小的行列式和求逆以及乘法问题，总计需要 $T(n) = 2T(n/2) + O(n^\omega) = O(n^\omega)$ 次元素运算。

7.5 优化

综上，矩阵的行列式可以用 $O(n^\omega)$ 次元素的运算解决，再结合矩阵元素乘法的 $O(n \log n \log\log n)$ 算法，总的复杂度 $O(n^{\omega+1} \log n \log\log n)$。

7.6 上述算法的扩展

事实上，对于任意定义在实数域上的矩阵都可以用类似的方法求逆，我们有

$$A^{-1} = A^T(AA^T)^{-1}$$

当 A 可逆时，可以证明 AA^T 由于有着优秀的性质，可以套用之前的算法。

对于行列式，如果用相同的方法，可以求出原行列式的平方，而开方会导致有两个答案，这使得问题变得十分复杂，有兴趣的读者可以自行阅读参考文献 [6]，本文不做展开。

不过即使如此，仍然可以用简单的随机化方法使得矩阵 A 可逆，如随机一个矩阵 F 对原矩阵做矩阵乘法来代替初等列变换，使得 $A' = A + BF, C' = C + DF$，可以取得不错的效果。

8 展望

事实上，本文提到的很多算法都可以进一步优化，例如使用大步小步递推矩阵数列、HALF-GCD 等方法。但由于本人数学水平实在有限，因此没有做更深的探讨。总而言之，希望本文可以起到抛砖引玉的作用，也希望有兴趣的读者，能够进一步研究。

9 致谢

感谢中国计算机学会提供学习和交流的平台。

感谢绍兴一中的陈合力老师和董烨华老师的关心和指导。
感谢国家集训队教练高闻远的指导和帮助。
感谢李佳衡、孟煜皓同学为本文审稿。
感谢其他对我有过帮助和启发的老师和同学。
感谢父母对我的关心、支持与无微不至的照顾。

10 参考文献

[1] David G. Cantor and Erich Kaltofen, ON FAST MULTIPLICATION OF POLYNOMIALS OVER ARBITRARY ALGEBRAS. Acta Informatica vol. 28, nr. 7, pp. 693 - 701 (1991).

[2] Erich Kaltofen and Gilles Villard. ON THE COMPLEXITY OF COMPUTING DETERMINANTS, comput. complex. 13 (2004), 91 - 130.

[3] Gilles Villard. Exact computation of the determinant and of the inverse of a matrix.Workshop on Complexity — FoCM Minneapolis, Aug. 12, 2002.

[4] Williams, Virginia Vassilevska (2012-05-19). Multiplying matrices faster than coppersmith-winograd. Proceedings of the 44th symposium on Theory of Computing - STOC '12. ACM. pp. 887 - 898.

[5] Gilles Villard. Kaltofen's division-free determinant algorithm differentiated for matrix adjoint computation. Journal of Symbolic Computation, Elsevier, 2011, 46 (7), pp.773-790.

[6] James R. Bunch and John E. Hopcroft. Triangular Factorization and Inversion by Fast Matrix Multiplication. mathematics of computation, volume 28, number 125, January, 1974.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. Introduction to Algorithms, 3rd ed. MIT Press, Cambridge, MA, 2009, §28.2.

[8] VOLKER STRASSEN. Gaussian Elimination is not Optimal. Numer. Math. t3, 354–356 (t969).

[9] Gilles Villard. Computation of the Inverse and Determinant of a Matrix. Algorithms Seminar 2001 - 2002, F. Chyzak (ed.), INRIA, (2003), pp. 29 - 32.

[10] Wikipedia, Schur complement.