# 线性代数

___debug

2018 年 1 月 5 日

# 矩阵乘法

LOJ 2325

维护一个集合，初始有两个数 $\{\infty, m\}$. 进行 $n$ 次操作，每次随机选一个数，把它减一. 如果结果为 $0$，把它删掉；否则，如果集合大小不超过 $k$，则添加一个 $m$. 最后问那个 $\infty$ 期望被减了多少次.
$T$ 组询问，求出 $n$ 不同时的答案.

$T \leq 1000, n \leq 10^{18}, m \leq 3, k \leq 8$

# 矩阵乘法

## LOJ 2325 - 题解

首先暴力表示出状态, 一共也就 $w = 165$ 种.

# 矩阵乘法

## LOJ 2325 - 题解

首先暴力表示出状态，一共也就 $w = 165$ 种.
然后可以写出一个转移矩阵，直接矩阵快速幂即可.

# 矩阵乘法

## LOJ 2325 - 题解

首先暴力表示出状态，一共也就 $w = 165$ 种.
然后可以写出一个转移矩阵，直接矩阵快速幂即可.

但是这样做会 TLE，因为询问太多了.

# 矩阵乘法

## LOJ 2325 - 题解

首先暴力表示出状态，一共也就 $w = 165$ 种.
然后可以写出一个转移矩阵，直接矩阵快速幂即可.

但是这样做会 TLE，因为询问太多了.
不过有一个经典的技巧可以加速这一过程.

# 矩阵乘法

## LOJ 2325 - 题解

首先暴力表示出状态，一共也就 $w = 165$ 种.
然后可以写出一个转移矩阵，直接矩阵快速幂即可.

但是这样做会 TLE，因为询问太多了.
不过有一个经典的技巧可以加速这一过程.
我们先预处理矩阵的 $2^t$ 次方，然后询问时相当于一堆矩阵乘一个向量.

# 矩阵乘法

## LOJ 2325 - 题解

首先暴力表示出状态，一共也就 $w = 165$ 种.
然后可以写出一个转移矩阵，直接矩阵快速幂即可.

但是这样做会 TLE，因为询问太多了.
不过有一个经典的技巧可以加速这一过程.
我们先预处理矩阵的 $2^t$ 次方，然后询问时相当于一堆矩阵乘一个向量. 由于矩阵乘法具有结合性，我们先将最右边的矩阵乘上向量，得到的还是一个向量，继续这一过程即可.

# 矩阵乘法

## LOJ 2325 - 题解

首先暴力表示出状态，一共也就 $w = 165$ 种.
然后可以写出一个转移矩阵，直接矩阵快速幂即可.

但是这样做会 TLE，因为询问太多了.
不过有一个经典的技巧可以加速这一过程.
我们先预处理矩阵的 $2^t$ 次方，然后询问时相当于一堆矩阵乘一个向量. 由于矩阵乘法具有结合性，我们先将最右边的矩阵乘上向量，得到的还是一个向量，继续这一过程即可.
时间复杂度 $O(w^3 \log n + Tw^2 \log n)$.

# 线性基

## BZOJ 3105

一个游戏, 共 $N$ 堆石子, 第 $i$ 堆有 $A_i$ 个.
在第一个回合中, 第一个游戏者可以直接拿走若干个整堆的火柴.
可以一堆都不拿, 但不可以全部拿走.
第二回合也一样, 第二个游戏者也有这样一次机会.
从第三个回合 (又轮到第一个游戏者) 开始, 规则和 Nim 游戏一样.

如果你先拿, 怎样才能保证获胜? 如果可以获胜的话, 还要让第一回合拿的火柴总数尽量小.

$N \leq 100, A_i \leq 10^9$

# 线性基

## BZOJ 3105 - 题解

这题就是要找出 $A$ 这群向量中所有的线性无关子集中和最大的那个.

# 线性基

## BZOJ 3105 - 题解

这题就是要找出 $A$ 这群向量中所有的线性无关子集中和最大的那个.

可以证明一个向量空间所有线性无关子集构成一个拟阵, 于是就可以直接贪心了. 先按 $A_i$ 从大到小排序, 然后如果将当前的算进去之后仍然线性无关就加入.

# 线性基

## BZOJ 3569

$n$ 个点 $m$ 条边无向连通图, $q$ 个询问: 删除指定的 $k$ 条边, 问图是否连通 (询问后立即复原). 强制在线.

$n \leq 10^5, m \leq 5 \times 10^5, 1 \leq k \leq 15$

# 线性基

## BZOJ 3569 - 题解

这是一道套路题.
先 DFS 出一个生成树, 对于非树边随机一个 $[0, 2^{64})$ 的权值, 树边的权值为覆盖它的非树边的权值异或和.

# 线性基

## BZOJ 3569 - 题解

这是一道套路题.
先 DFS 出一个生成树, 对于非树边随机一个 $[0, 2^{64})$ 的权值, 树边的权值为覆盖它的非树边的权值异或和.
然后删掉指定的 $k$ 条边后原图不连通当且仅当这些边的权值存在一个异或和为 $0$ 的子集.

# 线性基

## BZOJ 3569 - 题解

这是一道套路题.
先 DFS 出一个生成树, 对于非树边随机一个 $[0, 2^{64})$ 的权值, 树边的权值为覆盖它的非树边的权值异或和.
然后删掉指定的 $k$ 条边后原图不连通当且仅当这些边的权值存在一个异或和为 $0$ 的子集.

如果不考虑权值冲突, 可以这样证明:

# 线性基

## BZOJ 3569 - 题解

这是一道套路题.
先 DFS 出一个生成树, 对于非树边随机一个 $[0, 2^{64})$ 的权值, 树边的权值为覆盖它的非树边的权值异或和.
然后删掉指定的 $k$ 条边后原图不连通当且仅当这些边的权值存在一个异或和为 $0$ 的子集.

如果不考虑权值冲突, 可以这样证明:

考虑图不连通的时候, 一定存在一种方法将图划分为生成树上连通的两个部分, 满足这两部分之间没有边.

# 线性基

## BZOJ 3569 - 题解

这是一道套路题.
先 DFS 出一个生成树, 对于非树边随机一个 $[0, 2^{64})$ 的权值, 树边的权值为覆盖它的非树边的权值异或和.
然后删掉指定的 $k$ 条边后原图不连通当且仅当这些边的权值存在一个异或和为 $0$ 的子集.

如果不考虑权值冲突, 可以这样证明:

考虑图不连通的时候, 一定存在一种方法将图划分为生成树上连通的两个部分, 满足这两部分之间没有边.
考虑两个端点所在部分不同的边, 这些边的异或和为 $0$.

# 线性基

## BZOJ 3569 - 题解

这是一道套路题.
先 DFS 出一个生成树, 对于非树边随机一个 $[0, 2^{64})$ 的权值, 树边的权值为覆盖它的非树边的权值异或和.
然后删掉指定的 $k$ 条边后原图不连通当且仅当这些边的权值存在一个异或和为 $0$ 的子集.

如果不考虑权值冲突, 可以这样证明:

考虑图不连通的时候, 一定存在一种方法将图划分为生成树上连通的两个部分, 满足这两部分之间没有边.
考虑两个端点所在部分不同的边, 这些边的异或和为 $0$.
充分性类似.

# 线性基

## BZOJ 3569 - 题解

这是一道套路题.
先 DFS 出一个生成树, 对于非树边随机一个 $[0, 2^{64})$ 的权值, 树边的权值为覆盖它的非树边的权值异或和.
然后删掉指定的 $k$ 条边后原图不连通当且仅当这些边的权值存在一个异或和为 $0$ 的子集.

如果不考虑权值冲突, 可以这样证明:

考虑图不连通的时候, 一定存在一种方法将图划分为生成树上连通的两个部分, 满足这两部分之间没有边.
考虑两个端点所在部分不同的边, 这些边的异或和为 $0$.
充分性类似.

粗略分析, 期望考虑 $2^{64}$ 个集合就会出现一个异或和为 $0$ 的, 而这道题一共只会考虑 $q2^k$ 个集合. 所以错误概率可以忽略不计.

# 线性基

## 经典例题 1

给你一个 $n$ 个点的带权无向图. 对于图中的任意一个点集 (可以为空或者全集), 所有恰好有一个端点在这个点集中的边组成的集合被称为割. 一个割的权值被定义为所有在这个割上的边的异或和.
一开始这张图是空图, 现在, 考虑给这张无向图不断的加边. 加入每条边之后, 你都要求出当前权值最大的割的权值, 注意加入的边永远都不会消失.

$n \leq 500, m \leq 1000, 0 \leq \log_2 w < 1000$

# 线性基

## 经典例题 1 - 题解

将边权异或到点上之后发现问题变为：维护线性基，需要支持插入，修改 (这里是异或) 某个向量.

# 线性基

## 经典例题 1 - 题解

将边权异或到点上之后发现问题变为: 维护线性基, 需要支持插入, 修改 (这里是异或) 某个向量.

比较难的部分是如何快速应用修改操作.

# 线性基

## 经典例题 1 - 题解

将边权异或到点上之后发现问题变为：维护线性基，需要支持插入，修改 (这里是异或) 某个向量.

比较难的部分是如何快速应用修改操作.
考虑高斯消元的过程. 可以对于矩阵的每一行记录一个 $from(i)$，表示这一行是由哪些**向量**异或过来的.

# 线性基

## 经典例题 1 - 题解

将边权异或到点上之后发现问题变为：维护线性基，需要支持插入，修改 (这里是异或) 某个向量.

比较难的部分是如何快速应用修改操作.
考虑高斯消元的过程. 可以对于矩阵的每一行记录一个 $from(i)$，表示这一行是由哪些向量异或过来的. 然后每次要修改某一个向量 $x$ (将其变为 $x \oplus y$)，先找到最高位的 $1$ 最低的一个行 $p$ 满足 $x \in from(p)$.

# 线性基

## 经典例题 1 - 题解

将边权异或到点上之后发现问题变为: 维护线性基, 需要支持插入, 修改 (这里是异或) 某个向量.

比较难的部分是如何快速应用修改操作.
考虑高斯消元的过程. 可以对于矩阵的每一行记录一个 $from(i)$, 表示这一行是由哪些向量异或过来的. 然后每次要修改某一个向量 $x$ (将其变为 $x \oplus y$), 先找到最高位的 $1$ 最低的一个行 $p$ 满足 $x \in from(p)$. 对于其他的满足 $x \in from(i)$ 的行 $i$, 用 $p$ 去异或一下, 将向量 $x$ 的贡献消掉.

# 线性基

## 经典例题 1 - 题解

将边权异或到点上之后发现问题变为：维护线性基，需要支持插入，修改 (这里是异或) 某个向量.

比较难的部分是如何快速应用修改操作.
考虑高斯消元的过程. 可以对于矩阵的每一行记录一个 $from(i)$，表示这一行是由哪些向量异或过来的. 然后每次要修改某一个向量 $x$ (将其变为 $x \oplus y$)，先找到最高位的 $1$ 最低的一个行 $p$ 满足 $x \in from(p)$. 对于其他的满足 $x \in from(i)$ 的行 $i$，用 $p$ 去异或一下，将向量 $x$ 的贡献消掉. 最后将行 $p$ 异或上 $y$，然后直接维护线性基即可.

# 线性基

## 经典例题 1 - 题解

将边权异或到点上之后发现问题变为：维护线性基，需要支持插入，修改 (这里是异或) 某个向量.

比较难的部分是如何快速应用修改操作.
考虑高斯消元的过程. 可以对于矩阵的每一行记录一个 $from(i)$，表示这一行是由哪些向量异或过来的. 然后每次要修改某一个向量 $x$ (将其变为 $x \oplus y$)，先找到最高位的 $1$ 最低的一个行 $p$ 满足 $x \in from(p)$. 对于其他的满足 $x \in from(i)$ 的行 $i$，用 $p$ 去异或一下，将向量 $x$ 的贡献消掉. 最后将行 $p$ 异或上 $y$，然后直接维护线性基即可.

时间复杂度 $O(nm(n+l))$. 压位以后可以通过此题.

# Matrix-Tree 定理

对一个无向图 $G$ 定义基尔霍夫矩阵 $Q$, 满足:

$$Q_{ij} = \begin{cases} -adj_{ij}, & i \neq j \\ deg_i, & i = j \end{cases}$$

也就是就是度数矩阵减去邻接矩阵.

# Matrix-Tree 定理

对一个无向图 $G$ 定义基尔霍夫矩阵 $Q$, 满足:

$$Q_{ij} = \begin{cases} -adj_{ij}, & i \neq j \\ deg_i, & i = j \end{cases}$$

也就是就是度数矩阵减去邻接矩阵.

**Matrix-Tree 定理**

无向图 $G$ 的基尔霍夫矩阵 $Q$ 的任意余子式 $M_{ii}$ 等于其生成树个数.

# Matrix-Tree 定理

对一个无向图 $G$ 定义基尔霍夫矩阵 $Q$，满足：

$$Q_{ij} = \begin{cases} -adj_{ij}, & i \neq j \\ deg_i, & i = j \end{cases}$$

也就是就是度数矩阵减去邻接矩阵.

**Matrix-Tree 定理**

无向图 $G$ 的基尔霍夫矩阵 $Q$ 的任意余子式 $M_{ii}$ 等于其生成树个数.

对于有向图的树形图 (父亲连向儿子) 也有类似的结论，只需把度数矩阵换成入度矩阵就可以了.

# Matrix-Tree 定理

对一个无向图 $G$ 定义基尔霍夫矩阵 $Q$, 满足:

$$Q_{ij} = \begin{cases} -adj_{ij}, & i \neq j \\ deg_i, & i = j \end{cases}$$

也就是就是度数矩阵减去邻接矩阵.

**Matrix-Tree 定理**

无向图 $G$ 的基尔霍夫矩阵 $Q$ 的任意余子式 $M_{ii}$ 等于其生成树个数.

对于有向图的树形图 (父亲连向儿子) 也有类似的结论, 只需把度数矩阵换成入度矩阵就可以了.
应用: BEST theorem

# Matrix-Tree 定理

## 经典问题 2

给定一个无向图，求生成树个数对一个任意数 $m$ 取模的结果.

$n \leq 50$

# Matrix-Tree 定理

## 经典问题 2 - 题解

实际上就是任意模数的高斯消元.

# Matrix-Tree 定理

## 经典问题 2 - 题解

实际上就是任意模数的高斯消元.

用类似辗转相除的方式实现即可:

```
for (int j = i + 1; j < n; ++j) {
    while (G[j][i]) {
        std::swap(G[i], G[j]);
        sgn ^= 1;
        int t = G[j][i] / G[i][i];
        for (int k = i; k < n; ++k) {
            G[j][k] = (G[j][k] - (LL)G[i][k] * t % P + P) % P;
        }
    }
}
```

# Matrix-Tree 定理

## 经典问题 3

给定一个无向图，图上每条边是红色或蓝色，求恰有 $k$ 条红边的生成树个数.

$n \leq 50$

# Matrix-Tree 定理

## 经典问题 3 - 题解

将红边视为 $x$, 蓝边视为 $1$, 那么求出来的行列式的 $x^k$ 的系数即是答案.

# Matrix-Tree 定理

## 经典问题 3 - 题解

将红边视为 $x$, 蓝边视为 $1$, 那么求出来的行列式的 $x^k$ 的系数即是答案.
插值即可, $O(n^4)$.

# Matrix-Tree 定理

## Code+ 第二次网络赛 div1 C

给定 $n$ 行 $m$ 列的网格, 每个格子中有上下左右四个方向之一的箭头或 "." 符号 (整个网格中共有 $k$ 个).
在每个标有 "." 的格子中填入一个箭头, 使得从任何一个格子出发沿着箭头所指的方向行走, 都不会进入循环, 求填入的方案数.

$n, m \leq 200, k \leq 300$

# Matrix-Tree 定理

## Code+ 第二次网络赛 div1 C - 题解

通过下列方式建图:

# Matrix-Tree 定理

## Code+ 第二次网络赛 div1 C - 题解

通过下列方式建图:

- 将每个格子视作一个节点，并增加一个节点对应整个网格的外部区域

# Matrix-Tree 定理

## Code+ 第二次网络赛 div1 C - 题解

通过下列方式建图:

- 将每个格子视作一个节点，并增加一个节点对应整个网格的外部区域
- 对于每一个标有箭头的格子，建立一条由其对应的节点连向其中箭头指向的目标所对应节点的有向边

# Matrix-Tree 定理

## Code+ 第二次网络赛 div1 C - 题解

通过下列方式建图:

- 将每个格子视作一个节点，并增加一个节点对应整个网格的外部区域
- 对于每一个标有箭头的格子，建立一条由其对应的节点连向其中箭头指向的目标所对应节点的有向边
- 对于每一个标有 “.” 的格子，建立四条由其对应的节点分别连向四个方向邻居所对应节点的有向边 (保留重边)

# Matrix-Tree 定理

## Code+ 第二次网络赛 div1 C - 题解

通过下列方式建图:

- 将每个格子视作一个节点，并增加一个节点对应整个网格的外部区域
- 对于每一个标有箭头的格子，建立一条由其对应的节点连向其中箭头指向的目标所对应节点的有向边
- 对于每一个标有 “.” 的格子，建立四条由其对应的节点分别连向四个方向邻居所对应节点的有向边 (保留重边)

然后问题就转化为一个反向树形图计数问题，直接做会 TLE.

# Matrix-Tree 定理

## Code+ 第二次网络赛 div1 C - 题解

通过下列方式建图:

- 将每个格子视作一个节点，并增加一个节点对应整个网格的外部区域
- 对于每一个标有箭头的格子，建立一条由其对应的节点连向其中箭头指向的目标所对应节点的有向边
- 对于每一个标有 “.” 的格子，建立四条由其对应的节点分别连向四个方向邻居所对应节点的有向边 (保留重边)

然后问题就转化为一个反向树形图计数问题，直接做会 TLE.
可以先将确定了的点缩起来，时间复杂度变为 $O(nm + k^3)$.

# Lindström–Gessel–Viennot 引理

对于一个 DAG $G$, 起点集合 $A = \{a_1, ..., a_m\}$, 终点集合 $B = \{b_1, ..., b_m\}$, 定义矩阵:

$$M = \begin{pmatrix} e(a_1, b_1) & e(a_1, b_2) & \cdots & e(a_1, b_m) \\ e(a_2, b_1) & e(a_2, b_2) & \cdots & e(a_2, b_m) \\ \vdots & \vdots & \ddots & \vdots \\ e(a_m, b_1) & e(a_m, b_2) & \cdots & e(a_m, b_m) \end{pmatrix}$$

其中 $e(u, v)$ 表示 $u$ 到 $v$ 的路径方案数.

# Lindström–Gessel–Viennot 引理

对于一个 DAG $G$, 起点集合 $A = \{a_1, ..., a_m\}$, 终点集合 $B = \{b_1, ..., b_m\}$, 定义矩阵:

$$M = \begin{pmatrix} e(a_1, b_1) & e(a_1, b_2) & \cdots & e(a_1, b_m) \\ e(a_2, b_1) & e(a_2, b_2) & \cdots & e(a_2, b_m) \\ \vdots & \vdots & \ddots & \vdots \\ e(a_m, b_1) & e(a_m, b_2) & \cdots & e(a_m, b_m) \end{pmatrix}$$

其中 $e(u, v)$ 表示 $u$ 到 $v$ 的路径方案数.

定义一种合法的方案为将 $A$ 和 $B$ 完美匹配, 然后对于每一对匹配在 $G$ 中对应一条路径, 这些路径**两两不相交**.

# Lindström–Gessel–Viennot 引理

对于一个 DAG $G$, 起点集合 $A = \{a_1, ..., a_m\}$, 终点集合 $B = \{b_1, ..., b_m\}$, 定义矩阵:

$$M = \begin{pmatrix} e(a_1, b_1) & e(a_1, b_2) & \cdots & e(a_1, b_m) \\ e(a_2, b_1) & e(a_2, b_2) & \cdots & e(a_2, b_m) \\ \vdots & \vdots & \ddots & \vdots \\ e(a_m, b_1) & e(a_m, b_2) & \cdots & e(a_m, b_m) \end{pmatrix}$$

其中 $e(u, v)$ 表示 $u$ 到 $v$ 的路径方案数.

定义一种合法的方案为将 $A$ 和 $B$ 完美匹配, 然后对于每一对匹配在 $G$ 中对应一条路径, 这些路径**两两不相交**.
可以证明:

$$\det(M) = \sum_{(P_1,...,P_n):\ A \to B} \operatorname{sign}(\sigma(P))$$

其中 $\sigma(P)$ 表示合法路径 $P$ 的排列的逆序对数.

# Lindström–Gessel–Viennot 引理

## 经典例题 4

一个 $n \times n$ 的网格，指定了网格第一行的 $k$ 个点和最后一行的 $k$ 个点.
问用不相交的路径把这些点两两连起来的方案数，路径只能向右或向下.

$n \leq 10^5, k \leq 300$

# Lindström–Gessel–Viennot 引理

## 经典例题 4 - 题解

网格图上要求路径两两不相交，只可能是从左到右依次匹配.

# Lindström–Gessel–Viennot 引理

## 经典例题 4 - 题解

网格图上要求路径两两不相交，只可能是从左到右依次匹配.
于是利用组合数构造出刚才的那个矩阵 $M$.

# Lindström–Gessel–Viennot 引理

## 经典例题 4 - 题解

网格图上要求路径两两不相交，只可能是从左到右依次匹配.
于是利用组合数构造出刚才的那个矩阵 $M$.
因为只有一种可能的排列，根据刚才那个引理，$\det M$ 就等于这个方案数.

# 伴随矩阵

- 矩阵 $A$ 关于第 $i$ 行和第 $j$ 列的余子式 $M_{ij} = \det A_{ij}$

# 伴随矩阵

- 矩阵 $A$ 关于第 $i$ 行和第 $j$ 列的余子式 $M_{ij} = \det A_{ij}$
- 矩阵 $A$ 关于第 $i$ 行和第 $j$ 列的代数余子式 $C_{ij} = (-1)^{i+j} M_{ij}$

# 伴随矩阵

- 矩阵 $A$ 关于第 $i$ 行和第 $j$ 列的余子式 $M_{ij} = \det A_{ij}$
- 矩阵 $A$ 关于第 $i$ 行和第 $j$ 列的代数余子式 $C_{ij} = (-1)^{i+j} M_{ij}$
- 矩阵 $A$ 的伴随矩阵 $\operatorname{adj} A$ 是 $A$ 的余子矩阵的转置矩阵，即 $\operatorname{adj} A = C^T$

# 伴随矩阵

- 矩阵 $A$ 关于第 $i$ 行和第 $j$ 列的余子式 $M_{ij} = \det A_{ij}$
- 矩阵 $A$ 关于第 $i$ 行和第 $j$ 列的代数余子式 $C_{ij} = (-1)^{i+j} M_{ij}$
- 矩阵 $A$ 的伴随矩阵 $\operatorname{adj} A$ 是 $A$ 的余子矩阵的转置矩阵，即 $\operatorname{adj} A = C^T$

对于可逆矩阵 $A$，有如下式子成立：

$$A^{-1} = \frac{\operatorname{adj} A}{\det A}$$

# 伴随矩阵

- 矩阵 $A$ 关于第 $i$ 行和第 $j$ 列的余子式 $M_{ij} = \det A_{ij}$
- 矩阵 $A$ 关于第 $i$ 行和第 $j$ 列的代数余子式 $C_{ij} = (-1)^{i+j} M_{ij}$
- 矩阵 $A$ 的伴随矩阵 $\operatorname{adj} A$ 是 $A$ 的余子矩阵的转置矩阵，即 $\operatorname{adj} A = C^T$

对于可逆矩阵 $A$，有如下式子成立：

$$A^{-1} = \frac{\operatorname{adj} A}{\det A}$$

**证明**

考虑 $A \times \operatorname{adj} A$.

# 伴随矩阵

- 矩阵 $A$ 关于第 $i$ 行和第 $j$ 列的余子式 $M_{ij} = \det A_{ij}$
- 矩阵 $A$ 关于第 $i$ 行和第 $j$ 列的代数余子式 $C_{ij} = (-1)^{i+j} M_{ij}$
- 矩阵 $A$ 的伴随矩阵 $\operatorname{adj} A$ 是 $A$ 的余子矩阵的转置矩阵，即 $\operatorname{adj} A = C^T$

对于可逆矩阵 $A$，有如下式子成立：

$$A^{-1} = \frac{\operatorname{adj} A}{\det A}$$

**证明**

考虑 $A \times \operatorname{adj} A$.
其第 $i$ 行第 $i$ 列的系数为 $\sum_{j=1}^{n} A_{ij} C_{ij}$. 根据拉普拉斯展开，显然这个系数等于 $\det A$.

# 伴随矩阵

- 矩阵 $A$ 关于第 $i$ 行和第 $j$ 列的余子式 $M_{ij} = \det A_{ij}$
- 矩阵 $A$ 关于第 $i$ 行和第 $j$ 列的代数余子式 $C_{ij} = (-1)^{i+j} M_{ij}$
- 矩阵 $A$ 的伴随矩阵 $\operatorname{adj} A$ 是 $A$ 的余子矩阵的转置矩阵，即 $\operatorname{adj} A = C^T$

对于可逆矩阵 $A$，有如下式子成立：

$$A^{-1} = \frac{\operatorname{adj} A}{\det A}$$

**证明**

考虑 $A \times \operatorname{adj} A$.
其第 $i$ 行第 $i$ 列的系数为 $\sum_{j=1}^{n} A_{ij} C_{ij}$. 根据拉普拉斯展开，显然这个系数等于 $\det A$.
当 $i \neq j$ 时，第 $i$ 行第 $j$ 列的系数为 $\sum_{k=1}^{n} A_{ik} C_{jk}$，相当于用 $A$ 的第 $i$ 行替换第 $j$ 行后求行列式，显然等于 $0$.

# 伴随矩阵

- 矩阵 $A$ 关于第 $i$ 行和第 $j$ 列的余子式 $M_{ij} = \det A_{ij}$
- 矩阵 $A$ 关于第 $i$ 行和第 $j$ 列的代数余子式 $C_{ij} = (-1)^{i+j} M_{ij}$
- 矩阵 $A$ 的伴随矩阵 $\operatorname{adj} A$ 是 $A$ 的余子矩阵的转置矩阵，即 $\operatorname{adj} A = C^T$

对于可逆矩阵 $A$，有如下式子成立：

$$A^{-1} = \frac{\operatorname{adj} A}{\det A}$$

**证明**

考虑 $A \times \operatorname{adj} A$.
其第 $i$ 行第 $i$ 列的系数为 $\sum_{j=1}^{n} A_{ij} C_{ij}$. 根据拉普拉斯展开，显然这个系数等于 $\det A$.
当 $i \neq j$ 时，第 $i$ 行第 $j$ 列的系数为 $\sum_{k=1}^{n} A_{ik} C_{jk}$，相当于用 $A$ 的第 $i$ 行替换第 $j$ 行后求行列式，显然等于 $0$.
于是我们有 $A \times \operatorname{adj} A = \det A \times I_n$ 即 $A^{-1} = \frac{\operatorname{adj} A}{\det A}$.

# 积和式

和行列式的定义很类似:

$$\operatorname{perm}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)}$$

一个二分图的完美匹配个数等于积和式的值. 但是积和式的计算是 #P-完全的, 目前没有高效的算法可以计算.

# 积和式

和行列式的定义很类似:

$$\operatorname{perm}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)}$$

一个二分图的完美匹配个数等于积和式的值. 但是积和式的计算是 #P-完全的, 目前没有高效的算法可以计算.

其实 $\bmod 2^k$ 意义下的积和式都有多项式算法, 详见 https://cstheory.stackexchange.com/a/37576/48037.

# 积和式

## CF 736D

你要生成一个长度为 $n$ 的排列，有 $m$ 个可行数对，每个可行数对形如 “$b_i$ 这个数可以放在第 $a_i$ 个位置上”.
保证通过这 $m$ 个可行数对能够生成出来的排列的数量是奇数.
对于每个可行数对，你想知道如果把这个可行数对删掉，那么能够生成的排列的数量是否还是奇数.

$n \leq 2000, n \leq m \leq \min(n^2, 500000)$

# 积和式

## CF 736D - 题解

这道题就是要求一个二分图删去每条边后完美匹配的个数的奇偶性，转化为求积和式.

# 积和式

## CF 736D - 题解

这道题就是要求一个二分图删去每条边后完美匹配的个数的奇偶性，转化为求积和式.

注意到 $\bmod 2$ 意义下，$\mathrm{perm}(A) \equiv \det(A)$，于是转化为求行列式.

# 积和式

## CF 736D - 题解

这道题就是要求一个二分图删去每条边后完美匹配的个数的奇偶性，转化为求积和式.

注意到 $\bmod 2$ 意义下，$\mathrm{perm}(A) \equiv \det(A)$，于是转化为求行列式.
当我们把矩阵的一个 $1$ 变为 $0$ 时，可以发现它的行列式奇偶性的变化等于这个位置的代数余子式.

# 积和式

## CF 736D - 题解

这道题就是要求一个二分图删去每条边后完美匹配的个数的奇偶性，转化为求积和式.

注意到 $\bmod 2$ 意义下，$\operatorname{perm}(A) \equiv \det(A)$，于是转化为求行列式.
当我们把矩阵的一个 $1$ 变为 $0$ 时，可以发现它的行列式奇偶性的变化等于这个位置的代数余子式.
所以我们只需要求出伴随矩阵.

# 积和式

## CF 736D - 题解

这道题就是要求一个二分图删去每条边后完美匹配的个数的奇偶性，转化为求积和式.

注意到 $\bmod 2$ 意义下，$\operatorname{perm}(A) \equiv \det(A)$，于是转化为求行列式.
当我们把矩阵的一个 $1$ 变为 $0$ 时，可以发现它的行列式奇偶性的变化等于这个位置的代数余子式.
所以我们只需要求出伴随矩阵.

由于保证了原矩阵的行列式为奇数，所以可以转化为求逆矩阵.

# 积和式

## CF 736D - 题解

这道题就是要求一个二分图删去每条边后完美匹配的个数的奇偶性，转化为求积和式.

注意到 $\bmod 2$ 意义下，$\operatorname{perm}(A) \equiv \det(A)$，于是转化为求行列式.
当我们把矩阵的一个 $1$ 变为 $0$ 时，可以发现它的行列式奇偶性的变化等于这个位置的代数余子式.
所以我们只需要求出伴随矩阵.

由于保证了原矩阵的行列式为奇数，所以可以转化为求逆矩阵.
压位高斯消元即可.