线性代数

梦熊人才联盟

2024.9

向量

- 向量是什么？

向量

- 向量是什么？
- 高中主要介绍了平面向量和空间向量，其意义是平面或者空间中具有大小和方向的量。

向量

- 向量是什么？
- 高中主要介绍了平面向量和空间向量，其意义是平面或者空间中具有大小和方向的量。
- 例如空间向量 $(1,2,3)$ 就表示空间中从原点 $(0,0,0)$ 指向 $(1,2,3)$ 的一条有向线段，$1,2,3$ 分别代表这个向量在 x,y,z 三个维度上的偏移量。

向量

- 向量是什么？
- 高中主要介绍了平面向量和空间向量，其意义是平面或者空间中具有大小和方向的量。
- 例如空间向量 $(1,2,3)$ 就表示空间中从原点 $(0,0,0)$ 指向 $(1,2,3)$ 的一条有向线段，$1,2,3$ 分别代表这个向量在 x,y,z 三个维度上的偏移量。
- 而如果我们拓展到 n 维，就可以用 n 个数字来代表每一维的偏移量，例如

$$\begin{bmatrix} 1 & -2 & 3 & -1 & 4.5 & 0 \end{bmatrix}$$

向量

- 向量是什么？
- 高中主要介绍了平面向量和空间向量，其意义是平面或者空间中具有大小和方向的量。
- 例如空间向量 $(1,2,3)$ 就表示空间中从原点 $(0,0,0)$ 指向 $(1,2,3)$ 的一条有向线段，$1,2,3$ 分别代表这个向量在 x,y,z 三个维度上的偏移量。
- 而如果我们拓展到 n 维，就可以用 n 个数字来代表每一维的偏移量，例如

$$\begin{bmatrix} 1 & -2 & 3 & -1 & 4.5 & 0 \end{bmatrix}$$

- 也可以写成一列

$$\begin{bmatrix} 1 \\ -2 \\ 3 \end{bmatrix}$$

向量

- 向量是什么？
- 高中主要介绍了平面向量和空间向量，其意义是平面或者空间中具有大小和方向的量。
- 例如空间向量 $(1,2,3)$ 就表示空间中从原点 $(0,0,0)$ 指向 $(1,2,3)$ 的一条有向线段，$1,2,3$ 分别代表这个向量在 x,y,z 三个维度上的偏移量。
- 而如果我们拓展到 n 维，就可以用 n 个数字来代表每一维的偏移量，例如

$$\begin{bmatrix} 1 & -2 & 3 & -1 & 4.5 & 0 \end{bmatrix}$$

- 也可以写成一列

$$\begin{bmatrix} 1 \\ -2 \\ 3 \end{bmatrix}$$

- 我们分别称为行向量与列向量。

向量运算

- 对于维数相同的两个向量 $\mathbf{a}, \mathbf{b}$，我们将他们每一维的值加起来的向量记作 $\mathbf{a}+\mathbf{b}$。
  例如

$$\begin{bmatrix}1 & -2 & 3\end{bmatrix}+\begin{bmatrix}-4 & 0 & 7\end{bmatrix}=\begin{bmatrix}-3 & -2 & 10\end{bmatrix}$$

  同样的可以定义向量的减法。

向量运算

- 对于维数相同的两个向量 $\mathbf{a}, \mathbf{b}$，我们将他们每一维的值加起来的向量记作 $\mathbf{a}+\mathbf{b}$。
  例如

$$\begin{bmatrix}1 & -2 & 3\end{bmatrix}+\begin{bmatrix}-4 & 0 & 7\end{bmatrix}=\begin{bmatrix}-3 & -2 & 10\end{bmatrix}$$

  同样的可以定义向量的减法。
- 对于一个向量 $\mathbf{a}$ 以及一个实数 λ，定义 $\lambda\mathbf{a}$ 为向量的数乘运算，其结果为每一维的值乘上 λ。
  例如

$$3\times\begin{bmatrix}1 & -2 & 3\end{bmatrix}=\begin{bmatrix}3 & -6 & 9\end{bmatrix}$$

向量运算

- 容易验证向量加法具有结合律，交换律。

向量运算

- 容易验证向量加法具有结合律，交换律。
- 对于数乘，有一下性质：

$$\mu(\lambda\mathbf{a}) = (\mu\lambda\mathbf{a}) \tag{1}$$
$$(\mu+\lambda)\mathbf{a} = \mu\mathbf{a} + \lambda\mathbf{a} \tag{2}$$
$$\lambda(\mathbf{a}+\mathbf{b}) = \lambda\mathbf{a} + \lambda\mathbf{b} \tag{3}$$

内积

- 对于两个 n 维向量 $\mathbf{a},\mathbf{b}$，定义其内积运算 $\mathbf{a}\cdot\mathbf{b}=\sum_{i=1}^{n}\mathbf{a}_i\mathbf{b}_i$

  也称为点积，数量积。

内积

- 对于两个 n 维向量 $\mathbf{a}, \mathbf{b}$，定义其内积运算 $\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} \mathbf{a}_i \mathbf{b}_i$ 也称为点积，数量积。
- 内积的结果是一个标量，并且满足交换律，且也具有线性性：

$$(\lambda \mathbf{a}) \cdot \mathbf{b} = \lambda(\mathbf{a} \cdot \mathbf{b}) \tag{4}$$

$$(\mathbf{a} + \mathbf{b}) \cdot \mathbf{c} = \mathbf{ac} + \mathbf{b} \cdot \mathbf{c} \tag{5}$$

内积

- 对于两个 n 维向量 $\mathbf{a}, \mathbf{b}$，定义其内积运算 $\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} \mathbf{a}_i \mathbf{b}_i$

  也称为点积，数量积。
- 内积的结果是一个标量，并且满足交换律，且也具有线性性：

$$(\lambda \mathbf{a}) \cdot \mathbf{b} = \lambda(\mathbf{a} \cdot \mathbf{b}) \tag{4}$$

$$(\mathbf{a} + \mathbf{b}) \cdot \mathbf{c} = \mathbf{ac} + \mathbf{b} \cdot \mathbf{c} \tag{5}$$

- 内积在几何上可以用来判断向量垂直，求出向量夹角等。

矩阵

- 矩阵可以看成是若干个行向量或者若干个列向量放在一起形成的一个二维数组状物，如

$$\begin{bmatrix} 1 & -2 & 3 & 0.5 \\ 2 & 8 & 1 & 4 \\ 0 & 9 & 1 & 0 \end{bmatrix}$$

这就是一个 3×4 的矩阵。

矩阵

- 矩阵可以看成是若干个行向量或者若干个列向量放在一起形成的一个二维数组状物，如

$$\begin{bmatrix} 1 & -2 & 3 & 0.5 \\ 2 & 8 & 1 & 4 \\ 0 & 9 & 1 & 0 \end{bmatrix}$$

这就是一个 3×4 的矩阵。

- 行数列数相等的矩阵称为方阵，有时也用"n 阶矩阵"这样的称呼来表示一个 $n \times n$ 的方阵。

矩阵

- 矩阵可以看成是若干个行向量或者若干个列向量放在一起形成的一个二维数组状物，如

$$\begin{bmatrix} 1 & -2 & 3 & 0.5 \\ 2 & 8 & 1 & 4 \\ 0 & 9 & 1 & 0 \end{bmatrix}$$

这就是一个 3×4 的矩阵。

- 行数列数相等的矩阵称为方阵，有时也用"n 阶矩阵"这样的称呼来表示一个 $n \times n$ 的方阵。
- 将矩阵 $\mathbf{A}$ 的行列交换得到的矩阵称作 $\mathbf{A}$ 的转置，记作 $\mathbf{A}^T$，例如

$$\begin{bmatrix} 1 & -2 & 3 & 0.5 \\ 2 & 8 & 1 & 4 \\ 0 & 9 & 1 & 0 \end{bmatrix}^T = \begin{bmatrix} 1 & 2 & 0 \\ -2 & 8 & 9 \\ 3 & 1 & 1 \\ 0.5 & 5 & 0 \end{bmatrix}$$

矩阵运算

- 矩阵的加减法，数乘操作与向量是类似的，可自行类比一下，运算的性质也是类似的。

矩阵运算

- 矩阵的加减法，数乘操作与向量是类似的，可自行类比一下，运算的性质也是类似的。
- 矩阵的乘法操作可以看成是内积的扩展，一个 $n \times p$ 的矩阵 $\mathbf{A}$ 与一个 $p \times m$ 的矩阵 $\mathbf{B}$ 做矩阵乘法的结果是一个 $n \times m$ 的矩阵，记作 $\mathbf{A} \times \mathbf{B}$，其第 i 行第 j 列的值是 $\mathbf{A}$ 的第 i 行与 $\mathbf{B}$ 的第 j 列两个向量的内积，即

$$(\mathbf{A} \times \mathbf{B})_{i,j} = \sum_{k=1}^{p} \mathbf{A}_{i,p} \mathbf{B}_{p,j}$$

矩阵运算

- 矩阵的加减法，数乘操作与向量是类似的，可自行类比一下，运算的性质也是类似的。
- 矩阵的乘法操作可以看成是内积的扩展，一个 $n \times p$ 的矩阵 $\mathbf{A}$ 与一个 $p \times m$ 的矩阵 $\mathbf{B}$ 做矩阵乘法的结果是一个 $n \times m$ 的矩阵，记作 $\mathbf{A} \times \mathbf{B}$，其第 i 行第 j 列的值是 $\mathbf{A}$ 的第 i 行与 $\mathbf{B}$ 的第 j 列两个向量的内积，即

$$(\mathbf{A} \times \mathbf{B})_{i,j} = \sum_{k=1}^{p} \mathbf{A}_{i,p} \mathbf{B}_{p,j}$$

- 矩阵乘法具有结合律，一般不满足交换律，从定义可看出对于两个 $O(n)$ 阶矩阵求矩阵乘法的时间复杂度是 $O(n^3)$。

矩阵运算

- 矩阵的加减法，数乘操作与向量是类似的，可自行类比一下，运算的性质也是类似的。
- 矩阵的乘法操作可以看成是内积的扩展，一个 $n \times p$ 的矩阵 $\mathbf{A}$ 与一个 $p \times m$ 的矩阵 $\mathbf{B}$ 做矩阵乘法的结果是一个 $n \times m$ 的矩阵，记作 $\mathbf{A} \times \mathbf{B}$，其第 i 行第 j 列的值是 $\mathbf{A}$ 的第 i 行与 $\mathbf{B}$ 的第 j 列两个向量的内积，即

$$(\mathbf{A} \times \mathbf{B})_{i,j} = \sum_{k=1}^{p} \mathbf{A}_{i,p} \mathbf{B}_{p,j}$$

- 矩阵乘法具有结合律，一般不满足交换律，从定义可看出对于两个 $O(n)$ 阶矩阵求矩阵乘法的时间复杂度是 $O(n^3)$。
- 类比普通的幂运算，可以用 $\mathbf{A}^k$ 来代表 k 个矩阵 $\mathbf{A}$ 相乘的结果。

[Cnoi2021] 矩阵

- 给定两个长度为 n 的序列 $\{a_n\}$，$\{b_n\}$ 与一个整数 k。
  设矩阵 A 满足 $A_{ij} = a_i \times b_j$，求 A^k 的所有元素的和在模 998244353 意义下的结果。
  对于 100% 的数据保证 $1 \le n \le 10^5$，$0 \le k < 998244353$，$|a_i|, |b_i| \le 10^9$。

[GDKOI2023 提高组] 矩阵

- 多次给定三个 $n \times n$ 的矩阵 A, B, C，你需要判断 $A \times B$ 在模 998244353 意义下是否等于 C。其中 $\times$ 为矩阵乘法，$C_{i,j} = \sum_{k=1}^{n} A_{i,k} B_{k,j}$。
  对于 20% 的数据，满足 $\sum n \leq 300$。
  对于另外 20% 的数据，满足 $A_{i,j} \neq 0$ 的位置不超过 n 个。
  对于 100% 的数据，满足 $1 \leq T, n \leq 3000, \sum n \leq 3000, 0 \leq A_{i,j}, B_{i,j}, C_{i,j} < 998244353$。

矩阵快速幂

- 现在要计算矩阵 A（通常是方阵）的 k 次幂，暴力乘 k 次的复杂度是巨大的，我们可以类比快速幂的做法优化，时间复杂度就优化成了 $O(n^3 \log k)$。

矩阵快速幂

- 现在要计算矩阵 A（通常是方阵）的 k 次幂，暴力乘 k 次的复杂度是巨大的，我们可以类比快速幂的做法优化，时间复杂度就优化成了 $O(n^3 \log k)$。
- 在一些动态规划问题中，可以把 dp 的状态看成是一个向量，转移看成是乘上一个相同矩阵，那么此就时就可以用矩阵快速幂加速这个矩阵乘法的过程。

矩阵快速幂

- 现在要计算矩阵 A（通常是方阵）的 k 次幂，暴力乘 k 次的复杂度是巨大的，我们可以类比快速幂的做法优化，时间复杂度就优化成了 $O(n^3 \log k)$。
- 在一些动态规划问题中，可以把 dp 的状态看成是一个向量，转移看成是乘上一个相同矩阵，那么此就时就可以用矩阵快速幂加速这个矩阵乘法的过程。
- 例如，求解斐波那契数列 $F_n = F_{n-1} + F_{n-2}$ 的第 n 项时，我们可以写出下面的式子：

$$\begin{bmatrix} F_{n-2} & F_{n-1} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} F_{n-1} & F_n \end{bmatrix}$$

矩阵快速幂

- 现在要计算矩阵 A（通常是方阵）的 k 次幂，暴力乘 k 次的复杂度是巨大的，我们可以类比快速幂的做法优化，时间复杂度就优化成了 $O(n^3 \log k)$。
- 在一些动态规划问题中，可以把 dp 的状态看成是一个向量，转移看成是乘上一个相同矩阵，那么此就时就可以用矩阵快速幂加速这个矩阵乘法的过程。
- 例如，求解斐波那契数列 $F_n = F_{n-1} + F_{n-2}$ 的第 n 项时，我们可以写出下面的式子：

$$\begin{bmatrix} F_{n-2} & F_{n-1} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} F_{n-1} & F_n \end{bmatrix}$$

- 这样，记 $T = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$，我们每乘上一个 T 就可以从 F_{n-1} 转移到 F_n，那么我们最终的答案向量可以表示为 $\begin{bmatrix} 1 & 1 \end{bmatrix} T^{n-2}$，用矩阵快速幂加速即可在 $O(\log n)$ 的时间内求出斐波那契数量第 n 项了。

[HNOI2008] GT 考试

- 阿申准备报名参加 GT 考试，准考证号为 N 位数 $X_1, X_2 ... X_n$ $(0 \leq X_i \leq 9)$，他不希望准考证号上出现不吉利的数字。他的不吉利数字 $A_1, A_2, \cdots, A_m$ $(0 \leq A_i \leq 9)$ 有 M 位，不出现是指 $X_1, X_2 \cdots X_n$ 中没有一段恰好等于 $A_1, A_2, \cdots, A_m$，A_1 和 X_1 可以为 0。
  对于全部数据，$N \leq 10^9$，$M \leq 20$，$K \leq 10000$。

[ARC176D] Swap Permutation

- 给定 n 阶排列 $p_1 \sim p_n$，m 次操作选择 $1 \le i < j \le n$ 并交换 p_i, p_j。
  求最终每个排列的 $\sum_{i=1}^{n-1} |p_i - p_{i+1}|$ 之和。
  数据范围：$n, m \le 2 \times 10^5$。

线性方程组的解

- 我们是如何求解二元一次方程组的？

$$\begin{cases} 3x + 4y &= 11 \\ x - 2y &= -3 \end{cases}$$

线性方程组的解

- 我们是如何求解二元一次方程组的？

$$\begin{cases} 3x + 4y &= 11 \\ x - 2y &= -3 \end{cases}$$

- 加减消元法，把第二个方程 $\times 3$ 得到

$$\begin{cases} 3x + 4y &= 11 \\ 3x - 6y &= -9 \end{cases}$$

线性方程组的解

- 我们是如何求解二元一次方程组的？

$$\begin{cases} 3x + 4y &= 11 \\ x - 2y &= -3 \end{cases}$$

- 加减消元法，把第二个方程 $\times 3$ 得到

$$\begin{cases} 3x + 4y &= 11 \\ 3x - 6y &= -9 \end{cases}$$

- 用第二个方程减掉第一个得到

$$\begin{cases} 3x + 4y &= 11 \\ -10y &= -20 \end{cases}$$

线性方程组的解

- 我们是如何求解二元一次方程组的？

$$\begin{cases} 3x + 4y &= 11 \\ x - 2y &= -3 \end{cases}$$

- 加减消元法，把第二个方程 $\times 3$ 得到

$$\begin{cases} 3x + 4y &= 11 \\ 3x - 6y &= -9 \end{cases}$$

- 用第二个方程减掉第一个得到

$$\begin{cases} 3x + 4y &= 11 \\ -10y &= -20 \end{cases}$$

- 求出 y 之后带入第一个即可。

线性方程组的解

- 求解线性方程组时具有的性质：

线性方程组的解

- 求解线性方程组时具有的性质：
  - 交换两个方程，解不变

线性方程组的解

- 求解线性方程组时具有的性质：
  - 交换两个方程，解不变
  - 把一行整体乘上一个非零数，解不变

线性方程组的解

- 求解线性方程组时具有的性质：
  - 交换两个方程，解不变
  - 把一行整体乘上一个非零数，解不变
  - 把一行乘上一个数再加到另一行，解不变

线性方程组的解

- 求解线性方程组时具有的性质：
  - 交换两个方程，解不变
  - 把一行整体乘上一个非零数，解不变
  - 把一行乘上一个数再加到另一行，解不变
- 我们称这三条为初等行变换。

线性方程组的解

- 求解线性方程组时具有的性质：
  - 交换两个方程，解不变
  - 把一行整体乘上一个非零数，解不变
  - 把一行乘上一个数再加到另一行，解不变
- 我们称这三条为初等行变换。
- 方便起见我们可以把方程中每个变量的系数写成一个矩阵 $A_{i,j}$，然后每个方程右边的数写成一个列向量 b，例如：

$$A = \begin{bmatrix} 3 & -1 & 1 \\ 1 & -2 & 4 \\ 2 & 3 & -1 \end{bmatrix}, b = \begin{bmatrix} 6 \\ 11 \\ -3 \end{bmatrix}$$

就表示了方程组：

$$\begin{cases} 3x_1 - x_2 + x_3 = 6 \\ x_1 - 2x_2 + 4x_3 = 11 \\ 2x_1 + 3x_2 - x_3 = -3 \end{cases}$$

高斯消元

- 高斯消元法的核心思路就是利用初等行变换对系数矩阵进行操作，使得矩阵变成上三角矩阵或者对角矩阵：

高斯消元

- 高斯消元法的核心思路就是利用初等行变换对系数矩阵进行操作，使得矩阵变成上三角矩阵或者对角矩阵：
  - 对角矩阵：只有主对角线上的系数不为零，可直接求出每个变量的值

高斯消元

- 高斯消元法的核心思路就是利用初等行变换对系数矩阵进行操作，使得矩阵变成上三角矩阵或者对角矩阵：
  - 对角矩阵：只有主对角线上的系数不为零，可直接求出每个变量的值
  - 上三角矩阵：只有主对角线以及主对角线右上测的位置的系数非零，此时第 n 行求出变量的值后代回上一回行即可。

高斯消元

- 高斯消元法的核心思路就是利用初等行变换对系数矩阵进行操作，使得矩阵变成上三角矩阵或者对角矩阵：
  - 对角矩阵：只有主对角线上的系数不为零，可直接求出每个变量的值
  - 上三角矩阵：只有主对角线以及主对角线右上测的位置的系数非零，此时第 n 行求出变量的值后代回上一回行即可。
- 高斯消元的过程如下，依次考虑 $i = 1, 2, \ldots n$，假设前 $i-1$ 列已经消元完成满足只有对角线上有值：

高斯消元

- 高斯消元法的核心思路就是利用初等行变换对系数矩阵进行操作，使得矩阵变成上三角矩阵或者对角矩阵：
  - 对角矩阵：只有主对角线上的系数不为零，可直接求出每个变量的值
  - 上三角矩阵：只有主对角线以及主对角线右上测的位置的系数非零，此时第 n 行求出变量的值后代回上一回行即可。
- 高斯消元的过程如下，依次考虑 $i = 1, 2, \ldots n$，假设前 $i-1$ 列已经消元完成满足只有对角线上有值：
  - 1. 现在第 i 行前 $i-1$ 个变量的系数都是 0，如果第 i 个变量的系数也是 0，就找到 $j > i$ 的任意一个满足第 j 行第 i 个变量系数非零的 j，交换第 i 行和第 j 行。

高斯消元

- 高斯消元法的核心思路就是利用初等行变换对系数矩阵进行操作，使得矩阵变成上三角矩阵或者对角矩阵：
  - 对角矩阵：只有主对角线上的系数不为零，可直接求出每个变量的值
  - 上三角矩阵：只有主对角线以及主对角线右上测的位置的系数非零，此时第 n 行求出变量的值后代回上一回行即可。
- 高斯消元的过程如下，依次考虑 $i = 1, 2, \ldots n$，假设前 $i-1$ 列已经消元完成满足只有对角线上有值：
  - 1. 现在第 i 行前 $i-1$ 个变量的系数都是 0，如果第 i 个变量的系数也是 0，就找到 $j > i$ 的任意一个满足第 j 行第 i 个变量系数非零的 j，交换第 i 行和第 j 行。
  - 2. 现在假设 $A_{i,i} \neq 0$，我们用第三条变换去消掉其余行的第 i 个变量的系数。枚举 $j \neq i$ 且 $A_{j,i} \neq 0$ 的 j，那么我们只需要把第 i 行整体乘上 $\frac{A_{j,i}}{A_{i,i}}$ 然后把第 j 行整体减掉这一行即可，这样做完之后第 i 列就只有 $A_{i,i} \neq 0$ 了，就完成了对第 i 个变量的消元。

高斯消元

- 考虑一些特殊情况，如果在第 1 步里没有找到符合条件的 j，那么此时可以第 i 个变量在前 $i-1$ 行系数有可能非 0，我们去找是否有 $k<i$ 满足 $A_{k,k}=0$ 且 $A_{k,i}\neq 0$ 有的话交换过来即可。

高斯消元

- 考虑一些特殊情况，如果在第 1 步里没有找到符合条件的 j，那么此时可以第 i 个变量在前 $i-1$ 行系数有可能非 0，我们去找是否有 $k<i$ 满足 $A_{k,k}=0$ 且 $A_{k,i}\neq 0$ 有的话交换过来即可。
- 消元结束之后矩阵形如这个样子：

高斯消元

- 考虑一些特殊情况，如果在第 1 步里没有找到符合条件的 j，那么此时可以第 i 个变量在前 $i-1$ 行系数有可能非 0，我们去找是否有 $k<i$ 满足 $A_{k,k}=0$ 且 $A_{k,i}\neq 0$ 有的话交换过来即可。
- 消元结束之后矩阵形如这个样子：
  - 如果 $A_{i,i}\neq 0$，那么所有行在第 i 列都 $=0$，我们称这样的变量为主元。

高斯消元

- 考虑一些特殊情况，如果在第 1 步里没有找到符合条件的 j，那么此时可以第 i 个变量在前 $i-1$ 行系数有可能非 0，我们去找是否有 $k<i$ 满足 $A_{k,k}=0$ 且 $A_{k,i}\neq 0$ 有的话交换过来即可。
- 消元结束之后矩阵形如这个样子：
  - 如果 $A_{i,i}\neq 0$，那么所有行在第 i 列都 $=0$，我们称这样的变量为主元。
  - 如果 $A_{i,i}=0$，那么这一行所有系数一定都为 0，有可能在一些 $j<i, A_{j,j}\neq 0$ 的 $A_{j,i}=0$，我们称这样的变量为自由元，。

高斯消元

- 考虑一些特殊情况，如果在第 1 步里没有找到符合条件的 j，那么此时可以第 i 个变量在前 $i-1$ 行系数有可能非 0，我们去找是否有 $k<i$ 满足 $A_{k,k}=0$ 且 $A_{k,i}\neq 0$ 有的话交换过来即可。
- 消元结束之后矩阵形如这个样子：
  - 如果 $A_{i,i}\neq 0$，那么所有行在第 i 列都 $=0$，我们称这样的变量为主元。
  - 如果 $A_{i,i}=0$，那么这一行所有系数一定都为 0，有可能在一些 $j<i, A_{j,j}\neq 0$ 的 $A_{j,i}=0$，我们称这样的变量为自由元，。
- 如果某个变量 i 是自由元但是 $b_i\neq 0$，则方程组无解。

高斯消元

- 考虑一些特殊情况，如果在第 1 步里没有找到符合条件的 j，那么此时可以第 i 个变量在前 $i-1$ 行系数有可能非 0，我们去找是否有 $k<i$ 满足 $A_{k,k}=0$ 且 $A_{k,i}\neq 0$ 有的话交换过来即可。
- 消元结束之后矩阵形如这个样子：
  - 如果 $A_{i,i}\neq 0$，那么所有行在第 i 列都 $=0$，我们称这样的变量为主元。
  - 如果 $A_{i,i}=0$，那么这一行所有系数一定都为 0，有可能在一些 $j<i, A_{j,j}\neq 0$ 的 $A_{j,i}=0$，我们称这样的变量为自由元，。
- 如果某个变量 i 是自由元但是 $b_i\neq 0$，则方程组无解。
- 同时可以发现自由元可以任意取值，且自由元的取值确定之后主元的取值也就确定了。

高斯消元

- 考虑一些特殊情况，如果在第 1 步里没有找到符合条件的 j，那么此时可以第 i 个变量在前 $i-1$ 行系数有可能非 0，我们去找是否有 $k<i$ 满足 $A_{k,k}=0$ 且 $A_{k,i}\neq 0$ 有的话交换过来即可。
- 消元结束之后矩阵形如这个样子：
  - 如果 $A_{i,i}\neq 0$，那么所有行在第 i 列都 $=0$，我们称这样的变量为主元。
  - 如果 $A_{i,i}=0$，那么这一行所有系数一定都为 0，有可能在一些 $j<i, A_{j,j}\neq 0$ 的 $A_{j,i}=0$，我们称这样的变量为自由元，。
- 如果某个变量 i 是自由元但是 $b_i\neq 0$，则方程组无解。
- 同时可以发现自由元可以任意取值，且自由元的取值确定之后主元的取值也就确定了。
- 时间复杂度通常是 $O(n^3)$ 的。

[ABC366G] XOR Neighbors

- 给定一个无向图，你需要给每个点填上一个 $[1, 2^{60} - 1]$ 之间的数字，使得每个点邻居上的数字的异或和为 0。
  $n \leq 60$。

CF1616F - Tricolor Triangles

- 给一个简单无向图，每条边可以被染成 $1,2,3$ 三种颜色之一，有的边已经染过色了。
  你要对未被染色的边染色，使得所有三元环的边要么颜色都相同要么颜色都不同。
  $n \leq 64, m \leq 256$。

行列式

- 对于一个 n 阶方阵 A，定义其行列式为：

$$\det(A) = \sum_{p} (-1)^{\text{sign}(p)} \prod_{i=1}^{n} A_{i,p_i}$$

其中 p 枚举了所有 n 阶排列，$\text{sign}(p)$ 表示 p 的逆序对数量。

行列式

- 对于一个 n 阶方阵 A，定义其行列式为：

$$\det(A) = \sum_{p} (-1)^{\text{sign}(p)} \prod_{i=1}^{n} A_{i,p_i}$$

其中 p 枚举了所有 n 阶排列，$\text{sign}(p)$ 表示 p 的逆序对数量。

- 行列式具有一些性质：

行列式

- 对于一个 n 阶方阵 A，定义其行列式为：

$$\det(A) = \sum_{p} (-1)^{\text{sign}(p)} \prod_{i=1}^{n} A_{i,p_i}$$

  其中 p 枚举了所有 n 阶排列，$\text{sign}(p)$ 表示 p 的逆序对数量。
- 行列式具有一些性质：
  - 三角矩阵行列式为对角线上所有元素的乘积。

行列式

- 对于一个 n 阶方阵 A，定义其行列式为：

$$\det(A) = \sum_{p} (-1)^{\text{sign}(p)} \prod_{i=1}^{n} A_{i,p_i}$$

  其中 p 枚举了所有 n 阶排列，$\text{sign}(p)$ 表示 p 的逆序对数量。
- 行列式具有一些性质：
  - 三角矩阵行列式为对角线上所有元素的乘积。
  - 把矩阵 A 任意一行或一列乘以常数 k，行列式变为 $k\det(A)$。

行列式

- 对于一个 n 阶方阵 A，定义其行列式为：

$$\det(A) = \sum_{p} (-1)^{\text{sign}(p)} \prod_{i=1}^{n} A_{i,p_i}$$

  其中 p 枚举了所有 n 阶排列，$\text{sign}(p)$ 表示 p 的逆序对数量。
- 行列式具有一些性质：
  - 三角矩阵行列式为对角线上所有元素的乘积。
  - 把矩阵 A 任意一行或一列乘以常数 k，行列式变为 $k\det(A)$。
  - 交换行列式任意两行或两列，矩阵的行列式变为 $-\det(A)$。

行列式

- 对于一个 n 阶方阵 A，定义其行列式为：

$$\det(A) = \sum_{p} (-1)^{\text{sign}(p)} \prod_{i=1}^{n} A_{i,p_i}$$

  其中 p 枚举了所有 n 阶排列，$\text{sign}(p)$ 表示 p 的逆序对数量。
- 行列式具有一些性质：
  - 三角矩阵行列式为对角线上所有元素的乘积。
  - 把矩阵 A 任意一行或一列乘以常数 k，行列式变为 $k\det(A)$。
  - 交换行列式任意两行或两列，矩阵的行列式变为 $-\det(A)$。
  - 将某一行或列的所有元素乘上常数 k 之后加到另一行或列的对应元素上，行列式不变。

行列式

- 有了这些性质，我们很容易有一个想法是通过高斯消元法把矩阵变成对角矩阵或者三角矩阵，然后直接用对角线的乘积求出行列式的值即可。

行列式

- 有了这些性质，我们很容易有一个想法是通过高斯消元法把矩阵变成对角矩阵或者三角矩阵，然后直接用对角线的乘积求出行列式的值即可。
- 在高斯消元的过程中，我们会用到交换两行，以及把某一行乘上常数 k 之后加到另一行上的操作，前者会让行列式乘 -1，后者不改变行列式，所以可以方便维护。

行列式

- 有了这些性质，我们很容易有一个想法是通过高斯消元法把矩阵变成对角矩阵或者三角矩阵，然后直接用对角线的乘积求出行列式的值即可。
- 在高斯消元的过程中，我们会用到交换两行，以及把某一行乘上常数 k 之后加到另一行上的操作，前者会让行列式乘 -1，后者不改变行列式，所以可以方便维护。
- 时间复杂度 $O(n^3)$。

行列式

- 有了这些性质，我们很容易有一个想法是通过高斯消元法把矩阵变成对角矩阵或者三角矩阵，然后直接用对角线的乘积求出行列式的值即可。
- 在高斯消元的过程中，我们会用到交换两行，以及把某一行乘上常数 k 之后加到另一行上的操作，前者会让行列式乘 -1，后者不改变行列式，所以可以方便维护。
- 时间复杂度 $O(n^3)$。
- 行列式的本质是若干向量在高位空间夹出的有向几何体的体积。

线性组合，线性无关

- 对于若干向量 $\mathbf{a}_1, \mathbf{a}_2 \dots \mathbf{a}_n$，我们给每个向量分配一个系数 $k_1, k_2 \dots k_n$，称 $\sum_i k_i \mathbf{a}_i$ 为这些向量的线性组合。

线性组合，线性无关

- 对于若干向量 $\mathbf{a}_1, \mathbf{a}_2 \ldots \mathbf{a}_n$，我们给每个向量分配一个系数 $k_1, k_2 \ldots k_n$，称 $\sum_i k_i \mathbf{a}_i$ 为这些向量的线性组合。
- 若向量 β 可以表示为 $\mathbf{a}_1, \mathbf{a}_2 \ldots \mathbf{a}_n$ 的线性组合，则称 β 可被这些向量线性表出。

线性组合，线性无关

- 对于若干向量 $\mathbf{a}_1, \mathbf{a}_2 \dots \mathbf{a}_n$，我们给每个向量分配一个系数 $k_1, k_2 \dots k_n$，称 $\sum_i k_i \mathbf{a}_i$ 为这些向量的线性组合。
- 若向量 β 可以表示为 $\mathbf{a}_1, \mathbf{a}_2 \dots \mathbf{a}_n$ 的线性组合，则称 β 可被这些向量线性表出。
- 对于向量 $\mathbf{a}_1, \mathbf{a}_2 \dots \mathbf{a}_n$，如果任何一个向量都无法被其余向量线性表出，则称这些向量线性无关，否称为线性相关。另一个等价表达是，如果 $\sum_i k_i \mathbf{a}_i$ 是零向量，则 k_i 均等于 0。

极大线性无关组，秩

- 对于向量组 $\mathbf{a}_1, \mathbf{a}_2 \dots \mathbf{a}_n$，若从中选出一些向量 $\mathbf{b}_1, \mathbf{b}_2 \dots \mathbf{b}_n$ 使得这些向量线性无关，并且加入剩余任意一个向量都会导致线性相关，则称 $\mathbf{b}_1, \mathbf{b}_2 \dots \mathbf{b}_n$ 为一个极大线性无关组，或称为基，其大小称为秩 (rank)。

极大线性无关组，秩

- 对于向量组 $\mathbf{a}_1, \mathbf{a}_2 \dots \mathbf{a}_n$，若从中选出一些向量 $\mathbf{b}_1, \mathbf{b}_2 \dots \mathbf{b}_n$ 使得这些向量线性无关，并且加入剩余任意一个向量都会导致线性相关，则称 $\mathbf{b}_1, \mathbf{b}_2 \dots \mathbf{b}_n$ 为一个极大线性无关组，或称为基，其大小称为秩 (rank)。
- 一个向量组的任何极大线性无关组的大小都相等。

极大线性无关组，秩

- 对于向量组 $\mathbf{a}_1, \mathbf{a}_2 \dots \mathbf{a}_n$，若从中选出一些向量 $\mathbf{b}_1, \mathbf{b}_2 \dots \mathbf{b}_n$ 使得这些向量线性无关，并且加入剩余任意一个向量都会导致线性相关，则称 $\mathbf{b}_1, \mathbf{b}_2 \dots \mathbf{b}_n$ 为一个极大线性无关组，或称为基，其大小称为秩 (rank)。
- 一个向量组的任何极大线性无关组的大小都相等。
- 向量组里的任何一个向量都可以被其极大线性无关组线性表出。

极大线性无关组，秩

- 对于向量组 $\mathbf{a}_1, \mathbf{a}_2 \dots \mathbf{a}_n$，若从中选出一些向量 $\mathbf{b}_1, \mathbf{b}_2 \dots \mathbf{b}_n$ 使得这些向量线性无关，并且加入剩余任意一个向量都会导致线性相关，则称 $\mathbf{b}_1, \mathbf{b}_2 \dots \mathbf{b}_n$ 为一个极大线性无关组，或称为基，其大小称为秩 (rank)。
- 一个向量组的任何极大线性无关组的大小都相等。
- 向量组里的任何一个向量都可以被其极大线性无关组线性表出。
- 若若干向量构成了一个线性无关组，则做初等行变换后依旧是线性无关的，故我们可以通过对行向量构成的矩阵做高斯消元，消元完成后所有主元构成了一组基。

线性基

- OI 中最常用的是二进制异或线性基，我们可以认为所有数字都是 $0 \sim 2^d - 1$ 的整数，代表了一个 d 维 01 向量，所有加减乘除在 mod2 意义下定义。

线性基

- OI 中最常用的是二进制异或线性基，我们可以认为所有数字都是 $0 \sim 2^d - 1$ 的整数，代表了一个 d 维 01 向量，所有加减乘除在 mod2 意义下定义。
- 形式化地，对于 n 个整数 $a_1 \sim a_n$，我们希望求出若干整数 $b_1, b_2 \sim b_k$，使得 b 序列任意一个子集内的数异或 $\neq 0$，且任意 a 序列里的数字都可以表示成 b 序列的若干数的异或和，且 k 尽可能大。

线性基

- OI 中最常用的是二进制异或线性基，我们可以认为所有数字都是 $0 \sim 2^d - 1$ 的整数，代表了一个 d 维 01 向量，所有加减乘除在 mod2 意义下定义。
- 形式化地，对于 n 个整数 $a_1 \sim a_n$，我们希望求出若干整数 $b_1, b_2 \sim b_k$，使得 b 序列任意一个子集内的数异或 $\neq 0$，且任意 a 序列里的数字都可以表示成 b 序列的若干数的异或和，且 k 尽可能大。
- 一种构造方法是模拟高斯消元的过程。

线性基

- OI 中最常用的是二进制异或线性基，我们可以认为所有数字都是 $0 \sim 2^d - 1$ 的整数，代表了一个 d 维 01 向量，所有加减乘除在 mod2 意义下定义。
- 形式化地，对于 n 个整数 $a_1 \sim a_n$，我们希望求出若干整数 $b_1, b_2 \sim b_k$，使得 b 序列任意一个子集内的数异或 $\neq 0$，且任意 a 序列里的数字都可以表示成 b 序列的若干数的异或和，且 k 尽可能大。
- 一种构造方法是模拟高斯消元的过程。
- 另一种更为常见的过程是，我们依次向基里插入 $a_1 \sim a_n$，每次插入 a_i 时从高到低枚举每一位 j：

线性基

- OI 中最常用的是二进制异或线性基，我们可以认为所有数字都是 $0 \sim 2^d - 1$ 的整数，代表了一个 d 维 01 向量，所有加减乘除在 mod2 意义下定义。
- 形式化地，对于 n 个整数 $a_1 \sim a_n$，我们希望求出若干整数 $b_1, b_2 \sim b_k$，使得 b 序列任意一个子集内的数异或 $\neq 0$，且任意 a 序列里的数字都可以表示成 b 序列的若干数的异或和，且 k 尽可能大。
- 一种构造方法是模拟高斯消元的过程。
- 另一种更为常见的过程是，我们依次向基里插入 $a_1 \sim a_n$，每次插入 a_i 时从高到低枚举每一位 j：
  - 如果 a_i 在这一位是 0 就跳过。

线性基

- OI 中最常用的是二进制异或线性基，我们可以认为所有数字都是 $0 \sim 2^d - 1$ 的整数，代表了一个 d 维 01 向量，所有加减乘除在 mod2 意义下定义。
- 形式化地，对于 n 个整数 $a_1 \sim a_n$，我们希望求出若干整数 $b_1, b_2 \sim b_k$，使得 b 序列任意一个子集内的数异或 $\neq 0$，且任意 a 序列里的数字都可以表示成 b 序列的若干数的异或和，且 k 尽可能大。
- 一种构造方法是模拟高斯消元的过程。
- 另一种更为常见的过程是，我们依次向基里插入 $a_1 \sim a_n$，每次插入 a_i 时从高到低枚举每一位 j：
  - 如果 a_i 在这一位是 0 就跳过。
  - 否则，如果基里这一位上已经有主元了，就把当前数异或上那个主元，继续下一位；否则把 a_i 作为这一位上点的主元。

线性基

- OI 中最常用的是二进制异或线性基，我们可以认为所有数字都是 $0 \sim 2^d - 1$ 的整数，代表了一个 d 维 01 向量，所有加减乘除在 mod2 意义下定义。
- 形式化地，对于 n 个整数 $a_1 \sim a_n$，我们希望求出若干整数 $b_1, b_2 \sim b_k$，使得 b 序列任意一个子集内的数异或 $\neq 0$，且任意 a 序列里的数字都可以表示成 b 序列的若干数的异或和，且 k 尽可能大。
- 一种构造方法是模拟高斯消元的过程。
- 另一种更为常见的过程是，我们依次向基里插入 $a_1 \sim a_n$，每次插入 a_i 时从高到低枚举每一位 j：
  - 如果 a_i 在这一位是 0 就跳过。
  - 否则，如果基里这一位上已经有主元了，就把当前数异或上那个主元，继续下一位；否则把 a_i 作为这一位上点的主元。
  - 如果枚举到最低位也没有插入进去，说明 a_i 可以被基里的数线性表出，就不用插入了。

线性基

- OI 中最常用的是二进制异或线性基，我们可以认为所有数字都是 $0 \sim 2^d - 1$ 的整数，代表了一个 d 维 01 向量，所有加减乘除在 mod2 意义下定义。
- 形式化地，对于 n 个整数 $a_1 \sim a_n$，我们希望求出若干整数 $b_1, b_2 \sim b_k$，使得 b 序列任意一个子集内的数异或 $\neq 0$，且任意 a 序列里的数字都可以表示成 b 序列的若干数的异或和，且 k 尽可能大。
- 一种构造方法是模拟高斯消元的过程。
- 另一种更为常见的过程是，我们依次向基里插入 $a_1 \sim a_n$，每次插入 a_i 时从高到低枚举每一位 j：
  - 如果 a_i 在这一位是 0 就跳过。
  - 否则，如果基里这一位上已经有主元了，就把当前数异或上那个主元，继续下一位；否则把 a_i 作为这一位上点的主元。
  - 如果枚举到最低位也没有插入进去，说明 a_i 可以被基里的数线性表出，就不用插入了。
- 时间复杂度 $O(nd)$，其中 d 为二进制位数。

线性基

- 另一种更为常见的过程是，我们依次向基里插入 $a_1 \sim a_n$，每次插入 a_i 时从高到低枚举每一位 j：

线性基

- 另一种更为常见的过程是，我们依次向基里插入 $a_1 \sim a_n$，每次插入 a_i 时从高到低枚举每一位 j：
  - 如果 a_i 在这一位是 0 就跳过。

线性基

- 另一种更为常见的过程是，我们依次向基里插入 $a_1 \sim a_n$，每次插入 a_i 时从高到低枚举每一位 j：
  - 如果 a_i 在这一位是 0 就跳过。
  - 否则，如果基里这一位上已经有主元了，就把当前数异或上那个主元，继续下一位；否则把 a_i 作为这一位上点的主元。

线性基

- 另一种更为常见的过程是，我们依次向基里插入 $a_1 \sim a_n$，每次插入 a_i 时从高到低枚举每一位 j：
  - 如果 a_i 在这一位是 0 就跳过。
  - 否则，如果基里这一位上已经有主元了，就把当前数异或上那个主元，继续下一位；否则把 a_i 作为这一位上点的主元。
  - 如果枚举到最低位也没有插入进去，说明 a_i 可以被基里的数线性表出，就不用插入了。

线性基

- 另一种更为常见的过程是，我们依次向基里插入 $a_1 \sim a_n$，每次插入 a_i 时从高到低枚举每一位 j：
  - 如果 a_i 在这一位是 0 就跳过。
  - 否则，如果基里这一位上已经有主元了，就把当前数异或上那个主元，继续下一位；否则把 a_i 作为这一位上点的主元。
  - 如果枚举到最低位也没有插入进去，说明 a_i 可以被基里的数线性表出，就不用插入了。
- 正确性与高斯消元是类似的，朴素实现时间复杂度 $O(nd^2)$，其中 d 为二进制位数，使用 bitset 或者手写二进制压位可以变成 $O(nd^2/w)$。

线性基

- 另一种更为常见的过程是，我们依次向基里插入 $a_1 \sim a_n$，每次插入 a_i 时从高到低枚举每一位 j：
  - 如果 a_i 在这一位是 0 就跳过。
  - 否则，如果基里这一位上已经有主元了，就把当前数异或上那个主元，继续下一位；否则把 a_i 作为这一位上点的主元。
  - 如果枚举到最低位也没有插入进去，说明 a_i 可以被基里的数线性表出，就不用插入了。
- 正确性与高斯消元是类似的，朴素实现时间复杂度 $O(nd^2)$，其中 d 为二进制位数，使用 bitset 或者手写二进制压位可以变成 $O(nd^2/w)$。
- 这个构建线性基的思路即使是一般的实数线性基也是可以的，只需要把异或改成正常消元即可。

[JLOI2015] 装备购买

- 给 n 个 m 维向量，每个向量有一个权值，求最多可以选多少个线性无关向量，以及极大线性无关组的权值和最小是多少。

线性基

- 高斯消元后的矩阵应该满足每个主元仅在其那一行不为 0，而我们上述贪心法构建的线性基可能某些低位的 1 在高位也是 1，故为了具有这样的性质我们可以枚举每一个主元，如果其在第 j 位上是 1 且第 j 位也是主元就把该主元异或上第 j 为的主元，这样处理完之后就满足了上述性质，且容易说明这还是一组基。

线性基

- 高斯消元后的矩阵应该满足每个主元仅在其那一行不为 0，而我们上述贪心法构建的线性基可能某些低位的 1 在高位也是 1，故为了具有这样的性质我们可以枚举每一个主元，如果其在第 j 位上是 1 且第 j 位也是主元就把该主元异或上第 j 为的主元，这样处理完之后就满足了上述性质，且容易说明这还是一组基。
- 这样的线性基可以解决更多问题：

线性基

- 高斯消元后的矩阵应该满足每个主元仅在其那一行不为 0，而我们上述贪心法构建的线性基可能某些低位的 1 在高位也是 1，故为了具有这样的性质我们可以枚举每一个主元，如果其在第 j 位上是 1 且第 j 位也是主元就把该主元异或上第 j 为的主元，这样处理完之后就满足了上述性质，且容易说明这还是一组基。
- 这样的线性基可以解决更多问题：
  - 求能异或出的最大值：如果不做处理我们需要从高到低枚举每一位，如果答案这一位是 0 就异或上基里这一位的主元，否则就不异或；而处理之后直接输出所有主元的异或即可。

线性基

- 高斯消元后的矩阵应该满足每个主元仅在其那一行不为 0，而我们上述贪心法构建的线性基可能某些低位的 1 在高位也是 1，故为了具有这样的性质我们可以枚举每一个主元，如果其在第 j 位上是 1 且第 j 位也是主元就把该主元异或上第 j 为的主元，这样处理完之后就满足了上述性质，且容易说明这还是一组基。
- 这样的线性基可以解决更多问题：
  - 求能异或出的最大值：如果不做处理我们需要从高到低枚举每一位，如果答案这一位是 0 就异或上基里这一位的主元，否则就不异或；而处理之后直接输出所有主元的异或即可。
  - 求能异或出的第 k 小的数：此时因为主元之间互不影响，我们可以把每个主元当成一个二进制位，所以只需要把 k 做二进制分解之后得到的是 1 的那些位的主元异或起来即可。

线性基合并

要合并两个线性基很简单，只需要暴力把其中一个线性基里的元素插入另一个即可。

前缀线性基

- 考虑如下问题：

前缀线性基

- 考虑如下问题：

CF1100F Ivan and Burgers

给定 $a_1, a_2 \sim a_n$，q 次询问 $a_l, a_{l+1} \ldots a_r$ 选出一些数异或的最大值。

前缀线性基

- 考虑如下问题：

CF1100F Ivan and Burgers

给定 $a_1, a_2 \sim a_n$，q 次询问 $a_l, a_{l+1} \ldots a_r$ 选出一些数异或的最大值。

- 区间询问，可以考虑线段树维护区间线性基，每次把两边线性基合并起来，时间 $O(n \log^3 n)$

前缀线性基

- 考虑如下问题：

CF1100F Ivan and Burgers

给定 $a_1, a_2 \sim a_n$，q 次询问 $a_l, a_{l+1} \ldots a_r$ 选出一些数异或的最大值。

- 区间询问，可以考虑线段树维护区间线性基，每次把两边线性基合并起来，时间 $O(n \log^3 n)$
- 猫树分治，询问就只需要合并一次前后缀，时间 $O(n \log^2 n)$。

前缀线性基

- 离线下来对 r 扫描线，维护每个 $[1,r],[2,r]\dots[r,r]$ 的线性基，这个东西考虑如果从 r 开始倒着加数的话只有 $O(\log n)$ 次变化。

前缀线性基

- 离线下来对 r 扫描线，维护每个 $[1,r],[2,r]\dots[r,r]$ 的线性基，这个东西考虑如果从 r 开始倒着加数的话只有 $O(\log n)$ 次变化。
- 我们考虑对线性基里每个数维护一个时间戳 t_j，在插入一个数 a_i 时从高到低考虑每个是 1 的位 j，如果原本的数时间戳比现在的要早就把当前的替换过去:

前缀线性基

- 离线下来对 r 扫描线，维护每个 $[1,r],[2,r]\dots[r,r]$ 的线性基，这个东西考虑如果从 r 开始倒着加数的话只有 $O(\log n)$ 次变化。
- 我们考虑对线性基里每个数维护一个时间戳 t_j，在插入一个数 a_i 时从高到低考虑每个是 1 的位 j，如果原本的数时间戳比现在的要早就把当前的替换过去：
  - 如果 j 这一位上没数就把 a_i 存进去，时间戳也存一下。

前缀线性基

- 离线下来对 r 扫描线，维护每个 $[1,r],[2,r]\dots[r,r]$ 的线性基，这个东西考虑如果从 r 开始倒着加数的话只有 $O(\log n)$ 次变化。
- 我们考虑对线性基里每个数维护一个时间戳 t_j，在插入一个数 a_i 时从高到低考虑每个是 1 的位 j，如果原本的数时间戳比现在的要早就把当前的替换过去：
  - 如果 j 这一位上没数就把 a_i 存进去，时间戳也存一下。
  - 否则如果 $t_j < t_i$，把 a_i 存在这个位置，然后把原本这一位上存的数继续往下插入。

前缀线性基

- 离线下来对 r 扫描线，维护每个 $[1,r],[2,r]\dots[r,r]$ 的线性基，这个东西考虑如果从 r 开始倒着加数的话只有 $O(\log n)$ 次变化。
- 我们考虑对线性基里每个数维护一个时间戳 t_j，在插入一个数 a_i 时从高到低考虑每个是 1 的位 j，如果原本的数时间戳比现在的要早就把当前的替换过去：
  - 如果 j 这一位上没数就把 a_i 存进去，时间戳也存一下。
  - 否则如果 $t_j < t_i$，把 a_i 存在这个位置，然后把原本这一位上存的数继续往下插入。
- 这样维护的线性基里每个数就是尽可能靠后的，当查询区间 $[l,r]$ 的线性基时只需要把时间戳 $\geq l$ 的元素拿出来即可。

[SCOI2016] 幸运数字

- 给出一棵包含 n 个点的树，点带权。
  有 Q 次询问，每次询问给出两个点 x 和 y，求 x 到 y 的简单路径上，任意选择若干个点，使得其点权异或和最大。
  数据范围：$1 \leq n \leq 2 \times 10^4$，$1 \leq Q \leq 2 \times 10^5$，$0 \leq g_i \leq 2^{60}$。时空限制：4000 ms/256 MiB。

[集训队互测 2015] 最大异或和

- 我有一个数列 $a_1, a_2, \ldots, a_n$，每个 a_i 都是小于 2^m 的非负整数。
  现在请您实现三种操作，格式说明如下：
  * 1 x y w：对于所有 $x \le i \le y$，将 a_i 修改为 $a_i \operatorname{xor} w$。其中 w 是一个小于 2^m 的非负整数。
  * 2 x y w：对于所有 $x \le i \le y$，将 a_i 修改为 w。其中 w 是一个小于 2^m 的非负整数。
  * 3：从 $a_1, a_2, \ldots, a_n$ 中选出若干个数，使得选出的数异或和最大。请输出这个最大值。
  这里 xor 表示按位异或运算，$x_1, x_2, \ldots, x_l$ 的异或和是指 $x_1 \operatorname{xor} x_2 \operatorname{xor} \ldots \operatorname{xor} x_l$。
  对于 100% 的数据，$n, m, q \le 2000$。

[WC2011] 最大 XOR 和路径

- 考虑一个边权为非负整数的无向连通图，节点编号为 1 到 N，试求出一条从 1 号节点到 N 号节点的路径，使得路径上经过的边的权值的 XOR 和最大。
  路径可以重复经过某些点或边，当一条边在路径中出现了多次时，其权值在计算 XOR 和时也要被计算相应多的次数，具体见样例。
  对于 100% 的数据，$N \leq 50000$，$M \leq 100000$，$D_i \leq 10^{18}$。

albus 就是要第一个出场

- 已知一个长度为 n 的正整数序列 A（下标从 1 开始），令 $S = \{x|1 \le x \le n\}$，S 的幂集 2^S 定义为 S 所有子集构成的集合。定义映射
  $f : 2^S \to Z, f(\emptyset) = 0, f(T) = \mathrm{XOR}\{A_t\}, (t \in T)$。
  现在 albus 把 2^S 中每个集合的 f 值计算出来，从小到大排成一行，记为序列 B（下标从 1 开始）。
  给定一个数，那么这个数在序列 B 中第 1 次出现时的下标是多少呢？
  $1 \le N \le 10,0000$ 其他所有输入均不超过 10^9

Thanks!