# 第 3 章 数 学 基 础

## 3.1 函数增长与复杂性分类

### 3.1.1 渐进符号

**【基本概念】**

计算算法的时间复杂度时，往往并不需要花很大的力气算出太精确的结果，对于足够大的输入规模来说，我们只需要关心运行时间的增长量级，也就是研究算法的渐进效率。以下介绍几种渐进符号。

**$\Theta$记号**

$$\Theta\big(g(n)\big)=\{f(n)|\ \exists n_0,c_1,c_2>0,\text{s.t.}\forall n>n_0,0\leqslant c_1g(n)\leqslant f(n)\leqslant c_2g(n)\}$$

对于$\Theta$记号，$g(n)$本身必须是渐进非负的，否则集合就是空集了。所以在考虑渐进符号时，一般都要求$g(n)$本身为渐进非负。$\Theta$记号一般用来表示一个算法对于某些输入的运行时间的确界。因为$\Theta\big(g(n)\big)$是集合，可以写成$f(n)\in\Theta(g(n))$，通常也记为$f(n)=\Theta\big(g(n)\big)$。可以看出，对于任意函数$f(n)\in\Theta\big(g(n)\big)$，那么$c_2g(n)$和$c_1g(n)$分别是$f(n)$的上界和下界。所以$\Theta$记号渐进地给出了函数的上界和下界。

比如$f(n)=3n-1\in\Theta(n)$，取常数$n_0=1,c_1=2,c_2=3$，满足$0\leqslant 2n\leqslant 3n-1\leqslant 3n$对$n>1$成立，所以$3n-1=\Theta(n)$。

**$O$记号**

$$O\big(g(n)\big)=\{f(n)|\ \exists n_0,c>0,\text{s.t.}\forall n>n_0,0\leqslant f(n)\leqslant cg(n)\}$$

$O$记号表示的是一个渐进上界，有时也被非正式地用来描述渐进上确界。当用它来表示算法的最坏情况运行时间界限时，就隐含地给出了对于任意输入的时间的上界，即$f(n)=\Theta\big(g(n)\big)$隐含着$f(n)=O\big(g(n)\big)$。比如，对于插入排序，最坏情况下运行时间上界$O(n^2)$对于所有输入均成立，而界$\Theta(n^2)$却不是对每种输入都成立的，比如当输入已经是排好序的数列时，运行时间为$\Theta(n)\neq\Theta(n^2)$。

**$o$ 记号**

$$o\big(g(n)\big)=\{f(n)|\ \forall c>0,\exists n_0>0,\text{s.t.}\forall n>n_0,0\leqslant f(n)<cg(n)\}$$

也可以理解为 $\lim_{n\to\infty}\frac{f(n)}{g(n)}=0$。$o$记号表示的非渐进紧确的上界。

比如 $n=o(n^2)$，因为 $\lim_{n\to\infty}\frac{n}{n^2}=\lim_{n\to\infty}\frac{1}{n}=0$。如果函数 $f(n)=\sum_{i=0}^{k}a_i n^i$，那么 $f(n)\in o(n^{k+1})\subseteq o(n^{k+2})\subseteq\cdots$。

**$\Omega$记号**

$$\Omega\big(g(n)\big)=\{f(n)|\ \exists n_0,c>0,\text{s.t.}\forall n>n_0,0\leqslant cg(n)\leqslant f(n)\}$$

正如$O$记号给出了一个函数的渐进上界，$\Omega$记号则给出了函数的渐进下界。研究函数的渐进下界对改善算法复杂度的研究有重要意义。

比如$2n^2=\Omega(n^2)$，取$c=2$，即可证得。与$O$记号对称的，$f(n)=\Theta\big(g(n)\big)$隐含着$f(n)=\Omega\big(g(n)\big)$。进一步，对任意两个函数$f(n)$和$g(n)$，$f(n)=\Theta\big(g(n)\big)$当且仅当$f(n)=O\big(g(n)\big)$且$f(n)=\Omega\big(g(n)\big)$。

**$\omega$记号**

$$\omega\big(g(n)\big)=\{f(n)|\ \forall c>0,\exists n_0>0,\text{s.t.}\forall n>n_0,0\leqslant cg(n)<f(n)\}$$

也可以理解为 $\lim_{n\to\infty}\frac{f(n)}{g(n)}=\infty$

$\omega$记号与$\Omega$记号的关系，正如$o$记号与$O$记号的关系一样，表示的是非渐进紧确的下界。比如 $n^2=\omega(n)$，因为 $\lim_{n\to\infty}\frac{n^2}{n}=\lim_{n\to\infty}n=\infty$。如果函数 $f(n)=\sum_{i=0}^{k}a_i n^i$，那么 $f(n)\in\omega(n^{k-1})\subseteq\omega(n^{k-2})\subseteq\cdots\subseteq\omega(1)$。

**【性质】**

1. $O(g1(n))+O(g2(n))=O(\max\{g1(n),g2(n)\})$。同样，对于$\Theta$和$\Omega$记号也成立。
2. $O\big(g1(n)\big)\cdot O(g2(n))=O(g1(n)g2(n))$。同样，对于$\Theta$和$\Omega$记号也成立。

### 3.1.2 阶的计算

**【基本概念】**

在计算中，通常有如下复杂度关系：

$$c<\log n<n<n\log n<n^a<a^n<n!$$

其中$c$是常数，$a$是大于 1 的常数。

对于递归算法的时间复杂度的阶的计算，一般使用主定理计算

$$T(n)=aT(n/b)+f(n)$$

$T(n)$的阶需要分三种情况讨论：

（1）$\exists\epsilon>0,\text{s.t.}f(n)=O(n^{\log_b a-\epsilon})$，则$T(n)=\Theta(n^{\log_b a})$。

（2）$f(n)=\Theta(n^{\log_b a})$，则$T(n)=\Theta(n^{\log_b a}\log n)$。

（3）$\exists\epsilon>0,\text{s.t.}\,f(n)=\Omega(n^{\log_b a+\epsilon})$，且$\forall c<1,\exists n,\text{s.t.}af\left(\frac{n}{b}\right)\leqslant cf(n)$，则$T(n)=\Theta(f(n))$。

一般来说，$f(n)$的执行时间是多项式时间，即$T(n)=aT(n/b)+O(n^d)$，此时主定理可以写成更简单的形式：

（1）若$d>\log_b a$，则$T(n)=O(n^d)$。

（2）若$d=\log_b a$，则$T(n)=O(n^d\log n)$。

（3）若$d<\log_b a$，则$T(n)=O(n^{\log_b a})$。

在满二叉树的中序（前序、后序也成立）遍历中，对一颗大小为$n$的子树，先在左子树上遍历，经过根，再在右子树上遍历。也就是说把规模为$n$的问题分成两个相似的子问题，规模皆为$n/2$，而分解和合并的时间为$f(n)=\Theta(1)$，所以时间复杂度的计算式为

$$T(n)=2T(n/2)+f(n)$$

满足主定理的第一种情况，得$T(n)=\Theta(n)$。

在归并排序中，规模为$n$的问题分成两个相似的子问题，规模皆为$n/2$，再加上分解和合并的复杂度$f(n)=\Theta(n)$。将其写成主定理计算的标准形式，即

$$T(n)=2T(n/2)+f(n)$$

运用情况（2）的结论，得$T(n)=\Theta(n\log n)$。

**【用途】**

对于一般的递归形式的程序，皆可使用主定理计算其时间复杂度的阶。

### 3.1.3　复杂性分类

**【图灵机】**

图灵机是一种抽象的计算模型。它有 $k$ 个磁带，每个磁带都有一个磁头，可以对磁带的某个位置进行读或写操作。有一个磁带是输入磁带，该磁带只能进行读操作。剩下的 $k$–1 个磁带是工作磁带，可以读也可以写。

图灵机有一个寄存器，可以存储当前机器的状态。图灵机的状态集是有限的。

根据图灵机的当前状态，机器执行每一步时，会首先读取 $k$ 个磁带上磁头对应位置的内容，然后对 $k$–1 个工作磁带磁头位置的内容进行修改（修改的内容可以保持不变），接着改变寄存器中的状态（可以保持不变），最后分别将每个磁头移动一个位置或保持不变。对于确定性图灵机来说，如果确定了当前状态以及 $k$ 个磁头位置的内容，则机器执行的步骤是确定的，即机器有一套确定的规则。

可以把图灵机看做一种简化的电脑。磁带对应了内存，而规则和寄存器对应于中央处理器。

**【复杂度分类】**

P 复杂度类是指这样一类问题，这些问题都存在相应的确定性图灵机可以在输入长度的多项式时间内计算出结果。例如，两个整数的加法就是属于 P 的。

NP复杂度类是指这样一类问题，这些问题都存在相应的确定性图灵机可以在输入长度的多项式时间内判断一个结果是否正确。

以子集和问题为例：给定一个整数集合，需要判断该集合是否存在一个子集，该子集中的数加起来正好为 0。例如，$\{1,2,-3,4,-10\}$是一个满足条件的集合，因为子集$\{1,2,-3\}$里的数加起来为 0。虽然给定输入很难有一个多项式时间的算法来判断是否存在这样一个子集，但是只需要设计一个能做加法的机器，就能判断一个子集是不是加起来为 0。所以，子集和问题是属于NP的。

不难发现，所有属于 P 的问题也属于 NP，因此 P 是 NP 的一个子集。但是，P 是否是 NP 的一个真子集，即$P \neq NP$，却是计算机理论界尚未解决的一个很重要的问题。

NPC（NP-Complete）复杂度类也是NP的一个子集，它包含了NP里面最难的问题。如果能够证明某一个NPC问题是属于 P 的，就可以证明 P 是等于NP的。

**【扩展】**

除了图灵机，还有其他一些抽象计算模型，如递归函数、λ演算、生命游戏等。这些模型的计算能力与图灵机完全等价，因此有著名的邱奇-图灵论题：一切直觉上认为可以计算的函数都能构造出图灵机进行计算，反之亦然。

## 3.2 概 率 论

### 3.2.1 事件与概率

**【概念】**

1. 样本空间：一个随机试验的所有可能结果组成的集合，标记为$S$。
2. 事件：$S$的一个子集$A$，$A \subseteq S$，称为事件。
3. $A \cup B$，$A$与$B$的并，即发生$A$或$B$或者两者都发生。
   $A \cap B$，$A$与$B$的交，即$A$与$B$都发生。
   $A^c$，$A$的补，即$A$不发生。
   $A \backslash B$，$A$与$B$的差，即$A$发生，$B$不发生。
   $\emptyset$，空集，不可能事件。
4. 概率公理：满足以下 3 个条件的函数称为概率函数。

（a）$0 \leqslant P(A) \leqslant 1$。

（b）$P(S)=1$。

（c）如果$A_1A_2A_3\cdots$是一系列两两无关的事件，即对于任何$i,j,i\neq j$，$A_i\cap A_j=\emptyset$，则

$$P\left(\bigcup_{k=1}^{\infty}A_k\right)=\sum_{k=1}^{\infty}P(A_k)$$

5. 条件概率：令$B$为一个事件满足$P(B)>0$，对于任一事件$A$，定义$A$的关于$B$的条件概率。

$$P(A|B)=\frac{P(A\cap B)}{P(B)}$$

6. 独立：如果$A$，$B$满足$P(A\cap B)=P(A)P(B)$，称$A$，$B$独立。

并且可以推出$P(A)=P(A|B)$。

**【性质】**

1. 概率的加法公式：

$$P(A\cup B)=P(A)+P(B)-P(A\cap B)$$

2. 并的界：

$$P(A\cup B)\leqslant P(A)+P(B)$$

3. 全概率公式：

设$B_1B_2B_3\cdots B_n$是样本空间$S$中互不相交的一系列事件，并且满足$S=\cup_{k=1}^{n}B_k$，那么对于任意事件$A$

$$P(A)=\sum_{k=1}^{n}P(A|B_k)P(B_k)$$

**【举例】**

1. 假设有一枚硬币，对于任意一次投掷，正面朝上的概率为$p$。显而易见，正面朝下的概率为$1-p$。

投掷这枚硬币两次，两次都朝上的概率为$p^2$。因为$P$(两次都朝上)$=P$(第一次朝上$\cap$第二次朝上)。由于两次硬币投掷是独立的（互不影响的），所以$P$(第一次朝上$\cap$第二次朝上)$=P$(第一次朝上)$\times P$(第二次朝上)$=p^2$。

两次中至少一次朝上的概率$P$(至少一次朝上)$=P$(第一次朝上$\cup$第二次朝上)。由加法公式知概率为$2p-p^2$。也可以从补的角度思考，两次中至少一次朝上的反面是两次都朝下，概率为$(1-p)^2$，所以至少一次朝上的概率为$1-(1-p)^2=2p-p^2$。

2. 掷两次骰子，求两次和为 5 的概率。

掷两次骰子，总共有$6\times 6$种可能情况，其中和为 5 的有$1+4$，$2+3$，$3+2$，$4+1$，所以概率为

$$\frac{4}{36}=\frac{1}{9}$$

### 3.2.2 期望与方差

**【概念】**

1. 随机变量

在样本空间$S$上的一个随机变量$X$是$S$上的一个取值为实数的函数。只取有限个或者可数无穷多个值的随机变量称为离散随机变量。

例如，假设$X$是表示骰子点数的随机变量。那么对于一个公平的骰子来说，$P(X=1)=\frac{1}{6}$。

2. 数学期望

对于取值有限的离散型随机变量$X$，假设$X$有概率分布$p_j = P(X = x_j)$，则称$E(X)=\sum_{j=1}^{n} x_j p_j$为$X$的数学期望，其中$n$为$X$不同取值的个数。

例如对于一个公平的骰子来说，$X$是表示骰子点数的随机变量，则

$$E(X)=\frac{1}{6}\times(1+2+3+4+5+6)=3.5$$

3. 方差

定义离散随机变量$X$的方差为$\mathrm{var}(X) = E(X - E(X))^2$。称$\sigma_X = \sqrt{\mathrm{var}(X)}$为$X$的标准差。

**【性质】**

1. 期望的线性性

对于任意一组有限个离散随机变量来说$X_1, X_2, \cdots, X_n$

$$E\left(\sum_{i=1}^{n} X_i\right)=\sum_{i=1}^{n} E(x_i)$$

例如假设 $X$ 是表示骰子点数的随机变量，$Y$表示骰子投掷两次的点数和，由期望的线性性知$E(Y) = E(X + X) = 2E(X) = 7$。

2. 如果$X$和$Y$是两个独立的随机变量，那么

$$E(XY) = E(X)E(Y)$$

3. 马尔科夫不等式

假设$X$是只取非负数值的随机变量，对于任意$a > 0$，有

$$P(X \geqslant a) \leqslant \frac{E(X)}{a}$$

4. 切比雪夫不等式

对任意随机变量$X$及任意$a > 0$，有

$$P(|X-E(X)|\geqslant a)\leqslant\frac{\mathrm{var}(X)}{a^2}$$

**【概率分布】**

1. 二项分布，这个分布源于投硬币$n$次，假设每一次有$p$的概率正面朝上，那么这$n$次中总共出现$k$次正面朝上的概率。也可以解释为有一个事件$A$，它发生的概率为$p$，进行$n$次实验，事件$A$发生$k$次的概率。

二项分布的概率函数满足

$$p(k)=\binom{n}{k}p^k(1-p)^{n-k}$$

称为以$n,p$为参数的二项分布。

随机变量$X$表示$n$次试验中正面朝上的次数，由期望的线性性知，

$$E[X]=E_1[X]+\cdots+E_n[X]=n(0\times(1-p)+1\times p)=np$$

其中$E_i[X]$表示第$i$次朝上的期望。

$X$的方差$\mathrm{var}[X]=np(1-p)$。

2. 几何分布，这个分布源于投硬币$n$次，假设每一次有$p$的概率正面朝上，直到第$k$次才投出正面的概率。也可以解释为，一个事件$A$发生概率为$p$，第$k$次实验事件才发生的概率。

概率函数满足：

$$p(k)=p(1-p)^{k-1}$$

称做参数为$p$的几何分布。

随机变量$X$表示第一次朝上时掷了多少次。可以通过递归的思路列出方程求解$E(X)$

$$E[X]=p+(1-p)(E[X]+1)$$

解的期望为 $E[X]=\frac{1}{p}$。$X$ 的方差 $\mathrm{var}[X]=\frac{1-p}{p^2}$。

# 3.3 代 数 学

## 3.3.1 矩阵

**【基本概念】**

一个$n\times m$的矩阵是$n$行$m$列的矩形阵列，一般由数组成，下面是一个$2\times3$的矩阵。

$$\begin{pmatrix}1&2&3\\4&5&6\end{pmatrix}$$

矩阵 $\boldsymbol{A}$ 的第$i$行，第$j$列，通常记为$A_{i,j}$，上面例子中$A_{1,2}=2$。

单位矩阵$I \stackrel{\text{def}}{=} \begin{pmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{pmatrix}$，也就是对角线上为 1，其他都为 0 的矩阵。

**矩阵加（减）法：**

矩阵的加法定义为每行每列的元素分别相加（减），即

$$\boldsymbol{C} = \boldsymbol{A} \pm \boldsymbol{B} = \begin{pmatrix} A_{1,1} \pm B_{1,1} & \dots & A_{1,m} \pm B_{1,m} \\ \vdots & \ddots & \vdots \\ A_{n,1} \pm B_{n,1} & \dots & A_{n,m} \pm B_{n,m} \end{pmatrix}$$

**矩阵乘法：**

当一个矩阵的列数等于另一个的行数时，可定义该两个矩阵的乘积。

设$\boldsymbol{A}$是$n \times m$的矩阵，$\boldsymbol{B}$是$m \times p$的矩阵，他们的乘积$C$是一个$n \times p$的矩阵。其中，

$$C_{i,j} = \sum_{k=1}^{m} A_{i,k} \times B_{k,j}$$

**矩阵的幂：**

$$\boldsymbol{A}^0 = I$$

$$\boldsymbol{A}^n = \boldsymbol{A}^{n-1} \times \boldsymbol{A}\ (n > 0)$$

**矩阵的转置：**

$n \times m$的矩阵$\boldsymbol{A}$的转置$\boldsymbol{A}^{\mathrm{T}}$是个$m \times n$的矩阵，可以通过$\boldsymbol{A}$交换行列得到，即$\boldsymbol{A}^{\mathrm{T}}_{i,j} = \boldsymbol{A}_{j,i}$。

**矩阵的逆：**

只有$n \times n$的矩阵可能存在逆，矩阵$\boldsymbol{A}$的逆$\boldsymbol{B}$满足

$$\boldsymbol{A} \times \boldsymbol{B} = \boldsymbol{B} \times \boldsymbol{A} = I$$

如果逆存在，则$\boldsymbol{B}$是唯一的，一般记做$\boldsymbol{A}^{-1}$。

**【性质】**

1. 矩阵乘法满足分配率、结合律，不一定满足交换率。
2. 矩阵加法满足交换率和结合律。

**【算法】**

矩阵满足结合律，所以在求矩阵的幂的时候可以使用快速幂加速。

**【用途】**

递推可以表示成向量乘以矩阵，然后用快速幂优化，比如求 Fibonacci 数列。

Fibonacci 数列为，$a_0 = 1, a_1 = 1, a_n = a_{n-1} + a_{n-2}$

每次递推可以看为$\begin{pmatrix} a_n \\ a_{n+1} \end{pmatrix} = \begin{pmatrix} a_n \\ a_{n-1} + a_n \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \times \begin{pmatrix} a_{n-1} \\ a_n \end{pmatrix}$，令$\boldsymbol{A} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$，求数列第$n$项，等价于求$A^n \times \begin{pmatrix} 1 \\ 1 \end{pmatrix}$。该式可以用快速幂来加速。

### 3.3.2　行列式

**【基本概念】**

行列式是一个定义域为$n \times n$的矩阵，值域为一个标量的函数，通常记为$\det(A)$或者$|A|$。

行列式也可以表示为$n$维广义欧几里得空间中的有向体积。

一个$n \times n$的行列式定义为

$$\det(A) \overset{\text{def}}{=} \sum_{\sigma \in S_n} (-1)^{n(\sigma)} \prod_{i=1}^{n} a_{i,\sigma(i)}$$

其中$S_n$表示集合$\{1,2,\cdots,n\}$上的置换的全体，$n(\sigma)$是对于一个置换中逆序对的个数，逆序对的定义为满足$1 \leqslant i < j \leqslant n$且$\sigma(i) > \sigma(j)$的$(i,j)$。

比如一个三阶行列式：

$$\begin{vmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{vmatrix} = a_{1,1}a_{2,2}a_{3,3} + a_{1,2}a_{2,3}a_{3,1} + a_{1,3}a_{2,1}a_{3,2} - a_{1,3}a_{2,2}a_{3,1} - a_{1,1}a_{2,3}a_{3,2} - a_{1,2}a_{2,1}a_{3,3}$$

**【性质】**

1. 若矩阵中一行或一列全为 0，那么该矩阵行列式为 0。

2. 若矩阵中某一行有公因子$k$，那么可以提出$k$，比如：

$$\boldsymbol{D} = \begin{vmatrix} ka_{1,1} & \cdots & ka_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{vmatrix} = k \begin{vmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{vmatrix} = kD_1$$

3. 在矩阵中某一行可拆分成两个数和，那么该行列式可拆分成两个行列式相加。

$$\boldsymbol{D} = \begin{vmatrix} a_{1,1} + b_{1,1} & \cdots & a_{1,n} + b_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{vmatrix} = \begin{vmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{vmatrix} + \begin{vmatrix} b_{1,1} & \cdots & b_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{vmatrix}$$

4. 交换矩阵的两行，行列式取反。

$$\boldsymbol{D} = \begin{vmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{vmatrix} = - \begin{vmatrix} a_{n,1} & \cdots & a_{n,n} \\ \vdots & \ddots & \vdots \\ a_{1,1} & \cdots & a_{1,n} \end{vmatrix}$$

5. 将一行的$k$倍加到另一行上，行列式不变。

$$\boldsymbol{D} = \begin{vmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{vmatrix} = \begin{vmatrix} a_{1,1} + ka_{n,1} & \cdots & a_{1,n} + ka_{n,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{vmatrix}$$

6. 转置，行列式不变，$|\boldsymbol{A}| = |\boldsymbol{A}^{\mathrm{T}}|$

7. 由上述性质可以得出，当矩阵是上三角矩阵或者是下三角矩阵时，行列式的值等于对角线的乘积。

**【算法】**

一个常见的求解矩阵行列式的算法是利用性质 5 对矩阵做行变换，将矩阵转化成为上三角矩阵，再利用性质 7 得出行列式的值。

例如要求以下矩阵的行列式：

$$\begin{vmatrix}1 & 1 & 1\\1 & 3 & 2\\1 & 5 & 6\end{vmatrix}$$

通过行变换可以将矩阵消成上三角阵：

$$\begin{vmatrix}1 & 1 & 1\\0 & 2 & 1\\0 & 0 & 3\end{vmatrix}$$

由性质 7 可以知道，矩阵的行列式为$1 \times 2 \times 3 = 6$。

### 3.3.3 解线性方程组

**【基本概念】**

解线性方程组即求解方程组

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_1 + \cdots + a_{1,n}x_n = b_1 \\ \vdots \\ a_{m,1}x_1 + a_{m,2}x_1 + \cdots + a_{m,n}x_n = b_m \end{cases}$$

也可以表示为$\boldsymbol{Ax} = \boldsymbol{b}$，其中 $\boldsymbol{A}$ 是$m \times n$的矩阵，$\boldsymbol{x}$是 $n$ 维列向量，$\boldsymbol{b}$ 是 $m$ 维列向量，即

$$\begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$$

**【算法】**

1. 高斯消元

矩阵的初等变换：

（1）行初等变换：

a. 互换矩阵的两行。

b. 用非零的常数乘矩阵的某一行。

c. 某行的$k$倍加到另一行。

（2）列初等变换：

a. 互换矩阵的两列。

b. 用非零的常数乘矩阵的某一列。

c. 某行的$k$倍加到另一列。

对于上述的方程组，可以用一个增广矩阵表示出来：