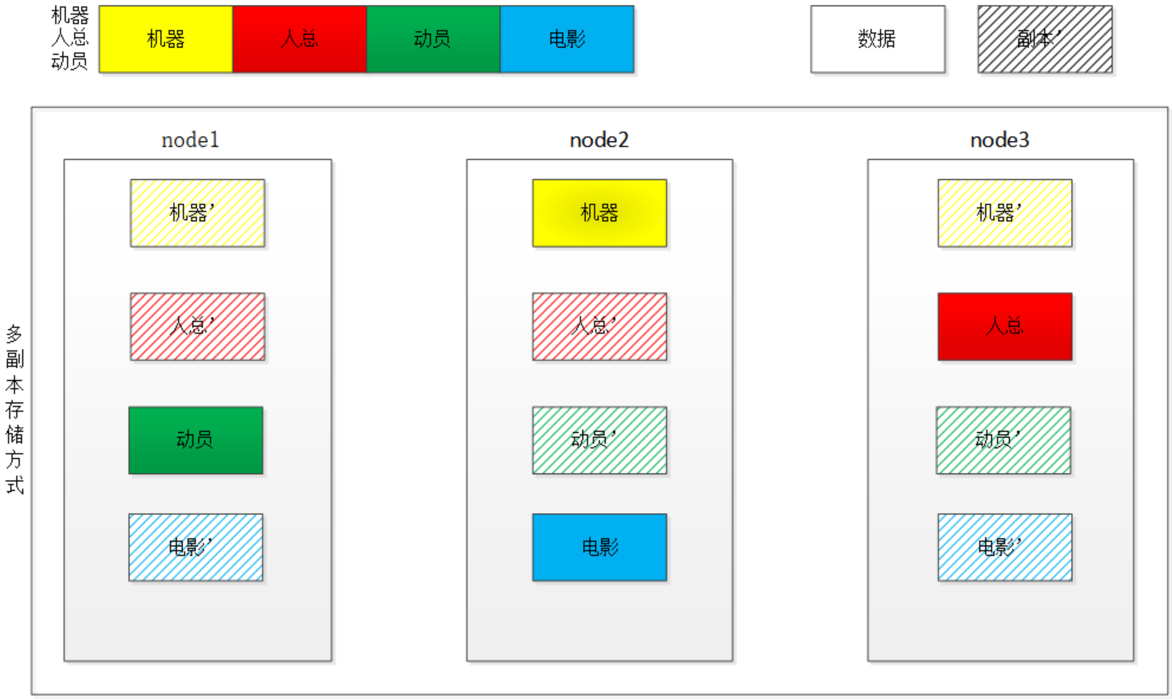
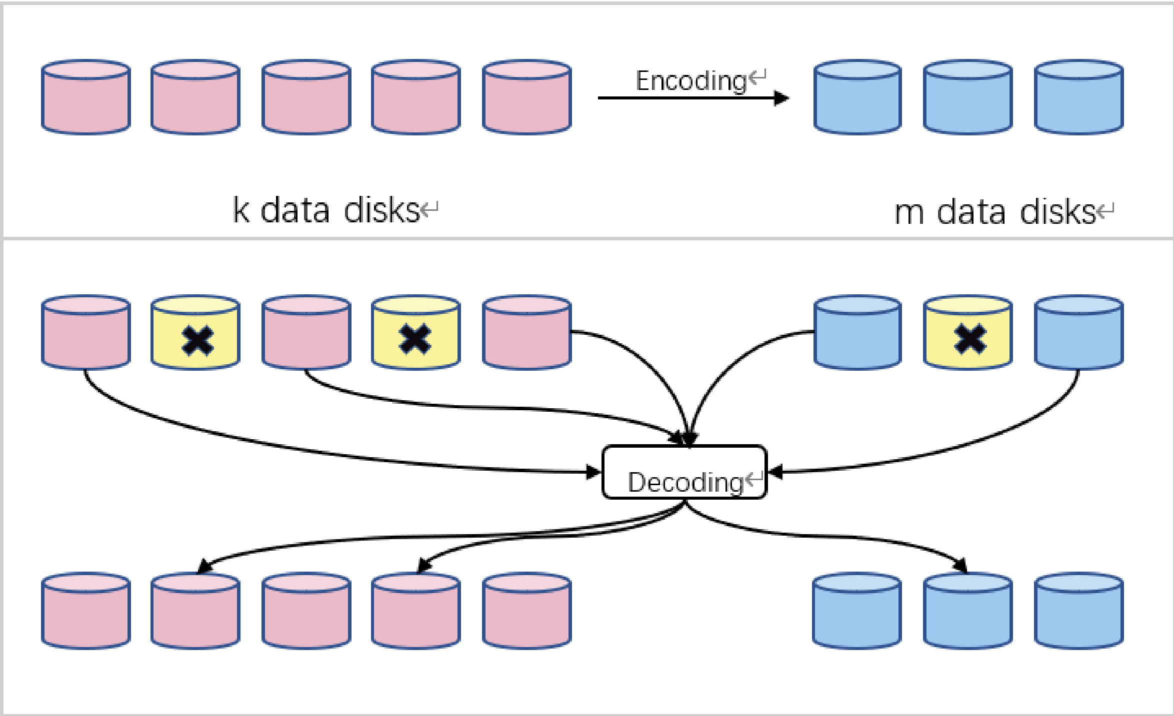
背景

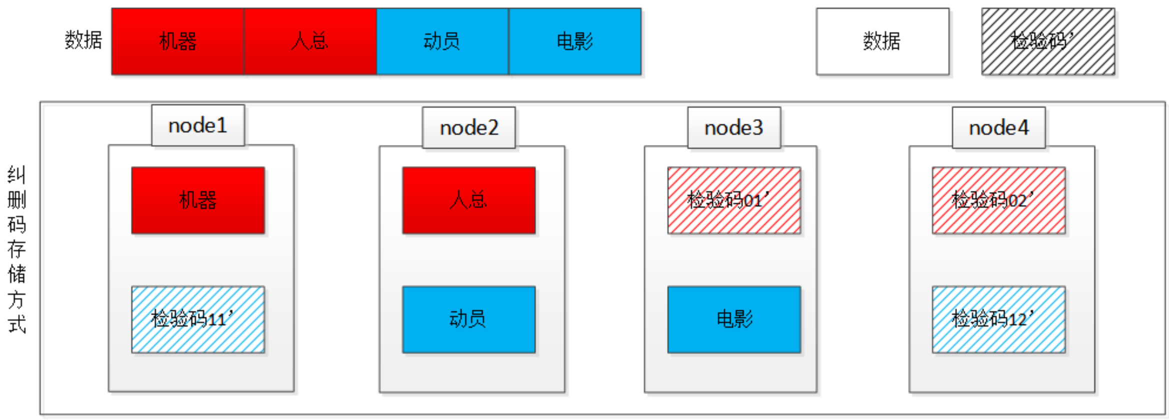
- 一般副本的数据存储方式：容错，但是导致空间利用率不高



- 纠删码：同样容错率，空间利用更高
  - 最早应用：通信领域
  - 约束条件：RS编码的数据节点个数n、校验节点个数m和伽罗瓦域GF(2^W)的位宽w三者之间满足:$2^W >= n + m$
  - 需要CPU参与更多计算



上图来源



伽罗瓦域

实数域求解方程

- 形如3<d> + 1<p>的冗余策略

> $d_0 + d_1 + d_2 = p_0$

eg: 数据块以4bit 为计算单元； $d_0 = 3$， $d_1 = 2$， $d_2 = 1$， $p_0 = 6$

任由丢失一个$d_i(i=0,1,2)$，可以通过四则运算计算出d_i

- 形如3<d> + 2<p>的冗余策略

> $d_0 + d_1 + d_2 = p_0 + 0*p_1 = p_0$
>
> $d_0 + 2d_1 + 4d_2 = 0*p_0 + *p_1 = p_1$

eg: 数据块以4bit 为计算单元； $d_0 = 3$，$d_1 = 2$，$d_2 = 1$，$p_0 = 6$，$p_1 = 11$

任由丢失一个或两个$d_i(i=0,1,2)$，可以通过四则运算计算出d_i

将代数方程系数写成矩阵的方式：

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \end{pmatrix} * \begin{pmatrix} D0 \\ D1 \\ D2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} P0 \\ P1 \end{pmatrix}$$

任意子方阵都是可逆：子方阵可以通过高斯消元单位化

推广：是否存在这样一个矩阵，任意的子方正都可逆，范德蒙德矩阵满足该特征

$$\begin{gathered} y_1 = d_1 + d_2 \cdot 1 + d_3 \cdot 1^2 + .. + d_k \cdot 1^{k-1} \\ y_2 = d_1 + d_2 \cdot 2 + d_3 \cdot 2^2 + .. + d_k \cdot 2^{k-1} \\ y_3 = d_1 + d_2 \cdot 3 + d_3 \cdot 3^2 + .. + d_k \cdot 3^{k-1} \\ \cdots \end{gathered}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \cdots \\ y_m \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1^2 & \cdots & 1^{k-1} \\ 1 & 2 & 2^2 & \cdots & 2^{k-1} \\ 1 & 3 & 3^2 & \cdots & 3^{k-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & m & m^2 & \cdots & m^{k-1} \end{bmatrix} \times \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \cdots \\ d_k \end{bmatrix}$$

缺陷：实数域四则运算运算的值，定义的位宽可能无法表示；是否存在一个集合，集合中按照定义的运算规则，运算结果也在这个封闭的集合中，伽罗瓦域横空出世

伽罗瓦域GF(p)

概念

首先需要明确群、环、域的定义：

- A1 加法封闭性：如果 a 和 b 属于 G，则 a + b 也属于G；
- A2 加法结合律：对 G 中的任意元素 a, b, c， a + (b + c) = (a + b) + c；
- A3 加法单位元：G 中存在一个元素0，使得对于G中的任意元素a，有 a + 0 = a；
- A4 加法逆元：对于 G 中的任意元素 a， G 中一定存在一个元素 a， 使得 a + (-a) = 0；
- A5 加法交换律：对于 G 中任意元素 a 和 b， 有 a + b = b + a；

- M1 乘法封闭性：如果 a 和 b 属于 G，则 a*b 也属于G

- M2 乘法结合律：对于 G 中任意元素 a, b, c ， 有 a * (b * c) = (a * b) * c
- M3 乘法分配律：对于 G 中的任意元素 a, b, c ， 有 a * (b + c) = a*b + a*c
- M4 乘法交换律：对于 G 中任意元素 a, b ， 有 a * b = b * a
- M5 乘法单位元：对于 G 中任意元素 a ， 在 G 中存在一个元素 a ， 使得 a * 1 = a
- M6 无零因子：对于 G 中的元素 a, b ， 若 a*b = 0 ， 则必有 a = 0 或 b = 0
- M7 乘法逆元：如果 $a \in G$，且 $a \neq 0$，则 G 中存在一个元素 a^{-1}，使得 $a * a^{-1} = a^{-1} * a = 1$

上述性质中，

- 满足 A1-A4 的，称为群 (group).
- 满足 A1-A5 的，称为交换群 / 阿贝尔群 (commutative group, Abelian group).
- 满足 A1-M3 的，称为环 (ring).
- 满足 A1-M4 的，称为交换环 (commutative ring).
- 满足 A1-M6 的，称为整环 (integral domain).
- 满足 A1-M7 的，称为域 (field).

可见，群支持一种运算；环支持两种运算（乘法、加法）并拥有分配律；域支持加、减、乘、除。让我们来看几个例子：

比如有理数集合就是一个域：

> 有理数在加法和乘法运算下构成一个域，0是加法单位元，1是乘法单位元，不包含0的有理数在乘法运算下成群

比如无理数集合就不是一个域：

> 无理数在加法和乘法下不能构成域，这是因为无理数之和可能是有理数，不满足封闭性。

以上的的域都是无限域，是否存在一种有限域也满足域的基本性质了

这个域就是伽罗瓦域（Galois field），记作GF(p)，其中p是质数。与我们生活中使用的实数域不同的是，有限域中的元素是有限的，每个有限域GF(p)中都有p个元素，分别是0到p-1，该域称为素域，例如GF(7)：{0,1,2,3,4,5,6}；运算规则：实数域的四则运算，然后mod p。

为什么p必须是质数？？？

eg GF(10):

> 其加法和乘法单位元分别是0和1。加法没有问题，所有元素都有加法逆元，
>
> 但对于乘法来说，比如元素2，它就没有乘法逆元。因为找不到一个数a，使得2*a mod 10等于1。这时，就不能进行除于2运算了

应用: 前面的例子在GF(17)计算：

$$\left(\begin{array}{ccc|cc} 1 & 1 & 1 & 1 & 0 \\ 1 & 2 & 4 & 0 & 1 \end{array}\right)$$

$d_0 = 8$，$d_1 = 7$，$d_2 = 12$，

GF(17)下运算结果 $p_0 = 10$，$p_1 = 2$

失效d_0，d_1

在GF(17)单位化第一列和第二列构成的方阵

$$\left(\begin{array}{ccc|cc} 1 & 1 & 1 & 1 & 0 \\ 1 & 2 & 4 & 0 & 1 \end{array}\right) => \left(\begin{array}{ccc|cc} 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 3 & 16 & 1 \end{array}\right) => \left(\begin{array}{ccc|cc} 1 & 0 & 15 & 2 & 16 \\ 0 & 1 & 3 & 16 & 1 \end{array}\right)$$

$d_0 =(2 * p_0 + 16 * p_1 - 15 * d_2) \bmod(17) = 8$

$d_1 =(16 * p_0 + p_1 - 3 * d_2) \bmod(17) = 7$

推广

在很多情况下，GF(p)是不够用的，比如在图像处理中，像素的色彩空间一般为RGB（红绿蓝）色彩空间，每种颜色的取值范围都是0~255。那么我们就需要另外一种有限域，它应该有2^8个元素。可是如果我们仍然用之前的加法运算与乘法运算的话，这是不能构成一个域的，所以我们需要引进新的加法运算与乘法运算。

对于任何正整数w，可以将素域GF(p)扩展成有p^w个元素的域，称它为GF(p)的扩域，并以GF(p^w)表示。为了保证单位元性质，GF(p^w)上的加法运算和乘法运算，不再使用一般的加法和乘法，而是使用多项式运算。

重点讨论GF(2^w)：

比如w为8，表示为位宽，在GF(2^8)中的元素都可以使用8个位宽表示

> 多项式的系数限定在GF(2)中{0， 1}
>
> 合并同类项时，系数们进行异或操作
>
> 无所谓的减法(减法就等于加法)，或者负系数；$x^4 - x^4$就等于$x^4 + x^4$。$-x^3$就是x^3

eg:多项式运算:

$$f(x) = x^6 + x^4 + x^2 + x + 1$$

$$g(x) = x^7 + x + 1$$

$$f(x) + g(x) = f(x) - g(x) = x^7 + x^6 + x^4 + x^2 + (1 \oplus 1) * x + (1 \oplus 1) * 1$$

$$= x^7 + x^6 + x^4 + x^2$$

$$f(x) * g(x) = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^1 + 1$$

$$g(x) / f(x) = x^6 + x^4 + x^2 + x + 1 \overset{\displaystyle x}{\overline{\big)\, x^7 + x + 1}}$$

$$\begin{array}{r} x^7 + x^5 + x^3 + x^2 + x \\ \hline x^5 + x^3 + x^2 + 1 \end{array}$$

素多项式

对于多项式也类似素数，有素多项式。其定义和素数类似，素多项式不能表示为其他两个多项式相乘的乘积，在素多项式的基础上也可以构造一个有限域

GF(2^3)的多项式有：0， 1，x，x + 1，x^2，x^2 + 1，x^2 + x，x^2 + x + 1。

GF(2^3)的其中一个素多项式为：x^3 + x + 1

上面8个多项式进行四则运算后 mod(x^3 + x + 1)的结果都在这个8个多项式的一个，并且每一个多项式都是有加法和乘法逆元的(0除外)，这个8个多项式构成了一个伽罗瓦域。注意，这些逆元都是和素多项式相关的，同一个多项式，取不同的素多项式，就有不同的逆元多项式。

GF(2^3) 模加	**0**	**1**	**x**	**x + 1**	**x^2**	**x^2 + 1**	**x^2 + x**	**x^2 + x + 1**
0	0	1	x	x + 1	x^2	x^2 + 1	x^2 + x	x^2 + x + 1
1	1	0	x + 1	x	x^2 + 1	x^2	x^2 + x + 1	x^2 + x
x	x	x + 1	0	1	x^2 + x	x^2 + x + 1	x^2	x^2 + 1
x + 1	x + 1	x	1	0	x^2 + x + 1	x^2 + x	x^2 + 1	x^2
x^2	x^2	x^2 + 1	x^2 + x	x^2 + x + 1	0	1	x	x + 1

$GF(2^3)$ 模加	0	1	x	x + 1	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
$x^2 + 1$	$x^2 + 1$	x^2	$x^2 + x + 1$	$x^2 + x$	1	0	x + 1	x
$x^2 + x$	$x^2 + x$	$x^2 + x + 1$	x^2	$x^2 + 1$	x	x + 1	0	1
$x^2 + x + 1$	$x^2 + x + 1$	$x^2 + x$	$x^2 + 1$	x^2	x + 1	x	1	0

$GF(2^3)$ 模乘	0	1	x	x + 1	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
0	0	0	0	0	0	0	0	0
1	0	1	x	x + 1	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
x	0	x	x^2	$x^2 + x$	x + 1	1	$x^2 + x + 1$	$x^2 + 1$
x + 1	0	x + 1	$x^2 + x$	$x^2 + 1$	$x^2 + x + 1$	x^2	1	x
x^2	0	x^2	x + 1	$x^2 + x + 1$	$x^2 + x$	x	$x^2 + 1$	1
$x^2 + 1$	0	$x^2 + 1$	1	x^2	x	$x^2 + x + 1$	x + 1	$x^2 + x$
$x^2 + x$	0	$x^2 + x$	$x^2 + x + 1$	1	$x^2 + 1$	x + 1	x	x^2
$x^2 + x + 1$	0	$x^2 + x + 1$	$x^2 + 1$	x	1	$x^2 + x$	x^2	x + 1

重点来了：到现在为止只看了多项式构成的一个$GF(2^w)$，有什么用了？？？？可以发现每个多项式的系数可构成一个数，而这些数的集合可以用w位宽进行表示。

eg: $GF(2^3)$

多项式	bit位	值
0	000	0

多项式	bit位	值
1	001	1
x	010	2
x + 1	011	3
x^2	100	4
$x^2 + 1$	101	5
$x^2 + x$	110	6
$x^2 + x + 1$	111	7

部分 GF（2^w）域经常使用的素多项式如下：

$$
\begin{aligned}
w=4: &\quad x^4+x+1 \\
w=8: &\quad x^8+x^4+x^3+x^2+1 \\
w=16: &\quad x^{16}+x^{12}+x^3+x+1 \\
w=32: &\quad x^{32}+x^{22}+x^2+x+1
\end{aligned}
$$

生成元

虽然可以用多项式的系数构成一个有限域的集合，但是按照多项式定义的加减乘除计算依然是繁琐的，生成元可以简化多项式的运算

> 生成元定义:如果一个群G里的元素都是某一个元素g的幂，则称G为循环群，g称为G的一个生成元。由g生成的的循环群记为(g)。
>
> 无限循环群可表示为：
>
> {..., g^{-2}, g^{-1}, g^0, g^1, g^2, ...}，其中g^0 = e
>
> 有限n阶循环群可以表示为：
>
> {g^0, g^1, g^2, ..., g^{n-1}}，其中g^0 = e
>
> eg:
>
> 正整数加法群Z是一个循环群。1是生成元，每一个元素都是1的“幂”。这里再次说明讨论的群里“乘法”是抽象的，只代表一种代数运算。在正整数加群中，“乘法”就是普通加法，那么“幂”就是一个元素的连加，例如
>
> 1^m = m = 1+....+1 <m个1相加>，且规定 0 = 1^0，即0个1相加

> 有限域的生成元：对于阶数为q的有限域，其生成元是一个元素g，该元素的前q-1个幂构成了域F的所有非零元素，即域F的元素为
>
> $0, g^0, g^1, ..., g^{q-2}$
>
> 性质：考虑有多项式f(x)定义的域F，如果F内的一个元素g，满足f(g) = 0，则称g为多项式f(x)的根，可证明一个素多项式的根g是这个素多项式定义的有限域的生成元

例子：一个素多项式$x^3 + x + 1$定义的有限域GF(2^3)，生成元是g，==> $g^3 + g + 1 = 0$ => $g^3 = g + 1$

生成元	多项式	二进制	数值	过程
0	0	000	0	
g^0	1	001	1	
g^1	x	010	2	
g^2	x^2	100	4	
g^3	$x + 1$	011	3	$g^3 = g + 1$
g^4	$x^2 + x$	110	6	$g^4 = g * g^3 = g * (g + 1) = g^2 + g$
g^5	$x^2 + x + 1$	111	7	$g * g^4 = g^3 + g^2 = g^2 + g + 1$
g^6	$x^2 + 1$	101	5	$g * g^5 = g^3 + g^2 + g = g + 1 + g^2 + g = g^2 + 1$
g^7	1	001	1	$g * g^6 = g^3 + g = g + 1 + g = 1$

多项式的乘法可以变为幂级数的乘法，简化运算，但是还是很麻烦，可以查表运算

i	0	1	2	3	4	5	6	7
生成元幂级数 gfilog[i]	1	2	4	3	6	7	5	-
二进制值对应的幂级数 gflog[i]	-	0	1	3	2	6	4	5

那么加法了？？？就是二进制数的异或运算

但是用的更多的GF(2^8)，GF(2^{16})，GF(2^{32})，可以通过将它们映射表和反向映射表构造好，运算过程中使用查表计算

到现在为止大部分数学工具都已经差不多了，最后一个问题是：虽然可以通过查表简化幂级数的运算，但是有没有更加方便的接工程应用的方法？？？

答案是：二进制矩阵

二进制矩阵

二进制矩阵运算规则

同构：伽罗瓦发现，有些表象不同的群之间，其实质是完全相同的。这样的群称为是“同构”的，也就是说，这样的群在结构和性质上都完全相同，只有表面符号上存在差别。同构的群在去掉表象之后，可以认为是同一个群。

群同构的严格定义是：存在两个群A、B之间的一个双射（即一一对应的映射）ϕ:A→B，满足ϕ(a*b)=ϕ(a)×ϕ(b)，其中a、b∈A，ϕ(a)、ϕ(b)和ϕ(a*b)∈B，*和×分别是群A和B的“乘法”。

类似的，域也有同构的情况。简单说两个域的同构定义为：两个域上的“加法”群同构，并且去除“加法”单位元之后的两个域上的“乘法”群也要同构。

p(x)是$GF(2^W)$的素多项式，$GF(2^W)$与$\phi(GF(2^W))$同构，

映射关系：对于任意元素$f_i \in GF(2^W)$，o(f)一个w*w的二进制数矩阵，其中它的第i列为x^i * f mod p(x)系数向量

按照定义构造 $\phi(GF(2^3))$的二进制矩阵

$$
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\begin{matrix} 0 & 0 & 0 \end{matrix} & \begin{matrix} g^0 & g^1 & g^2 \end{matrix} & \begin{matrix} g^1 & g^2 & g^3 \end{matrix} & \begin{matrix} g^3 & g^4 & g^5 \end{matrix} & \begin{matrix} g^2 & g^3 & g^4 \end{matrix} & \begin{matrix} g^6 & g^0 & g^1 \end{matrix} & \begin{matrix} g^4 & g^5 & g^6 \end{matrix} & \begin{matrix} g^5 & g^6 & g^0 \end{matrix} \\
\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} & \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} & \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}
\end{array}
$$

$$
4+5 = 4 \oplus 5 = x^2 + (x^2+1) = 1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}
$$

$$
\begin{aligned}
4*5 &= x^2 * (x^2+1) = (x^4 + x^2) \bmod (x^3 + x + 1) = x = 2 \\
&= g^2 * g^6 = g^{(2+6)\bmod 7} = g = 2 \\
&= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} = \begin{matrix} (0*1)\oplus(1*0)\oplus(0*1) & (0*1)\oplus(1*0)\oplus(0*0) & (0*0)\oplus(1*1)\oplus(0*0) \\ (0*1)\oplus(1*0)\oplus(1*1) & (0*1)\oplus(1*0)\oplus(1*0) & (0*0)\oplus(1*1)\oplus(1*0) \\ (1*1)\oplus(0*0)\oplus(1*1) & (1*1)\oplus(0*0)\oplus(1*0) & (1*0)\oplus(0*1)\oplus(1*0) \end{matrix} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}
\end{aligned}
$$

问题来了：构造二进制矩阵的目的是什么了？？？

从二进制的运算的规则来看都是bit的运算，一个bit位可以推广到8bit(1B)的运算，我们步子再拉大一点，4B？？，8B？？？ 甚至可以推广到数据块之间的运算

讨论$GF(2^3)$主要是运算简单，为了一步步的引出二进制矩阵的位运算，重点需要讨论的是$GF(2^4)$同构的二进制矩阵的运算规则，4是为了让参加计算的数据便于分片

$\phi(GF(2^4))$的讨论

$GF(2^4)$ 素多项式 $x^4 + x + 1$，生成元是g，==> $g^4 + g + 1 = 0$ => $g^4 = g + 1$

生成元	多项式	二进制	数值	过程
0	0	0000	0	
g^0	1	0001	1	
g^1	x	0010	2	
g^2	x^2	0100	4	
g^3	x^3	1000	8	
g^4	x + 1	0011	3	$g^4 = g + 1$
g^5	$x^2 + x$	0110	6	$g * g^4 = g *(g + 1) = g^2 + g$
g^6	$x^{3+x}2$	1100	12	$g * g^5 = g^3 + g^2$
g^7	x^3+x+1	1011	11	$g * g^6 = g^4 + g^3 = g^3 + g + 1$
g^8	x^2+1	0101	5	$g *g^7 = g * (g^3 + g + 1) = g^4 + g^2 + g = g + g + g^2 + 1 = g^2 + 1$
g^9	x^3+x	1010	10	$g * g^8 = g^3 + g$
g^{10}	x^2+x+1	0111	7	$g * g^9 = g^4 + g^2 = g^2 + g + 1$
g^{11}	$x^{3+x}2+x$	1110	14	$g * g^9 = g^3 + g^2 + g$
g^{12}	$x^{3+x}2+x+1$	1111	15	$g * g^{11} = g * (g^3 + g^2 + g) = g^4 + g^3 + g^2 = g^3 + g^2 + g + 1$

生成元	多项式	二进制	数值	过程
g^{13}	$x^{3+x}2+1$	1101	13	$g * g^{12} = g^4 + g^3 + g^2 + g = g^3 + g^2 + 1$
g^{14}	x^3+1	1001	9	$g * g^{13} = g^4 + g^3 + g = g^3 + 1$
g^{15}	1	0001	1	$g * g^{14} = g^4 + g = 1$

$$
\begin{array}{c} 0 \\ \hline \begin{array}{cccc} 0 & 0 & 0 & 0 \end{array} \\ \hline \begin{pmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{pmatrix} \end{array}
\quad
\begin{array}{c} 1 \\ \hline \begin{array}{cccc} g^0 & g^1 & g^2 & g^3 \end{array} \\ \hline \begin{pmatrix} 1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1 \end{pmatrix} \end{array}
\quad
\begin{array}{c} 2 \\ \hline \begin{array}{cccc} g^1 & g^2 & g^3 & g^4 \end{array} \\ \hline \begin{pmatrix} 0&0&0&1\\1&0&0&1\\0&1&0&0\\0&0&1&0 \end{pmatrix} \end{array}
\quad
\begin{array}{c} 3 \\ \hline \begin{array}{cccc} g^4 & g^5 & g^6 & g^7 \end{array} \\ \hline \begin{pmatrix} 1&0&0&1\\1&1&0&1\\0&1&1&0\\0&0&1&1 \end{pmatrix} \end{array}
\quad
\begin{array}{c} 4 \\ \hline \begin{array}{cccc} g^2 & g^3 & g^4 & g^5 \end{array} \\ \hline \begin{pmatrix} 0&0&1&0\\0&0&1&1\\1&0&0&1\\0&1&0&0 \end{pmatrix} \end{array}
\quad
\begin{array}{c} 5 \\ \hline \begin{array}{cccc} g^8 & g^9 & g^{10} & g^{11} \end{array} \\ \hline \begin{pmatrix} 1&0&1&0\\0&1&1&1\\1&0&1&1\\0&1&0&1 \end{pmatrix} \end{array}
\quad
\begin{array}{c} 6 \\ \hline \begin{array}{cccc} g^5 & g^6 & g^7 & g^8 \end{array} \\ \hline \begin{pmatrix} 0&0&1&1\\1&0&1&0\\1&1&0&1\\0&1&1&0 \end{pmatrix} \end{array}
\quad
\begin{array}{c} 7 \\ \hline \begin{array}{cccc} g^{10} & g^{11} & g^{12} & g^{13} \end{array} \\ \hline \begin{pmatrix} 1&0&1&1\\1&1&1&0\\1&1&1&1\\0&1&1&1 \end{pmatrix} \end{array}
$$

$$
\begin{array}{c} 8 \\ \hline \begin{array}{cccc} g^3 & g^4 & g^5 & g^6 \end{array} \\ \hline \begin{pmatrix} 0&1&0&0\\0&1&1&0\\0&0&1&1\\1&0&0&1 \end{pmatrix} \end{array}
\quad
\begin{array}{c} 9 \\ \hline \begin{array}{cccc} g^{14} & g^0 & g^1 & g^2 \end{array} \\ \hline \begin{pmatrix} 1&1&0&0\\0&0&1&0\\0&0&0&1\\1&0&0&0 \end{pmatrix} \end{array}
\quad
\begin{array}{c} 10 \\ \hline \begin{array}{cccc} g^9 & g^{10} & g^{11} & g^{12} \end{array} \\ \hline \begin{pmatrix} 0&1&0&1\\1&1&1&1\\0&1&1&1\\1&0&1&1 \end{pmatrix} \end{array}
\quad
\begin{array}{c} 11 \\ \hline \begin{array}{cccc} g^7 & g^8 & g^9 & g^{10} \end{array} \\ \hline \begin{pmatrix} 1&1&0&1\\1&0&1&1\\0&1&0&1\\1&0&1&0 \end{pmatrix} \end{array}
\quad
\begin{array}{c} 12 \\ \hline \begin{array}{cccc} g^6 & g^7 & g^8 & g^9 \end{array} \\ \hline \begin{pmatrix} 0&1&1&0\\0&1&0&1\\1&0&1&0\\1&1&0&1 \end{pmatrix} \end{array}
\quad
\begin{array}{c} 13 \\ \hline \begin{array}{cccc} g^{13} & g^{14} & g^0 & g^1 \end{array} \\ \hline \begin{pmatrix} 1&1&1&0\\0&0&0&1\\1&0&0&0\\1&1&0&0 \end{pmatrix} \end{array}
\quad
\begin{array}{c} 14 \\ \hline \begin{array}{cccc} g^{11} & g^{12} & g^{13} & g^{14} \end{array} \\ \hline \begin{pmatrix} 0&1&1&1\\1&1&0&0\\1&1&1&0\\1&1&1&1 \end{pmatrix} \end{array}
\quad
\begin{array}{c} 15 \\ \hline \begin{array}{cccc} g^{12} & g^{13} & g^{14} & g^0 \end{array} \\ \hline \begin{pmatrix} 1&1&1&1\\1&0&0&0\\1&1&0&0\\1&1&1&0 \end{pmatrix} \end{array}
$$

通过范德蒙德矩阵获取的RS二进制矩阵，规格RS(m, n, w) = R(4, 2, 4)，数据块4，校验块2，数据块宽度4

约束条件 $m + n <= 2^w$

失效节点条件 lost_n <= n

校验码生成矩阵：

$$
\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \end{pmatrix} * \begin{pmatrix} D0 \\ D1 \\ D2 \\ D3 \end{pmatrix} = \begin{pmatrix} P0 \\ P1 \end{pmatrix} = \begin{pmatrix} D0 \oplus D1 \oplus D2 \oplus D3 \\ D0 \oplus (2*D1) \oplus (4*D2) \oplus (8*D3) \end{pmatrix}
$$

二进制矩阵表达形式：D_{00}，D_{01}，D_{02}，D_{03}是数据条块D0的4个分片，其他也类似

$$
\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \end{pmatrix} * \begin{pmatrix} D0 \\ D1 \\ D2 \\ D3 \end{pmatrix} = \begin{pmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1
\end{pmatrix} * \begin{pmatrix} D_{00} \\ D_{01} \\ D_{02} \\ D_{03} \\ D_{10} \\ D_{11} \\ D_{12} \\ D_{13} \\ D_{20} \\ D_{21} \\ D_{22} \\ D_{23} \\ D_{30} \\ D_{31} \\ D_{32} \\ D_{33} \end{pmatrix}
$$

$$
= \begin{pmatrix} P_{00} \\ P_{01} \\ P_{02} \\ P_{03} \\ P_{10} \\ P_{11} \\ P_{12} \\ P_{13} \end{pmatrix} = \begin{pmatrix}
D_{00} \oplus D_{10} \oplus D_{20} \oplus D_{30} \\
D_{01} \oplus D_{11} \oplus D_{21} \oplus D_{31} \\
D_{02} \oplus D_{12} \oplus D_{22} \oplus D_{32} \\
D_{03} \oplus D_{13} \oplus D_{23} \oplus D_{33} \\
D_{01} \oplus D_{23} \oplus D_{22} \oplus D_{31} \\
D_{01} \oplus D_{10} \oplus D_{13} \oplus D_{22} \oplus D_{23} \oplus D_{31} \oplus D_{32} \\
D_{02} \oplus D_{11} \oplus D_{20} \oplus D_{23} \oplus D_{32} \oplus D_{33} \\
D_{03} \oplus D_{12} \oplus D_{21} \oplus D_{31} \oplus D_{33}
\end{pmatrix}
$$

通过二进制矩阵的计算方式，可以发现数据块的异或的计算量和二进制的矩阵的数值的"1"个数相关，1的个数称为二进制矩阵的重量

有没有一种矩阵，性质同范德蒙德矩阵一样，但是矩阵重量比范德蒙德要轻，这个矩阵是柯西矩阵

$$
\begin{pmatrix}
\frac{1}{x_0+y_0} & \frac{1}{x_0+y_1} & \cdots\cdots & \frac{1}{x_0+y_{m-1}} \\
\frac{1}{x_1+y_0} & \frac{1}{x_1+y_1} & \cdots\cdots & \frac{1}{x_1+y_{m-1}} \\
\cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\
\frac{1}{x_{n-2}+y_0} & \frac{1}{x_{n-2}+y_1} & \cdots\cdots & \frac{1}{x_{n-2}+y_{m-1}} \\
\frac{1}{x_{n-1}+y_0} & \frac{1}{x_{n-1}+y_0} & \cdots\cdots & \frac{1}{x_{n-1}+y_{m-1}}
\end{pmatrix}
$$

> 定义: 第i行，第j列的元素$a_{ij} = 1 / (x_i + y_j)$，$x_i + y_j \,!= 0$; $0 <= i <= n-1$, $0 <= j <= m-1$

> 性质: 任意子方阵是非奇异矩阵(AX=0有确定解) ==> 任意子方阵可逆 ==> 可以单位化

> 限制条件: 在GF(2^W)下的柯西矩阵,其中x_i,y_j都是伽罗瓦域中的元素，并且按照定义的运算规则进行运算

以下柯西矩阵是通过Jerasure库获取

```
/*编码矩阵*/
static const uint8 g_2_n_1_m[]  = {1,1};                                            /*2数据节点、1个校验节点*/
static const uint8 g_3_n_1_m[]  = {1,1,1};
static const uint8 g_4_n_1_m[]  = {1,1,1,1};
static const uint8 g_5_n_1_m[]  = {1,1,1,1,1};
static const uint8 g_6_n_1_m[]  = {1,1,1,1,1,1};
static const uint8 g_7_n_1_m[]  = {1,1,1,1,1,1,1};
static const uint8 g_8_n_1_m[]  = {1,1,1,1,1,1,1,1};
static const uint8 g_9_n_1_m[]  = {1,1,1,1,1,1,1,1,1};
static const uint8 g_10_n_1_m[] = {1,1,1,1,1,1,1,1,1,1};
static const uint8 g_11_n_1_m[] = {1,1,1,1,1,1,1,1,1,1,1};
static const uint8 g_12_n_1_m[] = {1,1,1,1,1,1,1,1,1,1,1,1};


static const uint8 g_3_n_2_m[]  = {1,1,1,                  1,2,9};      /*3数据节点、2个校验节点*/
static const uint8 g_4_n_2_m[]  = {1,1,1,1,                1,2,9,4};
static const uint8 g_5_n_2_m[]  = {1,1,1,1,1,              1,2,9,4,8};
static const uint8 g_6_n_2_m[]  = {1,1,1,1,1,1,            1,2,9,4,8,13,};
static const uint8 g_7_n_2_m[]  = {1,1,1,1,1,1,1,          1,2,9,4,8,13,3,};
static const uint8 g_8_n_2_m[]  = {1,1,1,1,1,1,1,1,        1,2,9,4,8,13,3,6};
static const uint8 g_9_n_2_m[]  = {1,1,1,1,1,1,1,1,1,      1,2,9,4,8,13,3,6,12,};
static const uint8 g_10_n_2_m[] = {1,1,1,1,1,1,1,1,1,1,    1,2,9,4,8,13,3,6,12,5};
static const uint8 g_11_n_2_m[] = {1,1,1,1,1,1,1,1,1,1,1,  1,2,9,4,8,13,3,6,12,5,11};
static const uint8 g_12_n_2_m[] = {1,1,1,1,1,1,1,1,1,1,1,1, 1,2,9,4,8,13,3,6,12,5,11,15};


static const uint8 g_4_n_3_m[]  = {1,1,1,1,                1,12,8,11,
12,9,1,6};          /*4数据节点、3个校验节点*/
static const uint8 g_5_n_3_m[]  = {1,1,1,1,1,              13,3,2,6,1,
12,9,1,6,3};
static const uint8 g_6_n_3_m[]  = {1,1,1,1,1,1,            9,6,4,12,2,1,
1,5,10,9,13,6};
static const uint8 g_7_n_3_m[]  = {1,1,1,1,1,1,1,          13,3,2,6,1,9,12,
1,5,10,9,13,6,8};
static const uint8 g_8_n_3_m[]  = {1,1,1,1,1,1,1,1,        9,6,4,12,2,1,11,13,
1,5,10,9,13,6,8,4};
static const uint8 g_9_n_3_m[]  = {1,1,1,1,1,1,1,1,1,      9,6,4,12,2,1,11,13,10,
1,5,10,9,13,6,8,4,3};
static const uint8 g_10_n_3_m[] = {1,1,1,1,1,1,1,1,1,1,    13,3,2,6,1,9,12,15,5,8,
1,5,10,9,13,6,8,4,3,12};
static const uint8 g_11_n_3_m[] = {1,1,1,1,1,1,1,1,1,1,1,  13,3,2,6,1,9,12,15,5,8,4,
1,5,10,9,13,6,8,4,3,12,15};
static const uint8 g_12_n_3_m[] = {1,1,1,1,1,1,1,1,1,1,1,1, 13,3,2,6,1,9,12,15,5,8,4,10,
1,5,10,9,13,6,8,4,3,12,15,2};
```

```
static const uint8 g_5_n_4_m[]  = {1,1,1,1,1,            8,1,3,9,13,
1,2,12,6,15,                    9,7,3,1,8};     /*5数据节点、4个校验节点*/
static const uint8 g_6_n_4_m[]  = {1,1,1,1,1,1,          8,1,3,9,13,6,
8,3,10,5,1,13,                  4,13,11,8,12,1};
static const uint8 g_7_n_4_m[]  = {1,1,1,1,1,1,1,        6,4,12,2,1,11,13,
12,11,15,14,8,2,1,              4,13,11,8,12,1,6};
static const uint8 g_8_n_4_m[]  = {1,1,1,1,1,1,1,1,      3,2,6,1,9,12,15,5,
6,12,14,7,4,1,9,2,              4,13,11,8,12,1,6,9};
static const uint8 g_9_n_4_m[]  = {1,1,1,1,1,1,1,1,1,    3,2,6,1,9,12,15,5,8,
8,3,10,5,1,13,15,9,2,           4,13,11,8,12,1,6,9,3};
static const uint8 g_10_n_4_m[] = {1,1,1,1,1,1,1,1,1,1,  8,1,3,9,13,6,14,11,4,2,
8,3,10,5,1,13,15,9,2,11,        8,9,5,3,11,2,12,1,6,13};
static const uint8 g_11_n_4_m[] = {1,1,1,1,1,1,1,1,1,1,1,  8,1,3,9,13,6,14,11,4,2,5,
6,12,14,7,4,1,9,2,8,10,13,      4,13,11,8,12,1,6,9,3,15,5};
static const uint8 g_12_n_4_m[] = {1,1,1,1,1,1,1,1,1,1,1,1, 2,13,4,15,14,8,10,6,1,9,12,5,
6,12,14,7,4,1,9,2,8,10,13,11,   4,13,11,8,12,1,6,9,3,15,5,2};
```

ϕ(GF(2^4))，元素的重量

值	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
重量	0	4	5	9	6	10	9	13	7	5	12	10	9	7	12	10

规格	范德蒙德矩阵重量	柯西矩阵重量
RS(3,2,4)	23 - 2*4 = 15	22 - 2*4 = 14
RS(4,2,4)	38 - 8 = 30	36 - 8 = 28
RS(5,2,4)	51 - 8 = 43	47 - 8 = 39
RS(6,2,4)	64 - 8 = 56	58 - 8 = 50
RS(4,3,4)	71 - 3*4 = 59	73 - 3*4 = 61
RS(5,3,4)	89 - 12 = 77	94 - 12 = 82
RS(8,3,4)	157 - 12 = 145	148 - 12 = 136

what??? 不是柯西矩阵的重量少于范德蒙德矩阵吗？？？可能是计算的矩阵有问题，可以确定的是柯西矩阵的编解码都没问题，GF(2^4)下范德蒙德需要再次求证编解码是否正确

解码 how to do

```
选择最优柯西矩阵:
 [1  1  1  1]
 [1  2  9  4]
 数据块D0,D1,D2,D3, 校验块P0, P1
```

```
        假设D0 D3 失效

  对|| 1 0 0 0  1 0 0 0     0 0 0 0  1 0 0 0     0 0 0 0  1 0 0 0     0 0 0 0
1 0 0 0     0 0 0 0  1 0 0 0     0 0 0 0  1 0 0 0     0 0 0 0  1 0 0 0     1 0
0 0  0 0 0 0
  角|| 0 1 0 0  0 1 0 0     0 0 0 0  0 1 0 0     0 0 0 0  0 1 0 0     0 0 0 0
0 1 0 0     0 0 0 0  0 1 0 0     0 0 0 0  0 1 0 0     0 0 0 0  0 1 0 0     0 1
0 0  0 0 0 0
  化|| 0 0 1 0  0 0 1 0     0 0 0 0  0 0 1 0     0 0 0 0  0 0 1 0     0 0 0 0
0 0 1 0     0 0 0 0  0 0 1 0     0 0 0 0  0 0 1 0     0 0 0 0  0 0 1 0     0 0
1 0  0 0 0 0
  行|| 0 0 0 1  0 0 0 1     0 0 0 0  0 0 0 1     0 0 0 0  0 0 0 1     0 0 0 0
0 0 0 1     0 0 0 0  0 0 0 1     0 0 0 0  0 0 0 1     0 0 0 0  0 0 0 1     0 0
0 1  0 0 0 0

       1 0 0 0  0 0 0 1     1 0 0 1  0 0 0 1     1 0 1 1  0 0 0 1     1 1 1 1
0 0 0 1     0 1 1 1  0 0 0 1     0 1 1 1  0 1 1 0     0 1 1 1  1 1 1 1     1 1
1 1  0 1 1 1
       0 1 0 0  1 0 0 1     1 1 0 1  1 0 0 1     1 1 1 1  1 0 0 1     1 0 1 1
1 0 0 1     1 0 1 0  1 0 0 1     1 0 1 0  1 0 1 1     1 0 1 0  1 0 0 0     1 0
0 0  1 1 0 0
       0 0 1 0  0 1 0 0     0 1 1 0  0 1 0 0     0 1 1 0  0 1 0 0     0 0 1 0
0 1 0 0     1 0 1 1  0 1 0 0     1 0 1 1  0 1 1 1     1 0 1 1  1 1 0 0     1 1
0 0  1 1 1 0
       0 0 0 1  0 0 1 0     0 0 1 1  0 0 1 0     0 0 1 1  0 0 1 0     0 1 1 1
0 0 1 0     1 1 1 1  0 0 1 0     1 1 1 1  0 1 0 1     1 1 1 1  1 1 1 0 交换  1 1
1 0  1 1 1 1
                   ====>>>                ====>>>                ====>>>
====>>>                ====>>>                ====>>>                ====>>>
  对|| 1 0 0 0  1 1 0 0     0 1 0 0  1 1 0 0     0 1 0 0  1 1 0 0     0 1 0 0
1 1 0 0     1 1 0 1  1 1 0 0     1 1 0 1  1 0 0 1     1 1 0 1  0 0 1 1     0 0
1 1  1 0 1 1
  角|| 0 1 0 0  0 0 1 0     0 1 1 0  0 0 1 0     0 1 1 0  0 0 1 0     0 0 1 0
0 0 1 0     1 0 1 1  0 0 1 0     1 0 1 1  0 0 0 1     1 0 1 1  1 0 1 0     1 0
1 0  1 1 1 0
  化|| 0 0 1 0  0 0 0 1     0 0 1 1  0 0 0 1     0 0 1 1  0 0 0 1     0 1 1 1
0 0 0 1     1 1 1 1  0 0 0 1     1 1 1 1  0 1 1 0     1 1 1 1  1 1 0 1     1 1
0 1  1 1 1 1
  行|| 0 0 0 1  1 0 0 0     1 0 0 1  1 0 0 0     1 0 1 1  1 0 0 0     1 1 1 1
1 0 0 0     0 1 1 1  1 0 0 0     0 1 1 1  1 1 1 1     0 1 1 1  0 1 1 0     0 1
1 0  0 1 1 1

       1 0 0 0  0 0 1 0     1 0 1 0  0 0 1 0     1 0 0 0  0 0 1 0     1 0 0 0
0 0 1 0     1 0 0 0  0 0 1 0     1 0 0 0  0 0 1 0     1 0 0 0  0 0 0 0     0 0
0 0  1 0 0 0
       0 1 0 0  0 0 1 1     0 1 1 1  0 0 1 1     0 1 1 1  0 0 1 1     0 0 1 1
0 0 1 1     0 0 1 0  0 0 1 1     0 0 1 0  0 0 0 1     0 0 1 0  0 0 0 0     0 0
0 0  0 1 0 0
       0 0 1 0  1 0 0 1     1 0 1 1  1 0 0 1     1 0 0 1  1 0 0 1     1 0 0 1
1 0 0 1     0 0 0 1  1 0 0 1     0 0 0 1  1 0 0 0     0 0 0 1  0 0 0 0     0 0
0 0  0 0 1 0
       0 0 0 1  0 1 0 0     0 1 0 1  0 1 0 0     0 1 0 1  0 1 0 0     0 1 0 1
0 1 0 0     0 1 0 0  0 1 0 0     0 1 0 0  0 0 0 0     0 1 0 0  0 0 0 0     0 0
0 0  0 0 0 1
       -----------------     -----------------     -----------------     ---------
--------     -----------------     -----------------     -----------------     --
```

```
---------------
        ^ ^ ^ ^   ^ ^ ^ ^       ^   ^                         ^ ^                     ^ ^ ^ ^
^ ^ ^                 ^   ^ ^
        | | | |   | | | |       |   |                         | |                     | | | |
| | |                 |   | |
        归归归归  | | | |       |   归                        归|                     归| | 归
归归归                归  归归
        零零零零                    零                        零                      零   零
零零零                零  零零

        1 0 0 0   0 0 0 0       1 0 0 0   0 0 0 0       1 0 1 0   0 0 0 0       1 1 1 0
0 0 0 0       1 1 1 0   0 0 0 0       1 1 1 0   0 1 1 0       1 1 1 0   0 1 0 1       0 1
0 1   1 1 0 1
        0 1 0 0   0 0 0 0       0 1 0 0   0 0 0 0       0 1 0 0   0 0 0 0       0 1 0 0
0 0 0 0       1 1 0 1   0 0 0 0       1 1 0 1   0 1 0 1       1 1 0 1   1 1 1 1       1 1
1 1   1 0 1 1
        0 0 1 0   0 0 0 0       0 0 1 0   0 0 0 0       0 0 1 0   0 0 0 0       0 1 1 0
0 0 0 0       0 1 1 0   0 0 0 0       0 1 1 0   0 1 1 0       0 1 1 0   0 1 1 1       0 1
1 1   0 1 0 1
        0 0 0 1   0 0 0 0       0 0 0 1   0 0 0 0       0 0 0 1   0 0 0 0       0 0 0 1
0 0 0 0       1 0 0 1   0 0 0 0       1 0 0 1   0 0 0 1       1 0 0 1   1 0 1 1       1 0
1 1   1 0 1 0

        0 0 0 0   1 0 0 0       1 0 0 0   1 0 0 0       1 0 1 0   1 0 0 0       1 1 1 0
1 0 0 0       1 1 1 0   1 0 0 0       1 1 1 0   1 1 1 0       1 1 1 0   1 1 0 1       1 1
0 1   1 1 0 1
        0 0 0 0   0 1 0 0       0 1 0 0   0 1 0 0       0 1 0 0   0 1 0 0       0 1 0 0
0 1 0 0       1 1 0 1   0 1 0 0       1 1 0 1   0 0 0 1       1 1 0 1   1 0 1 1       1 0
1 1   1 0 1 1
        0 0 0 0   0 0 1 0       0 0 1 0   0 0 1 0       0 0 1 0   0 0 1 0       0 1 1 0
0 0 1 0       0 1 1 0   0 0 1 0       0 1 1 0   0 1 0 0       0 1 1 0   0 1 0 1       0 1
0 1   0 1 0 1
        0 0 0 0   0 0 0 1       0 0 0 1   0 0 0 1       0 0 0 1   0 0 0 1       0 0 0 1
0 0 0 1       1 0 0 1   0 0 0 1       1 0 0 1   0 0 0 0       1 0 0 1   1 0 1 0       1 0
1 0   1 0 1 0
```

$$
\begin{pmatrix}
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1
\end{pmatrix}
*
\begin{pmatrix}
\mathrm{D}_{00} \\ \mathrm{D}_{01} \\ \mathrm{D}_{02} \\ \mathrm{D}_{03} \\ \mathrm{D}_{10} \\ \mathrm{D}_{11} \\ \mathrm{D}_{12} \\ \mathrm{D}_{13} \\ \mathrm{D}_{20} \\ \mathrm{D}_{21} \\ \mathrm{D}_{22} \\ \mathrm{D}_{23} \\ \mathrm{D}_{30} \\ \mathrm{D}_{31} \\ \mathrm{D}_{32} \\ \mathrm{D}_{33}
\end{pmatrix}
=
\begin{pmatrix}
D_{00} \oplus D_{10} \oplus D_{11} \oplus D_{12} \oplus D_{13} \oplus D_{22} \oplus D_{23} \\
D_{01} \oplus D_{10} \oplus D_{20} \oplus D_{22} \\
D_{02} \oplus D_{10} \oplus D_{11} \oplus D_{20} \oplus D_{21} \oplus D_{23} \\
D_{03} \oplus D_{10} \oplus D_{11} \oplus D_{12} \oplus D_{21} \oplus D_{22} \\
D_{11} \oplus D_{12} \oplus D_{13} \oplus D_{20} \oplus D_{22} \oplus D_{23} \oplus D_{30} \\
D_{10} \oplus D_{11} \oplus D_{20} \oplus D_{21} \oplus D_{22} \oplus D_{31} \\
D_{10} \oplus D_{11} \oplus D_{12} \oplus D_{20} \oplus D_{21} \oplus D_{22} \oplus D_{23} \oplus D_{32} \\
D_{10} \oplus D_{11} \oplus D_{12} \oplus D_{13} \oplus D_{21} \oplus D_{22} \oplus D_{23} \oplus D_{33}
\end{pmatrix}
$$

$$
=
\begin{pmatrix}
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0
\end{pmatrix}
\begin{pmatrix}
P_{00} \\ P_{01} \\ P_{02} \\ P_{03} \\ P_{10} \\ P_{11} \\ P_{12} \\ P_{13}
\end{pmatrix}
=
\begin{pmatrix}
P_{01} \oplus P_{03} \oplus P_{10} \oplus P_{11} \\
P_{00} \oplus P_{01} \oplus P_{02} \oplus P_{03} \oplus P_{10} \oplus P_{12} \oplus P_{13} \\
P_{01} \oplus P_{02} \oplus P_{03} \oplus P_{11} \oplus P_{13} \\
P_{00} \oplus P_{02} \oplus P_{03} \oplus P_{10} \oplus P_{12} \\
P_{00} \oplus P_{01} \oplus P_{10} \oplus P_{11} \oplus P_{13} \\
P_{00} \oplus P_{02} \oplus P_{03} \oplus P_{10} \oplus P_{12} \oplus P_{13} \\
P_{01} \oplus P_{03} \oplus P_{11} \oplus P_{13} \\
P_{00} \oplus P_{02} \oplus P_{10} \oplus P_{12}
\end{pmatrix}
$$

$$
=>
\begin{pmatrix}
D_{00} \\ D_{01} \\ D_{02} \\ D_{03} \\ D_{30} \\ D_{31} \\ D_{32} \\ D_{33}
\end{pmatrix}
=
\begin{pmatrix}
P_{01} \oplus P_{03} \oplus P_{10} \oplus P_{11} \oplus D_{10} \oplus D_{11} \oplus D_{12} \oplus D_{13} \oplus D_{22} \oplus D_{23} \\
P_{00} \oplus P_{01} \oplus P_{02} \oplus P_{03} \oplus P_{10} \oplus P_{12} \oplus P_{13} \oplus D_{10} \oplus D_{20} \oplus D_{22} \\
P_{01} \oplus P_{02} \oplus P_{03} \oplus P_{11} \oplus P_{13} \oplus D_{10} \oplus D_{11} \oplus D_{20} \oplus D_{21} \oplus D_{23} \\
P_{00} \oplus P_{02} \oplus P_{03} \oplus P_{10} \oplus P_{12} \oplus D_{10} \oplus D_{11} \oplus D_{12} \oplus D_{21} \oplus D_{22} \\
P_{00} \oplus P_{01} \oplus P_{10} \oplus P_{11} \oplus P_{13} \oplus D_{11} \oplus D_{12} \oplus D_{13} \oplus D_{20} \oplus D_{22} \oplus D_{23} \\
P_{00} \oplus P_{02} \oplus P_{03} \oplus P_{10} \oplus P_{12} \oplus P_{13} \oplus D_{10} \oplus D_{11} \oplus D_{20} \oplus D_{21} \oplus D_{22} \\
P_{01} \oplus P_{03} \oplus P_{11} \oplus P_{13} \oplus D_{10} \oplus D_{11} \oplus D_{12} \oplus D_{20} \oplus D_{21} \oplus D_{22} \oplus D_{23} \\
P_{00} \oplus P_{02} \oplus P_{10} \oplus P_{12} \oplus D_{10} \oplus D_{11} \oplus D_{12} \oplus D_{13} \oplus D_{21} \oplus D_{22} \oplus D_{23}
\end{pmatrix}
$$

方法：将失效矩阵的二进制矩阵的方阵单位化，然后可以求得失效块恢复计算方式

二进制矩阵的单位化很繁琐，有没有更好的选择？？？，答案是有=>使用同构运算传递性

还是GF(2^4)为例子，假设D0和D3失效

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 9 & 4 \end{pmatrix} * \begin{pmatrix} D0 \\ D1 \\ D2 \\ D3 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} P0 \\ P1 \end{pmatrix}$$

=>

$$\begin{pmatrix} g^0 & g^0 & g^0 & g^0 \\ g^0 & g^1 & g^{14} & g^2 \end{pmatrix} * \begin{pmatrix} D0 \\ D1 \\ D2 \\ D3 \end{pmatrix} = \begin{pmatrix} g^0 & 0 \\ 0 & g^0 \end{pmatrix} \begin{pmatrix} P0 \\ P1 \end{pmatrix}$$

=>

$$\begin{pmatrix} g^0 & g^0 & g^0 & g^0 \\ g^0 \ \text{<归零>} & g^1 & g^{14} & g^2 \end{pmatrix} = \begin{pmatrix} g^0 & 0 \\ 0 & g^0 \end{pmatrix} \quad => \quad \begin{pmatrix} g^0 & g^0 & g^0 & g^0 \\ 0 & g^1 \oplus g^0 & g^{14} \oplus g^0 & g^2 \oplus g^0 \end{pmatrix} = \begin{pmatrix} g^0 & 0 \\ g^0 & g^0 \end{pmatrix}$$

=>

$$\begin{pmatrix} g^0 & g^0 & g^0 & g^0 \\ 0 & g^4 & g^3 & g^8 \ \text{<单位化>} \end{pmatrix} = \begin{pmatrix} g^0 & 0 \\ g^0 & g^0 \end{pmatrix} \quad => \quad \begin{pmatrix} g^0 & g^0 & g^0 & g^0 \\ 0 * g^7 & g^4 * g^7 & g^3 * g^7 & g^8 * g^7 \end{pmatrix} = \begin{pmatrix} g^0 & 0 \\ g^0 * g^7 & g^0 * g^7 \end{pmatrix}$$

=>

$$\begin{pmatrix} g^0 & g^0 & g^0 & g^0 \ \text{<归零>} \\ 0 & g^{11} & g^{10} & g^0 \end{pmatrix} = \begin{pmatrix} g^0 & 0 \\ g^7 & g^7 \end{pmatrix} \quad => \quad \begin{pmatrix} g^0 & g^0 \oplus g^{11} & g^0 \oplus g^{10} & 0 \\ 0 & g^{11} & g^{10} & g^0 \end{pmatrix} = \begin{pmatrix} g^0 \oplus g^7 & g^7 \\ g^7 & g^7 \end{pmatrix}$$

=>

$$\begin{pmatrix} g^0 & g^{12} & g^5 & 0 \\ 0 & g^{11} & g^{10} & g^0 \end{pmatrix} = \begin{pmatrix} g^9 & g^7 \\ g^7 & g^7 \end{pmatrix}$$

=>

$$\begin{pmatrix} 1 & 15 & 6 & 0 \\ 0 & 14 & 7 & 1 \end{pmatrix} = \begin{pmatrix} 10 & 11 \\ 11 & 11 \end{pmatrix}$$

$$
\begin{pmatrix}
1&0&0&0&1&1&1&1&0&0&1&1&0&0&0&0\\
0&1&0&0&1&0&0&0&1&0&1&0&0&0&0&0\\
0&0&1&0&1&1&0&0&1&1&0&1&0&0&0&0\\
0&0&0&1&1&1&1&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&1&1&1&1&0&1&1&1&0&0&0\\
0&0&0&0&1&1&0&0&1&1&1&0&0&1&0&0\\
0&0&0&0&1&1&1&0&1&1&1&1&0&0&1&0\\
0&0&0&0&1&1&1&1&0&1&1&1&0&0&0&1
\end{pmatrix}
*
\begin{pmatrix}
\mathrm{D}_{00}\\ \mathrm{D}_{01}\\ \mathrm{D}_{02}\\ \mathrm{D}_{03}\\ \mathrm{D}_{10}\\ \mathrm{D}_{11}\\ \mathrm{D}_{12}\\ \mathrm{D}_{13}\\ \mathrm{D}_{20}\\ \mathrm{D}_{21}\\ \mathrm{D}_{22}\\ \mathrm{D}_{23}\\ \mathrm{D}_{30}\\ \mathrm{D}_{31}\\ \mathrm{D}_{32}\\ \mathrm{D}_{33}
\end{pmatrix}
=
\begin{pmatrix}
0&1&0&1&1&1&0&0\\
1&1&1&1&1&0&1&1\\
0&1&1&1&0&1&0&1\\
1&0&1&1&1&0&1&0\\
1&1&0&0&1&1&0&1\\
1&0&1&1&1&0&1&1\\
0&1&0&1&0&1&0&1\\
1&0&1&0&1&0&1&0
\end{pmatrix}
\begin{pmatrix}
P_{00}\\ P_{01}\\ P_{02}\\ P_{03}\\ P_{10}\\ P_{11}\\ P_{12}\\ P_{13}
\end{pmatrix}
$$

$$
==> \begin{pmatrix} 1 & 15 & 6 & 0 \\ 0 & 14 & 7 & 1 \end{pmatrix} * \begin{pmatrix} D0 \\ D1 \\ D2 \\ D3 \end{pmatrix} = \begin{pmatrix} 10 & 11 \\ 11 & 11 \end{pmatrix} \begin{pmatrix} P0 \\ P1 \end{pmatrix}
$$

首先GF(2^4)下完成失效方阵的单位化，然后将计算结果映射到 ϕ(GF(2^4))，可以简化运算

后续

- 生成矩阵的确认：范德蒙德的生成矩阵重量一定比柯西矩阵的重量重
  - 确认：应该是看到文章的论述有问题，不能用重量衡量，应该是柯西矩阵的计算逆矩阵的复杂度比范德蒙德矩阵复杂度低
- 数据块的异或运算使用英特尔® 高级矢量扩展指令集，提高运算速度
- 实际应用例子：在生产中处理遇到的整条写，重构写，读改写情况，以及RAID中涉及的“写洞”问题

参考文献

伽罗瓦理论之美

有限域上的多项式

另一种世界观——有限域

Finite fields - 有限域

有限域GF(2^8)的四则运算

伽罗华域（Galois Field）上的四则运算

有限域GF(2^8)的四则运算及拉格朗日插值

信息安全数学基础 许春香 第三章:循环群和群的结构

矩阵对角化的若干方法

从一元一次方程到伽罗瓦理论（修订版）

基于二进制矩阵RS编码优化算法编码优化算法

基于重码和二进制矩阵的RAID 编码算法研究