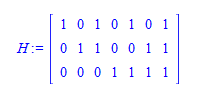
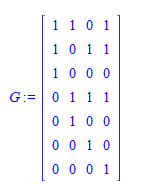
Error-correcting code

一个信息通常以binary block二进制block在一个嘈杂的频道中传递，有可能产生error，我们试图 1.检测到有error,2.纠正这个error

1.Parity Check Code, 在原有的基础上加一个parity(奇偶位)，等与前面的所有二进制数字加起来Mod2，如果有一个error,那么就会0变1 or 1 变0，奇偶就变了·， 就能检测到1个error，不能纠正，也不知道位置

2.Repetition code. 每一个信息digit都有充足的redundancy备用， 例如我们发出digit a，就发出aaaaa， 这样如果有error就能检测并改正，如果被传递的data小于等于(n-1)/2//五位错了两位能纠正，三位就挣不了， 如果收到了aaabb，那么结论就是a，但是效率很低为1/n

3.Humming Error-Detection and Correction Codes

这两个东西你可以想象成小虎队的卡

G:第一列是第一个数，第二列是第二个数，.以第一行为例，这一行代表第一个数加第二个数加第四个数

想要传递一个四位信息，例如

就要生成对应vector,，第一行为例，1+0+1是偶数，所以是0

这只是解释原理，我们实际应用中，例如要传递(1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1)

要把它四位四位切分



再让G\*他们组成的矩阵 Mod2



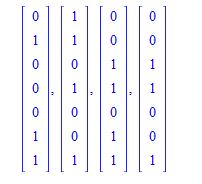
就可以直接生成

然后用第二张卡

让第二张卡\*每一生成列 mod 2 //也可以直接乘以前一步生成的U，这里是另外一个例子



如果那一列全部为0，代表没错误，



不然就从下往上看，

011,代表第一组数错误在第三位，

第一列应该是<0, 1, 1, 0, 0, 1, 1>

110代表第三组数错位在第1\*2^2+1\*2=第六位

第三列应该是<0, 0, 1, 1, 0, 0, 1>

这个方法只能检测一个error

以第四列为例，我们人工给他加两个error让他变成v:=1111001（前两位是error）

H\*v mod 2=,011 检测出来error是第三位，

只有一个一个加error

1011001

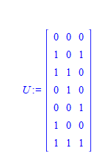
0111001 才能检测出正确的对应error

它实际上把两个error列combine了，导致检测的位置有问题

总结：第一步把要发的信息四位四位切分，组成一个新的

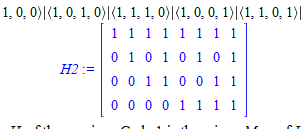
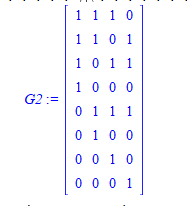


然后要让G乘以他们组合的矩阵Mod2



然后让H\*Umod 2为0的代表没问题，不然就有问题，倒着看二进制找到错的位(U的列)

4.Code2，允许检测到两个error，或纠正1个error

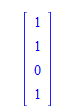


第一步还是切四位，G2\*四位数Mod2



假设error为第六位既00110111

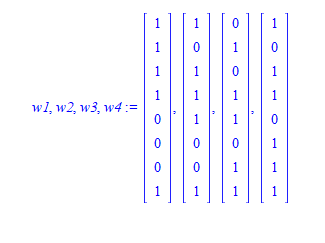
还是让H2\*Rmod2,

这时候区别来了，

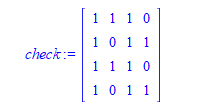
如果第一位是1，就代表有一个error可以纠正

 计算方法不再是倒着二进制，而是1+1\*2^0 1 2

例子：







第一二三列都能改正error,

0101不行，有可能是

1101+1000 可能性1

1001+1100 可能性2

Application2 : Hill Cryptosystem

第一步，把信息按照字母表转化成对应的数字串

个位数写成0X

第二步，利用n阶加密矩阵

n=几代表着几个数字一组，要切割，让数字串按组组成新矩阵

3.用加密矩阵A乘以原来的数字串mod 26(26取决于字母表长度)

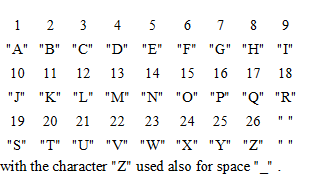
4.determinant(A)看是不是26的Unit,

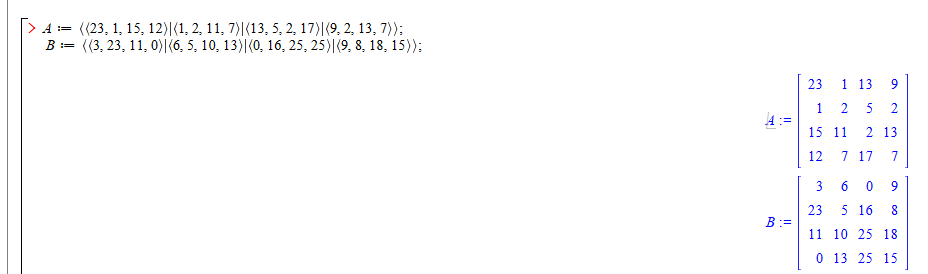
然后求逆矩阵

5.让逆矩阵乘以加密后的数字串 mod 26

例题：







第一步，看哪个加密矩阵能用，

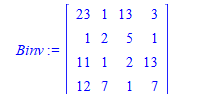


第二个等于7，互质用第二个

第二步：

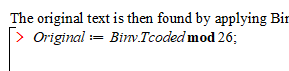
求逆矩阵

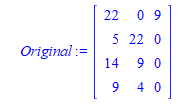




第三步：切割原数字，构建矩阵



第四部逆矩阵乘以原数字mod 26

竖着念