Práctica 9. Sesión 2 - GAP / GUAVA

7 de diciembre del 2022

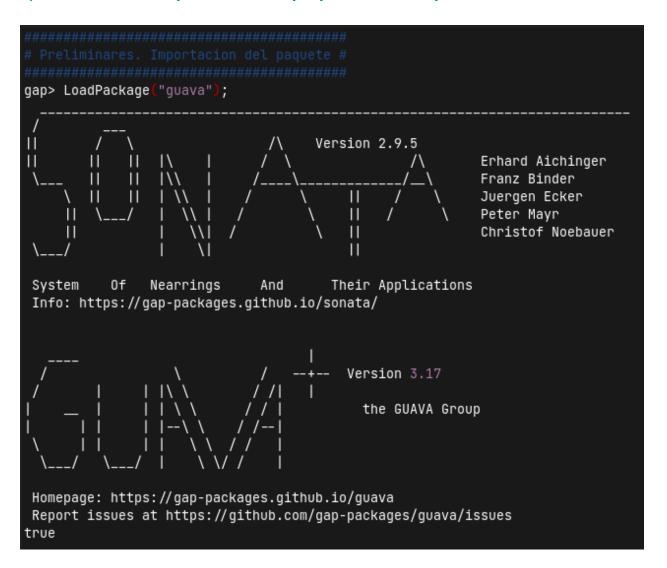
Marcos Hidalgo Baños

NOTA. El archivo .log que recoge cada comando con su resultados se encuentra en este <u>enlace</u> a mi repositorio personal de GitHub.

Apartado 1. Cotas

Encontrar un código lineal y binario con la mayor cantidad posible de palabras de longitud 63 y distancia mínima 5.

0) Preliminares. Importación del paquete GUAVA para GAP.



a) Calcular el valor que proporciona la cota de Hamming.

```
gap> uboundhamming := UpperBoundHamming(63,5,2);
4572817073304301

gap> Log2(Float(uboundhamming));
52.022
```

b) Calcular para los mismos valores la cota de Gilbert.

```
gap> lboundgilbert := LowerBoundGilbertVarshamov(63,5,2);
140737488355328

gap> Log2(Float(lboundgilbert));
47.
```

c) Calcular la cota de Singleton. ¿Qué ocurre?

```
gap> uboundsingleton := UpperBoundSingleton(63,5,2);
576460752303423488

gap> Log2(Float(uboundsingleton));
59.
```

d) Calcular la cota de Plotkin, para un valor de la distancia mínima posible.

```
gap> uboundplotkin := UpperBoundPlotkin(63,5,2);
180143985094819840

gap> Log2(Float(uboundplotkin));
57.3219
```

Apartado 2.

Dada la siguiente matriz generadora:

$$G = \left(\begin{array}{ccccc} 1 & 0 & 2 & 0 & 1 \\ 2 & 2 & 1 & 1 & 1 \end{array}\right)$$

 a) Calcular los parámetros del código, la capacidad correctora así como la distancia mínima.

b) ¿Cuántos son los posibles vectores de error dada la capacidad correctora del código?

```
gap> Check := CheckMat(C2);
[ [ Z(3)^0, 0*Z(3), Z(3)^0, 0*Z(3), 0*Z(3) ],
      [ 0*Z(3), Z(3)^0, 0*Z(3), Z(3)^0, 0*Z(3) ],
      [ Z(3), Z(3), 0*Z(3), 0*Z(3), Z(3)^0 ] ]
```

c) ¿Cuántos cosets diferentes hay?¿Cuántos elementos hay en cada coset?

```
gap> Codeword(Check);
[ [ 1 0 1 0 0 ], [ 0 1 0 1 0 ], [ 2 2 0 0 1 ] ]
```

d) Calcular la tabla de síndromes.

```
gap> SyndromeTable(C2);
                                                    0 0 1
      0 0 0 0 2
                     0 0 2
                                    0 0 0 1 0
                                                    0 1 0
      0 0 0 1 1
                     0 1 1
                                    0 1 0 0 0
                                                    0 1 2
      00020
                     0 2 0
                                    0 2 0 0 0
                                                    0 2 1
      0 0 0 2 2
                     0 2 2
                                    0 0 1 0 0
                                                    1 0 0
      0 0 1 1 0
                     1 1 0
                                    1 1 0 0 0
      0 2 1 0 0
                     1 2 1
                                    1 0 0 2 0
                                                    1 2 2
      0 0 2 0 2
                     2 0 2
                                    0 0 2 1 0
                                                    2 1 0
                     2 1 1
      2 2 0 0 0
                     2 2 2
```

e) Recibido a la salida del canal el vector w = (12212). Comprobar si es correcto o se ha producido algún error.

```
gap> c := Codeword("12212");
[ 1 2 2 1 2 ]
gap> c in C2;
false
gap> Syndrome(C2,c);
[ 0 0 2 ]
```

Podemos comprobar que la palabra no pertenece al código.

f) Recibido el vector w = (12202), decodificar.

```
gap> c := Codeword("12202");
[ 1 2 2 0 2 ]
gap> c in C2;
false
gap> Syndrome(C2,c);
[ 0 2 2 ]
```