Universidad de Granada

decsai.ugr.es

Teoría de la Información y la Codificación

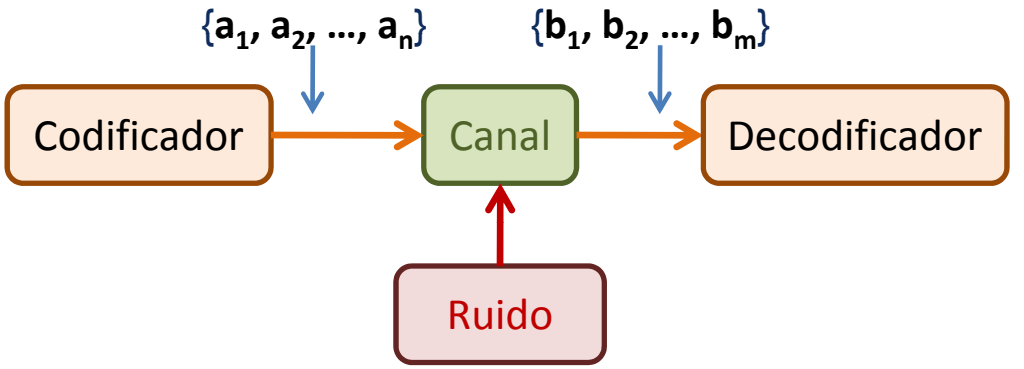
Grado en Ingeniería Informática

Tema 5.- Teoría y desarrollo de códigos detectores y correctores de errores.

DECSAI

Departamento de Ciencias de la Computación e Inteligencia Artificial

- Vamos a suponer una fuente **F** emisora de símbolos, y un receptor **S**.
- Supondremos también el alfabeto de la fuente **F** tiene **n** símbolos $\mathbf{a_1, a_2, \ldots, a_n}$, con probabilidades $\mathbf{p(a_1), p(a_2), \ldots, p(a_n)}$.
- Para generalizar, también supondremos que el receptor **S** puede percibir **m** símbolos $\mathbf{b_1, b_2, \ldots, b_m}$, con probabilidades $\mathbf{p(b_1), p(b_2), \ldots, p(b_m)}$



- Trabajaremos con **códigos uniformes**; es decir, códigos cuyas palabras tienen una **longitud fija *k***.

- A esos ***k*** bits del código, se le añadirán un conjunto de bits de redundancia ***r***, hasta llegar a formar códigos uniformes cuyas palabras tenga longitud ***n***.

$$n = k + r$$

- **Ejemplo:** Código de 4 símbolos {A, B, C, D} con k=2 bits, con redundancia r=1 bit: Código de n=3 bits.

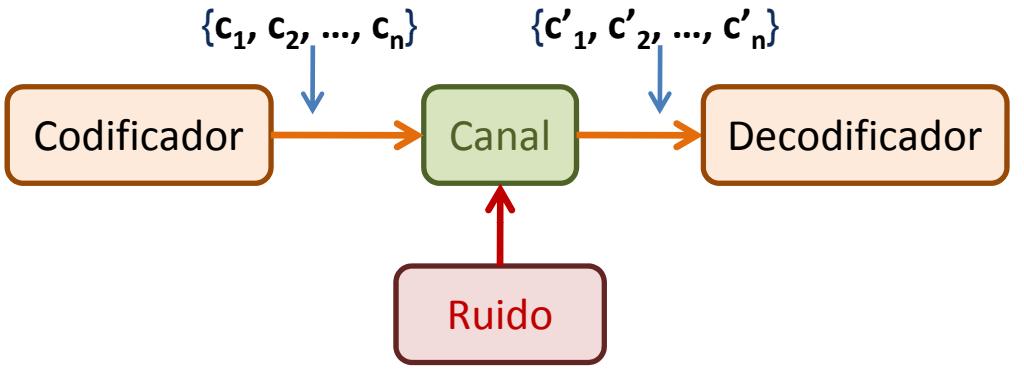
Redundancia mediante bit de paridad:

A: 00 → 000
B: 01 → 011
C: 10 → 101
D: 11 → 110

– En este modelo, el emisor envía un mensaje codificado **c**, de **n** bits

$$c=(c_1, c_2, c_3, \ldots, c_n)$$

– Si existe perturbación en el canal (ruido), el receptor recibirá otro mensaje diferente **c'**, del mismo número de bits (**porque asumimos códigos uniformes**).



- Para modelar el error, asumiremos que al vector **c** se le incorpora otro vector de error **e**, del mismo número de bits del mensaje, tal que:

$$c' = c + e$$

- **Ejemplo:**

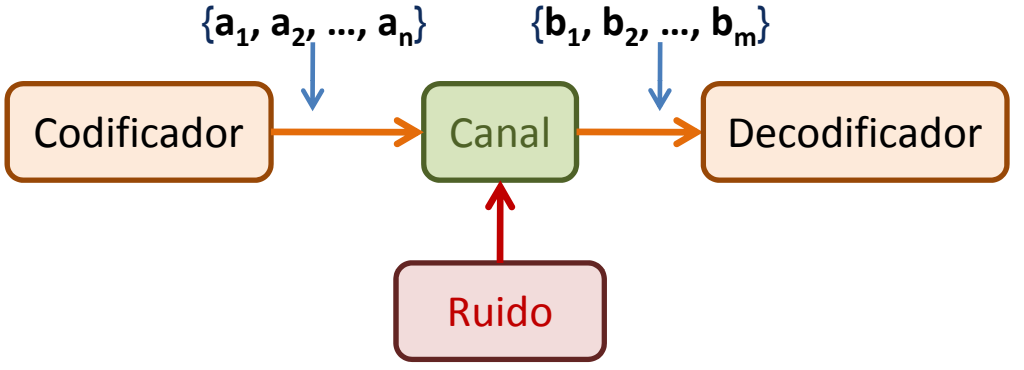
  La fuente envía un código (01101)
  El canal introduce un error en uno o varios bits al azar: (00100)
  El receptor recibe el mensaje perturbado: c'=(01101)

- **Trataremos cada mensaje y modelo de error como un álgebra modular en Z_2, bit a bit.**

- **Mensaje original c=(01101) + error e=(00100) → c'=(01001)**

Pregunta: En estas circunstancias, ¿es posible transmitir información por el canal de modo que se pueda reconstruir el mensaje en el destino?

- **La idea de Shannon: Aplicar la detección de errores en transmisiones.**
  - Demostrar que se puede enviar datos en canales con ruido.
  - Intentar acotar o encontrar límites para la transmisión.



- **TEOREMA DE SHANNON PARA CODIFICACIÓN CON RUIDO:**
  - Sea un canal simétrico binario con probabilidad de error p y sea R un número que satisface $0 < R < C(p)$
  - Entonces, para cualquier $\varepsilon>0$, para un *n* lo suficientemente grande existe un $(n, 2^k, d)$-código con ratio $k/n<=R$ tal que $e(C) < \varepsilon$.
  - Shannon nos asegura la existencia de al menos un código que puede emitirse y recibirse sin pérdida de información, con una probabilidad tan cercana a 0 como se desee.
  - Sin embargo, no nos dice cómo calcular el código.

  - En este tema estudiamos diferentes propuestas de códigos que, en caso de alteraciones, pueden ser reconstruidos en el destino.

- **TEOREMA DE HAMMING PARA CORRECCIÓN DE ERRORES:**
  - **Si se desea detectar errores:**
    - Para detectar d errores de un bit entre una palabra de un código, es necesario que el código tenga una distancia de Hamming de al menos d+1
  - **Si se desea corregir errores:**
    - Para corregir d errores de un bit en una palabra de un código, es necesario que el código tenga una distancia de Hamming de al menos 2*d+1
  - **Si tenemos un código con distancia d=2→ posibilidad de corregir:**

    (d-1)/2= (2-1)/2= 0.5 errores → No se puede corregir el error ni para un bit

- **TEOREMA DE HAMMING PARA CORRECCIÓN DE ERRORES:**

  - Si tenemos un código con distancia d=3→ posibilidad de corregir:

    (d-1)/2= (3-1)/2= 1 error → Es posible corregir errores que haya en 1 bit.

Ejemplo: Sea un alfabeto {A, B, C} codificado como

A= 000111
B= 111000
C= 110011

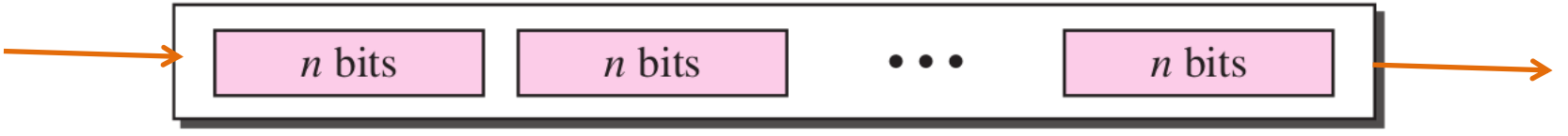
Distancia del código d=3

Se recibe 100111 → Se corrige a 000111
Se recibe 110111 → Se detecta error, pero no se puede corregir

- **Supondremos, por simplicidad que los códigos a enviar son uniformes.**

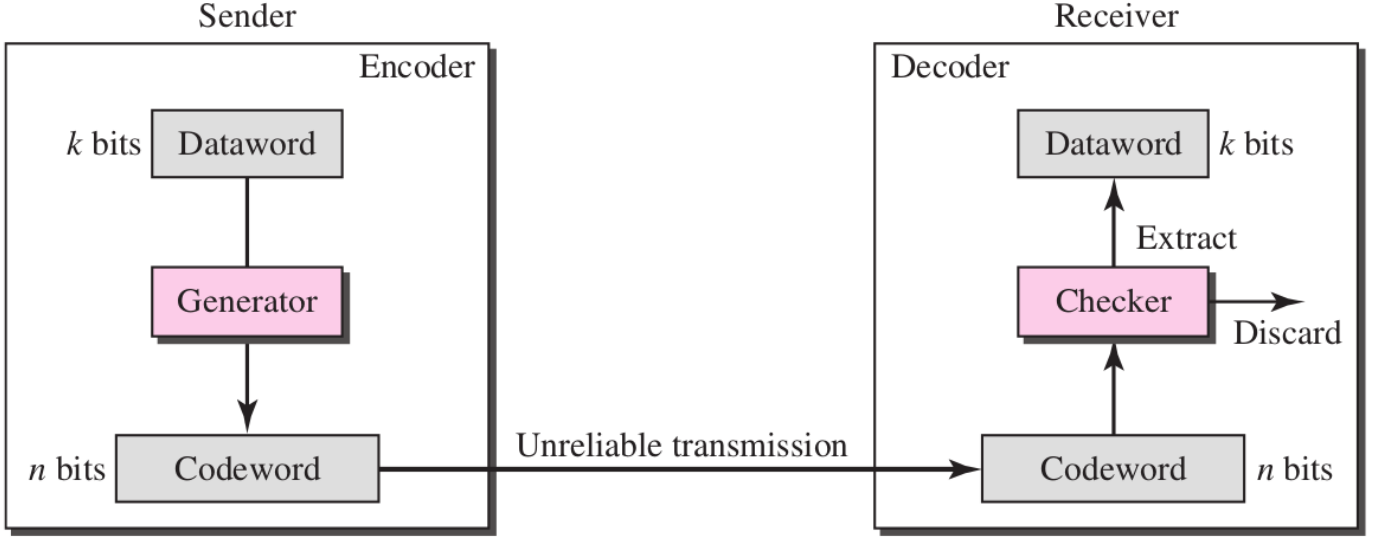
- **Codificación por bloques:**
  - **Tenemos un código con M palabras, $M<=2^k$**
  - La codificación por bloques envía el mensaje como secuencias de bloques de *k bits.*

  - Como estamos en canales con ruido, a los *k* bits del código habrá que introducir *r* bits de redundancia, emitiendo códigos de ***n=k+r* bits.**



– Esquema de transmisión:

- **Tenemos un código con M palabras, M<=2^k (k bits).**
- Se le añade la redundancia **r** hasta completar los **n=k+r** bits.
- ***Se envía el bloque, y se recibe en el receptor.***
- El receptor **comprueba la validez del mensaje,**
- **y lo decodifica en el código original de *k* bits.**



- Los códigos lineales facilitan enormemente la codificación y decodificación de mensajes.
  - Se basan en álgebra modular (módulo q)
  - Hay 2 operadores asociados: + y ·

$$a + b = (a + b) \bmod q$$
$$a \cdot b = (a * b) \bmod q$$

Los términos "a" y "b" son valores en {0, 1, 2, …, q-1}

Ejemplo: Si q=5 → alfabeto {0, 1, 2, 3, 4}

1+1= 2; 2+3= 0; 2+4= 1; 3+3= 1; 1+4= 0

1·1= 1; 1·2= 2; 2·2= 4; 3·3= 4; 2·4= 3

- Nos referiremos a códigos lineales como: F_q^n

  - *Un código lineal C que sea F_q^n es un código que utiliza álgebra modular módulo q, cuyas palabras tienen n bits.*

  - Además, el código C debe cumplir las siguientes propiedades:

    Si "x" e "y" son dos palabras del código, entonces "x+y" también lo es.

    Para cualquier valor L en {0, 1, 2, …, q-1}, L*x también es una palabra del código.

Nos centraremos en q=2, para codificar en binario.

La idea que perseguimos es añadir redundancia a nuestro código ya construido de *k* bits, transformándolo a *n* bits para corregir errores.

- **Ejemplo de un código lineal C en F_2^3 :**

$$\mathcal{C} = \{[0\ 0\ 0], [1\ 0\ 0], [0\ 1\ 0], [1\ 1\ 0]\}$$

- Se cumple que, sean cuales sean las palabras **x** e **y** que cojamos del código, su suma será otra palabra del código.
- **Ejemplo:**

[1 0 0] + [0 1 0] = [1 1 0]
[1 1 0] + [1 0 0] = [0 1 0]

- Se cumple que, sea cual sea la palabra **x** que cojamos del código y el número **L** en {0, 1}, **L*x** será una palabra del código.
- **Ejemplo:**

L= 0 y x=[1 1 0] → 0*[1 1 0]= 0
L= 1 y x=[0 1 0] → 1*[0 1 0]= [0 1 0]

– Propiedades de un código lineal (I)

1. **Son invariantes a traslaciones.** Si se coge cualquier palabra **a** del código, y se suma a todas las palabras del código, el resultado es el mismo código.

– Ejemplo:

a= [1 1 0]

$$\mathcal{C} = \{[0\ 0\ 0], [1\ 0\ 0], [0\ 1\ 0], [1\ 1\ 0]\}$$

a + [0 0 0] = [1 1 0] +[0 0 0]= [1 1 0], que está en el código
a + [1 0 0] = [1 1 0] +[1 0 0]= [0 1 0], que está en el código
a + [0 1 0] = [1 1 0] +[0 1 0]= [1 0 0], que está en el código
a + [1 1 0] = [1 1 0] +[1 1 0]= [0 0 0], que está en el código

– Propiedades de un código lineal (II)

1. El opuesto *–a* de una palabra del código a también pertenece al código.
2. Además, a+(-a) = $(0)^n$

– **Ejemplo:** $\mathcal{C} = \{[0\ 0\ 0], [1\ 0\ 0], [0\ 1\ 0], [1\ 1\ 0]\}$

a= [1 1 0] → -a= [-1 -1 0]= [1 1 0] → a-a = [0 0 0]
a= [0 1 0] → -a= [0 -1 0]= [0 1 0] → a-a = [0 0 0]
a= [1 0 0] → -a= [-1 0 0]= [1 0 0] → a-a = [0 0 0]
a= [0 0 0] → -a= [0 0 0] → a-a = [0 0 0]

- **Cálculo de la distancia de Hamming en un código lineal**

- Supongamos que el código lineal **C** tiene **M** palabras $\mathbf{c_1}$, $\mathbf{c_2}$, …, $\mathbf{c_M}$

- **Si las palabras c_1, c_2, c_{i-1}, c_{i+1}, …, c_M están a una distancia *d* de c_i, entonces también lo están a una distancia *d* de $(0)^n$**

- **Ejemplo:**

$$\mathcal{C} = \{[0\ 0\ 0], [1\ 0\ 0], [0\ 1\ 0], [1\ 1\ 0]\}$$

  - **[1 0 0] y [0 1 0] están a distancia d=1 de [1 1 0] → También están a distancia d=1 de [0 0 0].**

- **Basta con comparar la distancia de cada palabra del código con $(0)^n$ para saber la distancia del código, quedándonos con la mínima distancia.**
  - **D([1 0 0] , [0 0 0])= 1**
  - **D([0 1 0] , [0 0 0])= 1**
  - **D([1 1 0] , [0 0 0])= 2**
    - **Código de distancia d=1**

- **Representación matricial de los códigos lineales**

- Como son códigos lineales, por bloques, se pueden representar como matrices.

- **Como es posible representarlo como matrices, tienen asociados un espacio vectorial.**

- **Como tienen asociados un espacio vectorial, tienen una base del espacio vectorial.**

- **Conocer esta base es suficiente para poder generar todas las palabras del código.**

- **Suponiendo que cada palabra tenga longitud *n* y que la base esté formada por *k* vectores, la matriz tendrá *n* columnas y *k* filas.**

- **Se podrán codificar códigos con palabras de longitud *k* en códigos lineales, con redundancia, de longitud n**

- **Representación matricial de los códigos lineales**

- Ejemplo: Para el código siguiente:

$$\mathcal{C} = \{[0\ 0\ 0], [1\ 0\ 0], [0\ 1\ 0], [1\ 1\ 0]\}$$

- **Son linealmente independientes [1 0 0] y [0 1 0].**

- **Matriz generadora del código:**

$$M(C) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

- **Se dice que el código lineal es un código (n,k)→ Codifica palabras de longitud *k* en palabras de longitud *n*.**

- **El código anterior del ejemplo es un (3, 2).**

- **Cómo codificar utilizando la matriz de codificación**

- Según un código lineal (n, k), las **palabras a codificar tienen longitud *k*** y las **palabras codificadas tendrán longitud *n***.

- **En un código lineal (3, 2), como el del ejemplo, las palabras a codificar tienen longitud 2, y las palabras codificadas tendrán longitud 3.**

- **Ejemplo: Sea x=(x1, x2) la palabra a codificar. La codificación con la matriz M(C) del código es:**

$$x \cdot M(C) = (x_1 x_2) \begin{pmatrix} m_{11} & m_{12} & m_{13} & \dots & m_{1n} \\ m_{21} & m_{22} & m_{23} & \dots & m_{2n} \end{pmatrix} =$$

$$= x_1 (m_{11} m_{12} m_{13} \dots m_{1n}) + x_2 (m_{21} m_{22} m_{23} \dots m_{2n})$$

$$M(C) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \longrightarrow x \cdot M(C) = (x_1 x_2) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = (x_1, x_2, 0)$$

- Ejemplo: Codificar las palabras (00, 01, 10, 11)

$$x \cdot M(C) = (x_1 x_2)\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = (x_1, x_2, 0)$$

  - (x1 x2)→ (x1 x2)*M(C)= (x1, x2, 0)

(00)→ (00)*M(C)= (0, 0, 0)
(01)→ (01)*M(C)= (0, 1, 0)
(10)→ (10)*M(C)= (1, 0, 0)
(11)→ (11)*M(C)= (1, 1, 0)

  - También podríamos haber cogido otra matriz generadora del código. Por ejemplo, con los vectores (1, 0, 0) y (1, 1, 0).

$$M(C) = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

  - Sigue siendo un (3, 2)→ Codificamos longitudes 2 en longitudes 3.

- Ejemplo: Codificar las palabras (00, 01, 10, 11)

$$M(C) = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

- El proceso de codificación sería el mismo:

$$x \cdot M(C) = (x_1 x_2) \begin{pmatrix} m_{11} & m_{12} & m_{13} & \dots & m_{1n} \\ m_{21} & m_{22} & m_{23} & \dots & m_{2n} \end{pmatrix} =$$

$$= x_1 (m_{11} m_{12} m_{13} \dots m_{1n}) + x_2 (m_{21} m_{22} m_{23} \dots m_{2n})$$

- En este caso:

$$x \cdot M(C) = (x_1 x_2) \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} = x_1 (1,0,0) + x_2 (1,1,0) =$$

$$= (x_1 + x_2, x_2, 0)$$

- Ejemplo: Codificar las palabras (00, 01, 10, 11)

$$x \cdot M(C) = (x_1 x_2)\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} = x_1(1,0,0) + x_2(1,1,0) =$$

$$= (x_1 + x_2, x_2, 0)$$

- Si codificamos nuestras palabras originales (00), (01), (10), (11) con esta matriz:

(00)→ (00)*M(C)= (0, 0, 0)
(01)→ (01)*M(C)= (1, 1, 0)
(10)→ (10)*M(C)= (1, 0, 0)
(11)→ (11)*M(C)= (0, 1, 0)

- En este caso, vemos que el código es diferente, porque la matriz M(C) también lo es.

- El transmisor y el receptor deben conocer la matriz de codificación para poder decodificar los símbolos iniciales.

– Estructura de las matrices generadoras

– Se puede construir siempre, por permutación de filas y columnas, una matriz generadora M(C) que tenga la siguiente estructura:

$$M(C)=\begin{pmatrix} p_{11} & p_{12} & p_{13} & \dots & p_{1,n-k} & 1 & 0 & 0 & 0 & 0 \\ p_{21} & p_{22} & p_{23} & \dots & p_{2,n-k} & 0 & 1 & 0 & 0 & 0 \\ p_{31} & p_{32} & p_{33} & \dots & p_{3,n-k} & 0 & 0 & 1 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ p_{k1} & p_{k2} & p_{k3} & \dots & p_{k,n-k} & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

– A la parte izquierda se le denomina **matriz de paridad**, mientras que a la derecha se le añade la **matriz identidad**.

$$M(C)=[P_{k,n-k}I_k]$$

– **Si imponemos que la matriz tenga esta forma, podemos seguir codificando como antes.**

– **Sin embargo, al reorganizar la matriz, no se nos genera el mismo código, sino uno equivalente (mismo nº palabras, misma longitud y**

- **Decodificación con las matrices generadoras**

- **Sin embargo, tener codificado M(C)=[P I] nos da ventajas para la decodificación.**

- Usaremos la Matriz de Comprobación de Paridad **H(C)** para la decodificación.

- La Matriz H(C) tiene **n-k** filas y **n** columnas: $\mathbf{H_{n-k,n}(C)}$

- Se construye como: $H(C) = [I_{n-k} P^t_{k,n-k}]$

- Es decir, como la matriz identidad de tamaño **n-k** seguida de la **traspuesta de la matriz de paridad usada para la codificación.**

- En la decodificación, multiplicamos el código recibido **c** por la traspuesta de H(C) para comprobar si hay errores: $\mathbf{c \cdot H^T(C)}$
- **Posteriormente, lo decodificamos.**

– **El resultado de multiplicar $c \cdot H^T(C)$ es un vector s_{n-k}, con n-k bits.**

$$s_{n-k} = c \cdot H^T(C)$$

- **Al vector s_{n-k} se le denomina síndrome del código c.**

– El síndrome deberá ser el vector $(0)_{n-k}$ en caso de que no haya errores.

– En caso de error, **el síndrome también es capaz de indicar qué bits son los que se ven afectados por el error**, aunque siempre sujeto a la limitación del Teorema de Hamming para corrección de errores.

- **Para conocer los bits de error, es necesario calcular la tabla de síndromes y errores del código.**
- Esta tabla se puede calcular fácilmente de forma iterativa, calculando para errores de 1 bit en todos los bits posibles su síndrome.
- A continuación, pasar a errores de 2 bits, y así iterativamente hasta completar el total de errores corregibles según el Teorema de Hamming.
- **Si c es el código transmitido y c' es el código recibido, la decodificación pasa por:**
  - **Calcular su síndrome s(c')**
  - **Calcular el error cometido, si el síndrome es distinto de $(0)_{n-k}$**
  - **Calcular el error e del asociado al síndrome, según la tabla de síndromes y errores asociados.**
  - **Reconstruir el código c original como c=c'-e**

- **Un ejemplo para codificación y de codificación**

- ***Supongamos que queremos enviar mensajes de longitud = 4 bits, con capacidad para corregir errores de 1 bit.***

- **Construir una matriz generadora de un código lineal M(C)**

  - **Sabemos que tenemos que construir M(C)= $[P_{k,n-k}\ I_k]$**
  - Sabemos que **k=4**, porque es la longitud de nuestro código
  - **Para corregir 1 bit, sabemos también que el código debe tener una distancia de Hamming igual o superior a 2*1+1= 3**

- Tendremos que construir la matriz generadora sabiendo que la mínima distancia al vector 0 de cada fila debe ser de 3.

- Las filas deberán ser linealmente independientes.

- **Un ejemplo para codificación y de codificación**

- ***La siguiente matriz podría valer: Código lineal (7, 4)***

$$M(C)=\begin{pmatrix} ? & ? & ? & 1 & 0 & 0 & 0 \\ ? & ? & ? & 0 & 1 & 0 & 0 \\ ? & ? & ? & 0 & 0 & 1 & 0 \\ ? & ? & ? & 0 & 0 & 0 & 1 \end{pmatrix}$$

- Intentaremos también que P sea lo más distinta posible, pero consiguiendo siempre (al menos) una distancia d=3 al vector cero.

$$M(C)=\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

– Un ejemplo para codificación y de codificación

– Con la siguiente matriz:

$$M(C) = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

– Observamos que la hemos diseñado con la estructura: **M(C)= [$P_{k,n-k}$ I_k]**

– **La matriz P que se extrae de la matriz M(C) es:**

$$P_{k,n-k}(C) = P_{4,3} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

- **Un ejemplo para codificación y de codificación**

- ***Uso de la matriz para codificación de x=(x1, x2, x3, x4)***

$$x \cdot M(C) = (x_1 x_2 x_3 x_4) \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} = (c_1 c_2 c_3 c_4 c_5 c_6 c_7)$$

$$c_1 = x_1 + x_3 + x_4$$

$$c_2 = x_1 + x_2 + x_3$$

$$c_3 = x_2 + x_3 + x_4$$

$$c_4 = x_1$$

$$c_5 = x_2$$

$$c_6 = x_3$$

$$c_7 = x_4$$

- **Los bits c_1 a c_3 del código son de paridad.**

- **Un ejemplo para codificación y de codificación**

- ***Codificación de mensajes***

$$c_1 = x_1 + x_3 + x_4$$
$$c_2 = x_1 + x_2 + x_3$$
$$c_3 = x_2 + x_3 + x_4$$
$$c_4 = x_1$$
$$c_5 = x_2$$
$$c_6 = x_3$$
$$c_7 = x_4$$

(0000)·M(C)=(0000000)
(0001)·M(C)=(1101000)
(0010)·M(C)=(0110100)
…
(1011)·M(C)=(1001011)
…
(1100)·M(C)=(1011100)
…
(1111)·M(C)=(1111111)

- ***Podemos observar que el código resultante tiene una distancia de Hamming de 3, por lo que será posible encontrar un error en un bit.***

- **Un ejemplo para codificación y de codificación**

- ***Cálculo de síndromes:*** ***Como la distancia del código es d=3, entonces podemos corregir hasta (d-1)/2 errores; es decir con d=3 corregimos errores en 1 bit.***

  - Como el número de bits del código es n=7, hay 7 posibilidades de que se produzca un error de 1 bit:

    e1= (0000001)
    e2= (0000010)
    e3= (0000100)
    e4= (0001000)
    e5= (0010000)
    e6= (0100000)
    e7= (1000000)

  - Para calcular la tabla de síndromes y errores asociados, antes es necesario calcular $\mathbf{H^t(C)}$.

- **Un ejemplo para codificación y de codificación**

- ***Decodificación de mensajes: Construcción de la matriz H(C) para el control de errores:***

$$H(C) = [I_{n-k} P^t_{k,n-k}]$$

- ***Calculamos la traspuesta de P(C):***

$$P_{k,n-k}(C) = P_{4,3} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \Rightarrow P^t_{k,n-k}(C) = P^t_{4,3} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

- **Y diseñamos H(C):**

$$H(C) = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

- **Un ejemplo para codificación y de codificación**

- ***Calculamos la traspuesta de H(C):***

$$H^t(C) = \left(\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \hline 1 & 1 & 0 \\ 0 & 1 & 1 \\ \hline 1 & 1 & 1 \\ 1 & 0 & 1 \end{array}\right)$$

- ***Decodificación de mensajes: suponiendo un código $c=(c_1, c_2, c_3, c_4, c_5, c_6, c_7)$,***

¿cómo se calcula su síndrome s(c)? $s(c) = c \cdot H^t(C)$

- **Un ejemplo para codificación y de codificación**

- ***Supongamos, en el ejemplo que se envía c=(0000000), y que se produce un error en el primer bit, por lo que se recibe c'=(0000001)***

- ***¿Cuál sería su síndrome?*** $s(c) = c \cdot H^t(C)$

- **En nuestro ejemplo, s(c) tendría *n-k= 7-4= 3* bits, calculados así:**

$$(c_1c_2c_3c_4c_5c_6c_7)\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\\1&1&0\\0&1&1\\1&1&1\\1&0&1\end{pmatrix} = (c_1+c_4+c_6+c_7, c_2+c_4+c_5+c_6, c_3+c_5+c_6+c_7)$$

- **Un ejemplo para codificación y de codificación**

- ***Cálculo de la tabla de síndromes y errores asociados***

$$s(c) = (c_1 + c_4 + c_6 + c_7, c_2 + c_4 + c_5 + c_6, c_3 + c_5 + c_6 + c_7)$$

$$s(e1) = s(0000001) = (0+0+0+1, 0+0+0+0, 0+0+0+1) = (101)$$
$$s(e2) = s(0000010) = (0+0+1+0, 0+0+0+1, 0+0+1+0) = (111)$$
$$s(e3) = s(0000100) = (0+0+0+0, 0+0+1+0, 0+1+0+0) = (011)$$
$$s(e4) = s(0001000) = (0+1+0+0, 0+1+0+0, 0+0+0+0) = (110)$$
$$s(e5) = s(0010000) = (0+0+0+0, 0+0+0+0, 1+0+0+0) = (001)$$
$$s(e6) = s(0100000) = (0+0+0+0, 1+0+0+0, 0+0+0+0) = (010)$$
$$s(e7) = s(1000000) = (1+0+0+0, 0+0+0+0, 0+0+0+0) = (100)$$

- **Un ejemplo para codificación y de codificación**

- ***Cálculo de la tabla de síndromes y errores asociados***

Síndrome	Error
101	0000001
111	0000010
011	0000100
110	0001000
001	0010000
010	0100000
100	1000000

- ***Si calculamos el síndrome del código recibido c', podemos calcular el bit donde se ha producido el error, y recalcular el código inicial como***

$$c=c'-e$$

- Un ejemplo para codificación y de codificación

- *Ejemplo de detección y corrección de error:*
  - *Deseamos enviar el código (1011)*
  - *Lo codificamos en el mensaje a enviar c:*
    - $c = (1011) \cdot M(C) = (1001011)$
  - *Se produce un error de un bit en el canal, y se recibe:*
    $c' = (1001111)$
  - *Calculamos su síndrome:*
    $s(c') = c' \cdot H^t(C)$

    $s(c') = (1+1+1+1, 0+1+1+1, 0+1+1+1) = (0, 1, 1)$ → *Hay error.*

¿Cuál es el error? → *Según la tabla, e=(0000100)*
Corregimos: c'-e= (1001111)-(0000100)= (1001011)
Decodificamos: (1001011) → *1011*

- Otro ejemplo: Se desea transmitir los códigos {A, B, C, D} por un canal:

  - Construir un código uniforme con *k* bits que los represente.
  - Construir un código lineal (n,k) capaz de detectar y corregir 1 error.
  - Ejemplo de codificación
  - Ejemplo de decodificación
  - Ejemplo de detección y corrección de un error en un bit

– Otro ejemplo: Se desea transmitir los códigos {A, B, C, D} por un canal:

  – **Construir un código uniforme con *k* bits que los represente.**
  – Construir un código lineal (n,k) capaz de detectar y corregir 1 error.
  – Ejemplo de codificación
  – Ejemplo de decodificación
  – Ejemplo de detección y corrección de un error en un bit

- **Otro ejemplo: Se desea transmitir los códigos {A, B, C, D} por un canal:**

  - **Construir un código uniforme con *k* bits que los represente.**

  - Para codificar 4 símbolos, hacen falta k=2 bits. Por ejemplo, el código sería el siguiente:

    - **A: 00**
    - **B: 10**
    - **C: 01**
    - **D: 11**

– Otro ejemplo: Se desea transmitir los códigos {A, B, C, D} por un canal:

  – Construir un código uniforme con *k* bits que los represente.
  – **Construir un código lineal (n,k) capaz de detectar y corregir 1 error.**
  – Ejemplo de codificación
  – Ejemplo de decodificación
  – Ejemplo de detección y corrección de un error en un bit

– **Otro ejemplo: Se desea transmitir los códigos {A, B, C, D} por un canal:**

– **Construir un código lineal (n,k) capaz de detectar y corregir 1 error.**

– Como utilizamos k=2 bits, necesitamos diseñar **n**. Como tenemos que corregir errores en 1 bit, es necesario que la distancia del código ***d*** cumpla que (d-1)/2=1; es decir, **d>=3**. Así, necesitamos un **n** de, como mínimo, que nos pueda generar d=3. Diseñaremos la matriz M(C) del código como:

$$M(C) = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

– ¿Porqué? La distancia 1 se consigue por los *k* bits de la matriz identidad. Necesitamos otros 2 bits más que permitan d=3→ al menos 3 bits más nos permiten generar M(C).

– Total: **n=5 bits**

– **Otro ejemplo: Se desea transmitir los códigos {A, B, C, D} por un canal:**

– **Construir un código lineal (n,k) capaz de detectar y corregir 1 error.**

– Una vez calculada **M(C),** pasamos a calcular **H^t(C).**

$$H(C) = [I_{n-k} P^t_{k,n-k}]$$

– **n-k= 5-2= 3**

$$H(C) = \left(\begin{array}{ccccc|} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{array}\right)$$

$$H^t(C) = \left(\begin{array}{ccc|} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{array}\right)$$

– **Otro ejemplo: Se desea transmitir los códigos {A, B, C, D} por un canal:**

– **Construir un código lineal (n,k) capaz de detectar y corregir 1 error.**

– Errores posibles de 1 bit para palabras de tamaño **n=5:**

(00001)
(00010)
(00100)
(01000)
(10000)

– Tendremos que calcular la tabla de síndromes y sus errores asociados:

$$s(e) = e \cdot H^t(C)$$

– **Otro ejemplo: Se desea transmitir los códigos {A, B, C, D} por un canal:**

– **Construir un código lineal (n,k) capaz de detectar y corregir 1 error.**

– Tabla de síndromes, calculada como ***s(e)= e · H^t(C)***

$$e \cdot H^t(C) = (e_1 e_2 e_3 e_4 e_5) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

Síndrome	Error
101	00001
011	00010
001	00100
010	01000
100	10000

– Otro ejemplo: Se desea transmitir los códigos {A, B, C, D} por un canal:

– Construir un código uniforme con *k* bits que los represente.

– Construir un código lineal (n,k) capaz de detectar y corregir 1 error.

– **Ejemplo de codificación**

– Ejemplo de decodificación

– Ejemplo de detección y corrección de un error en un bit

– **Otro ejemplo: Se desea transmitir los códigos {A, B, C, D} por un canal:**

– **Ejemplo de codificación.**

– Se desea enviar el mensaje “DB”.

– **Paso 1: Codificación en bloque. D: 11, B: 10**
– **Paso 2: Inserción de redundancia (traducir a código lineal)**

$$M(C) = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

$$D \Rightarrow 11 \Rightarrow (11)\cdot M(C) = (11011)$$

$$B \Rightarrow 01 \Rightarrow (01)\cdot M(C) = (10101)$$

– **Paso 3: Se envía los bits por el canal**

1 1 0 1 1 1 0 1 0 1

– Otro ejemplo: Se desea transmitir los códigos {A, B, C, D} por un canal:

  – Construir un código uniforme con *k* bits que los represente.
  – Construir un código lineal (n,k) capaz de detectar y corregir 1 error.
  – Ejemplo de codificación
  – **Ejemplo de decodificación**
  – Ejemplo de detección y corrección de un error en un bit

– **Otro ejemplo: Se desea transmitir los códigos {A, B, C, D} por un canal:**

– **Ejemplo de decodificación.**

– El receptor recibe: **1 1 0 1 1 1 0 1 0 1**
– Como el tamaño del código lineal es n=5, se sabe que hay 2 símbolos:
  – **C1**→11011
  – **C2**→10101

– **Paso 1: Comprobación de error.** Calculamos los síndromes de cada mensaje por separado

s(C1)= (11011)·H^t(C)= (000) → No hay error
s(C2)= (10101)·H^t(C)= (000) → No hay error

$$S(c) = c \cdot H^t(C) = (c_1 c_2 c_3 c_4 c_5) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

– **Otro ejemplo: Se desea transmitir los códigos {A, B, C, D} por un canal:**

– **Ejemplo de decodificación.**

– Como el tamaño del código lineal es n=5, se sabe que hay 2 símbolos:
  – **C1**→11011
  – **C2**→10101

– **Paso 2: Quitamos redundancia. Nos quedamos con los k primeros bits de cada palabra recibida.**

Decodificar(C1)= Decodificar(11011) → 11
Decodificar(C2)= Decodificar(10101) → 01

– **Paso 3: Decodificamos los símbolos mediante su código de bloque. 11→ D; 01→ B**
– **Mensaje recibido: "DB"**

– **Otro ejemplo: Se desea transmitir los códigos {A, B, C, D} por un canal:**

– Construir un código uniforme con *k* bits que los represente.

– Construir un código lineal (n,k) capaz de detectar y corregir 1 error.

– Ejemplo de codificación

– Ejemplo de decodificación

– **Ejemplo de detección y corrección de un error en un bit**

– **Otro ejemplo: Se desea transmitir los códigos {A, B, C, D} por un canal:**

– **Ejemplo de detección y corrección de un error en un bit.**

– El emisor envía: **1 1 0 1 1 1 0 1 0 1**
– **… Pero el receptor recibe: 1 1 0 1 1 1 0 1 1 1**

– Como el tamaño del código lineal es n=5, se sabe que hay 2 símbolos:
  – **C1**→11011
  – **C2**→10111

– **Paso 1: Comprobación de error.**
– Calculamos los síndromes de cada mensaje

$$S(c) = (c_1 c_2 c_3 c_4 c_5) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

s(C1)= (11011)·H^t(C)= (000) → No hay error
s(C2)= (10111)·H^t(C)= (011) → Hay error

– Otro ejemplo: Se desea transmitir los códigos {A, B, C, D} por un canal:

– Ejemplo de detección y corrección de un error en un bit.

– Paso 2: Buscar el error en la tabla de síndromes

$s(C2)= (10111)\cdot H^t(C)= (011)$ → Hay error

Síndrome	Error
101	00001
011	**00010**
001	00100
010	01000
100	10000

- Otro ejemplo: Se desea transmitir los códigos {A, B, C, D} por un canal:

- Ejemplo de detección y corrección de un error en un bit.

  - Paso 3: Conociendo el error e=(00010) en el código C2=(10111)

    Corregir el error calculando C2-e

    C2'= (10111) – (00010) = (10101)

  - Paso 4: Quitamos redundancia. Nos quedamos con los k primeros bits de cada palabra recibida.

    Decodificar(C1)= Decodificar(11011) → 11

    Decodificar(C2')= Decodificar(10101) → 01

  - Paso 5: Decodificamos los símbolos mediante su código de bloque. 11→ D; 01→ B
  - Mensaje recibido: "DB"

- Los **códigos de Hamming** son también **códigos uniformes**; es decir, códigos cuyas palabras tienen una **longitud fija *n***.

- Se crean también incrementando los ***k*** bits del código de bloque de un símbolo con un conjunto de bits de redundancia ***r***, hasta llegar a formar palabras de longitud ***n***.

$$n = k + r$$

- **Son un tipo particular de códigos que sólo pueden utilizarse para detectar errores de 2 bits, y corregir errores de 1 bit.**

- **Son métodos de corrección de errores muy eficientes, en comparación con otros métodos computacionalmente más costosos de comprobación de errores.**

- **Aplicaciones:**
  - **Comprobación de errores de memoria RAM de un PC**, donde la probabilidad de error es muy baja.
  - **Corrección de caracteres ASCII**.

- **Generación de un código de Hamming**
  - Se deben insertar **bits de paridad par**, **pero estos se insertarán sólo en las posiciones que sean potencia de 2** (el primer bit, el segundo, el cuarto, el octavo…)
  - **Ejemplo: (D= bit de datos, P= bit de paridad)**

…DDDDPDDDDDDDPDDDPDPP

  - En el ejemplo, los bits de paridad expuestos son el 1, 2, 4, 8, 16 (de derecha a izquierda).
  - **Se puede calcular el número de datos que se pueden codificar con p bits de paridad:**

$$\text{nº bits de datos} = 2^p - p - 1$$

Ejemplo: Con 3 bits de paridad → $2^3-3-1=4$ bits de datos

- Generación de un código de Hamming
  - También se puede calcular el número total de bits que necesitaremos si utilizamos p bits de paridad:

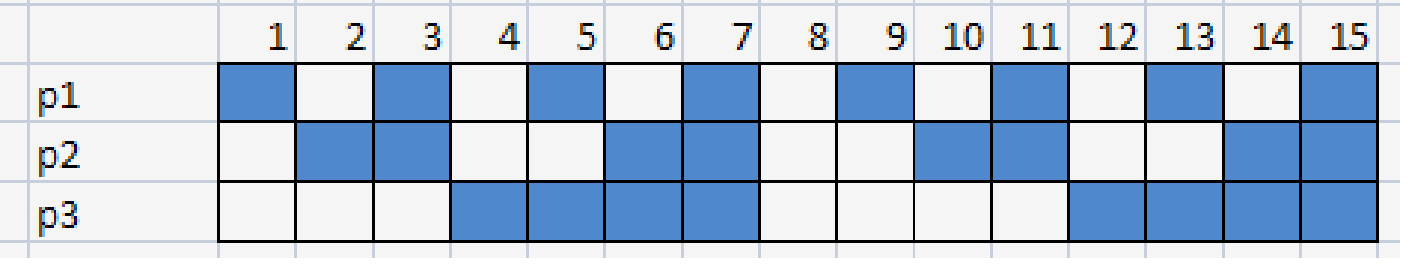
$$nº\ bits = 2^p - 1$$

Ejemplo: Con 3 bits de paridad → $2^3 - 1 = 7$ bits

  - Definiremos un código de Hamming (n, k) como un código de Hamming que utiliza un total de n bits, de los cuales k son para datos.
  - Ejemplo: Código de Hamming (7, 4). Mensajes de n=7 bits de los cuales k=4 serán de datos:

DDDPDPP

- **Generación de un código de Hamming**
  - La paridad se calcula:
    - **Para el bit de paridad 1** (posición 1): Desde él mismo, 1 bit sí, uno no, 1 bit sí, uno no…
    - **Para el bit de paridad 2** (posición 2): Desde él mismo, 2 bits sí, dos no, 2 bits sí, dos no…
    - **Para el bit de paridad 3** (posición 4): Desde él mismo, 4 bits sí, 4 no, 4 bits sí, 4 no…
    - **Para el bit de paridad 4** (posición 8): Desde él mismo, 8 bits sí, 8 no, 8 bits sí, 8 no…



- **Generación de un código de Hamming**

  - **Ejemplo de codificación** de un código de longitud **k=4** bits con un código de Hamming **(7,4)**

$$x = (x_4 x_3 x_2 x_1) \Rightarrow c = (x_4 x_3 x_2 p_3 x_1 p_2 p_1)$$

Ejemplo: x=(1101)

c= (1 1 0 p_3 1 p_2 p_1)

Calculando p_3 con paridad par para 1 1 0 p_3 → p_3 =0

c= (1 1 0 0 1 p_2 p_1)

Calculando p_2 con paridad par para 1 1 0 0 1 p_2 → p_2 =1

c= (1 1 0 0 1 1 p_1)

Calculando p_1 con paridad par para 1 1 0 0 1 1 p_1 → p_1 =0

c= (1 1 0 0 1 1 0)

- **Generación de un código de Hamming**

  - Un código de Hamming es un tipo particular de código lineal, por lo que también puede representarse mediante su forma matricial.

  - **¿Cómo calcular la matriz del código? Ejemplo para el código de Hamming (7,4).**

    Tenemos que el mensaje a enviar *x* ocupa 4 bytes, distribuidos como:

$$x = (x_4 x_3 x_2 x_1) \Rightarrow c = (x_4 x_3 x_2 p_3 x_1 p_2 p_1)$$

    La matriz generadora G deberá tener tamaño de k=4 filas y n=7 columnas.

- **Generación de un código de Hamming**
  - **¿Cómo calcular la matriz del código? Ejemplo para el código de Hamming (7,4).**

Las posiciones de los bits 3º, 5º, 6º y 7º son los bits del código, por lo que podemos diseñar parcialmente la matriz G:

$$(x_4x_3x_2x_1)\cdot G = (x_4x_3x_2p_3x_1p_2p_1) \Rightarrow G = \begin{pmatrix} 1 & 0 & 0 & ? & 0 & ? & ? \\ 0 & 1 & 0 & ? & 0 & ? & ? \\ 0 & 0 & 1 & ? & 0 & ? & ? \\ 0 & 0 & 0 & ? & 1 & ? & ? \end{pmatrix}$$

- **Generación de un código de Hamming**
  - **¿Cómo calcular la matriz del código? Ejemplo para el código de Hamming (7,4).**

Para el bit de paridad p_3, sabemos que debe tomar paridad par con los bits desde el 4º hasta el 7º. Es decir, consigo mismo y con los bits del código x_4, x_3 y x_2.

Podemos rellenar la columna de la matriz para calcular p_3:

$$p_3 = x_4 + x_3 + x_2$$

$$(x_4 x_3 x_2 x_1)\cdot G = (x_4 x_3 x_2 p_3 x_1 p_2 p_1) \Rightarrow G = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & ? & ? \\ 0 & 1 & 0 & 1 & 0 & ? & ? \\ 0 & 0 & 1 & 1 & 0 & ? & ? \\ 0 & 0 & 0 & 0 & 1 & ? & ? \end{pmatrix}$$

- **Generación de un código de Hamming**
  - **¿Cómo calcular la matriz del código? Ejemplo para el código de Hamming (7,4).**

Para el bit de paridad p_2, sabemos que debe tomar paridad par con los bits 2º (él mismo), 3º, 6º y 7º. Es decir, con los bits del código x_1, x_3 y x_4.

Podemos rellenar la columna de la matriz para calcular p_2:

$$p_2 = x_4 + x_3 + x_1$$

$$(x_4 x_3 x_2 x_1)\cdot G = (x_4 x_3 x_2 p_3 x_1 p_2 p_1) \Rightarrow G = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & ? \\ 0 & 1 & 0 & 1 & 0 & 1 & ? \\ 0 & 0 & 1 & 1 & 0 & 0 & ? \\ 0 & 0 & 0 & 0 & 1 & 1 & ? \end{pmatrix}$$

- **Generación de un código de Hamming**
  - **¿Cómo calcular la matriz del código? Ejemplo para el código de Hamming (7,4).**

Para el bit de paridad p_1, sabemos que debe tomar paridad par con los bits 1º (él mismo), 3º, 5º y 7º. Es decir, con los bits del código x_1, x_2 y x_4.

Podemos rellenar la columna de la matriz para calcular p_1:

$$p_1 = x_4 + x_2 + x_1$$

$$(x_4 x_3 x_2 x_1)\cdot G = (x_4 x_3 x_2 p_3 x_1 p_2 p_1) \Rightarrow G = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

- **Generación de un código de Hamming**

  - **Codificación utilizando la matriz generadora del código.**

Ejemplo: x=(1101)

$$c = x \cdot G = (1101) \cdot G = (1\ 1\ 0\ 0\ 1\ 1\ 0)$$

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

– **Generación de un código de Hamming**

Ya tenemos la matriz generadora (llamémosla ahora G') del código de Hamming (7,4).

Observamos cómo la distancia del código es de d=3 (3 bits a 1 en cada fila). Por tanto, verificamos que el código lineal puede corregir errores de 1 bit.

$$G' = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Podemos ver el código de Hamming como un código lineal. Sin embargo, esta matriz no está en forma canónica G= [I P]

Podemos transformarla a forma canónica permutando columnas.

- **Generación de un código de Hamming**
  - **¿Cómo calcular la matriz del código? Ejemplo para el código de Hamming (7,4).**

La permutación de columnas de la matriz únicamente afecta a la permutación de bits dentro del código→ Se generan códigos equivalentes, diferentes pero con las mismas propiedades.

Permutamos las columnas de G' para calcular G en forma canónica.

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

- **Generación de un código de Hamming**
  - **Codificación utilizando la matriz generadora del código en forma canónica.**

Ejemplo: x=(1101)

$$c = x \cdot G = (1101) \cdot G = (0\ 1\ 0\ 1\ 1\ 0\ 1)$$

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Podemos observar que el único cambio efectuado es la permutación de los bits del código resultante (el que tiene la redundancia).

- **Decodificación de un código de Hamming**
  - Volvamos a nuestra matriz original generadora del código **C** de Hamming (7,4).

$$G(C) = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

  - **¿Cómo decodificar?** Basta con comprobar los bits asociados al código de bloque original (en este caso, los bits 3º, 5º, 6º y 7º)
  - Se recibe c'= **(1 1 0 0 1 1 0) → se decodifica (1101)**
  - **¿Cómo saber si hay errores en la transmisión?** Con la matriz de control de paridad H(C).

- **Decodificación de un código de Hamming**
  - Cálculo de la matriz de control de paridad H(C):
    - Muy sencillo:
      1. Basta con incluir tantas columnas como bits de paridad.
      2. Tantas filas como longitud del código (**n**).
      3. Cada columna va asignada a un bit de paridad. **Se asignan de izquierda a derecha de menor a mayor:**

Ejemplo del código (7,4):

$$H(C) = \left(\begin{array}{ccc} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{array}\right) \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \end{array}$$

$$p_3 \; p_2 \; p_1$$

- **Decodificación de un código de Hamming**

  - Cálculo de la matriz de control de paridad H(C):

    4. **Por último, se rellenan las filas con los bits asignados para cada paridad, según el código diseñado.**

Ejemplo del código (7,4):

$$H(C) = \begin{array}{c} \\ x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \end{array} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{array}{c} \\ p_3 \\ p_2 \\ p_1 \end{array}$$

- **Decodificación de un código de Hamming**
  - **Cálculo del síndrome:**
    - El síndrome se calcula multiplicando el código recibido ***c'*** por la matriz: **s(c')= c'·H(C)**

Ejemplo del código (7,4):

Se recibe c'= (1100110)

$$c'\cdot H(C) = (1100110)\cdot H(C) = (1100110)\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = (000)$$

Síndrome (000)= No hay errores

- **Decodificación de un código de Hamming**
  - **Cálculo del error:**
    - **En códigos de Hamming**, por cómo está diseñada H(C), el síndrome proporciona una cadena de bits que se pueden interpretar como un número entero.
    - Este número indica el bit en el que se recibe el error.
    - **¡No necesitamos tablas de síndromes!**

Ejemplo del código (7,4):
Se envía c = (1100110)
Se recibe c'=(1110110)

$$c' \cdot H(C) = (1110110) \cdot H(C) = (1110110) \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = (101)$$

Según el síndrome, el error está en el bit 5
Error e=(0010000)
Corrección: c=c'-e=(1110110)-(0010000)= (1100110)

- **Decodificación de un código de Hamming**
  - **Cálculo del error:**

Otro ejemplo. **Se envía c = (1100110)**
Se recibe c'=(1100100)

Según el síndrome, el error está en el bit 2
Error e=(0000010)
Corrección: c=c'-e= (1100100)-(0000010)= (1100110)

$$c' \cdot H(C) = (1100100) \cdot H(C) = (1100100) \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = (010)$$

- **Decodificación de un código de Hamming**
  - **Cálculo del error:**

Otro ejemplo. **Se envía c = (1100110)**
Se recibe c'=(0100110)

Según el síndrome, el error está en el bit 7
Error e=(1000000)
Corrección: c=c'-e= (0100110)-(1000000)= (1100110)

$$c' \cdot H(C) = (0100110) \cdot H(C) = (0100110) \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = (111)$$

- **Decodificación de un código de Hamming para detección de 2 errores**
  - **Se incluye un último bit de paridad en la matriz generadora**

En nuestro ejemplo de código Hamming (7,4):

$$G(C) = \left(\begin{array}{ccccccc|c} 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right)$$

Al decodificar:

- Si el síndrome es 0 y el último bit de paridad es correcto, no hay errores.
- Si el bit de paridad está mal, hay error en un bit. Decodificar normalmente.
- Si el síndrome es distinto de 0 y la paridad está bien, entonces hay un número par de errores. No se decodifica.

- Los **códigos cíclicos** pertenecen a la familia de **códigos lineales,**
  - Un código se dice cíclico si, al rotar 1 posición una palabra del código, la palabra resultante también pertenece al código.
- **Ejemplo:**
  - Supongamos una palabra del código cíclico **C** de longitud **n**:

$$c = (c_1 c_2 c_2 \ldots c_n)$$

Entonces, la palabra **b** resultante de rotar **c** una posición, también es del código:

$$b = (c_n c_1 c_2 c_2 \ldots c_{n-1})$$

- Como un código cíclico es también un código lineal, tienen una **matriz generadora G.**

- **¿Porqué se llaman códigos cíclicos?**

  - **Si una palabra pertenece al código, entonces un desplazamiento a izquierda o derecha de la palabra también es otra palabra del código.**

  - **Ejemplo:**

    Si (1001101) es palabra del código, entonces (1100110) también es palabra del código.

- La matriz generadora del código **C**, **G(C)**, se obtiene también a partir de un polinomio **g(x)**, que también se llama **polinomio generador.**

- Este polinomio es de la forma:

$$g(x) = x^{n-k} + \ldots + 1$$

donde **n** es la longitud de la palabra y **k** el número de bits del código de bloque original (hay **k** bits de datos en el código cíclico).

- **Los coeficientes del polinomio g(x) están en Z_2, el conjunto {0, 1}**

- **Si conocemos el polinomio generador, es fácil construir la matriz generadora G.**

- De este modo, las palabras de un código también se representan como polinomios.

- **Sea la palabra $c=(c_0\ c_1\ c_2\ c_3\ \ldots\ c_n)$, entonces también se puede representar como el polinomio.**

$$c(x)= c_0 +c_1x +c_2x^2 +c_3x^3 +\ldots +c_nx^n$$

- También se modelan como polinomios los códigos de bloque iniciales a transformar a código cíclico. Si, por ejemplo, tenemos un código de bloque **b** de **k** bits ($b_0, b_1, b_2, \ldots, b_{k-1}$) que queremos pasar a un código **c(x)** cíclico, se calcularía como:

$$c(x)= b(x)*g(x)$$

Donde **b(x) es el polinomio modelado en el código de bloque inicial:**

$$b(x)= b_0 + b_1x + b_2x^2 +\ldots+ b_{k-1}\ x^{k-1}$$

-

– Realmente, es lo mismo que veníamos haciendo hasta ahora, con la salvedad de que el modelo subyacente del código que ahora representamos utiliza polinomios como mecanismos de representación:

– **Ejemplo: Código de bloque b de tamaño k=4**

$b=(b_0\ b_1\ b_2\ b_3)$ → **Ejemplo de palabra: (0101)**
→ **Polinomio asociado: $b(x)= x + x^3$**

¿Cómo diseñamos polinomios generadores para un código?

Asumiremos códigos de longitud **n** con un número de bits de datos **k** (códigos cíclicos **C(n,k)**).

– **¿Cómo diseñamos polinomios generadores para un código?**

Suponiendo un **código C(n,k)**, los polinomios generadores candidatos son aquellos que sean **factores de x^n+1**.

Nunca deberán tener un grado superior a n-k

Ejemplo: Para cualquier código (7, k), sus polinomios generadores deberán ser factores de x^7+1.

Factorizamos $x^7+1 \rightarrow x^7+1 = (x+1) \cdot (x^3+x+1) \cdot (x^3+x^2+1)$

– **¿Cómo diseñamos polinomios generadores para un código?**

Suponiendo un **código C(n,k)**, los polinomios generadores candidatos son aquellos que sean **factores de x^n+1**.

Nunca deberán tener un grado superior a n-k

Ejemplo: Para cualquier código (7, k), sus polinomios generadores deberán ser factores de x^7+1.

Factorizamos $x^7+1 \rightarrow x^7+1 = (x+1) \cdot (x^3+x+1) \cdot (x^3+x^2+1)$

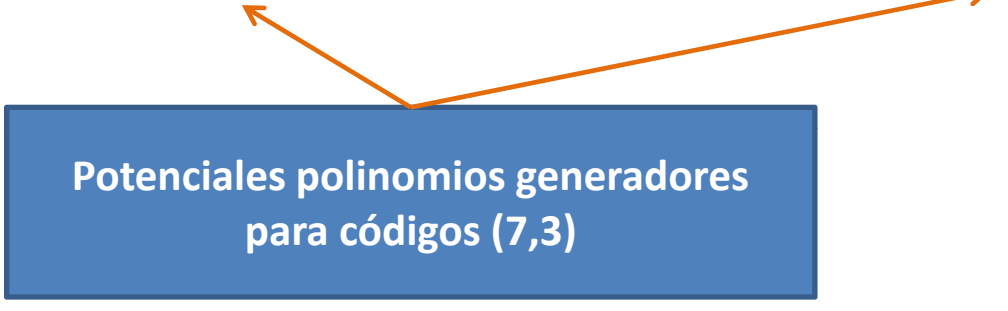
Potenciales polinomios generadores para códigos (7,1) y (7,4), respectivamente

– **¿Cómo diseñamos polinomios generadores para un código?**

Si quisiésemos diseñar un (7,3), bastaría con multiplicar los factores entre sí para dar un polinomio de grado 7-3=4.

$x^7+1 = (x+1) \cdot (x^3+x+1) \cdot (x^3+x^2+1)$

$(x+1) \cdot (x^3+x+1) = x^4+x^3+x^2+x$ $\qquad$ $(x+1) \cdot (x^3+x^2+1) = x^4+x^2+x+1$



¿Cómo codificamos con un código cíclico?

Basta con multiplicar el polinomio del código de bloque inicial, b(x), por el polinomio generador: c(x)= b(x)*g(x)

– **¿Cómo codificamos con un código cíclico?**

Ejemplo: Codificación del mensaje original b=(1001) en un código C(7, 4) con el polinomio generador g(x)= x^3+x^2+1.

b= (1001) → b(x) =x^3+1

c(x)= b(x)*g(x)= $(x^3+1)(x^3+x^2+1)=$

$$x^3(x^3+x^2+1) +$$
$$+ 0*(x^3+x^2+1)+$$
$$+ 0*(x^3+x^2+1)+$$
$$+1*(x^3+x^2+1)=$$
$$x^6+x^5+x^3+$$
$$+ x^3+x^2+1= x^6+x^5+(x^3+ x^3)+ x^2 +1=$$
$$= x^6+x^5+ x^2 +1$$

c(x)= $x^6+x^5+ x^2 +1$ → c= (1100101) → (**NOTA:** se cogen los coeficientes del polinomio)

A continuación estudiamos diferentes tipos de códigos cíclicos.

– **¿Cómo decodificamos con un código cíclico?**

Dividimos el código por el polinomio generador

$b(x)= c(x)/g(x)$

Ejemplo: c= (1100101) → $c(x)= x^6+x^5+ x^2 +1$

$c(x) / g(x)= (x^6+x^5+ x^2 +1) / (x^3+x^2+1)$ → $b(x)= (x^3+1)$

Decodificamos c(x) como b(x)

$b(x)= (x^3+1)$ → **Mensaje original b= (1001)**

- Los **códigos BCH** pertenecen a la familia de **códigos cíclicos, subconjunto de códigos lineales.**
- Operan bajo las siguientes condiciones:
  - La longitud del código debe tener **$n=2^m-1$ bits, con m>=3**
  - Así, se pueden utilizar **k** bits de datos, con **$k>= n-m \cdot t$**
  - **Son capaces de corregir t errores.**
  - Como hay **k** bits de datos, entonces hay **n-k** bits de redundancia.
- Notaremos a un código BCH como código BCH **(n, k, t)**.
- **Ejemplos:**
  - **Código BCH (7, 4, 1)** → 7 bits totales, con 4 de datos. Corrige 1 error.
  - **Código BCH (15, 11, 1)** → 15 bits totales, 11 para datos. Corrige 1 error.
  - **Código BCH (15, 7, 2)** → 15 bits totales, 7 para datos. Corrige 2 errores.
  - **Código BCH (15, 5, 3)** → 15 bits totales, 5 para datos. Corrige 3 errores.

- Como son códigos cíclicos, tienen un **polinomio generador.**

- **Ejemplos:**
  - **Código BCH (7, 4, 1)** → $p(x)= x^3+x+1$
    - Corrige 1 error→ 2*1+1= 3 bits de paridad
  - **Código BCH (15, 11, 1)** → $p(x)= x^4+x+1$
    - Corrige 1 error→ 2*1+1= 3 bits de paridad
  - **Código BCH (15, 7, 2)** → $p(x)= x^8+x^7+x^6+x^4+1$
    - Corrige 2 errores→ 2*2+1= 5 bits de paridad
  - **Código BCH (15, 5, 3)** → $p(x)= x^{10}+x^8+x^5+x^4+x^2+x+1$
    - Corrige 3 errores→ 2*3+1= 7 bits de paridad

- Los códigos BCH se estudian como una **generalización de los códigos de Hamming para corrección de múltiples errores.**

- **Generación de polinomios generadores de códigos BCH:**

  - Se toman las raíces del polinomio x^n-1, con **coeficientes módulo 2**, donde **n** es la longitud de las palabras del código.

  - Tienen unas propiedades que hacen que el polinomio generador para un código sea único.

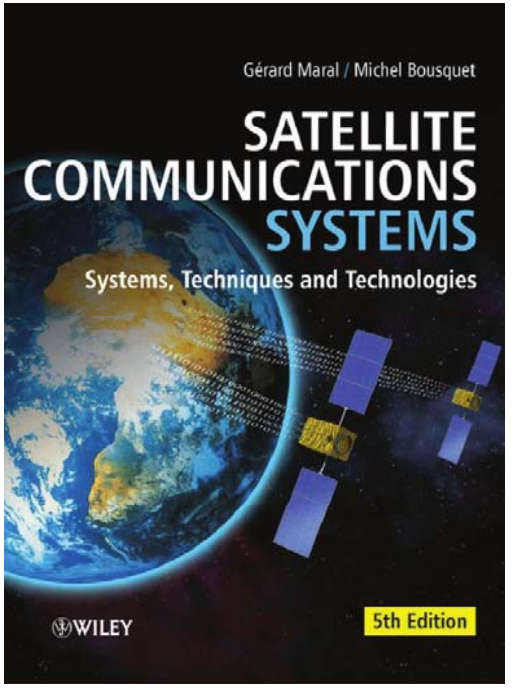
  - Ejemplo para códigos de longitud **n=3**

$$x^3-1= (x+1)(x^2 + x + 1) = x3+x2+x+x2+x+1$$

– Debido a su robustez, son ampliamente utilizados en:

Comunicaciones por satélite (incluida TV por satélite).

Codificación de DVDs



Control de errores de discos duros

Comunicaciones con sondas espaciales

- Los **códigos de Reed-Muller** pertenecen a la familia de **códigos cíclicos, subconjunto de códigos lineales.**

- Notaremos a un código Reed-Muller como código **RM(r, m)**.
  - **Esto quiere decir "Código Reed-Muller de orden r y longitud 2^m** (las palabras del código tendrán longitud **2^m**).
  - **Debe cumplirse siempre la siguiente condición:**

$$0 <= r <= m$$

  - **Propiedad: La distancia de un código RM(r, m) es $d=2^{m-r}$**

- **Ejemplos:**
  - RM(0,0) → Palabras de longitud $2^0=1$ → código {0, 1]
  - RM(0,1) → 2 palabras de longitud 2→ código {00, 11}
  - RM(1, 1) → 4 palabras de longitud 2 → {00, 01, 10, 11}

– **La definición de un código de Reed-Muller se realiza de forma recursiva.**

Partiendo de los casos base y general:

RM(0, m) = { $(0)^m$, $(1)^m$ } → 2 palabras de longitud **m**

RM(r,m)= { {x,x+y} | x está en RM(r, m-1) e y está en RM(r-1, m-1) }

– **Ejemplos: Supongamos ya conocidos los códigos RM(1, 1) y RM(0, 1)**

  – RM(0,1) → 2 palabras de longitud 2→ código {00, 11}
  – RM(1, 1) → 4 palabras de longitud 2 → {00, 01, 10, 11}

¿Cómo se calcularía el código RM(1, 2)

RM(1,2)= { {x,x+y} | x está en RM(1, 1) e y está en RM(0, 1) }
RM(1,2)= {(x, x+y) | x está en {00, 01, 10, 11} e y está en {00, 11}}

- **La definición de un código de Reed-Muller por su matriz generadora.**

- Como RM son códigos cíclicos y estos, a su vez, son códigos lineales, tienen una matriz generadora G(C).

- El cálculo de esta matriz generadora también se realiza de forma recursiva. Por simplicidad a la hora de la definición, notaremos **G(r,m)= G(C):**

$$G(r,m) = \begin{pmatrix} G(r,m-1) & G(r,m-1) \\ 0 & G(r-1,m-1) \end{pmatrix}$$

- **Con los casos particulares:**

$$G(0,m) = (1)^m$$

$$G(m,m) = \begin{pmatrix} G(m-1,m) \\ 0\ldots01 \end{pmatrix}$$

- **La definición de un código de Reed-Muller por su matriz generadora.**

- **Ejemplo: Matriz generadora de R(0,1)**

$$G(0,1) = \begin{pmatrix} 1 & 1 \end{pmatrix}$$

- **Ejemplo: Matriz generadora de R(1,1)**

$$G(1,1) = \begin{pmatrix} G(0,1) \\ 0...01 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

- **Ejemplo: Matriz generadora de R(1,2)**

$$G(1,2) = \begin{pmatrix} G(1,1) & G(1,1) \\ 0 & G(0,1) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

- La definición de un código de Reed-Muller por su matriz generadora.

- Ejemplo: Matriz generadora de R(2,2)

$$G(2,2) = \begin{pmatrix} G(1,2) \\ 0\ldots01 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Ejemplo: Matriz generadora de R(1,3)

$$G(1,3) = \begin{pmatrix} G(1,2) & G(1,2) \\ 0 & G(0,2) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

- **Otros códigos cíclicos:**
  - **Códigos de Reed-Solomon**
    - Tipo de códigos BCH
  - **Códigos Goppa**
    - Basados en propiedades geométricas de los polinomios.

Universidad de Granada

decsai.ugr.es

Teoría de la Información y la Codificación

Grado en Ingeniería Informática

Tema 5.- Teoría y desarrollo de códigos detectores y correctores de errores.

DECSAI

Departamento de Ciencias de la Computación e Inteligencia Artificial