Miscelánea Matemática **80** (2025) 35-45 SMM

DOI: https://doi.org/10.47234/mm.8004

Códigos que se corrigen: «se vale equivocarse, pero hay que corregir»

Carlos Bosch Giral
Departamento Académico de Matemáticas
ITAM
bosch@itam.mx
y
Lucía Ramírez David
Departamento Académico de Matemáticas
ITAM
lucia.ramirez@itam.mx

1. Introducción

El matemático estadounidense Richard Hamming fue pionero en el cómputo científico. Después de su doctorado en 1945, trabajó en el proyecto Manhattan en Los Álamos. En ese entonces, las máquinas IBM se usaban para hacer distintos cálculos sobre la bomba atómica, en particular, para saber si al explotar la bomba, la atmósfera se incendiaría. Al año siguiente, en 1946, Hamming se unió al equipo de los laboratorios Bell. [5]

Un viernes de 1947, el Dr. Hamming programó las computadoras para que efectuaran cierto cálculo y se fue de fin de semana dejando a las máquinas trabajando. De regreso el lunes, se sorprendió cuando llegó a recoger los cálculos y se dio cuenta que las computadoras no habían efectuado la tarea asignada. Las máquinas digitales procesan la información proporcionada con unos y ceros. En esa época si algún dígito en la secuencia se enviaba equivocadamente la máquina lo detectaba, pero la tarea asignada no se llevaba a cabo. En palabras de Hamming: «si la computadora puede indicar cuando ocurrió un error, también debería poderlo corregir». Ante esta afirmación, él mismo se dio a la tarea de resolver este problema y entonces dedicó sus fines de semana para poder trabajar con las máquinas sin interferencia de otros usuarios, y

Palabras clave: códigos que se corrigen, códigos de Hamming, geometría proyectiva, paridad.

así, tratar de hacer que la máquina además de detectar, pudiese corregir algún error. En los años sesenta y setenta, para programar las máquinas, se usaban las tarjetas perforadas que son, a final de cuentas unas cartulinas con un código binario donde se almacenan y ordenan datos para el funcionamiento de las computadoras. Las máquinas lectoras de tarjetas lo hacían a razón de 300 a 3000 tarjetas por minuto. En las figuras siguientes aparecen una tarjeta y un programa donde se hacían marcas especiales para que el orden no se perdiera. Cada vez que había algún error la máquina detectaba en qué tarjeta estaba y se debía perforar esa tarjeta nuevamente.

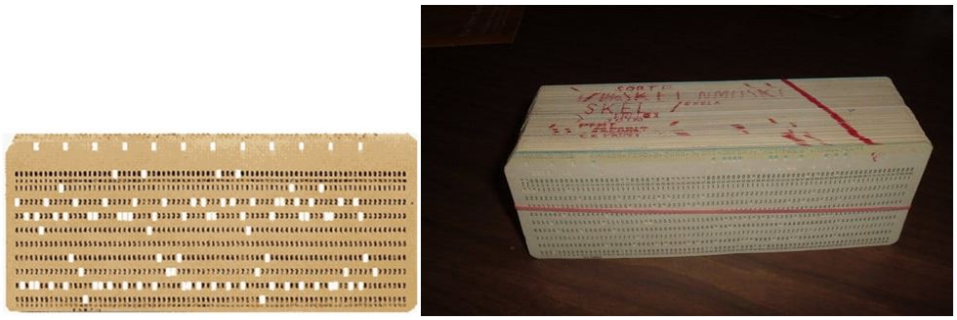

Figura 1. Tarjetas perforadas que se usaban en los años sesentas y setentas para la programación de máquinas.

Actualmente la mayoría de las transmisiones electrónicas se hacen a través de líneas telefónicas, fibra óptica u otros medios. Esto se hace con el envío sucesivo de bits formados por ceros y unos. Se ha medido que la frecuencia de errores de transmisión es muy pequeña, del orden de 10^{-5} para una línea telefónica y de 10^{-11} para la fibra óptica, es decir que casi no hay errores. Nos preguntamos entonces: ¿cómo pudo, Richard Hamming, llegar a corregir los errores que se generan durante la transmisión? La respuesta se encuentra a continuación y son precisamente los códigos que se autocorrigen, los llamados códigos de Hamming [1], [2], [4].

Pero antes de empezar fijemos un poco las ideas. En este caso entenderemos un código como los signos que componen el mensaje. Seamos más claros, los signos son unos y ceros y con ellos se forma el mensaje, es decir la información que se quiere transmitir. Además pediremos la condición de que al transmitir el mensaje, al enviarlo, este debe ser entendible por el receptor.

2. Primeros intentos

Podemos ilustrar un canal de comunicación con el siguiente esquema [2]:

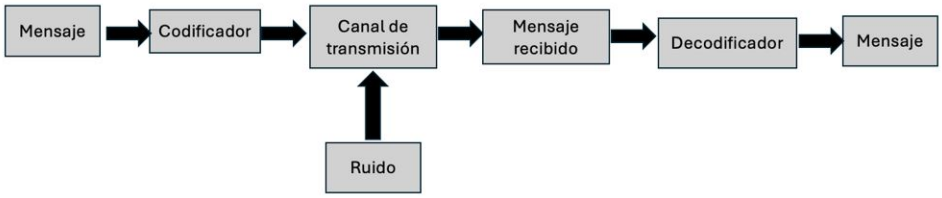

Figura 2. Canal de comunicación.

La meta es crear un sistema que permita codificar el mensaje antes de la transmisión y, decodificarlo al ser recibido antes de la lectura. De manera que, si se distorsiona «un poco» el mensaje en el canal, el decodificador pueda detectar el error y de ser posible corregirlo.

Ilustraremos tres formas de hacerlo: primero con repeticiones luego con un bit de paridad y finalmente con un código auto corrector. Para eso las palabras con las que trabajaremos serán palabras de cuatro bits.

a) Repetición de bits:
Es claro que, si repetimos una palabra varias veces, al comparar una con otra sabremos si hay algún error en la transmisión. Sin embargo, mandar dos veces una palabra no nos permite detectar si hay o no un error pues si enviamos 1001 y 1000 no sabemos si el error está en los primeros cuatro bits o en los siguientes. Si se quiere usar este método se debe enviar al menos tres veces la palabra y así por comparación será más fácil detectar si existe un error. En este caso, hay que enviar al menos 12 bits para una palabra de cuatro bits. La repetición del envío de la palabra tres veces hace que la transmisión sea larga y poco eficiente, pero se puede detectar un error.

b) Bit de paridad:
Otra forma de enviar una palabra y poder detectar si hay algún error es usando un bit de paridad. Es decir, en este caso se enviará una palabra de cinco bits donde los primeros cuatro son la palabra y el último es el de paridad. Este último bit es cero si la suma de los dígitos que forman la palabra es par y uno si la suma es impar, es decir que siempre hace que la palabra de cinco bits tenga una suma de dígitos par. Por ejemplo, si la palabra que se desea transmitir es 1001 se envía 10010 donde el último bit es el bit de paridad, y si queremos transmitir 1101 se envía 11011, el último bit es el de paridad. Veamos un ejemplo de cómo funciona. Si

se quiere transmitir la palabra 1001 se enviará 10010, pero si se recibe 10011 es claro que hay un error pues el bit de paridad es 1 y la suma de los bits de la palabra es par. Lo que aquí no se puede detectar es dónde está el error. Además, si hay un número par de errores en la transmisión, tampoco se detectará el error pues la paridad se preserva. Al colocar un bit de paridad podemos detectar un número impar de errores, pero no se sabe dónde está el error.

c) Código de Hamming:
Para generar un código de Hamming se requieren dos tipos de bits:

- Los bits de datos (o mensaje)
- Los bits de paridad, que en el caso de palabras de cuatro bits serán tres.

Los bits de paridad se envían junto con los datos y son los que permiten detectar y corregir errores. Esto se ilustra en la siguiente sección mediante un ejemplo.

3. Código de Hamming (7,4)

Un código de Hamming usa únicamente unos y ceros y la longitud del mensaje es el número total de ceros y unos que tiene. Por ejemplo 101 tiene longitud tres y 00101 tiene longitud cinco. El código (7,4) usa mensajes de longitud siete o siete bits para codificar palabras de cuatro bits. Veamos a grandes rasgos el trabajo que hizo Hamming para que al enviar un código de cuatro bits y añadirle de manera muy ingeniosa tres bits de paridad el código resulte auto corregible. En el caso del código (7,4) se envían 7 bits (longitud siete): los 4 primeros bits corresponden a los bits de datos, este es el mensaje que se quiere enviar, y los siguientes 3 bits a los de paridad. En general se codificará una palabra D1,D2,D3,D4 (posición de los bits de datos) como
D1, D2, D3, D4, P1, P2, P3, donde:

- P1 es el bit de paridad de D1, D2 y D4
- P2 es el bit de paridad de D1, D2 y D3
- P3 es el bit de paridad de D1, D3 y D4

Podemos resumir esto con un diagrama de Venn

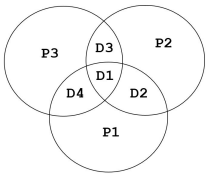

Figura 3. Diagrama de Venn para calcular bits de paridad.

Veamos un ejemplo, queremos enviar el mensaje 1101 Se codifica

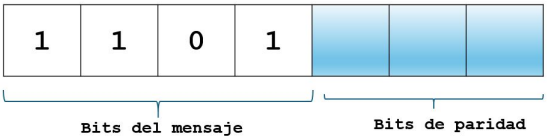

Figura 4. Bits del mensaje.

asignando a cada bit de paridad: cero si la suma de los bits dentro del círculo del diagrama de Venn es par y uno si la suma es impar.

- P1 es el bit de paridad de D1, D2 y D4, asi que P1 es 1.
- P2 es el bit de paridad de D1, D2 y D3, asi que P2 es 0.
- P3 es el bit de paridad de D1, D3 y D4, asi que P3 es 0.

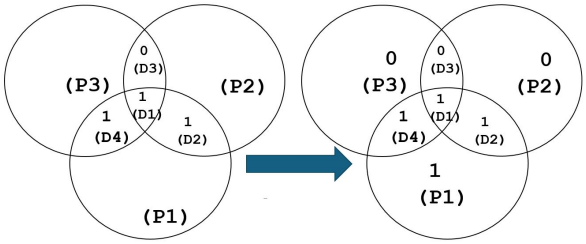

Figura 5. Calculando los bits de paridad.

- Se envía el mensaje:

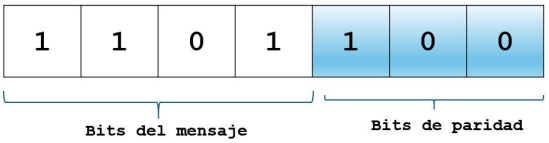

Figura 6. Mensaje enviado.

- Se recibe el mensaje siguiente: sabemos que tiene un error (puesto

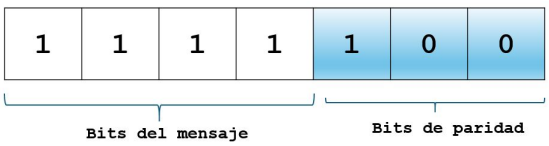

Figura 7. Mensaje recibido (con un error).

que P2 y P3 no corresponden a la paridad necesaria)
- Se decodifica usando un diagrama de Venn

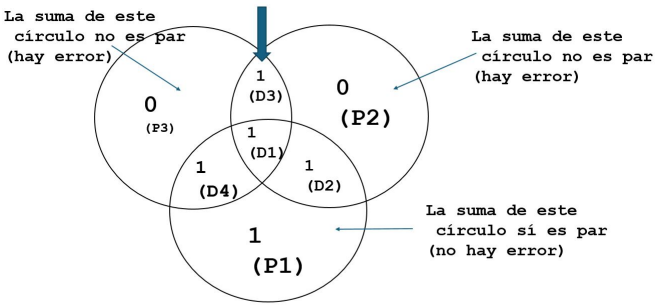

Figura 8. Decodificación.

Podemos entonces concluir que el error se produjo en D3. Este tipo de código permite detectar un error y corregirlo, y en el caso de producirse dos errores los puede detectar mas no corregir.
Observemos que no todas las palabras de 7 bits pertenecen al código (7,4) de Hamming. Por ejemplo la palabra 1000100 o 111100 no son una palabra del código de Hamming

4. La geometría y el código (7,4)

Desde que Descartes fundamentó las relaciones entre la geometría y el álgebra una de estas áreas no ha dejado de apoyarse en la otra y vice versa. Uno de estos ejemplos es la relación que guardan la geometría proyectiva y la teoría de códigos. Incluso hay artículos como «The packing problem in statistics, coding theory and finite projective spaces» por Hirschfeld y Storme [3] que relacionan la geometría proyectiva con la estadística y la teoría de códigos. Aquí veremos de manera muy sencilla la relación entre el plano proyectivo de siete puntos y el código de Hamming (7,4).
Un plano proyectivo se puede definir con cuatro axiomas de incidencia (entre rectas y puntos):

1. *Dos puntos determinan una única recta.*
2. *En cada recta hay al menos tres puntos.*
3. *Hay tres puntos no alineados.*
4. *Dos rectas diferentes se cortan en un punto.*

Aquí nos vamos a enfocar en el plano proyectivo de 7 puntos debido a que estamos trabajando con códigos con siete bits. Ese plano se puede representar de la siguiente manera. Observemos que la curva 3,1,5 representa una recta y con eso se cumplen todos los axiomas de la geometría proyectiva. Hay 7 puntos y 7 rectas, por cada punto pasan tres rectas y cada recta tiene tres puntos.

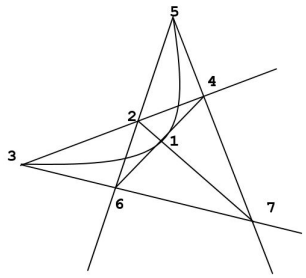

Figura 9. Plano proyectivo de 7 puntos.

Gracias a esos números vamos a construir el código de Hamming (7,4). Consideremos una de las líneas, por ejemplo, la que pasa por los puntos 4,5 y 7. Para construir una palabra colocamos en los lugares 4, 5 y 7 ceros y unos en los demás, es decir que obtendremos 1110010. Como hay 7 líneas tendremos siete palabras. Cada una de esas palabras está compuesta por tres ceros y cuatro unos. Tomaremos además de esas palabras las palabras duales es decir las palabras donde se sustituye un cero por un 1 y un uno por un 0. Es decir, si tenemos 1011001 también estará la palabra 0100110. De esta manera tendremos siete palabras más es decir 14 palabras y con 0000000 y 1111111 completaremos las 16 palabras del código (7,4) de Hamming. Nuevamente no todas las palabras de 7 bits están en el código de Hamming. Por ejemplo la palabra 0101110 no está pues correspondería a los puntos 137 y esos puntos no corresponden a una recta en el plano proyectivo. Observemos

Puntos / Rectas	1	2	3	4	5	6	7
	0	0	0	0	0	0	0
Dual 457	0	0	0	1	1	0	1
Dual 367	0	0	1	0	0	1	1
127	0	0	1	1	1	1	0
Dual 256	0	1	0	0	1	1	0
135	0	1	0	1	0	1	1
146	0	1	1	0	1	0	1
Dual 234	0	1	1	1	0	0	0
234	1	0	0	0	1	1	1
Dual 146	1	0	0	1	0	1	0
Dual 457	0	0	0	1	1	0	1
256	1	0	1	1	0	0	1
Dual 127	1	1	0	0	0	0	1
367	1	1	0	1	1	0	0
457	1	1	1	0	0	1	0
	1	1	1	1	1	1	1
Bits de código	D1	D2	D3	D4	P1	P2	P3

Figura 10. Palabras del código de Hamming obtenidas usando las rectas del plano proyectivo.

que el 124 no corresponde a una recta, así como la palabra que le correspondería, es decir 1101000 no es una palabra en el código de Hamming.

5. Para divertirse

Es claro que hay una gran cantidad de aplicaciones que usan este tipo de códigos. Para finalizar, vamos a describir un juego:

- En este juego participan dos personas: un jugador y un adivinador.
- El jugador elige un número entre 0 y 15 sin revelarlo.
- El adivinador realizará siete veces la misma pregunta al jugador: «¿el número que eligió aparece en la tarjeta que muestro?»

TARJETA 1	TARJETA 2	TARJETA 3	TARJETA 4	TARJETA 5	TARJETA 6	TARJETA 7
8	4	2	1	1	2	1
9	5	3	3	3	3	2
10	6	6	5	4	4	5
11	7	7	7	6	5	6
12	12	10	9	8	8	8
13	13	11	11	10	9	11
14	14	14	13	13	14	12
15	15	15	15	15	15	15

Figura 11. Tarjetas que muestra el adivinador.

- El jugador solo puede responder «Sí» o «No».
- El adivinador presentará las tarjetas de manera secuencial iniciando con la tarjeta 1, luego la 2, etcétera, y anotará un ?1? si la respuesta del jugador es afirmativa («sí») o un «0» si es negativa («no»). Este procedimiento generará una palabra binaria de 7 bits.
- El jugador puede decir la verdad en todas las respuestas o mentir en a lo más una de ellas, sin avisarle al adivinador.
- El objetivo del adivinador es descubrir el número elegido y además determinar si el jugador mintió, y de ser así en qué tarjeta lo hizo.

Si el participante no miente, el adivinador revelará el número que escogió el jugador usando la siguiente tabla: en el caso que el partici-

Tarjeta1	Tarjeta 2	Tarjeta 3	Tarjeta 4	Tarjeta 5	Tarjeta 6	Tarjeta 7	El número pensado es
0	0	0	0	0	0	0	**0**
0	0	0	1	1	0	1	**1**
0	0	1	0	0	1	1	**2**
0	0	1	1	1	1	0	**3**
0	1	0	0	1	1	0	**4**
0	1	0	1	0	1	1	**5**
0	1	1	0	1	0	1	**6**
0	1	1	1	0	0	0	**7**
1	0	0	0	1	1	1	**8**
1	0	0	1	0	1	0	**9**
1	0	1	0	1	0	0	**10**
1	0	1	1	0	0	1	**11**
1	1	0	0	0	0	1	**12**
1	1	0	1	1	0	0	**13**
1	1	1	0	0	1	0	**14**
1	1	1	1	1	1	1	**15**

Figura 12. Tabla para adivinar el número si el jugador no miente.

pante haya mentido en exactamente una de las tarjetas, su palabra no se encontrará en la lista de arriba. Por ejemplo, 1000001 no está en la lista de palabras admitidas, así que el adivinador debe descifrarla, usando el método descrito anteriormente, es decir usando el código de Hamming.

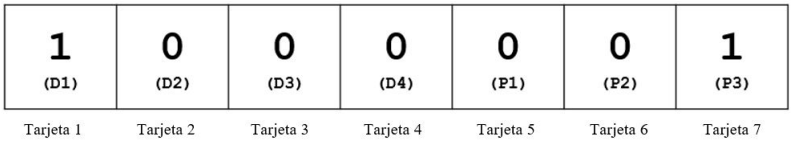

Figura 13. Secuencia de respuestas que da el jugador.

El dígito erróneo es el que pertenece a los dos círculos dónde se detectó un error, es decir el participante mintió en la tarjeta 2. La palabra binaria correcta es 1100001 la cual corresponde al número 12.

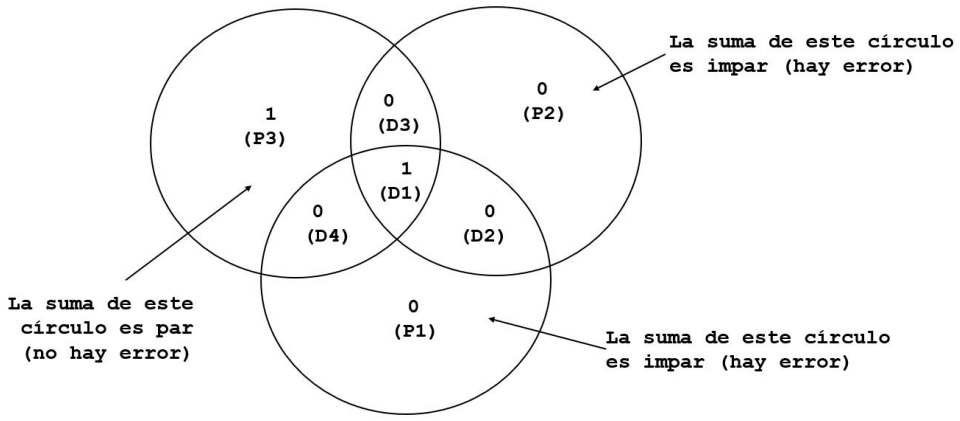

Figura 14. Usando el diagrama de Venn se puede determinar en qué tarjeta mintió el jugador.

6. Para concluir

El código Hamming, es un sistema de detección y corrección de errores el cual asocia una serie de bits de paridad, en nuestro caso 3 , a los bits de datos, en nuestro caso 4, de tal forma que una alteración en alguno de esos bits de datos pueda ser detectada y corregida adecuadamente. Aquí solamente hemos visto un ejemplo sencillo de cómo funcionan este tipo de códigos pero, se pueden desarrollar otros conceptos, como el de distancia Hamming, que permitirán establecer el número de bits erróneos que pueden ser corregidos ó detectados. Los códigos Hamming se usan principalmente en sistemas donde es crucial la fiabilidad, como por ejemplo en las telecomunicaciones. Hay otro tipo de códigos que son más adecuados en otras áreas como los códigos Reed-Solomon que son útiles para corregir múltiples errores en bloques de datos y se usan en CD o DVD. También los códigos BCH (Bose-Chaudhuri-Hocquenghem) son códigos de bloque que pueden corregir múltiples errores. Esta es un área fascinante que está continuamente en renovación y tiene muchas aplicaciones para el bienestar de la humanidad.

Bibliografía

[1] J. Baylis, *Error-correcting codes*, Chapman & Hall/CRC, 1998.
[2] J. Hall, «Notes on coding theory», https://users.math.msu.edu/users/halljo/classes/codenotes/Topstuff.pdf, 2010, Consultado en 2024.
[3] J. Hirschfeld y L. Storme, «The packing problem in statistics, coding theory and finite projective spaces», *Journal of Statistical Planning and Inference*, vol. 72, núm. 1, 1998, 355–380.

[4] V. Pless, *Introduction to the theory of error-correcting codes*, 3.ª ed., John Wiley & Sons, Inc., 1998.
[5] IEEE Computer Society, «Richard wesley hamming», https://history.computer.org/pioneers/pdfs/H/Hamming.pdf, Actualizado en 2012, Consultado en 2024.