WIKIPEDIA
La enciclopedia libre

Código Hamming

En informática, el **código de Hamming** es un código detector y corrector de errores que lleva el nombre de su inventor, Richard Hamming. En los datos codificados en Hamming se pueden detectar errores en un bit y corregirlos, sin embargo no se distingue entre errores de dos bits y de un bit (para lo que se usa Hamming extendido). Esto representa una mejora respecto a los códigos con bit de paridad, que pueden detectar errores en solo un bit, pero no pueden corregirlo.

Códigos pre-Hamming

Antes de los códigos Hamming se utilizaron ciertos códigos detectores de error, como lo fueron el código linteing, pero ninguno llegó a ser tan eficaz como los de Hamming. A continuación se describen algunos de estos códigos.

Paridad

La paridad consiste en añadir un bit, denominado bit de paridad, que indique si el número de los bits de valor 1 en los datos precedentes es par o impar. Si un solo bit cambiara por error en la transmisión, el mensaje cambiará de paridad y el error se puede detectar (nótese que el bit donde se produzca el error puede ser el mismo bit de paridad). La convención más común es que un valor de paridad **1** indica que hay un número impar de unos en los datos, y un valor de paridad de **0** indica que hay un número par de unos en los datos.

La comprobación de paridad no es muy robusta, dado que si cambia de forma uniforme un número par de bits, el bit de paridad será válido y el error no será detectado. Se utiliza cuando se cumplen simultáneamente dos condiciones: que la probabilidad de que falle un bit es baja y que las fallas de bits son sucesos independientes. De esta forma la probabilidad de que fallen dos (o más) bits es muy baja, por lo que cuando no detecta error es altamente probable que el código sea efectivamente correcto. Cabe destacar que dichas condiciones se ajustan al caso de las memorias de las computadoras modernas pero no ocurre lo mismo con los dispositivos de almacenamiento que guardan la información en forma serial (un bit a continuación de otro) ni con los sistemas de transmisión de datos seriales ya que en estos casos el hecho que falle un bit está vinculado, en forma no despreciable, a la falla de otro adyacente.

Por otro lado, la paridad, aunque puede detectar que hay error, no indica en qué bit se cometió, si bien la sobrecarga que produce este método es muy baja, desde un punto de vista computacional como de espacio (1 solo bit). Los datos se deben desechar por entero y volverse a transmitir. En un medio ruidoso, una transmisión correcta podría tardar mucho tiempo o incluso, en el peor de los casos, no darse nunca.

Dos entre cinco

En los años 40, Bell utilizó un código algo más sofisticado conocido como dos-entre-cinco. Este código se basa en que cada bloque de cinco bits (conocido como penta-bit) tuviera exactamente dos unos, asegurando así que tenga una Distancia de Hamming igual a dos. De este modo, la computadora podría detectar posibles errores cuando en su entrada no había exactamente dos unos en cada penta-bit.

Este código seguía únicamente detectando errores por cambio en un solo bit; si en un mismo penta-bit(cadena de 5 bit's) un 0 cambiaba a 1 y un 1 cambiaba a 0, la regla de dos-entre-cinco se seguía cumpliendo y el error quedaba sin descubrir.

Repetición

Otro código utilizado, consistía en repetir cada bit de datos varias veces para asegurarse de que la transmisión era correcta. Por ejemplo, si el bit de datos que se envía fuera un 1, un código de repetición con n=3, enviaría "111". Si los tres bits recibidos no eran idénticos, había un error. En un ambiente sin demasiado ruido, la mayoría de las veces solamente cambiaría un bit en cada paquete de tres bits. Por lo tanto, datos del tipo 001, 010, y 100 se corresponden al bit 0, mientras que 110, 101, y 011 se corresponden con el bit 1. Un código con esta capacidad de reconstruir el mensaje original en la presencia de errores se conoce como código corrector de errores.

Sin embargo, este código no puede reparar correctamente todos los errores. En nuestro ejemplo, si el error en la transmisión provocara el cambio simultáneo de dos bits y el receptor recibiera "001", el sistema detectaría el error, pero considerando que el bit original era 0, lo cual es incorrecto. Si se aumenta el número de veces que se repite cada bit a cuatro (n=4), es posible detectar los errores en dos bits pero obviamente no se podrán corregir; con cinco, es posible corregir errores de dos bits, pero no lo podrá hacer en errores de tres bits.

Por otra parte, el código de la repetición es extremadamente ineficaz, pues reduce la velocidad de transmisión por tres en nuestro ejemplo original y su eficacia cae drásticamente al aumentar el número de veces que cada bit se repite para detectar y corregir más errores. El uso del código de bloques no lineales para detección de errores no es muy implementado por lo tanto emplearemos el código de errores lineales para la corrección de errores.

Códigos Hamming

Si se añaden junto al mensaje más bits detectores-correctores de error y si esos bits se pueden ordenar de modo que diferentes bits de error producen diferentes resultados, entonces los bits erróneos podrían ser identificados. En un conjunto de siete bits, hay solo siete posibles errores de bit, por lo que con tres bits de control de error se podría especificar, además de que ocurrió un error, en qué bit fue.

Hamming estudió los esquemas de codificación existentes, incluido el de dos entre cinco, y generalizó sus conclusiones. Para empezar, desarrolló una nomenclatura para describir el sistema, incluyendo el número de los bits de datos y el de los bits detectores-correctores de error en un bloque. Por ejemplo, la paridad incluye un solo bit para cualquier palabra de datos, así que las palabras del Código ASCII que son de siete bits, Hamming las describía como un código (8.7), esto es, un total de 8 bits de los cuales 7 son datos. Con base a la anterior repetición, sería un código (3.1), siguiendo la misma lógica. La relación de la información es el segundo número dividido por el primero, por nuestro ejemplo de la repetición, 1/3.

Hamming también estudió los problemas que surgían al cambiar dos o más bits a la vez y describió esto como "distancia" (ahora llamada distancia de Hamming en su honor). La paridad tiene una distancia de 2, dado que cualquier error en dos bits no será detectado. La repetición (3.1)tiene una distancia de 3, pues son necesarios el cambio simultáneo de tres bits para obtener otra palabra de código. La repetición (4.1) (cada bit se repite cuatro veces) tiene una distancia de 4, así que el cambio de dos bits en el mismo grupo quedará sin definir.

Hamming estaba interesado en solucionar simultáneamente dos problemas: aumentar la distancia tanto como sea posible, a la vez que se aumentan al máximo los bits de información. Durante los años 40 desarrolló varios esquemas de codificación que mejoraban notablemente los códigos existentes. La clave de todos sus sistemas era intercalar entre los bits de datos los de paridad.

Hamming (7,4)

Hoy, el código de Hamming se refiere al (7.4) que Hamming introdujo en 1950. El código de Hamming agrega tres bits adicionales de comprobación por cada cuatro bits de datos del mensaje. El algoritmo de Hamming (7.4) puede corregir cualquier error de un solo bit, pero cuando hay errores en más de un bit, la palabra transmitida se confunde con otra con error en un solo bit, siendo corregida, pero de forma incorrecta, es decir que la palabra que se corrige es otra distinta a la original, y el mensaje final será incorrecto sin saberlo. Para poder detectar (aunque sin corregirlos) errores de dos bits, se debe añadir un bit más, y el código se llama Hamming extendido. El procedimiento para esto se explica al final. El algoritmo es el siguiente:

1. Todos los bits cuya posición es potencia de dos se utilizan como bits de paridad (posiciones 1, 2, 4, 8, 16, 32, 64, etc.).

2. Los bits del resto de posiciones son utilizados como bits de datos (posiciones 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, etc.).

3. Cada bit de paridad se obtiene calculando la paridad de alguno de los bits de datos. La posición del bit de paridad determina la secuencia de los bits que alternativamente comprueba y salta, a partir de éste, tal y como se explica a continuación. Posición 1: salta 0, comprueba 1, salta 1, comprueba 1, etc. Posición 2: salta 1, comprueba 2, salta 2, comprueba 2, etc. Posición 4: salta 3, comprueba 4, salta 4, comprueba 4, etc. Posición 8: salta 7, comprueba 8, salta 8, comprueba 8, etc. Posición 16: salta 15, comprueba 16, salta 16, comprueba 16, etc. Regla general para la posición n es: salta n-1 bits, comprueba n bits, salta n bits, comprueba n bits... Y así sucesivamente. En otras palabras, el bit de paridad de la posición comprueba los bits en las posiciones que tengan al bit k en su representación binaria. Dicho a la inversa, el bit 4, chequea los bits 4, 5, 6, 7, al ser estos los de su representación binaria: 4=100(2), 5=101(2), 6=110(2) y 7=111(2). Por el contrario, el mismo bit de paridad no comprueba el bit 8, debido a que en su representación binaria el bit número 3 (=4) es igual a 0 (8=1000B). Así, por ejemplo, para los primeros términos se tiene: En la Posición 1 (2^0 = 1), comprobaríamos los bits: 1, 3, 5, 7, 9, 11, 13... En la Posición 2 (2^1 = 2), los bits: 2, 3, 6, 7, 10, 11, 14, 15... En la Posición 4 (2^2 = 4), los bits: 4, 5, 6, 7, 12, 13, 14, 15, 20, 21, 22, 23... En la Posición 8 (2^3 = 8) tendríamos: 8, 9, 10, 11, 12, 13, 14, 15, 24-31... Siguiendo el algoritmo hasta completar la nueva cadena.

Ejemplo

Consideremos la palabra de datos de 7 bits "0110101". Para ver cómo se generan y utilizan los códigos Hamming para detectar un error, observe las tablas siguientes. Se utiliza la **d** para indicar los bits de datos y la **p** para los de paridad.

En primer lugar los bits de datos se insertan en las posiciones apropiadas y los bits de paridad calculados en cada caso usando la paridad par.

	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7
Palabra de datos (sin paridad):			**0**		**1**	**1**	**0**		**1**	**0**	**1**
p_1	**1**		0		1		0		1		1
p_2		**0**	0			1	0			0	1
p_3				**0**	1	1	0				
p_4								**0**	1	0	1
Palabra de datos (con paridad):	**1**	**0**	0	**0**	1	1	0	**0**	1	0	1

Cálculo de los bits de paridad en el código Hamming

$$P_1 = D_1 \oplus D_2 \oplus D_4 \oplus D_5 \oplus D_7$$
$$P_2 = D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_7$$
$$P_3 = D_2 \oplus D_3 \oplus D_4$$
$$P_4 = D_5 \oplus D_6 \oplus D_7$$

La nueva palabra de datos (con los bits de paridad) es ahora "10001100101". Consideremos ahora que el bit de la derecha, por error, cambia de 1 a 0. La nueva palabra de datos será ahora "10001100100".

Sin errores

	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7	Prueba de paridad	Bit de comprobación
Palabra de datos recibida:	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**1**	**1**	
p_1	**1**		0		1		0		1		1	**Correcto**	**0**
p_2		**0**	0			1	0			0	1	**Correcto**	**0**
p_3				**0**	1	1	0					Correcto	0
p_4								**0**	1	0	1	**Correcto**	**0**

Comprobación de los bits de paridad (con primer bit de la derecha sin cambiar)

Con errores

	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7	Prueba de paridad	Bit de comprobación
Palabra de datos recibida:	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	
p_1	**1**		0		1		0		1		0	**Error**	**1**
p_2		**0**	0			1	0			0	0	**Error**	**1**
p_3				**0**	1	1	0					Correcto	0
p_4								**0**	1	0	0	**Error**	**1**

Comprobación de los bits de paridad (con primer bit de la derecha cambiado)

Si se analiza en la tabla anterior la paridad que se debe obtener a la derecha tras la llegada del mensaje sin errores debe ser siempre 0 (por cada fila), pero en el momento en que ocurre un error esta paridad cambia a 1, de allí el nombre de la columna "prueba de paridad 1". Se observa que en la fila en que el cambio no afectó la paridad es cero y llega sin errores. SI se observa que en la fila en que el cambio no afecte la paridad es cero y llega sin errores

El paso final es evaluar los bits de paridad (recuerde que el fallo se encuentra en $\mathbf{d_7}$). El valor entero que representan los bits de paridad es 11 (si no hubieran ocurrido errores este valor sería 0), lo que significa que el bit décimo primero de la palabra de datos (bits de paridad incluidos) es el erróneo y necesita ser cambiado.

	p_4	p_3	p_2	p_1	
Binario	1	0	1	1	
Decimal	**8**		**2**	**1**	**Σ = 11**

Cambiando el bit undécimo primero 10001100101**0** se obtiene de nuevo 10001100101**1**. Eliminando los bits de patrón de la paridad no se tienen en cuenta los bits de paridad. Si el error se produjera en uno de ellos, en la comprobación solo se detectaría un error, justo el correspondiente al bit de paridad causante del mismo.

Hamming Extendido

Finalmente, para detectar errores en 2 bits se utiliza un bit adicional de paridad (Hamming Extendido) donde puede darse el caso de 3 posibilidades:

```
1.-' No hay error ->   Hamming =0, Paridad OK

2.- Un bit de error -> Paridad Fallo entonces
a) Hamming = 0, P = incorrecto, en este caso se cambia el valor del bit de paridad.
b) Hamming <> 0, corrijo según Hamming.

3.- Dos bit en error -> Paridad Ok, Hamming <> 0, por lo tanto informó, NO corrijo.
```
