Detección de errores y códigos de corrección

Sistemas de Procesamiento de Datos

UTN-FRA

Técnico Superior en Programación

2017

Detección de errores y códigos de corrección

Al transmitir información entre dispositivos electrónicos, los datos pueden sufrir alteraciones provocando una incorrecta interpretación de la información recibida.

Es por eso que existen distintos métodos para codificar los datos a enviar y para detectar posibles errores en la información recibida.

Detección de errores y códigos de corrección

- Paridad
- Hamming
- Check Sum

Método de paridad para la detección de errores

Un bit de paridad es un dígito binario que indica si el número de bits con un valor de 1 en un conjunto de bits es par o impar.

7 bits de datos	byte con bit de paridad	
	par	impar
0000000	00000000	00000001
1010001	10100011	10100010
1101001	11010010	11010011

Los bits de paridad conforman el método de detección de errores más simple.

Método de paridad para la detección de errores

Ventajas

- Simple de implementar

Desventajas

- Un único bit de paridad permite detectar errores en un único bit.
- No permite determinar cuál es el bit erróneo

Código de Hamming

- Si se pretende corregir un error detectado, se necesita más información, ya que hay que identificar la posición del bit erróneo antes de poder corregirlo.
- Debe incluirse más de un bit de paridad en un grupo de bits para poder corregir el error detectado.
- En un código de 7 bits, existen siete posibles bits erróneos.
- El código Hamming proporciona un método de corrección de un único bit erróneo.

Código de Hamming

Si el número de bits de datos se designa por d, entonces el número de bits de paridad, p, se determina mediante la siguiente relación:

$$2^p \geq d + p + 1$$

Por ejemplo, si tenemos siete bits de datos, entonces p se calcula por el método de prueba y error.

SI p=3 y d=7 => $2^3 \geq 7+3+1$ => $8 \geq 11$ => NO

SI p=4 y d=7 => $2^4 \geq 7+4+1$ => $16 \geq 12$ => SI

Generar Código de Hamming

Una vez determinado el número necesario de bits de paridad se debe colocar correctamente los bits dentro del código. En nuestro ejemplo 7 bits de datos y 4 de paridad.

El bit más a la izquierda es el bit 1, el siguiente bit es el bit 2, y así sucesivamente:

bit 1, bit 2, bit 3, bit 4, bit 5, … bit 11

Generar Código de Hamming

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.).

P1, P2, D1, P3, D2, D3, D4, P4, D5, D6, D7

Generar Código de Hamming

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 datos.

	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

Generar Código de Hamming

		0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011
		P1	**P2**	D1	**P3**	D2	D3	D4	**P4**	D5	D6	D7
P1	XXX1	**P1**		D1		D2		D4		D5		D7
P2	XX1X		**P2**	D1			D3	D4			D6	D7
P3	X1XX				**P3**	D2	D3	D4				
P4	1XXX								**P4**	D5	D6	D7

Comprobar Código de Hamming

	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**

Recibir Código de Hamming

Supongamos que hubo un error en la transferencia de información y en lugar de ser “0110101” es “0110100” y hacemos una comprobación de bits de paridad.

	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	**0**		0		1		0		1		0	**Error**	**1**
p_2		**1**	0			1	0			0	0	**Error**	**1**
p_3				**0**	1	1	0					Correcto	0
p_4								**1**	1	0	0	**Error**	**1**

Error: 1011 => Posición 11

Código de Hamming

El código hamming muestra que es eficiente para resolver errores de un conjunto de bits mientras ese error solo sea uno por conjunto.

El algoritmo de Hamming puede corregir cualquier error de un solo bit, pero cuando hay errores en más de un bit, la palabra que se corrige es distinta a la original, y el mensaje final será incorrecto sin saberlo.

Checksum

- Una suma de verificación es una función hash que tiene como propósito principal detectar cambios en una secuencia de datos para proteger la integridad de estos
- Verifica que no existan discrepancias entre los valores obtenidos al hacer una comprobación inicial y otra final tras la transmisión.
- La idea es que se transmita el dato junto con su valor hash, de esta forma el receptor puede calcular dicho valor y compararlo así con el valor hash recibido.
- Si hay una discrepancia se pueden rechazar los datos o pedir una retransmisión.

Función Hash

- Tiene como entrada un conjunto de elementos, que suelen ser cadenas, y los convierte en un rango de salida finito, normalmente cadenas de longitud fija. Es decir, la función actúa como una proyección del conjunto U sobre el conjunto M.

- Observar que M puede ser un conjunto definido de enteros. En este caso podemos considerar que la longitud es fija.