Práctica 1

# Corrección de errores

26 de noviembre de 2013

¿Has pensado alguna vez en cómo es posible que tu lector de discos compactos reproduzca el sonido con tanta exactitud a pesar del polvo, las huellas y los rayajos que habitualmente hay en un disco compacto? El secreto radica en que los discos compactos están grabados utilizando un *código corrector de errores*, que el lector de discos utiliza para superar, en la medida de lo posible, los fallos de lectura.

Los códigos correctores de errores se utilizan cuando se transmite información a través de un canal de comunicación *ruidoso*, es decir, un canal que puede producir errores, como son los soportes de almacenamiento magnético, en transmisión de datos por satélite, en transacciones bancarias... Añadiendo un poco de redundancia a la información transmitida, se puede detectar y, a menudo, corregir un error producido en la transmisión.

Para entender el funcionamiento básico de los códigos de corrección de errores, pongamos un ejemplo real: con gran esfuerzo se ha conseguido situar un robot sobre la superficie de Marte (ver figura 1). Dicho robot puede moverse obedeciendo a cuatro órdenes diferentes de movimiento, una por cada dirección: norte, sur, este y oeste. Supongamos que tenemos el siguiente código para enviar las órdenes al robot:

| Orden de movimiento | Código |
|---|---|
| norte | 0 0 |
| este | 0 1 |
| oeste | 1 0 |
| sur | 1 1 |

Si se produce una interferencia al transmitir las órdenes de desplazamiento al robot, dicha interferencia puede modificar el código enviado. Si el código consta de dos bits, cada uno de ellos puede sufrir un cambio durante la transmisión. Pongamos por caso que queremos enviar la orden de ir hacia el norte, 00, y se produce sólo un error; el mensaje recibido puede ser 01 o 10. Los errores en el código que estamos considerando son muy graves, ya que un error de transmisión implica que el robot se mueva en una dirección no deseada con los problemas que esto puede acarrear.

Una forma de solucionar este problema es enviar órdenes que *no se parezcan* tanto entre sí, como ocurría con las anteriores. Supongamos que tenemos el siguiente código:

| Orden de movimiento | Código |
|---|---|
| norte | 0 0 0 |
| este | 0 1 1 |
| oeste | 1 1 0 |
| sur | 1 0 1 |

Enviar estos mensajes es algo más costoso que en el caso anterior ya que cada orden consta de tres señales, en lugar de dos. Pero este código tiene una gran ventaja con respecto al anterior: si se produce un único error, el mensaje recibido no corresponde a ninguno de los mensajes posibles, y por tanto, puede *detectarse* que la orden que ha llegado es errónea.

En efecto, supongamos que enviamos la orden de ir hacia el este, 011. Si se produce un único error en la transmisión, sólo uno de los bits del mensaje cambia y podemos recibir

Figura 1: Robot enviado por la NASA a Marte

el mensaje 111, 001 o 010, según sea el bit alterado el primero, el segundo o el tercero respectivamente. En cualquiera de los casos, el mensaje recibido no coincide con ninguna de las órdenes y por tanto el robot puede saber que la orden que le ha llegado es defectuosa y no realizar ningún movimiento.

Este tipo de códigos que *detectan* errores son útiles en multitud de ocasiones, pero no en el ejemplo que estamos considerando. Si el robot detecta una orden defectuosa debería informar de ello y esperar a recibir una nueva orden de movimiento. Teniendo en cuenta lo que tardan los mensajes en recorrer la distancia de la Tierra a Marte (¡12 minutos, sólo de ida, a la velocidad de la luz, en su acercamiento máximo!), parece inadecuado depender de una comunicación bidireccional.

Pero si los mensajes enviados se *diferencian mucho* entre sí, quizás podamos *restaurarlos* aunque se haya producido algún error. Consideremos el siguiente código:

| Orden de movimiento | Código |
|---|---|
| norte | 0 0 0 0 0 |
| este | 0 1 1 0 1 |
| oeste | 1 0 1 1 0 |
| sur | 1 1 0 1 1 |

Si suponemos que se produce como mucho un error en la transmisión de cada mensaje, entonces el mensaje recibido sigue estando *próximo* al mensaje enviado y por tanto podemos reconocer el mensaje original. Supongamos que el mensaje enviado al robot es moverse hacia el norte, 00000. Si suponemos que se produce un único error, entonces el robot puede recibir alguno de los mensajes 10000, 01000, 00100, 00010 o 00001. Cada uno de estos mensajes está más *próximo* al original que a cualquier otro de los códigos y por tanto el robot puede entender que la orden enviada fue ir al norte.

Por supuesto hay que definir formalmente qué se entiende por *próximo*: en teoría de códigos se suele utilizar la *distancia de Hamming*...

## 1 Distancia de Hamming

Dados dos vectores, $v_1$ y $v_2$, se define la *distancia de Hamming*, $d(v_1, v_2)$, como el número de posiciones en que difieren los vectores $v_1$ y $v_2$. Por ejemplo, si $v_1 = 00001$ y $v_2 = 10100$,

$d(v_1, v_2) = 3$ ya que existen tres posiciones (la primera, la tercera y la quinta) en las que los vectores tienen diferente valor.

Escribe una función que, dadas dos palabras de igual longitud pertenecientes a un código, indique cuál es la distancia de Hamming entre ellas.

## 2 Distancia mínima de un código

Si $C = \{p_1, \ldots, p_n\}$ es un código con $n$ palabras, se define la *mínima distancia del código* $C$ así:

$$d(C) = \text{mín}\{d(p_i, p_j) \mid p_i, p_j \in C, i \neq j\}$$

Esta distancia mínima informa de los errores que podremos detectar y corregir con el código que estamos considerando. Si $d(C) \geq 2k + 1$, entonces podemos detectar si se han producido hasta $2k$ errores; y podremos corregir hasta $k$ errores.

Escribe un subprograma que, dado un código (es decir, las palabras que lo componen), calcule la mínima distancia de dicho código e informe de la posibilidad de detección y corrección de errores de dicho código.

## 3 Simulación de ruido

Para simular un canal de transmisión de datos ruidoso se han de tener en cuenta ciertas hipótesis:

- Todos los bits de un mensaje tienen la misma probabilidad de ser recibidos con error.
- La probabilidad $p$ con la que cada bit transmitido se recibe con error ha de ser menor que $\frac{1}{2}$.

Escribe un subprograma que, fijada una cierta probabilidad de fallo, $p$, y teniendo en cuenta las condiciones anteriores, simule la transmisión de datos a través de un canal que puede producir fallos.

## 4 Detección de errores

Escribe un subprograma que, dado un código $C$, simule la transmisión de datos a través de un canal ruidoso e informe de la detección de errores.

## 5 Corrección de errores

Escribe un subprograma que, dado un código $C$, simule la transmisión de datos a través de un canal ruidoso y, si el código lo permite, corrija los posibles errores de transmisión que se produzcan.

## 6 Simulación global

Escribe un programa que permita hacer una simulación total del uso de un código corrector de errores.

## 7 Notas bibliográficas

En el ejercicio, por motivos de claridad y sencillez, hemos asumido ciertas simplificaciones. Por ejemplo hemos considerado únicamente códigos sobre el alfabeto binario 0 y 1, pero las mismas técnicas de codificación se pueden utilizar sobre otros alfabetos. El código ISBN de los libros utiliza los números del 0 al 9 y la letra X.

En Internet, hay un buen sitio dedicado al funcionamiento de los ordenadores personales que explica los mecanismos de codificación de datos utilizados en diversos componentes: discos compactos (`http://www.pcguide.com/ref/cd/mediaECC-c.html`), disco duro (`http://www.pcguide.com/ref/hdd/geom/error_ECC.htm`) y memoria (`http://www.pcguide.com/ref/ram/err.htm`).

Libros para conocer más sobre la teoría de la codificación son [Hil99] y [HLL+92].

**Fecha límite de entrega:** 21 de febrero del 2010. No olvides concertar una cita con el profesor, a través de la web de la asignatura, con el objeto de fijar el momento de la entrega.

## Referencias


[Hil99] Raymond Hill. *A first course in coding theory.* Oxford Clarendon Press, 1999.

[HLL+92] D. G. Hoffman, D. A. Leonard, C. C. Lindner, K. T. Phelps, C. A. Rodger, and J. R. Wall. *Coding theory: the essentials.* Marcel Dekker, 1992.