Módulo 07

Detección y Corrección de Errores (Pt. 3)

Organización de Computadoras
Depto. Cs. e Ing. de la Comp.
Universidad Nacional del Sur

Copyright

- Copyright © **2011-2024** A. G. Stankevicius
- Se asegura la libertad para copiar, distribuir y modificar este documento de acuerdo a los términos de la **GNU** Free Documentation License, Versión 1.2 o cualquiera posterior publicada por la Free Software Foundation, sin secciones invariantes ni textos de cubierta delantera o trasera
- Una copia de esta licencia está siempre disponible en la página **http://www.gnu.org/copyleft/fdl.html**
- La versión transparente de este documento puede ser obtenida de la siguiente dirección:

http://cs.uns.edu.ar/~ags/teaching

Contenidos

- Concepto de error
- Mínima distancia de un código
- Mecanismos de detección de errores
- Paridad aplicada en los códigos **VRC** y **LRC**
- Generación y verificación de código **CRC**
- Mecanismos de corrección de errores
- Códigos correctores simples
- Hamming mínima distancia 3 y 4

Corrección de errores

- Recordemos que para poder implementar algún mecanismo de detección es necesario agregar redundancia al mensaje transmitido:
  - A la hora de enviar **m** bits de dato, se incorporan **r** bits adicionales (también denominados de código)
  - En general habrá 2^m mensajes válidos, si bien no todos los 2^{m+r} patrones de bits van a representar combinaciones válidas
  - Es decir, el agregado de esta redundancia es lo que posibilita que un dado código evidencie una mínima distancia mayor

Corrección de errores

- Hasta el momento nos hemos contentado con poder detectar que se produjo un error
- Usualmente cuando se detecta un error se suele solicitar la retransmisión del mensaje recibido incorrectamente
- No obstante, existen dominios de aplicación donde la retransmisión del mensaje original resulta costosa o incluso imposible
- En esos casos tiene sentido intentar establecer dónde se produjo el error, a fin de corregirlo

Ejemplo

- Consideremos el siguiente código, cuya mínima distancia es **3**:

A:	`000000`	**C:**	`101010`
B:	`010101`	**D:**	`111111`

- En este contexto, analicemos qué se puede afirmar al recibir los siguientes patrones:
  - `011111`
  - `101011`
  - `001011`

Análisis

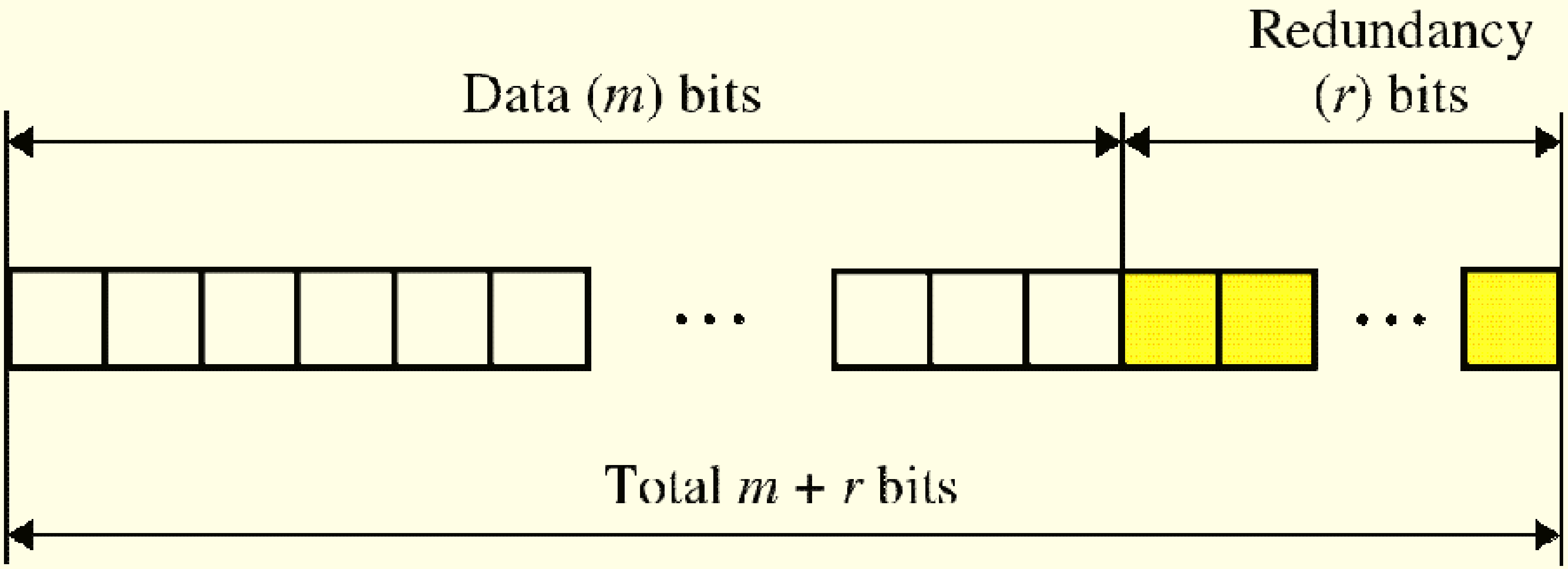
- Del análisis del ejemplo anterior se pueden sacar dos conclusiones interesantes:
  - Aparentemente ante los errores simples es posible reconocer con certeza en qué bit se produjo el error
  - No obstante, el adoptar esta política de corrección de errores parece afectar la capacidad de detección
- De alguna manera, al optar por corregir debemos asegurarnos que ningún otro patrón válido esté a la misma distancia que el que determinamos era el patrón original

Análisis

- Continúa:
  - Por caso, si al recibir el patrón **011111** interpretaremos que el se produjo un error simple ya que el patrón original era **111111**, no detectaremos el error doble cuando el patrón original hubiera sido **010101**
  - Es decir, en el código del ejemplo si optamos por corregir errores simples, ¡debemos dejar de detectar los errores dobles!
  - En general, para **M = 3** se observa que se puede detectar errores simples y dobles, o bien detectar y corregir sólo errores simples

Diseño de un código

- Supongamos que se desea diseñar un código para corregir errores simples con $n = m + r$



Diseño de un código

- En este escenario, para cada patrón válido debemos reservar cada uno de los $m + r$ patrones que se obtienen al adulterar un bit
  - Es decir, para cada uno de los 2^m mensajes debemos reservar de $m + r + 1$ patrones
  - Estos patrones tiene que codificarse con $m + r$ bits:

$$2^m \times (m + r + 1) \leq 2^{m+r} \rightarrow m + r + 1 \leq 2^r$$

  - En otras palabras, una vez elegido un cierto m, es posible determinar matemáticamente cuál será el valor óptimo de r

Diseño de un código

- Extendiendo este análisis para el caso de errores dobles se observa lo siguiente:
  - Para cada patrón válido se debe reservar por un lado $m + r$ patrones para corregir errores simples pero por otro lado hacen falta $[(m + r)(m + r - 1)]/2$ patrones para corregir los errores dobles
  - En síntesis, simplificando hora tenemos que:

$$m + r + [(m + r)(m + r - 1)]/2 + 1 \leq 2^r$$

  - Nuevamente, fijado m se puede determinar matemáticamente el valor óptimo para r

El rol de la mínima distancia

- El concepto de mínima distancia de un código desempeña un rol central en la determinación de la capacidad de detección y de corrección:
  - Para una cierta distancia mínima $\mathbf{M}$ es posible detectar hasta $\mathbf{d = M - 1}$ bits en error
  - No obstante, al incorporar la capacidad de corrección, por cada error corregido debemos disminuir de manera acorde la capacidad de detección
  - En otras palabras, $\mathbf{d = M - 1 - c}$, lo que equivale a afirmar que $\mathbf{M - 1 = c + d}$

El rol de la mínima distancia

- Continúa:
  - Por otra parte, como por cada bit corregido debemos asegurarnos que ningún otro patrón válido esté a la misma distancia aparte del elegido, también se debe verificar que $c \leq (M - 1) / 2$
  - Reemplazando $M - 1$ por el valor antes obtenido se deriva que $2c \leq c + d$, es decir, que $c \leq d$
- En resumidas cuentas:
  - $M - 1 = c + d$
  - $c \leq d$

Ejemplo

- Resolvamos este conjunto de inecuaciones para valores concretos de **M**:
  - Para **M = 2** (por caso, paridad), se observa que la única solución posible es **c = 0** y **d = 1**
  - Para **M = 3** (por caso, Hamming mínima distancia 3), existen dos solucione posibles: bien **c = 1** y **d = 1** o bien **c = 0** y **d = 2**
  - Para **M = 4** (por caso, Hamming mínima distancia 4), también existen dos soluciones: bien **c = 1** y **d = 2** o bien **c = 0** y **d = 3**. Nótese que ni **c = 2** y **d = 1** ni tampoco **c = 3** y **d = 0** satisfacen que $c \leq d$

Código corrector naive

- Una manera no muy eficiente de introducir redundancia consiste en replicar los datos
  - Por ejemplo, una posibilidad es repetir **k** veces cada bit del mensaje original
  - ¿Qué mínima distancia manifestará este código?
  - Luego, ¿qué capacidad de detección y de corrección tendrá este código?
- El código del primer ejemplo que vimos hacía uso de esta propiedad:

A:	**000000**	**C:**	**101010**
B:	**010101**	**D:**	**111111**

¿Preguntas?