Códigos de Corrección de Errores. 1ra Clase

Daniel Penazzi

2 de junio de 2021

Tabla de Contenidos

Ejemplo

- Supongamos que A desea mandarle a B el siguiente mensaje:

Ejemplo

- Supongamos que A desea mandarle a B el siguiente mensaje:
- “YO TE QUIERO”.

Ejemplo

- Supongamos que A desea mandarle a B el siguiente mensaje:
- “YO TE QUIERO”.
- Si llega bien, OK, pero que pasa si hay algun error en el camino?

Ejemplo

- Supongamos que A desea mandarle a B el siguiente mensaje:
- “YO TE QUIERO”.
- Si llega bien, OK, pero que pasa si hay algun error en el camino?
- Por “error” nos referimos a que una letra sea cambiada por otra por algún ruido en el medio de transmisión.

Ejemplo

- Supongamos que A desea mandarle a B el siguiente mensaje:
- “YO TE QUIERO”.
- Si llega bien, OK, pero que pasa si hay algun error en el camino?
- Por “error” nos referimos a que una letra sea cambiada por otra por algún ruido en el medio de transmisión.
- Por ejemplo, podria llegar: “YO TI QUIERO”.

Ejemplo

- Supongamos que A desea mandarle a B el siguiente mensaje:
- “YO TE QUIERO”.
- Si llega bien, OK, pero que pasa si hay algun error en el camino?
- Por “error” nos referimos a que una letra sea cambiada por otra por algún ruido en el medio de transmisión.
- Por ejemplo, podria llegar: “YO TI QUIERO”.
- En este caso, B se da cuenta que hubo un error en el camino, y mas aún, puede determinar cual era el mensaje original.

Ejemplo

- Supongamos que A desea mandarle a B el siguiente mensaje:
- “YO TE QUIERO”.
- Si llega bien, OK, pero que pasa si hay algun error en el camino?
- Por “error” nos referimos a que una letra sea cambiada por otra por algún ruido en el medio de transmisión.
- Por ejemplo, podria llegar: “YO TI QUIERO”.
- En este caso, B se da cuenta que hubo un error en el camino, y mas aún, puede determinar cual era el mensaje original.
- Pero podria haber llegado: “NO TE QUIERO”

Ejemplo

- Supongamos que A desea mandarle a B el siguiente mensaje:
- “YO TE QUIERO”.
- Si llega bien, OK, pero que pasa si hay algun error en el camino?
- Por “error” nos referimos a que una letra sea cambiada por otra por algún ruido en el medio de transmisión.
- Por ejemplo, podria llegar: “YO TI QUIERO”.
- En este caso, B se da cuenta que hubo un error en el camino, y mas aún, puede determinar cual era el mensaje original.
- Pero podria haber llegado: “NO TE QUIERO”
- con consecuencias desastrosas.

- Un caso intermedio podria ser que llegue:“QO TE QUIERO”

- Un caso intermedio podria ser que llegue:“QO TE QUIERO”
- En este caso, esta claro que hubo un error,

- Un caso intermedio podria ser que llegue:“QO TE QUIERO”
- En este caso, esta claro que hubo un error,
- pero no se sabe si el mensaje original era “YO TE QUIERO” o “NO TE QUIERO”.

- Un caso intermedio podria ser que llegue:“QO TE QUIERO”
- En este caso, esta claro que hubo un error,
- pero no se sabe si el mensaje original era “YO TE QUIERO” o “NO TE QUIERO”.
- Esto tiene que ver con las ambigüedades del lenguaje natural

- Un caso intermedio podria ser que llegue:“QO TE QUIERO”
- En este caso, esta claro que hubo un error,
- pero no se sabe si el mensaje original era “YO TE QUIERO” o “NO TE QUIERO”.
- Esto tiene que ver con las ambigüedades del lenguaje natural
- Lo que veremos a continuación es tratar de diseñar sistemas de comunicaciones tal que cuando ocurran errores de transmisión, podamos:

- Un caso intermedio podria ser que llegue:“QO TE QUIERO”
- En este caso, esta claro que hubo un error,
- pero no se sabe si el mensaje original era “YO TE QUIERO” o “NO TE QUIERO”.
- Esto tiene que ver con las ambigüedades del lenguaje natural
- Lo que veremos a continuación es tratar de diseñar sistemas de comunicaciones tal que cuando ocurran errores de transmisión, podamos:
    1. Determinar que han ocurrido errores.

- Un caso intermedio podria ser que llegue:“QO TE QUIERO”
- En este caso, esta claro que hubo un error,
- pero no se sabe si el mensaje original era “YO TE QUIERO” o “NO TE QUIERO”.
- Esto tiene que ver con las ambigüedades del lenguaje natural
- Lo que veremos a continuación es tratar de diseñar sistemas de comunicaciones tal que cuando ocurran errores de transmisión, podamos:
    1. Determinar que han ocurrido errores.
    2. Detectar cuales fueron esos errores y corregirlos.

- Un caso intermedio podria ser que llegue:“QO TE QUIERO”
- En este caso, esta claro que hubo un error,
- pero no se sabe si el mensaje original era “YO TE QUIERO” o “NO TE QUIERO”.
- Esto tiene que ver con las ambigüedades del lenguaje natural
- Lo que veremos a continuación es tratar de diseñar sistemas de comunicaciones tal que cuando ocurran errores de transmisión, podamos:
  1. Determinar que han ocurrido errores.
  2. Detectar cuales fueron esos errores y corregirlos.
- Estos son los llamados “Códigos de corrección de errores”

- En general 1) es mas fácil que 2) y un código tendrá:

- En general 1) es mas fácil que 2) y un código tendrá:
    - una “capacidad de detección de errores”mas allá de la cual puede no detectar que ha habido errores.

- En general 1) es mas fácil que 2) y un código tendrá:
  - una “capacidad de detección de errores”mas allá de la cual puede no detectar que ha habido errores.
  - Y una “capacidad de corrección de errores”mas allá de la cual quizás pueda detectar que ha habido errores pero no corregirlos.

- En general 1) es mas fácil que 2) y un código tendrá:
  - una “capacidad de detección de errores”mas allá de la cual puede no detectar que ha habido errores.
  - Y una “capacidad de corrección de errores”mas allá de la cual quizás pueda detectar que ha habido errores pero no corregirlos.
- Como ejemplo extremo, si la cantidad de errores es tanta que cambian todas las letras, no podemos hacer nada.

- En general 1) es mas fácil que 2) y un código tendrá:
  - una “capacidad de detección de errores”mas allá de la cual puede no detectar que ha habido errores.
  - Y una “capacidad de corrección de errores”mas allá de la cual quizás pueda detectar que ha habido errores pero no corregirlos.
- Como ejemplo extremo, si la cantidad de errores es tanta que cambian todas las letras, no podemos hacer nada.
- Por ejemplo, podria haber llegado “ZX OF WRUTPX”.

- En general 1) es mas fácil que 2) y un código tendrá:
  - una “capacidad de detección de errores”mas allá de la cual puede no detectar que ha habido errores.
  - Y una “capacidad de corrección de errores”mas allá de la cual quizás pueda detectar que ha habido errores pero no corregirlos.
- Como ejemplo extremo, si la cantidad de errores es tanta que cambian todas las letras, no podemos hacer nada.
- Por ejemplo, podria haber llegado “ZX OF WRUTPX”.
- Hay muchos tipos de códigos, veremos algunos, y los que veremos estarán todos dentro de una sola clase (que es la mas usual, pero no la única).

Definición de códigos

Definición:

Definición de códigos

Definición:

- Un código es un conjunto $\neq \emptyset$ de palabras sobre un alfabeto A.

Definición de códigos

Definición:

- Un código es un conjunto $\neq \emptyset$ de palabras sobre un alfabeto A.
- Un codigo de bloque o block-code es un código en donde todas las palabras tienen la misma longitud, es decir, existe un n tal que el código es un subconjunto de A^n. En este caso, n se llama la longitud del bloque.

Definición:

- Un código es un conjunto $\neq \emptyset$ de palabras sobre un alfabeto A.
- Un codigo de bloque o block-code es un código en donde todas las palabras tienen la misma longitud, es decir, existe un n tal que el código es un subconjunto de A^n. En este caso, n se llama la longitud del bloque.
- Un código binario es un código en donde el alfabeto es $\{0, 1\}$

Definición de códigos

Definición:

- Un código es un conjunto $\neq \emptyset$ de palabras sobre un alfabeto A.
- Un codigo de bloque o block-code es un código en donde todas las palabras tienen la misma longitud, es decir, existe un n tal que el código es un subconjunto de A^n. En este caso, n se llama la longitud del bloque.
- Un código binario es un código en donde el alfabeto es $\{0, 1\}$

En general, salvo que digamos lo contrario, de ahora en mas cuando digamos "código" nos estaremos refiriendo a códigos binarios de bloque, es decir, a subconjuntos de $\{0, 1\}^n$ para algún n.

Ejemplos

- El código Morse es un ejemplo de un código que no es de bloque, porque distintas letras tienen distinta longitud en el código Morse.

Ejemplos

- El código Morse es un ejemplo de un código que no es de bloque, porque distintas letras tienen distinta longitud en el código Morse.
- Aunque “parece”binario, en realidad no lo es, pues ademas de los simbolos . y -, también estan los “simbolos" de pausas.

Ejemplos

- El código Morse es un ejemplo de un código que no es de bloque, porque distintas letras tienen distinta longitud en el código Morse.
- Aunque “parece”binario, en realidad no lo es, pues ademas de los simbolos . y -, también estan los “simbolos" de pausas.
- Un ejemplo de un código binario no de bloque seria $C = \{0, 01, 001, 100\}$.

Ejemplos

- El código Morse es un ejemplo de un código que no es de bloque, porque distintas letras tienen distinta longitud en el código Morse.
- Aunque “parece”binario, en realidad no lo es, pues ademas de los simbolos . y -, también estan los “simbolos" de pausas.
- Un ejemplo de un código binario no de bloque seria $C = \{0, 01, 001, 100\}$.
- Un ejemplo de código binario de bloque seria $C = \{011, 101, 001, 100\}$

Ejemplos

- El código Morse es un ejemplo de un código que no es de bloque, porque distintas letras tienen distinta longitud en el código Morse.
- Aunque “parece”binario, en realidad no lo es, pues ademas de los simbolos . y -, también estan los “simbolos" de pausas.
- Un ejemplo de un código binario no de bloque seria $C = \{0, 01, 001, 100\}$.
- Un ejemplo de código binario de bloque seria $C = \{011, 101, 001, 100\}$
- Supondremos siempre que C tiene al menos dos palabras, pues un código con una sola palabra no puede transmitir ningún mensaje mas que “te mandé un mensaje".

Situación

- La situación que estamos modelando es que tendemos un transmisor que enviará algun mensaje a travez de un canal, para que lo reciba un receptor

Situación

- La situación que estamos modelando es que tendemos un transmisor que enviará algun mensaje a travez de un canal, para que lo reciba un receptor
- $\text{Transmisor} \xrightarrow{\text{Canal}} \text{Receptor}$

Situación

- La situación que estamos modelando es que tendemos un transmisor que enviará algun mensaje a travez de un canal, para que lo reciba un receptor
- $\text{Transmisor} \xrightarrow{\text{Canal}} \text{Receptor}$
- El canal tendrá ruido, por lo que puede pasar que el receptor no reciba lo que fue mandado.

Suposiciones sobre el canal

- Haremos las siguientes suposiciones acerca del canal: (cuando algunas de estas no son ciertas, se requieren otros códigos distintos de los que veremos)

Suposiciones sobre el canal

- Haremos las siguientes suposiciones acerca del canal: (cuando algunas de estas no son ciertas, se requieren otros códigos distintos de los que veremos)
  1. El canal no pierde ni añade bits, solo los transforma.

Suposiciones sobre el canal

- Haremos las siguientes suposiciones acerca del canal: (cuando algunas de estas no son ciertas, se requieren otros códigos distintos de los que veremos)
  1. El canal no pierde ni añade bits, solo los transforma.
  2. La probabilidad de un cambio de 1 a 0 es la misma que la de un cambio de 0 a 1.

Suposiciones sobre el canal

- Haremos las siguientes suposiciones acerca del canal: (cuando algunas de estas no son ciertas, se requieren otros códigos distintos de los que veremos)
  1. El canal no pierde ni añade bits, solo los transforma.
  2. La probabilidad de un cambio de 1 a 0 es la misma que la de un cambio de 0 a 1.
  3. La probabilidad de un cambio en el bit i es independiente de la probabilidad de un cambio en el bit j

Suposiciones sobre el canal

- Haremos las siguientes suposiciones acerca del canal: (cuando algunas de estas no son ciertas, se requieren otros códigos distintos de los que veremos)
  1. El canal no pierde ni añade bits, solo los transforma.
  2. La probabilidad de un cambio de 1 a 0 es la misma que la de un cambio de 0 a 1.
  3. La probabilidad de un cambio en el bit i es independiente de la probabilidad de un cambio en el bit j
  4. Esa probabilidad es independiente de i, es decir, es la misma para todos los bits.(la llamaremos p).

Suposiciones sobre el canal

- Haremos las siguientes suposiciones acerca del canal: (cuando algunas de estas no son ciertas, se requieren otros códigos distintos de los que veremos)
  1. El canal no pierde ni añade bits, solo los transforma.
  2. La probabilidad de un cambio de 1 a 0 es la misma que la de un cambio de 0 a 1.
  3. La probabilidad de un cambio en el bit i es independiente de la probabilidad de un cambio en el bit j
  4. Esa probabilidad es independiente de i, es decir, es la misma para todos los bits.(la llamaremos p).
  5. $0 < p < \frac{1}{2}$.

Discusión sobre estas suposiciones

- La primera suposición nos permite suponer que las fronteras entre palabras se mantienen, y que podemos usar un código de bloque en forma eficiente. Si esa suposición no puede hacerse, es necesario usar otros códigos o añadir técnicas para detectar pérdida de sincronización, cosa que escapa a un curso tan corto.

Discusión sobre estas suposiciones

- La primera suposición nos permite suponer que las fronteras entre palabras se mantienen, y que podemos usar un código de bloque en forma eficiente. Si esa suposición no puede hacerse, es necesario usar otros códigos o añadir técnicas para detectar pérdida de sincronización, cosa que escapa a un curso tan corto.
- La propiedad 2 podría no ser cierta si pej el 1 se representa con mas voltage y es mas fácil que baje el voltaje a que suba. Otra vez, puede resolverse pero escapa a nuestros objetivos limitados en el tiempo.

Discusión sobre estas suposiciones

- 3 y 4 pueden no ser ciertas en ciertas circunstancias.

Discusión sobre estas suposiciones

- 3 y 4 pueden no ser ciertas en ciertas circunstancias.
- pej, si hay tormentas atmosféricas, y el bit i tiene errores porque el ruido atmosférico aumentó de repente, entonces la probabilidad de que los bits $i+1, i+2, ...$ tengan errores etc tambien aumentan.

Discusión sobre estas suposiciones

- 3 y 4 pueden no ser ciertas en ciertas circunstancias.
- pej, si hay tormentas atmosféricas, y el bit i tiene errores porque el ruido atmosférico aumentó de repente, entonces la probabilidad de que los bits $i+1, i+2, \dots$ tengan errores etc tambien aumentan.
- Esto se soluciona con códigos que permiten corregir "errores en ráfaga" . Veremos algunos de estos.

Discusión sobre estas suposiciones

- La última propiedad es muy razonable:

Discusión sobre estas suposiciones

- La última propiedad es muy razonable:
  - Si $p = 0$ no hay errores, no necesito usar códigos.

Discusión sobre estas suposiciones

- La última propiedad es muy razonable:
  - Si $p = 0$ no hay errores, no necesito usar códigos.
  - Si $p = 1/2$ entonces para decidir cual es cada bit puedo simplemente tirar una moneda.

Discusión sobre estas suposiciones

- La última propiedad es muy razonable:
  - Si $p = 0$ no hay errores, no necesito usar códigos.
  - Si $p = 1/2$ entonces para decidir cual es cada bit puedo simplemente tirar una moneda.
  - En este caso, todos los mensajes son igualmente probables y no hay teoría capaz de solucionar esto.

Discusión sobre estas suposiciones

- La última propiedad es muy razonable:
  - Si $p = 0$ no hay errores, no necesito usar códigos.
  - Si $p = 1/2$ entonces para decidir cual es cada bit puedo simplemente tirar una moneda.
  - En este caso, todos los mensajes son igualmente probables y no hay teoría capaz de solucionar esto.
  - Si $p > 1/2$, entonces los bits que llegan tienen MAS probabilidad de estar mal que de estar bien.

Discusión sobre estas suposiciones

- La última propiedad es muy razonable:
  - Si $p = 0$ no hay errores, no necesito usar códigos.
  - Si $p = 1/2$ entonces para decidir cual es cada bit puedo simplemente tirar una moneda.
  - En este caso, todos los mensajes son igualmente probables y no hay teoría capaz de solucionar esto.
  - Si $p > 1/2$, entonces los bits que llegan tienen MAS probabilidad de estar mal que de estar bien.
  - Conviene simplemente flipear todos los bits y trabajar como si los hubieramos mandado por un canal con probabilidad de error $1 - p$.

Estrategia de corrección de errores

- Con esas suposiciones sobre el canal, la estrategia de corrección de errores es:

Estrategia de corrección de errores

- Con esas suposiciones sobre el canal, la estrategia de corrección de errores es:
- Si nos llega una palabra w, deducir cual es la palabra v “mas probable" que nos hayan mandado y que pueda producir w.

Estrategia de corrección de errores

- Con esas suposiciones sobre el canal, la estrategia de corrección de errores es:
- Si nos llega una palabra w, deducir cual es la palabra v “mas probable" que nos hayan mandado y que pueda producir w.
- Pej, supongamos que para mandar u y que llegue w se tiene que producir un error en el camino.

Estrategia de corrección de errores

- Con esas suposiciones sobre el canal, la estrategia de corrección de errores es:
- Si nos llega una palabra w, deducir cual es la palabra v “mas probable" que nos hayan mandado y que pueda producir w.
- Pej, supongamos que para mandar u y que llegue w se tiene que producir un error en el camino.
- Pero que para mandar x y que llegue w se tienen que producir dos errores en el camino.

Estrategia de corrección de errores

- Con esas suposiciones sobre el canal, la estrategia de corrección de errores es:
- Si nos llega una palabra w, deducir cual es la palabra v “mas probable" que nos hayan mandado y que pueda producir w.
- Pej, supongamos que para mandar u y que llegue w se tiene que producir un error en el camino.
- Pero que para mandar x y que llegue w se tienen que producir dos errores en el camino.
- Entonces la probabilidad de mandar u y que llegue w es p, pero la probabilidad de mandar x y que llegue w es p^2.

Estrategia de corrección de errores

- Con esas suposiciones sobre el canal, la estrategia de corrección de errores es:
- Si nos llega una palabra w, deducir cual es la palabra v “mas probable" que nos hayan mandado y que pueda producir w.
- Pej, supongamos que para mandar u y que llegue w se tiene que producir un error en el camino.
- Pero que para mandar x y que llegue w se tienen que producir dos errores en el camino.
- Entonces la probabilidad de mandar u y que llegue w es p, pero la probabilidad de mandar x y que llegue w es p^2.
- Como $p^2 < p$, concluimos que es mas probable que la palabra mandada sea u.

Estrategia de corrección de errores

- Con esas suposiciones sobre el canal, la estrategia de corrección de errores es:
- Si nos llega una palabra w, deducir cual es la palabra v “mas probable" que nos hayan mandado y que pueda producir w.
- Pej, supongamos que para mandar u y que llegue w se tiene que producir un error en el camino.
- Pero que para mandar x y que llegue w se tienen que producir dos errores en el camino.
- Entonces la probabilidad de mandar u y que llegue w es p, pero la probabilidad de mandar x y que llegue w es p^2.
- Como $p^2 < p$, concluimos que es mas probable que la palabra mandada sea u.
- Obvio que no es IMPOSIBLE que hayan mandado x, solo es menos probable.

Estrategia de corrección de errores

- Los códigos de corrección de errores corrigen la palabra recibida a la palabra mas probable que se haya mandado.

Estrategia de corrección de errores

- Los códigos de corrección de errores corrigen la palabra recibida a la palabra mas probable que se haya mandado.
- Lo que veremos es cómo se diseñan estos códigos, y que algoritmos están ligados a ellos.

Estrategia de corrección de errores

- Los códigos de corrección de errores corrigen la palabra recibida a la palabra mas probable que se haya mandado.
- Lo que veremos es cómo se diseñan estos códigos, y que algoritmos están ligados a ellos.
- Ademas de su capacidad de corrección de errores, tambien es crucial cuanta información puede un código codificar.

Estrategia de corrección de errores

- Los códigos de corrección de errores corrigen la palabra recibida a la palabra mas probable que se haya mandado.
- Lo que veremos es cómo se diseñan estos códigos, y que algoritmos están ligados a ellos.
- Ademas de su capacidad de corrección de errores, tambien es crucial cuanta información puede un código codificar.
- Pej, si necesitamos codificar un conjunto de 4 instrucciones para un movil: avanzar (a), retroceder (r), girar a la izquierda (i) o derecha (d), con algún código, el código deberá tener 4 palabras.

Primer código de ejemplo

- Una posibilidad seria usar $C1 = \{00, 01, 10, 11\}$.

Primer código de ejemplo

- Una posibilidad seria usar $C1 = \{00, 01, 10, 11\}$.
- Cada palabra de este código tiene exactamente dos bits asi que es lo mas eficiente posible a la hora de codificar las intstrucciones anteriores.

Primer código de ejemplo

- Una posibilidad seria usar $C1 = \{00, 01, 10, 11\}$.
- Cada palabra de este código tiene exactamente dos bits asi que es lo mas eficiente posible a la hora de codificar las intstrucciones anteriores.
- Pej, asociar que 00 es r, 11 es a, 01 es d y 10 es i.

Primer código de ejemplo

- Una posibilidad seria usar $C1 = \{00, 01, 10, 11\}$.
- Cada palabra de este código tiene exactamente dos bits asi que es lo mas eficiente posible a la hora de codificar las intstrucciones anteriores.
- Pej, asociar que 00 es r, 11 es a, 01 es d y 10 es i.
- Pero tiene un gran problema.

Primer código de ejemplo

- El problema es que si se da la orden 10 (girar a izquierda) para pej esquivar un obstaculo, y hay un error de transmisión y el 0 se cambia a 1, entonces la palabra recibida será 11.

Primer código de ejemplo

- El problema es que si se da la orden 10 (girar a izquierda) para pej esquivar un obstaculo, y hay un error de transmisión y el 0 se cambia a 1, entonces la palabra recibida será 11.
- Que es avanzar, no esquivar el obstaculo.

Primer código de ejemplo

- El problema es que si se da la orden 10 (girar a izquierda) para pej esquivar un obstaculo, y hay un error de transmisión y el 0 se cambia a 1, entonces la palabra recibida será 11.
- Que es avanzar, no esquivar el obstaculo.
- Lo que queremos es un código que tambien tenga 4 palabras, pero que le permita al movil “darse cuenta” de una situación asi.

Segundo código de ejemplo

- Podemos usar $C2 = \{000, 101, 110, 011\}$.

Segundo código de ejemplo

- Podemos usar $C2 = \{000, 101, 110, 011\}$.
- Veamos que pasa si mandamos por ejemplo 110 y cambia un sólo bit.

Segundo código de ejemplo

- Podemos usar $C2 = \{000, 101, 110, 011\}$.
- Veamos que pasa si mandamos por ejemplo 110 y cambia un sólo bit.
- Puede llegar 100, 111 o 010, dependiendo cual sea al bit que cambia.

Segundo código de ejemplo

- Podemos usar $C2 = \{000, 101, 110, 011\}$.
- Veamos que pasa si mandamos por ejemplo 110 y cambia un sólo bit.
- Puede llegar 100, 111 o 010, dependiendo cual sea al bit que cambia.
- Ninguna de esas palabras está en $C2$, asi que el movil SABE que ha habido un error, y no ejecutará la orden.

Segundo código de ejemplo

- Podemos usar $C2 = \{000, 101, 110, 011\}$.
- Veamos que pasa si mandamos por ejemplo 110 y cambia un sólo bit.
- Puede llegar 100, 111 o 010, dependiendo cual sea al bit que cambia.
- Ninguna de esas palabras está en $C2$, asi que el movil SABE que ha habido un error, y no ejecutará la orden.
- Asi que $C2$ es mejor que $C1$.

Segundo código de ejemplo

- Podemos usar $C2 = \{000, 101, 110, 011\}$.
- Veamos que pasa si mandamos por ejemplo 110 y cambia un sólo bit.
- Puede llegar 100, 111 o 010, dependiendo cual sea al bit que cambia.
- Ninguna de esas palabras está en $C2$, asi que el movil SABE que ha habido un error, y no ejecutará la orden.
- Asi que $C2$ es mejor que $C1$.
- El problema es que si bien sabe que hay un error, no sabe cual es.

Segundo código de ejemplo

- En realidad esto pasará con todos los códigos, pero como explicamos antes, si bien no sabemos con 100% de seguridad cual es la palabra enviada, queremos ver cual es la palabra, de entre todas las posibles, que tenga la mayor probabilidad de haber sido enviadas.

Segundo código de ejemplo

- En realidad esto pasará con todos los códigos, pero como explicamos antes, si bien no sabemos con 100% de seguridad cual es la palabra enviada, queremos ver cual es la palabra, de entre todas las posibles, que tenga la mayor probabilidad de haber sido enviadas.
- Lo cual por la suposición sobre p, es equivalente a detectar cual es la palabra que necesita la menor cantidad de errores para que haya llegado la palabra que llegó.

Segundo código de ejemplo

- En realidad esto pasará con todos los códigos, pero como explicamos antes, si bien no sabemos con 100% de seguridad cual es la palabra enviada, queremos ver cual es la palabra, de entre todas las posibles, que tenga la mayor probabilidad de haber sido enviadas.
- Lo cual por la suposición sobre p, es equivalente a detectar cual es la palabra que necesita la menor cantidad de errores para que haya llegado la palabra que llegó.
- Pero en este caso, le llega 100, vemos que tanto 000, como 101 o 110 requieren sólo un error para que llegue 100.

Segundo código de ejemplo

- En realidad esto pasará con todos los códigos, pero como explicamos antes, si bien no sabemos con 100% de seguridad cual es la palabra enviada, queremos ver cual es la palabra, de entre todas las posibles, que tenga la mayor probabilidad de haber sido enviadas.
- Lo cual por la suposición sobre p, es equivalente a detectar cual es la palabra que necesita la menor cantidad de errores para que haya llegado la palabra que llegó.
- Pero en este caso, le llega 100, vemos que tanto 000, como 101 o 110 requieren sólo un error para que llegue 100.
- Todas igualmente probables.

- Algunas veces basta con determinar que hubo un error, aún si no se puede corregir, si se está en una situación donde se puede pedir retransmisión.

- Algunas veces basta con determinar que hubo un error, aún si no se puede corregir, si se está en una situación donde se puede pedir retransmisión.
- Pero muchas veces no podemos pedir retransmisión.

- Algunas veces basta con determinar que hubo un error, aún si no se puede corregir, si se está en una situación donde se puede pedir retransmisión.
- Pero muchas veces no podemos pedir retransmisión.
- Pej, si estan escuchando su música favorita, codificada con alguno de estos códigos, si se produce un error no es lógico esperar que se pause la reproducción y mandarle a preguntar al que grabó la musica que nos re-envie los bits que necesitamos.

- Algunas veces basta con determinar que hubo un error, aún si no se puede corregir, si se está en una situación donde se puede pedir retransmisión.
- Pero muchas veces no podemos pedir retransmisión.
- Pej, si estan escuchando su música favorita, codificada con alguno de estos códigos, si se produce un error no es lógico esperar que se pause la reproducción y mandarle a preguntar al que grabó la musica que nos re-envie los bits que necesitamos.
- Asi que en realidad estamos mas interesados en corregir que en detectar

- Algunas veces basta con determinar que hubo un error, aún si no se puede corregir, si se está en una situación donde se puede pedir retransmisión.
- Pero muchas veces no podemos pedir retransmisión.
- Pej, si estan escuchando su música favorita, codificada con alguno de estos códigos, si se produce un error no es lógico esperar que se pause la reproducción y mandarle a preguntar al que grabó la musica que nos re-envie los bits que necesitamos.
- Asi que en realidad estamos mas interesados en corregir que en detectar
- El siguiente paso seria un código en el cual las probabilidades de las posibles palabras enviadas no fueran todas iguales, para poder elegir la mas probable

Tercer código para el carrito en Marte

- Podemos usar $C3 = \{000111, 101010, 110001, 011100\}$.

Tercer código para el carrito en Marte

- Podemos usar $C3 = \{000111, 101010, 110001, 011100\}$.
- Supongamos que mandamos 110001 y se produce un error en un bit.

Tercer código para el carrito en Marte

- Podemos usar $C3 = \{000111, 101010, 110001, 011100\}$.
- Supongamos que mandamos 110001 y se produce un error en un bit.
- Pueden llegar $010001, 100001, 111001, 110101, 110011, 110000$

Tercer código para el carrito en Marte

- Podemos usar $C3 = \{000111, 101010, 110001, 011100\}$.
- Supongamos que mandamos 110001 y se produce un error en un bit.
- Pueden llegar $010001, 100001, 111001, 110101, 110011, 110000$
- Ninguno de los cuales está en $C3$, asi que, al igual que $C2$, el movil sabe que ha habido un error de transmisión.

Tercer código para el carrito en Marte

- Podemos usar $C3 = \{000111, 101010, 110001, 011100\}$.
- Supongamos que mandamos 110001 y se produce un error en un bit.
- Pueden llegar $010001, 100001, 111001, 110101, 110011, 110000$
- Ninguno de los cuales está en $C3$, asi que, al igual que $C2$, el movil sabe que ha habido un error de transmisión.
- Pero $C3$ necesita 6 bits, mientras que $C2$ sólo 3 bits. ¿Hay alguna ventaja de C3 sobre C2 que justifique usar el doble de bits?

Tercer código para el carrito en Marte

- Podemos usar $C3 = \{000111, 101010, 110001, 011100\}$.
- Supongamos que mandamos 110001 y se produce un error en un bit.
- Pueden llegar $010001, 100001, 111001, 110101, 110011, 110000$
- Ninguno de los cuales está en $C3$, asi que, al igual que $C2$, el movil sabe que ha habido un error de transmisión.
- Pero $C3$ necesita 6 bits, mientras que $C2$ sólo 3 bits. ¿Hay alguna ventaja de C3 sobre C2 que justifique usar el doble de bits?
- Bueno, pensemos que pasa si llega pej 010001.

Tercer código para el carrito en Marte

- Para cada palabra del código, ¿Cual es la probabilidad de que se haya enviado esa palabra dado que ser recibió 010001?

Tercer código para el carrito en Marte

- Para cada palabra del código, ¿Cual es la probabilidad de que se haya enviado esa palabra dado que ser recibió 010001?
- Para llegar de 000111 a 010001, necesitamos 3 errores. Probabilidad p^3.

Tercer código para el carrito en Marte

- Para cada palabra del código, ¿Cual es la probabilidad de que se haya enviado esa palabra dado que ser recibió 010001?
- Para llegar de 000111 a 010001, necesitamos 3 errores. Probabilidad p^3.
- Veamos las otras palabras de C3:

Tercer código para el carrito en Marte

- Para cada palabra del código, ¿Cual es la probabilidad de que se haya enviado esa palabra dado que ser recibió 010001?
- Para llegar de 000111 a 010001, necesitamos 3 errores. Probabilidad p^3.
- Veamos las otras palabras de C3:
  - 101010 a 010001 son 5 errores. Probabilidad p^5.

Tercer código para el carrito en Marte

- Para cada palabra del código, ¿Cual es la probabilidad de que se haya enviado esa palabra dado que ser recibió 010001?
- Para llegar de 000111 a 010001, necesitamos 3 errores. Probabilidad p^3.
- Veamos las otras palabras de C3:
  - 101010 a 010001 son 5 errores. Probabilidad p^5.
  - 110001 a 010001 es 1 error. Probabilidad p.

Tercer código para el carrito en Marte

- Para cada palabra del código, ¿Cual es la probabilidad de que se haya enviado esa palabra dado que ser recibió 010001?
- Para llegar de 000111 a 010001, necesitamos 3 errores. Probabilidad p^3.
- Veamos las otras palabras de C3:
  - 101010 a 010001 son 5 errores. Probabilidad p^5.
  - 110001 a 010001 es 1 error. Probabilidad p.
  - 011100 a 010001 son 3 errores. Probabilidad p^3.

Tercer código para el carrito en Marte

- Para cada palabra del código, ¿Cual es la probabilidad de que se haya enviado esa palabra dado que ser recibió 010001?
- Para llegar de 000111 a 010001, necesitamos 3 errores. Probabilidad p^3.
- Veamos las otras palabras de C3:
  - 101010 a 010001 son 5 errores. Probabilidad p^5.
  - 110001 a 010001 es 1 error. Probabilidad p.
  - 011100 a 010001 son 3 errores. Probabilidad p^3.
- Entonces aca podemos deducir que la palabra mas probable enviada es 110001.

Tercer código para el carrito en Marte

- Para cada palabra del código, ¿Cual es la probabilidad de que se haya enviado esa palabra dado que ser recibió 010001?
- Para llegar de 000111 a 010001, necesitamos 3 errores. Probabilidad p^3.
- Veamos las otras palabras de C3:
  - 101010 a 010001 son 5 errores. Probabilidad p^5.
  - 110001 a 010001 es 1 error. Probabilidad p.
  - 011100 a 010001 son 3 errores. Probabilidad p^3.
- Entonces aca podemos deducir que la palabra mas probable enviada es 110001.
- Veremos que podemos hacer esto con toda palabra recibida, asi que esto es una ventaja de C3 sobre C2.

Tercer código para el carrito en Marte

- Para cada palabra del código, ¿Cual es la probabilidad de que se haya enviado esa palabra dado que ser recibió 010001?
- Para llegar de 000111 a 010001, necesitamos 3 errores. Probabilidad p^3.
- Veamos las otras palabras de C3:
  - 101010 a 010001 son 5 errores. Probabilidad p^5.
  - 110001 a 010001 es 1 error. Probabilidad p.
  - 011100 a 010001 son 3 errores. Probabilidad p^3.
- Entonces aca podemos deducir que la palabra mas probable enviada es 110001.
- Veremos que podemos hacer esto con toda palabra recibida, asi que esto es una ventaja de C3 sobre C2.
- Necesitamos algunas herramientas.

Distancia entre palabras

Definición:

Dadas dos palabras $v, w \in \{0, 1\}^n$, la distancia de Hamming (o simplemente "distancia") entre v, w es

$$d(v, w) = d_H(v, w) = \#\{\text{bits de diferencia entre } v \text{ y } w\}$$

Distancia entre palabras

Definición:

Dadas dos palabras $v, w \in \{0, 1\}^n$, la distancia de Hamming (o simplemente "distancia") entre v, w es

$$d(v, w) = d_H(v, w) = \#\{\text{bits de diferencia entre } v \text{ y } w\}$$

Definimos ademas, dado un código C:

$$\delta(C) = \text{Min}\{d_H(v, w) : v, w \in C, v \neq w\}$$

Distancia entre palabras

Definición:

Dadas dos palabras $v, w \in \{0,1\}^n$, la distancia de Hamming (o simplemente "distancia") entre v, w es

$$d(v,w) = d_H(v,w) = \#\{\text{bits de diferencia entre } v \text{ y } w\}$$

Definimos ademas, dado un código C:

$$\delta(C) = \text{Min}\{d_H(v,w) : v, w \in C, v \neq w\}$$

Calculemos distancias entre palabras y δs para los códigos que dimos de ejemplo.

δ para C1

	00	01	10	11
00	0	1	1	2
01	1	0	2	1
10	1	2	0	1
11	2	1	1	0

$$\delta(C1) = 1$$

δ para C2

	000	101	110	011
000	0	2	2	2
101	2	0	2	2
110	2	2	0	2
011	2	2	2	0

$$\delta(C2) = 2$$

δ para C3

	000111	101010	110001	011100
000111	0	4	4	4
101010	4	0	4	4
110001	4	4	0	4
011100	4	4	4	0

$$\delta(C3) = 4$$

La distancia de Hamming es una distancia

- ¿Para qué sirve calcular δ?

La distancia de Hamming es una distancia

- ¿Para qué sirve calcular δ?
- Enseguida lo veremos, pero primero, probemos que el nombre de “distancia” es correcto.

La distancia de Hamming es una distancia

- ¿Para qué sirve calcular δ?
- Enseguida lo veremos, pero primero, probemos que el nombre de “distancia” es correcto.

Propiedad:

La distancia de Hamming es una distancia, es decir:

La distancia de Hamming es una distancia

- ¿Para qué sirve calcular δ?
- Enseguida lo veremos, pero primero, probemos que el nombre de "distancia" es correcto.

Propiedad:

La distancia de Hamming es una distancia, es decir:

1. $d_H(v, w) = d_H(w, v)$

La distancia de Hamming es una distancia

- ¿Para qué sirve calcular δ?
- Enseguida lo veremos, pero primero, probemos que el nombre de "distancia" es correcto.

Propiedad:

La distancia de Hamming es una distancia, es decir:

1. $d_H(v, w) = d_H(w, v)$
2. $d_H(v, w) \geq 0$

La distancia de Hamming es una distancia

- ¿Para qué sirve calcular δ?
- Enseguida lo veremos, pero primero, probemos que el nombre de "distancia" es correcto.

Propiedad:

La distancia de Hamming es una distancia, es decir:

1. $d_H(v, w) = d_H(w, v)$
2. $d_H(v, w) \geq 0$
3. $d_H(v.w) = 0 \iff v = w$

La distancia de Hamming es una distancia

- ¿Para qué sirve calcular δ?
- Enseguida lo veremos, pero primero, probemos que el nombre de "distancia" es correcto.

Propiedad:

La distancia de Hamming es una distancia, es decir:

1. $d_H(v, w) = d_H(w, v)$
2. $d_H(v, w) \geq 0$
3. $d_H(v.w) = 0 \iff v = w$
4. $d_H(v.w) \leq d_H(v.u) + d_H(u, w)$ (desigualdad triangular).

Prueba

- 1) y 2) son obvias, pues estoy contando cuantos bits de diferencia hay entre v y w.

Prueba

- 1) y 2) son obvias, pues estoy contando cuantos bits de diferencia hay entre v y w.
- 3) tambien, pues si no hay bits de diferencia, entonces $v = w$, y si $v = w$, entonces no hay bits de diferencia.

Prueba

- 1) y 2) son obvias, pues estoy contando cuantos bits de diferencia hay entre v y w.
- 3) tambien, pues si no hay bits de diferencia, entonces $v = w$, y si $v = w$, entonces no hay bits de diferencia.
- Asi que lo único no obvio es la desigualdad triangular

Prueba

- 1) y 2) son obvias, pues estoy contando cuantos bits de diferencia hay entre v y w.
- 3) tambien, pues si no hay bits de diferencia, entonces $v = w$, y si $v = w$, entonces no hay bits de diferencia.
- Asi que lo único no obvio es la desigualdad triangular
- Sean $A = \{i : v_i = w_i\}$,$B = \{i : v_i = u_i\}$,$C = \{i : u_i = w_i\}$.

Prueba

- 1) y 2) son obvias, pues estoy contando cuantos bits de diferencia hay entre v y w.
- 3) tambien, pues si no hay bits de diferencia, entonces $v = w$, y si $v = w$, entonces no hay bits de diferencia.
- Asi que lo único no obvio es la desigualdad triangular
- Sean $A = \{i : v_i = w_i\}$,$B = \{i : v_i = u_i\}$,$C = \{i : u_i = w_i\}$.
- $i \in B \cap C \Rightarrow (v_i = u_i \text{ y } u_i = w_i) \Rightarrow v_i = w_i \Rightarrow i \in A$

Prueba

- 1) y 2) son obvias, pues estoy contando cuantos bits de diferencia hay entre v y w.
- 3) tambien, pues si no hay bits de diferencia, entonces $v = w$, y si $v = w$, entonces no hay bits de diferencia.
- Asi que lo único no obvio es la desigualdad triangular
- Sean $A = \{i : v_i = w_i\}$,$B = \{i : v_i = u_i\}$,$C = \{i : u_i = w_i\}$.
- $i \in B \cap C \Rightarrow (v_i = u_i \text{ y } u_i = w_i) \Rightarrow v_i = w_i \Rightarrow i \in A$
- Por lo tanto, $B \cap C \subseteq A$.

Prueba

- 1) y 2) son obvias, pues estoy contando cuantos bits de diferencia hay entre v y w.
- 3) tambien, pues si no hay bits de diferencia, entonces $v = w$, y si $v = w$, entonces no hay bits de diferencia.
- Asi que lo único no obvio es la desigualdad triangular
- Sean $A = \{i : v_i = w_i\}$,$B = \{i : v_i = u_i\}$,$C = \{i : u_i = w_i\}$.
- $i \in B \cap C \Rightarrow (v_i = u_i \text{ y } u_i = w_i) \Rightarrow v_i = w_i \Rightarrow i \in A$
- Por lo tanto, $B \cap C \subseteq A$.
- Si denotamos al complemento de un conjunto con una barra por arriba del mismo, entonces tenemos que:

Prueba

- 1) y 2) son obvias, pues estoy contando cuantos bits de diferencia hay entre v y w.
- 3) tambien, pues si no hay bits de diferencia, entonces $v = w$, y si $v = w$, entonces no hay bits de diferencia.
- Asi que lo único no obvio es la desigualdad triangular
- Sean $A = \{i : v_i = w_i\}$,$B = \{i : v_i = u_i\}$,$C = \{i : u_i = w_i\}$.
- $i \in B \cap C \Rightarrow (v_i = u_i \text{ y } u_i = w_i) \Rightarrow v_i = w_i \Rightarrow i \in A$
- Por lo tanto, $B \cap C \subseteq A$.
- Si denotamos al complemento de un conjunto con una barra por arriba del mismo, entonces tenemos que:
- $\overline{A} \subseteq \overline{B \cap C}$ (pues el complemento invierte la inclusión)

Prueba

- De acuerdo con las leyes de De Morgan $\overline{B \cap C} = \overline{B} \cup \overline{C}$, asi que :

Prueba

- De acuerdo con las leyes de De Morgan $\overline{B \cap C} = \overline{B} \cup \overline{C}$, asi que :
- $\overline{A} \subseteq \overline{B} \cup \overline{C}$.

Prueba

- De acuerdo con las leyes de De Morgan $\overline{B \cap C} = \overline{B} \cup \overline{C}$, asi que :
- $\overline{A} \subseteq \overline{B} \cup \overline{C}$.
- Tomando cardinalidades, obtenemos que

Prueba

- De acuerdo con las leyes de De Morgan $\overline{B \cap C} = \overline{B} \cup \overline{C}$, asi que :
- $\overline{A} \subseteq \overline{B} \cup \overline{C}$.
- Tomando cardinalidades, obtenemos que
- $\#\overline{A} \leq \#(\overline{B} \cup \overline{C})$.

Prueba

- De acuerdo con las leyes de De Morgan $\overline{B \cap C} = \overline{B} \cup \overline{C}$, asi que :
- $\overline{A} \subseteq \overline{B} \cup \overline{C}$.
- Tomando cardinalidades, obtenemos que
- $\#\overline{A} \leq \#(\overline{B} \cup \overline{C})$.
- Pero $\#(\overline{B} \cup \overline{C}) \leq \#\overline{B} + \#\overline{C}$ asi que:

Prueba

- De acuerdo con las leyes de De Morgan $\overline{B \cap C} = \overline{B} \cup \overline{C}$, asi que :
- $\overline{A} \subseteq \overline{B} \cup \overline{C}$.
- Tomando cardinalidades, obtenemos que
- $\#\overline{A} \leq \#(\overline{B} \cup \overline{C})$.
- Pero $\#(\overline{B} \cup \overline{C}) \leq \#\overline{B} + \#\overline{C}$ asi que:
- $\#\overline{A} \leq \#\overline{B} + \#\overline{C}$

Prueba

- De acuerdo con las leyes de De Morgan $\overline{B \cap C} = \overline{B} \cup \overline{C}$, asi que :
- $\overline{A} \subseteq \overline{B} \cup \overline{C}$.
- Tomando cardinalidades, obtenemos que
- $\#\overline{A} \leq \#(\overline{B} \cup \overline{C})$.
- Pero $\#(\overline{B} \cup \overline{C}) \leq \#\overline{B} + \#\overline{C}$ asi que:
- $\#\overline{A} \leq \#\overline{B} + \#\overline{C}$
- Como $\#\overline{A} = d_H(v.w)$, $\#\overline{B} = d_H(v,u)$ y $\#\overline{C} = d_H(u,w)$, hemos entonces probado la desigualdad triangular.

Definición:

Dada una palabra $v \in \{0,1\}^n$, y un número natural $r \geq 0$, definimos el disco de radio r alrededor de v como

$$D_r(v) = \{w \in \{0,1\}^n : d_H(v,w) \leq r\}$$

Definición:

Dada una palabra $v \in \{0,1\}^n$, y un número natural $r \geq 0$, definimos el disco de radio r alrededor de v como

$$D_r(v) = \{w \in \{0,1\}^n : d_H(v,w) \leq r\}$$

- Esto nos permite definir formalmente la idea intuitiva que tenemos de detectar y corregir errores.

Definición:

Dada una palabra $v \in \{0,1\}^n$, y un número natural $r \geq 0$, definimos el disco de radio r alrededor de v como

$$D_r(v) = \{w \in \{0,1\}^n : d_H(v,w) \leq r\}$$

- Esto nos permite definir formalmente la idea intuitiva que tenemos de detectar y corregir errores.
- Queremos detectar r errores si al mandar una palabra v y llegar w con r errores o menos, nos damos cuenta de que hubo un error porque w no está en el código.

Definición:

Dada una palabra $v \in \{0,1\}^n$, y un número natural $r \geq 0$, definimos el disco de radio r alrededor de v como

$$D_r(v) = \{w \in \{0,1\}^n : d_H(v,w) \leq r\}$$

- Esto nos permite definir formalmente la idea intuitiva que tenemos de detectar y corregir errores.
- Queremos detectar r errores si al mandar una palabra v y llegar w con r errores o menos, nos damos cuenta de que hubo un error porque w no está en el código.
- Y vamos a poder corregirlo si v es la única palabra que puede haber provocado r errores o menos y que llegue w.

Definición:

Dada una palabra $v \in \{0,1\}^n$, y un número natural $r \geq 0$, definimos el disco de radio r alrededor de v como

$$D_r(v) = \{w \in \{0,1\}^n : d_H(v,w) \leq r\}$$

- Esto nos permite definir formalmente la idea intuitiva que tenemos de detectar y corregir errores.
- Queremos detectar r errores si al mandar una palabra v y llegar w con r errores o menos, nos damos cuenta de que hubo un error porque w no está en el código.
- Y vamos a poder corregirlo si v es la única palabra que puede haber provocado r errores o menos y que llegue w.
- Asi que definimos:

Definición:

Definición:

- Un código C detecta r errores si $D_r(v) \cap C = \{v\} \quad \forall v \in C$.

Definición:

- Un código C detecta r errores si $D_r(v) \cap C = \{v\} \quad \forall v \in C$.
- Un código C corrige r errores si
  $D_r(v) \cap D_r(w) = \emptyset \quad \forall v, w \in C : v \neq w$.

Definición:

- Un código C detecta r errores si $D_r(v) \cap C = \{v\} \quad \forall v \in C$.
- Un código C corrige r errores si $D_r(v) \cap D_r(w) = \emptyset \quad \forall v, w \in C : v \neq w$.

Ahora podemos enunciar el teorema que es fundamental en toda la teoría de códigos de corrección de errores.

Teorema de δ

Sea C un código y $\delta = \delta(C)$. Entonces:

Teorema de δ

Sea C un código y $\delta = \delta(C)$. Entonces:

1. C detecta $\delta - 1$ errores, pero no detecta δ.

Teorema de δ

Sea C un código y $\delta = \delta(C)$. Entonces:

1. C detecta $\delta - 1$ errores, pero no detecta δ.
2. Si $t = \lfloor \frac{\delta - 1}{2} \rfloor$ entonces C corrige t errores pero no corrige $t + 1$ errores.

Teorema de δ

Sea C un código y $\delta = \delta(C)$. Entonces:

1. C detecta $\delta - 1$ errores, pero no detecta δ.
2. Si $t = \lfloor \frac{\delta-1}{2} \rfloor$ entonces C corrige t errores pero no corrige $t + 1$ errores.

donde $\lfloor . \rfloor$ es la parte entera inferior de un número, es decir $\lfloor x \rfloor$ es el mayor entero n con $n \leq x$.

Prueba del Teorema

- Veamos primero que detecta $\delta - 1$ errores.

Prueba del Teorema

- Veamos primero que detecta $\delta - 1$ errores.
- Sea $v \in C$ y $x \in D_{\delta-1}(v) \cap C$.

Prueba del Teorema

- Veamos primero que detecta $\delta - 1$ errores.
- Sea $v \in C$ y $x \in D_{\delta-1}(v) \cap C$.
- Por definición de D, tenemos que $d_H(v, x) \leq \delta - 1 < \delta$

Prueba del Teorema

- Veamos primero que detecta $\delta - 1$ errores.
- Sea $v \in C$ y $x \in D_{\delta-1}(v) \cap C$.
- Por definición de D, tenemos que $d_H(v, x) \leq \delta - 1 < \delta$
- Como $d_H(v, x)$ es menor que δ, que es la menor distancia entre palabras distintas de C, concluimos que $x = v$.

Prueba del Teorema

- Veamos primero que detecta $\delta - 1$ errores.
- Sea $v \in C$ y $x \in D_{\delta-1}(v) \cap C$.
- Por definición de D, tenemos que $d_H(v,x) \leq \delta - 1 < \delta$
- Como $d_H(v,x)$ es menor que δ, que es la menor distancia entre palabras distintas de C, concluimos que $x = v$.
- Por lo tanto $D_{\delta-1}(v) \cap C = \{v\}$ y C detecta $\delta - 1$ errores.

Prueba del Teorema, la otra parte del punto 1)

- Ahora veamos que NO detecta δ errores.

Prueba del Teorema, la otra parte del punto 1)

- Ahora veamos que NO detecta δ errores.
- Como $\delta =$Min$\{d_H(v, w) : v, w \in C, v \neq w\}$ entonces existen $v, w \in C, v \neq w$ con $\delta = d_H(v, w)$.

Prueba del Teorema, la otra parte del punto 1)

- Ahora veamos que NO detecta δ errores.
- Como $\delta =$Min$\{d_H(v, w) : v, w \in C, v \neq w\}$ entonces existen $v, w \in C, v \neq w$ con $\delta = d_H(v, w)$.
- Por definición de D,esto dice que $w \in D_\delta(v)$.

Prueba del Teorema, la otra parte del punto 1)

- Ahora veamos que NO detecta δ errores.
- Como $\delta =$Min$\{d_H(v,w) : v,w \in C, v \neq w\}$ entonces existen $v,w \in C, v \neq w$ con $\delta = d_H(v,w)$.
- Por definición de D,esto dice que $w \in D_\delta(v)$.
- Como tambien está en C, concluimos que $w \in D_\delta(v) \cap C$.

Prueba del Teorema, la otra parte del punto 1)

- Ahora veamos que NO detecta δ errores.
- Como $\delta =$Min$\{d_H(v,w) : v,w \in C, v \neq w\}$ entonces existen $v,w \in C, v \neq w$ con $\delta = d_H(v,w)$.
- Por definición de D,esto dice que $w \in D_\delta(v)$.
- Como tambien está en C, concluimos que $w \in D_\delta(v) \cap C$.
- Como $w \neq v$, entonces $D_\delta(v) \cap C \neq \{v\}$ y C no detecta δ errores.

Prueba del Teorema, primera parte del punto 2)

- Ahora veamos que corrige t errores. Supongamos que no sea cierto.

Prueba del Teorema, primera parte del punto 2)

- Ahora veamos que corrige t errores. Supongamos que no sea cierto.
- Entonces existen $v, w \in C, v \neq w$ con $D_t(v) \cap D_t(w) \neq \emptyset$.

Prueba del Teorema, primera parte del punto 2)

- Ahora veamos que corrige t errores. Supongamos que no sea cierto.
- Entonces existen $v, w \in C, v \neq w$ con $D_t(v) \cap D_t(w) \neq \emptyset$.
- Por lo tanto existe $u \in D_t(v) \cap D_t(w)$.

Prueba del Teorema, primera parte del punto 2)

- Ahora veamos que corrige t errores. Supongamos que no sea cierto.
- Entonces existen $v, w \in C, v \neq w$ con $D_t(v) \cap D_t(w) \neq \emptyset$.
- Por lo tanto existe $u \in D_t(v) \cap D_t(w)$.
- Entonces $d_H(v, u) \leq t$ y $d_H(u, w) \leq t$.

Prueba del Teorema, primera parte del punto 2)

- Ahora veamos que corrige t errores. Supongamos que no sea cierto.
- Entonces existen $v, w \in C, v \neq w$ con $D_t(v) \cap D_t(w) \neq \emptyset$.
- Por lo tanto existe $u \in D_t(v) \cap D_t(w)$.
- Entonces $d_H(v, u) \leq t$ y $d_H(u, w) \leq t$.
- Por la desigualdad triangular, $d_H(v, w) \leq d_H(v, u) + d_H(u, w) \leq 2t$.

Prueba del Teorema, primera parte del punto 2)

- Ahora veamos que corrige t errores. Supongamos que no sea cierto.
- Entonces existen $v, w \in C, v \neq w$ con $D_t(v) \cap D_t(w) \neq \emptyset$.
- Por lo tanto existe $u \in D_t(v) \cap D_t(w)$.
- Entonces $d_H(v, u) \leq t$ y $d_H(u, w) \leq t$.
- Por la desigualdad triangular, $d_H(v, w) \leq d_H(v, u) + d_H(u, w) \leq 2t$.
- Como δ es la menor distancia entre palabras distintas del código y $v \neq w$ y $v, w \in C$, concluimos que:

Prueba del Teorema, primera parte del punto 2)

- Ahora veamos que corrige t errores. Supongamos que no sea cierto.
- Entonces existen $v, w \in C, v \neq w$ con $D_t(v) \cap D_t(w) \neq \emptyset$.
- Por lo tanto existe $u \in D_t(v) \cap D_t(w)$.
- Entonces $d_H(v, u) \leq t$ y $d_H(u, w) \leq t$.
- Por la desigualdad triangular, $d_H(v, w) \leq d_H(v, u) + d_H(u, w) \leq 2t$.
- Como δ es la menor distancia entre palabras distintas del código y $v \neq w$ y $v, w \in C$, concluimos que:
- $\delta \leq d_H(v, w) \leq 2t$.

Prueba del Teorema, primera parte del punto 2)

- Ahora veamos que corrige t errores. Supongamos que no sea cierto.
- Entonces existen $v, w \in C, v \neq w$ con $D_t(v) \cap D_t(w) \neq \emptyset$.
- Por lo tanto existe $u \in D_t(v) \cap D_t(w)$.
- Entonces $d_H(v, u) \leq t$ y $d_H(u, w) \leq t$.
- Por la desigualdad triangular, $d_H(v, w) \leq d_H(v, u) + d_H(u, w) \leq 2t$.
- Como δ es la menor distancia entre palabras distintas del código y $v \neq w$ y $v, w \in C$, concluimos que:
- $\delta \leq d_H(v, w) \leq 2t$.
- Entonces $\frac{\delta}{2} \leq t = \lfloor \frac{\delta-1}{2} \rfloor \leq \frac{\delta-1}{2}$, absurdo.

Prueba del Teorema, segunda parte del punto 2)

- Esta es la parte mas complicada.

Prueba del Teorema, segunda parte del punto 2)

- Esta es la parte mas complicada.
- Como $\delta =$Min$\{d_H(v, w) : v, w \in C, v \neq w\}$ entonces existen $v, w \in C, v \neq w$ con $\delta = d_H(v, w)$.

Prueba del Teorema, segunda parte del punto 2)

- Esta es la parte mas complicada.
- Como $\delta =$Min$\{d_H(v,w) : v,w \in C, v \neq w\}$ entonces existen $v,w \in C, v \neq w$ con $\delta = d_H(v,w)$.
- Vamos a probar que para ese v,w, $D_{t+1}(v) \cap D_{t+1}(w) \neq \emptyset$ con lo cual veremos que C no corrige $t+1$ errores.

Prueba del Teorema, segunda parte del punto 2)

- Esta es la parte mas complicada.
- Como $\delta =$Min$\{d_H(v,w) : v,w \in C, v \neq w\}$ entonces existen $v,w \in C, v \neq w$ con $\delta = d_H(v,w)$.
- Vamos a probar que para ese v,w, $D_{t+1}(v) \cap D_{t+1}(w) \neq \emptyset$ con lo cual veremos que C no corrige $t+1$ errores.
- Para probar $D_{t+1}(v) \cap D_{t+1}(w) \neq \emptyset$ vamos a tener que construir un u en esa intersección.

Prueba del Teorema, segunda parte del punto 2)

- Esta es la parte mas complicada.
- Como $\delta =$Min$\{d_H(v,w) : v,w \in C, v \neq w\}$ entonces existen $v,w \in C, v \neq w$ con $\delta = d_H(v,w)$.
- Vamos a probar que para ese v,w, $D_{t+1}(v) \cap D_{t+1}(w) \neq \emptyset$ con lo cual veremos que C no corrige $t+1$ errores.
- Para probar $D_{t+1}(v) \cap D_{t+1}(w) \neq \emptyset$ vamos a tener que construir un u en esa intersección.
- Ahora bien, como $\delta = d_H(v,w)$, entonces v y w son distintos en exactamente δ bits.

Prueba del Teorema, segunda parte del punto 2)

- Esta es la parte mas complicada.
- Como $\delta =$Min$\{d_H(v, w) : v, w \in C, v \neq w\}$ entonces existen $v, w \in C, v \neq w$ con $\delta = d_H(v, w)$.
- Vamos a probar que para ese v, w, $D_{t+1}(v) \cap D_{t+1}(w) \neq \emptyset$ con lo cual veremos que C no corrige $t + 1$ errores.
- Para probar $D_{t+1}(v) \cap D_{t+1}(w) \neq \emptyset$ vamos a tener que construir un u en esa intersección.
- Ahora bien, como $\delta = d_H(v, w)$, entonces v y w son distintos en exactamente δ bits.
- Tomemos $t + 1$ de esos bits, y cambiemos los bits de v en esos lugares, y llamemos u al resultado.

Prueba del Teorema, segunda parte del punto 2)

- Esta es la parte mas complicada.
- Como $\delta =$Min$\{d_H(v,w) : v,w \in C, v \neq w\}$ entonces existen $v,w \in C, v \neq w$ con $\delta = d_H(v,w)$.
- Vamos a probar que para ese v,w, $D_{t+1}(v) \cap D_{t+1}(w) \neq \emptyset$ con lo cual veremos que C no corrige $t+1$ errores.
- Para probar $D_{t+1}(v) \cap D_{t+1}(w) \neq \emptyset$ vamos a tener que construir un u en esa intersección.
- Ahora bien, como $\delta = d_H(v,w)$, entonces v y w son distintos en exactamente δ bits.
- Tomemos $t+1$ de esos bits, y cambiemos los bits de v en esos lugares, y llamemos u al resultado.
- Por construcción, u es distinto de v en exactamente $t+1$ bits, asi que $d_H(v,u) = t+1$ y $u \in D_{t+1}(v)$.

Prueba del Teorema, segunda parte del punto 2)

- Esta es la parte mas complicada.
- Como $\delta =$Min$\{d_H(v,w) : v,w \in C, v \neq w\}$ entonces existen $v,w \in C, v \neq w$ con $\delta = d_H(v,w)$.
- Vamos a probar que para ese v,w, $D_{t+1}(v) \cap D_{t+1}(w) \neq \emptyset$ con lo cual veremos que C no corrige $t+1$ errores.
- Para probar $D_{t+1}(v) \cap D_{t+1}(w) \neq \emptyset$ vamos a tener que construir un u en esa intersección.
- Ahora bien, como $\delta = d_H(v,w)$, entonces v y w son distintos en exactamente δ bits.
- Tomemos $t+1$ de esos bits, y cambiemos los bits de v en esos lugares, y llamemos u al resultado.
- Por construcción, u es distinto de v en exactamente $t+1$ bits, asi que $d_H(v,u) = t+1$ y $u \in D_{t+1}(v)$.
- Asi que sólo necesitamos probar que $u \in D_{t+1}(w)$.

continuación prueba segunda parte del punto 2)

- Ahora bien, v y w diferian en δ bits, y u difiere de v en $t+1$ de esos δ bits y coincide en los otros.

continuación prueba segunda parte del punto 2)

- Ahora bien, v y w diferian en δ bits, y u difiere de v en $t+1$ de esos δ bits y coincide en los otros.
- Por lo tanto, en esos $t+1$ bits, u coincide con w. Sólo difiere en los otros $\delta-(t+1)$ bits.

continuación prueba segunda parte del punto 2)

- Ahora bien, v y w diferian en δ bits, y u difiere de v en $t+1$ de esos δ bits y coincide en los otros.
- Por lo tanto, en esos $t+1$ bits, u coincide con w. Sólo difiere en los otros $\delta-(t+1)$ bits.
- Concluimos que $d_H(u,w)=\delta-(t+1)$ y $u \in D_{\delta-(t+1)}(w)$.

continuación prueba segunda parte del punto 2)

- Ahora bien, v y w diferian en δ bits, y u difiere de v en $t+1$ de esos δ bits y coincide en los otros.
- Por lo tanto, en esos $t+1$ bits, u coincide con w. Sólo difiere en los otros $\delta-(t+1)$ bits.
- Concluimos que $d_H(u, w) = \delta-(t+1)$ y $u \in D_{\delta-(t+1)}(w)$.
- Asi que para completar la prueba sólo necesitamos ver que $\delta-(t+1) \leq t+1$ lo cual ocurre sii $\delta \leq 2(t+1)$.(*)

continuación prueba segunda parte del punto 2)

- Ahora bien, v y w diferian en δ bits, y u difiere de v en $t+1$ de esos δ bits y coincide en los otros.
- Por lo tanto, en esos $t+1$ bits, u coincide con w. Sólo difiere en los otros $\delta-(t+1)$ bits.
- Concluimos que $d_H(u,w)=\delta-(t+1)$ y $u\in D_{\delta-(t+1)}(w)$.
- Asi que para completar la prueba sólo necesitamos ver que $\delta-(t+1)\leq t+1$ lo cual ocurre sii $\delta\leq 2(t+1)$.(*)
- Si δ es impar, digamos $\delta=2k+1$, entonces $t=\lfloor\frac{\delta-1}{2}\rfloor=k$.

continuación prueba segunda parte del punto 2)

- Ahora bien, v y w diferian en δ bits, y u difiere de v en $t+1$ de esos δ bits y coincide en los otros.
- Por lo tanto, en esos $t+1$ bits, u coincide con w. Sólo difiere en los otros $\delta-(t+1)$ bits.
- Concluimos que $d_H(u, w) = \delta-(t+1)$ y $u \in D_{\delta-(t+1)}(w)$.
- Asi que para completar la prueba sólo necesitamos ver que $\delta-(t+1) \leq t+1$ lo cual ocurre sii $\delta \leq 2(t+1)$.(*)
- Si δ es impar, digamos $\delta = 2k+1$, entonces $t = \lfloor \frac{\delta-1}{2} \rfloor = k$.
- En este caso tenemos $\delta = 2k+1 = 2t+1 < 2t+2$ y (*) se cumple.

continuación prueba segunda parte del punto 2)

- Ahora bien, v y w diferian en δ bits, y u difiere de v en $t+1$ de esos δ bits y coincide en los otros.
- Por lo tanto, en esos $t+1$ bits, u coincide con w. Sólo difiere en los otros $\delta-(t+1)$ bits.
- Concluimos que $d_H(u,w)=\delta-(t+1)$ y $u\in D_{\delta-(t+1)}(w)$.
- Asi que para completar la prueba sólo necesitamos ver que $\delta-(t+1)\leq t+1$ lo cual ocurre sii $\delta\leq 2(t+1)$.(*)
- Si δ es impar, digamos $\delta=2k+1$, entonces $t=\lfloor\frac{\delta-1}{2}\rfloor=k$.
- En este caso tenemos $\delta=2k+1=2t+1<2t+2$ y (*) se cumple.
- Si δ es par, digamos $\delta=2k$, entonces $t=\lfloor\frac{\delta-1}{2}\rfloor=\lfloor\frac{2k-1}{2}\rfloor=k-1$.

continuación prueba segunda parte del punto 2)

- Ahora bien, v y w diferian en δ bits, y u difiere de v en $t+1$ de esos δ bits y coincide en los otros.
- Por lo tanto, en esos $t+1$ bits, u coincide con w. Sólo difiere en los otros $\delta-(t+1)$ bits.
- Concluimos que $d_H(u,w)=\delta-(t+1)$ y $u\in D_{\delta-(t+1)}(w)$.
- Asi que para completar la prueba sólo necesitamos ver que $\delta-(t+1)\leq t+1$ lo cual ocurre sii $\delta\leq 2(t+1)$.(*)
- Si δ es impar, digamos $\delta=2k+1$, entonces $t=\lfloor\frac{\delta-1}{2}\rfloor=k$.
- En este caso tenemos $\delta=2k+1=2t+1<2t+2$ y (*) se cumple.
- Si δ es par, digamos $\delta=2k$, entonces $t=\lfloor\frac{\delta-1}{2}\rfloor=\lfloor\frac{2k-1}{2}\rfloor=k-1$.
- Por lo tanto $\delta=2k=2(k-1+1)=2(t+1)$ y (*) tambien se cumple. Fin.

Ejemplos

- Veamos que nos dice el teorema sobre los códigos C1-C3 que vimos antes.

Ejemplos

- Veamos que nos dice el teorema sobre los códigos C1-C3 que vimos antes.
- Vimos que $\delta(C1) = 1$. Por lo tanto $\delta - 1 = 0$, asi que C1 no detecta ni corrige errores.

Ejemplos

- Veamos que nos dice el teorema sobre los códigos C1-C3 que vimos antes.
- Vimos que $\delta(C1) = 1$. Por lo tanto $\delta - 1 = 0$, asi que C1 no detecta ni corrige errores.
- Vimos que $\delta(C2) = 2$, asi que C2 detecta 1 error (pues $\delta - 1 = 1$).

Ejemplos

- Veamos que nos dice el teorema sobre los códigos C1-C3 que vimos antes.
- Vimos que $\delta(C1) = 1$. Por lo tanto $\delta - 1 = 0$, asi que C1 no detecta ni corrige errores.
- Vimos que $\delta(C2) = 2$, asi que C2 detecta 1 error (pues $\delta - 1 = 1$).
- Pero $\lfloor \frac{\delta-1}{2} \rfloor = \lfloor \frac{1}{2} \rfloor = 0$ asi que no corrige errores.

Ejemplos

- Veamos que nos dice el teorema sobre los códigos C1-C3 que vimos antes.
- Vimos que $\delta(C1) = 1$. Por lo tanto $\delta - 1 = 0$, asi que C1 no detecta ni corrige errores.
- Vimos que $\delta(C2) = 2$, asi que C2 detecta 1 error (pues $\delta - 1 = 1$).
- Pero $\lfloor \frac{\delta-1}{2} \rfloor = \lfloor \frac{1}{2} \rfloor = 0$ asi que no corrige errores.
- $\delta(C3) = 4$ por lo tanto detecta 3 errores y corrige $\lfloor \frac{3}{2} \rfloor = 1$ error.

Mas ejemplos

- $C4 = \{000000, 111111, 111000, 000111\}$.

	000000	111111	111000	000111
000000	0	6	3	3
111111	6	0	3	3
111000	3	3	0	6
000111	3	3	6	0

$$\delta(C4) = 3$$

Mas ejemplos

- $C4 = \{000000, 111111, 111000, 000111\}$.

	000000	111111	111000	000111
000000	0	6	3	3
111111	6	0	3	3
111000	3	3	0	6
000111	3	3	6	0

$$\delta(C4) = 3$$

- Por lo tanto $C4$ detecta $\delta - 1 =$dos errores y corrige $\lfloor \frac{\delta-1}{2} \rfloor = \lfloor \frac{2}{2} \rfloor = 1$ error.

C4 vs C3

- ¿Hay alguna ventaja de C4 sobre C3?

C4 vs C3

- ¿Hay alguna ventaja de C4 sobre C3?
- Ambos usan 6 bits para transmitir 2 bits de información, asi que en este respecto no hay ventajas.

C4 vs C3

- ¿Hay alguna ventaja de C4 sobre C3?
- Ambos usan 6 bits para transmitir 2 bits de información, asi que en este respecto no hay ventajas.
- Ambos corrijen un error, pero no 2, tampoco hay diferencias aca.

C4 vs C3

- ¿Hay alguna ventaja de C4 sobre C3?
- Ambos usan 6 bits para transmitir 2 bits de información, asi que en este respecto no hay ventajas.
- Ambos corrijen un error, pero no 2, tampoco hay diferencias aca.
- C4 puede detectar dos errores, pero C3 puede detectar tres errores.

C4 vs C3

- ¿Hay alguna ventaja de C4 sobre C3?
- Ambos usan 6 bits para transmitir 2 bits de información, asi que en este respecto no hay ventajas.
- Ambos corrijen un error, pero no 2, tampoco hay diferencias aca.
- C4 puede detectar dos errores, pero C3 puede detectar tres errores.
- Asi que pareceria que C3 es “mejor” código.

C4 vs C3

- ¿Hay alguna ventaja de C4 sobre C3?
- Ambos usan 6 bits para transmitir 2 bits de información, asi que en este respecto no hay ventajas.
- Ambos corrijen un error, pero no 2, tampoco hay diferencias aca.
- C4 puede detectar dos errores, pero C3 puede detectar tres errores.
- Asi que pareceria que C3 es “mejor” código.
- Bueno, si en este sentido, y no en otro sentido.

C4 vs C3

- En general se está mas interesado en la capacidad de corrección que en la de detección asi que la ventaja de C3 sobre C4 no es tanta.

C4 vs C3

- En general se está mas interesado en la capacidad de corrección que en la de detección asi que la ventaja de C3 sobre C4 no es tanta.
- Y C4 tiene una ventaja sobre C3 que es clave.

C4 vs C3

- En general se está mas interesado en la capacidad de corrección que en la de detección asi que la ventaja de C3 sobre C4 no es tanta.
- Y C4 tiene una ventaja sobre C3 que es clave.
- En realidad esta ventaja no es tanta en un código con tan pocas palabras, pero en códigos con mas palabras, veremos que ciertos códigos que tienen una cierta propiedad que C4 tiene, poseen una ventaja respecto de códigos que no tienen esa propiedad, como C3.

C4 vs C3

- Y es que con códigos, no sólo queremos códigos que detecten y corrijan la mayor cantidad posible de errores, sino ademas, queremos un algoritmo EFICIENTE que permita corregir los errores.

C4 vs C3

- Y es que con códigos, no sólo queremos códigos que detecten y corrijan la mayor cantidad posible de errores, sino ademas, queremos un algoritmo EFICIENTE que permita corregir los errores.
- Con C3, dada una palabra recibida, hay que chequearla contra todas las posibles palabras enviadas para ver cual es la mas probable.

C4 vs C3

- Y es que con códigos, no sólo queremos códigos que detecten y corrijan la mayor cantidad posible de errores, sino ademas, queremos un algoritmo EFICIENTE que permita corregir los errores.
- Con C3, dada una palabra recibida, hay que chequearla contra todas las posibles palabras enviadas para ver cual es la mas probable.
- Como son sólo 4 palabras, esto no es problema, pero si el código tuviese 20 bits de información (mas de un millón de palabras) entonces por cada palabra recibida deberiamos contrastar contra un millón de palabras.

C4 vs C3

- Y es que con códigos, no sólo queremos códigos que detecten y corrijan la mayor cantidad posible de errores, sino ademas, queremos un algoritmo EFICIENTE que permita corregir los errores.
- Con C3, dada una palabra recibida, hay que chequearla contra todas las posibles palabras enviadas para ver cual es la mas probable.
- Como son sólo 4 palabras, esto no es problema, pero si el código tuviese 20 bits de información (mas de un millón de palabras) entonces por cada palabra recibida deberiamos contrastar contra un millón de palabras.
- En cambio C4 tiene un algoritmo mucho mas eficiente para hacer esto, que veremos luego.

C4 vs C3

- Y es que con códigos, no sólo queremos códigos que detecten y corrijan la mayor cantidad posible de errores, sino ademas, queremos un algoritmo EFICIENTE que permita corregir los errores.
- Con C3, dada una palabra recibida, hay que chequearla contra todas las posibles palabras enviadas para ver cual es la mas probable.
- Como son sólo 4 palabras, esto no es problema, pero si el código tuviese 20 bits de información (mas de un millón de palabras) entonces por cada palabra recibida deberiamos contrastar contra un millón de palabras.
- En cambio C4 tiene un algoritmo mucho mas eficiente para hacer esto, que veremos luego.
- Para sólo 4 palabras no vale mucho la pena, pero con mas palabras, si.

C5

- $C5 = \{00000, 11101, 11010, 00111\}$

	000000	11101	11010	00111
000000	0	4	3	3
11101	4	0	3	3
11010	3	3	0	4
00111	3	3	4	0

$$\delta(C5) = 3$$

C5

- $C5 = \{00000, 11101, 11010, 00111\}$

	000000	11101	11010	00111
000000	0	4	3	3
11101	4	0	3	3
11010	3	3	0	4
00111	3	3	4	0

$$\delta(C5) = 3$$

- Por lo tanto $C5$ tiene el mismo δ que $C4$ y detecta dos errores y corrige 1 error, pero es mejor que $C4$ pues necesita sólo 5 bits para transmitir 2 bits de información, y $C4$ necesita 6.

- $C5 = \{00000, 11101, 11010, 00111\}$

	000000	11101	11010	00111
000000	0	4	3	3
11101	4	0	3	3
11010	3	3	0	4
00111	3	3	4	0

$$\delta(C5) = 3$$

- Por lo tanto $C5$ tiene el mismo δ que $C4$ y detecta dos errores y corrige 1 error, pero es mejor que $C4$ pues necesita sólo 5 bits para transmitir 2 bits de información, y $C4$ necesita 6.
- Ademas veremos que $C5$ comparte esa propiedad especial de $C4$ que $C3$ no tiene.

Otra vista a los ejemplos

- En los ejemplos que dimos $C1$ era muy eficiente: mandaba dos bits de información usando 2 bits, pero no podia detectar no corregir errores.

Otra vista a los ejemplos

- En los ejemplos que dimos $C1$ era muy eficiente: mandaba dos bits de información usando 2 bits, pero no podia detectar no corregir errores.
- C2 podia detectar pero no corregir errores, y su costo era tener que mandar 3 bits para brindar dos bits de información, es decir, un bit extra por sobre la información.

Otra vista a los ejemplos

- En los ejemplos que dimos $C1$ era muy eficiente: mandaba dos bits de información usando 2 bits, pero no podia detectar no corregir errores.
- C2 podia detectar pero no corregir errores, y su costo era tener que mandar 3 bits para brindar dos bits de información, es decir, un bit extra por sobre la información.
- C3 y C4 podian corregir un error, pero necesitaban mandar 6 bits para hacerlo, es decir, 4 bits extras por sobre los bits de información.

Otra vista a los ejemplos

- En los ejemplos que dimos $C1$ era muy eficiente: mandaba dos bits de información usando 2 bits, pero no podia detectar no corregir errores.
- C2 podia detectar pero no corregir errores, y su costo era tener que mandar 3 bits para brindar dos bits de información, es decir, un bit extra por sobre la información.
- C3 y C4 podian corregir un error, pero necesitaban mandar 6 bits para hacerlo, es decir, 4 bits extras por sobre los bits de información.
- C5 era mas eficiente, podia tambien corregir un error con sólo 5 bits.

Otra vista a los ejemplos

- En los ejemplos que dimos $C1$ era muy eficiente: mandaba dos bits de información usando 2 bits, pero no podia detectar no corregir errores.
- C2 podia detectar pero no corregir errores, y su costo era tener que mandar 3 bits para brindar dos bits de información, es decir, un bit extra por sobre la información.
- C3 y C4 podian corregir un error, pero necesitaban mandar 6 bits para hacerlo, es decir, 4 bits extras por sobre los bits de información.
- C5 era mas eficiente, podia tambien corregir un error con sólo 5 bits.
- ¿Podriamos hacerlo con menos, usando sólo 4 bits o 3 bits?

Restricciones

- Este es uno de los problemas de teoría de códigos: diseñar códigos que corrijan errores mandando la menor cantidad posible de bits “extras” por sobre los bits de “información”.

Restricciones

- Este es uno de los problemas de teoría de códigos: diseñar códigos que corrijan errores mandando la menor cantidad posible de bits “extras” por sobre los bits de “información”.
- Lamentablemente, veremos que no podemos tener todo lo que queremos.

Restricciones

- Este es uno de los problemas de teoría de códigos: diseñar códigos que corrijan errores mandando la menor cantidad posible de bits "extras" por sobre los bits de "información".
- Lamentablemente, veremos que no podemos tener todo lo que queremos.
- Mientras mas errores querramos corregir, mas largas van a tener que ser las palabras del código.

Restricciones

- Este es uno de los problemas de teoría de códigos: diseñar códigos que corrijan errores mandando la menor cantidad posible de bits “extras” por sobre los bits de “información”.
- Lamentablemente, veremos que no podemos tener todo lo que queremos.
- Mientras mas errores querramos corregir, mas largas van a tener que ser las palabras del código.
- Hay varias cotas de este tipo, veamos la primera y mas famosa:

Teorema de la Cota De Hamming

Teorema (Cota de Hamming)

Sea C un código de longitud n, $\delta = \delta(C)$ y $t = \lfloor \frac{\delta - 1}{2} \rfloor$. Entonces:

$$\#C \leq \frac{2^n}{1 + n + \cdots + \binom{n}{t}}$$

Teorema de la Cota De Hamming

Teorema (Cota de Hamming)

Sea C un código de longitud n, $\delta = \delta(C)$ y $t = \lfloor \frac{\delta - 1}{2} \rfloor$. Entonces:

$$\#C \leq \frac{2^n}{1 + n + \cdots + \binom{n}{t}}$$

En el Biggs este teorema esta enunciado sólo para códigos “lineales" (los cuales veremos en la proxima “clase"), pero es un teorema válido para cualquier tipo de códigos.

Prueba de la Cota De Hamming

- Sea $A = \cup_{v \in C} D_t(v)$.

Prueba de la Cota De Hamming

- Sea $A = \cup_{v \in C} D_t(v)$.
- Como $t = \lfloor \frac{\delta - 1}{2} \rfloor$, entonces C corrige t errores, lo cual implica que $D_t(v) \cap D_t(w) = \emptyset$.

Prueba de la Cota De Hamming

- Sea $A = \cup_{v \in C} D_t(v)$.
- Como $t = \lfloor \frac{\delta - 1}{2} \rfloor$, entonces C corrige t errores, lo cual implica que $D_t(v) \cap D_t(w) = \emptyset$.
- Por lo tanto la union que forma a A es una union disjunta, lo cual nos permite calcular la cardinalidad de A como:

Prueba de la Cota De Hamming

- Sea $A = \cup_{v \in C} D_t(v)$.
- Como $t = \lfloor \frac{\delta - 1}{2} \rfloor$, entonces C corrige t errores, lo cual implica que $D_t(v) \cap D_t(w) = \emptyset$.
- Por lo tanto la union que forma a A es una union disjunta, lo cual nos permite calcular la cardinalidad de A como:
- $\#A = \sum_{v \in C} \#D_t(v)$.

Prueba de la Cota De Hamming

- Sea $A = \cup_{v \in C} D_t(v)$.
- Como $t = \lfloor \frac{\delta - 1}{2} \rfloor$, entonces C corrige t errores, lo cual implica que $D_t(v) \cap D_t(w) = \emptyset$.
- Por lo tanto la union que forma a A es una union disjunta, lo cual nos permite calcular la cardinalidad de A como:
- $\#A = \sum_{v \in C} \#D_t(v)$.
- Para calcular la cardinalidad de los D_ts, definamos los siguientes conjuntos:

Prueba de la Cota De Hamming

- Sea $A = \cup_{v \in C} D_t(v)$.
- Como $t = \lfloor \frac{\delta - 1}{2} \rfloor$, entonces C corrige t errores, lo cual implica que $D_t(v) \cap D_t(w) = \emptyset$.
- Por lo tanto la union que forma a A es una union disjunta, lo cual nos permite calcular la cardinalidad de A como:
- $\# A = \sum_{v \in C} \# D_t(v)$.
- Para calcular la cardinalidad de los D_ts, definamos los siguientes conjuntos:
- $S_r(v) = \{w \in \{0,1\}^n : d_H(v,w) = r\}$.

Prueba de la Cota De Hamming

- Sea $A = \cup_{v \in C} D_t(v)$.
- Como $t = \lfloor \frac{\delta - 1}{2} \rfloor$, entonces C corrige t errores, lo cual implica que $D_t(v) \cap D_t(w) = \emptyset$.
- Por lo tanto la union que forma a A es una union disjunta, lo cual nos permite calcular la cardinalidad de A como:
- $\#A = \sum_{v \in C} \#D_t(v)$.
- Para calcular la cardinalidad de los D_ts, definamos los siguientes conjuntos:
- $S_r(v) = \{w \in \{0,1\}^n : d_H(v,w) = r\}$.
- Recordemos que $D_t(v) = \{w \in \{0,1\}^n : d_H(v,w) \leq t\}$, por lo tanto $D_t(v) = \cup_{r=0}^{t} S_r(v)$.

Prueba de la Cota De Hamming

- Sea $A = \cup_{v \in C} D_t(v)$.
- Como $t = \lfloor \frac{\delta - 1}{2} \rfloor$, entonces C corrige t errores, lo cual implica que $D_t(v) \cap D_t(w) = \emptyset$.
- Por lo tanto la union que forma a A es una union disjunta, lo cual nos permite calcular la cardinalidad de A como:
- $\#A = \sum_{v \in C} \#D_t(v)$.
- Para calcular la cardinalidad de los D_ts, definamos los siguientes conjuntos:
- $S_r(v) = \{w \in \{0,1\}^n : d_H(v,w) = r\}$.
- Recordemos que $D_t(v) = \{w \in \{0,1\}^n : d_H(v,w) \leq t\}$, por lo tanto $D_t(v) = \cup_{r=0}^{t} S_r(v)$.
- Y como no podemos tener $d_H(v,w) = r$ y al mismo tiempo $d_H(v,w) = r'$ para $r' \neq r$, entonces esa union es disjunta.

Prueba de la Cota De Hamming

- Asi que $\#D_t(v) = \sum_{r=0}^{t} \#S_r(v)$.

Prueba de la Cota De Hamming

- Asi que $\#D_t(v) = \sum_{r=0}^{t} \#S_r(v)$.
- Necesitamos calcular $\#S_r(v)$.

Prueba de la Cota De Hamming

- Asi que $\#D_t(v) = \sum_{r=0}^{t} \#S_r(v)$.
- Necesitamos calcular $\#S_r(v)$.
- Un $w \in S_r(v)$ es una palabra que difiere de v en exactamente r de los n lugares posibles.

Prueba de la Cota De Hamming

- Asi que $\#D_t(v) = \sum_{r=0}^{t} \#S_r(v)$.
- Necesitamos calcular $\#S_r(v)$.
- Un $w \in S_r(v)$ es una palabra que difiere de v en exactamente r de los n lugares posibles.
- Por lo tanto, podemos asociar a cada $w \in S_r(v)$ un subconjunto de cardinalidad r de $\{1, 2, ..., n\}$: el conjunto de indices i tales que $w_i \neq v_i$.

Prueba de la Cota De Hamming

- Asi que $\#D_t(v) = \sum_{r=0}^{t} \#S_r(v)$.
- Necesitamos calcular $\#S_r(v)$.
- Un $w \in S_r(v)$ es una palabra que difiere de v en exactamente r de los n lugares posibles.
- Por lo tanto, podemos asociar a cada $w \in S_r(v)$ un subconjunto de cardinalidad r de $\{1, 2, ..., n\}$: el conjunto de indices i tales que $w_i \neq v_i$.
- Como estamos trabajando con códigos binarios, podemos hacer la vuelta: dado un subconjunto de cardinalidad r de $\{1, 2, ..., n\}$, crear un $w \in S_r(v)$, pues dado un conjunto de is, simplemente cambiamos los bits de v en esos is.

- Formalmente, sea $\Xi_r = \{Z \subseteq \{1, 2, ..., n\} : \#Z = r\}$

Prueba de la Cota De Hamming

- Formalmente, sea $\Xi_r = \{Z \subseteq \{1, 2, ..., n\} : \#Z = r\}$
- Definamos $\Psi : S_r(v) \mapsto \Xi_r$ por medio de $\Psi(w) = \{i : w_i \neq v_i\}$.

Prueba de la Cota De Hamming

- Formalmente, sea $\Xi_r = \{Z \subseteq \{1, 2, ..., n\} : \#Z = r\}$
- Definamos $\Psi : S_r(v) \mapsto \Xi_r$ por medio de $\Psi(w) = \{i : w_i \neq v_i\}$.
- Y definimos $\Phi : \Xi_r \mapsto S_r(v)$ por medio de $\Phi(Z) =$ el único w tal que $w_i = v_i$ si $i \notin Z$ y $w_i = 1 \oplus v_i$ si $i \in Z$.

Prueba de la Cota De Hamming

- Formalmente, sea $\Xi_r = \{Z \subseteq \{1, 2, ..., n\} : \#Z = r\}$
- Definamos $\Psi : S_r(v) \mapsto \Xi_r$ por medio de $\Psi(w) = \{i : w_i \neq v_i\}$.
- Y definimos $\Phi : \Xi_r \mapsto S_r(v)$ por medio de $\Phi(Z) =$ el único w tal que $w_i = v_i$ si $i \notin Z$ y $w_i = 1 \oplus v_i$ si $i \in Z$.
- Entonces Ψ, Φ son inversas una de la otra.

Prueba de la Cota De Hamming

- Formalmente, sea $\Xi_r = \{Z \subseteq \{1, 2, ..., n\} : \#Z = r\}$
- Definamos $\Psi : S_r(v) \mapsto \Xi_r$ por medio de $\Psi(w) = \{i : w_i \neq v_i\}$.
- Y definimos $\Phi : \Xi_r \mapsto S_r(v)$ por medio de $\Phi(Z) =$ el único w tal que $w_i = v_i$ si $i \notin Z$ y $w_i = 1 \oplus v_i$ si $i \in Z$.
- Entonces Ψ, Φ son inversas una de la otra.
- Por lo tanto, $\#S_r(v) = \#\Xi_r$.

Prueba de la Cota De Hamming

- Formalmente, sea $\Xi_r = \{Z \subseteq \{1, 2, ..., n\} : \#Z = r\}$
- Definamos $\Psi : S_r(v) \mapsto \Xi_r$ por medio de $\Psi(w) = \{i : w_i \neq v_i\}$.
- Y definimos $\Phi : \Xi_r \mapsto S_r(v)$ por medio de $\Phi(Z) =$ el único w tal que $w_i = v_i$ si $i \notin Z$ y $w_i = 1 \oplus v_i$ si $i \in Z$.
- Entonces Ψ, Φ son inversas una de la otra.
- Por lo tanto, $\#S_r(v) = \#\Xi_r$.
- Pero vieron en Discreta I que la cardinalidad de Ξ_r es el número combinatorio $\binom{n}{r}$.

Prueba de la Cota De Hamming

- Por lo tanto, $\#S_r(v) = \binom{n}{r}$

Prueba de la Cota De Hamming

- Por lo tanto, $\#S_r(v) = \binom{n}{r}$
- Y $\#D_t(v) = \sum_{r=0}^{t} \binom{n}{r}$

Prueba de la Cota De Hamming

- Por lo tanto, $\#S_r(v) = \binom{n}{r}$
- Y $\#D_t(v) = \sum_{r=0}^{t} \binom{n}{r}$
- Y $\#A = \sum_{v \in C} \left(\sum_{r=0}^{t} \binom{n}{r} \right)$.

Prueba de la Cota De Hamming

- Por lo tanto, $\#S_r(v) = \binom{n}{r}$
- Y $\#D_t(v) = \sum_{r=0}^{t} \binom{n}{r}$
- Y $\#A = \sum_{v \in C} \left(\sum_{r=0}^{t} \binom{n}{r} \right)$.
- Observar que la suma interior no depende de v, asi que esto queda:

Prueba de la Cota De Hamming

- Por lo tanto, $\#S_r(v) = \binom{n}{r}$
- Y $\#D_t(v) = \sum_{r=0}^{t} \binom{n}{r}$
- Y $\#A = \sum_{v \in C} \left(\sum_{r=0}^{t} \binom{n}{r} \right)$.
- Observar que la suma interior no depende de v, asi que esto queda:
- $\#A = \left(\sum_{r=0}^{t} \binom{n}{r} \right) \times \#C$. Por lo tanto:

Prueba de la Cota De Hamming

- Por lo tanto, $\#S_r(v) = \binom{n}{r}$
- Y $\#D_t(v) = \sum_{r=0}^{t} \binom{n}{r}$
- Y $\#A = \sum_{v \in C} \left(\sum_{r=0}^{t} \binom{n}{r} \right)$.
- Observar que la suma interior no depende de v, asi que esto queda:
- $\#A = \left(\sum_{r=0}^{t} \binom{n}{r} \right) \times \#C$. Por lo tanto:

$$\#C = \frac{\#A}{\sum_{r=0}^{t} \binom{n}{r}}$$

Prueba de la Cota De Hamming

- Por lo tanto, $\#S_r(v) = \binom{n}{r}$
- Y $\#D_t(v) = \sum_{r=0}^{t} \binom{n}{r}$
- Y $\#A = \sum_{v \in C} \left(\sum_{r=0}^{t} \binom{n}{r} \right)$.
- Observar que la suma interior no depende de v, asi que esto queda:
- $\#A = \left(\sum_{r=0}^{t} \binom{n}{r} \right) \times \#C$. Por lo tanto:

$$\#C = \frac{\#A}{\sum_{r=0}^{t} \binom{n}{r}}$$

Solo resta observar que como A es un subconjunto de $\{0,1\}^n$

Prueba de la Cota De Hamming

- Por lo tanto, $\#S_r(v) = \binom{n}{r}$
- Y $\#D_t(v) = \sum_{r=0}^{t} \binom{n}{r}$
- Y $\#A = \sum_{v \in C} \left(\sum_{r=0}^{t} \binom{n}{r} \right)$.
- Observar que la suma interior no depende de v, asi que esto queda:
- $\#A = \left(\sum_{r=0}^{t} \binom{n}{r} \right) \times \#C$. Por lo tanto:

$$\#C = \frac{\#A}{\sum_{r=0}^{t} \binom{n}{r}}$$

Solo resta observar que como A es un subconjunto de $\{0,1\}^n$, su cardinalidad es menor o igual que 2^n

- Por lo tanto, $\#S_r(v) = \binom{n}{r}$
- Y $\#D_t(v) = \sum_{r=0}^{t} \binom{n}{r}$
- Y $\#A = \sum_{v \in C} \left(\sum_{r=0}^{t} \binom{n}{r} \right)$.
- Observar que la suma interior no depende de v, asi que esto queda:
- $\#A = \left(\sum_{r=0}^{t} \binom{n}{r} \right) \times \#C$. Por lo tanto:

$$\#C = \frac{\#A}{\sum_{r=0}^{t} \binom{n}{r}} \leq \frac{2^n}{\sum_{r=0}^{t} \binom{n}{r}}$$

Solo resta observar que como A es un subconjunto de $\{0,1\}^n$, su cardinalidad es menor o igual que 2^n , lo cual finaliza la prueba.↑

Ejemplo de aplicación de la Cota De Hamming

- Antes nos preguntabamos si podriamos enviar dos bits de información con un código de longitud 4 que corrigiera 1 error.

Ejemplo de aplicación de la Cota De Hamming

- Antes nos preguntabamos si podriamos enviar dos bits de información con un código de longitud 4 que corrigiera 1 error.
- Es decir, necesitariamos $n = 4$, $\#C = 2^2 = 4$, $t = 1$.

Ejemplo de aplicación de la Cota De Hamming

- Antes nos preguntabamos si podriamos enviar dos bits de información con un código de longitud 4 que corrigiera 1 error.
- Es decir, necesitariamos $n = 4$, $\#C = 2^2 = 4$, $t = 1$.
- Si se pudiera, la Cota de Hamming nos diria que debe valer la siguiente desigualdad:

Ejemplo de aplicación de la Cota De Hamming

- Antes nos preguntabamos si podriamos enviar dos bits de información con un código de longitud 4 que corrigiera 1 error.
- Es decir, necesitariamos $n = 4$, $\#C = 2^2 = 4$, $t = 1$.
- Si se pudiera, la Cota de Hamming nos diria que debe valer la siguiente desigualdad:

$$4 \leq \frac{2^4}{\sum_{r=0}^{1} \binom{4}{r}} = \frac{16}{1+4} = \frac{16}{5} = 3,2$$

Ejemplo de aplicación de la Cota De Hamming

- Antes nos preguntabamos si podriamos enviar dos bits de información con un código de longitud 4 que corrigiera 1 error.
- Es decir, necesitariamos $n = 4$, $\#C = 2^2 = 4$, $t = 1$.
- Si se pudiera, la Cota de Hamming nos diria que debe valer la siguiente desigualdad:

$$4 \leq \frac{2^4}{\sum_{r=0}^{1} \binom{4}{r}} = \frac{16}{1+4} = \frac{16}{5} = 3,2$$

Absurdo. Asi que C5 o algo similar es lo mejor a lo que podemos aspirar para el caso de $\#C = 4$ si queremos corregir un error.

Ejemplo de aplicación de la Cota De Hamming

- Observar que si con ciertos parametros $n, \#C, t$ la cota de Hamming no se satisface, entonces sabemos que no existe un código con esos parametros, pero si la cota de Hamming SI se satisface eso no implica que EXISTA un código con esos parametros.

Ejemplo de aplicación de la Cota De Hamming

- Observar que si con ciertos parametros $n, \#C, t$ la cota de Hamming no se satisface, entonces sabemos que no existe un código con esos parametros, pero si la cota de Hamming SI se satisface eso no implica que EXISTA un código con esos parametros.
- La cota de Hamming sirve para probar imposibilidad, no existencia.

Ejemplo de aplicación de la Cota De Hamming

- Observar que si con ciertos parametros $n, \#C, t$ la cota de Hamming no se satisface, entonces sabemos que no existe un código con esos parametros, pero si la cota de Hamming SI se satisface eso no implica que EXISTA un código con esos parametros.
- La cota de Hamming sirve para probar imposibilidad, no existencia.
- Es decir, si les preguntamos ¿Existe código tal que....?, la cota de Hamming sólo les va a servir si la respuesta es NO.

Ejemplo de aplicación de la Cota De Hamming

- Observar que si con ciertos parametros $n, \#C, t$ la cota de Hamming no se satisface, entonces sabemos que no existe un código con esos parametros, pero si la cota de Hamming SI se satisface eso no implica que EXISTA un código con esos parametros.
- La cota de Hamming sirve para probar imposibilidad, no existencia.
- Es decir, si les preguntamos ¿Existe código tal que....?, la cota de Hamming sólo les va a servir si la respuesta es NO.
- Si la respuesta es SI, van a tener que DAR un código que satisfaga las condiciones que les pedimos, no basta con decir “bueno, la cota de Hamming no impide que exista, asi que debe existir"

Códigos perfectos

- Un código es perfecto si el $\leq$ en la cota de Hamming es un $=$.

Códigos perfectos

- Un código es perfecto si el $\leq$ en la cota de Hamming es un $=$.
- Es decir, C es perfecto si es de longitud n y si, con $t = \lfloor \frac{\delta(C)-1}{2} \rfloor$ se cumple:

Códigos perfectos

- Un código es perfecto si el $\leq$ en la cota de Hamming es un $=$.
- Es decir, C es perfecto si es de longitud n y si, con $t = \lfloor \frac{\delta(C)-1}{2} \rfloor$ se cumple:

$$\#C = \frac{2^n}{\sum_{r=0}^{t} \binom{n}{r}}$$

Códigos perfectos

- Un código es perfecto si el $\leq$ en la cota de Hamming es un $=$.
- Es decir, C es perfecto si es de longitud n y si, con $t = \lfloor \frac{\delta(C)-1}{2} \rfloor$ se cumple:

$$\#C = \frac{2^n}{\sum_{r=0}^{t} \binom{n}{r}}$$

Se llaman asi porque son lo mejor que puede haber, dadas las limitaciones del universo

Códigos perfectos

- Un código es perfecto si el $\leq$ en la cota de Hamming es un $=$.
- Es decir, C es perfecto si es de longitud n y si, con $t = \lfloor \frac{\delta(C)-1}{2} \rfloor$ se cumple:

$$\#C = \frac{2^n}{\sum_{r=0}^{t} \binom{n}{r}}$$

Se llaman asi porque son lo mejor que puede haber, dadas las limitaciones del universo
Son extremadamente raros, aunque hay una cantidad infinita de ellos (haciendo variar el n, básicamente, veremos una familia luego)