Universidad de Granada

decsai.ugr.es

Teoría de la Información y la Codificación

Grado en Ingeniería Informática

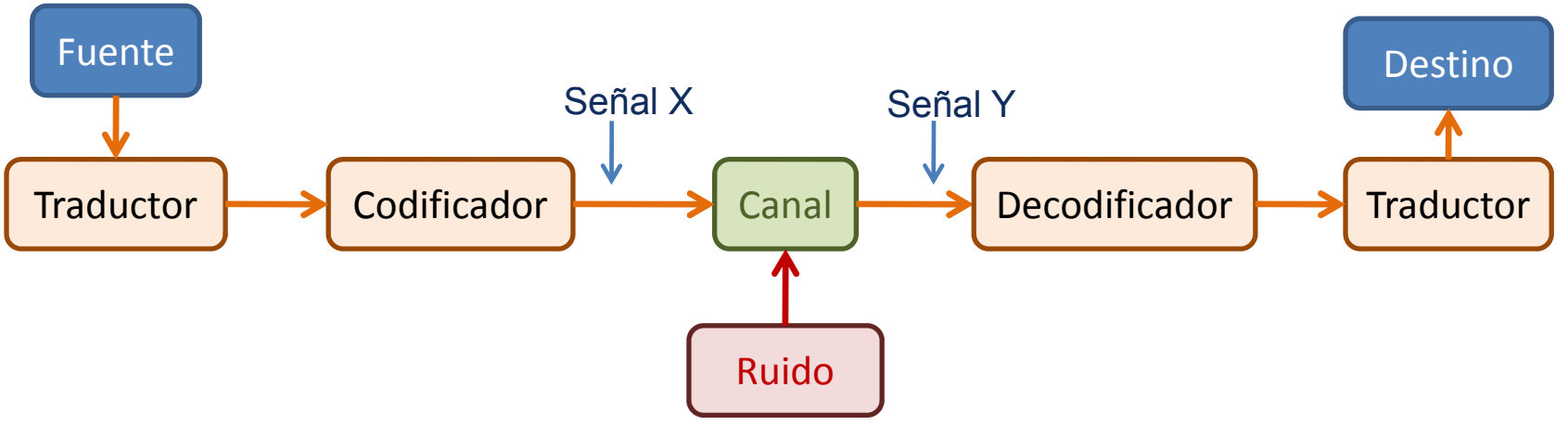
Tema 3.- Información en canales sin ruido.

DECSAI

Departamento de Ciencias de la Computación e Inteligencia Artificial

1. El teorema fundamental para canales sin ruido
2. Tipos de códigos
3. Códigos instantáneos
4. El método de Shannon-Fano para codificación
5. El método de Huffman para codificación
6. Fundamentos de la compresión de dato

- **Base para la codificación en un canal:**



- En este esquema, el **emisor** se convierte en traductor de símbolos del alfabeto de la fuente a símbolos o señales que se pueden transmitir por el canal.
- Tras traducir los símbolos de la fuente, los codifica.
- El **receptor** realiza las operaciones inversas:
  - Decodifica el mensaje del canal.
  - Lo traduce a símbolos legibles por el destino.

- **Teorema fundamental de Shannon para un canal sin ruido**

- **Sea una fuente con entropía H (bits por símbolo) y un canal con capacidad C (bits por segundo). Entonces, es posible codificar la salida de la fuente de tal modo que se transmita a una velocidad de C/H−ε símbolos por segundo sobre el canal, donde ε es tan pequeño como se quiera. No es posible transmitir a un régimen promedio mayor que C/H.**

- El Teorema fundamental de **Shannon** para codificación en canales sin ruido es de extrema importancia hoy día. Indica el límite máximo de bits que se pueden codificar por un canal con una capacidad.
- Con este teorema, **Shannon** prueba la existencia de un **límite a la eficiencia de la codificación de una fuente**.

- **Tipos de códigos:**
  - Según el número de símbolos del alfabeto a utilizar:
    - *Binario, Ternario, etc.*
  - Según la longitud que tenga una palabra transmitida:
    - **Uniformes**: Todas las palabras tienen la misma longitud
      - *Por ejemplo: ASCII*
    - **No uniformes:** Hay palabras que tienen longitud diferente a otras
      - *Por ejemplo: Morse*

- **Códigos de traducción única:**
  - Se dice que un código es de traducción única si, para cualquier sucesión única de mensajes a transmitir, corresponde una única sucesión de símbolos transmitidos
  - **Ejemplo: El código siguiente es de traducción única.**

$m_1 \rightarrow 0$
$m_2 \rightarrow 01$

  - **Para transmitir un mensaje cualquiera (por ejemplo $m_1\ m_1\ m_2\ m_1$), su secuencia codificada es única: 00010.**
  - **Ejemplo: El código siguiente NO es de traducción única. Para el mensaje codificado 00010, no sabríamos decodificarlo unívocamente:**

$m_1 \rightarrow 0$
$m_2 \rightarrow 01$
$m_3 \rightarrow 1$

- **Códigos instantáneos:**
  - Se dice que un código es instantáneo si ninguna palabra codificada coincide con el comienzo de otra.
  - Los códigos instantáneos **son un subconjunto de los códigos de traducción única.**
  - Existe un algoritmo capaz de generar automáticamente códigos instantáneos.
  - **Tendremos un conjunto de mensajes a enviar (alfabeto de la fuente).**
  - **Tendremos un conjunto de símbolos transmisibles por el canal (alfabeto del código).**

– Códigos instantáneos:

– Un ejemplo simple: Tenemos los símbolos “Hola”, “Adiós”, “Encender”, Apagar”, a transmitir entre una fuente y un destino. Proponer un código binario instantáneo que lo codifique:

– **Basta con asignar:**
  – **“Hola” : 0**
  – **“Adiós” : 10**
  – **“Encender” : 110**
  – **“Apagar” : 1110**

– El código propuesto es instantáneo, pues ninguna palabra es comienzo de otra.

– Algoritmo de generación de códigos instantáneos:

– Sea **M= {m_1, m_2, …, m_N}** un conjunto de mensajes a enviar (alfabeto de la fuente). Sea **A= {a_1, a_2, …, a_D}** el conjunto de símbolos transmisibles por el canal (alfabeto del código).

1. **Se divide M en D subconjuntos M_1= {m_1, m_2, …, m_k}, M_2= {m_{k+1}, m_{k+2}, …, m_{2*k}}, …, M_D= {m_r, m_{r+1}, …, m_N} . A cada M_i se le asigna el símbolo a_i del alfabeto del código.**

2. **Repetir la operación para cada M_i , generando M_{i1} , M_{i2} , …, M_{iD} y asignando a cada M_{ij} los códigos {a_i a_j},** hasta que cada subconjunto generado tenga un único elemento.

El código obtenido con este procedimiento es instantáneo, pues ninguna palabra (codificación) se corresponde con el comienzo de otra.

- **Ejemplo: Codificar los 6 primeros símbolos del alfabeto utilizando un código binario instantáneo.**
  - M= {*A, B, C, D, E, F*}; A= {0, 1}
  - Comenzamos dividiendo M en:
    M1= {*A, B, C*}, M2= {D, E, F}; entonces se asigna M1→0; M2→1
  - Ahora dividimos M1 en:
    M11= {*A, B*}, M12= {C} ; entonces se asigna M11→00; M12→01
  - Ahora dividimos M1 en:
    M111= {*A*}, M112= {B} ; entonces se asigna M111→000; M112→001

Ya hemos codificado la mitad M1:
A→000; B→001; C→01

- **Ejemplo: Codificar los 6 primeros símbolos del alfabeto utilizando un código binario instantáneo.**

  - Ahora dividimos M2 en:
    M21= {*D, E*}, M22= {F} ; entonces se asigna M21→10; M22→11

  - Ahora dividimos M21 en:
    M211= {*D*}, M212= {E} ; entonces se asigna M211→100; M212→101

  Ya hemos codificado la otra mitad M2. El código completo queda:

  A→000; B→001; C→01
  D→100; E→101; F→11

  Comprobamos que es instantáneo, dado que ninguna palabra es comienzo de otra.

– Árbol de un código.

A través del proceso anterior, podríamos pensar que un código podría representarse en un árbol.

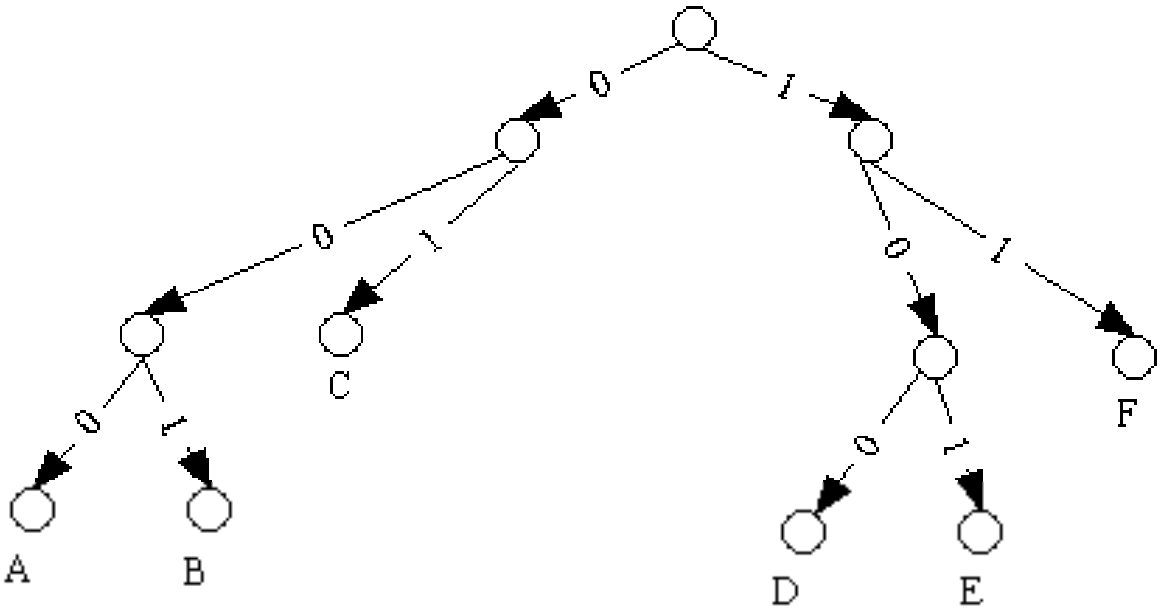
Este árbol se representaría de la siguiente forma:

- Un nodo raíz, identificado con la cadena vacía.
- Por cada nodo, tantos arcos como símbolos tenga el alfabeto del código.
- En los nodos terminales, un símbolo del alfabeto de la fuente asociado a la codificación.

El código de cada símbolo del alfabeto de la fuente se representaría por la secuencia de símbolos del alfabeto del código, desde la raíz hasta el nodo hoja pertinente.

– **Árbol de un código. Ejemplo:**

A→000; B→001; C→01
D→100; E→101; F→11



Cualquier palabra que no pueda representarse con este árbol (camino desde la raíz a alguna hoja), y fuese distinta de la palabra vacía, no podría identificarse con ningún elemento válido del código generado.

- **Códigos completos.**

En todo código unívocamente decodificable, la suma de las probabilidades de todos los símbolos es siempre menor o igual que uno.

Si la suma de las probabilidades de los símbolos existentes en el árbol es exactamente 1, se dice que el código es **completo.**

– Desigualdad de Kraft.

Sea **M= {m_1, m_2, …, m_N}** un conjunto de mensajes a enviar (alfabeto de la fuente). Sea **A= {a_1, a_2, …, a_D}** los símbolos a utilizar por un codificador.

Sean **n_1, n_2, …n_N** las longitudes de los mensajes codificados **m_1, m_2, …, m_N** .

Entonces la condición necesaria y suficiente para que tal código exista es:

$$\sum_{i=1}^{N} D^{-n_i} \leq 1$$

Si no se cumple la desigualdad de Kraft, es imposible que exista un código de decodificación instantánea con longitudes de palabra **n_1, n_2, …n_N** para los mensajes a codificar.

– Desigualdad de Kraft. Ejemplo:

Sea **M= {A, B, C, D, E, F}** un conjunto de mensajes a enviar (alfabeto de la fuente). Sea **A= {0, 1}** los símbolos a utilizar por un codificador.

Sean las longitudes de los mensajes codificados |c(A)=3|, |c(B)=3|, |c(C)=2|, |c(D)=3|, |c(E)=3|, |c(F)=2| .

Según esto, entonces verificaremos si puede existir dicho código:

$$Entonces: \sum_{i=1}^{N} D^{-n_i} \leq 1$$

$$2^{-3} + 2^{-3} + 2^{-2} + 2^{-3} + 2^{-3} + 2^{-2} \leq 1$$

$$1 \leq 1 \Rightarrow \text{Dicho código existe}$$

Si quisiésemos un código de las mismas características con |c(X)=2|, con X=A…F, ¿existiría dicho código?

$$Entonces: \sum_{i=1}^{N} D^{-n_i} \leq 1 \Rightarrow 6 \cdot 2^{-2} \leq 1;\ 1{,}5 > 1 \Rightarrow \text{Dicho código no existe}$$

– Igualdad de Kraft.

Sea **M= {m_1, m_2, …, m_N}** un conjunto de mensajes a enviar (alfabeto de la fuente). Sea **A= {a_1, a_2, …, a_D}** los símbolos a utilizar por un codificador.

Sean **n_1, n_2, …n_N** las longitudes de los mensajes codificados **m_1, m_2, …, m_N** , utilizando un **código completo.**

Entonces la condición necesaria y suficiente para que tal código completo exista es:

$$\sum_{i=1}^{N} D^{-n_i} = 1$$

Si no se cumple la igualdad de Kraft ni la desigualdad, el código no existe.

Si cumple la desigualdad de Kraft pero no la igualdad, el código existe pero no es completo.

Si cumple la desigualdad de Kraft pero no la igualdad, el código existe y es completo.

– Igualdad de Kraft. Ejemplo:

Sea **M= {A, B, C, D, E, F}** un conjunto de mensajes a enviar (alfabeto de la fuente). Sea **A= {0, 1}** los símbolos a utilizar por un codificador.

Sean las longitudes de los mensajes codificados |c(A)|= |c(B)|= |c(C)|= |c(D)|= |c(E)|= |c(F)|=3 .

Según esto, entonces verificaremos si puede existir dicho código y es completo:

$$Entonces: \sum_{i=1}^{N} D^{-n_i} = 1$$

$$2^{-3} + 2^{-3} + 2^{-2} + 2^{-3} + 2^{-3} + 2^{-2} = 1$$

$$1 = 1 \Rightarrow \text{Dicho código existe y es completo}$$

Ante códigos de decodificación única, la desigualdad de Kraft nos puede ayudar a saber si el código es óptimo o no→ Si no es completo, entonces !estamos desaprovechando símbolos en la codificación!

– Igualdad de Kraft. Ejemplo:

Sea **M= {A, B, C, D, E, F}** un conjunto de mensajes a enviar (alfabeto de la fuente). Sea **A= {0, 1}** los símbolos a utilizar por un codificador.

Sean las longitudes de los mensajes codificados |c(A)=3|, |c(B)=3|, |c(C)=2|, |c(D)=3|, |c(E)=3|, |c(F)=2| .

Ejemplo de tal código sería:

A→000; B→001; C→010
D→100; E→101; F→110

Es de decodificación única, porque ningún código es inicio de otro.

Sin embargo:

$$\sum_{i=1}^{N} D^{-n_i} = 2^{-3} + 2^{-3} + 2^{-3} + 2^{-3} + 2^{-3} + 2^{-3} = 0{,}75 < 1$$

El código no es completo. Se podrían utilizar menos bits para codificar los mensajes.

- **Consecuencias de la igualdad de Kraft:**

Sea **M= {m_1, m_2, …, m_N}** un conjunto de mensajes a enviar (alfabeto de la fuente). Sea **A= {a_1, a_2, …, a_D}** los símbolos a utilizar por un codificador.

Para que exista un código completo que codifique los mensajes en M, es necesario que exista un **entero positivo λ tal que:**

$$N = 1 + \lambda(D-1)$$

- **En nuestro ejemplo de antes:**

N= 6 mensajes (*A, B, C, D, E, F*)
D= 2 (codificación binaria).

Es fácil encontrar λ despejando, con valor λ =5.

– Códigos óptimos:

Nos referiremos a ***códigos óptimos*** según la eficiencia de los mismos; es decir, al mínimo número de símbolos del codificador necesarios para codificar cada mensaje de la fuente.

Para medir la optimalidad de un código instantáneo, deberemos hacerlo en términos del número promedio de símbolos de códigos emitidos en una transmisión (bits en codificación binaria, trits en codificación ternaria, etc.)**.**

Sea **M= {m_1, m_2, …, m_N}** un conjunto de mensajes a enviar (alfabeto de la fuente), **n_1, n_2, …n_N las longitudes de los códigos** de dichos mensajes, y **{p(m_1), p(m_2), …, p(m_N)}** las probabilidades de que la fuente emita dichos mensajes. Entonces, se define el promedio de símbolos codificados y transmitidos como:

$$\overline{n} = \sum_{i=1}^{n} p(m_i) \cdot n_i$$

(Es el número de bits que se espera transmitir por mensaje en una transmisión infinita)

– **Códigos óptimos:**

Un código óptimo cumplirá con la condición de que la información transmitida será igual a la capacidad del canal.

Según el Teorema de Shannon para codificación por ruido, no se puede transmitir a una tasa mayor que C/H.

Por tanto, si la tasa de eficiencia del código C/H=1 entonces se transmite tanta información como es posible por el canal. El código es óptimo.

– **Primer teorema fundamental de la codificación sin ruido:**

Sea **M= {m_1, m_2, …, m_N}** un conjunto de mensajes a enviar (alfabeto de la fuente). Sea **A= {a_1, a_2, …, a_D}** los símbolos a utilizar por un codificador.

La longitud promedia de cada palabra codificada en un código instantáneo siempre será mayor o igual a:

$$\overline{n} \geq \frac{H(M)}{\log_2 D}$$

Además, existe al menos un código instantáneo tal que

$$\overline{n} < \frac{H(M)}{\log_2 D} + 1$$

Este teorema **establece el mínimo número de símbolos que será necesario utilizar, en promedio, para codificar un símbolo de la fuente con un código instantáneo.**

– El método de Shannon-Fano para codificación sin ruido:

Es una propuesta que trata de dar un código óptimo a transmitir por un canal, codificando los mensajes de la fuente.

Es un método de codificación binaria, aunque extensible a codificación con mayor número de símbolos (n-arias).

Sea **M= {m_1, m_2, …, m_N}** un conjunto de mensajes a enviar (alfabeto de la fuente), junto con las probabilidades de que la fuente genere cada uno de los mensajes, **{p(m_1), p(m_2), …, p(m_N)}** . Sea **A= {a_1, a_2}** los símbolos a utilizar por un codificador.

El método trata de codificar los mensajes más probables en códigos de una menor longitud, mediante una técnica descendente.

– El método de Shannon-Fano para codificación sin ruido:

1. Ordenar los mensajes $\{m_i\}$ en orden decreciente de probabilidades $\{p_i\}$. Llamaremos M' al conjunto de mensajes ordenado.

2. Escoger el punto ***k*** *de* ***M'*** tal que divida ***M'*** en 2 partes equiprobables, denominadas $\boldsymbol{M'_1}$ y $\boldsymbol{M'_2}$.

3. Asignar $\mathbf{a_1}$ a $\boldsymbol{M'_1}$ y $\mathbf{a_2}$ a $\boldsymbol{M'_2}$.

4. Volver a realizar la operación desde el punto 2 sobre los subconjuntos generados $\boldsymbol{M'_1}$ y $\boldsymbol{M'_2}$, hasta que el número de símbolos en cada subconjunto sea 1.

Al finalizar, el método de **Shannon-Fano** crea un árbol de codificación para cada mensaje de la fuente.

- **Ejemplo: Supongamos una fuente X que emite 6 mensajes $\{x_1, x_2, x_3, x_4, x_5, x_6\}$, con probabilidades para cada símbolo $\{p(x_1)= 0,05, p(x_2)= 0,10, p(x_3)= 0,25, p(x_4)= 0,20, p(x_5)= 0,10, p(x_6)= 0,3\}$. Crear un código binario para esta fuente.**

- **Método de Shannon-Fano:**

1. Ordenación por probabilidades:

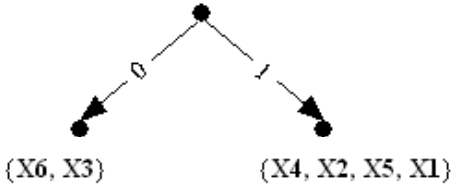
x_6	x_3	x_4	x_2	x_5	x_1
0,3	0,25	0,2	0,1	0,1	0,05

2. Dividir por k=2, tratando de hacer las partes equiprobables:

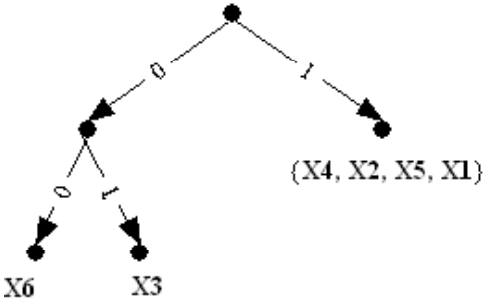
$M'_1 = \{x_6, x_3\}$; $M'_2 = \{x_4, x_2, x_5, x_1\}$

- **Ejemplo: Supongamos una fuente X que emite 6 mensajes $\{x_1, x_2, x_3, x_4, x_5, x_6\}$, con probabilidades para cada símbolo $\{p(x_1)= 0,05, p(x_2)= 0,10, p(x_3)= 0,25, p(x_4)= 0,20, p(x_5)= 0,10, p(x_6)= 0,3\}$.**

- **El árbol queda:**

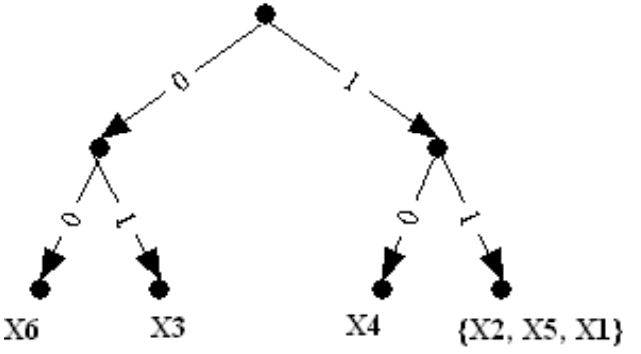


- Volvemos a hacer lo mismo con el conjunto $M'_1 = \{x_6, x_3\}$:



- **Ejemplo: Supongamos una fuente X que emite 6 mensajes {x_1, x_2, x_3, x_4, x_5, x_6}, con probabilidades para cada símbolo {p(x_1)= 0,05, p(x_2)= 0,10, p(x_3)= 0,25, p(x_4)= 0,20, p(x_5)= 0,10, p(x_6)= 0,3}.**

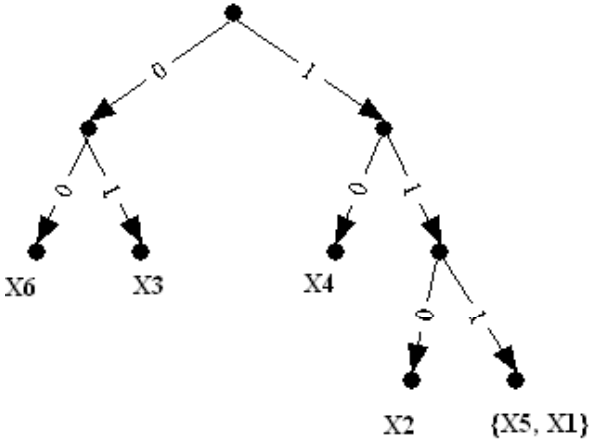
- **Y también** hacer lo mismo con el conjunto M'_2 ={x_4, x_2 , x_5 , x_1}. En este caso, seleccionamos k=1 porque divide al conjunto de forma equiprobable en M'_{21}={x_4} y M'_{22}={x_2, x_5, x_1}:



- Como M'_{21} tiene un único elemento, pasamos a dividir sólo M'_{22}, en M'_{221}={x_2} y M'_{222}={x_5, x_1}:

- **Ejemplo: Supongamos una fuente X que emite 6 mensajes $\{x_1, x_2, x_3, x_4, x_5, x_6\}$, con probabilidades para cada símbolo $\{p(x_1)= 0,05, p(x_2)= 0,10, p(x_3)= 0,25, p(x_4)= 0,20, p(x_5)= 0,10, p(x_6)= 0,3\}$.**

- **El árbol queda:**

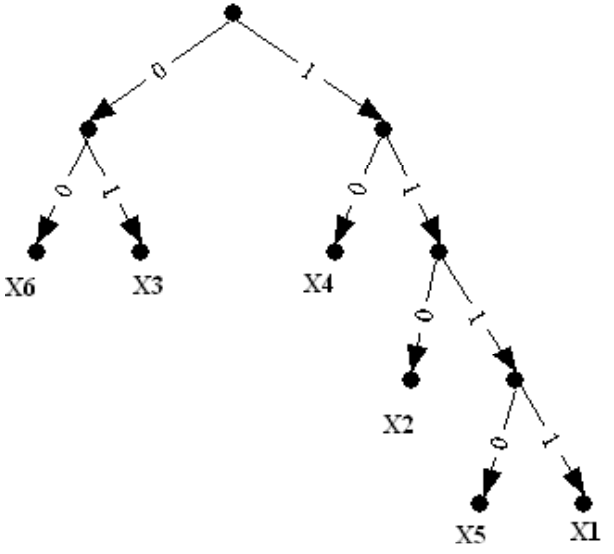


- Como M'_{221} tiene un único elemento, pasamos a dividir sólo M'_{222}, en $M'_{2221}=\{x_5\}$ y $M'_{2222}=\{x_1\}$, **que sólo tienen un símbolo.**

- **El algoritmo termina.**

- **Ejemplo: Supongamos una fuente X que emite 6 mensajes {x_1, x_2, x_3, x_4, x_5, x_6}, con probabilidades para cada símbolo {$p(x_1)= 0,05$, $p(x_2)= 0,10$, $p(x_3)= 0,25$, $p(x_4)= 0,20$, $p(x_5)= 0,10$, $p(x_6)= 0,3$}.**

- **El árbol final queda:**



- El código final queda: **x_6→00; x_3→01; x_4→10; x_2→110; x_5→1110; x_1→1111** .

- Podemos observar cómo, a mensajes más probables, se les asigna códigos más cortos.

- Ejemplo: ¿Es el código completo? Lo comprobaremos con la igualdad de Kraft

- N=6: Número de mensajes distintos
- $n_1=|x_1|=4$; $n_2=|x_2|=3$; $n_3=|x_3|=2$; $n_4=|x_4|=2$; $n_5=|x_5|=4$; $n_6=|x_6|=2$

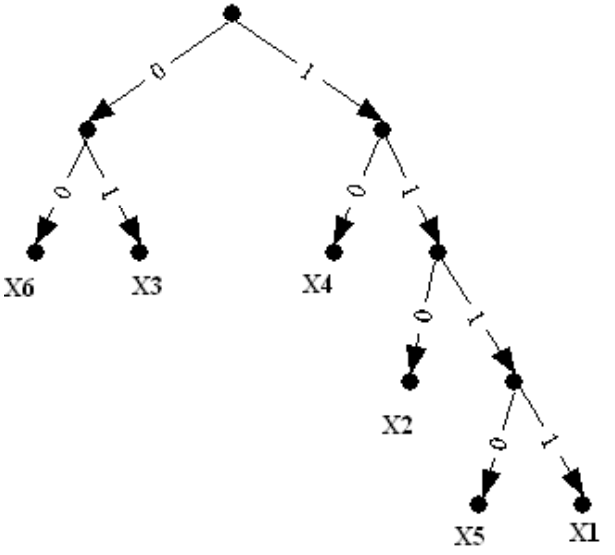
$$\sum_{i=1}^{n} D^{-n_i} = 2*2^{-4} + 1*2^{-3} + 3*2^{-2} = 1$$

Se cumple que el código es completo

– Decodificación con el método de Shannon-Fano

- Imaginemos que nos llega la siguiente cadena codificada: **0001110**.
- **Para decodificar una cadena de mensajes** recibida por el método de Shannon-Fano, es necesario que el receptor **mantenga en memoria el árbol de codificación.**
- **Así, basta con comenzar desde la raíz e ir recorriendo el árbol según los bits que se estén recibiendo.**
- **Cuando lleguemos a un nodo hoja, se habrá decodificado un mensaje.**
- **En este caso, volveremos a aplicar el procedimiento estableciendo el nodo actual de nuevo como la raíz.**

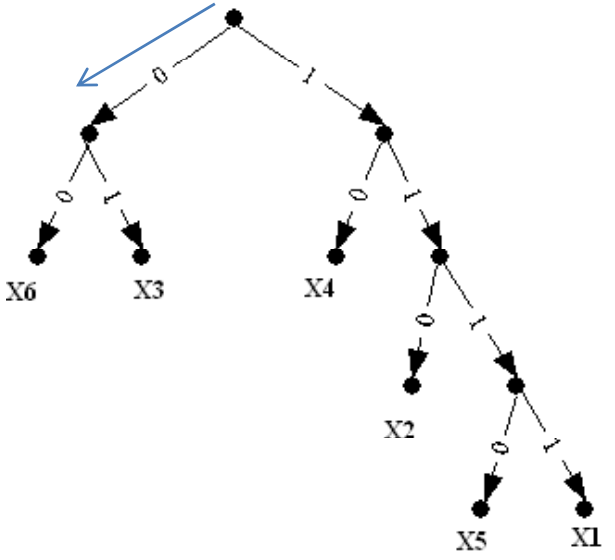
- **Partiremos del siguiente árbol: (inicializado en la raíz)**



- **Decodificación con el método de Shannon-Fano**

- Decodificación del primer símbolo: **0001110**.

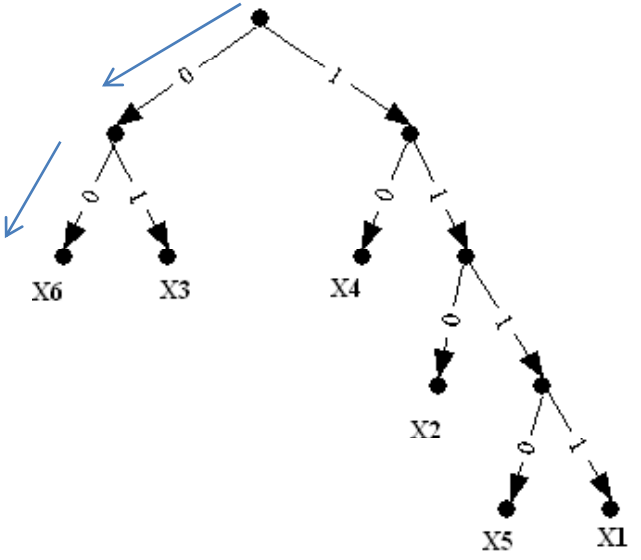
- **Nos movemos desde la raíz siguiendo el 0:**



- **Decodificación con el método de Shannon-Fano**

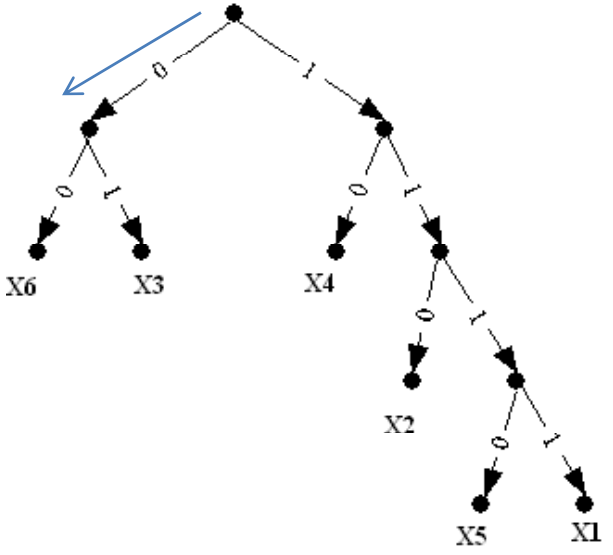
- Decodificación del segundo símbolo: **0001110**.

- **Nos movemos desde el nodo actual siguiendo el 0:**



- **Acabamos de decodificar el símbolo X6. Inicializamos el nodo actual a la raíz y continuamos.**

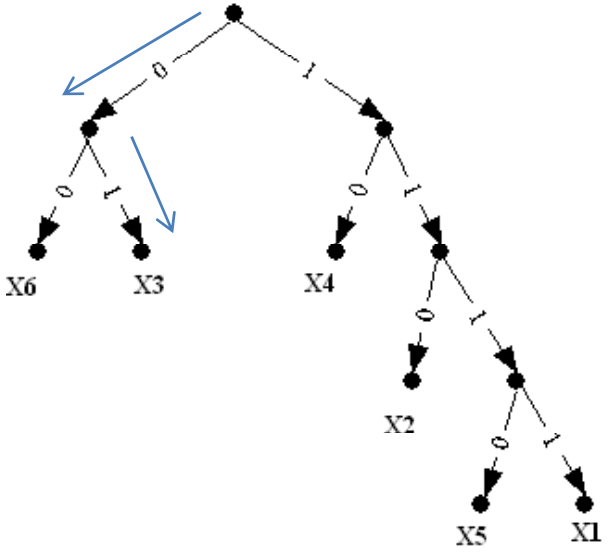
- **Decodificación con el método de Shannon-Fano**
- Decodificación del tercer símbolo: **0001110**.
- **Nos movemos desde el nodo actual siguiendo el 0:**



- **Decodificación con el método de Shannon-Fano**

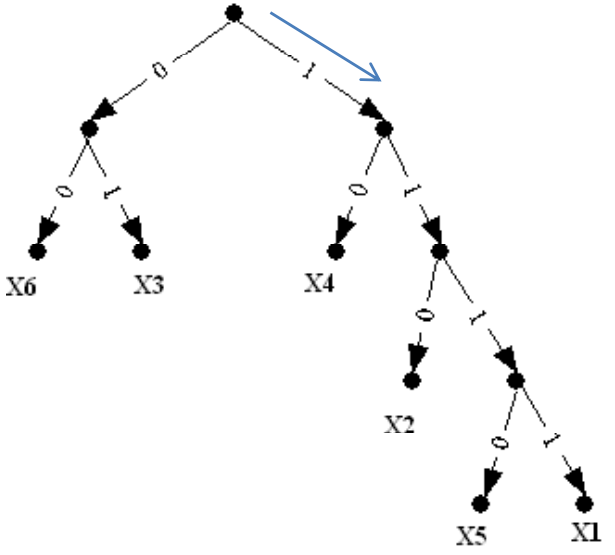
- Decodificación del cuarto símbolo: **0001110**.

- **Nos movemos desde el nodo actual siguiendo el 0:**

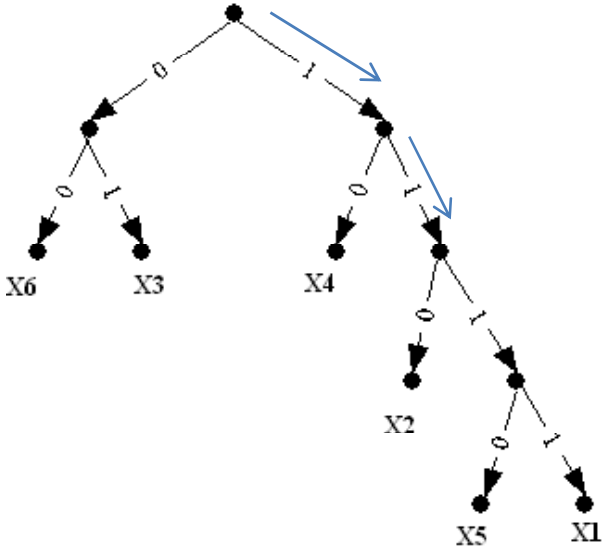


- **Acabamos de decodificar el símbolo X3. Inicializamos el nodo actual a la raíz y continuamos.**

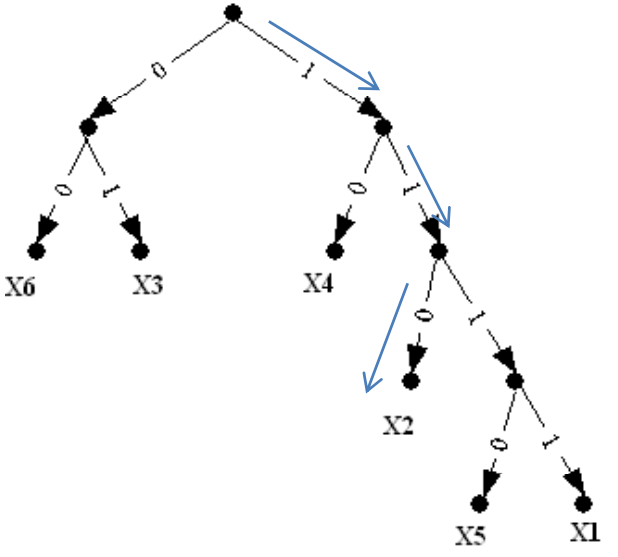
- **Decodificación con el método de Shannon-Fano**
- Decodificación del quinto símbolo: **0001110**.
- **Nos movemos desde el nodo actual siguiendo el 1:**



- **Decodificación con el método de Shannon-Fano**
- Decodificación del sexto símbolo: **0001110**.
- **Nos movemos desde el nodo actual siguiendo el 1:**



- **Decodificación con el método de Shannon-Fano**
- Decodificación del séptimo símbolo: **0001110**.
- **Nos movemos desde el nodo actual siguiendo el 0:**



- **Acabamos de decodificar el símbolo X2. En total, la fuente ha enviado 3 mensajes: X6, X3, X2.**
- **La decodificación es posible hacerla sin bits especiales de parada, porque los códigos generados son CÓDIGOS INSTANTÁNEOS.**

– **Limitación importante del método de Shannon-Fano para codificación sin ruido:**

- **El método de Shannon-Fano no siempre devuelve un código con eficiencia óptima.**

- **En el ejemplo anterior, coincide que el código sí es óptimo, y la longitud media de las palabras a enviar será:**

$$\overline{n} = \sum_{i=1}^{n} p(m_i) \cdot n_i =$$

$$= 0{,}05 * 4 + 0{,}1 * 3 + 0{,}25 * 2 + 0{,}2 * 2 + 0{,}1 * 4 + 0{,}3 * 2 = 2{,}4 bits / palabra$$

- **A continuación, veremos códigos Huffman óptimos.**

– El método de Huffman para codificación sin ruido:

Es una propuesta que **proporciona un código óptimo** a transmitir por un canal, codificando los mensajes de la fuente.

Es un método de codificación binaria, aunque extensible a codificación con mayor número de símbolos (n-arias).

Sea **M= {m_1, m_2, …, m_N}** un conjunto de mensajes a enviar (alfabeto de la fuente), junto con las probabilidades de que la fuente genere cada uno de los mensajes, **{p(m_1), p(m_2), …, p(m_N)}** . Sea **A= {a_1, a_2}** los símbolos a utilizar por un codificador.

El método trata de codificar los mensajes más probables en códigos de una menor longitud, mediante una técnica ascendente.

– El método de Huffman para codificación sin ruido:

1. Ordenar los mensajes $\{m_i\}$ en orden decreciente de probabilidades $\{p_i\}$. Llamaremos M' al conjunto de mensajes ordenado.

2. Los dos últimos símbolos (de menos probabilidad) se agrupan en uno sólo, asignando 0 al primero y 1 al segundo.

3. Se reordenan las probabilidades de nuevo, y se vuelve al paso 2 hasta que sólo quede un símbolo.

Al finalizar, el método de **Huffman** crea un árbol de codificación para cada mensaje de la fuente, al igual que **Shannon-Fano**.

- **Ejemplo: Supongamos una fuente X que emite 6 mensajes {x_1, x_2, x_3, x_4, x_5, x_6}, con probabilidades para cada símbolo {$p(x_1)$= 0,05, $p(x_2)$= 0,10, $p(x_3)$= 0,25, $p(x_4)$= 0,20, $p(x_5)$= 0,10, $p(x_6)$= 0,3}. Crear un código binario para esta fuente.**

- **Método de Huffman:**

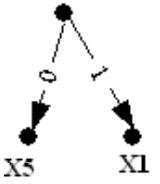
1. Ordenación por probabilidades:

x_6	x_3	x_4	x_2	x_5	x_1
0,3	0,25	0,2	0,1	0,1	0,05

2. Se cogen los dos últimos símbolos y se agrupan en uno sólo, x_{51}, y se asigna x_5 →0; x_1 →1.

– **Ejemplo: Supongamos una fuente X que emite 6 mensajes {x_1, x_2, x_3, x_4, x_5, x_6}, con probabilidades para cada símbolo {$p(x_1)$= 0,05, $p(x_2)$= 0,10, $p(x_3)$= 0,25, $p(x_4)$= 0,20, $p(x_5)$= 0,10, $p(x_6)$= 0,3}.**

El árbol queda:



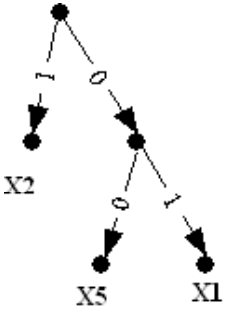
1. Ordenación por probabilidades:

x_6	x_3	x_4	X_{51}	X_2
0,3	0,25	0,2	0,15	0,1

2. Se cogen los dos últimos símbolos y se agrupan en uno sólo, x_{512}, y se asigna x_{51} →0; x_2 →1.

– **Ejemplo: Supongamos una fuente X que emite 6 mensajes {x_1, x_2, x_3, x_4, x_5, x_6}, con probabilidades para cada símbolo {$p(x_1)$= 0,05, $p(x_2)$= 0,10, $p(x_3)$= 0,25, $p(x_4)$= 0,20, $p(x_5)$= 0,10, $p(x_6)$= 0,3}.**

El árbol queda:



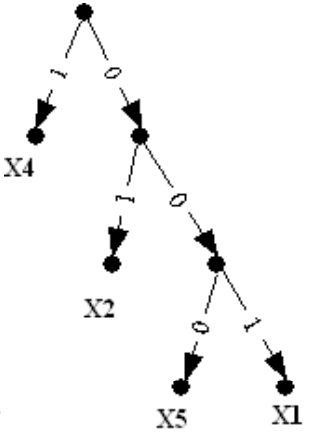
1. Ordenación por probabilidades:

x_6	x_3	x_{512}	X_4
0,3	0,25	0,25	0,2

2. Se cogen los dos últimos símbolos y se agrupan en uno sólo, x_{5124}, y se asigna x_{512}→0; x_4 →1.

– **Ejemplo: Supongamos una fuente X que emite 6 mensajes {x_1, x_2, x_3, x_4, x_5, x_6}, con probabilidades para cada símbolo {$p(x_1)$= 0,05, $p(x_2)$= 0,10, $p(x_3)$= 0,25, $p(x_4)$= 0,20, $p(x_5)$= 0,10, $p(x_6)$= 0,3}.**

El árbol queda:



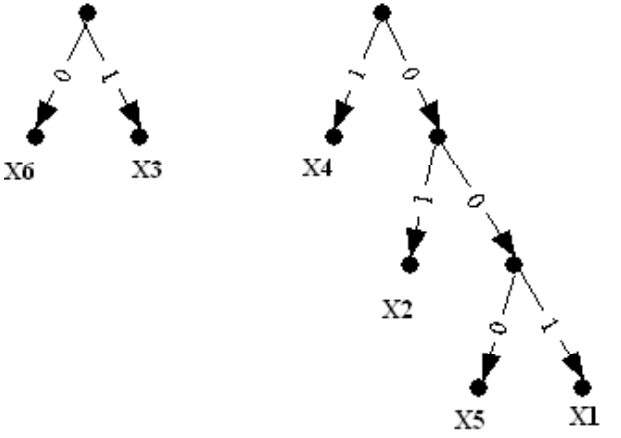
1. Ordenación por probabilidades

X_{5124}	x_6	x_3
0,45	0,3	0,25

2. Se cogen los dos últimos símbolos y se agrupan en uno sólo, x_{63}, y se asigna $x_6 \rightarrow 0$; $x_3 \rightarrow 1$.

– **Ejemplo: Supongamos una fuente X que emite 6 mensajes {x_1, x_2, x_3, x_4, x_5, x_6}, con probabilidades para cada símbolo {$p(x_1)$= 0,05, $p(x_2)$= 0,10, $p(x_3)$= 0,25, $p(x_4)$= 0,20, $p(x_5)$= 0,10, $p(x_6)$= 0,3}.**

El árbol queda:



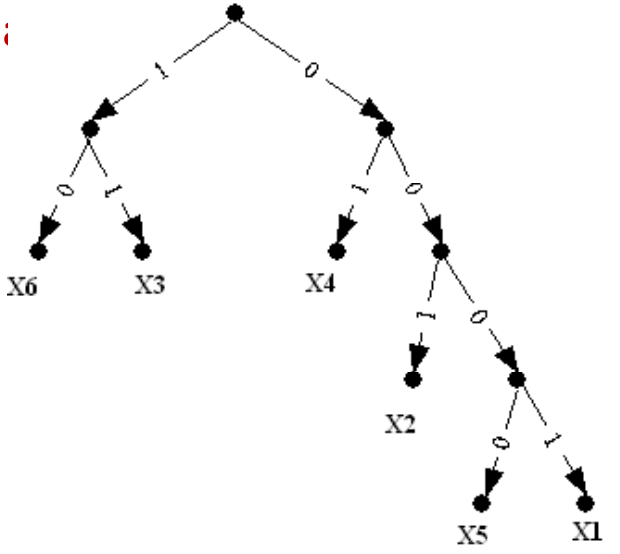
1. Ordenación por probabilidades:

X_{5124}	x_{63}
0,45	0,55

2. Se cogen los dos últimos símbolos y se agrupan en uno sólo, x_{512463}, y se asigna $x_{5124} \rightarrow 0$; $x_{63} \rightarrow 1$.

– **Ejemplo: Supongamos una fuente X que emite 6 mensajes {x_1, x_2, x_3, x_4, x_5, x_6}, con probabilidades para cada símbolo {$p(x_1)$= 0,05, $p(x_2)$= 0,10, $p(x_3)$= 0,25, $p(x_4)$= 0,20, $p(x_5)$= 0,10, $p(x_6)$= 0,3}.**

El árbol final queda:



– El código final queda: x_6→10; x_3→11; x_4→01; x_2→001; x_5→0000; x_1→0001 .
– En este caso, tiene la misma longitud que el código generado con el método de **Shannon-Fano.**

- **Diferencias y similitudes entre el método de Shannon-Fano y el método de Huffman:**
  - Los códigos **Huffman** son óptimos.
  - Los códigos **Shannon-Fano** no son óptimos, aunque en la mayoría de los casos se proporciona códigos eficientes..
  - Ambos se basan en las probabilidades (decrecientes) de los mensajes a codificar por la fuente.
  - Los códigos **Huffman** son métodos **ascendentes** (se parte desde las hojas del árbol y se termina en la raíz).
  - Los códigos **Shannon-Fano** son métodos **descendentes** (se parte desde la raíz del árbol y se termina en las hojas.
  - La decodificación de un código **Huffman** y de un código **Shannon-Fano** sigue el mismo algoritmo y tiene los mismos requisitos.
  - **Huffman es simple de implementar iterativamente.**
  - **Shannon-Fano es simple de implementar recursivamente.**

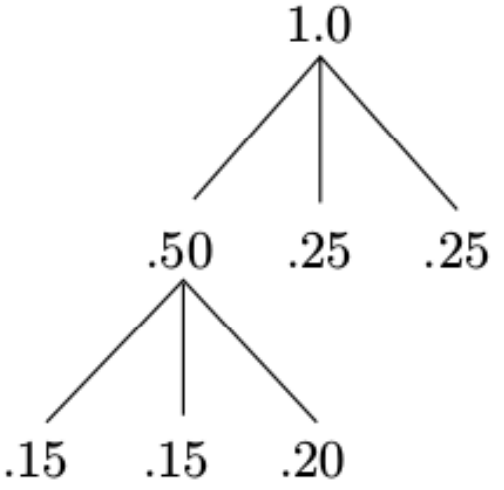
– El método de Huffman para codificación sin ruido (extensión a códigos n-arios):

1. Ordenar los mensajes $\{m_i\}$ en orden decreciente de probabilidades $\{p_i\}$. Llamaremos M' al conjunto de mensajes ordenado.

2. Los **n** últimos símbolos (de menos probabilidad) se agrupan en uno sólo, asignando 0 al primero, 1 al segundo, …, **n-1 al último.**

3. Se reordenan las probabilidades de nuevo, y se vuelve al paso 2 hasta que sólo quede un símbolo.

Ejemplo: Árbol generado para 5 símbolos {A, B, C, D, E} con probabilidades 0,25, 0,15, 0,25, 0,15, 0,2, respectivamente.



El código resultante es: A→1; C→2; B→00; D→01; E→02

Los códigos óptimos para codificación sin ruido pueden utilizarse para comprimir información.

Por ejemplo: Textos escritos en ASCII → 7 bits/símbolo. **¡Muy ineficiente!**

La codificación de textos en un idioma puede realizarse con mucha más eficiencia, utilizando códigos Huffman.

Esta idea es la base de los algoritmos de compresión más comunes hoy día, en combinación con otras técnicas.

Ejemplos de algoritmos de compresión simples:

- **Run-length Coding**
- **Lempel–Ziv Coding (precursor del LHZ y del ZIP)**

Método de compresión Run-Length Coding:

Se utiliza cuando en la secuencia de mensajes a enviar hay muchos repetidos secuencialmente:

Por ejemplo: **0 0 0 0 1 0 1 1 0 0 0 0 0 0**

Run-Length Coding consiste en contar el número de ocurrencias del código, y enviar el código y su número de ocurrencias:

0 0 0 0 1 0 1 1 0 0 0 0 0 0 → (4, 0), (1, 1), (1, 0), (2, 1), (6, 0)

Se comprime mucho cuando el mensaje a transmitir es estático (por ejemplo, una videocámara grabando todo el tiempo, se comprime la información enviando sólo las diferencias con respecto a la imagen anterior usando Run-Length Coding . Usado por JPEG y por los discos duros, a nivel hardware, para comprimir datos.

Método de compresión Lempel-Ziv:

Es una técnica de codificación de datos orientada a la reducción del número de bits que codifican una cadena de mensajes.

Algunos formatos de codificación conocidos que lo utilizan: GIF, TIFF, ZIP (método *deflate*)

Se basa en mantener en memoria un diccionario

La idea básica consiste en dividir la cadena de mensajes en ***frases***. Cada frase estará formada por una frase anteriormente utilizada y existente en el diccionario + su símbolo siguiente.

Ejemplo: Cadena 0101101000100011 0

Se divide en frases: 0 | 1 | 01 | 10 | 100 | 010 | 00 | 11 | 0

Método de compresión Lempel-Ziv:

Es una técnica de codificación de datos orientada a la reducción del número de bits que codifican una cadena de mensajes.

Algunos formatos de codificación conocidos que lo utilizan: GIF, TIFF, ZIP (método *deflate*)

Se basa en mantener en memoria un diccionario

La idea básica consiste en dividir la cadena de mensajes en ***frases***. Cada frase estará formada por una frase anteriormente utilizada y existente en el diccionario + su símbolo siguiente.

Estudiaremos la variante más conocida: LZ78

Ejemplo: Cadena 0101101000100011 0

Se divide en frases: 0 | 1 | 01 | 10 | 100 | 010 | 00 | 11 | 0

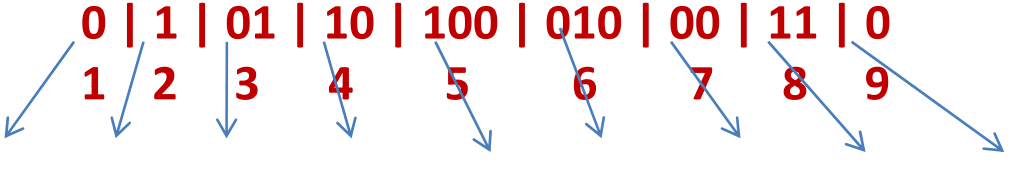
Método de compresión LZ78 (continuación, idea general):

Se numeran las frases: 0 | 1 | 01 | 10 | 100 | 010 | 00 | 11 | 0
1 2 3 4 5 6 7 8 9

El número 0 se guarda para la cadena vacía (por eso empezamos a numerar en 1).

Cada frase se corresponde con una entrada en un diccionario.

La cadena de mensajes se codifica como un par (entrada E, símbolo S), que significa *"El código que hay en la entrada E seguido del símbolo S:*



(0, 0), (0, 1), (1, 1), (2, 0), (4, 0), (3, 0), (1, 0), (2, 1), (0, 0)

Método de compresión LZ78. Descompresión:

El receptor va creando dinámicamente el diccionario según va recibiendo datos. Ejemplo para la recepción de:

(0, 0), (0, 1), (1, 1), (2, 0), (4, 0), (3, 0), (1, 0), (2, 1), (0, 0)

Diccionario:

Entrada	Código
1	0

Paso 1: Se recibe (0,0)

Significado: Cadena vacía + símbolo 0

Se introduce el código 0 en la primera entrada libre del diccionario.

Cadena descomprimida: 0

Método de compresión LZ78. Descompresión:

El receptor va creando dinámicamente el diccionario según va recibiendo datos. Ejemplo para la recepción de:

(0, 0), (0, 1), (1, 1), (2, 0), (4, 0), (3, 0), (1, 0), (2, 1), (0, 0)

Diccionario:

Entrada	Código
1	0
2	1

Paso 2: Se recibe (0,1)

Significado: Cadena vacía + símbolo 1

Se introduce el código 1 en la primera entrada libre del diccionario.

Cadena descomprimida: 01

Método de compresión LZ78. Descompresión:

El receptor va creando dinámicamente el diccionario según va recibiendo datos. Ejemplo para la recepción de:

(0, 0), (0, 1), (1, 1), (2, 0), (4, 0), (3, 0), (1, 0), (2, 1), (0, 0)

Diccionario:

Entrada	Código
1	0
2	1
3	01

Paso 3: Se recibe (1,1)

Significado: Entrada 1 + símbolo 1

Se introduce el código 01 en la primera entrada libre del diccionario.

Cadena descomprimida: 0101

Método de compresión LZ78. Descompresión:

El receptor va creando dinámicamente el diccionario según va recibiendo datos. Ejemplo para la recepción de:

(0, 0), (0, 1), (1, 1), (2, 0), (4, 0), (3, 0), (1, 0), (2, 1), (0, 0)

Diccionario:

Entrada	Código
1	0
2	1
3	01
4	10

Paso 4: Se recibe (2,0)

Significado: Entrada 2 + símbolo 0

Se introduce el código 10 en la primera entrada libre del diccionario.

Cadena descomprimida: 010110

Método de compresión LZ78. Descompresión:

El receptor va creando dinámicamente el diccionario según va recibiendo datos. Ejemplo para la recepción de:

(0, 0), (0, 1), (1, 1), (2, 0), (4, 0), (3, 0), (1, 0), (2, 1), (0, 0)

Diccionario:

Entrada	Código
1	0
2	1
3	01
4	10
5	100

Paso 5: Se recibe (4,0)

Significado: Entrada 4 + símbolo 0

Se introduce el código 100 en la primera entrada libre del diccionario.

Cadena descomprimida: 010110100

Método de compresión LZ78. Descompresión:

El receptor va creando dinámicamente el diccionario según va recibiendo datos. Ejemplo para la recepción de:

(0, 0), (0, 1), (1, 1), (2, 0), (4, 0), (3, 0), (1, 0), (2, 1), (0, 0)

Diccionario:

Entrada	Código
1	0
2	1
3	01
4	10
5	100
6	010

Paso 6: Se recibe (3,0)

Significado: Entrada 3 + símbolo 0

Se introduce el código 010 en la primera entrada libre del diccionario.

Cadena descomprimida: 010110100010

Método de compresión LZ78. Descompresión:

El receptor va creando dinámicamente el diccionario según va recibiendo datos. Ejemplo para la recepción de:

(0, 0), (0, 1), (1, 1), (2, 0), (4, 0), (3, 0), (1, 0), (2, 1), (0, 0)

Diccionario:

Entrada	Código
1	0
2	1
3	01
4	10
5	100
6	010
7	00

Paso 7: Se recibe (1,0)

Significado: Entrada 1 + símbolo 0

Se introduce el código 00 en la primera entrada libre del diccionario.

Cadena descomprimida: 01011010001000

Método de compresión LZ78. Descompresión:

El receptor va creando dinámicamente el diccionario según va recibiendo datos. Ejemplo para la recepción de:

(0, 0), (0, 1), (1, 1), (2, 0), (4, 0), (3, 0), (1, 0), (2, 1), (0, 0)

Diccionario:

Entrada	Código
1	0
2	1
3	01
4	10
5	100
6	010
7	00
8	11

Paso 8: Se recibe (2,1)

Significado: Entrada 2 + símbolo 1

Se introduce el código 11 en la primera entrada libre del diccionario.

Cadena descomprimida: 0101101000100011

Método de compresión LZ78. Descompresión:

El receptor va creando dinámicamente el diccionario según va recibiendo datos. Ejemplo para la recepción de:

(0, 0), (0, 1), (1, 1), (2, 0), (4, 0), (3, 0), (1, 0), (2, 1), (0, 0)

Diccionario:

Entrada	Código
1	0
2	1
3	01
4	10
5	100
6	010
7	00
8	11

Paso 9: Se recibe (0,0)

Significado: Cadena vacía + símbolo 0

No se introduce el código 0 en la primera entrada libre del diccionario, porque ya está contenido

Fin del algoritmo.

Cadena descomprimida: 01011010001000110

Ventajas del método de compresión LZ78:

- **El diccionario se va creando dinámicamente** (no es necesario realizar 2 pasadas sobre los datos, por lo que es menos costoso computacionalmente que otros códigos como Huffman).
- **El receptor no necesita conocer el decodificador para descomprimir los datos** (en Shannon-Fano y en Huffman es necesario que tanto el emisor como el receptor compartan el mismo árbol de codificación).

Desventajas del método de compresión LZ78:

- **El diccionario puede crecer a un ritmo elevado, sin límite**.

Existen variantes del algoritmo básico Lempel-Ziv que solucionan este problema.

- **La compresión no es óptima.**

Comparación de compresión Huffman frente a variantes de LZ (I):

- **Experimento:** Redes de sensores inalámbricas, datos sobre temperatura, humedad, ECG y texto escrito

- **Datos del experimento:**

Asral Bahari Jambek and Nor Alina Khairi , *Performance comparison of Huffman and Lempel-Ziv-Welch data compression for wireless sensor node application*, American Journal of Applied Sciences 11 (1): 119-126, 2014

Comparación de compresión Huffman frente a variantes de LZ (II):

- **Comparación del número de bits usados:**

Data type	Size before compression (Bits)	Size after compression (Bits)			
		Huffman	LZW	HLZ	LZH
Temperature	200	106	200	296	106
	400	247	400	544	247
	600	398	592	776	396
	800	550	784	992	546
Humidity	200	102	200	272	102
	400	240	400	536	240
	600	363	584	720	363
	800	485	752	896	488
ECG	200	92	184	264	88
	400	243	384	536	237
	600	411	584	800	404
	800	555	776	1000	549
Text	800	367	504	728	328
	1200	567	696	1000	491
	1600	753	840	1264	626
	2000	936	960	1480	743

Comparación de compresión Huffman frente a variantes de LZ (III):

- **Comparación del tiempo de compresión:**

		Time taken (sec)							
	Size before Compression	Huffman		LZW		HLZ		LZH	
Data type	(Bits)	Encoder	Decoder	Encoder	Decoder	Encoder	Decoder	Encoder	Decoder
Temperature	200	0.143	0.073	0.360	0.027	0.733	0.143	0.492	0.053
	400	0.790	0.183	0.848	0.119	2.166	0.916	1.040	0.209
	600	0.481	0.669	1.298	0.098	3.568	4.587	1.684	0.353
	800	0.313	1.225	2.102	0.120	3.950	6.957	2.009	0.445
Humidity	200	0.207	0.065	0.509	0.029	0.790	0.163	0.543	0.059
	400	0.341	0.569	1.506	0.059	4.098	0.518	0.783	0.231
	600	0.230	0.279	1.863	0.096	4.036	3.838	1.473	0.229
	800	0.648	0.505	1.805	0.292	2.923	6.339	3.181	0.558
ECG	200	0.187	0.072	0.748	0.043	1.237	0.163	1.156	0.058
	400	0.586	0.300	1.151	0.068	3.814	0.531	1.429	0.137
	600	0.650	0.403	3.284	0.084	4.923	4.317	2.362	0.311
	800	0.506	0.943	2.581	0.191	3.132	7.605	4.171	0.582
Text	200	0.178	0.053	0.697	0.055	0.702	0.147	0.823	0.054
	400	0.222	0.135	1.730	0.075	3.294	0.341	1.462	0.098
	600	0.447	0.106	1.984	0.107	4.316	0.629	2.424	0.175
	800	0.446	0.136	2.046	0.171	3.837	3.263	1.926	0.372
	Average	0.398	0.357	1.532	0.102	2.970	2.529	1.685	0.245

Universidad de Granada

decsai.ugr.es

Teoría de la Información y la Codificación

Grado en Ingeniería Informática

Tema 3.- Información en canales sin ruido.

DECSAI

Departamento de Ciencias de la Computación e Inteligencia Artificial