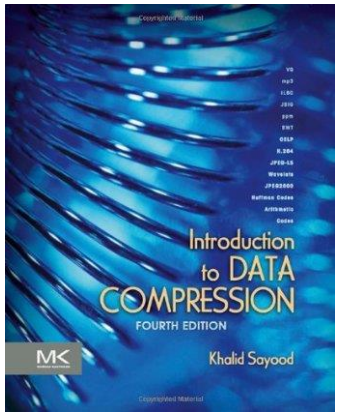
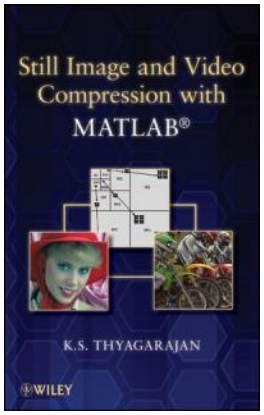
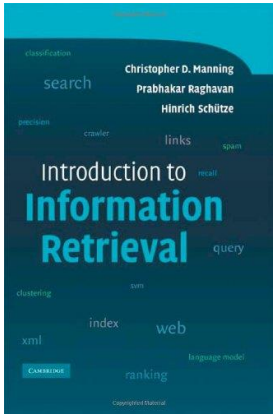
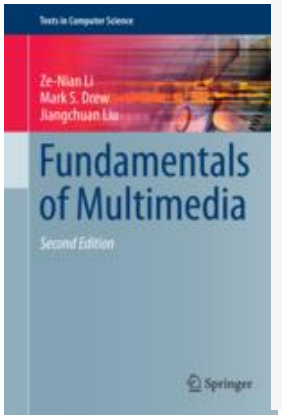








Tema 2: Codificación Huffman

Rafael Molina

Depto. de Ciencias de la Computación

e Inteligencia Artificial

Universidad de Granada

Contenidos

I. Objetivos del tema

En este tema comenzaremos describiendo un algoritmo de codificación muy conocido: el código de Huffman.

A continuación veremos algunas variaciones sobre dicho código: de mínima varianza y canónicos. Analizaremos también su longitud y su relación con la entropía.

Finalmente discutiremos algunos ejemplos de aplicación a compresión de imágenes, sonido y texto.

II El algoritmo del código de Huffman.

La codificación usando el método de Huffman[1] (código Huffman) es **un código prefijo óptimo** para un conjunto dado de probabilidades $\{p_i\}$. Es decir, su longitud media,

$$\sum_i l_i p_i \ ,$$

donde l_i es la longitud de la i-ésima palabra del código, es la menor entre todos los códigos prefijo

Puede alcanzar la entropía, aunque no lo hace siempre.
Se usa en JPEG y MPEG.

[1] Lo inventó David Huffman en un trabajo de clase. La clase, la primera en teoría de la información, la impartía Robert Fano

El código de Huffman está basado en dos propiedades de los códigos prefijo óptimos:

1. En un código prefijo óptimo, los símbolos más frecuentes –los que tienen mayor probabilidad- tienen palabras del código más cortas que los símbolos menos frecuentes.
2. En un código prefijo óptimo, los dos símbolos que ocurren con menos frecuencia tendrán la misma longitud.

Demostración de que estas condiciones las cumple un código prefijo óptimo:

- La primera condición es obvia. El número medio de bits por símbolo sería mayor si esta condición no se cumpliese.

Para probar la segunda condición usaremos reducción al absurdo.

Supongamos que existe un código prefijo óptimo, ***C***, en el que los dos símbolos menos probables no tienen la misma longitud. Veremos que esta hipótesis nos lleva a un absurdo.

- Sean P1 y P2 las correspondientes palabras del código y supongamos que long(P1)=k y long(P2)=n con k<n.

- Al ser un código prefijo, P1 no puede ser prefijo de P2. Por tanto podemos suprimir los n-k últimos dígitos de P2 y tener aún dos códigos distintos.

- Sea ***C´*** el nuevo código, ¿sigue siendo prefijo?

- Sí ya que como estos símbolos son los menos probables ninguna otra palabra puede ser más larga que estas dos y por tanto no hay peligro de que al hacer más corta la palabra P2, la nueva representación se convierta en prefijo de alguna otra.

Por tanto, ***C´*** sigue siendo prefijo. Hemos creado un nuevo código, ***C´***, que hace que ***C*** no sea óptimo, lo que contradice la hipótesis inicial sobre ***C***.

TERCERA CONDICIÓN:

Para construir el código de Huffman, Huffman le añadió el siguiente proceso de construcción a las dos propiedades vistas de los códigos prefijo óptimos:

Si u y v son los dos símbolos menos probables en el alfabeto,

si la codificación de u es m*0, la de v será m*1.

donde m es una hilera de ceros y unos y * denota concatenación.

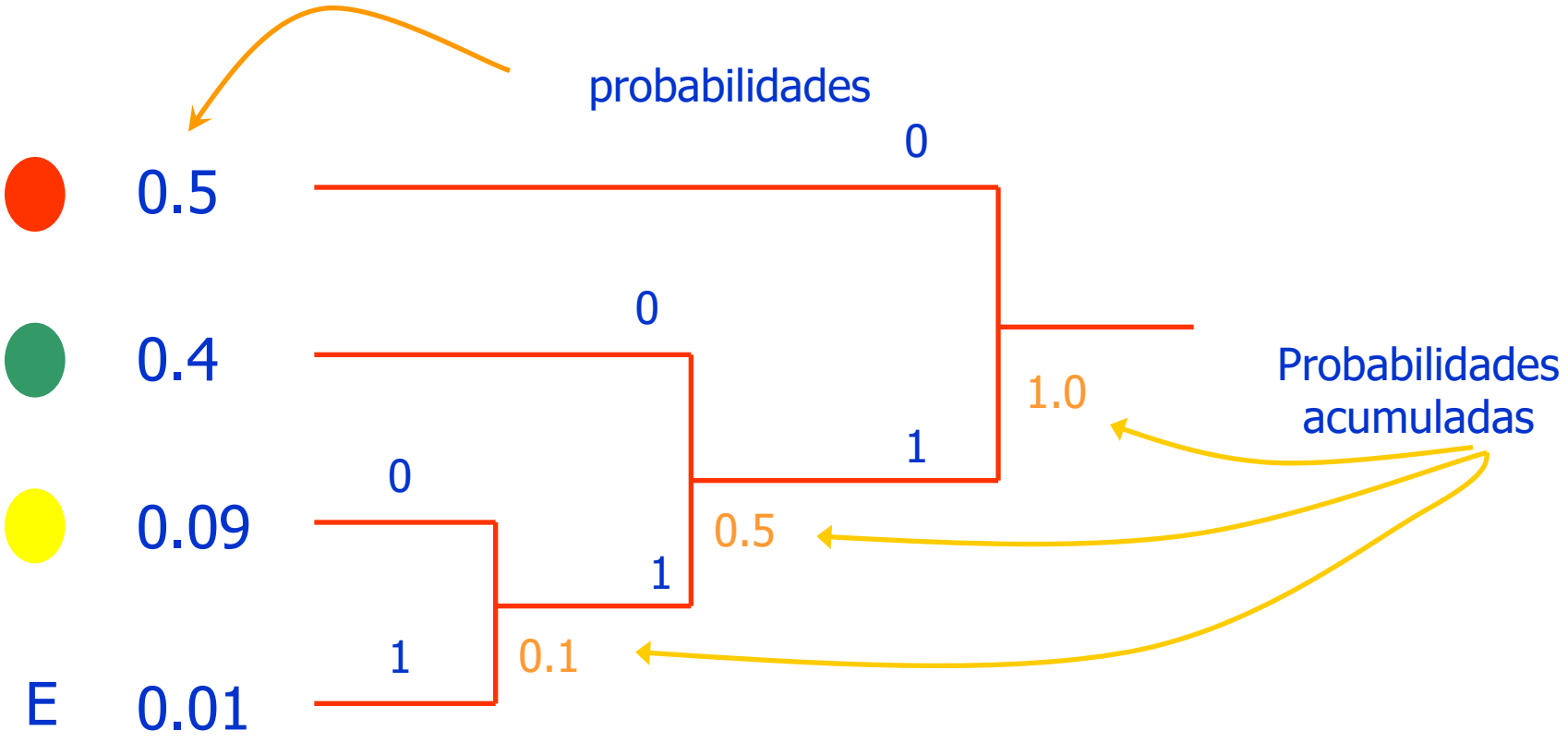
Observa que este proceso de construcción no modifica nada las dos condiciones anteriores.

Algoritmo del código de Huffman

1. Ordena los símbolos de más probable a menos probable,
2. Comienza a construir un árbol por las hojas combinando los dos símbolos menos probables,
3. Itera el procedimiento

Genera el código de Huffman correspondiente a las siguientes probabilidades

P( )=0.5
P( )=0.4
P( )=0.09
P(E)=0.01



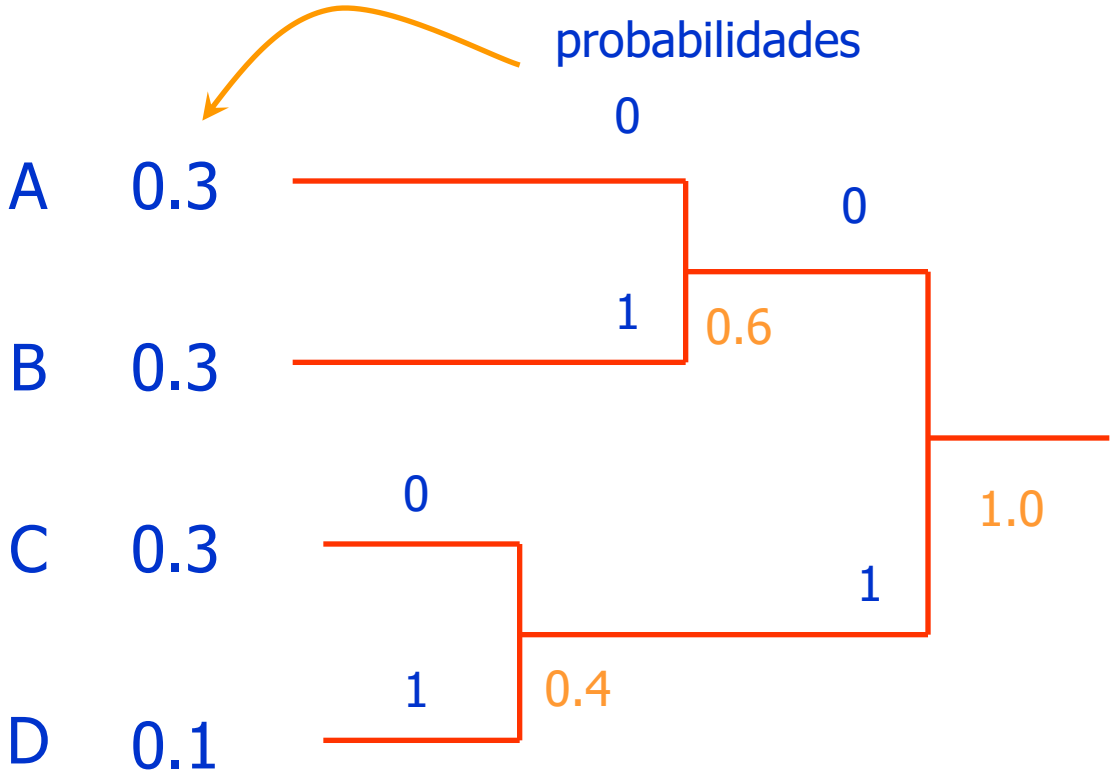
Símbolo	Código	Símbolo	Código
●	0	●	110
●	10	E	111

Nota: asignamos 1 a la rama menos probable

¿Cuál es la longitud media del código?

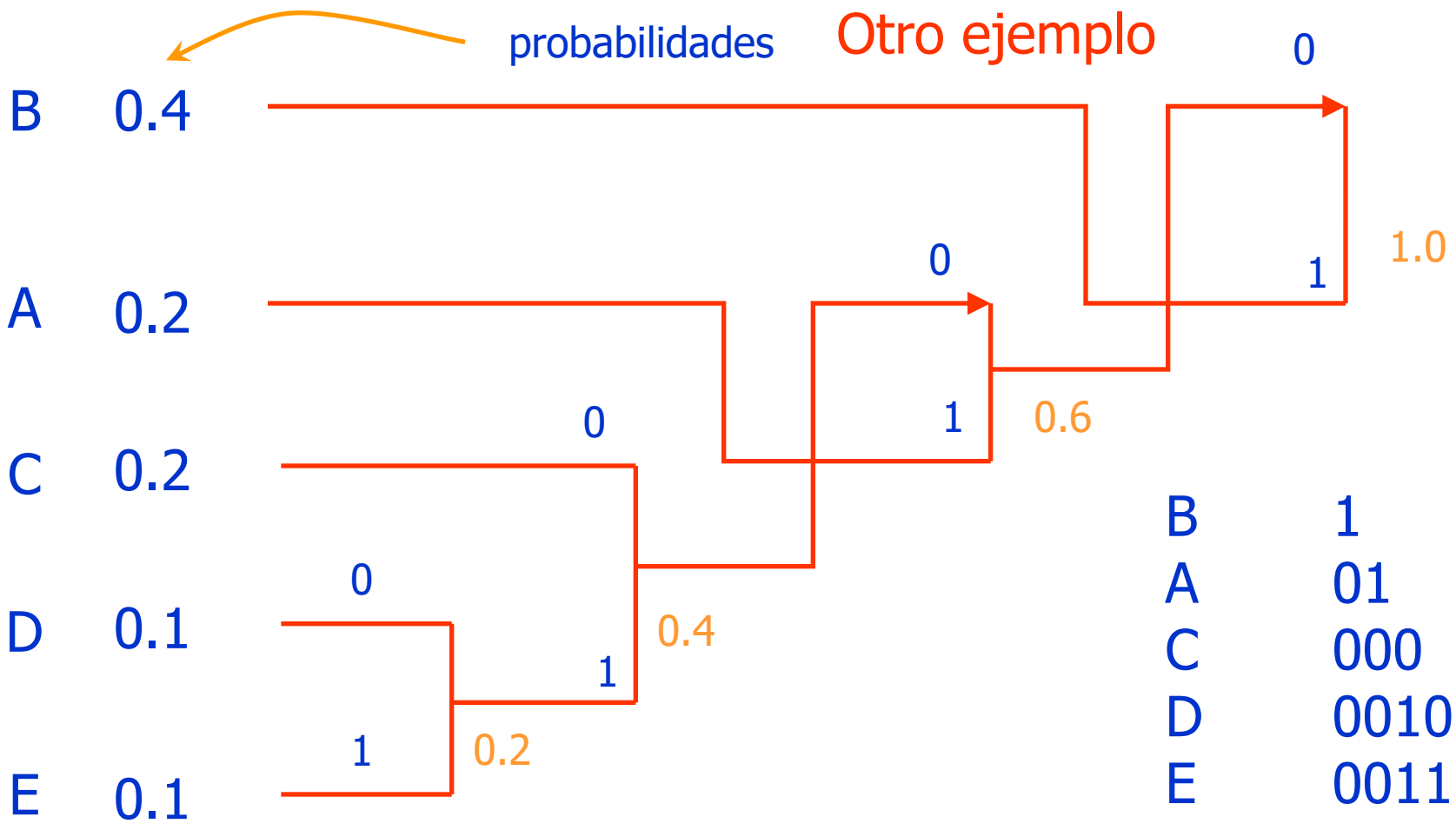
¿Cuál es la entropía de la fuente?

Otro ejemplo



A 00 B 01 C 10 D 11

Nota: asignamos 1 a la rama menos probable



Para decodificar una secuencia de palabras sólo tenemos que recorrer el árbol del código prefijo correspondiente al código Huffman como vimos en el capítulo anterior.

Nota: asignamos 1 a la rama menos probable

El código de Huffman tiene algunas características que en algunos casos se convierten en problemas:

- Como mínimo usa un bit por símbolo
- No es fácil hacer que se adapte a los cambios en la estadística (probabilidades) de las letras del alfabeto
- Es óptimo sólo si las probabilidades son de la forma 2^{-k_i}

Problema

$$P(a) = 0.9375, \quad P(b) = 0.0625$$

$$H(S) = 0.3373$$

La codificación con un bit por símbolo (Huffman) está muy lejana de la entropía.

Para este tipo de distribuciones de probabilidad muy descompensadas, (skew), tenemos que usar otras aproximaciones o ver si combinamos símbolos sucesivos

III Códigos de Huffman de mínima varianza

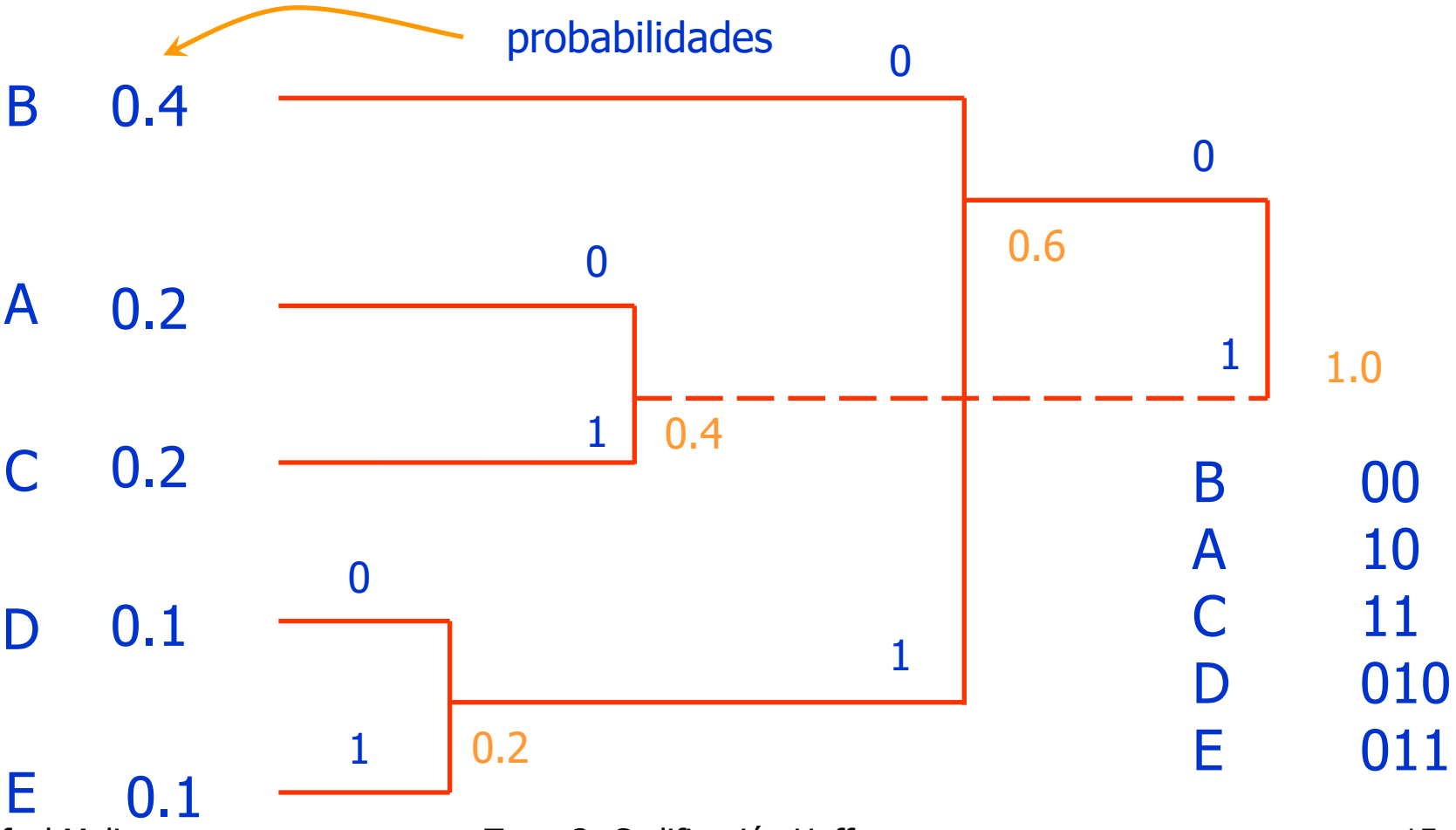
Consideremos el código de Huffman del ejemplo anterior. El número medio de bits por símbolo es

L=0.4x1+0.2x2+0.2x3+0.1x4+0.1x4=2.2 bits/símbolo

Supongamos que queremos transmitir 10000 símbolos/sg. Con una capacidad de transmisión de 22000 bits/sg el canal iría en media bien. Sin embargo, si tenemos que transmitir muchas veces seguidas los símbolos D o E podemos necesitar un ancho de banda mucho mayor.

Lo mejor sería que, manteniendo la misma longitud media, pudiésemos diseñar un código de Huffman con menor varianza en la longitud de la codificación de cada símbolo.

Para disminuir la varianza colocamos, en caso de empate en las probabilidades de los nodos, los nodos ya utilizados lo más alto posible procurando utilizarlos lo más tarde posible. Retomemos el ejemplo de la sección anterior.



IV. Códigos de Huffman Canónicos

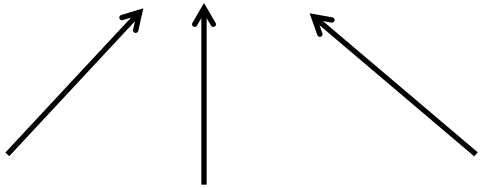
En muchas situaciones reales es conveniente transmitir el propio código de Huffman de forma eficiente: éste puede cambiar y no queremos utilizar muchos bits en su transmisión.

Consideremos nuestro segundo ejemplo del código de Huffman

A	01
B	1
C	000
D	0010
E	0011

Lo podríamos transmitir de la forma siguiente:

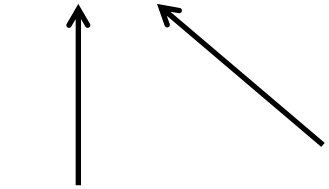
('A',2,01,'B',1,1,'C',3,000,'D',4,0010,'E',4,0011)



Letra (símbolo) longitud Palabra del código

Si el alfabeto se transmite siempre de la misma forma (con los símbolos en el mismo orden) podríamos hacer más compacta la representación enviando sólo

(2,01,1,1,3,000,4,0010,4,0011)



longitud Palabra del código

¿Podríamos suprimir las palabras del código y enviar sólo las longitudes?, es decir

(2,1,3,4,4)

La respuesta es sí pero tenemos que modificar las palabras del código de Huffman inicial. Veamos cómo

El secreto está en perder la libertad de asignar 0s y 1s a las ramas del árbol, incluyendo restricciones adicionales. En concreto, podemos hacer lo siguiente:

- Reordenar los códigos por longitud y, adicionalmente, cuando coincidan en longitud reordenarlos alfabéticamente. En nuestro caso, pasaríamos

de

A	01
B	1
C	000
D	0010
E	0011

a

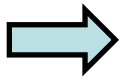
B	1
A	01
C	000
D	0010
E	0011

Observa que si la codificación fuera D=0011 y E=0010, la columna de la derecha contendría D=0011 y después E=0010

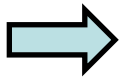
Ahora aplicamos tres reglas:

- El primer símbolo se representa mediante una palabra del código con su misma longitud pero todos ceros,
- Al símbolo siguiente se le asigna el número binario siguiente,
- Cuando la palabra de código es más larga, después de haber hecho el incremento binario de la misma longitud, añadimos ceros a la palabra hasta alcanzar la longitud de la antigua palabra de código.

B	1
A	01
C	000
D	0010
E	0011

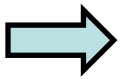


B	0
A	01
C	000
D	0010
E	0011



B	0
A	10
C	000
D	0010
E	0011

B	0
A	10
C	110
D	0010
E	0011



B	0
A	10
C	110
D	1110
E	1111

Observa que para construir la tabla final sólo necesitamos que nos transmitan (2,1,3,4,4).

En resumen:

- obtener el código de Huffman, ordenado alfabéticamente,
- ordenarlo por longitudes,
- aplicar las reglas de la transparencia anterior para obtener el código de Huffman canónico,
- codificar usando el código de Huffman canónico y transmitir las longitudes de los códigos correspondientes a las letras ordenadas alfabéticamente.
- Reconstruir el código canónico y después decodificar.

V. Longitud de los códigos de Huffman

Dado un código de Huffman, puede probarse que, su longitud media $\underline{l}$ cumple

$$H(S) \leq \underline{l} \leq H(S) + 1$$

La demostración se puede encontrar en la página 59 del libro de Sayood.

La cota superior parece sugerir agrupar los símbolos en bloques de tamaño m ya que en este caso la cota superior a la longitud media de bits por símbolo sería

$$H(S) + \frac{1}{m}$$

¿Lo entiendes?

Sin embargo, esto puede crear problemas, ¿Cuáles?

VI Aplicaciones del código de Huffman

En la clase de prácticas estudiaremos la aplicación del código de Huffman a:

Imágenes: Usaremos imágenes de niveles de gris, es decir, en cada punto toman un valor entre 0 y 255. Estimaremos la probabilidad de ocurrencia de cada nivel de gris usando el histograma y aplicaremos Huffman. El modelo será por tanto no adaptativo.

Como los valores de los píxeles están muy correlados podemos codificar la diferencia entre un píxel y el anterior, o una imagen y la anterior, usando Huffman. Veremos como mejoramos el nivel de compresión.

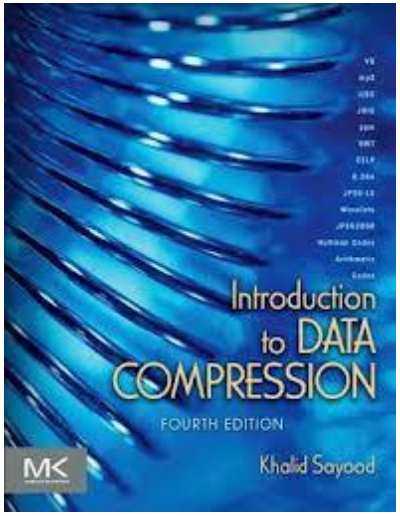
Texto: Usaremos diferentes tipos de texto y aplicaremos Huffman habiendo calculado antes las probabilidades de cada letra a partir del texto.

Eliminar la correlación aquí es más complicado y veremos como hacerlo en capítulos posteriores.

Sonido: No veremos ejemplos en este capítulo pero recuerda que cada canal estéreo se muestrea a 44.1 kHz y utiliza una representación de 16 bits. No parece razonable construir el código Huffman (tendría 2^{16} entradas), se puede calcular la entropía y, por tanto, una estimación de la mejor compresión que podemos alcanzar sin transformar los datos. Podemos también eliminar (o disminuir) la correlación entre los datos, ¿cómo? ,y volver a calcular la entropía y por tanto una estimación de la compresión a alcanzar con este modelo.

VIII Bibliografía

K. Sayood, "Introduction to Data Compression", Morgan and Kaufmann, 2012.



Apuntes de Prof. Luis Torres, Codificación de Contenidos Audiovisuales, Escuela Técnica Superior de Ingeniería de Telecomunicación de Barcelona, Universidad Politécnica de Catalunya