

Curso:  
EL271 Codificación de Fuente y de Canal  
Unidad 1: TEORÍA DE LA INFORMACIÓN Y  
CODIFICACIÓN DE FUENTE  
Semana 2, Sesión 2

---

CARLOS VALDEZ VELÁSQUEZ-LÓPEZ, DR. ING

2023-02

A solid green horizontal bar spanning the width of the slide, located at the bottom.


# Logro de la sesión

---

Al finalizar la sesión, el alumno explica los conceptos más importantes sobre la codificación de fuente discreta sin memoria, el teorema de Shannon sobre la codificación de fuente, y algunos ejemplos de códigos

# Contenido

---

- 1) Codificación de fuente discreta sin memoria (DMS)
  - 2) Códigos únicamente descifrables, longitud promedio del código
  - 3) Teorema de Shannon de la codificación de fuente, el código óptimo y la desigualdad de Kraft
  - 4) El código Morse
  - 5) El código Shannon-Fano
  - 6) El código Huffman
  - 7) Otros códigos
- 

# 1) Codificación de fuente discreta sin memoria (DMS)

---

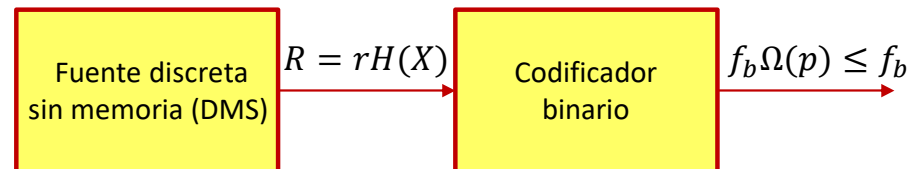
- Cuando una fuente discreta sin memoria (Discrete Memoryless Source, DMS) produce  $M$  símbolos equiprobables, se cumple que:

$$R = rH(X) = r \log M$$

- Cuando la anterior condición no se cumple, entonces:

$$R = rH(X) < r \log_2 M$$

- A continuación investigaremos la codificación de fuente con un codificador binario
- El codificador binario (de fuente) mostrado convierte los símbolos de la fuente en palabras codificadas que consisten en dígitos binarios producidos a la velocidad  $f_b$



## 2) Códigos únicamente descifrables, longitud promedio del código

- Visto desde su salida, el codificador de fuente aparece como una fuente binaria con entropía  $\Omega(p)$  y velocidad de información:

$$f_b \Omega(p) \leq f_b \log_2 2 = f_b$$

- Siempre que el código de fuente sea *únicamente descifrable*, el codificador no genera información adicional, ni tampoco la destruye, de modo que se cumple:

$$R = rH(X) = f_b \Omega(p) \leq f_b$$

- Equivalentemente:

$$f_b/r \geq H(X)$$

- La cantidad  $\bar{N} = f_b/r$  es un parámetro importante denominado la longitud promedio del código, lo cual corresponde al número promedio de dígitos binarios por símbolo:

$$\bar{N} = \sum_{i=1}^M P_i N_i$$

- En donde  $N_i$  es la longitud de la palabra codificada correspondiente al  $i$ -ésimo símbolo

### 3) Teorema de Shannon de la codificación de fuente, el código óptimo, y la desigualdad de Kraft

- El Teorema de Shannon de la codificación de fuente sostiene que el mínimo valor de  $\bar{N}$  está acotado por:

$$H(X) \leq \bar{N} \leq H(X) + \varepsilon$$

en donde  $\varepsilon$  es una cantidad positiva

- El valor de  $\varepsilon$  se puede hacer arbitrariamente pequeño
- En teoría, una codificación óptima de fuente puede lograr el mínimo valor:

$$H(X) = \bar{N}$$

- En la práctica, se suele adoptar una codificación sub óptima con  $H(X) < \bar{N}$ , si es que el código tiene una eficiencia razonablemente buena
- La razón:

$$R/f_b = H(X)/\bar{N} \leq 1$$

es una medida de la eficiencia de los códigos sub óptimos

### 3) Teorema de Shannon de la codificación de fuente, el código óptimo, y la desigualdad de Kraft

---

- El teorema de codificación de fuente asume que el código es únicamente descifrable para asegurar que no se pierde información de la fuente
- Esto impone una condición suficiente para el código binario únicamente descifrable, en donde las longitudes  $N_i$  deben satisfacer la siguiente condición (desigualdad de Kraft):

$$K = \sum_{i=1}^M 2^{-N_i} \leq 1$$

### 3) Teorema de Shannon de la codificación de fuente, el código óptimo, y la desigualdad de Kraft

---

- El proceso de codificación más simple genera un código de longitud fija, en donde todas las palabras codificadas tienen la misma longitud  $N_i = \bar{N}$
- Entonces, en la desigualdad de Kraft:

$$K = \sum_{i=1}^M 2^{-N_i} = \sum_{i=1}^M 2^{-\bar{N}} = M 2^{-\bar{N}} \leq 1$$

de donde:  $\bar{N} \geq \log M$

- Y la eficiencia:  $H(X)/\bar{N} \leq H(X)/\log M$
- Cuando  $H(X) < \log M$ , se debe codificar con longitud variable a fin de reducir el valor de la longitud promedio del código  $\bar{N}$



# Ejemplo

---

$x_i$	$P_i$	Código I	Código II	Código III	Código IV
A	1/2	00	0	0	0
B	1/4	01	1	01	10
C	1/8	10	10	011	110
D	1/8	11	11	0111	111
$\bar{N}$		2,0	1,25	1,875	1,75
$K$		1,0	1,5	0,9375	1,0

# Evaluación del código I

$x_i$	$P_i$	Código I
A	1/2	00
B	1/4	01
C	1/8	10
D	1/8	11
$\bar{N}$		2,0
$K$		1,0

➤ Longitud promedio:  $\bar{N} = \frac{1}{2} \times 2 + \frac{1}{4} \times 2 + \frac{1}{8} \times 2 + \frac{1}{8} \times 2 = 2$  binit/símbolo

➤ La entropía:

$$H(X) = \frac{1}{2} \times \log 2 + \frac{1}{4} \times \log 4 + \frac{1}{8} \times \log 8 + \frac{1}{8} \times \log 8 = 0,5 + 0,5 + \frac{3}{8} + \frac{3}{8} = 1,75 \text{ bits/símbolo}$$

➤ La eficiencia:

$$H(X)/\bar{N} = \frac{1,75}{2} = 88\% \text{ (no está mal, pero se puede mejorar con un código de longitud variable)}$$

➤ Condición de código únicamente descifrable:

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

➤ Si  $r = 2000$  símbolos/seg,  $f_b = r\bar{N} = 2000 \times 2 = 4000$  binit/sseg

➤ La velocidad de información:  $R = rH(X) = 2000 \times 1,75 = 3500$  bits/seg

# Evaluación del código II

$x_i$	$P_i$	Código II
A	1/2	0
B	1/4	1
C	1/8	10
D	1/8	11
$\bar{N}$		1,25
$K$		1,5

➤ Longitud promedio:  $\bar{N} = \frac{1}{2} \times 1 + \frac{1}{4} \times 1 + \frac{1}{8} \times 2 + \frac{1}{8} \times 2 = 1,25$  binit/símbolo

➤ La entropía:

$$H(X) = \frac{1}{2} \times \log 2 + \frac{1}{4} \times \log 4 + \frac{1}{8} \times \log 8 + \frac{1}{8} \times \log 8 = 0,5 + 0,5 + \frac{3}{8} + \frac{3}{8} = 1,75 \text{ bits/símbolo}$$

➤ Sin embargo,  $\frac{H(X)}{\bar{N}} = \frac{1,75}{1,25} = 1,4$ , lo cual no tiene sentido desde el punto de vista de la eficiencia

➤ Condición de código únicamente descifrable:

$$K = 2^{-1} + 2^{-1} + 2^{-2} + 2^{-2} = 1,5 \text{ (no cumple } K \leq 1)$$

➤ Por ejemplo, la secuencia codificada 10011 podría decodificarse como BAABB, CABB, CAD, etc.

# Evaluación del código III (código coma)

$x_i$	$P_i$	Código III
A	1/2	0
B	1/4	01
C	1/8	011
D	1/8	0111
$\bar{N}$		1,875
$K$		0,9375

➤ Longitud promedio:  $\bar{N} = \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{8} \times 4 = 1,875$  binit/símbolo

➤ La entropía:

$$H(X) = \frac{1}{2} \times \log 2 + \frac{1}{4} \times \log 4 + \frac{1}{8} \times \log 8 + \frac{1}{8} \times \log 8 = 0,5 + 0,5 + \frac{3}{8} + \frac{3}{8} = 1,75 \text{ bits/símbolo}$$

➤ En este caso,  $\frac{H(X)}{\bar{N}} = \frac{1,75}{1,875} = 93\%$  de eficiencia

➤ Condición de código únicamente descifrable:

$$K = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} = 0,9375 \text{ (cumple } K \leq 1)$$

➤ En este código se asegura que sea descifrable marcando con un dígito 0 el inicio de la palabra codificada

➤ Si  $r = 2000$  símbolos/seg,  $f_b = r\bar{N} = 2000 \times 1,875 = 3750$  binit/seg

➤ La velocidad de información:  $R = rH(X) = 2000 \times 1,75 = 3500$  bits/seg

# Evaluación del código IV (código árbol)

$x_i$	$P_i$	Código IV
A	1/2	0
B	1/4	10
C	1/8	110
D	1/8	111
$\bar{N}$		1,75
$K$		1,0

➤ Longitud promedio:  $\bar{N} = \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{8} \times 3 = 1,75$  binit/símbolo

➤ La entropía:

$$H(X) = \frac{1}{2} \times \log 2 + \frac{1}{4} \times \log 4 + \frac{1}{8} \times \log 8 + \frac{1}{8} \times \log 8 = 0,5 + 0,5 + \frac{3}{8} + \frac{3}{8} = 1,75 \text{ bits/símbolo}$$

➤ En este caso,  $\frac{H(X)}{\bar{N}} = \frac{1,75}{1,75} = 100\%$  de eficiencia (código óptimo)

➤ Condición de código únicamente descifrable:

$$K = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 1 \text{ (cumple } K \leq 1)$$

➤ En este código árbol se asegura que sea descifrable haciendo que ninguna palabra codificada aparezca como prefijo de otra palabra codificada

➤ Si  $r = 2000$  símbolos/seg,  $f_b = r\bar{N} = 2000 \times 1,75 = 3500$  binit/seg

➤ La velocidad de información:  $R = rH(X) = 2000 \times 1,75 = 3500$  bits/seg

# Comparación con el Código 0

$x_i$	$P_i$	Código 0
A	1/4	00
B	1/4	01
C	1/4	10
D	1/4	11
$\bar{N}$		2,0
$K$		1,0

➤ Longitud promedio:  $\bar{N} = \frac{1}{4} \times 2 + \frac{1}{4} \times 2 + \frac{1}{4} \times 2 + \frac{1}{4} \times 2 = 2$  binit/símbolo

➤ La entropía:

$$H(X) = \frac{1}{4} \times \log 4 + \frac{1}{4} \times \log 4 + \frac{1}{4} \times \log 4 + \frac{1}{4} \times \log 4 = 2 \text{ bits/símbolo}$$

➤ La eficiencia:  $H(X)/\bar{N} = \frac{2}{2} = 100\%$

➤ Condición de código únicamente descifrable:

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

➤ La velocidad de información (con  $r = 2000$  símbolos/seg):

$$R = rH(X) = r \log M = 2000 \times \log 4 = 4000 \text{ bits/seg}$$

➤ La velocidad o frecuencia binaria:

$$f_b = r\bar{N} = 2000 \times 2 = 4000 \text{ binit/s/seg}$$

# Demostración general del teorema de la codificación de fuente

❖ En primer lugar, hacemos que:  $Q_i = 2^{-N_i}/K$

❖ Luego, en  $\sum_i P_i \log \frac{Q_i}{P_i} \leq 0$ :

$$\sum_i P_i \log \frac{Q_i}{P_i} = \sum_i P_i \log \frac{2^{-N_i}/K}{P_i} = \sum_i P_i \left( \log \frac{1}{P_i} - N_i - \log K \right) \leq 0$$

$$H(X) - \bar{N} - \log K \leq 0$$

$$H(X) - \bar{N} \leq 0 \text{ (ya que } \log K \geq 0 \text{)}$$

❖ Por lo tanto:

$$H(X) \leq \bar{N}$$

❖ La igualdad se obtiene cuando  $K = 1, P_i = Q_i$

❖ El código óptimo con  $\bar{N} = H(X)$  requiere  $K = 1$ , y  $P_i = 2^{-N_i}, i = 1, 2, \dots, M$

❖ Ello implica:  $N_i = -\log P_i = I_i$

Lectura  
opcional

# Comentarios

---

- 1) Un código óptimo debe tener 1s y 0s equiprobables a fin de maximizar la entropía binaria  $\Omega(p) = 1$  (el Código IV del ejemplo mostrado cumple con estas propiedades óptimas)
- 2) Sin embargo, no es posible controlar las propiedades estadísticas de la fuente de información
- 3) No obstante lo anterior, la expresión  $N_i = -\log P_i = \log 1/P_i$  implica que a los símbolos que ocurren con una alta probabilidad se les debe asignar palabras codificadas de menor longitud comparada con aquellos que ocurren con una baja probabilidad



# Construyendo códigos eficientes

---

- Sea el valor entero  $N_i$  la longitud de la  $i$ -ésima palabra codificada que cae en el rango:

$$I_i \leq N_i < I_i + 1$$

- O, equivalentemente:  $\log 1/P_i \leq N_i < \log 1/P_i + 1$  (cumple la desigualdad de Kraft)
- Si se multiplica por  $P_i$  y se suma para todo  $i$ , se obtiene:  $H(X) \leq \bar{N} < H(X) + 1$
- Por lo tanto, si  $H(X) \gg 1$ , el lado derecho de la desigualdad anterior es prácticamente igual a  $\bar{N}$ , con lo cual la eficiencia será razonablemente buena
- Similarmente, si se cumple que  $N_i \approx \log 1/P_i$ , se obtendrá lo mismo
- Cuando ninguna de las condiciones anteriores se cumpla, es posible recurrir al proceso denominado codificación de extensión  $n$ -ésima
- En un código de extensión,  $n$  símbolos sucesivos se agrupan en bloques, y el codificador opera con los bloques en lugar de los símbolos individuales

# Construyendo códigos eficientes

---

- Ya que cada bloque consiste de  $n$  símbolos independientes, la entropía del bloque es justamente  $nH(X)$

- En tal caso, aplicando las reglas de codificación anteriores. se obtiene:

$$nH(X) \leq n\bar{N} < nH(X) + 1$$

- En donde  $n\bar{N}$  es el número promedio de dígitos binarios por bloque, de donde:

$$H(X) \leq \bar{N} < H(X) + \frac{1}{n}$$

- Lo cual reitera el teorema de la codificación con  $\epsilon = 1/n$
- Asimismo, demuestra que  $\bar{N} \rightarrow H(X)$  cuando  $n \rightarrow \infty$ , independientemente de la estadística de la fuente
- El código de extensión  $n$ -ésima se aproxima arbitrariamente cerca de un código óptimo

## 4) El código Morse

- Mucho antes de Shannon, Morse aplicó este principio de sentido común a su famoso código telegráfico para letras en inglés (previamente estimó las probabilidades de las letras en una imprenta)

### INTERNATIONAL MORSE CODE

1. A dash is equal to three dots.
2. The space between parts of the same letter is equal to one dot.
3. The space between two letters is equal to three dots.
4. The space between two words is equal to five dots.

A	• —	U	• • —
B	— • • •	V	• • • —
C	— • — •	W	• — —
D	— • •	X	— • • —
E	•	Y	— • — —
F	• • — •	Z	— — • •
G	— — •		
H	• • • •		
I	• •		
J	• — — —		
K	— • —	1	• — — — —
L	• — • •	2	• • — — —
M	— —	3	• • • — —
N	— •	4	• • • • —
O	— — —	5	• • • • •
P	• — — •	6	— • • • •
Q	— • — •	7	— — • • •
R	• — •	8	— — — • •
S	• • •	9	— — — — •
T	—	0	— — — — —

## 5) El código Shannon-Fano

---

### Algoritmo

$x_i$	$P_i$
A	0,5
B	0,15
C	0,15
D	0,08
E	0,08
F	0,02
G	0,01
H	0,01

- 1) Dividir los símbolos en 2 grupos de modo que las probabilidades de cada grupo sean lo más parecidas posible
- 2) Asignar el dígito 0 a cada símbolo en el grupo superior y el dígito 1 a cada dígito en el grupo inferior
- 3) En los siguientes pasos subdividir cada grupo en subgrupos y asignar los dígitos 0 y 1, siguiendo lo indicado en 1) y 2)
- 4) Cuando uno de los grupos contenga un solo dígito, no es posible otra subdivisión y la codificación del símbolo ya está completa
- 5) Cuando todos los grupos se reducen a 1 símbolo, las palabras codificadas están dadas por los dígitos asignados de izquierda a derecha

## 5) El código Shannon-Fano (ejemplo 1)

$x_i$	$P_i$	1	2	3	4	5	6	Palabra codificada
A	0,5	0						0
B	0,15	1	0	0				100
C	0,15	1	0	1				101
D	0,08	1	1	0				110
E	0,08	1	1	1	0			1110
F	0,02	1	1	1	1	0		11110
G	0,01	1	1	1	1	1	0	111110
H	0,01	1	1	1	1	1	1	111111
$H(X) = 2,15$								$\bar{N} = 2,18$

## 5) El código Huffman (ejemplo 1)

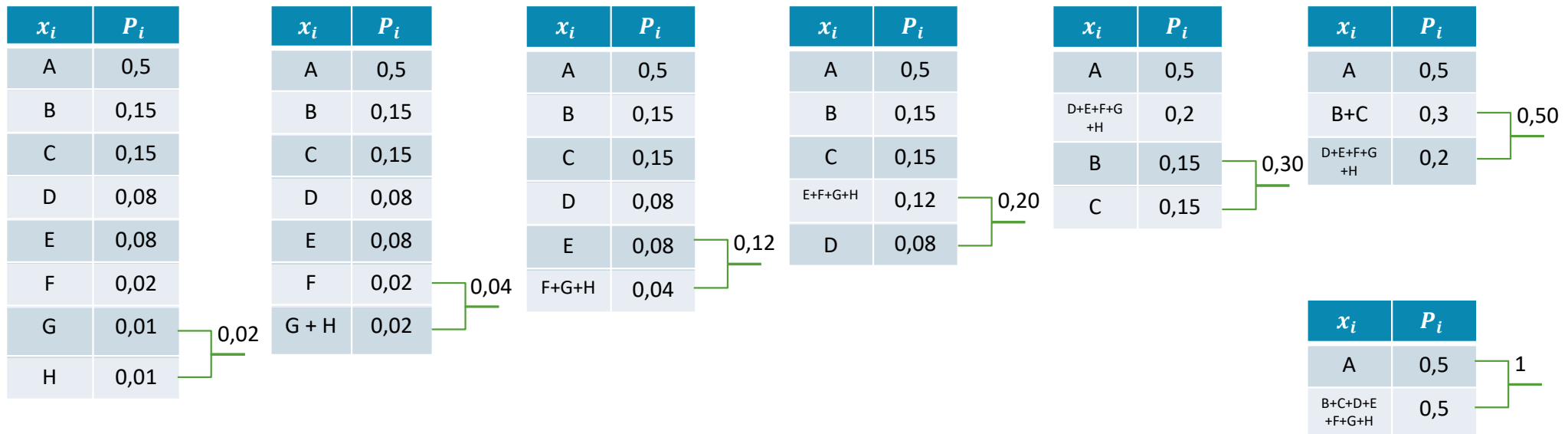
---

### Algoritmo

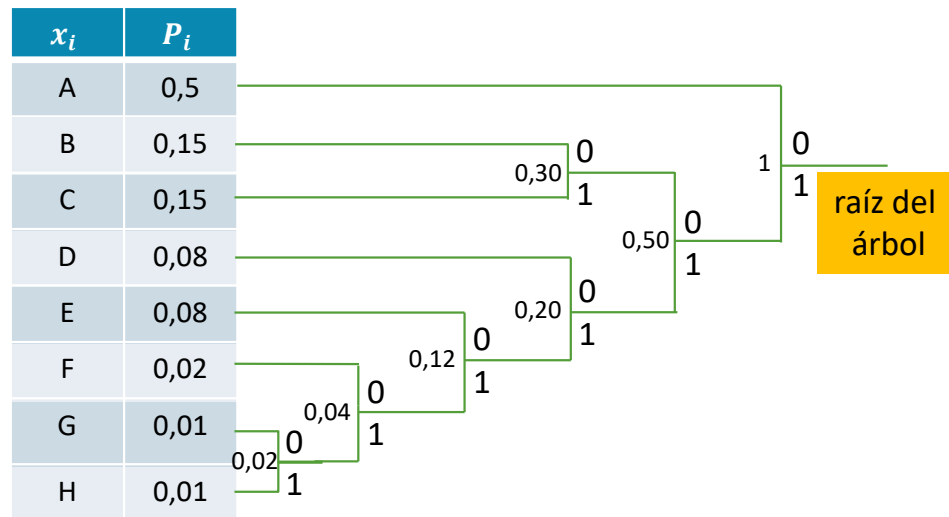
$x_i$	$P_i$
A	0,5
B	0,15
C	0,15
D	0,08
E	0,08
F	0,02
G	0,01
H	0,01

- 1) Ordenar los símbolos con los valores de probabilidades de ocurrencia de mayor a menor
- 2) Agrupar los últimos símbolos (en el ejemplo, G y H) en un símbolo equivalente, con probabilidad de ocurrencia igual a la suma de cada uno de éstos
- 3) Repetir los pasos 1) y 2)
- 4) Observando el “árbol” originado en los pasos precedentes, asociar los dígitos binarios 0 y 1 a cada par de ramas que nacen de los nodos intermedios
- 5) La palabra codificada de cada símbolo es la secuencia binaria partiendo de la raíz del “árbol” y llegando hasta el nodo terminal asociado a cada símbolo

## 5) El código Huffman (ejemplo 1)



## 5) El código Huffman (ejemplo 1)



$x_i$	$P_i$	Palabra codificada
A	0,5	0
B	0,15	100
C	0,15	101
D	0,08	110
E	0,08	1110
F	0,02	11110
G	0,01	111110
H	0,01	111111
$H(X)$		2,15
$\bar{N}$		2,18

$$\bar{N} = 0,5 \times 1 + 0,15 \times 3 + 0,15 \times 3 + 0,08 \times 3 + 0,08 \times 4 + 0,02 \times 5 + 0,01 \times 6 + 0,01 \times 6 = 2,18 \text{ bits/símbolo}$$

$$H(X) = \frac{1}{2} \times \log 2 + 2 \times 0,15 \times \log \frac{1}{0,15} + 2 \times 0,08 \times \log \frac{1}{0,08} + 0,02 \times \log \frac{1}{0,02} + 2 \times 0,01 \times \log \frac{1}{0,01} = 0,5 + 0,82 + 0,58 + 0,11 + 0,13 = 2,15 \text{ bits/símbolo}$$



# Comparación entre los códigos Shannon-Fano y Huffman

---

Sucedió en el MIT:

“En 1951, a David Huffman y a sus compañeros de clase de la asignatura “Teoría de la Información” se les permitió optar entre la realización de un examen final o la presentación de un trabajo. El profesor Robert. M. Fano asignó las condiciones del trabajo bajo la premisa de encontrar el código binario más eficiente. Huffman, ante la imposibilidad de demostrar qué código era más eficiente, se rindió y empezó a estudiar para el examen final. Mientras estaba en este proceso vino a su mente la idea de usar árboles binarios de frecuencia ordenada y rápidamente probó que éste era el método más eficiente.

Con este estudio, Huffman superó a su profesor, quien había trabajado con el inventor de la teoría de la información [Claude Shannon](#) con el fin de desarrollar un código similar. Huffman solucionó la mayor parte de los errores en el algoritmo de codificación Shannon-Fano. La solución se basaba en el proceso de construir el árbol de abajo a arriba en vez de al contrario.”

Fuente: [https://es.wikipedia.org/wiki/Codificaci%C3%B3n\\_Huffman](https://es.wikipedia.org/wiki/Codificaci%C3%B3n_Huffman)

## 5) El código Huffman (ejemplo 2)

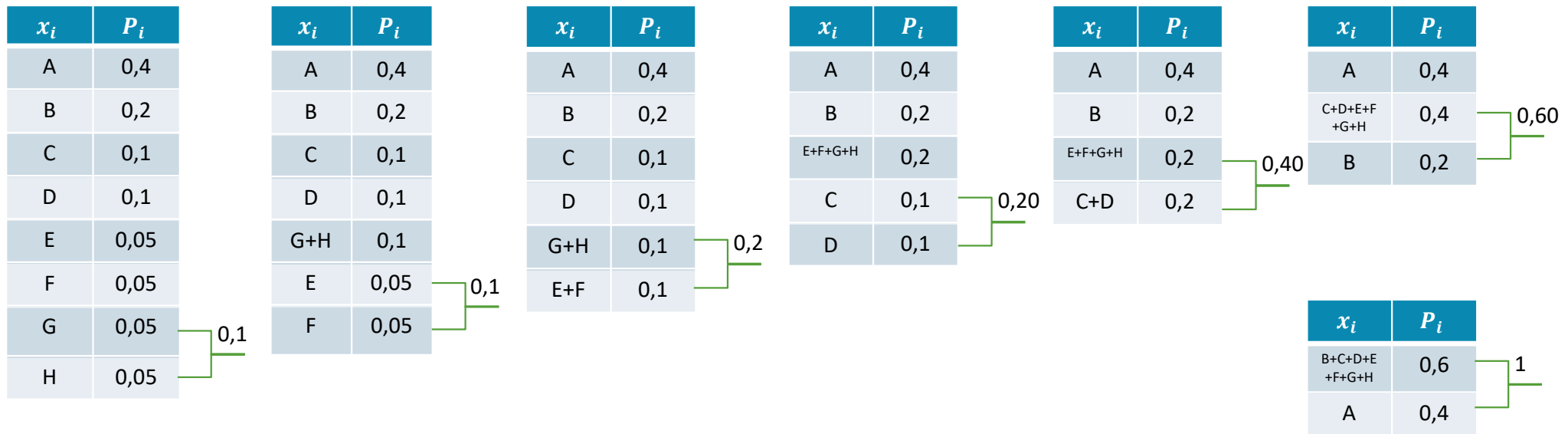
---

### Algoritmo

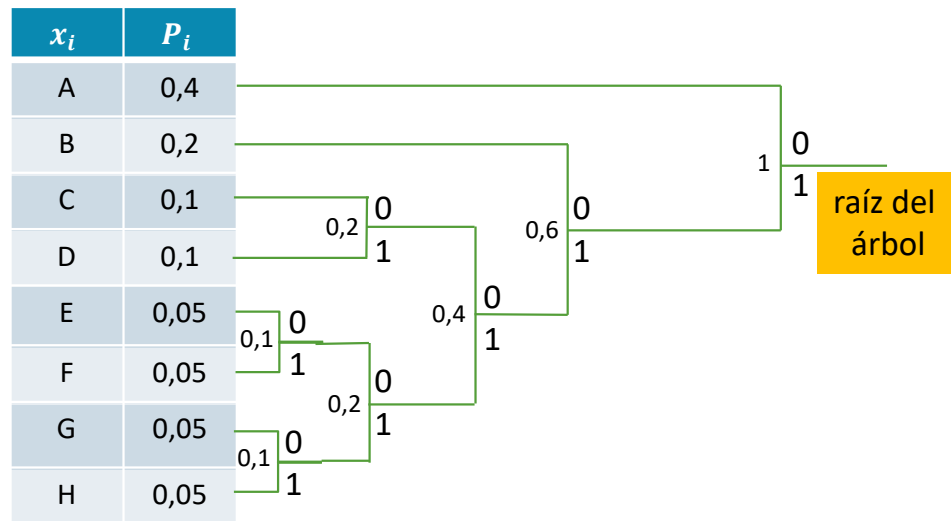
$x_i$	$P_i$
A	0,4
B	0,2
C	0,1
D	0,1
E	0,05
F	0,05
G	0,05
H	0,05

- 1) Ordenar los símbolos con los valores de probabilidades de ocurrencia de mayor a menor
- 2) Agrupar los últimos símbolos (en el ejemplo, G y H) en un símbolo equivalente, con probabilidad de ocurrencia igual a la suma de cada uno de éstos
- 3) Repetir los pasos 1) y 2)
- 4) Observando el “árbol” originado en los pasos precedentes, asociar los dígitos binarios 0 y 1 a cada par de ramas que nacen de los nodos intermedios
- 5) La palabra codificada de cada símbolo es la secuencia binaria partiendo de la raíz del “árbol” y llegando hasta el nodo terminal asociado a cada símbolo

## 5) El código Huffman (ejemplo 2)



## 5) El código Huffman (ejemplo 2)



$x_i$	$P_i$	Palabra codificada
A	0,4	0
B	0,2	01
C	0,1	0011
D	0,1	1011
E	0,05	00111
F	0,05	10111
G	0,05	01111
H	0,05	11111
$H(X)$		2,52
$\bar{N}$		2,6

$$\bar{N} = 0,4 \times 1 + 0,2 \times 2 + 0,1 \times 4 + 0,1 \times 4 + 4 \times 0,05 \times 5 = 2,6 \text{ bits/símbolo}$$

$$H(X) = 0,4 \times \log_{0,4} \frac{1}{0,4} + 0,2 \times \log_{0,2} \frac{1}{0,2} + 2 \times 0,1 \times \log_{0,1} \frac{1}{0,1} + 4 \times 0,05 \times \log_{0,05} \frac{1}{0,05} = 0,53 + 0,46 + 0,66 + 0,86 = 2,52 \text{ bits/símbolo}$$

## 6) Otros códigos

---

- Existen muchos códigos de fuente implementados mediante algoritmos denominados de compresión de datos
- Existen algoritmos de compresión de datos con pérdidas y sin pérdidas
- En todos los casos el concepto principal es la representación eficiente de los símbolos generados por la fuente, eliminando la redundancia contenida

### Ejemplos

- LZW (Liv-Zempel-Welch): algoritmo de compresión basado en diccionarios
- Zip, Rar, Winrar
- JPEG, MPEG
- WAV

# Conclusiones

---