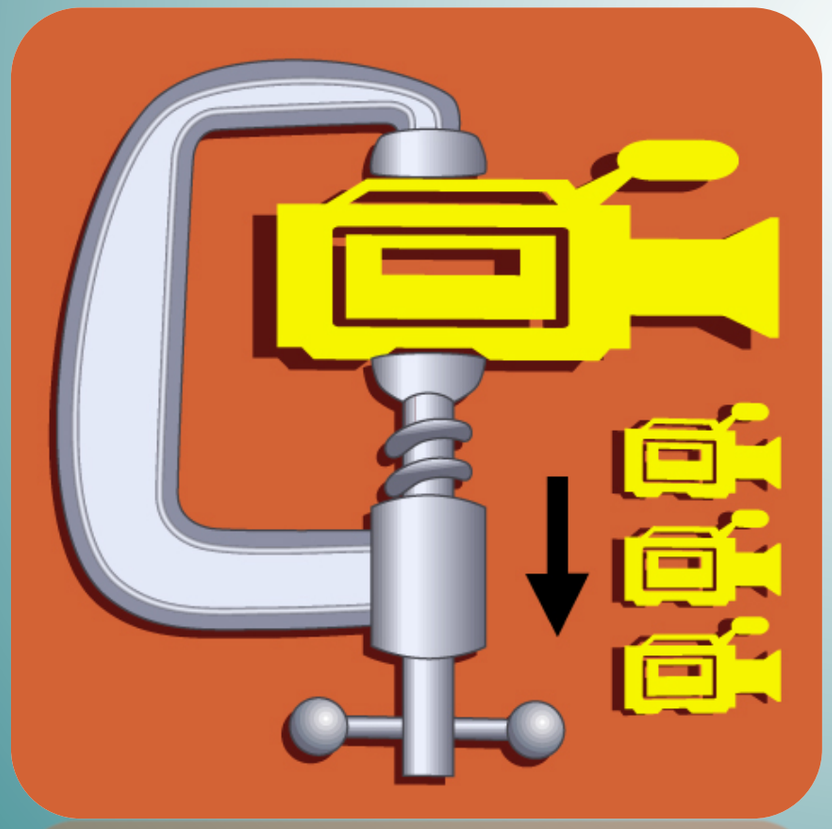
Compresión de Vídeo

Tema 1.5. Codificación entrópica



Juan A. Michell Martín
Gustavo A. Ruiz Robredo

Departamento de Electrónica y Computadores

SISTEMAS DE COMPRESIÓN DE IMAGEN

- **Compresión de Imagen:**

  Reducción de la cantidad de información necesaria para representar una imagen digital.

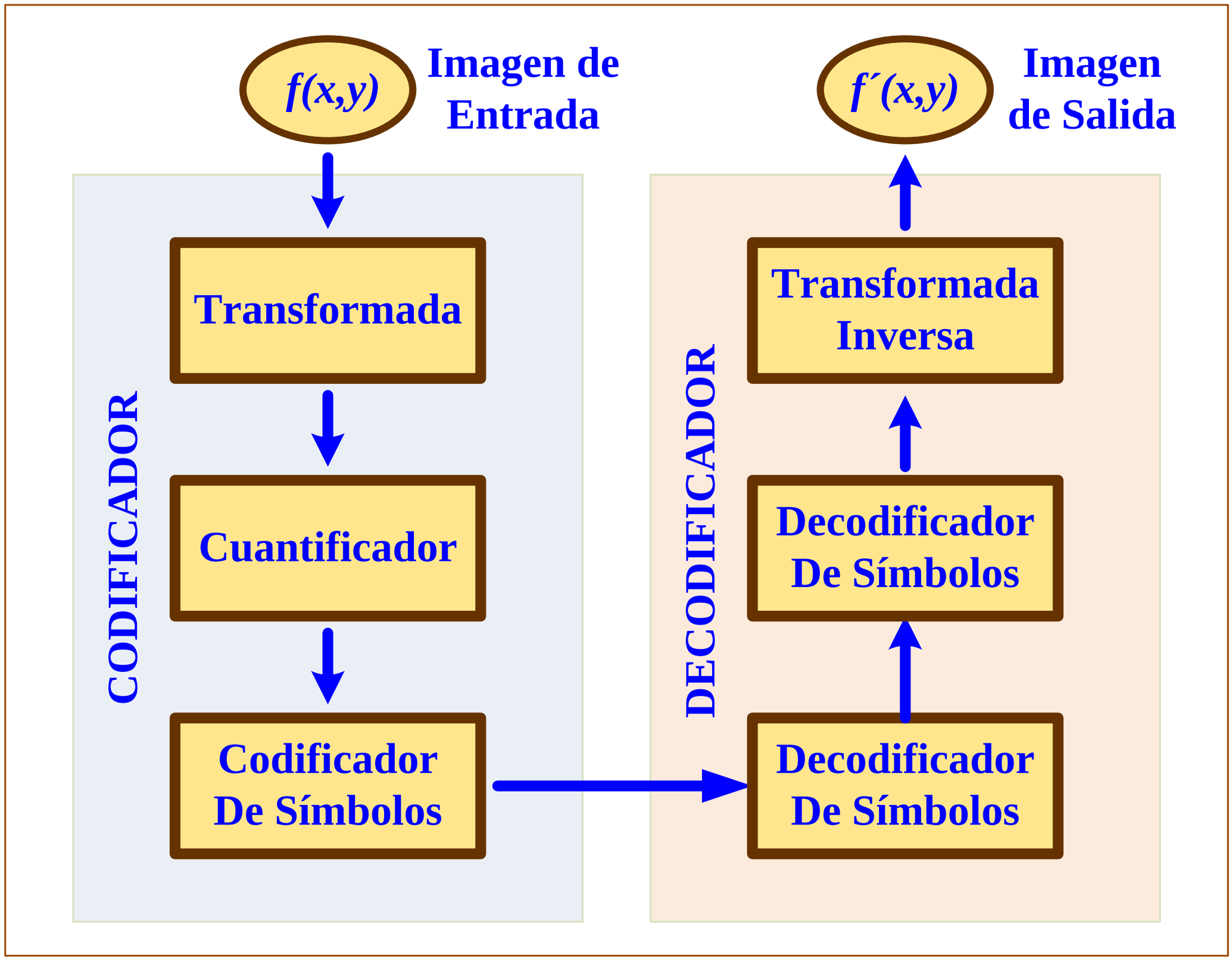
- **Objetivo:**

  Eliminar las redundancias de datos.

- **Tipos de Redundancias de Datos:**
  - **Redundancias de codificación**
  - **Redundancias entre pixeles**
  - **Redundancias psicovisuales**

- **Relación de compresión:**

$$C_R = \frac{n_o}{n_c} \quad \wedge \quad \begin{cases} n_o : no.\,bits\ imagen\ original \\ n_c : no.\,bits\ imagen\ comprimida \end{cases}$$


f(x,y)
Imagen de Entrada
CODIFICADOR
Transformada
Cuantificador
Codificador De Símbolos
f´(x,y)
Imagen de Salida
DECODIFICADOR
Transformada Inversa
Decodificador De Símbolos
Decodificador De Símbolos

- **Compresión sin Pérdidas (*lossless*):**

$$f'(x,y) = f(x,y) \quad \wedge \quad f' : \textit{imagen recuperada}$$

- **Compresión con Pérdidas (*lossy*):**

$$f'(x,y) \neq f(x,y)$$

- **Error cuadrático medio (*rmse*):**

$$e_{rms} = \left[\frac{1}{M \times N} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[f'(x,y) - f(x,y) \right]^2 \right]^{\frac{1}{2}}$$

$$e_{rms} = 0 \quad \rightarrow \quad \textit{libre de error}$$

$$e_{rms} \neq 0 \quad \rightarrow \quad \textit{imagen de salida distorsionada}$$

SISTEMAS DE COMPRESIÓN DE IMAGEN LOSSLESS

Los L niveles de gris de una imagen se representan mediante un conjunto de variables aleatorias discretas (r_k) y sus probabilidades ($p_r(r_k)$):

$$\{r_1, r_2, \ldots, r_L\} \quad \wedge \quad \{p_r(r_1), p_r(r_2), \ldots, p_r(r_L)\}$$

$$p_r(r_k) = \frac{n_k}{n} \quad \wedge \quad \begin{cases} n_k : \textit{numero de veces que aparece } r_k \\ n : \textit{número total de píxeles} \end{cases}$$

Promedio de *bits* por cada pixel de la imagen:

$$L_{avg} = \sum_{k=1}^{L} l(r_k)\, p_r(r_k)$$

$$\wedge \quad l(r_k) : \textit{no. de bits utilizado para representar } r_k$$

Ejemplo 1. Redundancias de codificación.

r_k	$p_r(r_k)$	Código 1	$l_1(r_k)$	Código 2	$l_2(r_k)$
r_1	0.1875	00	2	011	3
r_2	0.5000	01	2	1	1
r_3	0.1250	10	2	010	3
r_4	0.1875	11	2	00	2

$$Código\ 1: L_{avg1} = 2\ bits$$

$$Código\ 2: L_{avg2} = \sum_{k=1}^{4} l_2(r_k)\, p_r(r_k) = 1.8125\ bits$$

REDUNDANCIAS DE CODIFICACIÓN

- **Objetivo:**

  Minimizar el número de bits para representar los niveles de gris de una imagen, sin distorsión.

- **Información transmitida por r_k:**

$$I(r_k) = \log_2 \frac{1}{p_r(r_k)} = -\log_2 p_r(r_k)$$

- **Entropía de una imagen:**

  Mínimo número de bits por pixel (teorema de Shannon), necesario para codificar la imagen, sin pérdida de información:

$$H = \sum_{k=1}^{L} p_r(r_k) \cdot I(r_k) = -\sum_{k=1}^{L} p_r(r_k) \cdot \log_2\left(p_r(r_k)\right)$$

Ejemplo 2. Redundancias de codificación.

r_k	$p_r(r_k)$	Código 1	$l_1(r_k)$	Código 2	$l_2(r_k)$
r_1	0.500	00	2	0	1
r_2	0.250	01	2	10	2
r_3	0.125	10	2	110	3
r_4	0.125	11	2	111	3

$$H = -\sum_{k=1}^{L} p_r\left(r_k\right) \cdot \log_2\left(p_r\left(r_k\right)\right) = 1.75\ bits$$

$$Código\ 1: L_{avg1} = 2\ bits \rightarrow código\ redundante$$

$$Código\ 2: L_{avg2} = 1.75\ bits \rightarrow código\ óptimo$$

Ejemplo 3. Redundancias de codificación.

r_k	r_1	r_2	r_3	r_4	r_5	Suma
$p(r_k)$	0.1	0.15	0.3	0.16	0.29	=1
Código	010	011	11	00	10	
$l(r_k)$	3	3	2	2	2	
$l(r_k)\ p(r_k)$	0.3	0.45	0.6	0.32	0.58	=2.25
$-log_2(p(r_k))$	3.32	2.74	1.74	2.64	1.79	
$H=-p(r_k)\ log_2(p(r_k))$	0.332	0.441	0.521	0.423	0.518	=2.20

CODIFICACIÓN HUFFMAN

- **Principio básico:**

  Utilizar un código de longitud variable, asignando las palabras de código de menor longitud a los niveles de gris más probables.

- **Generación del árbol de codificación Huffman:**

  1. **Ordenar la lista de símbolos de datos en orden de probabilidad creciente.**
  2. **Combinar los dos símbolos de menor probabilidad en un nudo y asignar la probabilidad conjunta de estos símbolos al nudo.**
  3. **Reordenar los restantes símbolos de datos y nudos en orden de probabilidad creciente y repetir el paso 2, hasta llegar al nudo raíz.**

Ejemplo 1. Generación de código de Huffman.

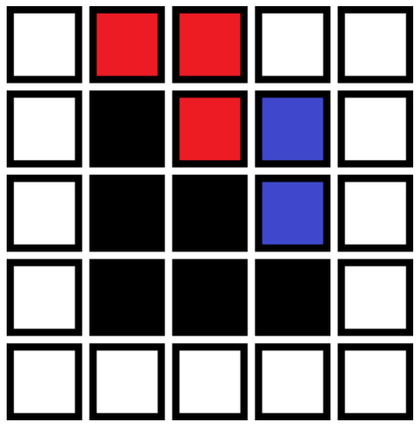


Imagen de 5*5 de 256 colores.
Tamaño: 5*5*8=200 bits
Tamaño usando paleta: 5*5*2=50bits
Ordenación de mayor a menor probabilidad:

Árbol Huffman

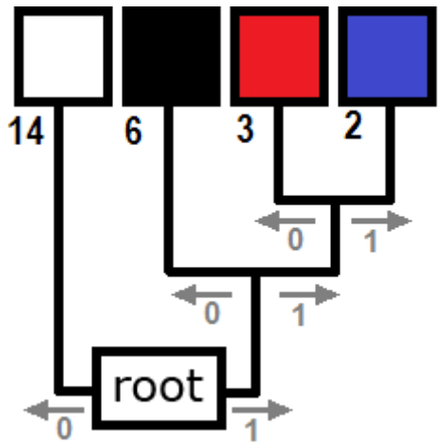


Tabla de código

color	freq.	bit code
	14	0
	6	10
	3	110
	2	111

OJO: El código no es único

Tamaño: 14*1+6*2+3*3+2*3=41 bits

Secuencia

0101111100

r_k					Suma
$p(r_k)$	14/25	6/25	3/25	2/25	=1
Código	0	10	110	111	
$l(r_k)$	1	2	3	3	
$l(r_k)\ p(r_k)$	0.56	0.48	0.36	0.24	=1.64
$-log_2(p(r_k))$	0.84	2.06	3.06	3.64	
$H=-p(r_k)\ log_2(p(r_k))$	0.47	0.49	0.37	0.29	=1.62

CODIFICACIÓN HUFFMAN EN MATLAB

`E=entropy(I)`

Devuelve un valor escalar `E` que representa la entropía de una imagen en escala de grises `I`. La entropía es una medida estadística de aleatoriedad que se puede utilizar para caracterizar la textura de una imagen.

`[count,x]=imhist(I)`

Devuelve el número de veces `count` que se repiten los niveles de gris `x` de una imagen `I`;
`stem(x,count)`: muestra el histograma.

`imhist(I)`

Visualización del histograma de una imagen.

`[dict,avglen]=huffmandict(symbols,p)`

Genera un diccionario de códigos Huffman `dict` para una fuente con un modelo de probabilidad; `symbols`: valores de la fuente; `p`: probabilidad de los símbolos; `avglen`: longitud media de las palabras de código, ponderadas con `p`.

`comp=huffmanenco(sig,dict)`

Codifica la señal `sig` utilizando los códigos de Huffman descritos por el diccionario `dict`.

`dsig=huffmandeco(comp,dict)`

Decodifica el vector de código numérico Huffman `comp` utilizando el diccionario de códigos `dict`.

Ejemplo 2. Generación de código de Huffman.

Vector	Probabilidad *P*	$\log_2(1/P)$
-1.5	0.014	6.16
-1	0.024	5.38
-0.5	0.117	3.10
0	0.701	0.63
0.5	0.101	3.31
1	0.027	5.21
1.5	0.016	5.97

$$\textit{Entropía}: \quad H = 1.507 \; \textit{bits}$$

Vector	*P*
-1.5	0.014
1.5	0.016
-1	0.024
1	0.027
0.5	0.101
-0.5	0.117
0	0.701

Primera Ordenación

Vector	*P*
-1	0.024
1	0.027
A	0.030
0.5	0.101
-0.5	0.117
0	0.701

Segunda Ordenación

Vector	*P*
-1	0.024
1	0.027
A	0.030
0.5	0.101
-0.5	0.117
0	0.701

Segunda Ordenación

Vector	*P*
A	0.030
B	0.051
0.5	0.101
-0.5	0.117
0	0.701

Tercera Ordenación

Vector	*P*
A	0.030
B	0.051
0.5	0.101
-0.5	0.117
0	0.701

Tercera Ordenación

Vector	*P*
C	0.081
0.5	0.101
-0.5	0.117
0	0.701

Cuarta Ordenación

Vector	*P*
C	0.081
0.5	0.101
-0.5	0.117
0	0.701

Cuarta Ordenación

Vector	*P*
D	0.182
-0.5	0.117
0	0.701

Quinta Ordenación

Vector	*P*
-0.5	0.117
D	0.182
0	0.701

Quinta Ordenación

Vector	*P*
E	0.299
0	0.701

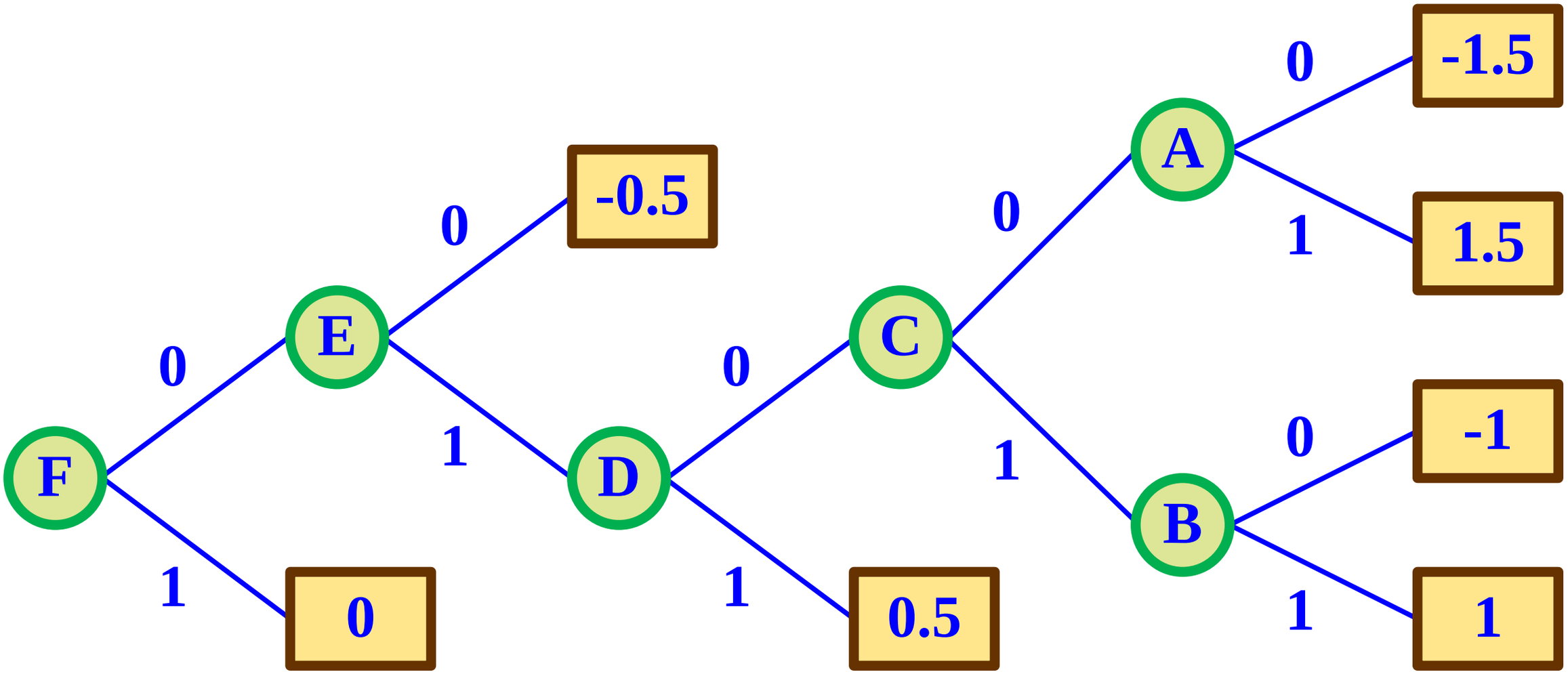
Sexta Ordenación

Vector	*P*
E	0.299
0	0.701

Sexta Ordenación

Vector	*P*
F	1

Nudo Final



Árbol de Código Huffman

Código de Huffman: Lista de símbolos

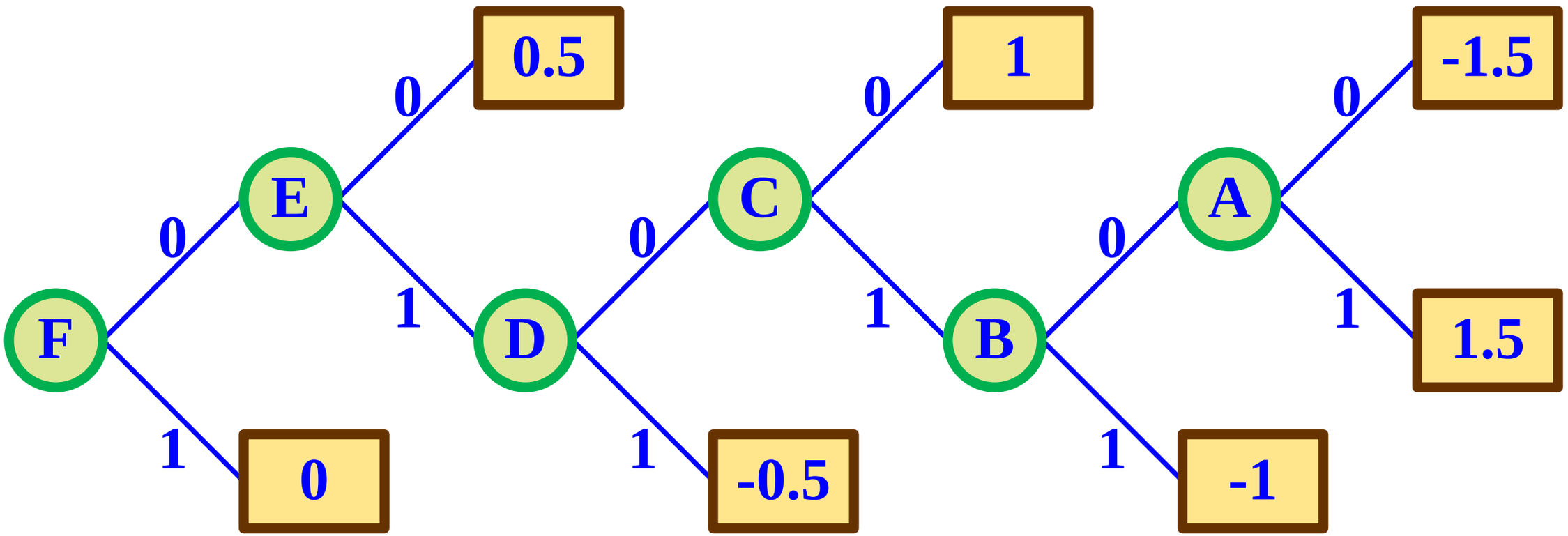
Vector	Código	*Bits* (actual)	*Bits* (ideal)
0	1	1	0.63
-0.5	00	2	3.10
0.5	011	3	3.31
-1.5	01000	5	6.16
1.5	01001	5	5.97
-1	01010	5	5.38
1	01011	5	5.21

$$H = 1.507\ bits \quad ; \quad L_{avg} = 1.643\ bits$$

Ejemplo 3. Generación de código de Huffman.

Vector	Probabilidad *P*	$\log_2(1/P)$
-1.5	0.001	9.66
-1	0.003	8.38
-0.5	0.018	5.80
0	0.953	0.07
0.5	0.021	5.57
1	0.003	8.38
1.5	0.001	9.66

$$\textit{Entropía}: \quad H = 0.3578\ \textit{bits}$$



Árbol de Código Huffman

Código de Huffman: Lista de símbolos

Vector	Código	*Bits* (actual)	*Bits* (ideal)
0	1	1	0.07
0.5	00	2	5.57
-0.5	011	3	5.8
1	0100	4	8.38
-1	01011	5	8.38
-1.5	010100	6	9.66
1.5	010101	6	9.66

$$H = 0.3578\ bits \quad ; \quad L_{avg} = 1.088\ bits$$

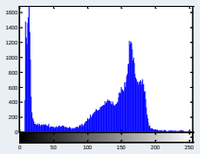
Ejemplo 4. Codificación Huffman de una imagen con Matlab

Imagen de entrada

Tablas

Símbolos
Probabilidad

Histograma
imhist()



Tablas de Huffman
huffmandict()

Tabla de Código
Huffman

Conversión
2D →1D

Codificación
Huffman
huffmanenco()

CODIFICADOR

Código binario

110010101····

Decodificación
Huffman
huffmandeco()

Conversión
1D →2D

Imagen decodificada

DECODIFICADOR

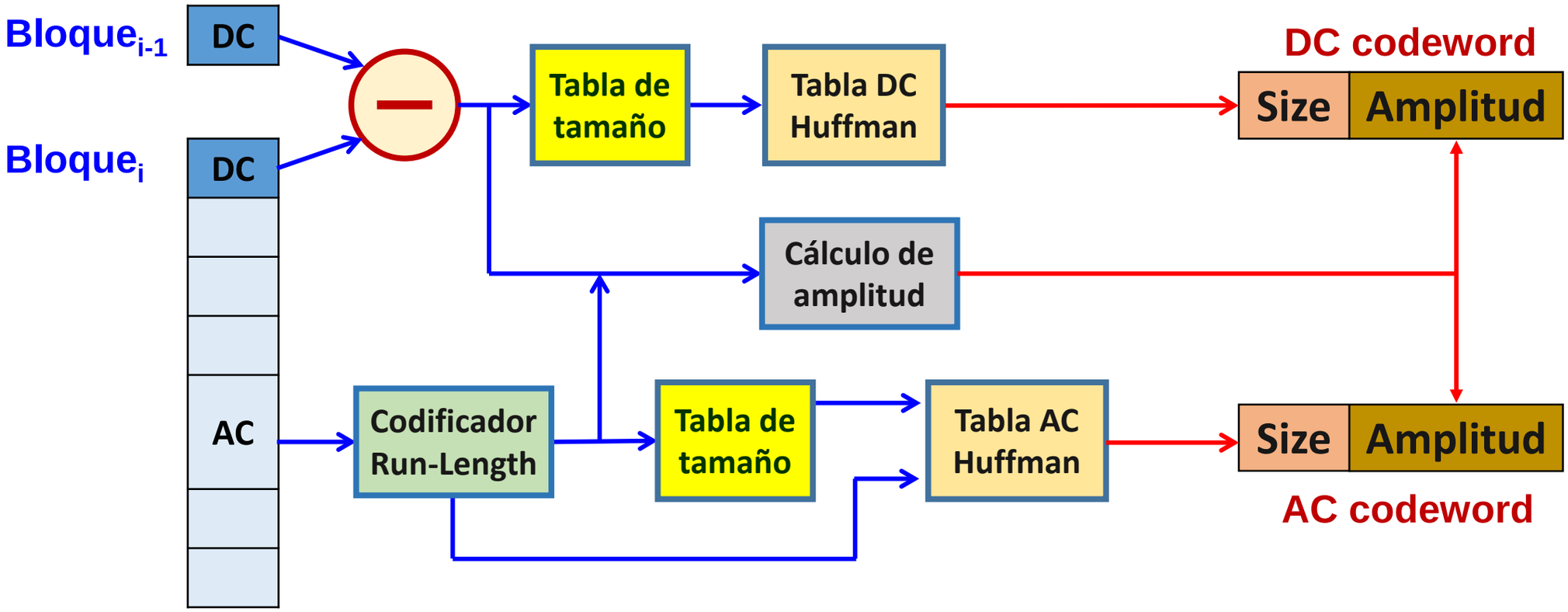
CODIFICACIÓN HUFFMAN: LIMITACIONES

- **El codificador tiene que transmitir la información contenida en la tabla de probabilidades, por lo que se reduce la eficiencia de compresión.**
- **El cálculo de la tabla de probabilidades para secuencias de gran tamaño es muy costoso.**

Solución;

- **Definir conjuntos de palabras de código basados en las distribuciones de probabilidad de un gran número de casos prácticos.**
- **Tablas de codificación *precalculadas*: Evitan generar y transmitir las tablas de probabilidad, a cambio de reducir la eficiencia de compresión.**

CODIFICACIÓN ENTRÓPICA: JPEG



Codificación Huffman para JPEG baseline

Tablas de categorías

Tablas de categorías

Cat.	Long. (bits)	Código Cat.	Rango	Codificación rango
0	2	00	0	-
1	3	010	-1,1	-1=‘0’, 1=‘1’
2	3	011	-3,-2,2,3	3=‘00’,-2=’01’,2=’10’, 3=‘11’
3	3	100	-7,…,-4,4,…7	-7=‘000’,-6=’001’,…,6=’110’, 7=‘1111’
4	3	101	-15,…,-8,8,….,15	-15=‘0000’,-14=’0001’,…,14=’1110’, 15=‘1111’
5	3	110	-31,…,-16,16,…,31	-31=‘00000’,-30=’00001’,…,30=‘11110’,31=‘11111’
6	4	1110	-63,…,-32,32,…,63	-63=‘000000’,-62=’000001’,…,62=‘111110’,63=‘111111’
7	5	11110	-127,…,-64,64,…,127	-127=‘0000000’,…,127=‘1111111’
8	6	111110	-255,…,-128,128,…,255	-255=‘00000000’,…,255=‘11111111’
9	7	1111110	-511,…,256,256,…,511	-511=‘000000000’,…,511=‘111111111’
10	8	11111110	-1023,…,-512,512,…,1023	-1023=‘0000000000’,…,1023=‘1111111111’
11	9	111111110	-2047,…,-1024,1024,…,2047	-2047=‘00000000000’,…,2047=‘11111111111’

Tablas parciales de los coeficientes del JPEG

Run/ Category	Base Code	Length	Run/ Category	Base Code	Length
0/0	**1010 (= EOB)**	**4**			
0/1	00	3	8/1	11111010	9
0/2	01	4	8/2	1111111110000000	17
0/3	100	6	8/3	1111111110110111	19
0/4	1011	8	8/4	1111111110111000	20
0/5	11010	10	8/5	1111111110111001	21
0/6	111000	12	8/6	1111111110111010	22
0/7	1111000	14	8/7	1111111110111011	23
0/8	1111110110	18	8/8	1111111110111100	24
0/9	1111111110000010	25	8/9	1111111110111101	25
0/A	1111111110000011	26	8/A	1111111110111110	26
1/1	1100	5	9/1	111111000	10
1/2	111001	8	9/2	1111111110111111	18
1/3	1111001	10	9/3	1111111111000000	19
1/4	111110110	13	9/4	1111111111000001	20
1/5	11111110110	16	9/5	1111111111000010	21
1/6	1111111110000100	22	9/6	1111111111000011	22
1/7	1111111110000101	23	9/7	1111111111000100	23
1/8	1111111110000110	24	9/8	1111111111000101	24
1/9	1111111110000111	25	9/9	1111111111000110	25
1/A	1111111110001000	26	9/A	1111111111000111	26
2/1	11011	6	A/1	111111001	10
2/2	11111000	10	A/2	1111111111001000	18
2/3	1111110111	13	A/3	1111111111001001	19
2/4	1111111110001001	20	A/4	1111111111001010	20
2/5	1111111110001010	21	A/5	1111111111001011	21
2/6	1111111110001011	22	A/6	1111111111001100	22
2/7	1111111110001100	23	A/7	1111111111001101	23
2/8	1111111110001101	24	A/8	1111111111001110	24
2/9	1111111110001110	25	A/9	1111111111001111	25
2/A	1111111110001111	26	A/A	1111111111010000	26
3/1	111010	7	B/1	111111010	10
3/2	111110111	11	B/2	1111111111010001	18
3/3	11111110111	14	B/3	1111111111010010	19
3/4	1111111110010000	20	B/4	1111111111010011	20
3/5	1111111110010001	21	B/5	1111111111010100	21
3/6	1111111110010010	22	B/6	1111111111010101	22
3/7	1111111110010011	23	B/7	1111111111010110	23
3/8	1111111110010100	24	B/8	1111111111010111	24
3/9	1111111110010101	25	B/9	1111111111011000	25
3/A	1111111110010110	26	B/A	1111111111011001	26
4/1	111011	7	C/1	1111111010	11
4/2	1111111000	12	C/2	1111111111011010	18
4/3	1111111110010111	19	C/3	1111111111011011	19
4/4	1111111110011000	20	C/4	1111111111011100	20
4/5	1111111110011001	21	C/5	1111111111011101	21
4/6	1111111110011010	22	C/6	1111111111011110	22
4/7	1111111110011011	23	C/7	1111111111011111	23
4/8	1111111110011100	24	C/8	1111111111100000	24
4/9	1111111110011101	25	C/9	1111111111100001	25
4/A	1111111110011110	26	C/A	1111111111100010	26

Ejemplo de codificación entrópica JPEG

Se aplica a los bloques 8x8 de coeficientes ordenados en zig-zag.

Step 1. Coeficiente DC. Se resta al coeficiente DC del bloque 8x8 anterior.

29 **Coeficiente DC del bloque anterior.**

22,-2,3,1,1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,…

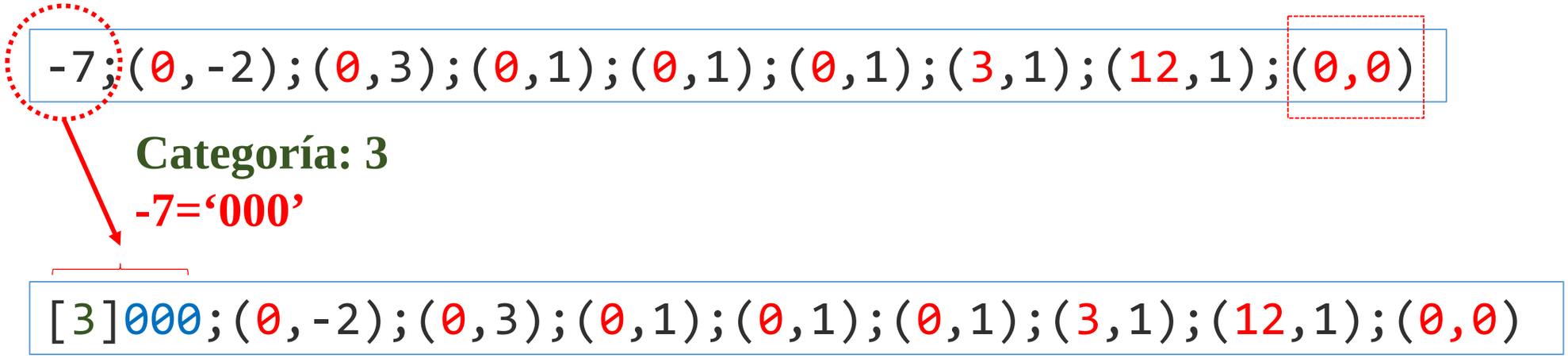
-7,-2,3,1,1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,…

Step 2. Run-length coding (número de ceros precedentes,valor coeficiente)

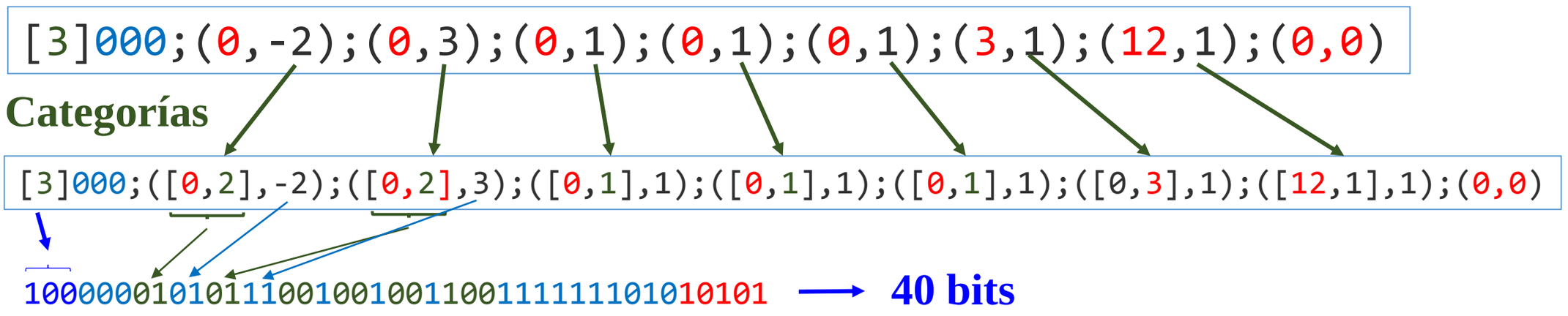
EOB (end of block)

-7;(0,-2);(0,3);(0,1);(0,1);(0,1);(3,1);(12,1);(0,0)

- **Aplicación de tablas de categorías a DC.**



- **Aplicación de tablas de categorías a AC.**



Ejemplo de decodificación entrópica JPEG

0110101101101001011101101111111110111001010

DC

011 → Cat 2
01 → -2
0110110100101110110111111111110111001010 resto

AC

01 → 0/2 (Run length 0, Cat. 2)
10 → 2
1101001011101101111111110111001010 resto

11010 → 0/5 (Run length 0, Cat. 5)
01011 → -21
1011011111111110111001010 resto

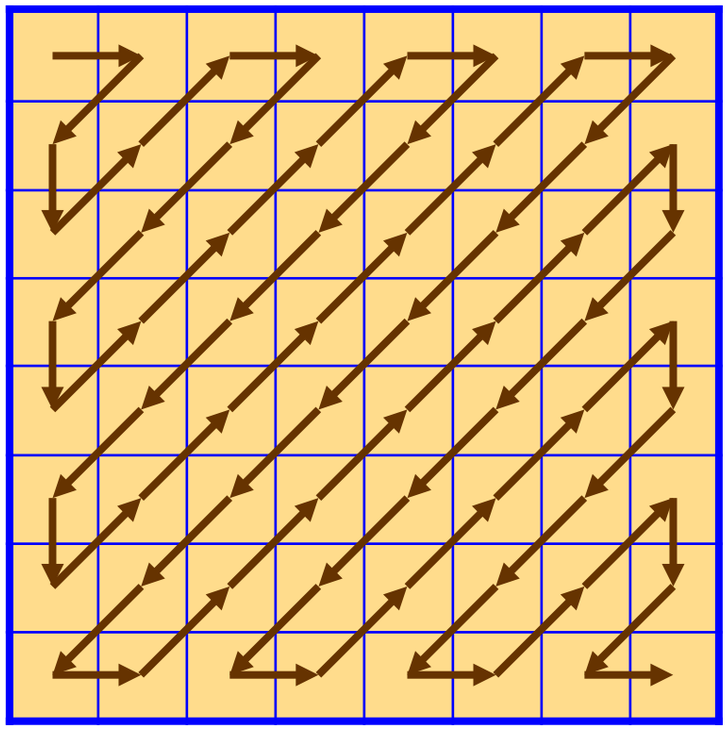
1011 → 0/4 (Run length 0, Cat. 4)
0111 → -8
1111111011100 1010 resto

11111110111 → 7/2 (Run length 7, Cat. 2)
00 → -3
1010 EOB

-2,2,-21,-8,0,0,0,0,0,0,0,0,-3, resto 0

➢ Reordenación en zig-zag

-2,2,-21,-8,0,0,0,0,0,0,0,-3, resto 0



-2	2	0	0	0	0	0	0
-21	0	0	0	0	0	0	0
-8	0	0	0	0	0	0	0
0	-3	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

JPEG BITSTREAM

Width*height
Componentes
Matriz de cuantización, …

Tablas de Huffman, componentes,…

Tablas de Huffman, componentes,…

Más información página 38 del standard JPEG