IMAGE COMPRESSION

Le Thanh Ha, Ph.D
Assoc. Prof. at University of Engineering and Technology,
Vietnam National University
ltha@vnu.edu.vn; lthavnu@gmail.com; 0983 692 592

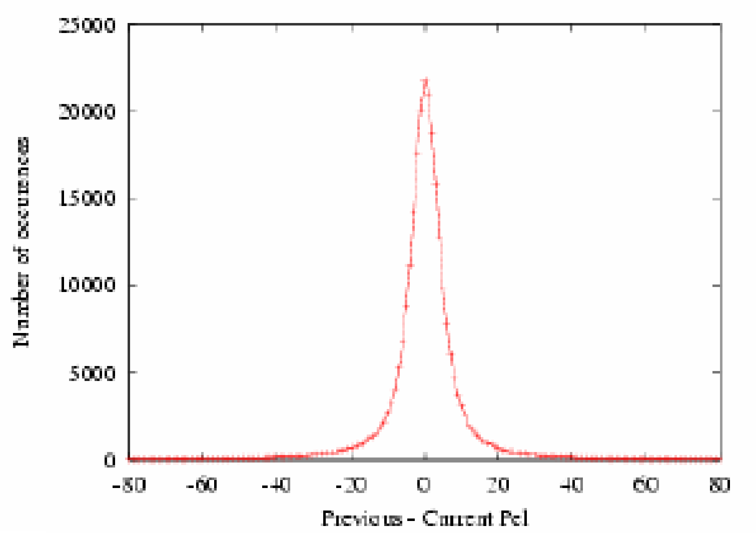
Why image compression?

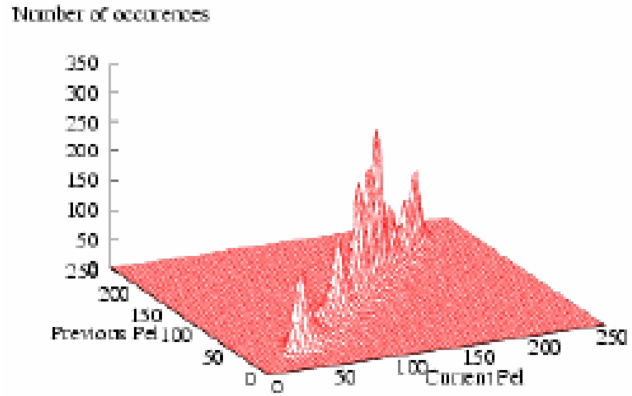
- Digital image acquiring devices develop fast $\rightarrow$ Image data become huge.
- However: storage and transmission systems are still limited.

➢ We need methods and algorithms to reduce the size of sound and image to efficiently store and transmit over network.

Spatial dependency

- Neighboring pixels are high corellated.
- The value of a pixel is probability depending on those of its neighbors.





Predictive and lossless Encoding

- Entropy coding
  - Based on statistical properties of the information.
  - The original information and decoded information are exactly the same.

- Predictive coding
  - Based on the spatial and temporal dependency.
  - The decoded data may different from the original data.

Entropy

- Information measure
  - A symbol x with probability p contains information measured by:

$$I(x) = -\log(p(x))$$

  - Information measure does not depend on the value of symbol.
  - Information measure depends on the probability of symbol.
  - When the base of log function is 2, the unit of information measure is bit.

Entropy

- Information Entropy
  - Entropy is defined as the average information measure of all symbols in a source. Entropy, *H*, is defined as follows:

$$H(X) = \sum_{x \in \chi} I(x) = \sum_{x \in \chi} -p(x) \cdot \log(p(x))$$

  - It means that entropy of an information source is a function of its symbol probabilities. Entropy is maximum when all symbols appear with the same probability.

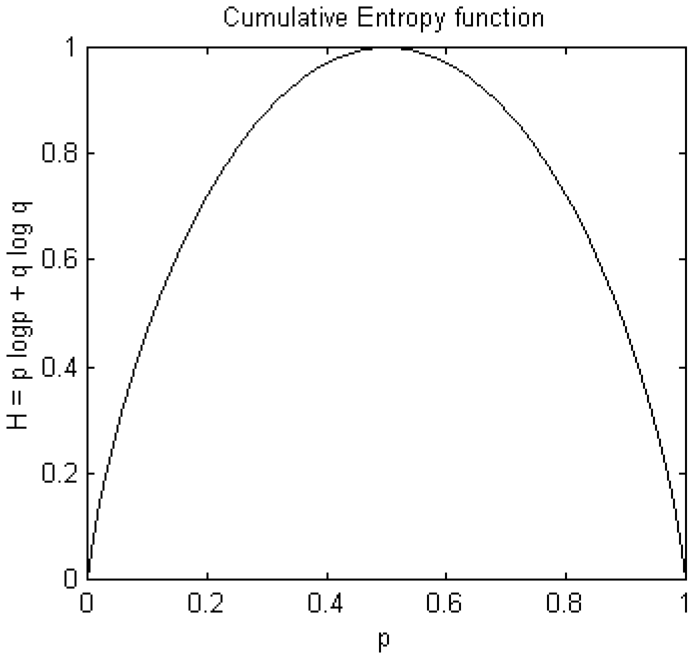
Entropy

- For example, an information source χ={0,1}, p(0)=1/3, p(1)=1-p(0)=2/3

$$
\begin{aligned}
H(X) &= -p(0)\log(p(0) - p(1)\log(p(1)) \\
&= -\frac{1}{3}\log(\frac{1}{3}) - \frac{2}{3}\log(\frac{2}{3}) \\
&= 0.646
\end{aligned}
$$

Entropy

- Entropy of source with 2 symbols
  - A convex function of p
  - When p comes to 0 or 1, source χ carries smaller information.
  - Maximum at p=0.5



Entropy

- Calculate H(X):

$$X = \begin{cases} a & p(a) = 1/2 \\ b & p(b) = 1/4 \\ c & p(c) = 1/8 \\ d & p(d) = 1/8 \end{cases}$$

Entropy – Binary image

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

$$P_0 = \frac{63}{64}$$

$$P_1 = \frac{1}{64}$$

$$H = -\frac{63}{64}\log_2 \frac{63}{64} - \frac{1}{64}\log_2 \frac{1}{64}$$

$$= 0.116 \text{ bits/pixel}$$

$$\begin{matrix} 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{matrix}$$

$$P_0 = \frac{32}{64} = \frac{1}{2}$$

$$P_1 = \frac{32}{64} = \frac{1}{2}$$

$$H = -\frac{1}{2}\log_2 \frac{1}{2} - \frac{1}{2}\log_2 \frac{1}{2}$$

$$= 1.0 \text{ bits/pixel}$$

Data compression

- Symbols with high probability are assigned with short codes, wheres symbols with low probability are assigned with longer codes.

- The length of coded data is shortened because the average length is reduced.

Data compression theory

- Code C for a source X is a mapping from χ (space of X), to D* where D* is a set of code sequence. For example:
  - C(Red)=00, C(Blue)=11 is code of χ={Red,Blue) and D={0,1}

- The average length of C:

$$L(C) = \sum_{x \in \chi} p(x) l(x)$$

Data compression theory

- Code C is called prefix code if there is no code is prefix of any other code.
  - 00 is prefix of 001
  - Code C with C(1)=0, C(2)=10, C(3)=110, C(4)=111 is a prefix code.
  - With prefix code, decode process can be done on-the-fly.

Entropy coding

- Example – fixed length code

Symbol x	Probability	Code C(x)	Code length L(c)
A	0.75	00	2
B	0.125	01	2
C	0.0625	10	2
D	0.0625	11	2

Average bits = 0.75*2 + 0.125*2 + 0.0625*2 + 0.0625*2 = 2.0 bits/symbol

Entropy coding

- For example – variable length code

Symbol x	Probability	Code C(x)	Code length L(c)
A	0.75	0	1
B	0.125	10	2
C	0.0625	110	3
D	0.0625	111	3

Average bits = 0.75*1 + 0.125*2 + 0.0625*3 + 0.0625*3 = 1.375 bits/symbol

Entropy coding

- Exp-Golomb codes
  - Used in coding video (motion vectors)
- Huffman codes
  - Data compression, still image compression.
- Arthimetic codes
  - Entropy coding for H.264/AVC
- Run-length codes

Exp-Golomb code

- Easy and simple to implement on hardware devices.
- Suitable for source having laplace or exponential distribution.
- Used for coding the length of motion vectors in video coding.

Exp-Golomb code

- Structure of Exp-Golomb code is as following:

  [M zeros][1][INFO]

  – INFO is M-bit field number
  – M = floor($\log_2$[code_num+1])
  – INFO = code_num+1 – 2^M

- Decode:
  – Read M characters ‘0’, follow by a ‘1’
  – Read M bits of INFO
  – Code_num = 2^M + INFO - 1

Exp-Golomb codes

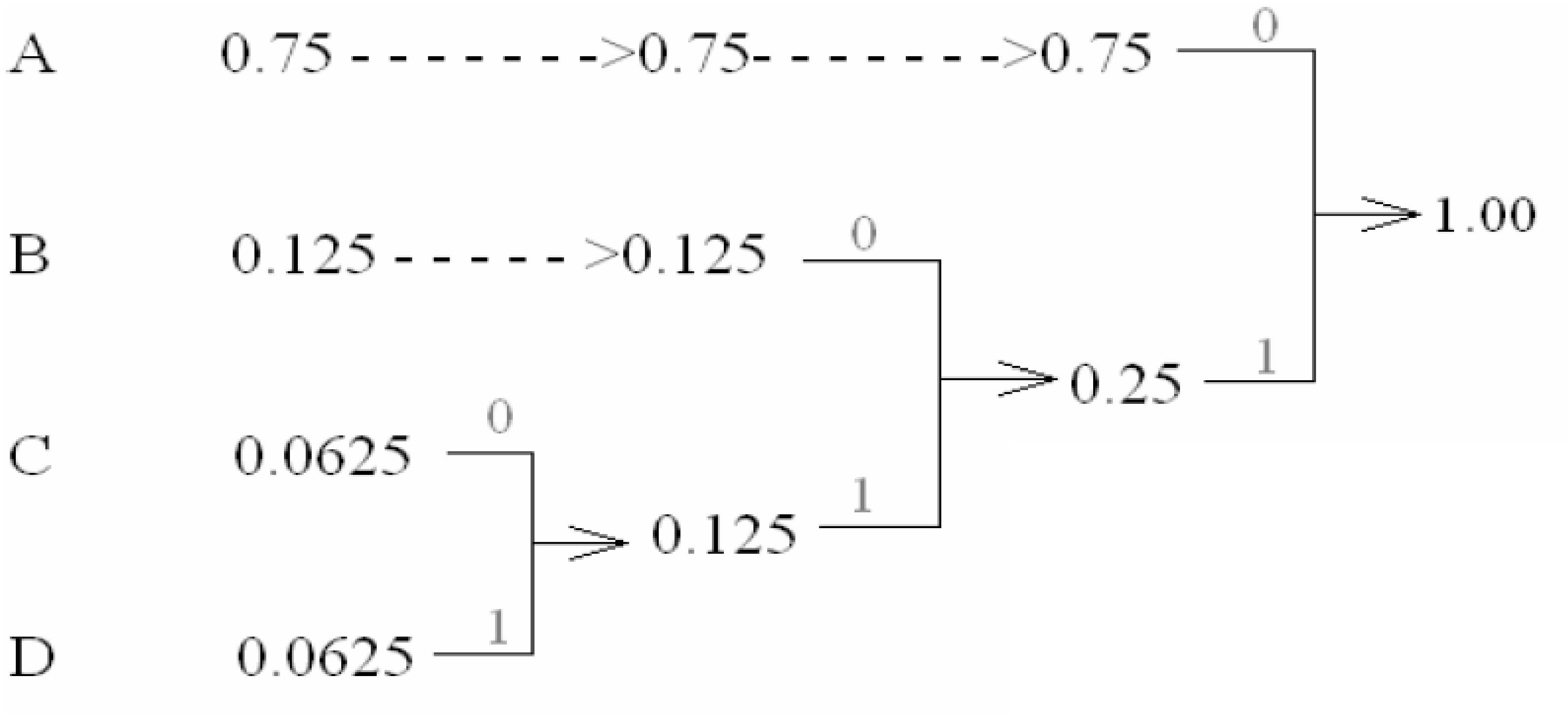
Code	Unsigned number	Signed number
1	0	0
010	1	1
011	2	-1
00100	3	2
00101	4	-2
00110	5	3
00111	6	-3
0001000	7	4
0001001	8	-4
0001010	9	5
0001011	10	-5
0001100	11	6
…	…	…

Huffman codes

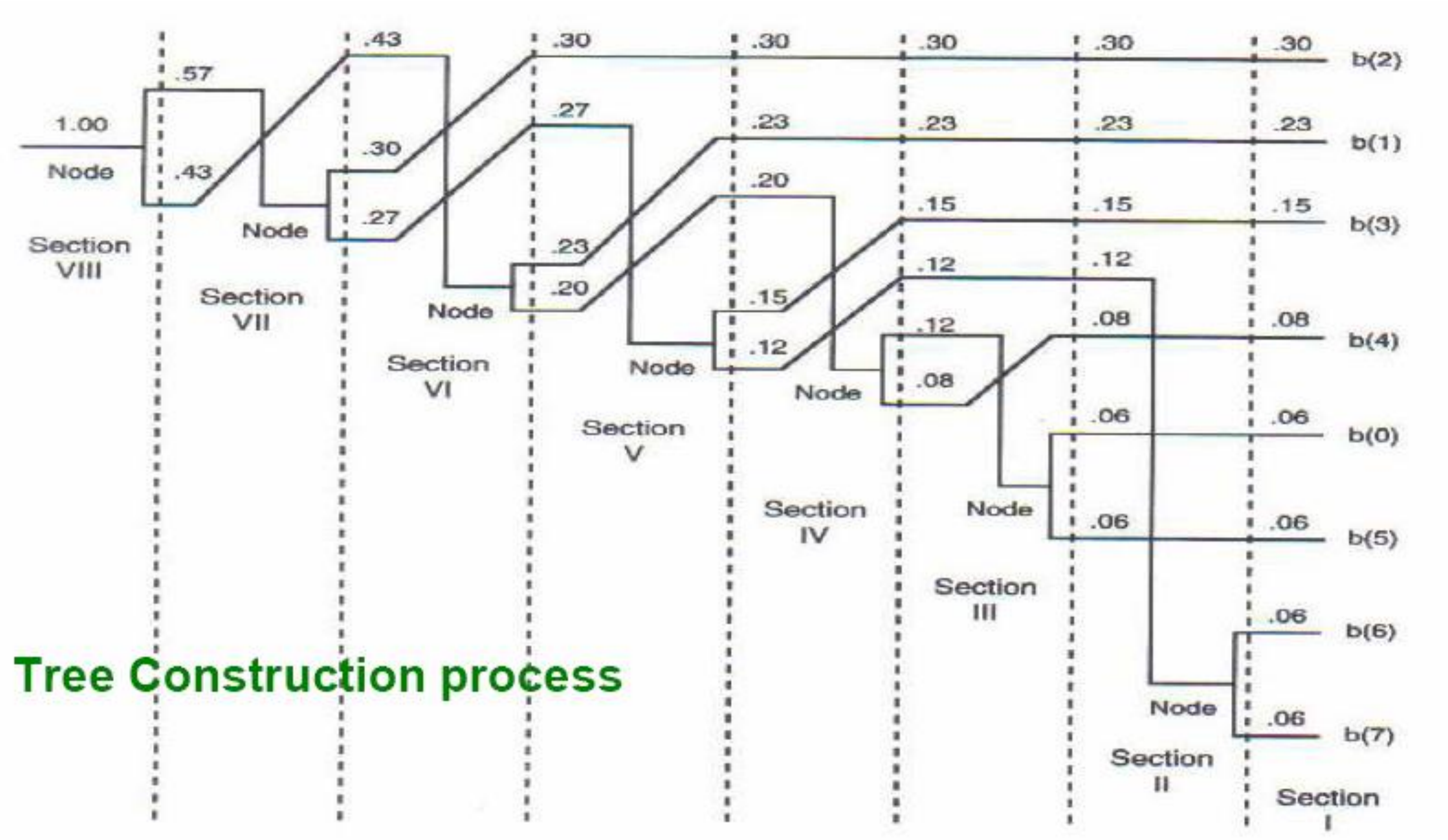
- Were discovered by Huffman in 1952 at MIT.
- These are prefix codes assigned to symbols depending on the probability of the symbols.

Huffman codes

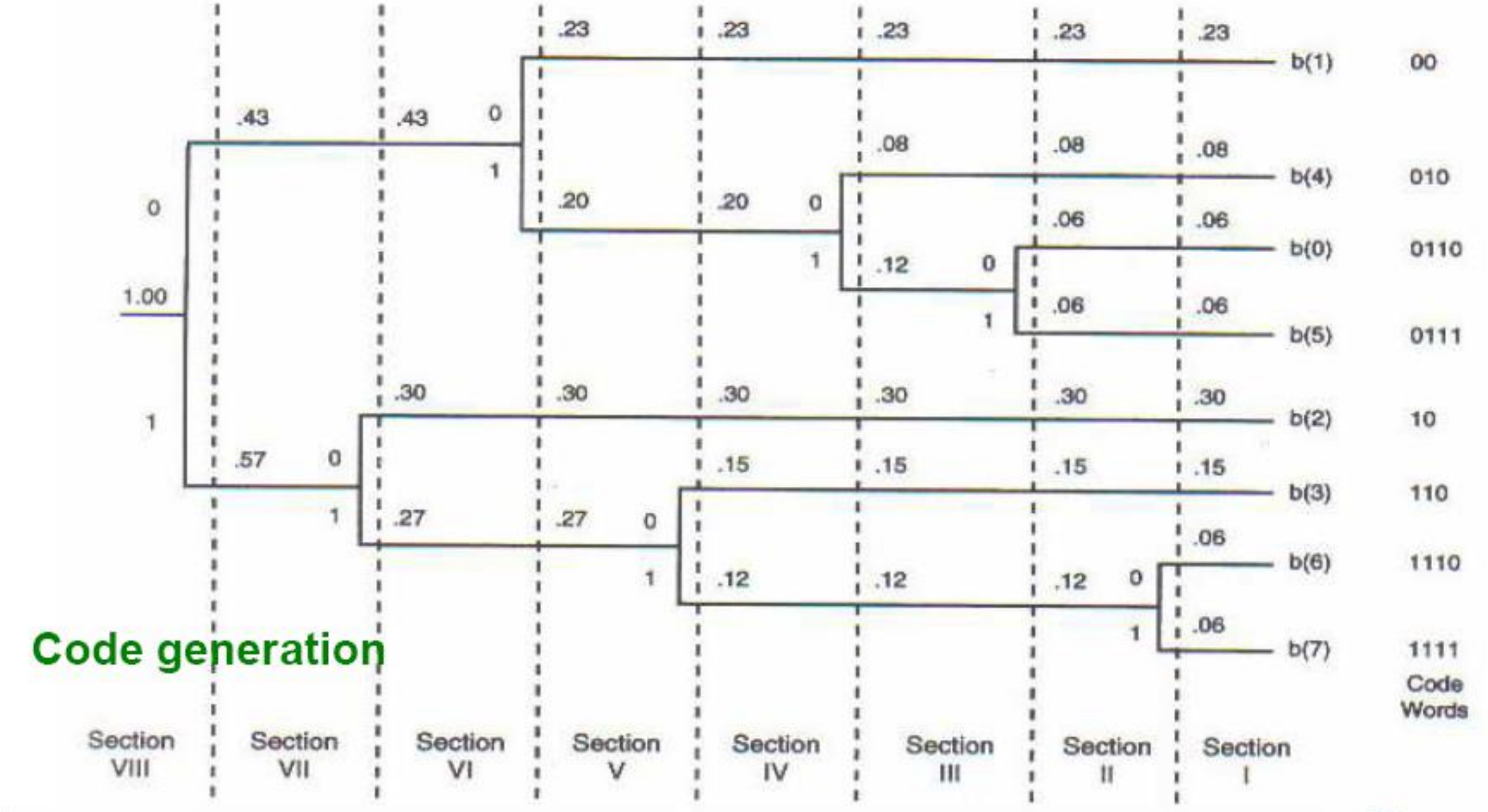
- Generating Huffman codes from a source



Huffman codes



Huffman codes



Huffman codes

Disadvantages of Huffman codes:

- Huffman codes must have integer number of bits.
- Eg: an information source has 2 symbols *a* and *b* with probability p(*a*) = 0.9999, p(*b*) = 0.0001. Huffman assigns each symbol with a code word length 1. Therefore the average code length is 1 wheres the entropy of the source is 0.00147 bits/symbol.
- Probability of symbol must be known before encoding and encoding, the symbol-code mapping musts be agreed between encoder and decoder.
- When the number of symbol is large, the mapping table becomes very large.

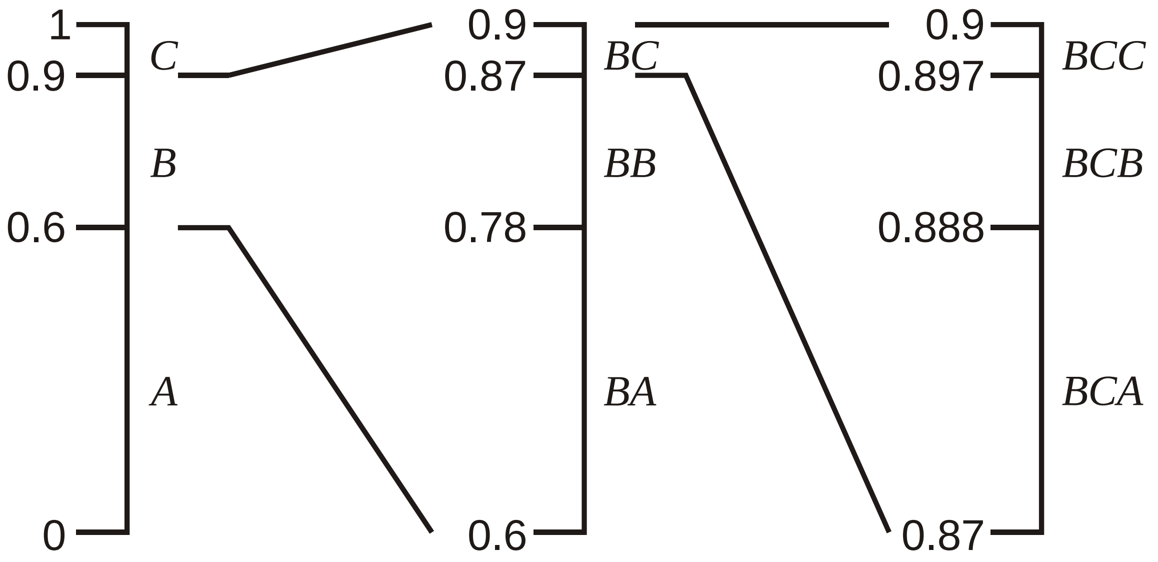
Arithmetic codes

- Arithmetic Coding
  - Code words can be non-integer length, and the probability of symbols can be calculated during encoding and decoding process.
  - Instead of building the mapping between symbol and code word, arithmetic coding finds the mapping from the sequence of symbol to a real number in [0,1].

Arithmetic codes – Example 1

- Given a source {A, B, C}, and a sequence BCA to be encoded.
- p(A) = 0.6, p(B) = 0.3, p(C) = 0.1

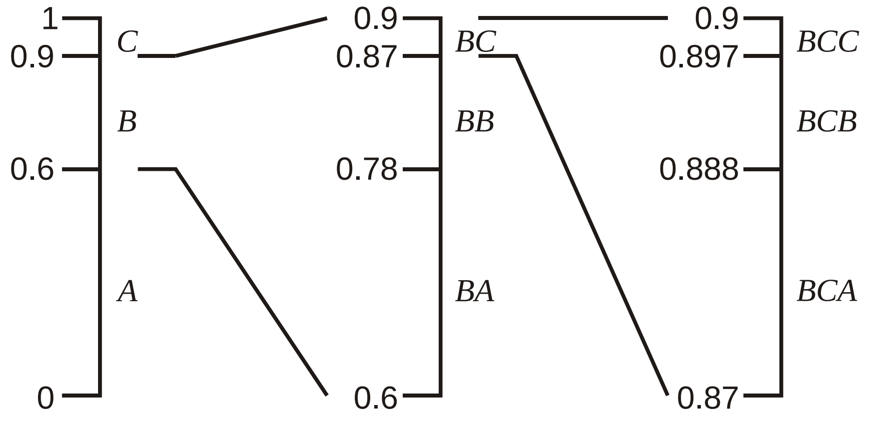
- The sequence starts with A, B, and C is mapped to a half-open interval [0, 0.6), [0.6, 0.9), and [0.9, 1) depending on its probability. In this example interval [0.6, 0.9) is first used for the symbol B.



Arithmetic codes – Example 1

- Given a source {A, B, C}, and a sequence BCA to be encoded.
- p(A) = 0.6, p(B) = 0.3, p(C) = 0.1

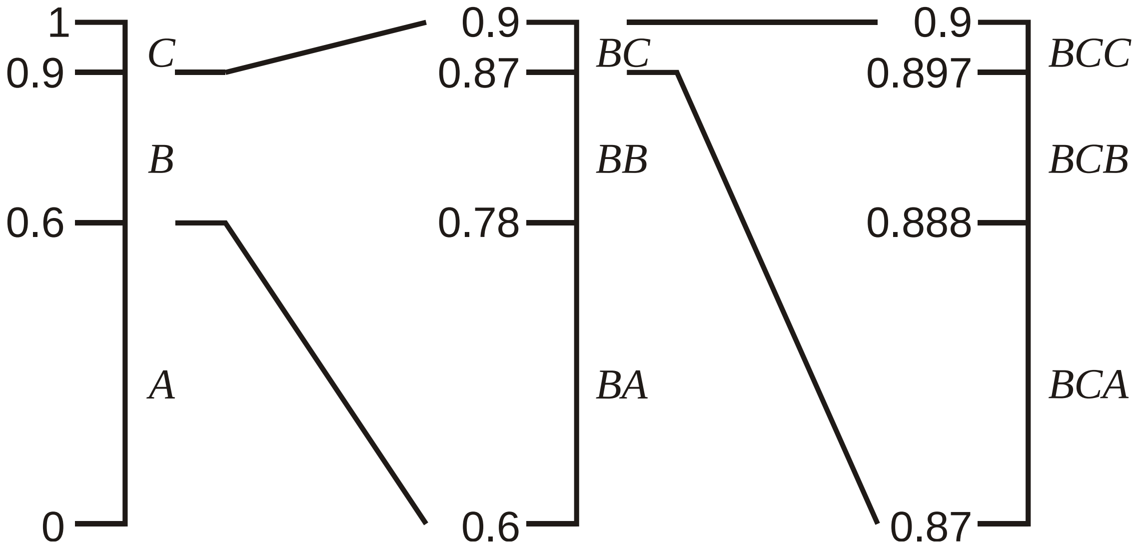
- This interval is further devided to smaller interval [0.6, 0.78), [0.78, 0.87), [0.87, 0.9), for sequences start with BA, BB, BC. Note that: the ratio among the sub-interval is equal to the ratio among probabilities of symbols:
  - 0.18 : 0.09 : 0.03 = 6 : 3 : 1 = p(A) : p(B) : p(C)



Arithmetic codes – Example 1

- Given a source {A, B, C}, and a sequence BCA to be encoded.
- p(A) = 0.6, p(B) = 0.3, p(C) = 0.1

- Repeat this step, interval [0.87, 0.9) is devided to 3 smaller intervals. Sequences start wht BCA are mapped to [0.87, 0.888).



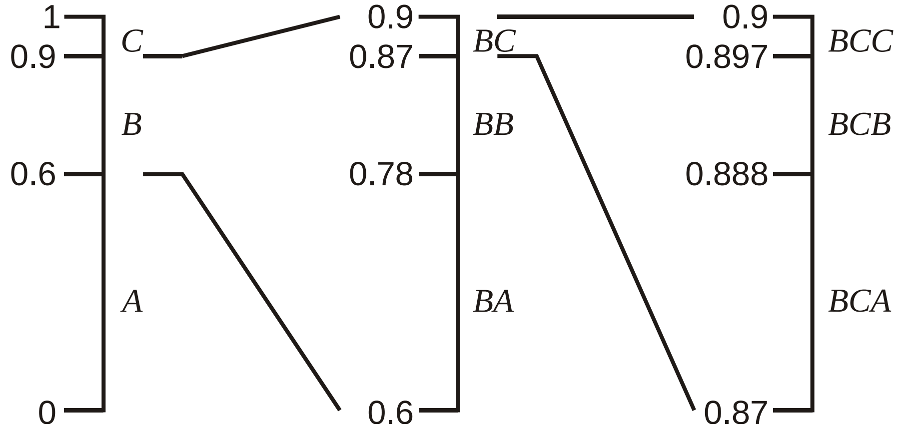
Arithmetic codes – Example 1

- Given a source {A, B, C}, and a sequence BCA to be encoded.
- p(A) = 0.6, p(B) = 0.3, p(C) = 0.1

- 2 ends of interval [0.87, 0.888) is presented by a binary number:

$$0.87 = \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^4} + \frac{1}{2^5} + \cdots = 0.11011\cdots,$$

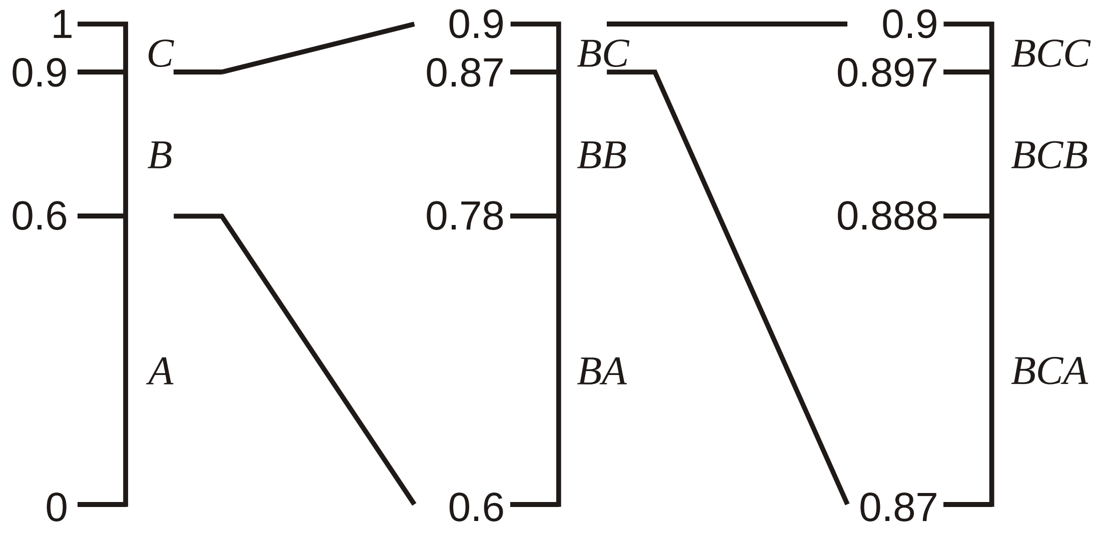
$$0.888 = \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^7} + \cdots = 0.11100\cdots.$$



Arithmetic codes – Example 1

- Given a source {A, B, C}, and a sequence BCA to be encoded.
- p(A) = 0.6, p(B) = 0.3, p(C) = 0.1

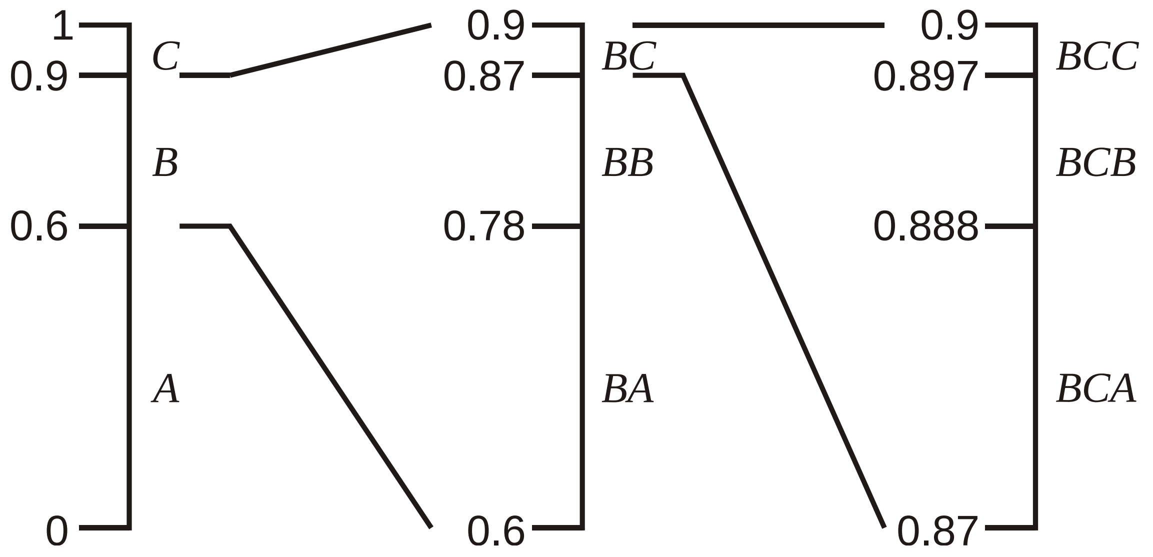
- Encoder can send any number in the interval [0.87, 0.9) to indicate the encoded sequence stars with BCA. For example, it can send 3 bits 111 corresponding to 0.875 in decimal.



Arithmetic codes – Example 1

- Given a source {A, B, C}, and a sequence BCA to be encoded.
- p(A) = 0.6, p(B) = 0.3, p(C) = 0.1

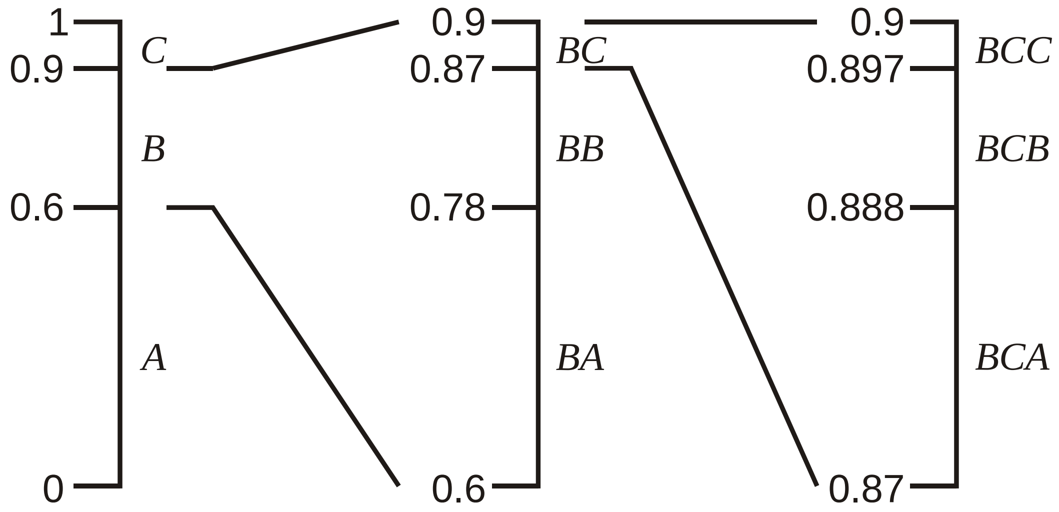
- After receiving 111, decoder processes the interval dividing procedure and know that the decoded sequence starts by BCA. Because it knows that 0.875 is in the interval [0.87, 0.888)



Arithmetic codes – Example 1

- Given a source {A, B, C}, and a sequence BCA to be encoded.
- p(A) = 0.6, p(B) = 0.3, p(C) = 0.1

- However, 0.875 can be the presentation of B, BA, or BAC. So it needs an additional symbol to indicate the end of sequence (EOS).



Arithmetic codes – Example 2

Example 4.3.1:

Consider a three-letter alphabet $\mathcal{A} = \{a_1, a_2, a_3\}$ with $P(a_1) = 0.7$, $P(a_2) = 0.1$, and $P(a_3) = 0.2$. Using the mapping of Equation (4.1), $F_X(1) = 0.7$, $F_X(2) = 0.8$, and $F_X(3) = 1$. This partitions the unit interval as shown in Figure 4.1.

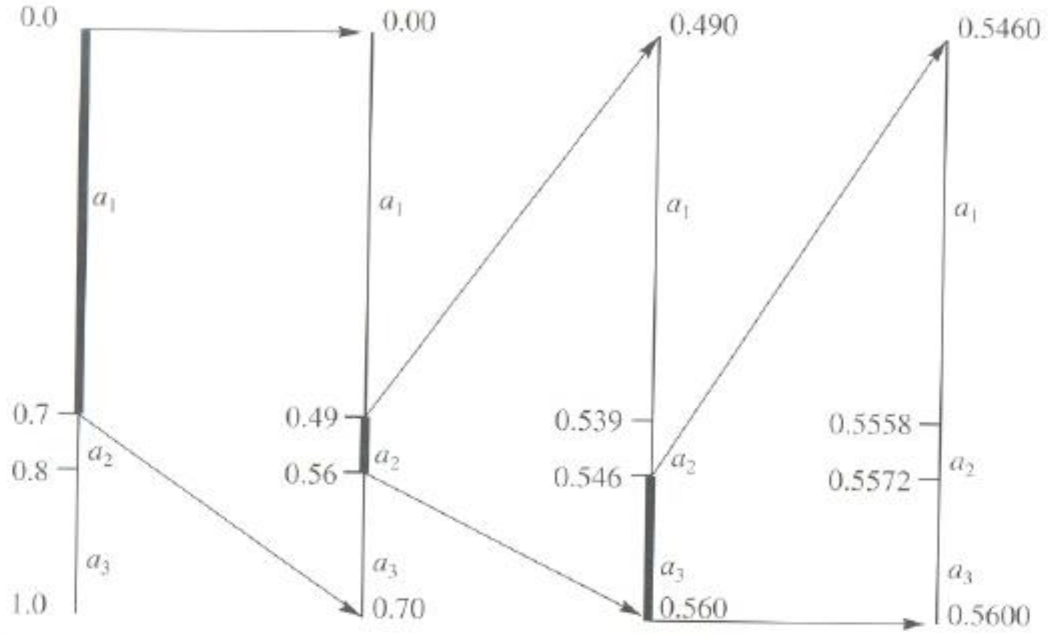


FIGURE 4.1 **Restricting the interval containing the tag for the input sequence $\{a_1, a_2, a_3, \ldots\}$.**

Arthimetic codes

- Binary presentation:

i	2^-i	X	Bit
1	0.5	0.3203125	0
2	0.25	0.3203125	1
3	0.125	0.0703125	0
4	0.0625	0.0703125	1
5	0.03125	0.0078125	0
6	0.015625	0.0078125	0
7	0.0078125	0.0078125	1
8	0.00390625	0	0

Arthimetic codes

- Concerned issues for developing arithmetic codes for industrial applications:
  - Using integer numbers instead of floating point numbers.
  - Subinterval shinks rapidly during encoding process $\rightarrow$ rescale mechanism.
  - Reference to CABAC entropy coding of H.264/AVC.

Arithmetic codes

- Encode the following sequence using arithmetic codes: *badcd*
- Distribution of symbols:

$$X = \begin{cases} a & p(a) = 1/2 \\ b & p(b) = 1/4 \\ c & p(c) = 1/8 \\ d & p(d) = 1/8 \end{cases}$$

Run-length coding

- Similar consecutive symbols are grouped and presented by a code
- Start from a specific code, following by a symbol and the number indicating the number of its copies.
- It is a entropy coding techning used in H.264, CAVLC

Run-length coding

- The simplest case of Run-length coding is to group the similar consecutive symbol and replace by a pair [N][S]
  - N – number of symbols
  - S – symbol
- For example: aaaabbbddaaddcccccccа
- Is is compressed to: 4a3b2d2a2d7c1a

Run-length coding

- It is applied to sub-sequence with the number of similar consecutive symbols more than 3.
- Replace the sub-sequence by [E][S][N-1]
  - E: A specific symbol
  - S: Symbol
  - N: Number of symbols

Predictive coding

- The temporal and spatial dependencies are very high.
- Data are not directly encoded, they are predicted from the their neighbors.
- The difference from data and their predicted values are quantized and encoded by entropy code.

Predictive coding

- Combine with transform to form hybrid-coding methods.
- It is fundamental in multimedia data compression.
- Quantization:
  - Build the relationship between the data lost and coding efficiency.
  - Plays vital role in multimedia communication in public channels.

Predictive coding

- For example: previous pixels are used to predict the following pixels:

90	96(+6)	92(-4)
80	100(+20)	100(+0)
81	82(+1)	80(-2)

- In theoritical speaking: this method remove the dependency between two consecutive pixels and reduce the entropy.

- Better prediction, entropy of the difference are smaller.

Predictive coding

"Lena" image

1-D prediction error of "Lena" image

Entropy = 3.06 bits/pixel

Predictive coding

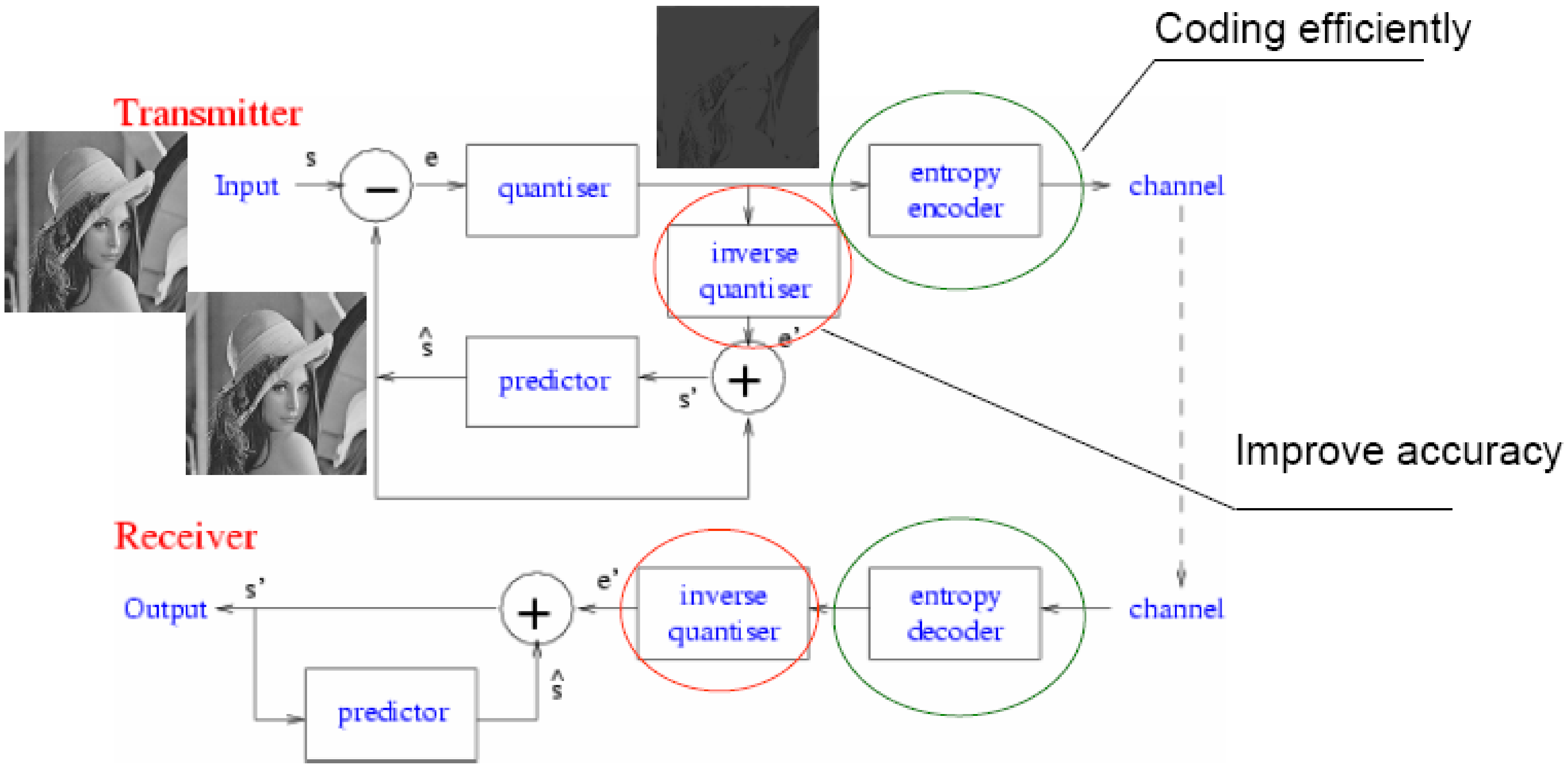
- 2-D relationship

B	C	D
A	X	

$$\hat{x} = k_1 A + k_2 B + k_3 C + K_4 D$$

Entropy = 1.44. bits/pixel

Predictive coding



Quantization

- Data lost during quantization process.
- However, if the quantization is performed carefully in frequency domain, it is not easy to distinguish the difference.

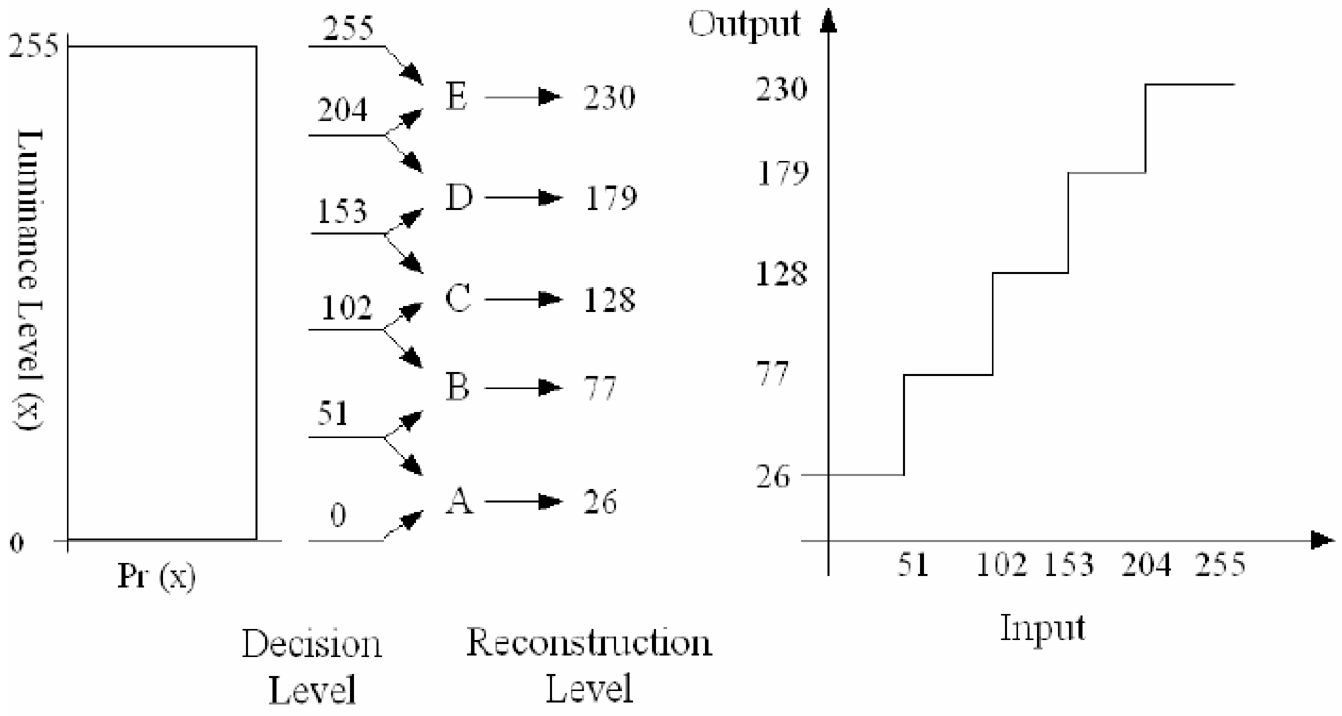
Quantization

Original

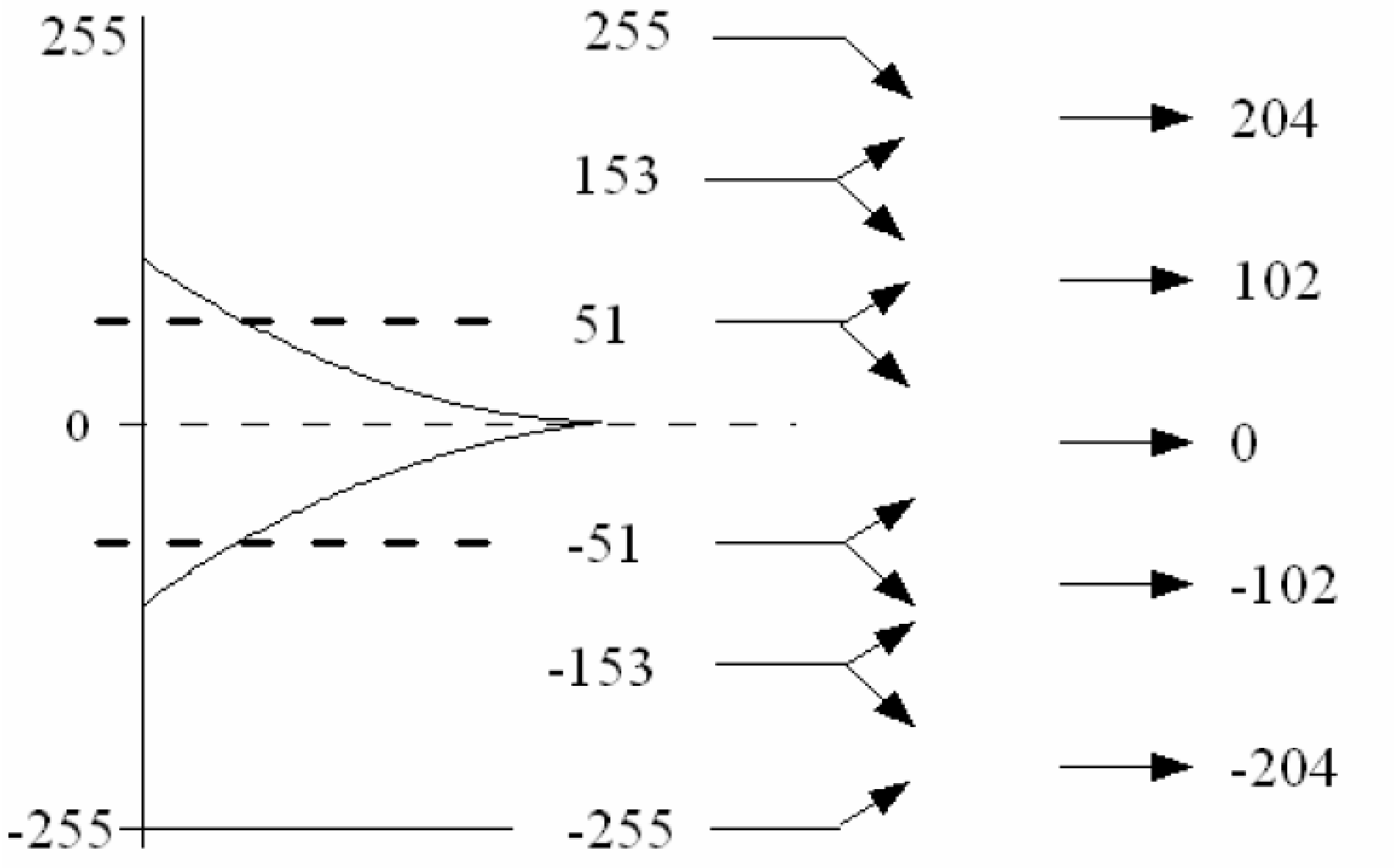
JPEG (compressed 70%)

Quantization

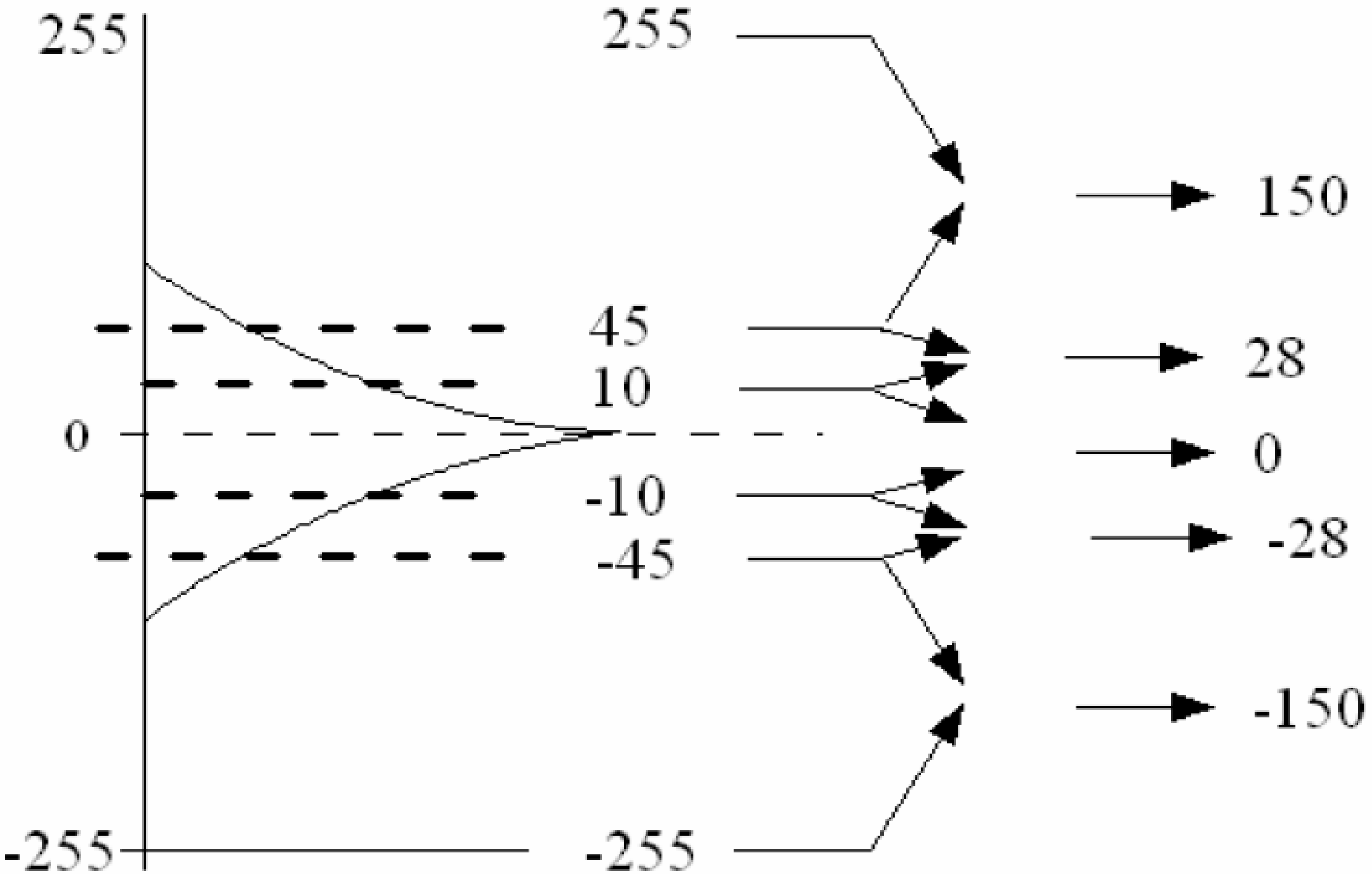
- Uniform quantization: use the same quantization step. Max quantization error is ½ quantization step



Uniform quantization



Non-uniform quantization



Biến đổi DCT 2D

$$\mathbf{A} = \begin{bmatrix} \frac{1}{2}\cos(0) & \frac{1}{2}\cos(0) & \frac{1}{2}\cos(0) & \frac{1}{2}\cos(0) \\ \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{5\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{7\pi}{8}\right) \\ \sqrt{\frac{1}{2}}\cos\left(\frac{2\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{6\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{10\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{14\pi}{8}\right) \\ \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{9\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{15\pi}{8}\right) & \sqrt{\frac{1}{2}}\cos\left(\frac{21\pi}{8}\right) \end{bmatrix}$$

– Được xấp xỉ thành:

$$A = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.653 & 0.271 & -0.271 & -0.653 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.271 & -0.653 & 0.653 & -0.271 \end{bmatrix}$$

Biến đổi DCT 2D

- Ví dụ: Khối X được lấy từ ảnh, và nó được chuyển đổi như sau:

	$j=0$	1	2	3
$i=0$	5	11	8	10
1	9	8	4	12
2	1	10	11	4
3	19	6	15	7

$$\mathbf{Y}' = \mathbf{AX} = \begin{bmatrix} 17 & 17.5 & 19 & 16.5 \\ -6.981 & 2.725 & -6.467 & 4.125 \\ 7 & -0.5 & 4 & 0.5 \\ -9.015 & 2.660 & 2.679 & -4.414 \end{bmatrix}$$

$$\mathbf{Y} = \mathbf{AXA}^{\mathrm{T}} = \begin{bmatrix} 35.0 & -0.079 & -1.5 & 1.115 \\ -3.299 & -4.768 & 0.443 & -9.010 \\ 5.5 & 3.029 & 2.0 & 4.699 \\ -4.045 & -3.010 & -9.384 & -1.232 \end{bmatrix}$$

Biến đổi Hadamard

- Là một dạng biến đổi Fourier đơn giản
- Được sử dụng cho nén frame I trong nén video
- H_m là ma trận $2^m \times 2^m$ phần tử, được định nghĩa đệ quy như sau:

$$H_0 = [1] \qquad H_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad H_m = \frac{1}{\sqrt{2}}\begin{bmatrix} H_m & H_m \\ H_m & -H_m \end{bmatrix}$$

Biến đổi Hadamard

$$H_2 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

$$Y = H_2 \cdot X \cdot H_2^T = H_2 \cdot X \cdot H_2$$

JPEG

- JPEG is a compression standard for still images. It was accepted as an international standard in1992:
  – Developed by Joint Photographic Expert Group of ISO/IEC
  – Is to compress color or monochrome images.
  – Compression rate is about 10:1

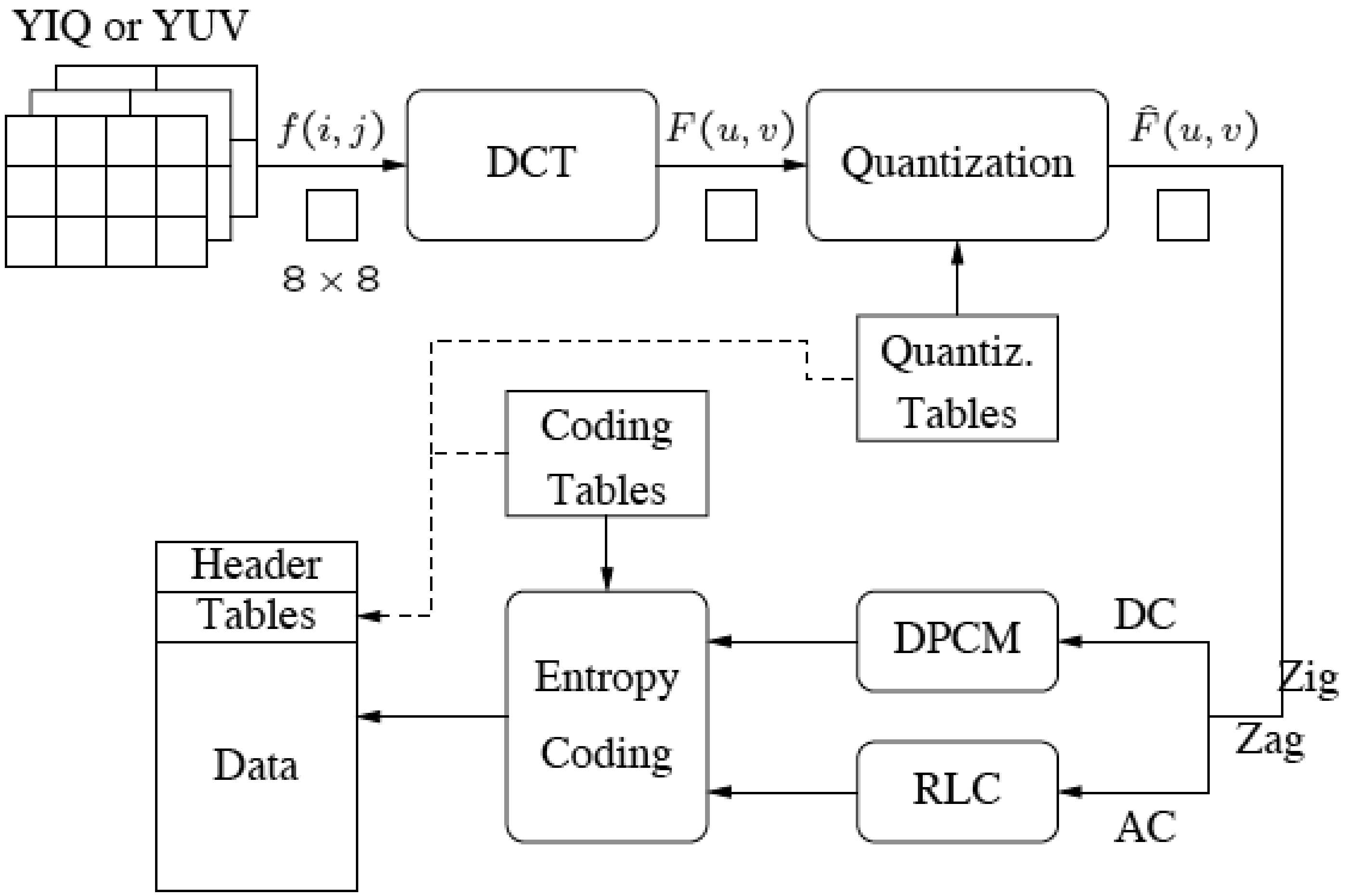
JPEG

- JPEG is lossness compression method:
  - Based on DCT
  - Compression process is independent from:
    - Image size
    - Color model
    - Image complexity
  - The video compression standard based on JPEG is Motion JPEG (MJPEG)

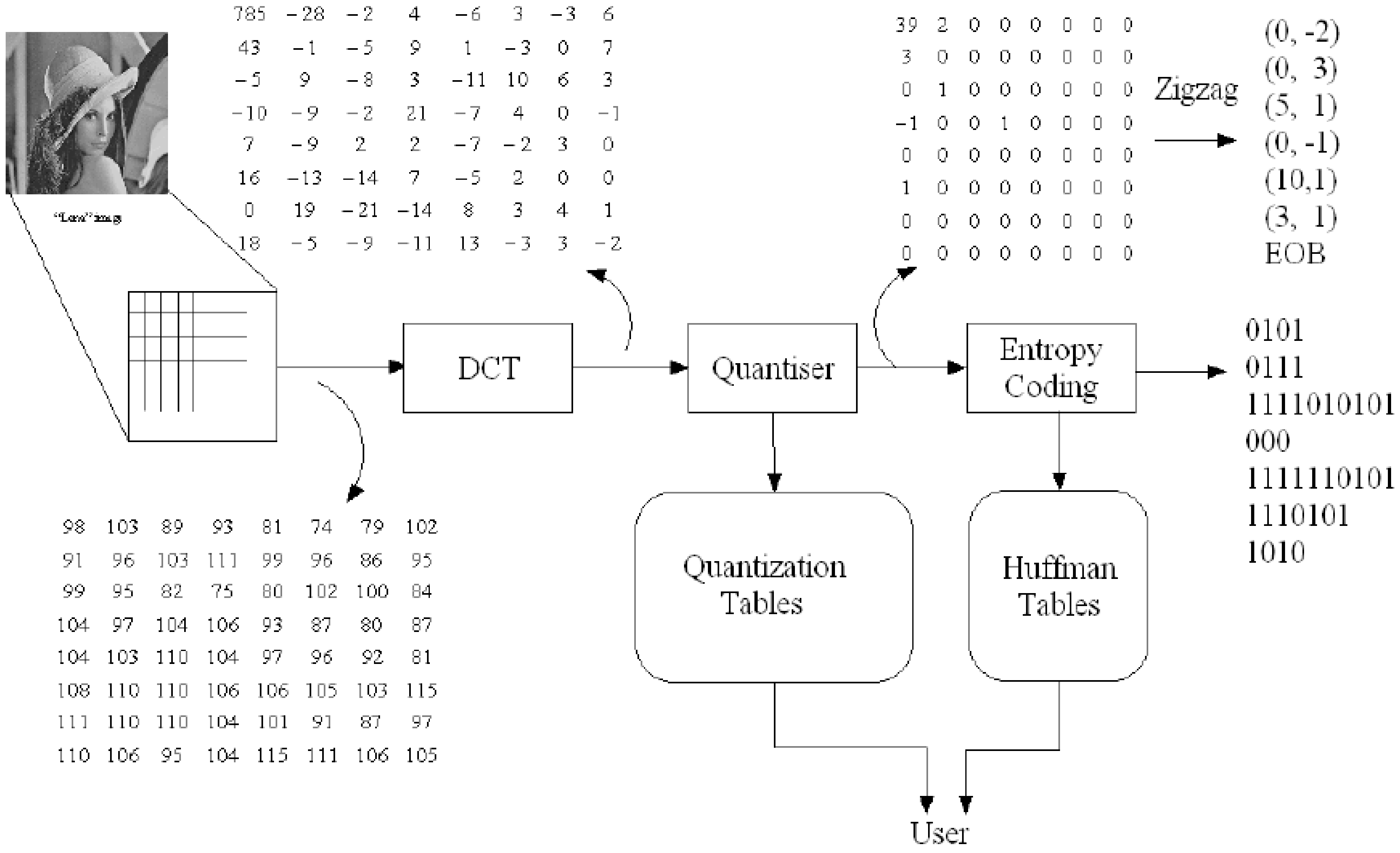
JPEG

- Human eyes sense luminance much better than chrominance:
  - Therefore, JPEG uses sampling of luminance twice as chrominance (4:2:0)

JPEG Encoding



JPEG encoding based on DCT



JPEG encoding

- Encoding step of JPEG:
  - Transform colors from RGB color model to YUV or YIQ.
  - DCT transform for all 8x8 block
  - Quantization
  - For each 8x8 block, scan coefficients with zig-zag pattern and use run-length coding.
  - Entropy coding

DCT transform for each block

- Image is devided into 8x8 blocks
  - 2D DCT transform is applied for all blocks.
  - When compress images with low bitrate. There are blocky discontinuty appearing at the edges of blocks.

Quantization

- Quantization matrix $N_{j,k}$:

For luminance:

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

For chrominance:

$$\begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix}$$

JPEG encoding based on DCT

- Quantization:

$$\hat{Y}_{j,k} = Q[Y_{j,k}] = round\left(\frac{Y_{j,k}}{N_{j,k} \times Q_s}\right)$$

- Q_s is user parameter indicating the quality of compression.

DCT transform in smooth area

An 8 × 8 block from the Y image of 'Lena'

200	202	189	188	189	175	175	175
200	203	198	188	189	182	178	175
203	200	200	195	200	187	185	175
200	200	200	200	197	187	187	187
200	205	200	200	195	188	187	175
200	200	200	200	200	190	187	175
205	200	199	200	191	187	187	175
210	200	200	200	188	185	187	186

$f(i, j)$

515	65	-12	4	1	2	-8	5
-16	3	2	0	0	-11	-2	3
-12	6	11	-1	3	0	1	-2
-8	3	-4	2	-2	-3	-5	-2
0	-2	7	-5	4	0	-1	-4
0	-3	-1	0	4	1	-1	0
3	-2	-3	3	3	-1	-1	3
-2	5	-2	4	-2	2	-3	0

$F(u, v)$

Fig. 9.2: JPEG compression for a smooth image block.

Quantization

$$\hat{F}(u,v) = \begin{matrix} 32 & 6 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

$$\tilde{F}(u,v) = \begin{matrix} 512 & 66 & -10 & 0 & 0 & 0 & 0 & 0 \\ -12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -14 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\ -14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Inverse DCT transform

199	196	191	186	182	178	177	176
201	199	196	192	188	183	180	178
203	203	202	200	195	189	183	180
202	203	204	203	198	191	183	179
200	201	202	201	196	189	182	177
200	200	199	197	192	186	181	177
204	202	199	195	190	186	183	181
207	204	200	194	190	187	185	184

$\tilde{f}(i,j)$

1	6	-2	2	7	-3	-2	-1
-1	4	2	-4	1	-1	-2	-3
0	-3	-2	-5	5	-2	2	-5
-2	-3	-4	-3	-1	-4	4	8
0	4	-2	-1	-1	-1	5	-2
0	0	1	3	8	4	6	-2
1	-2	0	5	1	1	4	-6
3	-4	0	6	-2	-2	2	2

$\epsilon(i,j) = f(i,j) - \tilde{f}(i,j)$

DCT transform in detailed area

Another 8×8 block from the Y image of 'Lena'

70	70	100	70	87	87	150	187
85	100	96	79	87	154	87	113
100	85	116	79	70	87	86	196
136	69	87	200	79	71	117	96
161	70	87	200	103	71	96	113
161	123	147	133	113	113	85	161
146	147	175	100	103	103	163	187
156	146	189	70	113	161	163	197

$f(i,j)$

-80	-40	89	-73	44	32	53	-3
-135	-59	-26	6	14	-3	-13	-28
47	-76	66	-3	-108	-78	33	59
-2	10	-18	0	33	11	-21	1
-1	-9	-22	8	32	65	-36	-1
5	-20	28	-46	3	24	-30	24
6	-20	37	-28	12	-35	33	17
-5	-23	33	-30	17	-5	-4	20

$F(u,v)$

Quantization and inverse quantization

$$\begin{matrix} -5 & -4 & 9 & -5 & 2 & 1 & 1 & 0 \\ -11 & -5 & -2 & 0 & 1 & 0 & 0 & -1 \\ 3 & -6 & 4 & 0 & -3 & -1 & 0 & 1 \\ 0 & 1 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

$\hat{F}(u,v)$

$$\begin{matrix} -80 & -44 & 90 & -80 & 48 & 40 & 51 & 0 \\ -132 & -60 & -28 & 0 & 26 & 0 & 0 & -55 \\ 42 & -78 & 64 & 0 & -120 & -57 & 0 & 56 \\ 0 & 17 & -22 & 0 & 51 & 0 & 0 & 0 \\ 0 & 0 & -37 & 0 & 0 & 109 & 0 & 0 \\ 0 & -35 & 55 & -64 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

$\tilde{F}(u,v)$

Inverse DCT and the errors

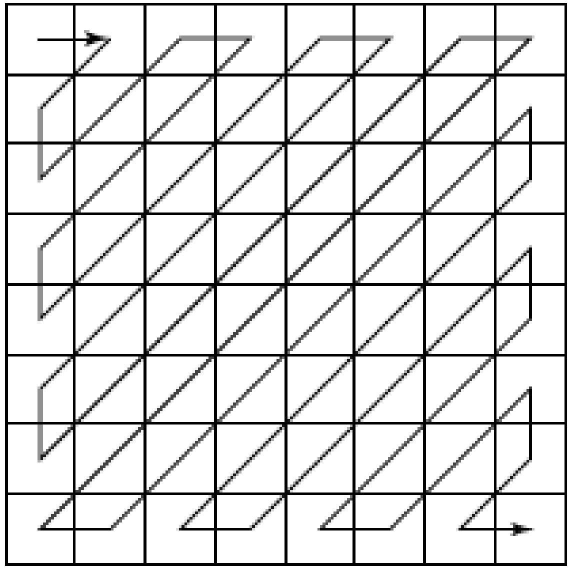
70	60	106	94	62	103	146	176
85	101	85	75	102	127	93	144
98	99	92	102	74	98	89	167
132	53	111	180	55	70	106	145
173	57	114	207	111	89	84	90
164	123	131	135	133	92	85	162
141	159	169	73	106	101	149	224
150	141	195	79	107	147	210	153

$\tilde{f}(i,j)$

0	10	-6	-24	25	-16	4	11
0	-1	11	4	-15	27	-6	-31
2	-14	24	-23	-4	-11	-3	29
4	16	-24	20	24	1	11	-49
-12	13	-27	-7	-8	-18	12	23
-3	0	16	-2	-20	21	0	-1
5	-12	6	27	-3	2	14	-37
6	5	-6	-9	6	14	-47	44

$\epsilon(i,j) = f(i,j) - \tilde{f}(i,j)$

Zig-zag scan



32	6	-1	0	0	0	0	0
-1	0	0	0	0	0	0	0
-1	0	1	0	0	0	0	0
-1	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	0

(**32**, 6, -1, -1, 0, -1, 0, 0, 0, -1, 0, 0, 1, 0, 0, ..., 0)

Hệ số DC

Các hệ số AC

Run-length coding for AC coefficients

- Run-length coding represent each AC coefficient (after zig-zag scanning) as a pair (Runlength, value).
  – Runlength: number of zeros before a non-zero coef.
  – Value: value of non-zero number.

- For example:
  – Sequence: 32, 6, -1, -1, 0, -1, 0, 0, 0, -1, 0, 0, 1, 0, 0, …, 0
  – After run-length coding: (0,6) (0,-1) (0,-1) (1,-1) (3,-1) (2,1) (0,0). Skip the first coefficient (DC).

Coding for DC coefficient

- Each block has a DC coeffficient
- DC values vary not much in neighbor blocks. Use zig-zag scan to read DC values block by block.
- Use Differential Pulse Code Modulation (DPCM) for coding DC values:
  - The current DC value is predicted from the previous one.
  - For example, first 5 DC values: 150, 155, 149, 152, 144.
  - DPCM codes are:150, 5, -6, 3, -8.

Coding for DC coefficient

- Run-length coded values of AC and DC coefficients are entropy coded:
  – Each value is presented by a pair of symbols (Size, amplitude)
    - Size: Number of bits to present the value and coded by Huffman codes.
    - Amplitute has uniform distribution, then no compression needed.