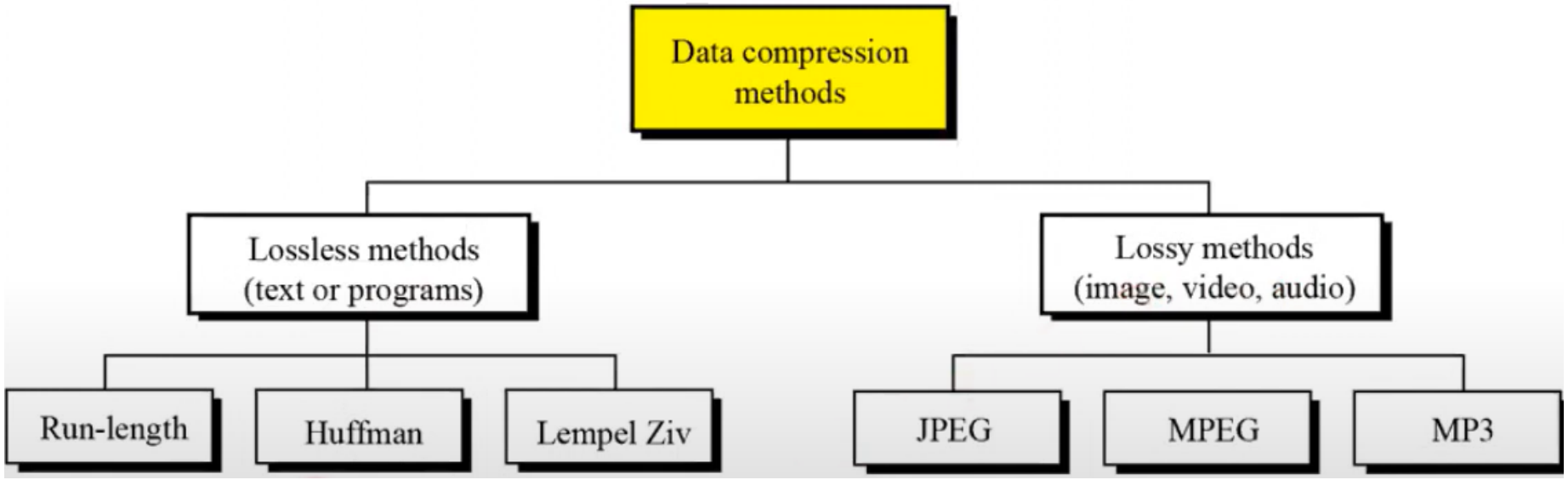
Xử lý ảnh và Thị giác máy tính

Chương 4. Nén ảnh và phân đoạn ảnh

- Data compression
  - Lossless methods
  - Lossy methods



PHẦN 1. NÉN ẢNH

1. Nén dữ liệu không tổn thất (lossless)

- Dữ liệu được bảo toàn, nhưng tỉ lệ nén thấp

Các kiểu mã hóa thông dụng trong lossless compression

- Run-length coding (RLC)
- Huffman coding
- Lempel-Ziv

Run-length coding

- Tận dụng tính chất lặp lại giá trị trong ảnh

Nguyên tắc:

- Dùng cặp giá trị (độ dài chuỗi, giá trị) để thay thế cho loạt dữ liệu lặp lại

5 5 5 5 5 5 7 7 7 7 10 10 10 → (6, 5) (4, 7) (3, 10)

* Chú ý nếu chiều dài chuỗi lặp > 255 thì 1 byte sẽ không đủ, khi đó tách ra hai chuỗi, một chuỗi có chiều dài 255, chuỗi kia chứa phần còn lại

Nhược điểm

- RLC chỉ đạt được hiệu quả khi chiều dài chuỗi lặp lớn hơn một ngưỡng nhất định, khi dữ liệu đơn lẻ xuất hiện nhiều RLC sẽ giảm hiệu quả

Ví dụ

2 4 6 8 10 12 → (1, 2) (1, 4) (1, 6) (1, 8) (1, 10) (1, 12)

Tỉ lệ nén: 6/12 = 0.5

Run-length coding cải tiến (1)

- Chỉ mã hoá độ dài loạt dữ liệu lặp lại
- Thêm kí tự tiền tố vào trước độ dài loạt

Ví dụ:

4 7 3 7 7 7 7 7 7 7 7 9 9 9 9 9 9

→4 7 3 + 8 7 + 6 9

Tỉ lệ nén: 17/9 = 1.89

Run-length coding cải tiến (2)

- Kiểm tra mức độ tương quan giữa các dữ liệu kề nhau

Ví dụ:

5 7 9 11 13 18 28 38 48 58 55 60 65 70 75 80 85 →

5 2 2 2 2 5 10 10 10 10 -3 5 5 5 5 5 5 →

(5, 1) (4, 2) (5,1) (4, 10) (1, -3) (6, 5)

Tỉ số nén 17/12 = 1.42 (nếu 1 số tương ứng 1 byte)

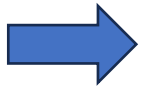
Thuật toán RLC cải tiến (3)

- Tiến hành duyệt trên từng hàng cho đến khi kết thúc vùng dữ liệu ảnh, chú ý những kí hiệu xuống dòng (hay kết thúc dòng), kết thúc ảnh Bitmap
- Khi gặp loạt có độ dài > 3 thì nhảy đến chế độ nén ngược lại nhảy đến chế độ không nén tuy nhiên nếu loạt > 255 thì sẽ tách ra chỉ mã ≤ 255 sau đó mã tiếp phần còn lại
- Kết thúc khi gặp kí hiệu kết thúc bitmap (end-of bitmap)

RLC trong ảnh

- Tiến hành duyệt trên từng hàng cho đến khi kết thúc vùng dữ liệu ảnh

Row 1	**255**	**255**	**255**	**0**	**0**	**255**	**255**	**255**
Row 2	**0**	**0**	**255**	**255**	**255**	**255**	**255**	**255**
Row 3	**255**	**255**	**0**	**0**	**0**	**255**	**255**	**255**



Row 1: (3, 255) (2, 0) (3, 255)
Row 2: (2, 0) (6, 255)
Row 3: (2, 255) (3, 0) (3, 255)

Mã hóa Huffman (encoding)

Thuật toán

Bước 1. Liệt kê các phần tử dưới dạng tần suất xuất hiện, sắp xếp bảng dưới dạng giảm dần

Bước 2. Kết hợp 2 phần tử có tần suất thấp nhất lại thành 1 phần tử mới. Cập nhật lại bảng (loại bỏ 2 phần tử đó)

Bước 3. Lặp lại bước 2 cho đến khi bảng chỉ còn 1 phần tử (tạo cây Huffman)

Bước 4. Từ gốc (phần tử có tần suất lớn nhất) đi xuống, mỗi lần xuống bên phải thêm 1 bit “1”, bên trái thêm bit “0”

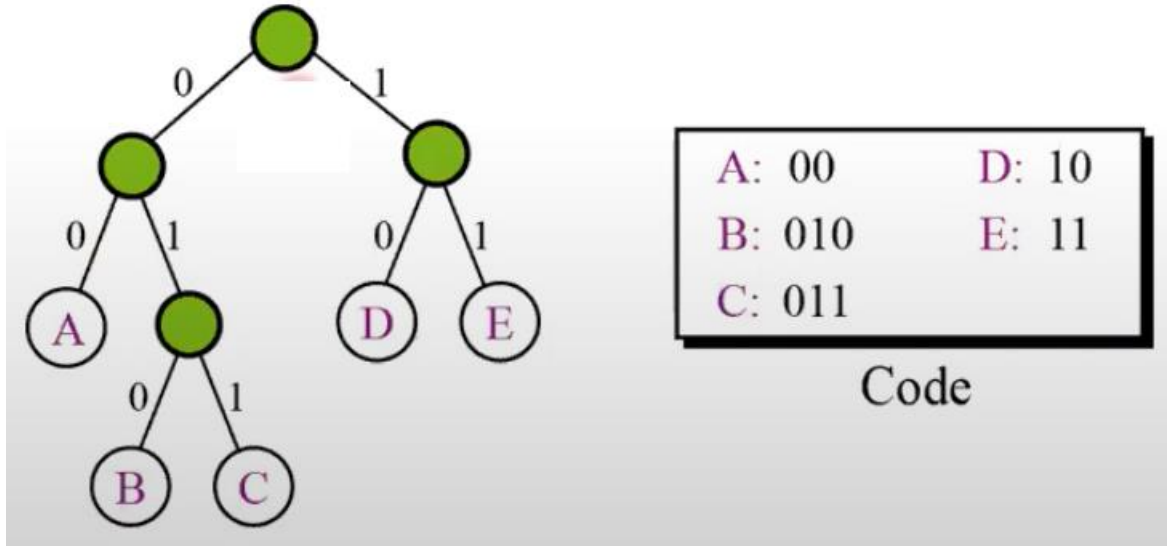
Chú ý

- Đánh dấu 0 hay 1 cho nhánh sẽ không ảnh hưởng đến việc giải mã
- Chúng ta chỉ quan tâm đến số lượng bit giảm xuống bao nhiêu, tức số bit như nhau dù đánh dấu mã khác nhau

Ví dụ

- Xây dựng cây Huffman từ bảng thống kê tần suất xuất hiện sau:

Character	A	B	C	D	E
Frequency	17	12	12	27	32



Ví dụ

Mã hóa dùng mã Huffman

- AABCBAD

Ví dụ

Mã hóa dùng mã Huffman

- AAAAAABCCCCCCDDEEEEE

- AAAAABCDCEEEEEF

Giải nén (decoding)

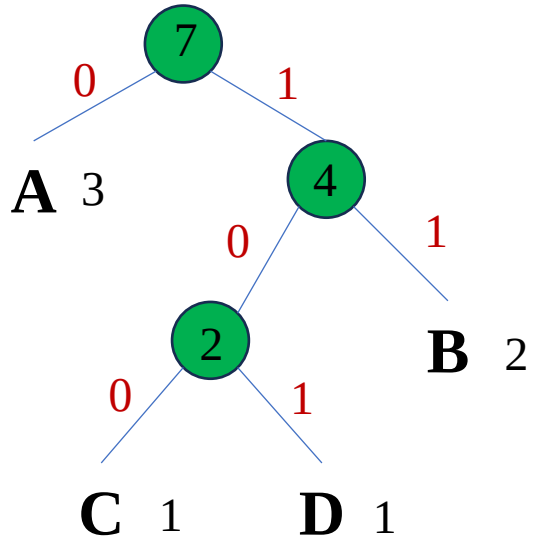
- Trong phương pháp mã Huffman mã của ký tự là duy nhất và không mã nào là phần bắt đầu của mã trước

- Việc giải mã chắc chắn phải sử dụng cây nhị phân giống như trong mã hoá. Để giải mã được yêu cầu phải sử dụng theo đúng tiêu chuẩn nhất định

Ví dụ Mã hóa và giải mã Huffman

Mã hóa dùng mã Huffman

- AABCBAD



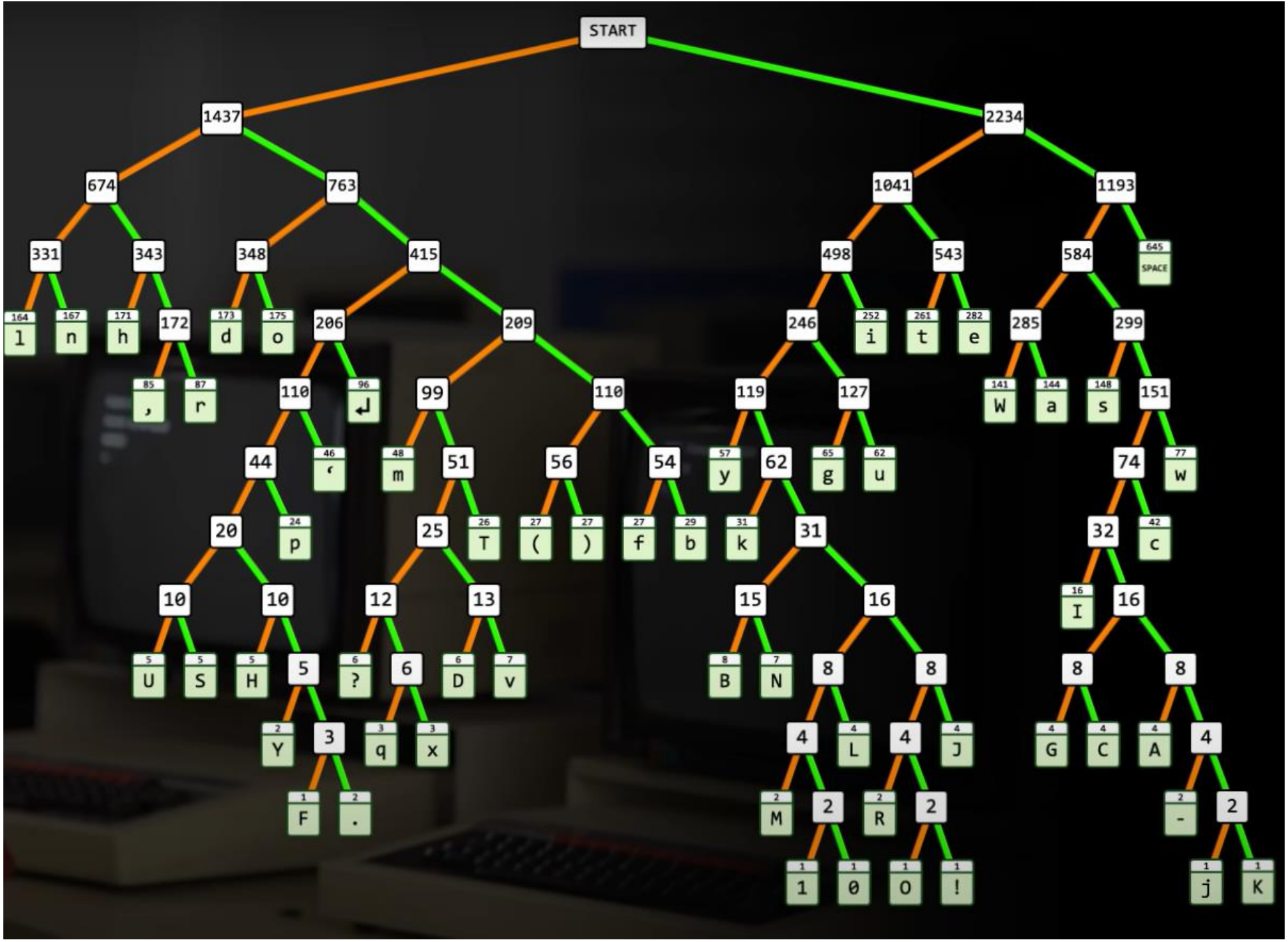
Code

A	**0**
B	**11**
C	**100**
D	**101**

AABCBAD

0 0 11 100 11 0 101 Huffman code

AABCBAD


START
1437
2234
674
763
1041
1193
331
343
348
415
498
543
584
645
SPACE
164
l
167
n
171
h
172
173
d
175
o
206
209
246
252
i
261
t
282
e
285
299
85
,
87
r
110
96
99
110
119
127
141
W
144
a
148
s
151
44
46
48
m
51
56
54
57
y
62
65
g
62
74
77
w
20
24
p
25
26
T
27
(
27
)
27
f
29
b
31
k
31
32
42
c
10
10
12
13
15
16
16
I
16
5
U
5
S
5
H
5
6
?
6
6
D
7
v
8
B
7
N
8
8
8
8
2
Y
3
3
q
3
x
4
4
L
4
4
J
4
G
4
C
4
A
4
1
F
2
.
2
M
2
2
R
2
2
-
2
1
1
1
0
1
O
1
!
1
j
1
K

Mã hóa Huffman trong xử lý ảnh

- Thực hiện mã hóa ảnh sau dùng thuật toán Huffman. Biết ảnh được chia than các khối 2x2 để

$$I = \left[\begin{array}{cc:cc:cc:cc:cc:cc:cc:cc} 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ \hdashline 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ \hdashline 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right]$$

Mã hóa LWZ

- Lempel, Ziv, Welch
- Dùng một bộ từ điển tự động cập nhật và đánh số khi có mã mới xuất hiện
- Mã từ 0 đến 255 miêu tả 1 ký tự 8-bit
- Mã từ 256 đến 4095 được tạo và cập nhật khi xuất hiện chuỗi ký tự mới
- Tại mỗi bước, một byte được nhập vào 1 chuỗi cho đến khi chuỗi đó không tồn tại trong từ điển

Thuật toán mã hóa (encoding)

Bước 1. Khởi tạo từ điển chứa tất cả chuỗi có 1 ký tự

Bước 2. Tìm chuỗi W dài nhất trong từ điển đối chiếu với dữ liệu nhập hiện tại

Bước 3. Xuất vị trí từ điển cho W ra file output và xóa W khỏi dữ liệu nhập.

Bước 4. Thêm W và ký tự tiếp theo trong dữ liệu nhập vào từ điển

Bước 5. Đến bước 2.

Ví dụ

BABAABAAA

output	Dictionary	
	Code Word	String
66	256	BA
65	257	AB
256	258	BAA
257	259	ABA
65	260	AA
260		

<66><65><256><257><65><260>

Ví dụ

BABAABRRRA

output	Dictionary	
	Code Word	String
66	256	BA
65	257	AB
256	258	BAA
257	259	ABR
82	260	RR
260	261	RRA
65		

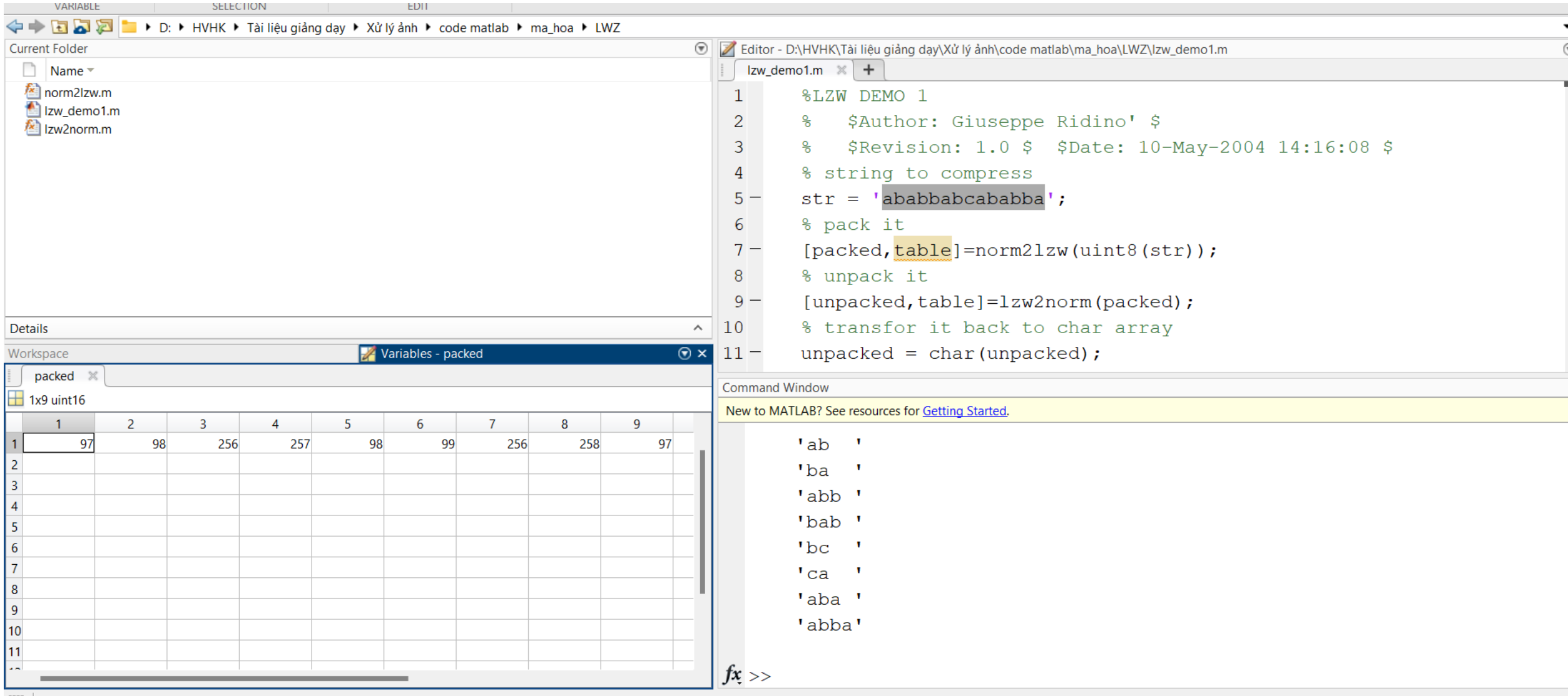
<66><65><256><257><82><260><65>

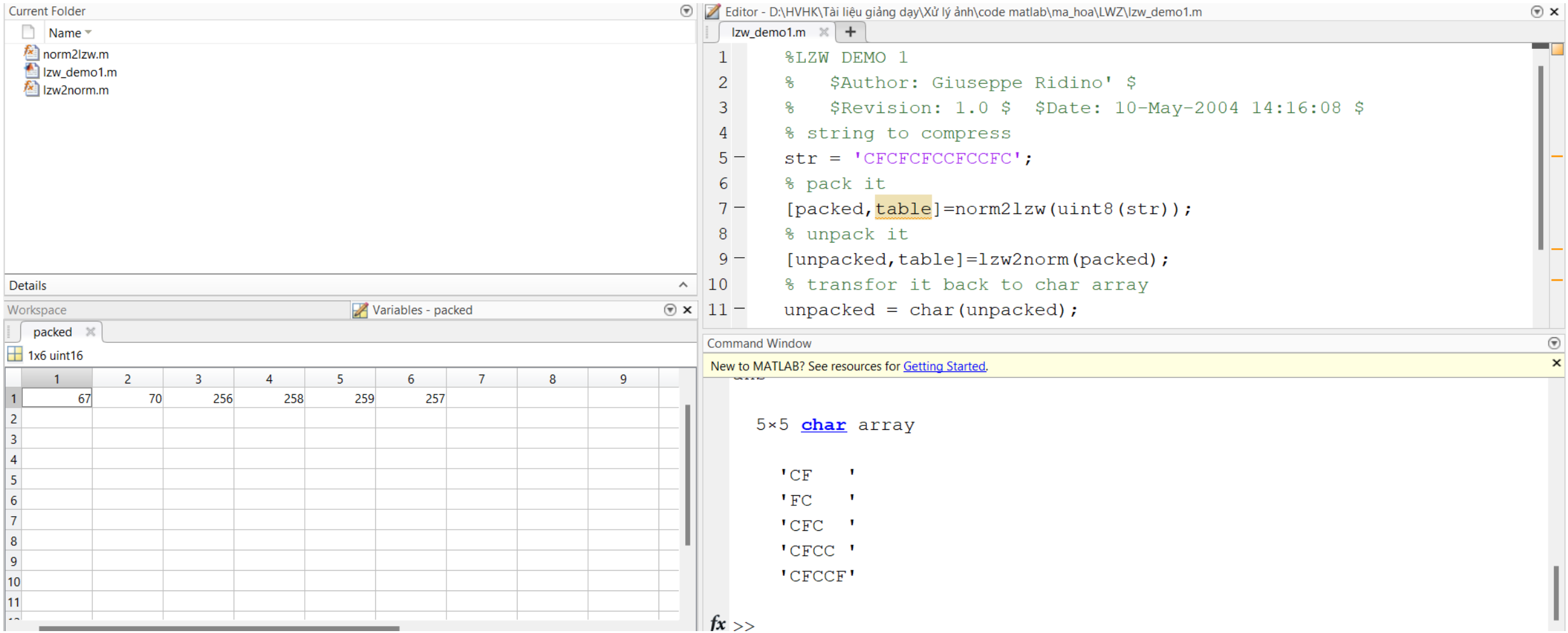
Bài tập

Dùng mã LZW để nén các chuỗi ký tự sau

- CFCFCFCCFCCFC
- !BAN!BAA!BAA!BAR!RA

MÃ ASCII:
! → 33
N → 78
R → 82


VARIABLE
SELECTION
EDIT
D: ▸ HVHK ▸ Tài liệu giảng dạy ▸ Xử lý ảnh ▸ code matlab ▸ ma_hoa ▸ LWZ
Current Folder
Name
norm2lzw.m
lzw_demo1.m
lzw2norm.m
Details
Workspace
Variables - packed
packed
1x9 uint16
1 2 3 4 5 6 7 8 9
1 97 98 256 257 98 99 256 258 97
Editor - D:\HVHK\Tài liệu giảng dạy\Xử lý ảnh\code matlab\ma_hoa\LWZ\lzw_demo1.m
lzw_demo1.m
1 %LZW DEMO 1
2 % $Author: Giuseppe Ridino' $
3 % $Revision: 1.0 $ $Date: 10-May-2004 14:16:08 $
4 % string to compress
5 str = 'ababbabcababba';
6 % pack it
7 [packed,table]=norm2lzw(uint8(str));
8 % unpack it
9 [unpacked,table]=lzw2norm(packed);
10 % transfor it back to char array
11 unpacked = char(unpacked);
Command Window
New to MATLAB? See resources for Getting Started.
'ab '
'ba '
'abb '
'bab '
'bc '
'ca '
'aba '
'abba'
fx >>


Current Folder
Name
norm2lzw.m
lzw_demo1.m
lzw2norm.m
Details
Workspace
Variables - packed
packed
1x6 uint16
1 2 3 4 5 6 7 8 9
1 67 70 256 258 259 257
Editor - D:\HVHK\Tài liệu giảng dạy\Xử lý ảnh\code matlab\ma_hoa\LWZ\lzw_demo1.m
lzw_demo1.m
1 %LZW DEMO 1
2 % $Author: Giuseppe Ridino' $
3 % $Revision: 1.0 $ $Date: 10-May-2004 14:16:08 $
4 % string to compress
5 str = 'CFCFCFCCFCCFC';
6 % pack it
7 [packed,table]=norm2lzw(uint8(str));
8 % unpack it
9 [unpacked,table]=lzw2norm(packed);
10 % transfor it back to char array
11 unpacked = char(unpacked);
Command Window
New to MATLAB? See resources for Getting Started.
5×5 char array
'CF '
'FC '
'CFC '
'CFCC '
'CFCCF'
fx >>

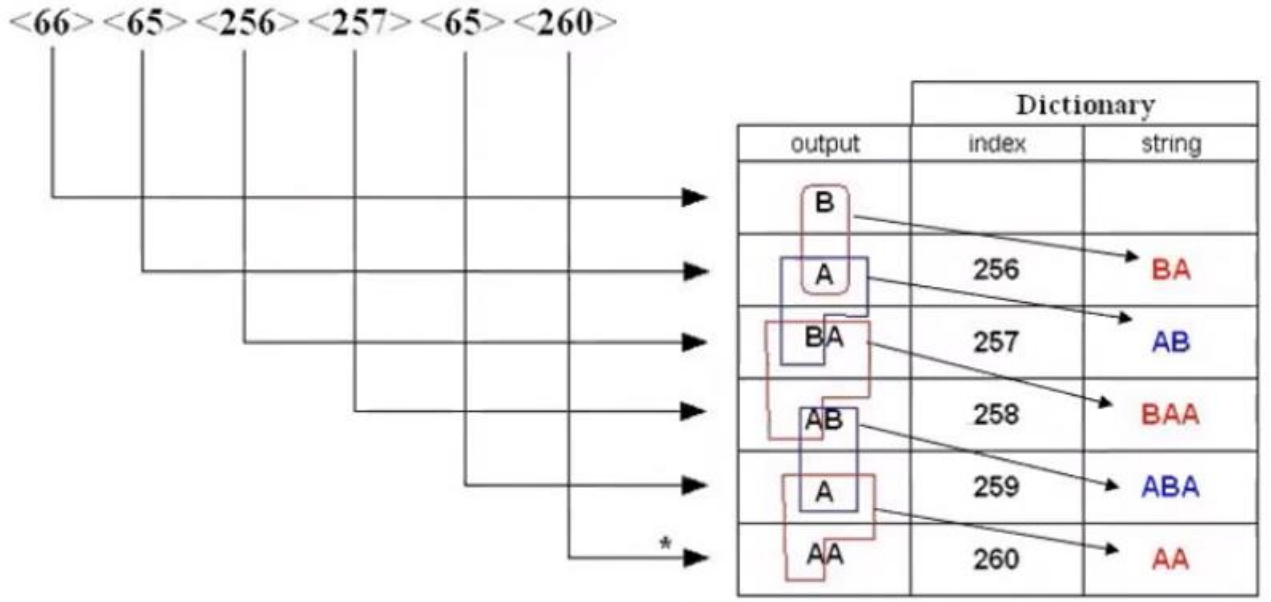
Thuật toán giải nén (decoding)

- Đọc giá trị từ dữ liệu nhập đã mã hóa và xuất ra chuỗi tương ứng từ từ điển đã được khởi tạo
- Tại cùng 1 thời điểm nó thu được giá trị tiếp theo từ dữ liệu nhập,thêm vào từ điển xích chuỗi của chuỗi xuất. Và ký tự đầu tiên của chuỗi nhận được khi mã hóa ký tự tiếp theo
- Sau đó trình giải nén xử lý giá trị nhập tiếp theo, quá trình lặp cho đến khi dữ liệu nhập không còn, tại thời điểm giá trị nhập cuối cùng được mã hóa không còn bất kỳ giá trị nào thêm vào từ điển

Ví dụ

Giải nén dữ liệu sau dùng LZW

<66><65><256><257><65><260>



1. **66** is in Dictionary; output **string(66)** i.e. **B**
2. **65** is in Dictionary; output **string(65)** i.e. **A**, insert **BA**
3. **256** is in Dictionary; output **string(256)** i.e. **BA**, insert **AB**
4. **257** is in Dictionary; output **string(257)** i.e. **AB**, insert **BAA**
5. **65** is in Dictionary; output **string(65)** i.e. **A**, insert **ABA**
6. **260** is **not** in Dictionary; output
   previous output + previous output first character: AA, insert AA

Bài tập

Giải nén dữ liệu sau dùng LZW

<67> <70> <256> <258> <259> <257>

<86> <65> <65> <256> <257> <86>

86 = ký tự V

70 = ký tự F

Ứng dụng

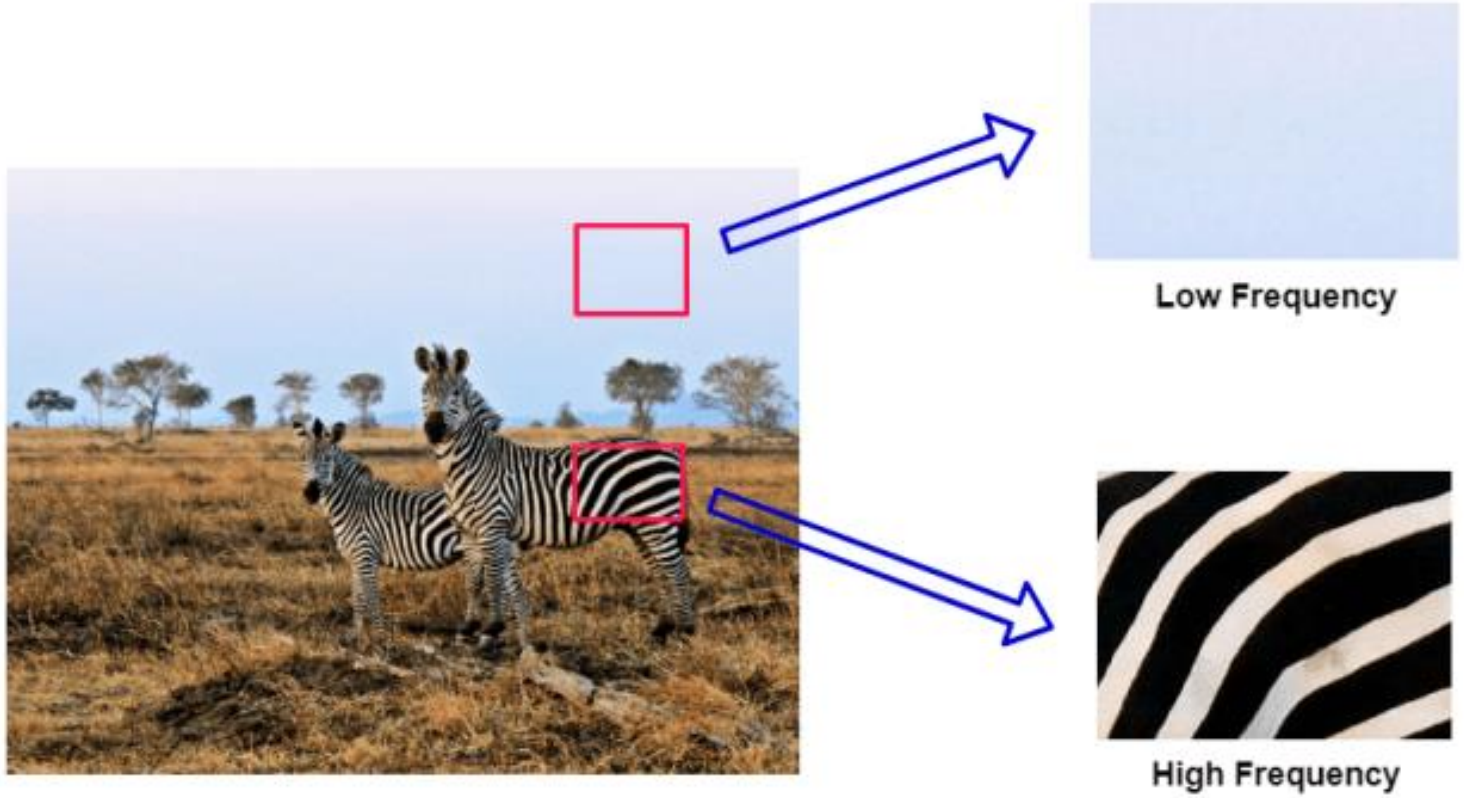
- LZW đã được sử dụng trong phần mềm nén mã nguồn mở, nó đã trở thành 1 phần không thể thiếu trong HDH UNIX CIRCA 1986
- LZW đã trở nên phổ biến khi nó được sử dụng làm 1 phần của file GIF năm 1987. Nó cũng có thể được sử dụng trong TIFF và PDF file

2. Nén có mất mát (lossy methods)

- Sau khi giải nén không thu lại dữ liệu gốc
- Tận dụng tính chất của mắt người, chấp nhận một số mất mát chấp nhận được (tương tự nhạc mp3)
- Loại bỏ chủ yếu ở các thành phần tần số cao (do mắt người khó phát hiện mất mát tại vùng tần số cao)

Chuẩn nén JPEG

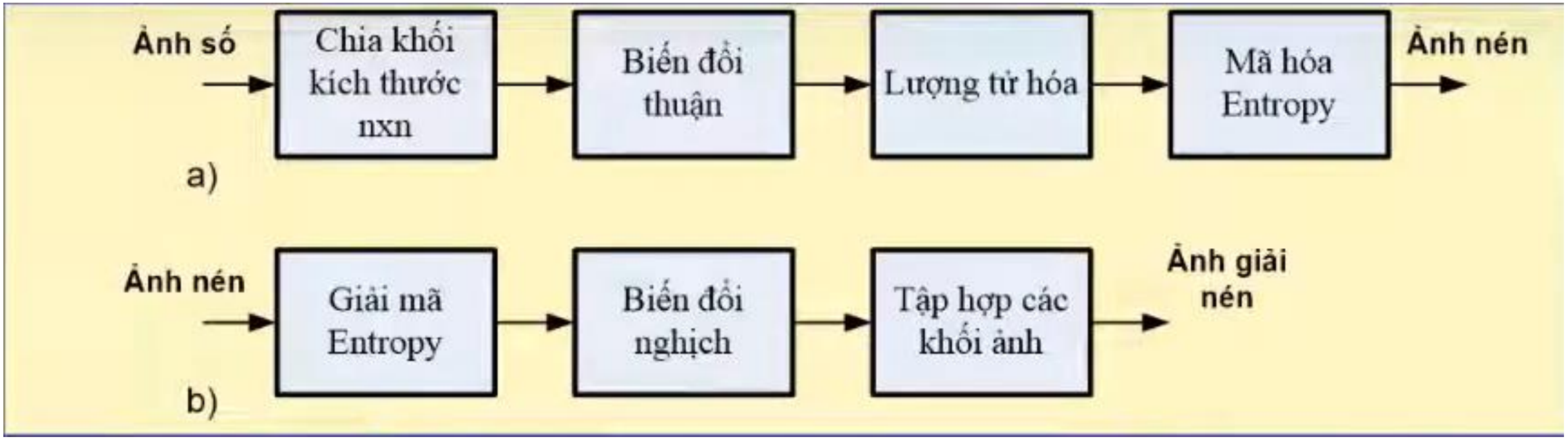
- Trong một khối ảnh đủ nhỏ (ví dụ 8x8) thì thành phần tần số thấp sẽ chứa nhiều thông tin hơn do cường độ xám ít thay đổi
- Mắt người khó phát hiện mất mát đối với vùng tần số cao hơn so với vùng tần số thấp



Quy trình nén ảnh jpeg

DCT thuận

lossless compression



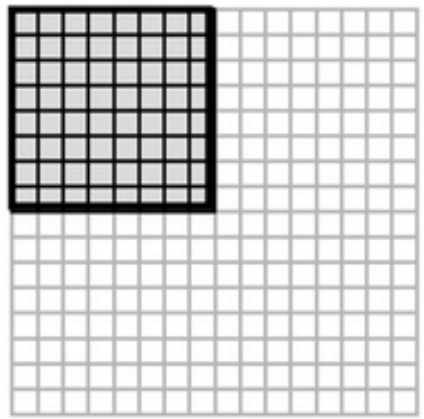
DCT nghịch

JPEG compression

Input

Orignal gray image
(large data size)

Step-1



Group 8x8 pixels block

Step-2

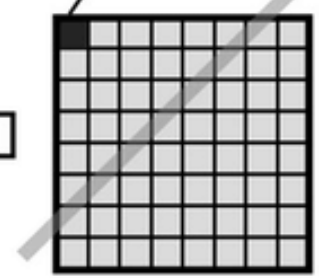
Discrete Cosine Transform (DCT) & Quantization

(Keep)
Low frequency

Threshold value

DC

High frequency (Cut)



DCT basis functions

Step-3

Entropy coding

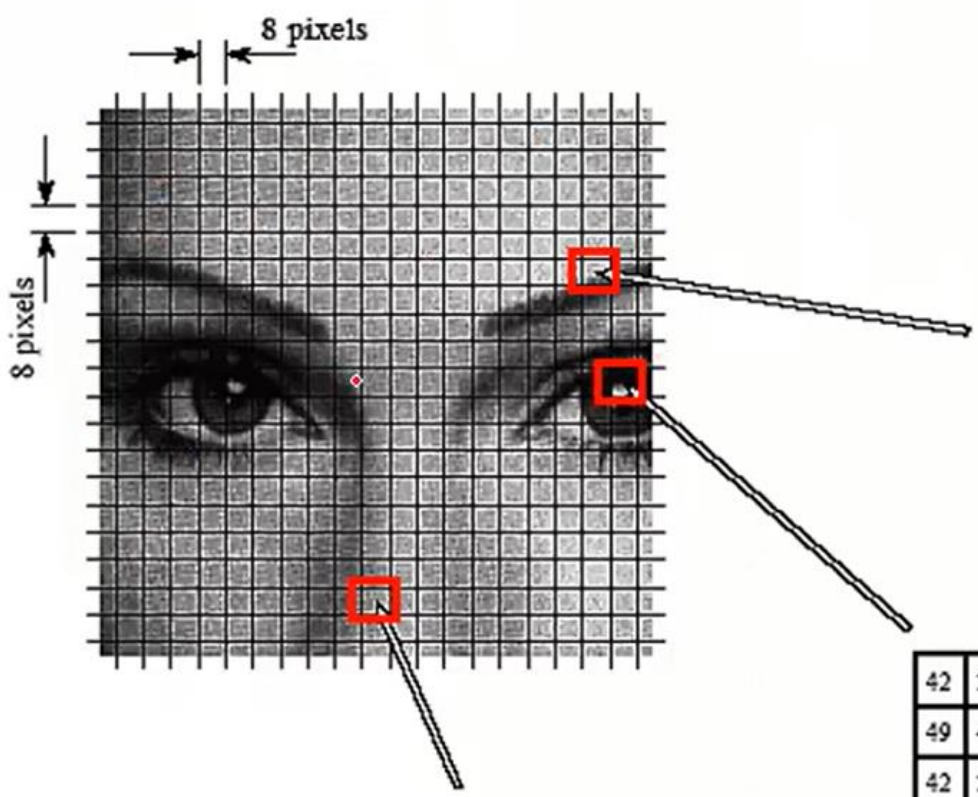
Step-4

Output

Compressed JPEG image
(small data size)

Chia ảnh

- Nếu ảnh kích thước nhỏ có thể chia thành các khối 8x8



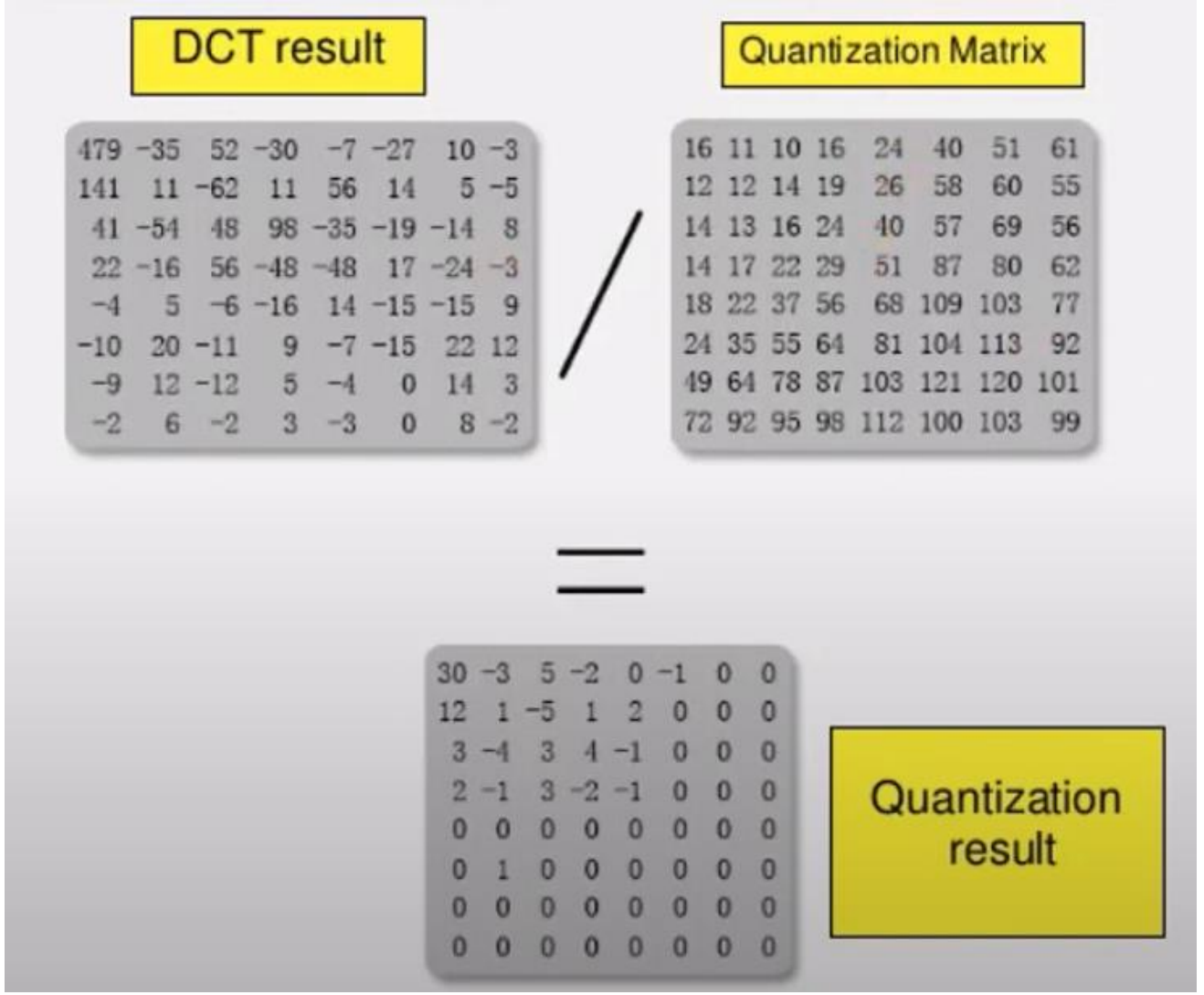
231	224	224	217	217	203	189	196
210	217	203	189	203	224	217	224
196	217	210	224	203	203	196	189
210	203	196	203	182	203	182	189
203	224	203	217	196	175	154	140
182	189	168	161	154	126	119	112
175	154	126	105	140	105	119	84
154	98	105	98	105	63	112	84

42	28	35	28	42	49	35	42
49	49	35	28	35	35	35	42
42	21	21	28	42	35	42	28
21	35	35	42	42	28	28	14
56	70	77	84	91	28	28	21
70	126	133	147	161	91	35	14
126	203	189	182	175	175	35	21
49	189	245	210	182	84	21	35

154	154	175	182	189	168	217	175
154	147	168	154	168	168	196	175
175	154	203	175	189	182	196	182
175	168	168	168	140	175	168	203
133	168	154	196	175	189	203	154
168	161	161	168	154	154	189	189
147	161	175	182	189	175	217	175
175	175	203	175	189	175	175	182

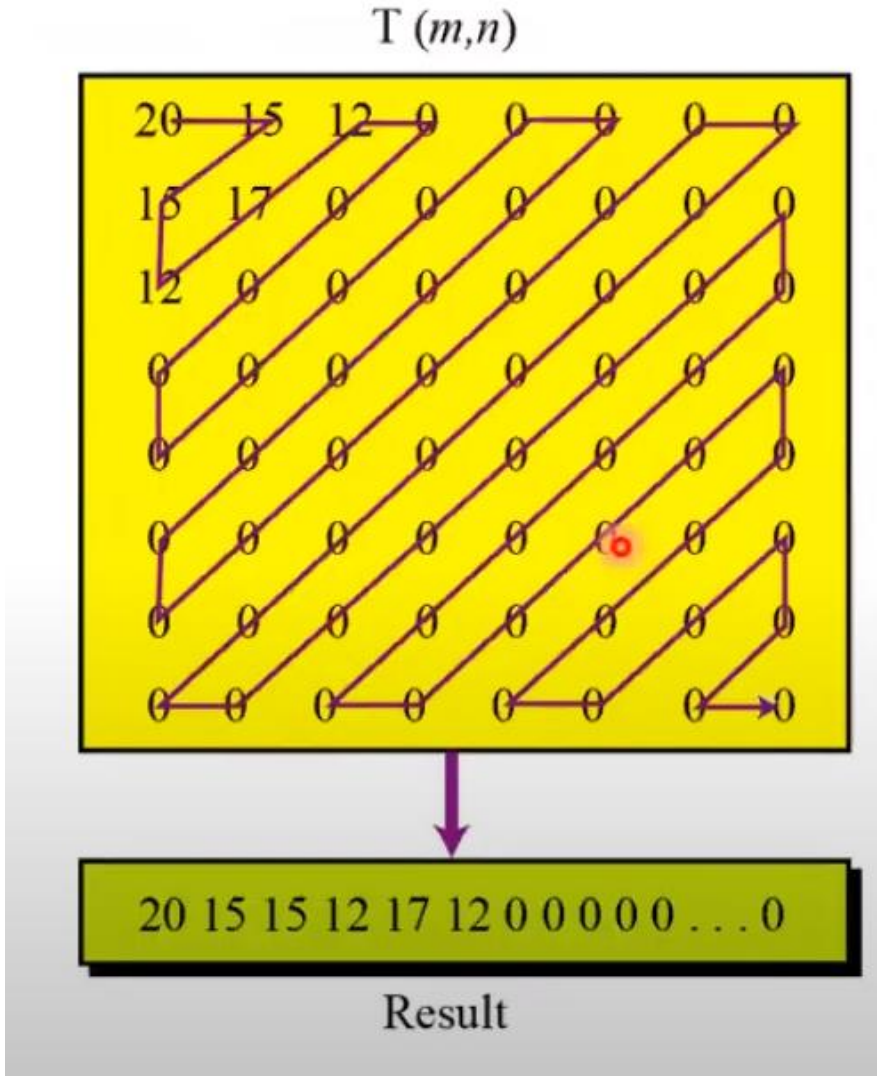
DCT thuận và Lượng tử hóa

- Sau khi chia khối 8x8 và biến đổi DCT thuận, ma trận DCT sẽ được lượng tử hóa (quantization)



Đọc zig-zag và mã hóa

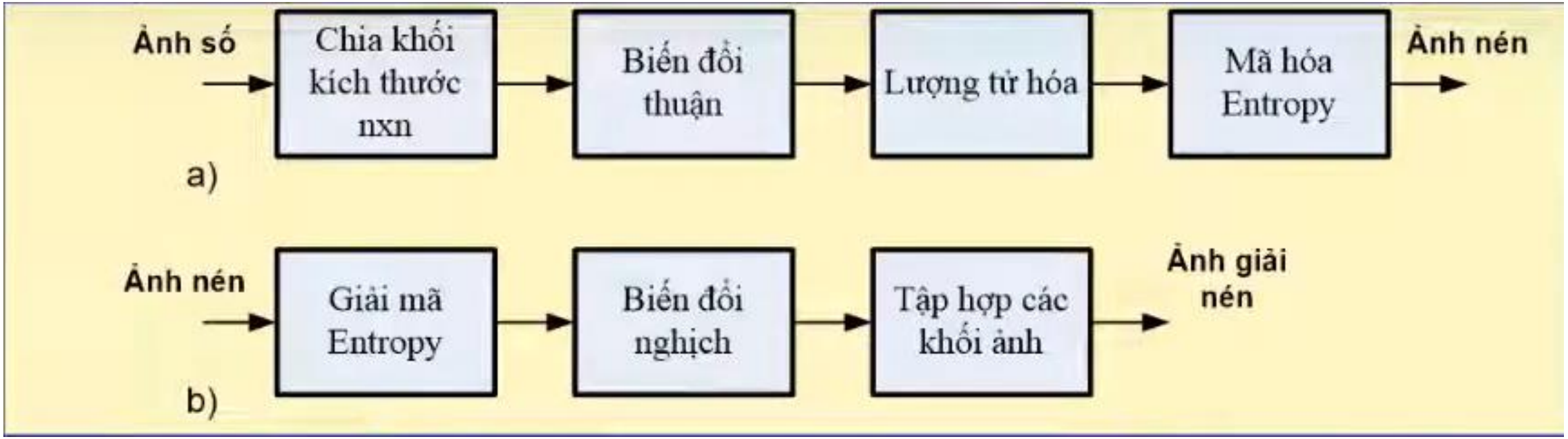
- Sau khi lượng tử hóa, ma trận T sẽ được đọc theo kiểu zig-zag để tối ưu hóa việc mã hóa Run length code (gom nhiều số 0)
- Mã hóa entropy



Quy trình nén ảnh jpeg

lossless compression

DCT thuận



DCT nghịch

- Sau đó ảnh nén sẽ được giải nén

PHẦN 2. Phân đoạn ảnh
(Thị giác máy tính)

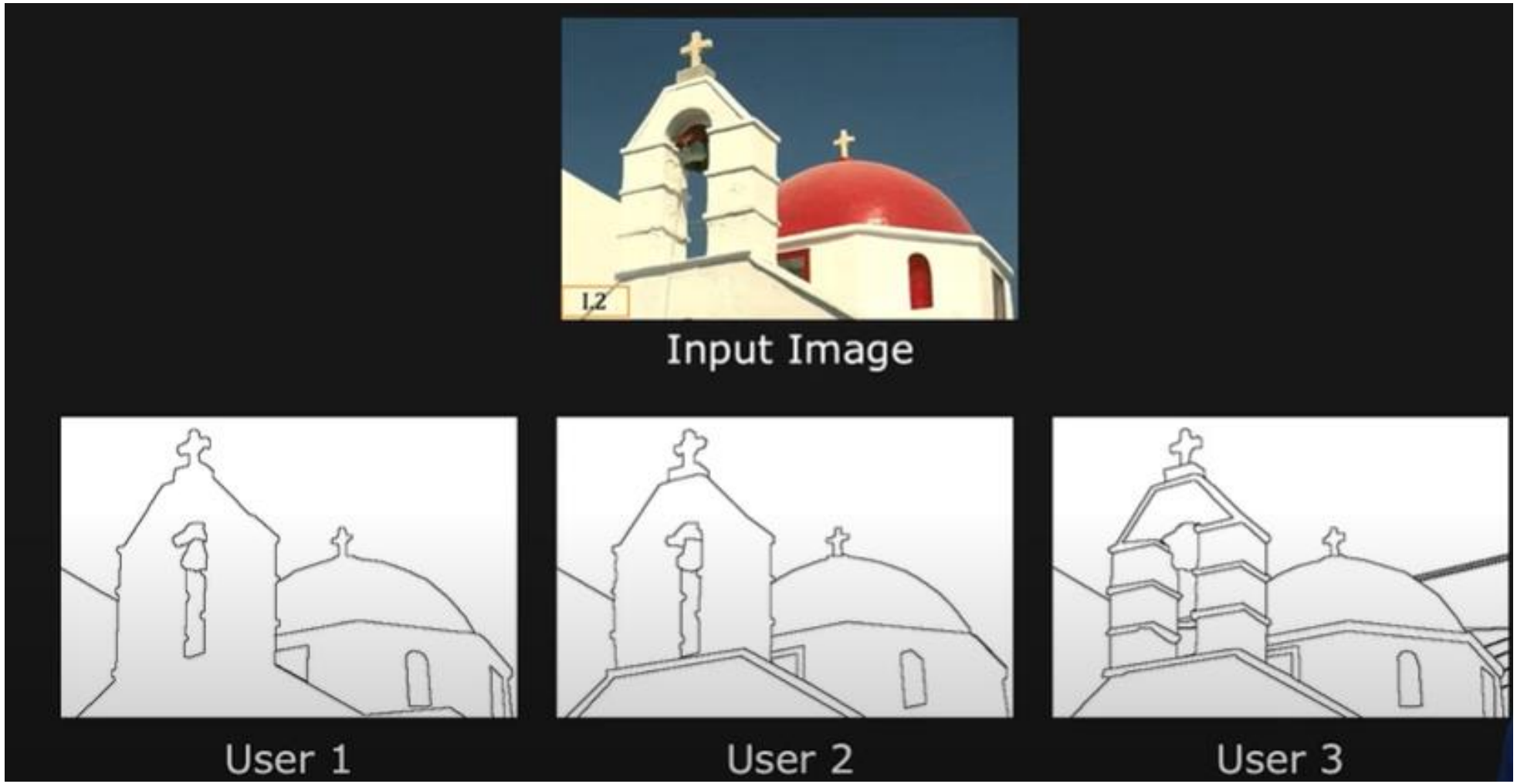
Mục tiêu của phân đoạn ảnh

- ❑ Hiểu các đối tượng/vùng trong ảnh
- ❑ Đối sánh và nhận dạng ảnh
- ❑ Dò tìm đối tượng trong ảnh
- ❑ Phân lớp

...

1. Phân đoạn ảnh ở người

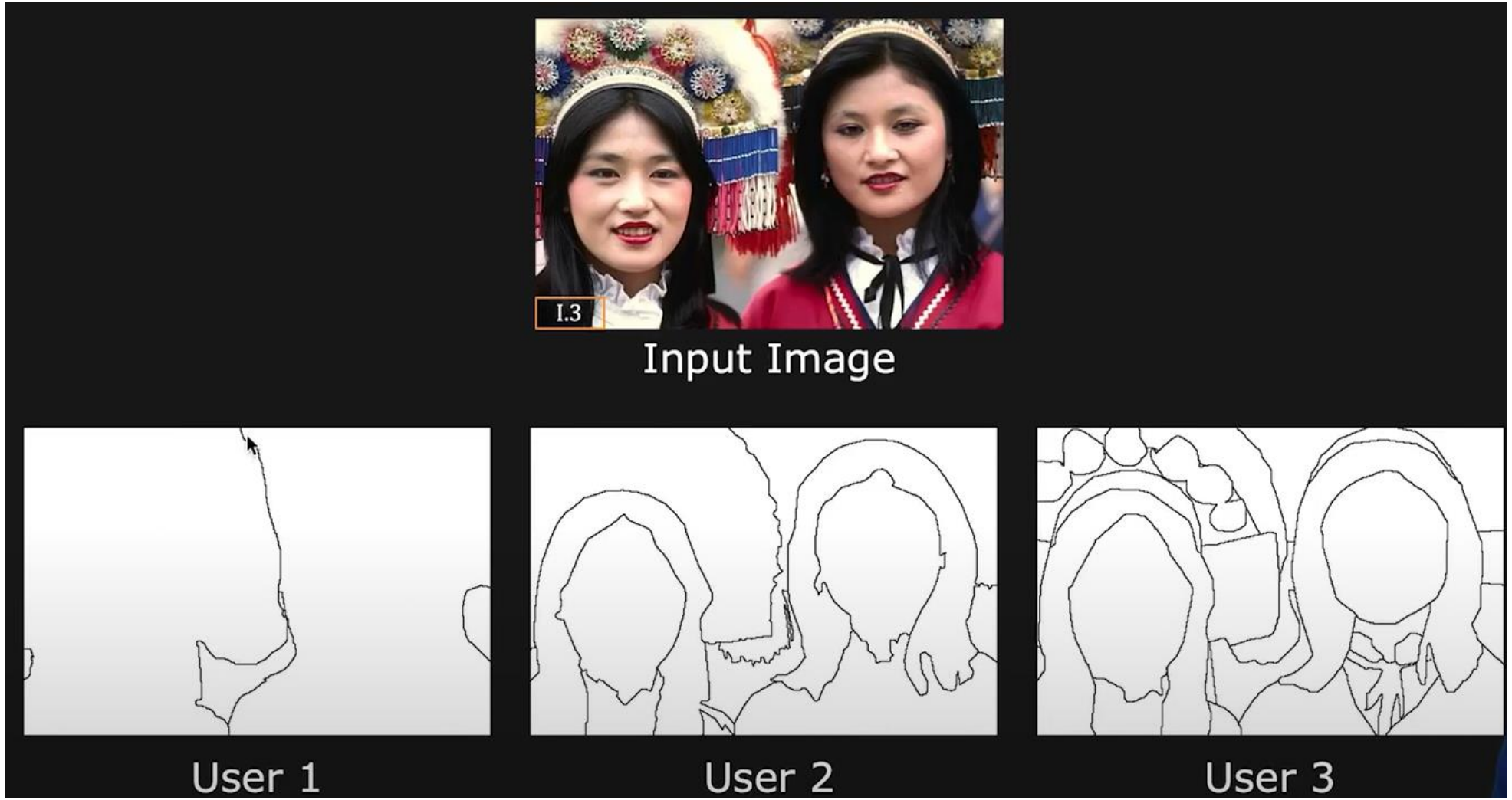
- Phân đoạn ảnh ở mắt người là chủ quan
- Rất khó để chuyển sự trực quan ở mắt người sang máy tính



Martin, 2001

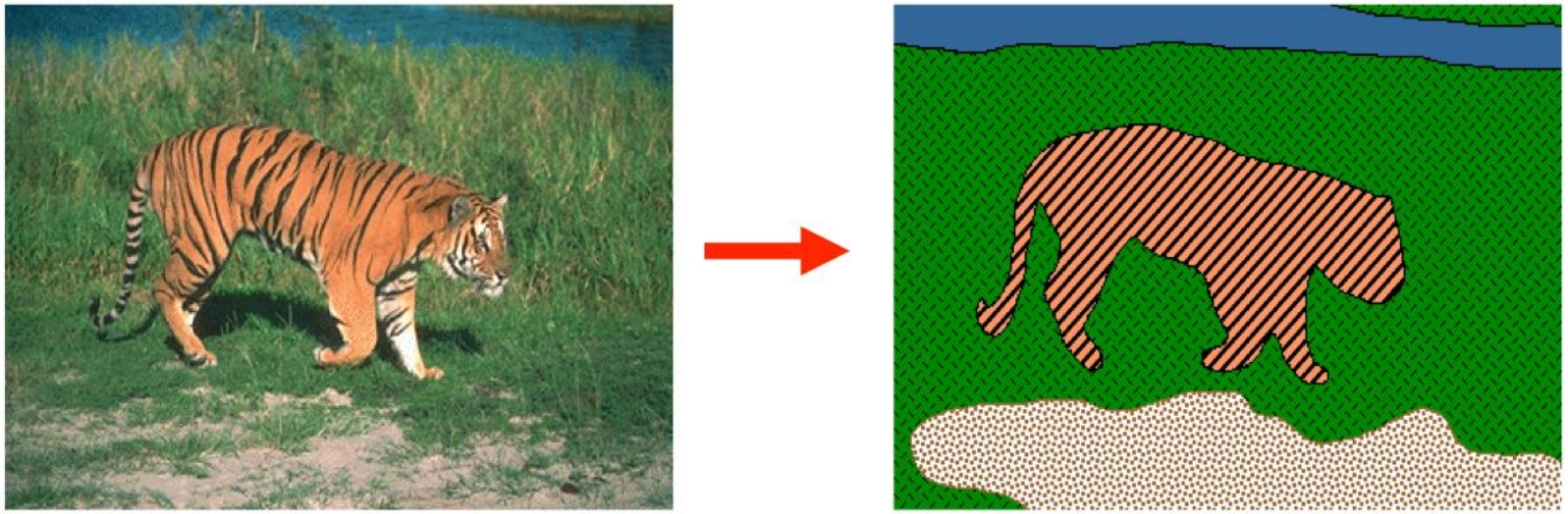
1. Phân đoạn ảnh ở người

- Phân đoạn ảnh ở mắt người là chủ quan
- Rất khó để chuyển sự trực quan ở mắt người sang máy tính



2. Phân đoạn ảnh trên máy tính

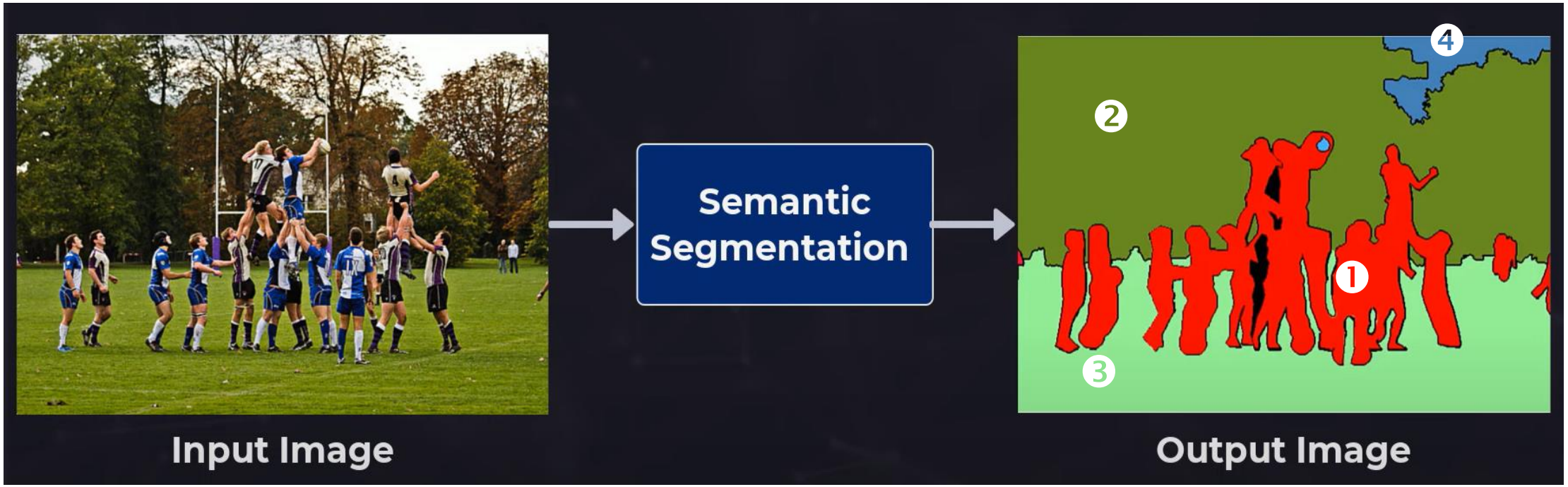
- Phân chia một ảnh (R) thành các vùng con chứa các pixel dựa trên một tiêu chí nào đó $R_1, R_2, R_3, \ldots R_n$ với các tính chất:
  - $R_1 \cup R_2 \cup R_3 \cup \ldots \cup R_n = R$
  - $R_i \cap R_j = \emptyset$ với $i \neq j$



*Tiêu chí: cường độ, histogram, mean…

2.1 Phân đoạn theo ngữ nghĩa

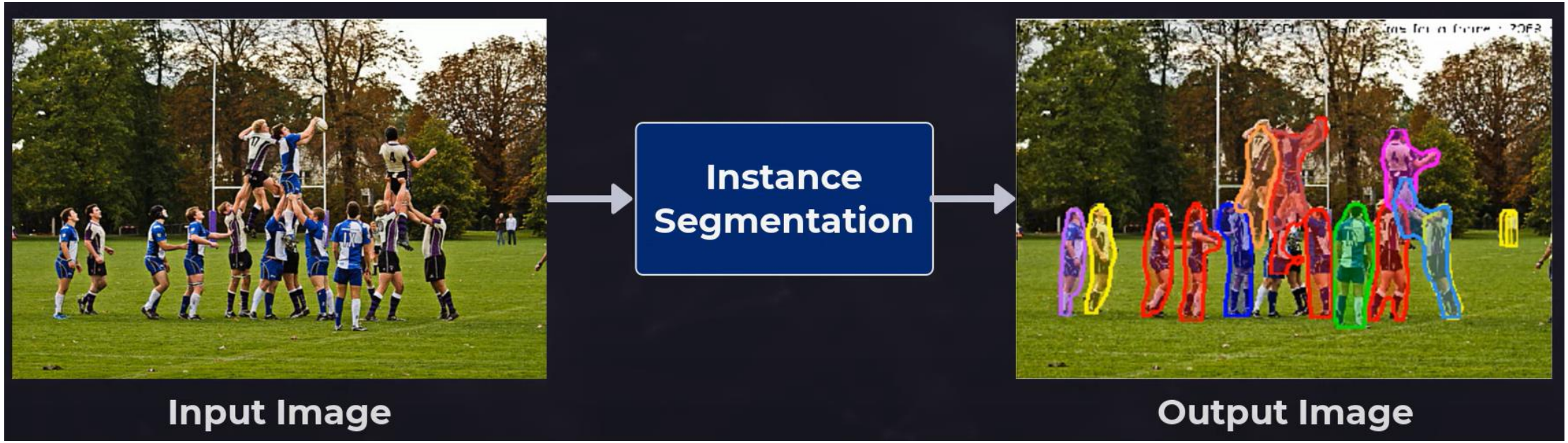
- Phân đoạn theo ngữ nghĩa (semantic segmentation) dùng để gán nhãn cho từng lớp như xe hơi, người, chó, mèo… cho mỗi pixel của ảnh



© LearnOpenCV

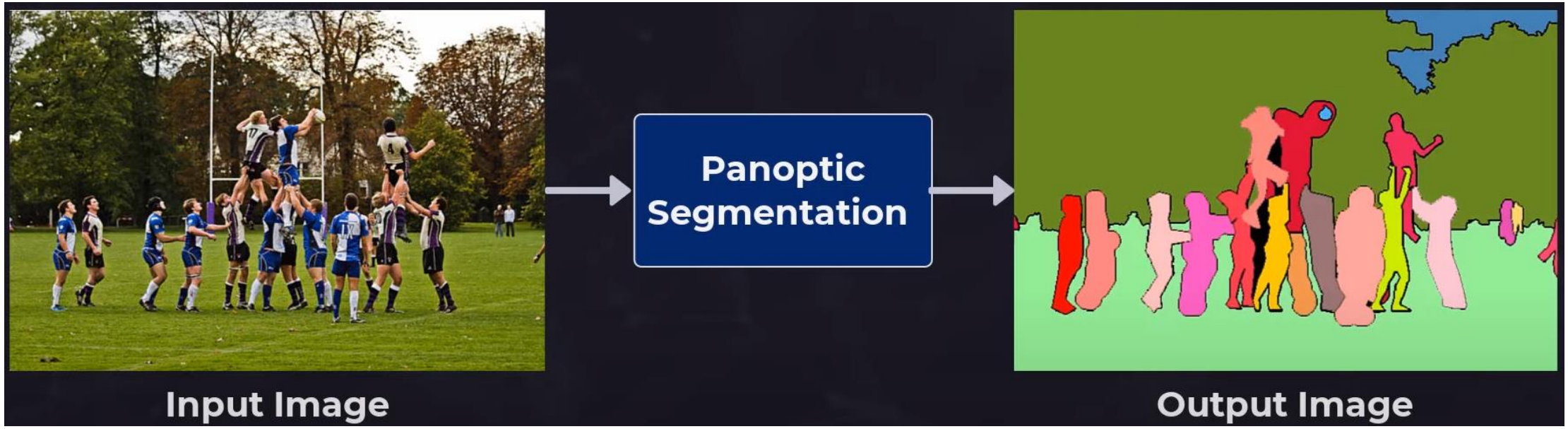
2.2 Phân đoạn theo đối tượng cụ thể

- Phân đoạn theo đối tượng cụ thể (instance segmentation) sử dụng mask để phân biệt từng đối tượng cụ thể chung một lớp
- Quan tâm đến boundary của đối tượng cụ thể

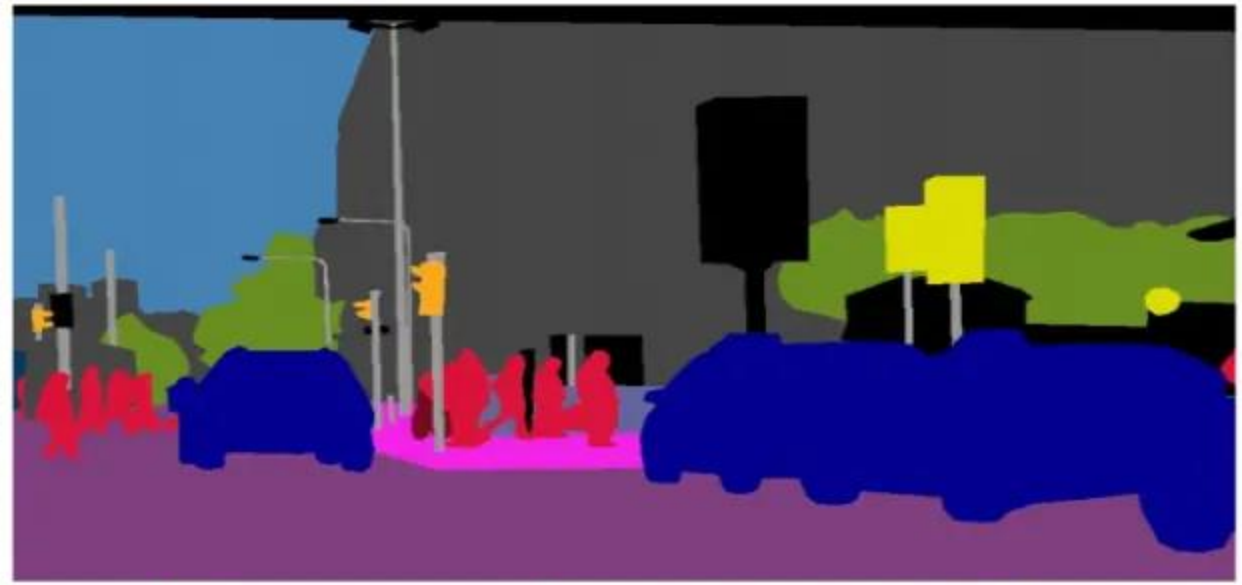


1.3 Phân đoạn Panoptic

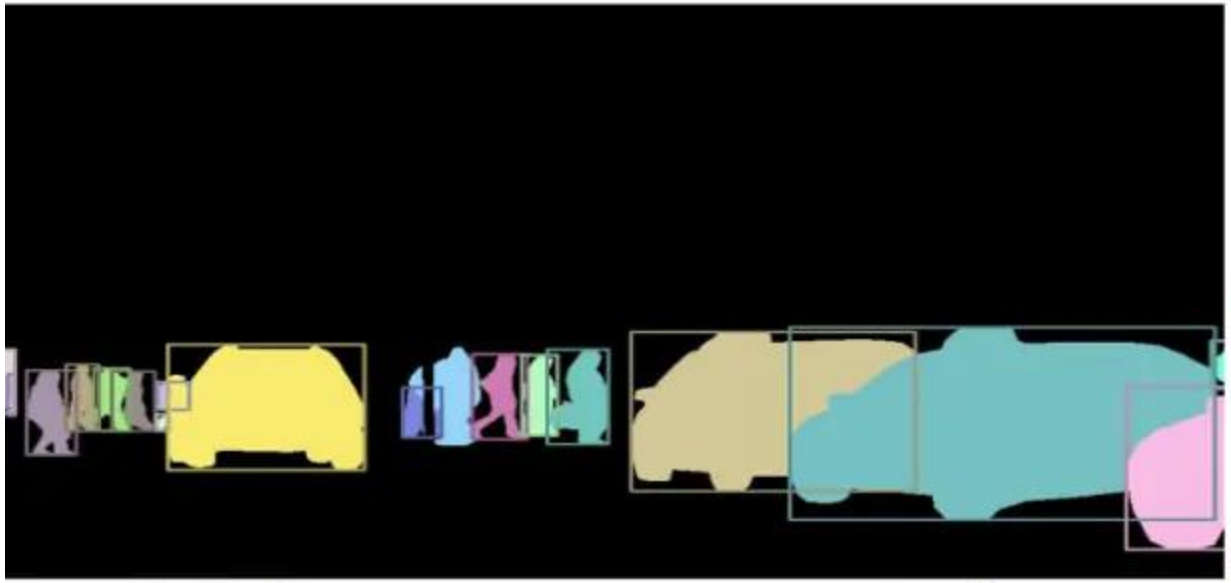
- Kết hợp cả phân đoạn semantic và instance, ta được phân đoạn kiểu Panoptic



(a) image



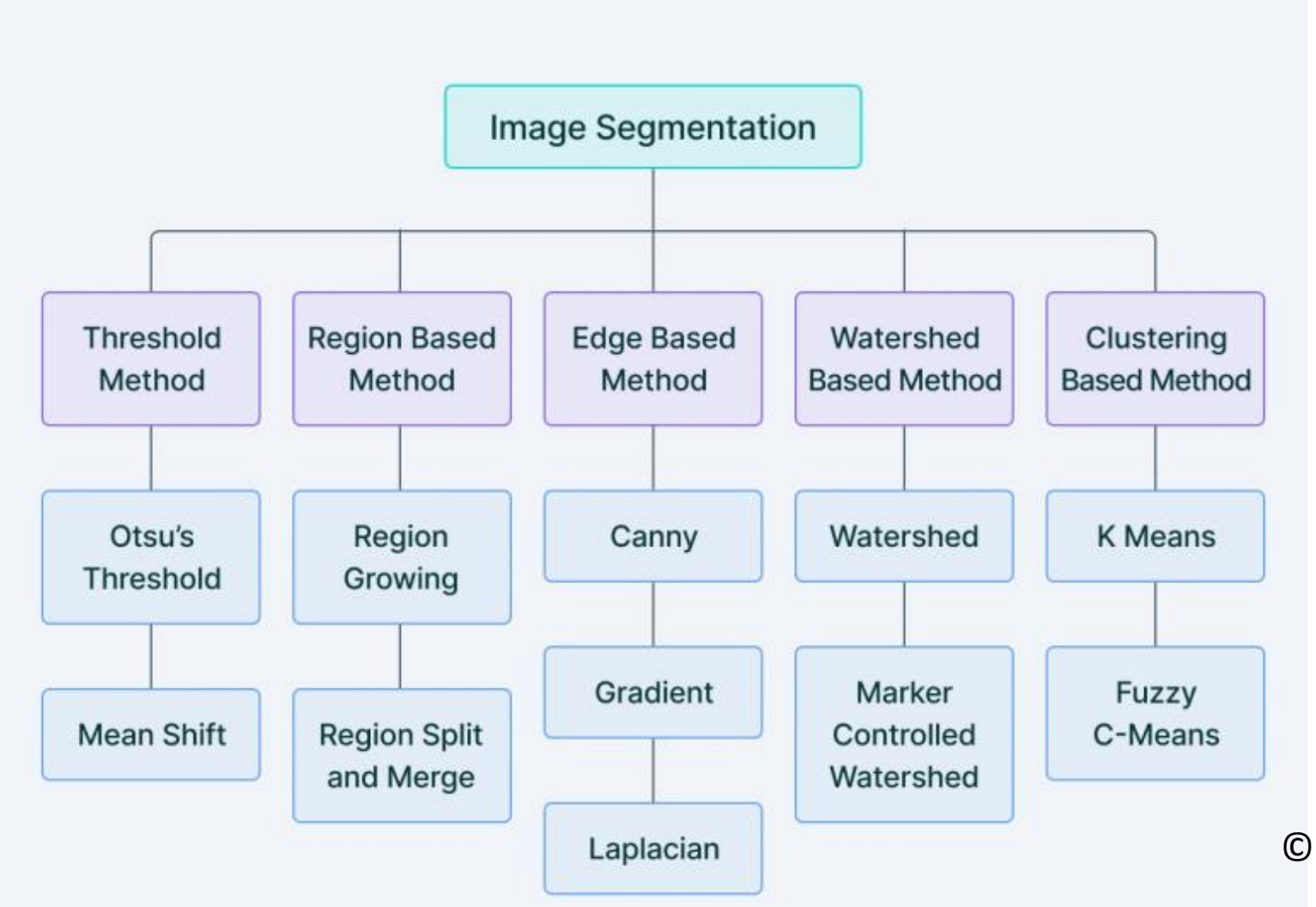
(b) semantic segmentation



(c) instance segmentation

(d) panoptic segmentation

3. Các phương pháp phân đoạn ảnh



© V7 Lab