XỬ LÝ ẢNH SỐ

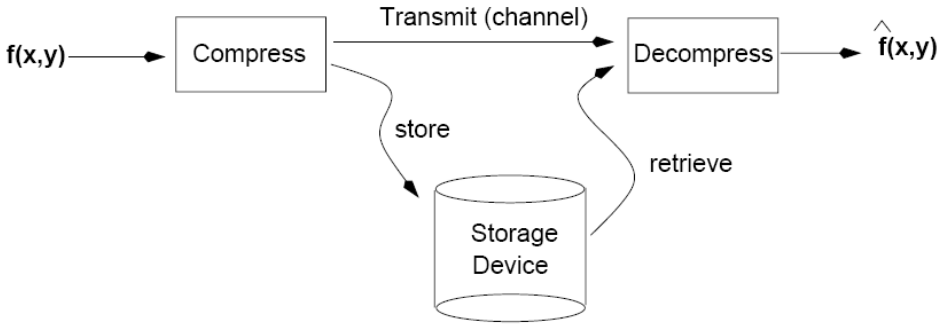
GIỚI THIỆU

Trong phần này chúng ta sẽ tìm hiểu về các kỹ thuật được áp dụng để nén ảnh:

- Mục đích của nén ảnh
- Các kiểu dư thừa dữ liệu
- Các bước trong mô hình nén và giải nén
- Các thuật toán mã hóa và giải mã

Mục đích của nén ảnh

- Ảnh số không được nén sẽ cần một lượng lớn không gian để lưu trữ và băng thông để truyền
  - Ví dụ ảnh màu kích thước 640 x 480 cần không gian lưu trữ lên tới 1MB.
- Mục đích của việc nén ảnh là để giảm số lượng dữ liệu cần để biểu diễn ảnh. Do đó sẽ giảm không gian cần để lưu trữ và tăng được tốc độ đường truyền



Các cách tiếp cận

- Phương pháp nén bảo toàn thông tin
  - ✓ Thông tin được bảo toàn
  - ✓ Tỷ lệ nén thấp
- Phương pháp nén làm mất thông tin
  - ✓ Thông tin không được bảo toàn
  - ✓ Tỷ lệ nén cao

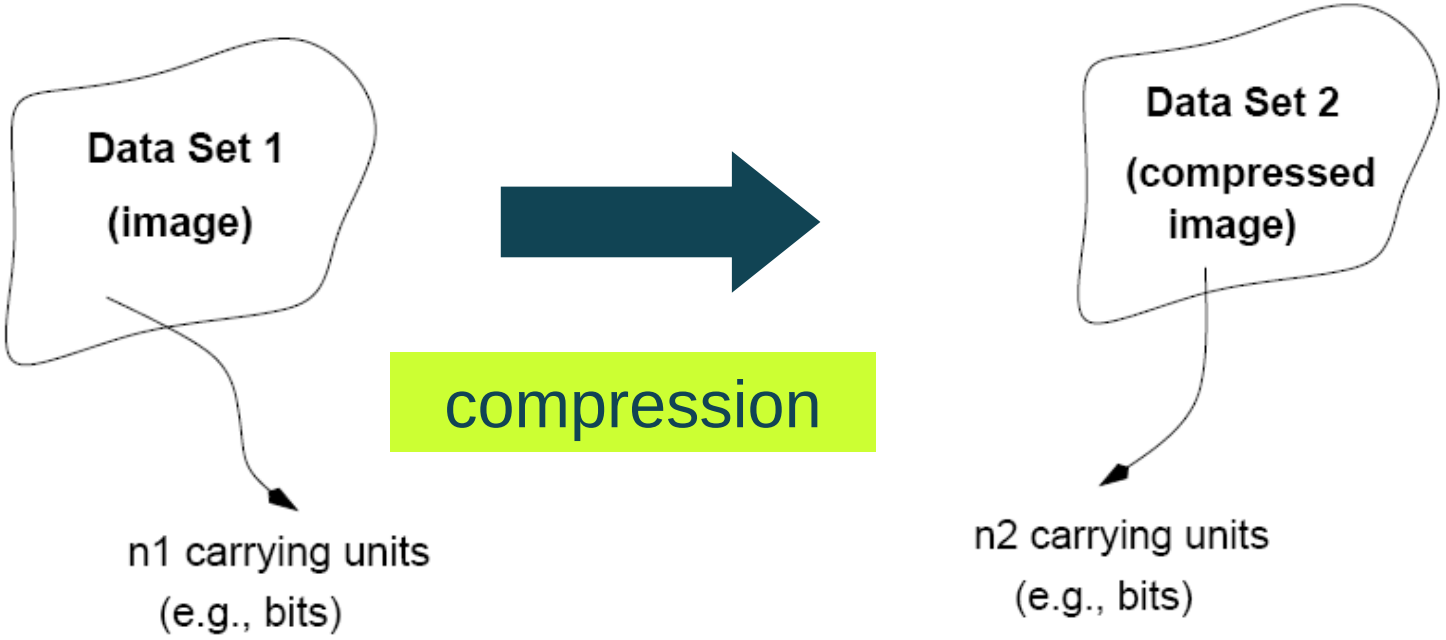
Cân nhắc: chất lượng ảnh và tỷ lệ nén

Dữ liệu vs. thông tin

- Dữ liệu (data) là phương tiện được dùng để truyền tải thông tin

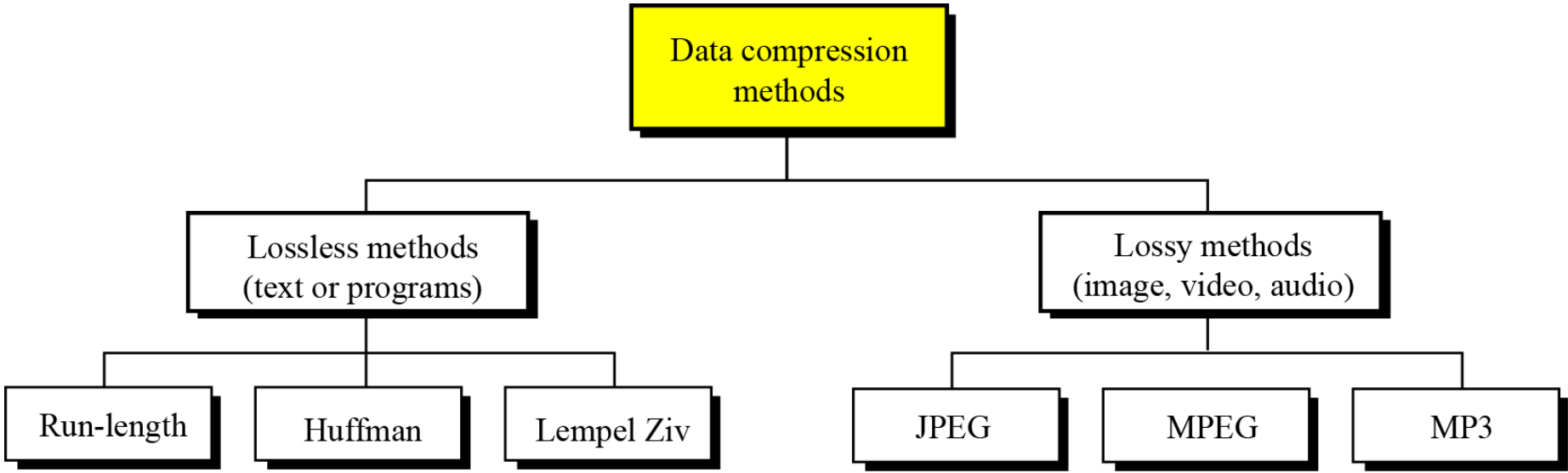
- Nén dữ liệu là phương pháp giảm số lượng dữ liệu cần để biểu diễn 1 lượng thông tin cho trước với tiêu chí giữ lại được càng nhiều thông tin càng tốt

Tỷ lệ nén



Tỷ lệ nén: $C_R = \frac{n_1}{n_2}$

Các phương pháp nén dữ liệu



Nén dữ liệu không tổn thất

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

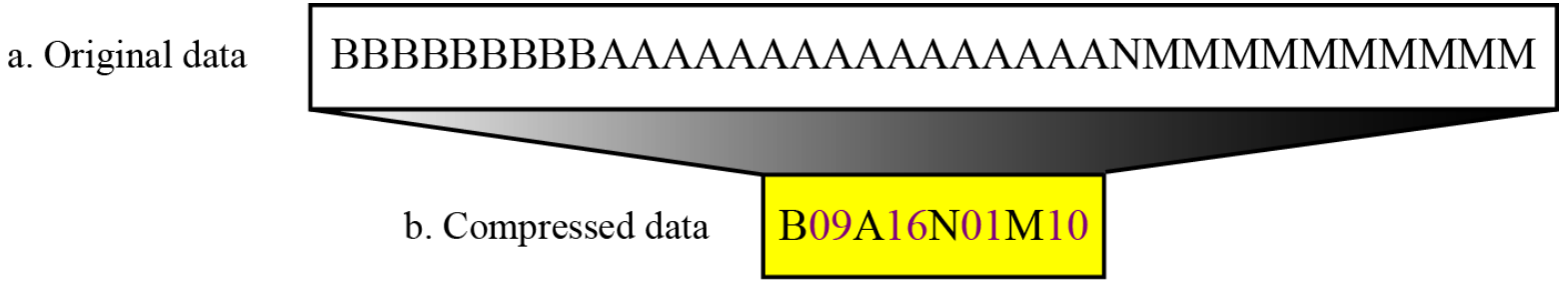
Nén dữ liệu không tổn thất

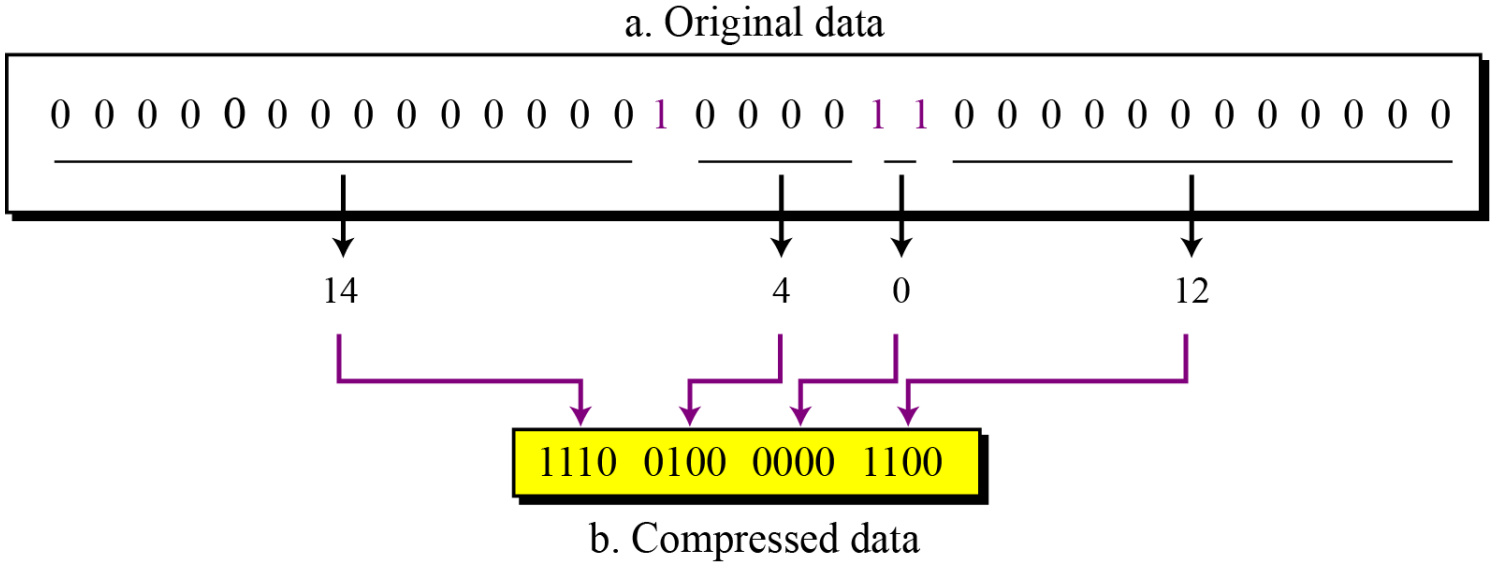
- Trong nén dữ liệu không tổn thất, tính toàn vẹn của dữ liệu được bảo toàn.
- Dữ liệu gốc và dữ liệu sau khi nén và giải nén giống nhau hoàn toàn vì trong các phương pháp này, thuật toán nén và giải nén chính xác là biến đổi ngược của nhau.
- Dữ liệu dư thừa được loại bỏ trong quá trình nén và được thêm vào trong quá trình giải nén.
- Nén dữ liệu không tổn thất thường được sử dụng khi chúng ta không muốn mất mát thông tin.

Run-length coding (RLC)

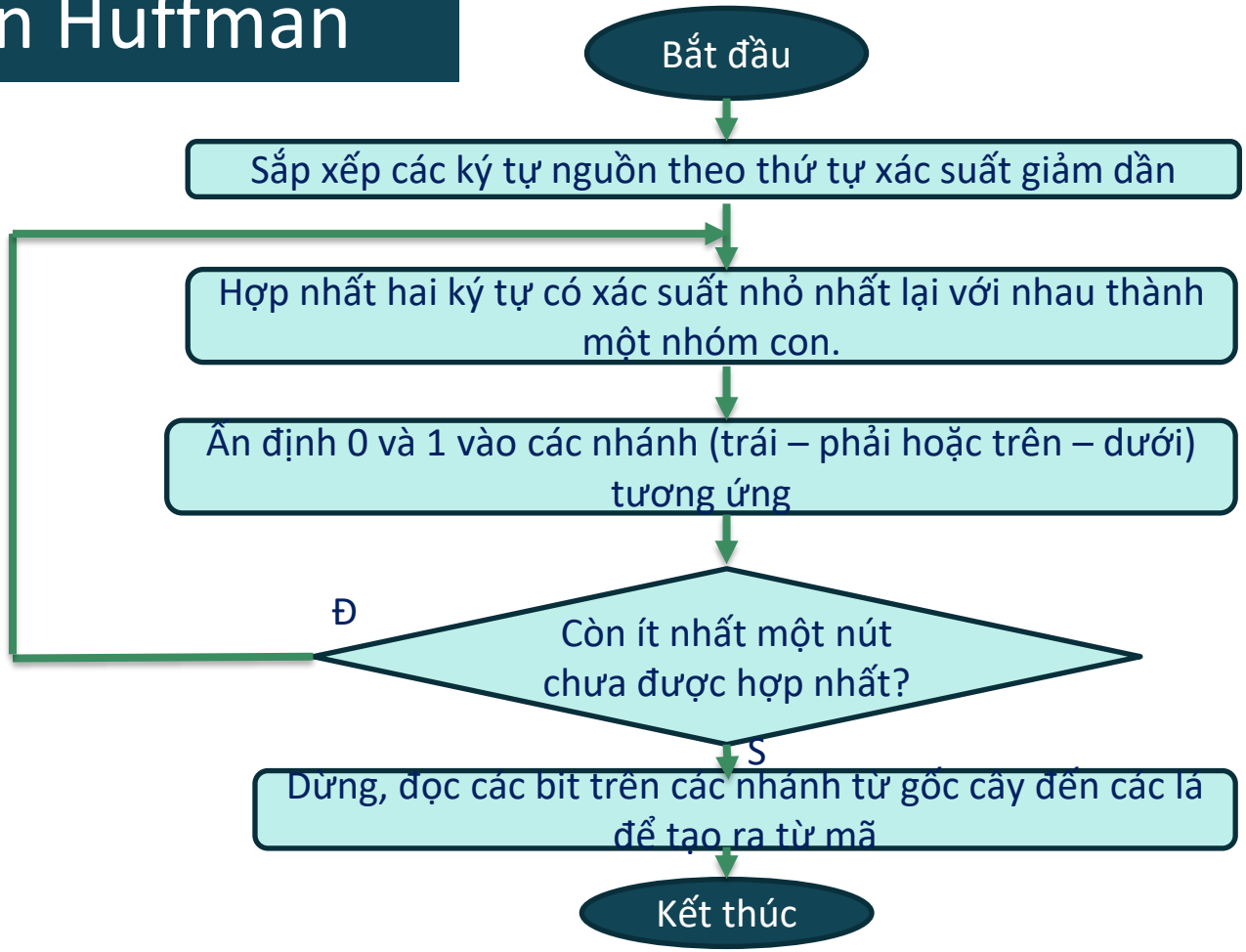
- Mã hoá chiều dài chạy (RLC) là kỹ thuật mã hoá đơn giản nhất và được sử dụng rộng rãi.
- Sử dụng mã này không cần biết tần suất xuất hiện các ký tự và rất hiệu quả trong trường hợp dữ liệu được hiển thị bằng các bit 0 và 1.
- Ta thay thế một chuỗi các ký tự lặp đi lặp lại nhiều lần bằng một cặp (*chiều dài chạy, ký tự*).
- Đối với ảnh, chiều dài chạy cực đại chính là kích thước của một hàng.

Ví dụ mã RLC


a. Original data
BBBBBBBBBAAAAAAAAAAAAAAANMMMMMMMMMM
b. Compressed data
B09A16N01M10


a. Original data
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
14
4
0
12
1110 0100 0000 1100
b. Compressed data

Thuật toán Huffman



Ví dụ

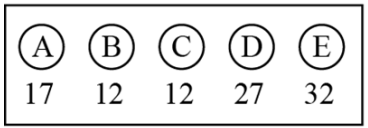
Cho tập 5 ký tự A, B, C, D, E với tần suất xuất hiện tương ứng trong bảng dưới đây.

A. Thực hiện mã hoá Huffman cho tập ký tự này.
B. Tính hiệu suất mã

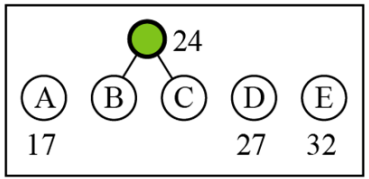
Ký tự	**A**	**B**	**C**	**D**	**E**
Tần suất xuất hiện	17	12	12	27	32
	0,17	0,12	0,12	0,27	0,32

Giải

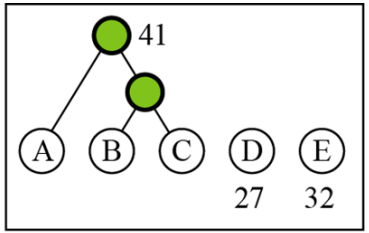
Ký tự	A	B	C	D	E
Tần suất xuất hiện	17	12	12	27	32



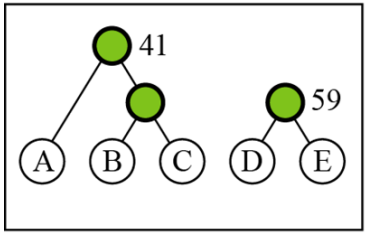
a.



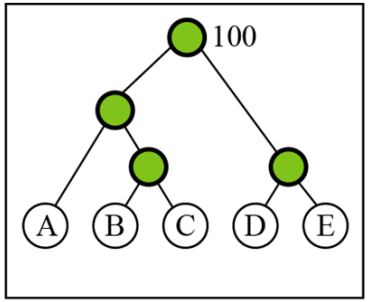
b.



c.

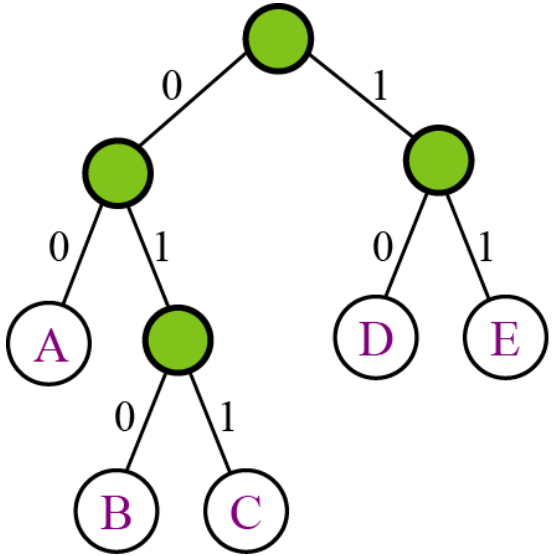


d.



e.

Ví dụ: Phát hiện điểm

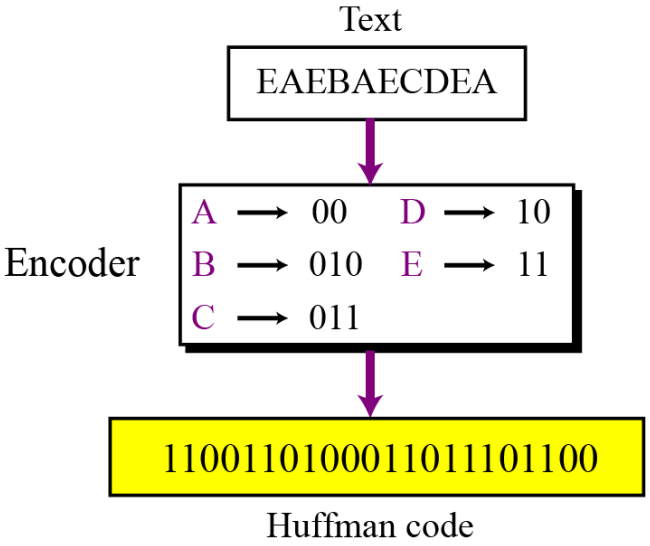


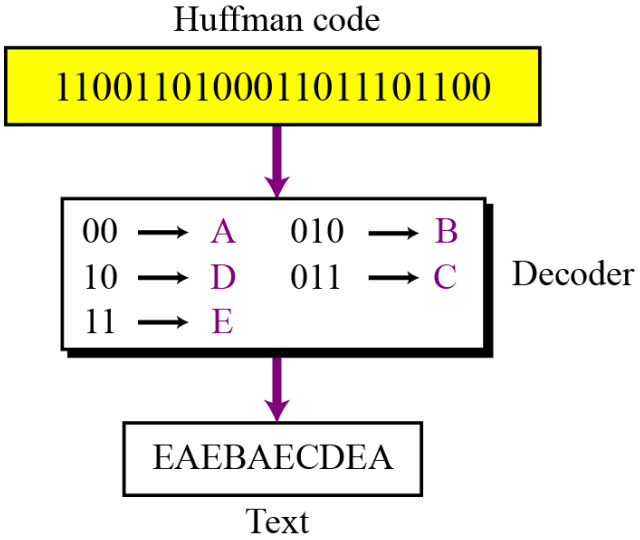
A: 00 D: 10
B: 010 E: 11
C: 011

Code

Ký tự	A	B	C	D	E
Tần suất xuất hiện	17	12	12	27	32

Ví dụ quá trình mã hóa và giải mã





Đặc điểm của mã Huffman

- Quá trình mã hóa các ký tự được thực hiện chỉ 1 lần.
- Mã Huffman được gọi là *mã khối* vì mỗi ký tự được biểu diễn bằng một từ mã cố định.
- Việc giải mã từng ký hiệu được thực hiện tức thời vì không phụ thuộc ký hiệu trước và sau.
- Nhược điểm của mã này là phải biết xác suất xuất hiện của các ký tự trước khi mã hoá.

Bài tập

Thực hiện mã hóa ảnh sau bằng thuật toán Huffman. Được biết ảnh được chia làm các khối kích thước 2x2 để làm đơn vị mã hóa (Mỗi khối này sẽ như là một chữ cái của bức ảnh).

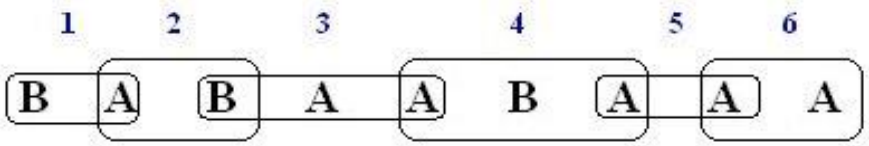
$$I = \left[\begin{array}{cc:cc:cc:cc:cc:cc:cc:cc:cc:cc} 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ \hdashline 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 &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

Mã LZW – Lempel-Ziv-Welch

- Ý tưởng là tạo ra từ điển (1 bảng) các chuỗi được sử dụng trong phiên truyền thông.
- LZW sử dụng các từ mã chiều dài cố định để biểu diễn các chuỗi ký tự chiều dài thay đổi thường xảy ra cùng nhau, ví dụ các từ trong đoạn văn bản.
- Bộ mã hoá và giải mã LZW cùng xây dựng một bộ từ điển trong quá trình nhận dữ liệu.
- Nếu cả bên gửi và bên nhận đều có bản copy của cuốn từ điển (dictionary) thì các chuỗi đã gặp trước đó sẽ được thay thế bằng mục lục của chúng để làm giảm lượng thông tin cần truyền.

Mã LZW – Lempel-Ziv-Welch

***Ví dụ 1*: Mã hóa dãy ký tự sau:**



1. **BA** is not in the Dictionary; insert **BA**, output the code for its prefix: **code(B)**
2. **AB** is not in the Dictionary; insert **AB**, output the code for its prefix: **code(A)**
3. **BA** is in the Dictionary.
   BAA is not in Dictionary; insert **BAA**, output the code for its prefix: **code(BA)**
4. **AB** is in the Dictionary.
   ABA is not in the Dictionary; insert **ABA**, output the code for its prefix: **code(AB)**
5. **AA** is not in the Dictionary; insert **AA**, output the code for its prefix: **code(A)**
6. **AA** is in the Dictionary and it is the last pattern; output its code: **code(AA)**

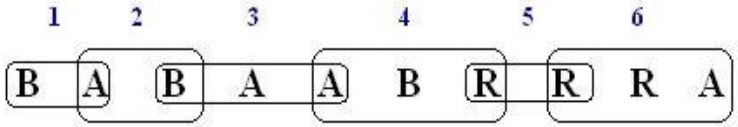
output	Dictionary	
	Code Word	String
66	256	BA
65	257	AB
256	258	BAA
257	259	ABA
65	260	AA
260		

Bản tin sau nén:

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

Mã LZW – Lempel-Ziv-Welch

***Ví dụ 2*: Mã hóa dãy ký tự sau:**



1. **BA** is not in the Dictionary; insert **BA**, output the code for its prefix: **code(B)**
2. **AB** is not in the Dictionary; insert **AB**, output the code for its prefix: **code(A)**
3. **BA** is in the Dictionary.
   BAA is not in Dictionary; insert **BAA**, output the code for its prefix: **code(BA)**
4. **AB** is in the Dictionary.
   ABR is not in the Dictionary; insert **ABR**, output the code for its prefix: **code(AB)**
5. **RR** is not in the Dictionary; insert **RR**, output the code for its prefix: **code(R)**
6. RR is in the Dictionary.
   RRA is not in the Dictionary and it is the last pattern; insert **RRA**, output code for its prefix: **code(RR),** then output code for last character: **code(A)**

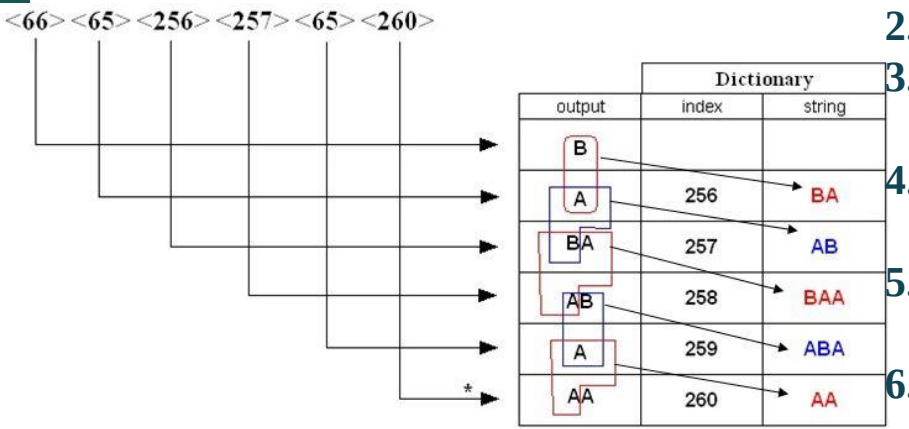
output	Dictionary	
	Code Word	String
66	256	BA
65	257	AB
256	258	BAA
257	259	ABR
82	260	RR
260	261	RRA
65		

Bản tin sau nén:

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

Giải mã LZW

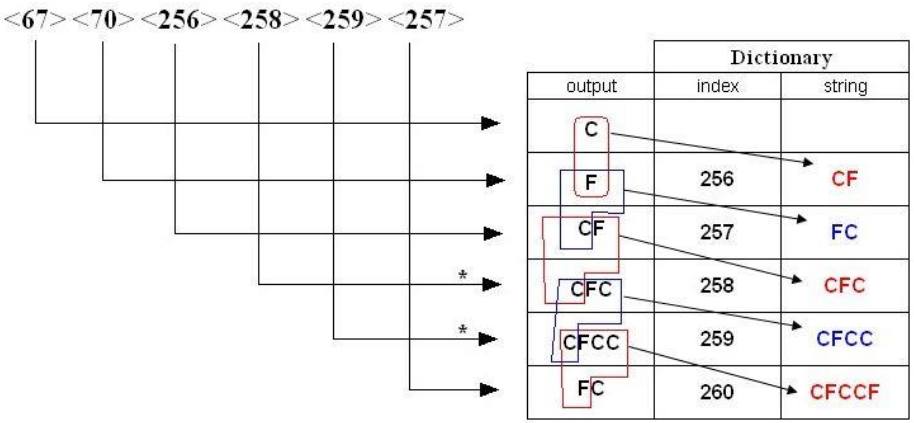
Sử dụng mã LZW để giải nén cho chuỗi đầu ra **<66> <65> <256> <257> <65> <260>**



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**

Giải mã LZW

Giải mã LZW cho chuỗi **<67> <70> <256> <258> <259> <257>**



1. **67** is in Dictionary; output **string(67)** i.e. **C**
2. **70** is in Dictionary; output **string(70)** i.e. **F**, insert **CF**
3. **256** is in Dictionary; output **string(256)** i.e. **CF**, insert **FC**
4. **258** is **not** in Dictionary; output **previous output + C** i.e. **CFC**, insert **CFC**
5. **259** is **not** in Dictionary; output **previous output + C** i.e. **CFCC**, insert **CFCC**
6. **257** is in Dictionary; output **string(257)** i.e. **FC**, insert **CFCCF**

Bài tập

Mã hóa bức ảnh sau bằng mã LZW:

39	39	126	126
39	39	126	126
39	39	126	126
39	39	126	126

Bài tập

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

Bài tập

Thực hiện mã hóa ảnh sau bằng kỹ thuật LZW. Được biết ảnh được chia làm các khối kích thước 1x2 để làm đơn vị mã hóa. Và từ điền gốc bao gồm 4 đơn vị mã hóa sau 00, 01, 10, 11 tương đương với giá trị từ 0 đến 3, từ điển sẽ được xây dựng tiếp theo từ giá trị 4. Bức ảnh sẽ được đọc từ trái qua phải và từ trên xuống dưới.

a. Thực hiện mã hóa và giải mã ảnh trên với LZW. Coi từ điền là đủ lớn để không thiếu chỗ.

b. Ý tưởng cơ bản của mã hóa LZW là ở đâu? LZW có vấn đề gì và có cách nào để giải quyết nó không?

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

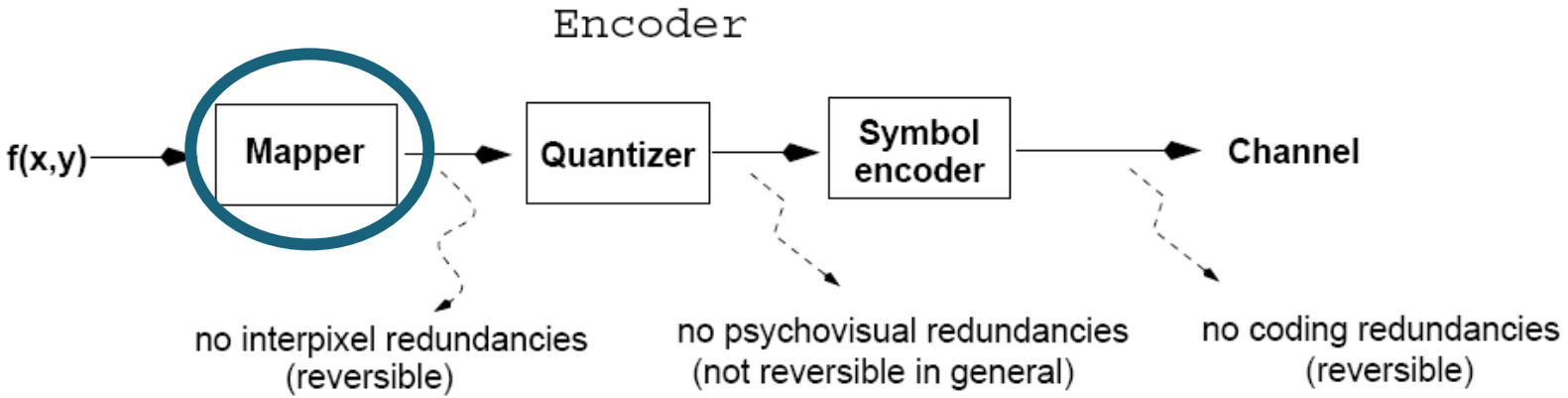
Bài tập

- Mã hóa và giải mã bức ảnh sau bằng thuật toán LZW với bộ từ điển gốc gồm các khối ảnh kích thước 1 x 3 tương ứng với các giá trị từ 0 đến 7, từ điển sẽ được xây dựng tiếp theo từ giá trị 8 và coi từ điển là đủ lớn.

$$I = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

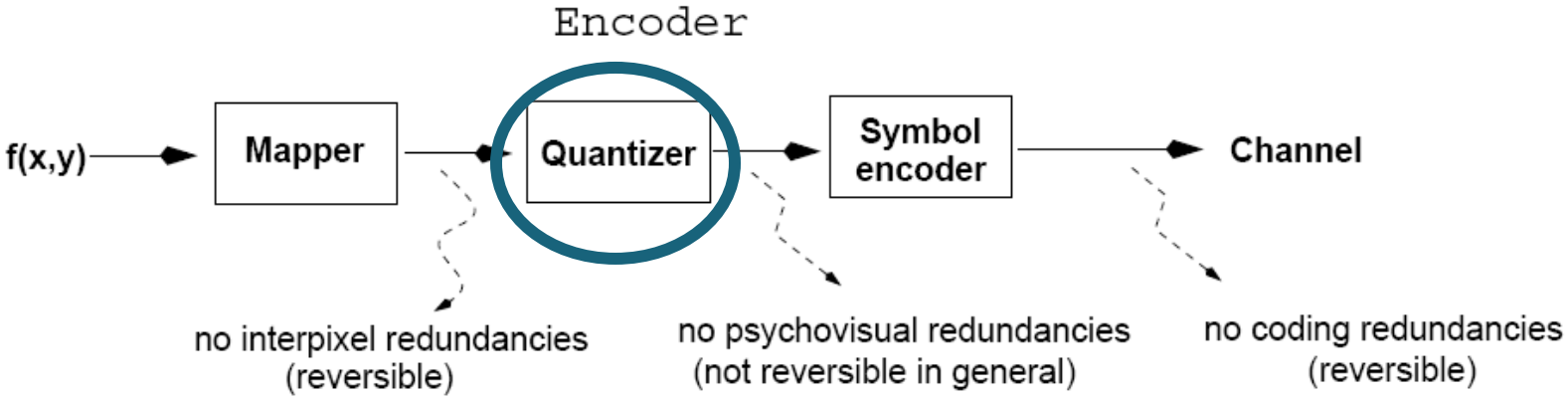
MÔ HÌNH MÃ HÓA ẢNH VÀ CHUẨN JPEG

Mô hình mã hóa ảnh



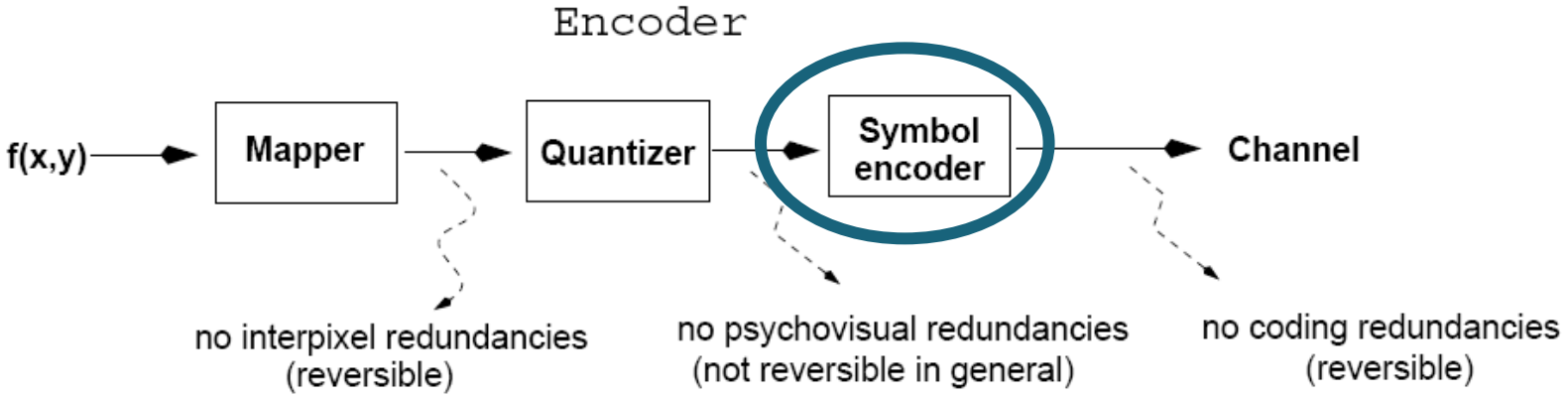
- ***Mapper***: Biến đổi ảnh đầu vào theo cách có thể làm giảm dư thừa thông tin giữa các pixel

Mô hình mã hóa ảnh



- ***Quantizer:*** Lượng tử hóa là bước tăng hoặc giảm các giá trị đầu ra của bước mapper theo các mức ngưỡng đã đặt ra và theo tiêu chí về chất lượng ảnh mong muốn

Mô hình mã hóa ảnh



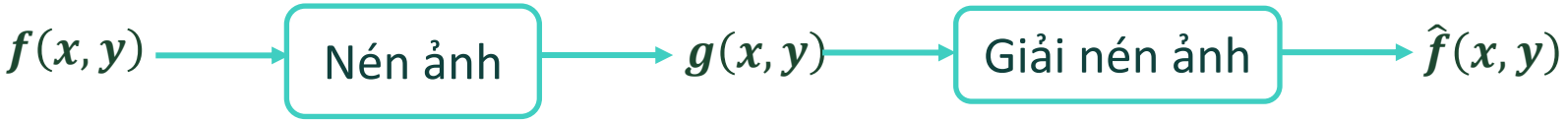
- ***Symbol encoder***: Mã hóa ký tự là bước gán mã có độ dài ngắn nhất cho ký tự có tần suất xảy ra nhiều nhất.

Mô hình giải mã ảnh

Bộ giải mã

- Tại phía giải mã, các bước của mã hóa được thực hiện theo thứ tự ngược lại.
- Riêng bước giải lượng tử không thể khôi phục lại các giá trị như ban đầu trước khi lượng tử.

Tiêu chí đánh giá chất lượng



$$\hat{f}(x,y) = f(x,y) + e(x,y)$$

- Trong quá trình nén ảnh, ngoài việc quan tâm tới tỷ lệ nén, chất lượng ảnh sau khi giải nén cũng phải được xem xét.
- Đánh giá chất lượng ảnh đầu ra bằng cách so sánh $f(x,y)$ và $\hat{f}(x,y)$.
- Tiêu chí đánh giá:
  - ***Chủ quan***: dựa vào ý kiến người quan sát
  - ***Khách quan***: dựa vào công cụ toán học để tính

Đánh giá chủ quan

Value	Rating	Description
1	Excellent	An image of extremely high quality, as good as you could desire.
2	Fine	An image of high quality, providing enjoyable viewing. Interference is not objectionable.
3	Passable	An image of acceptable quality. Interference is not objectionable.
4	Marginal	An image of poor quality; you wish you could improve it. Interference is somewhat objectionable.
5	Inferior	A very poor image, but you could watch it. Objectionable interference is definitely present.
6	Unusable	An image so bad that you could not watch it.

Đánh giá khách quan

- Root mean square error (RMS)

$$e_{rms} = \sqrt{\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(\hat{f}(x,y) - f(x,y))^2}$$

- Mean-square signal-to-noise ratio (SNR)

$$SNR_{rms} = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(\hat{f}(x,y))^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(\hat{f}(x,y) - f(x,y))^2}$$

Chuẩn nén ảnh JPEG

- Lý do ra đời của JPEG
- Sơ đồ tổng quan
- Biến đổi DCT
- Lượng tử hóa
- Mã hóa entropy

Lý do ra đời của JPEG

- Tỷ lệ nén của các phương pháp không tổn thất (ví dụ Huffman, Arithmetic, LZW) là không đủ cao đối với nén ảnh và nén video.

- JPEG sử dụng mã hóa chuyển đổi, và phần lớn dựa vào các quan sát dưới đây.

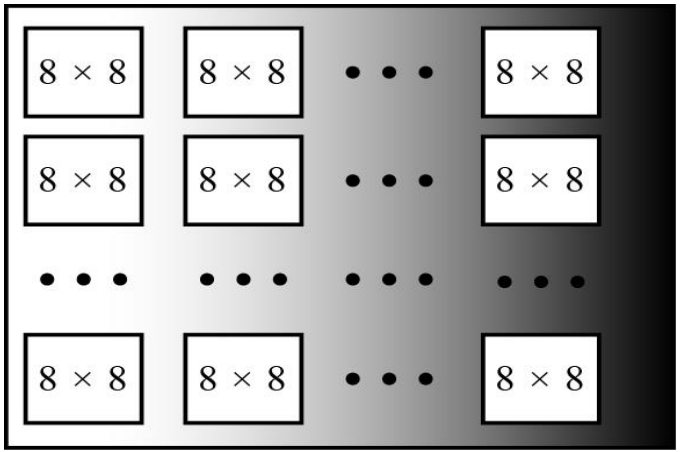
Lý do ra đời của JPEG

- ***Quan sát 1***: Phần lớn nội dung quan trọng của ảnh thay đổi tương đối chậm trên toàn bộ bức ảnh. Điều đó có nghĩa là giá trị cường độ ít khi thay đổi lên xuống nhiều lần trong một vùng nhỏ, ví dụ trong một block ảnh 8 x 8. Khi chuyển một bức ảnh sang miền tần số, các thành phần tần số thấp chứa nhiều thông tin hơn thành phần tần số cao. Thành phần tần số cao thường tương ứng với nhiễu và các chi tiết ít hữu ích.

- ***Quan sát 2***: Các thí nghiệm cho thấy mắt người khó phát hiện ra các mất mát ở vùng tần số cao so với vùng tần số thấp.

Chia ảnh

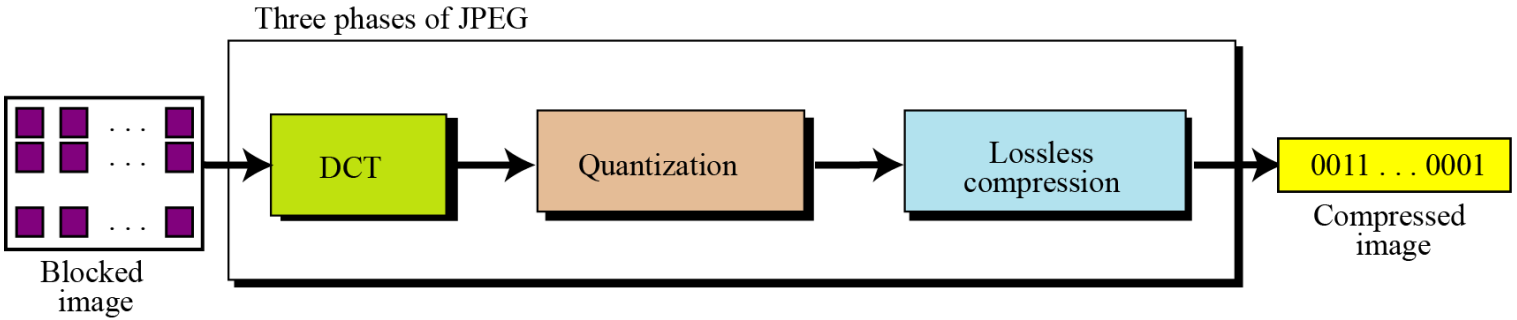
Trong JPEG, ảnh xám được chia thành các khối 8 x 8 để làm giảm số các phép tính.



Ví dụ ảnh xám kích thước 640 × 480 pixels

Qúa trình mã hóa

- Ý tưởng chính của JPEG là biến đổi ảnh thành một tập (vector) các số tuyến tính để loại bỏ sự dư thừa.
- Sự dư thừa có thể được loại bỏ sử dụng một trong các phương pháp nén không tổn thất trình bày ở trên.
- Mô hình dưới đây mô tả đơn giản quá trình mã hoá JPEG



Biến đổi cosin rời rạc

- Trong bước này, mỗi khối 64 pixel trải qua một bước chuyển đổi gọi là biến đổi cosin rời rạc DCT (**discrete cosine transform)**.
- DCT tạo ra 64 giá trị sao cho mối quan hệ tương đối giữa các pixel được giữ nguyên nhưng làm bộc lộ sự dư thừa.

1-D DCT:

$$F(\omega)=\frac{a(u)}{2}\sum_{n=0}^{N-1} f(n)\cos\frac{(2n+1)\omega\pi}{16}$$

1-D Inverese DCT:

$$f'(n)=\frac{a(u)}{2}\sum_{\omega=0}^{N-1} F(\omega)\cos\frac{(2n+1)\omega\pi}{16}$$

$$a(0)=\frac{1}{\sqrt{2}}$$
$$a(p)=1 \quad [\,p\neq 0\,]$$

Biến đổi DCT 2 chiều

Forward DCT

$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\left[\frac{\pi(2x+1)u}{2N}\right] \cos\left[\frac{\pi(2y+1)v}{2N}\right]$$

for $u = 0,\ldots,N-1$ and $v = 0,\ldots,N-1$

where $N = 8$ and $C(k) = \begin{cases} 1/\sqrt{2} & \text{for } k = 0 \\ 1 & \text{otherwise} \end{cases}$

Inverse DCT

$$f(x,y) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)F(u,v) \cos\left[\frac{\pi(2x+1)u}{2N}\right] \cos\left[\frac{\pi(2y+1)v}{2N}\right]$$

for $x = 0,\ldots,N-1$ and $y = 0,\ldots,N-1$ where $N = 8$

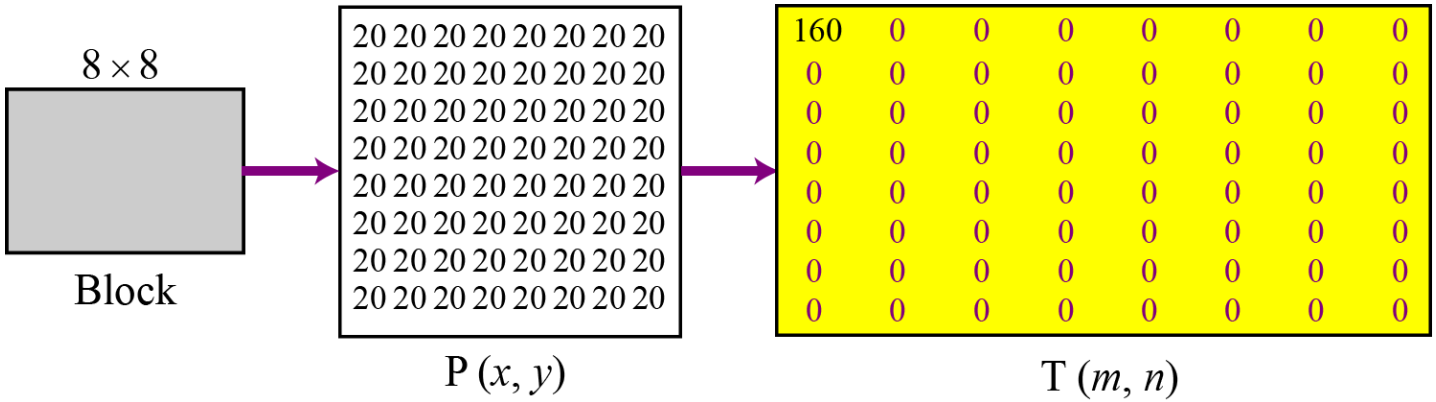
Ưu điểm của biến đổi DCT

- Biến đổi DCT có thể tập trung năng lượng của tín hiệu chuyển đổi vào tần số thấp.

- Đối với nén ảnh, DCT có thể giảm hiệu ứng blocking.

Ví dụ về DCT (1)

Ảnh có mức xám đồng nhất

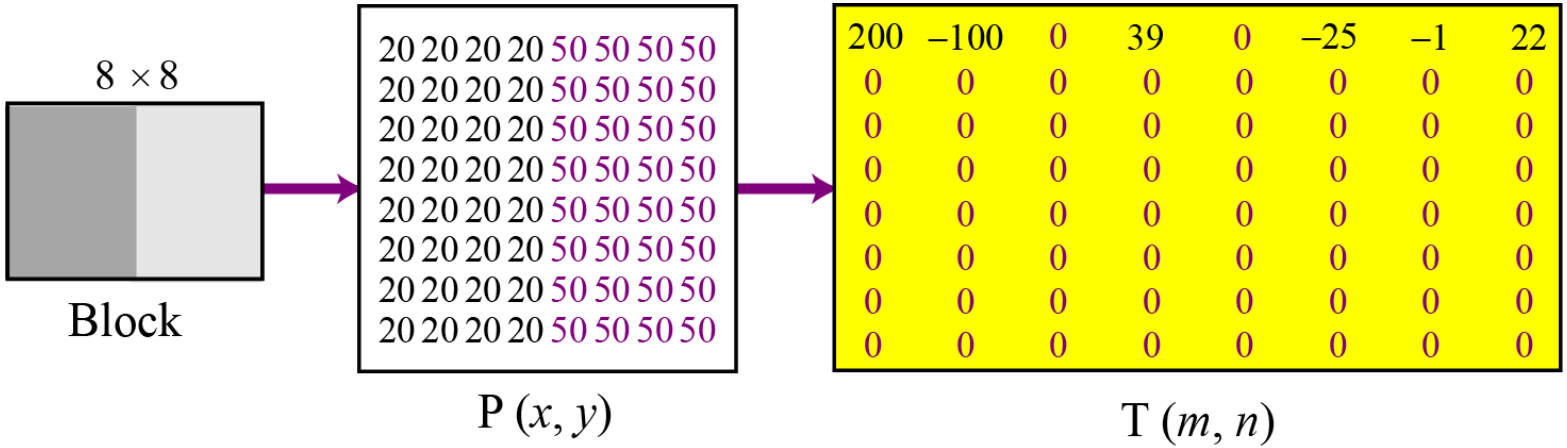


P(x,y) là giá trị trước khi biến đổi

T(m,n) là giá trị sau khi biến đổi DCT

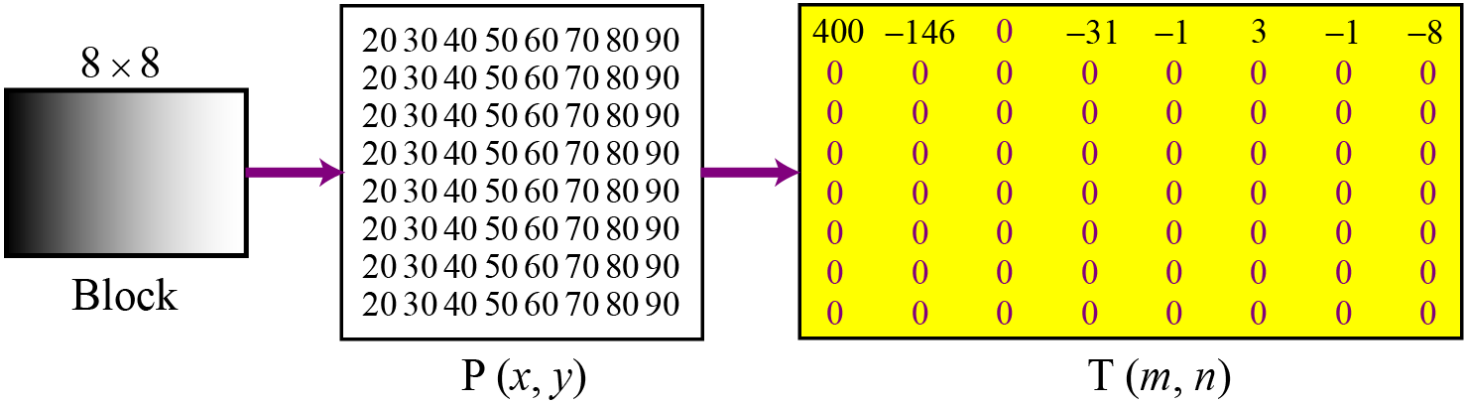
Ví dụ về DCT (2)

Ảnh gồm 2 phần



Ví dụ về DCT (3)

Ảnh có mức xám thay đổi dần dần



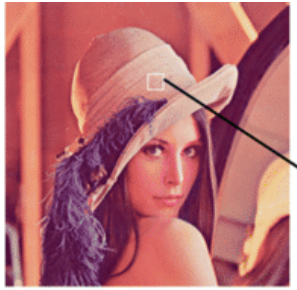
Lượng tử hóa

- Sau khi tạo bảng các giá trị T, các giá trị này được lượng tử hóa để giảm số bit mã hóa.
- Lượng tử hóa chia các giá trị trong bảng này cho một hằng số và sau đó lấy phần nguyên.
- Trong thực tế, một bảng lượng tử (8 x 8) sẽ quyết định mỗi giá trị được lượng tử hóa bao nhiêu.
- Hệ số lượng tử hóa phụ thuộc vào vị trí của giá trị trong bảng T.

Lượng tử hóa

- Lý do – Để giảm số bit trên một mẫu
  T’(u,v) = round(T(u,v)/q(u,v))
- Ví dụ: 101101 = 45 (6 bits).
  - Cắt xuống 4 bits: 1011 = 11.(So sánh 11 x 4 =44 với 45)
  - Cắt xuống 3 bits: 101 = 5. (So sánh 5 x 8 =40 với 45)
    Chú ý: Cắt càng nhiều bit, tính chính xác càng giảm.
- *Sai số lượng tử hóa là nguyên nhân chính của nén có tổn thất*.

Lượng tử hóa



Luminance Component

173	184	190	198	194	192	192	192
180	187	182	189	193	191	194	196
180	180	181	182	184	189	196	199
178	174	174	175	180	186	175	177
170	171	176	184	185	175	172	181
173	167	170	176	178	181	179	177
165	167	165	168	174	171	175	181
162	163	163	158	162	169	172	175

DCT Transformation

1430	-32	-4	-1	4	-4	0	-2
64	-2	-9	-4	-1	0	-3	-3
0	-7	3	-6	-6	0	-4	0
7	3	-16	-3	5	1	-2	-1
-12	8	-8	-2	-1	-1	4	-1
-3	0	0	2	-2	0	2	0
3	-4	2	0	-4	6	3	0
5	-4	-1	-10	4	-1	-3	2

JPEG Quantization Table

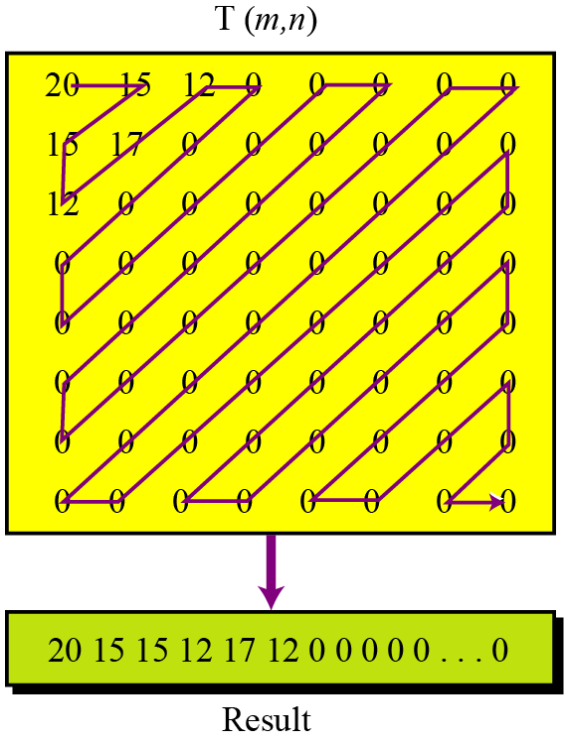
8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

Quantization

179	-2	0	0	0	0	0	0
4	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	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

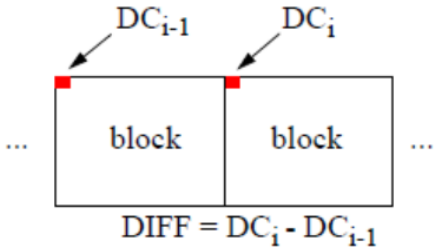
Lượng tử hóa

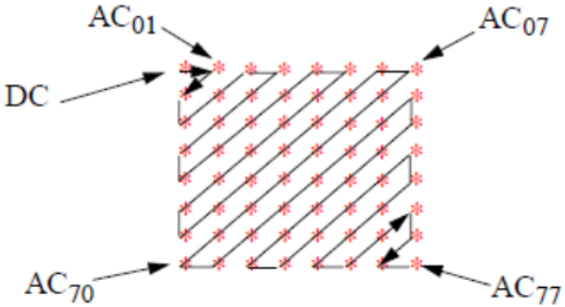
- Sau khi lượng tử hóa, các giá trị trong bảng được đọc theo đường zig-zag chứ không phải theo hàng hay theo cột.
- Mục đích là để nhóm các bit 0 lại với nhau để phục vụ cho quá trình mã hóa entropy ở bước tiếp theo (RLE).



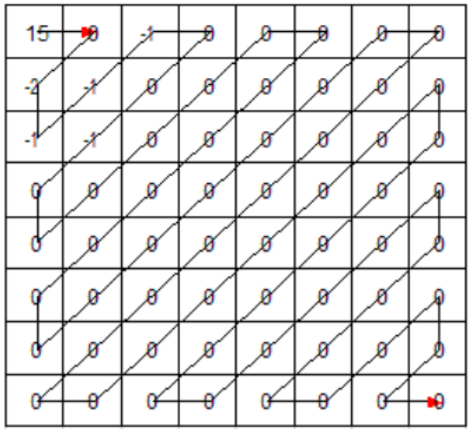
Mã hóa entropy

- Các hệ số DC được mã hóa bằng mã Điều chế xung mã vi sai DPCM.
- Các hệ số AC được mã hóa bằng mã RLE.
- Sau đó các hệ số đã được mã hóa này được tiếp tục mã hóa bằng mã hóa Entropy (mã Huffman).



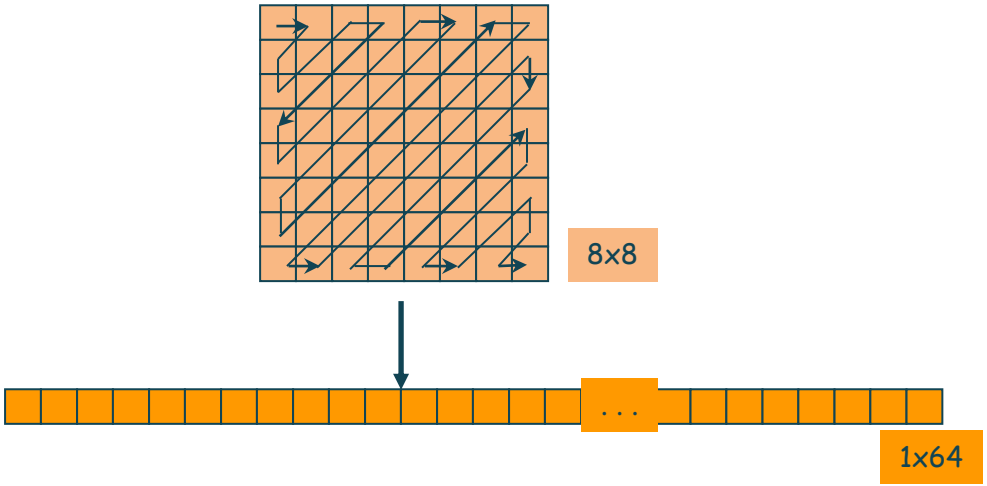


16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99



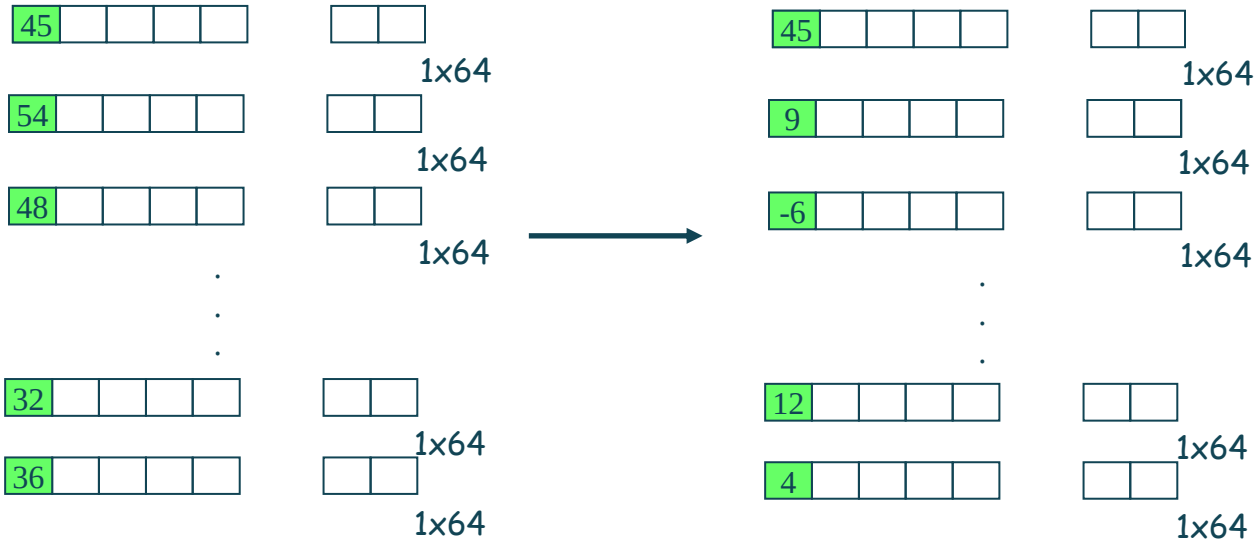
Quét zig-zag

- Lý do – để nhóm các hệ số tần số thấp vào phần đầu của vector và hệ số tần số cao vào phần sau.
- Ánh xạ ma trận 8 x 8 thành vector 1 x 64



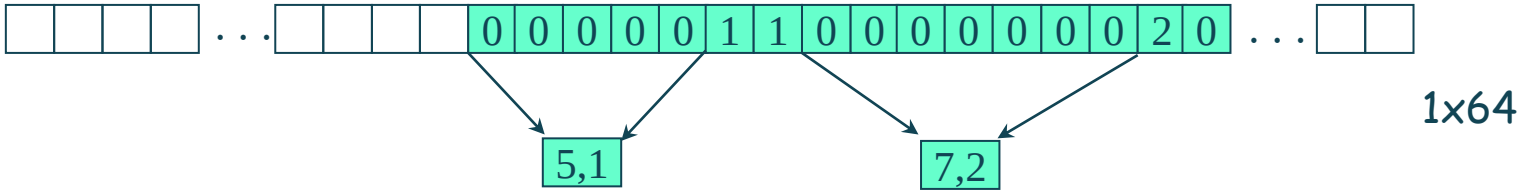
Mã DPCM cho thành phần DC

- Gía trị thành phần DC trong mỗi khối 8x8 là lớn và thay đổi tùy theo khối nhưng nói chung khá gần thành phần DC của khối trước.
- Điều chế xung mã vi sai DPCM: Mã hóa sự khác nhau giữa khối 8 x8 hiện thời và khối trước đó. Nên nhớ rằng, số càng nhỏ thì cần càng ít bit.



Mã RLE cho thành phần AC

- Vector 1x64 có nhiều bit 0 và càng nhiều ở phía cuối của vector.
  - Gía trị của vector ở các thành phần tần số càng cao có xu hướng chứa càng ít nội dung.
  - Có thể coi như kết quả của việc sử dụng bảng lượng tử.
- Mã hóa một chuỗi các bit 0 với cặp (***skip,value***), ở đó *skip* là số các bit 0 và *value* là thành phần khác 0 kế tiếp.
  - Gửi (0,0) là giá trị kết thúc khối.



Kết luận

Trong phần này, chúng ta đã được giới thiệu về:

- Mã RLC
- Mã Huffman
- Mã LZW
- Chuẩn JPEG