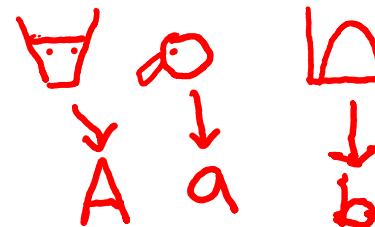


## VII. Data Compression (A)

◆壓縮的通則：

利用資料的一致性



資料越一致的資料，越能夠進行壓縮

### [References]

- 酒井善則，吉田俊之原著，原島博監修，白執善編譯，“影像壓縮術”，全華印行，2004.
- 戴顯權，“資料壓縮 Data Compression,” 旗標出版社，2007.
- I. Bocharova, *Compression for Multimedia*, Cambridge, UK, Cambridge University Press, 2010.
- D. Salomon, *Introduction to Data Compression*, Springer, 3<sup>rd</sup> ed., New York , 2004.

## ◎ 7-A 壓縮的哲學：

### (1) 利用資料的一致性，規則性，與可預測性

(exploit redundancies and predictability, find the compact or sparse representation)

(2) 通常而言，若可以用比較精簡的自然語言來描述一個東西，那麼也就越能夠對這個東西作壓縮

Q: 最古老的壓縮技術是什麼？

Ans: 自然語言

entropy 越低 >> 資料越一致 >> 越可以做壓縮

entropy (亂度；熵)

(3) 資料越一致，代表統計特性越集中

coding length for each input =  $\frac{\text{entropy}}{\log 2}$

包括 Fourier transform domain, histogram, eigenvalue ..... 等方面的

集中度  $\text{entropy} = \sum_{i=1}^s -P_i \log P_i$        $P_1, P_2, \dots, P_s$  the probability  
for each case

如果教室裡都是白色的椅子，  
P1 是教室裡的椅子是白色的  
機率 = 1

(i) When  $P_1 = 1, s = 1$ , entropy = 0

(ii) When  $P_1 = 0.5, P_2 = 0.5$ , entropy =  $-0.5 \log(0.5) - 0.5 \log 0.5 = 0.5 \times \log 2 \times 2 = \log 2 = 0.693$

可用 1 bit 來表示  
兩種椅子的顏色

(iii) When  $P_1 = P_2 = P_3 = P_4 = 0.25$ , entropy =  $4 \times (-0.25 \log 0.25) = \log 4 = 2 \log 2 = 1.3862$

- entropy 越低 >> 編碼量就越少

用 4 bit 來表示四  
種椅子的顏色

Compression rate 的差異是因為 dimension, 1D >> 可以用來做壓縮的資訊比較少 (音訊之間前後比較沒有關聯，只有頻率上的一致性，因此音訊如果是音樂訊號，可以壓縮的依據只有集中在少數幾個頻率，語音訊號除了基頻的整數倍以外也不為零，所以不能只記錄 f\_s 的整數倍)

276

ex: 對一個影像而言，如果一個點是白色的，他的四周也很可能是白色的，如果不是在 edge，預測通常很準確

video 又比 image  
多利用了時間上的一致性

Data type	Compression technique	Compression rate
Audio 1D	MP3 (MPEG3) 一部分	*.mp3 1/3
Image 2D	JPEG	*.jpg gray : 1/10 color : 1/20 (due to 4:2:0) 因為只有彩色影像有 R, G, B 的值，所以才能用 4:2:0 壓縮
Video 3D	MPEG4 (H.264) H.265, AVI... *.wmv *.mov	*.mpg *.mpeg *.mp4 *.avi gray : 1/30 color : 1/60

For a video, there are 30 frames per second 視覺暫留 1/24  
 without compression  
 $60 \times 30 \times 1M \times 3 \times 8 / 8 \text{ (bytes)}$   
 $= 486G$   
 $486G / 60 = 8.1G$

假設一個視訊影像是  $1000 \times 1000 = 1M$ , 假設是彩色的所以要  $\times 3$ , 一個值要用 8 bit

思考：如何對以下的資料作壓縮

Article: 常用字；常用字母

e, t, a, ... x, q, z  
. - .

比較常用的字母 a, e 用比較短的編碼

音訊在做 FT 之前沒有一致性，但做完 FT 就會集中在  $f_0$  的整數倍，就會有一致性

Song: (i) 能量集中在  $f_0, 2f_0, 3f_0, \dots$  (ii)  $f_0: f_{p_0} \times 2^{\frac{k}{12}}$  (iii) interval: b, 2b, 4b,  $\frac{1}{2}b$   
(iv) repeated melody (v) for a note, the frequency is almost fixed

Cartoon or Mark: (i) uniform color within a region \* gif  
(ii) boundary can be approximated by lines or curves

基頻出現的位置大部分都會在 Do

的頻率的  $2^{\{k/12\}}$  處

音符的長度大部分是一拍，如果一拍是 0.5 秒，那除了一拍以外，兩拍 = 1 秒、四拍 = 2 秒的長度出現的頻率較高

Compression: Original signal → Compact representation + residual information

## ◎ 7-B Compression for Images

- 影像的「一致性」：

Space domain: 每一點的值，會和相鄰的點的值非常接近

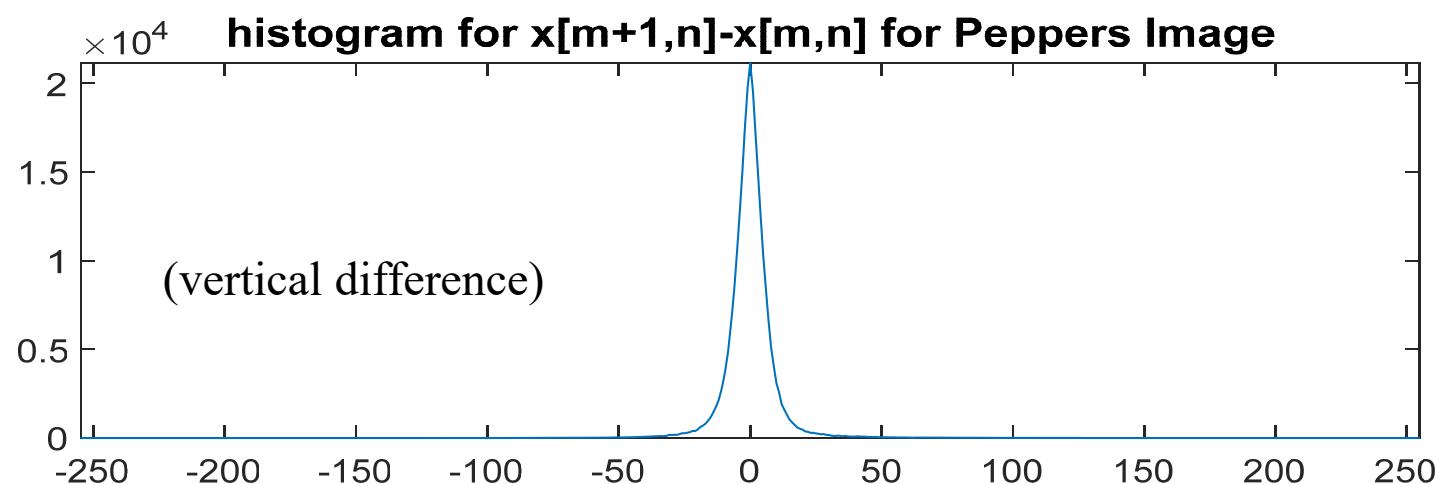
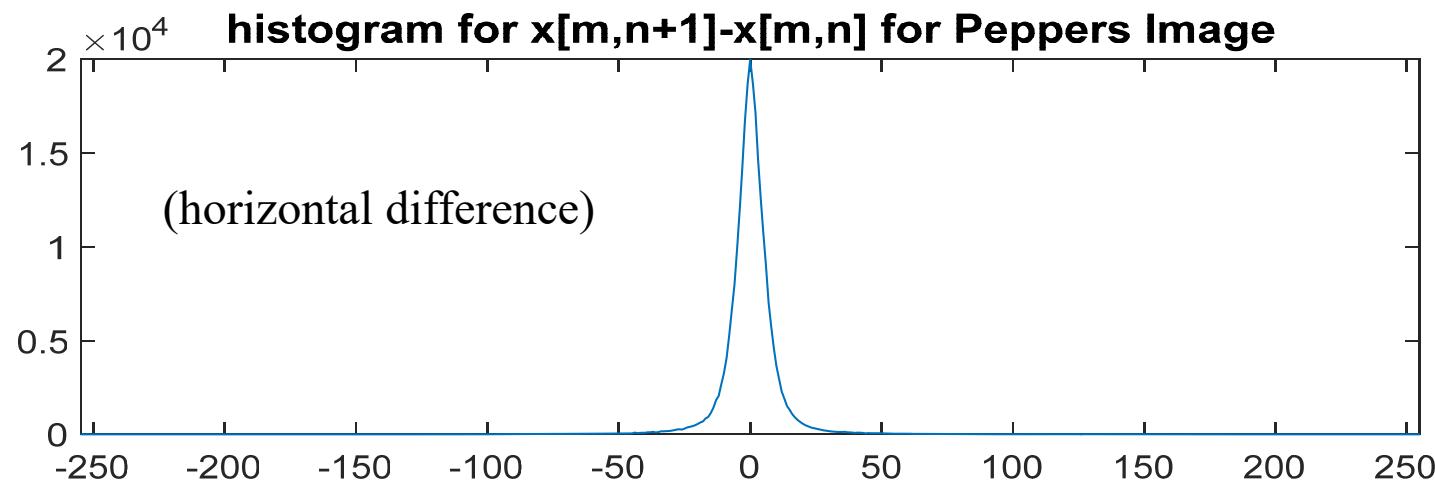
$$F[m, n+1] \approx F[m, n], \quad F[m+1, n] \approx F[m, n]$$

Frequency domain: 大多集中在低頻的地方。

## Peppers Image 在 space domain 上的一致性

左右兩點相減的值大多集中在零附近

&gt;&gt; 空間上高度的一致性



Histogram:

一個 vector 或一個 matrix 當中，有多少點會等於某一個值

例如： $x[n] = [1\ 2\ 3\ 4\ 4\ 5\ 5\ 3\ 5\ 5\ 4]$

則  $x[n]$  的 histogram 為

$$h[1] = 1, h[2] = 1, h[3] = 2, \quad h[4] = 3, h[5] = 4$$

~~Peppers~~

影像：同時有空間和頻率的一致性，因此有比較高的壓縮率

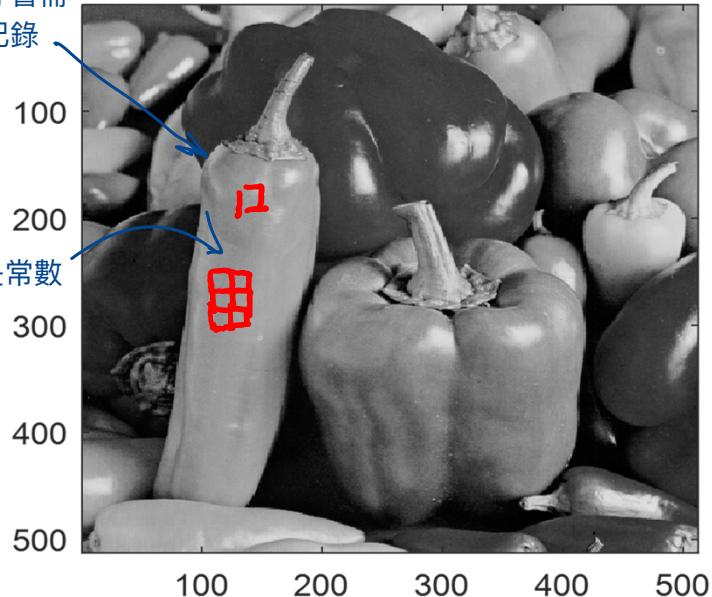
~~Lena Image 頻譜 (frequency domain) 的一致性~~

我們取絕對值，但原本有實數和虛數部分，要記錄這兩個部分就比較不利於壓縮

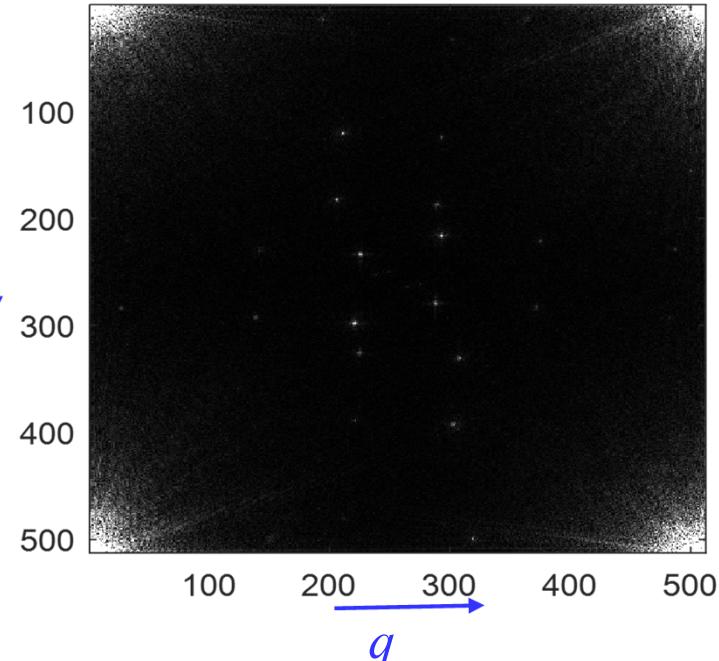
$$L_F[p, q] = |\text{fft2}(L[m, n])|$$

(用亮度來代表 amplitude)

只有碰到邊緣的地方才會需要比較多的資料量來記錄

 $L[m, n]$ 

某些方格的值可能是常數



只需要記錄少數幾個低頻的地方，就能記錄原來的影像

$$L_F[p, q] = \text{fft2}\{L[m, n]\} = \sum_{m=1}^M \sum_{n=1}^N L[m, n] e^{-j2\pi \frac{pm}{M}} e^{-j2\pi \frac{qn}{N}}$$

$$L_F[p, q] = L_F[p + M, q] = L_F[p, q + N] = L_F[p + M, q + N]$$

## 影像的「頻率」：frequency in the space domain

discrete fourier transform 的 kernel

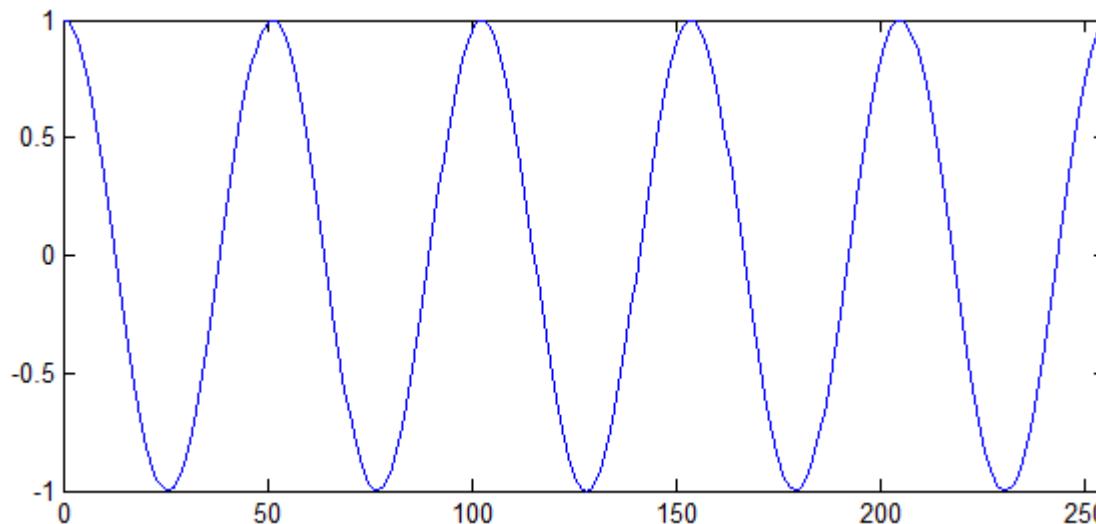
$$e^{j2\pi \frac{pm}{M}}$$

點數

從  $m = 0$  至  $m = M-1$  之間有  $p$  個週期

$$p = 5$$

$$\text{Re}\{e^{j2\pi \frac{pm}{M}}\}$$



larger  $p$  : more variation in the space domain

## ◎ 7.C JPEG Standard

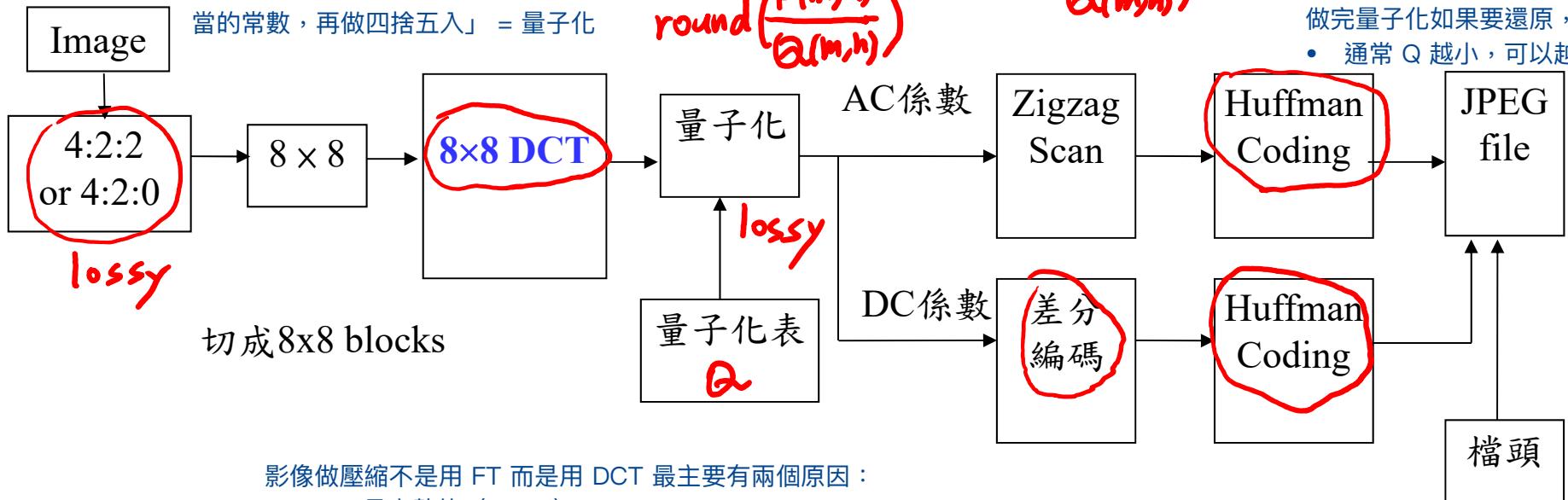
### Process of JPEG Image Compression

因為做完 DCT 是 float，所以我們會「除適當的常數，再做四捨五入」 = 量子化

$$\text{round}\left(\frac{F(m,n)}{Q(m,n)}\right)$$

ex: if  $F(m,n)=100.3, Q(m,n)=16$   
 $\text{round}\left(\frac{F(m,n)}{Q(m,n)}\right)=6, F(m,n)=6 \times 16=96$

做完量子化如果要還原，就直接乘 Q  
 • 通常 Q 越小，可以越精準的還原



影像做壓縮不是用 FT 而是用 DCT 最主要有兩個原因：

1. DCT 是實數的 (p.294)
- 2.

- 主要用到四個技術：
  - (1) 4:2:2 or 4:2:0 (和 space domain 的一致性相關)
  - (2)  $8 \times 8$  DCT (和 frequency domain 的一致性相關)
  - (3) 差分編碼 (和 space domain 的一致性相關)
  - (4) Huffman coding (和 lossless 編碼技術相關)

JPEG：影像編碼的國際標準 全名：Joint Photographic Experts Group

JPEG 官方網站：<http://www.jpeg.org/>

參考論文：G. K. Wallace, “The JPEG still picture compression standard,” *IEEE Transactions on Consumer Electronics*, vol. 38, issue 1, pp. 18-34, 1992.

JPEG 的 FAQ 網站：<http://www.faqs.org/faqs/jpeg-faq/>

JPEG 的免費 C 語言程式碼：

<http://opensource.apple.com/source/WebCore/WebCore-1C25/platform/image-decoders/jpeg/>

一般的彩色影像，可以壓縮 20 倍。

簡單的影像甚至可以壓縮超過 30 倍。

- 壓縮的技術分成兩種

### **lossy compression techniques**

無法完全重建原來的資料

Examples: DFT, **DCT**, KLT (with quantization and truncation),

4:2:2 or 4:2:0, polynomial approximation

壓縮率較高

### **lossless compression techniques**

可以完全重建原來的資料

Examples: binary coding, Huffman coding, arithmetic coding,

Golomb coding

壓縮率較低

## ◎ 7-D 4:2:2 and 4:2:0 (for color images)

係數由醫學定義

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

R, G, B 三個顏色重要性沒有特別懸殊的差異，所以在壓縮時沒辦法特別說要壓哪一個，但是轉換成 Y, C\_b, C\_r 之後就會有明顯差異，Y 明顯比其他兩者重要（如果亮度有差別，眼睛會比較敏感），如果是顏色，眼睛比較不敏感

$R$ : red,  $G$ : green,  $B$ : blue

$Y$ : 亮度,  $C_b: 0.565(B-Y)$ ,  $C_r: 0.713(R-Y)$ ,

$4:4:4$

N	
M	Y

N	
M	$C_b$

N	
M	$C_r$

$4:2:2$

N	
M	Y

M/2	N
C_b	

M/2	N
$C_r$	

*complementary of red  
recover:  $(2n+1)^{\text{th}}$  row =  
 $C_b C_r (2n^{\text{th}} \text{ row} + (2n+2)^{\text{th}} \text{ row})/2$*

$4:2:0$   $MN + \frac{MN}{4} + \frac{MN}{4} = \frac{3}{2} MN$

50%

假設要拿掉奇數的 row，要還原的話，就把前後兩個 row 的值加起來平均

>> 不會完全還原

>> 但眼睛對  $C_b, C_r$  比較不敏感，就算有誤差也比較看不出來

N/2	
M/2	$C_b$

N/2	
M/2	$C_r$

4: 2: 0 資料量變一半

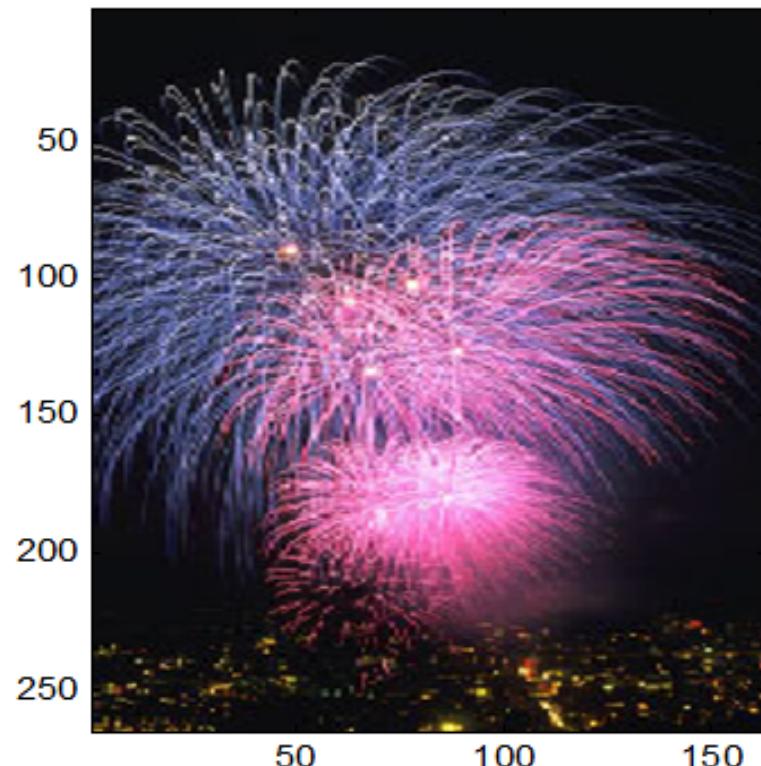
4:2:0

24 bits/pixel → 16 bits/pixel → 12 bits/pixel

同樣使資料量省一半的(b)(d)圖，(d)圖和原來差不多，  
然而(b)圖邊緣會有失真現象。

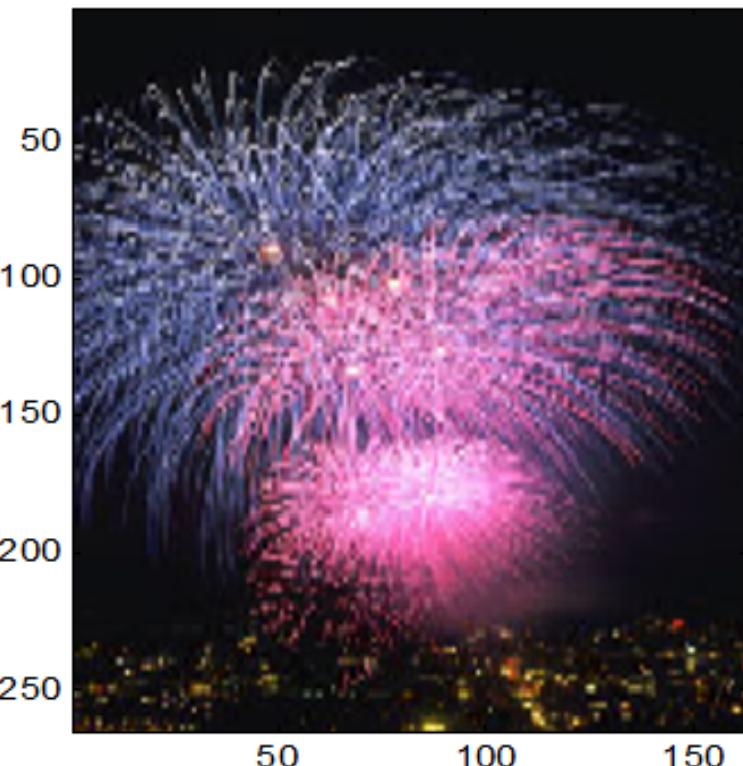
還原時，用 interpolation 的方式

原圖



(a)

直接在縱軸取一半的pixels 再還原

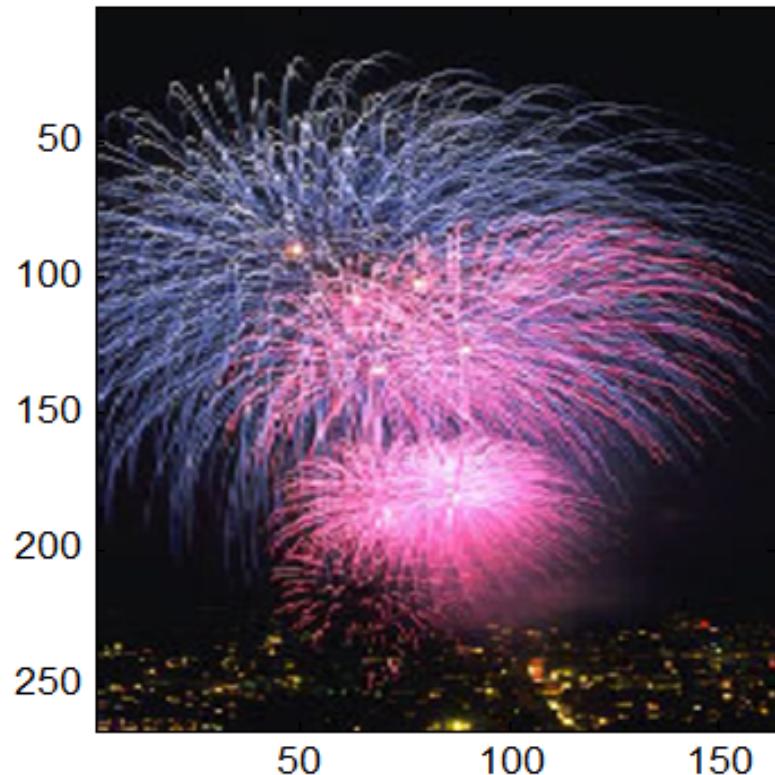


(b)

4 : 2: 2

資料量是原來的：

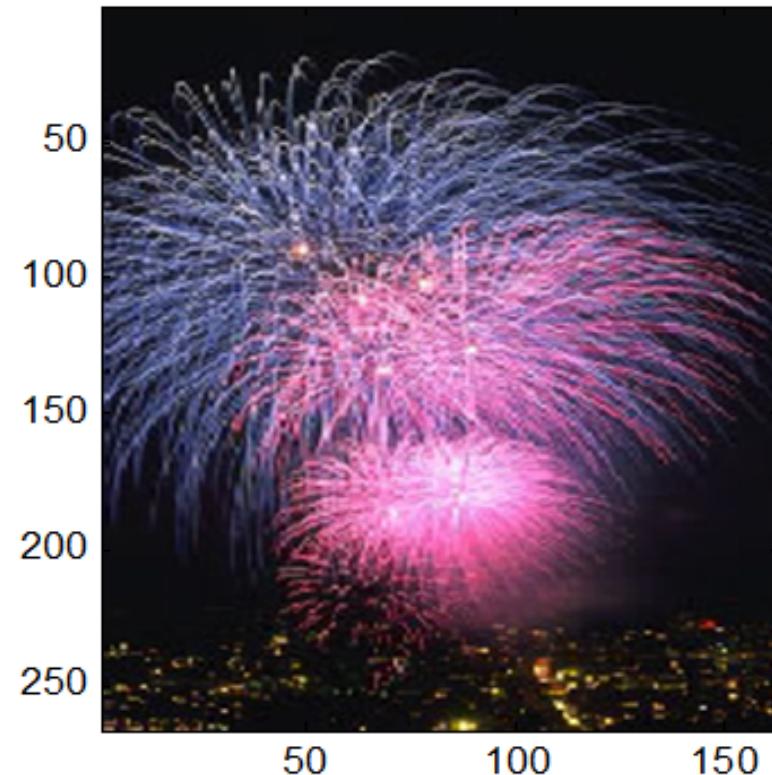
66.7%



(c)

4 : 2: 0

50%



(d)

## ◎ 7-E Optimal Transform--KLT

複習：DFT 的優缺點

- Karhunen-Loeve Transform (KLT)

(similar to Principal component analysis (PCA))

It is optimal, but dependent  
on the input

經過轉換後，能夠將影像的能量分佈變得最為集中

雖然 FT 做完 frequency 也會將在低頻的地方集中，但是 KLT 可以更集中在低頻的地方

分析影像的主要成份，第二主要成份，第三主要成份，.....

- 1-D Case  $X[u] = \sum_{n=0}^{N-1} x[n] K[u, n]$  假設最佳的轉換是  $K$

$$\begin{aligned} & \text{correlation coefficient of } X, Y \\ & = \frac{E((X-\bar{X})(Y-\bar{Y}))}{\sigma_X \sigma_Y} \end{aligned}$$

$$K[u, n] = e_n[u] \quad (K = [e_0, e_1, e_2, \dots, e_{N-1}]^T)$$

$e_n$  為 covariance matrix  $C$  的 eigenvector

$$C[m, n] = \text{cov}(x[m], x[n]) = E[(x[m] - \bar{x}[m])(x[n] - \bar{x}[n])]$$

mean

Note: cov 代表 covariance

## KLT 的理論基礎：

經過 KLT 之後，當  $u_1 \neq u_2$  時， $X[u_1]$  和  $X[u_2]$  之間的 covariance 必需近於零 (即 decorrelation)

$$\text{即 } \text{cov}(X[u_1], X[u_2]) = E[(X[u_1] - \overline{X[u_1]})(X[u_2] - \overline{X[u_2]})] = 0$$

If we set

$$\mathbf{x} = [x[0], x[1], x[2], \dots, x[N-1]]^T \quad \mathbf{X} = [X[0], X[1], X[2], \dots, X[N-1]]^T$$

and

$$\mathbf{C} = E((\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T) \quad \text{where} \quad \bar{\mathbf{x}} = E(\mathbf{x})$$

$$\mathbf{C}_X = E((\mathbf{X} - \bar{\mathbf{X}})(\mathbf{X} - \bar{\mathbf{X}})^T) \quad \text{where} \quad \bar{\mathbf{X}} = E(\mathbf{X})$$

then

$$\mathbf{C}[m, n] = \underbrace{\text{corr}}_{\substack{\text{x 經過轉換後變成 } X, X[u_1] \text{ 和 } X[u_2] \text{ 的 covariance} \\ \text{為 } 0 \text{ (如上面說明) }}, \text{ 否則還有壓縮空間}}(x(m), x(n))$$

$$\mathbf{C}_X[u_1, u_2] = \underbrace{\text{corr}}_{\substack{\text{ideal: } C_X[u_1, u_2] = 0 \text{ if } u_1 \neq u_2}}(X(u_1), X(u_2))$$

$\mathbf{C}_X$  should be a diagonal matrix.

*ideal:  $C_X[u_1, u_2] = 0$  if  $u_1 \neq u_2$*

*$C_X$  is a diagonal matrix.*

Also note that

$$\mathbf{X} = \mathbf{K}\mathbf{x}$$

$$\bar{\mathbf{X}} = \mathbf{K}\bar{\mathbf{x}}$$

$$\mathbf{C}_x = E\left((\mathbf{X} - \bar{\mathbf{X}})(\mathbf{X} - \bar{\mathbf{X}})^T\right)$$

Since  $\mathbf{X} = \mathbf{K}\mathbf{x}$        $\bar{\mathbf{X}} = \mathbf{K}\bar{\mathbf{x}}$

經過 transform 的 covariance 的矩陣

$$\begin{aligned}\boxed{\mathbf{C}_x} &= E\left((\mathbf{K}\mathbf{x} - \mathbf{K}\bar{\mathbf{x}})(\mathbf{K}\mathbf{x} - \mathbf{K}\bar{\mathbf{x}})^T\right) = E\left(\mathbf{K}(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T \mathbf{K}^T\right) \\ &= \mathbf{K}E\left((\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T\right)\mathbf{K}^T = \mathbf{K}\mathbf{C}\mathbf{K}^T\end{aligned}$$

To make  $\mathbf{C}_x$  a diagonal matrix, the KLT transform matrix  $\mathbf{K}$  should diagonalize  $\mathbf{C}$ . Suppose that the rows of  $\mathbf{K}$  are the eigenvectors of  $\mathbf{C}$  is:

$$\mathbf{C} = \mathbf{E}\mathbf{D}\mathbf{E}^T$$

$$\begin{aligned}C &= E D E^{-1} \\ \text{since } C &= C^T, E^{-1} = E^T\end{aligned}$$

where each column of  $\mathbf{E}$  is an orthonormalized eigenvector of  $\mathbf{C}$  and each diagonal entry of the diagonal matrix  $\mathbf{D}$  is the eigenvalue, then we can set

$$\underline{\mathbf{K} = \mathbf{E}^T} \quad \text{optimal } \mathbf{K}$$

Then

$$\mathbf{C}_x = \mathbf{E}^T \mathbf{E} \mathbf{D} \mathbf{E}^T \mathbf{E} = \mathbf{D}$$

is a diagonal matrix.

$\mathbf{C}$  的 eigenvector

>> 但  $\mathbf{C}$  depend on input 的東西 (ex: 用醫學影像和用卡通影像得出來的 eigenvector 不同)

- 2-D Case 
$$X[u, v] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x[m, n] K[u, m] K[v, n]$$

KLT 缺點: dependent on image

(不實際，需要一併記錄 transform matrix)

## Reference

W. D. Ray and R. M. Driver, “Further decomposition of the Karhunen-Loeve series representation of a stationary random process,” *IEEE Trans. Inf. Theory*, vol. 16, no. 6, pp. 663-668, Nov. 1970.

因為 KLT 的缺點（上一頁），我們轉為尋找 suboptimal transform，也就是 DCT

294

## ◎ 7-F Suboptimal Transform-- DCT

### • DCT: Discrete Cosine Transform

Suboptimal, but independent of  
the input

$$F[u, v] = \frac{2C[u]C[v]}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f[m, n] \cos \frac{(2m+1)u\pi}{2M} \cos \frac{(2n+1)v\pi}{2N}$$

$$C[0] = 1/\sqrt{2} \quad , C[u] = 1 \text{ for } u \neq 0$$

IDCT: inverse discrete cosine transform

$$f[m, n] = \frac{2}{\sqrt{MN}} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F[u, v] C[u] C[v] \cos \frac{(2m+1)u\pi}{2M} \cos \frac{(2n+1)v\pi}{2N}$$

對於大部分的影像而言，DCT 能夠近似 KLT (near optimal)

尤其是當  $\text{corr}\{f[m, n], f[m+\tau, n+\eta]\} = \rho^{|\tau|} \rho^{|\eta|}$ ,  $\rho \rightarrow 1$  時

有 fast algorithm

反映一個影像中距離越遠的點相關度越低，越近的點相關度越高

大部分的影像都可以用這個數學式表示  
(除非這個影像受到大量雜訊影響，才有可能點跟點之間的相關性 decay 的非常快)

Advantage: (1) independent of the input (2) near optimal (3) real output

## DCT

$$F[u, v] = \frac{2C[u]C[v]}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f[m, n] \cos \frac{(2m+1)u\pi}{2M} \cos \frac{(2n+1)v\pi}{2N}$$

$$C[0] = 1/\sqrt{2}, \quad C[u] = 1 \text{ for } u \neq 0$$

>> DC term 和 AC term 會用不同的方式做編碼

$[u, v]$  =  $[0, 0]$ : DC term

$$F[0, 0] = \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f[m, n]$$

$u \neq 0$  or  $v \neq 0$ : AC terms

DCT 的 complexity 近似  
FT 的 complexity

$N(M \log M)$

$M(N \log N)$

It can also be rewritten as

**Step 1**  $F_1[u, n] = \sqrt{\frac{2}{M}} C[u] \sum_{m=0}^{M-1} f[m, n] \cos \frac{(2m+1)u\pi}{2M}$

$m \rightarrow u$

要計算很多次 1 dimension DCT

$$n = 0, 1, 2, \dots, N-1,$$

( $N$  times of  $M$ -point 1D DCT)

**Step 2**  $F[u, v] = \sqrt{\frac{2}{N}} C[v] \sum_{n=0}^{N-1} F_1[u, n] \cos \frac{(2n+1)v\pi}{2N}$

$n \rightarrow v$

( $M$  times of  $N$ -point 1D DCT)

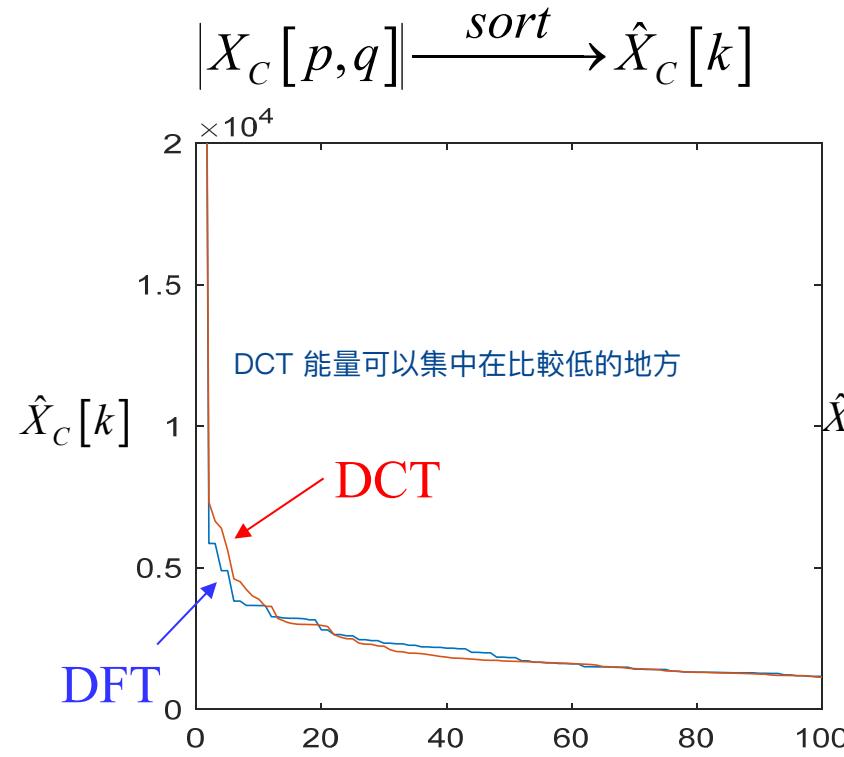
$$N(M \log M) + M(N \log N) = MN(\log M + \log N) = MN \log MN$$

左圖：將 DFT，DCT 各點能量(開根號)由大到小排序

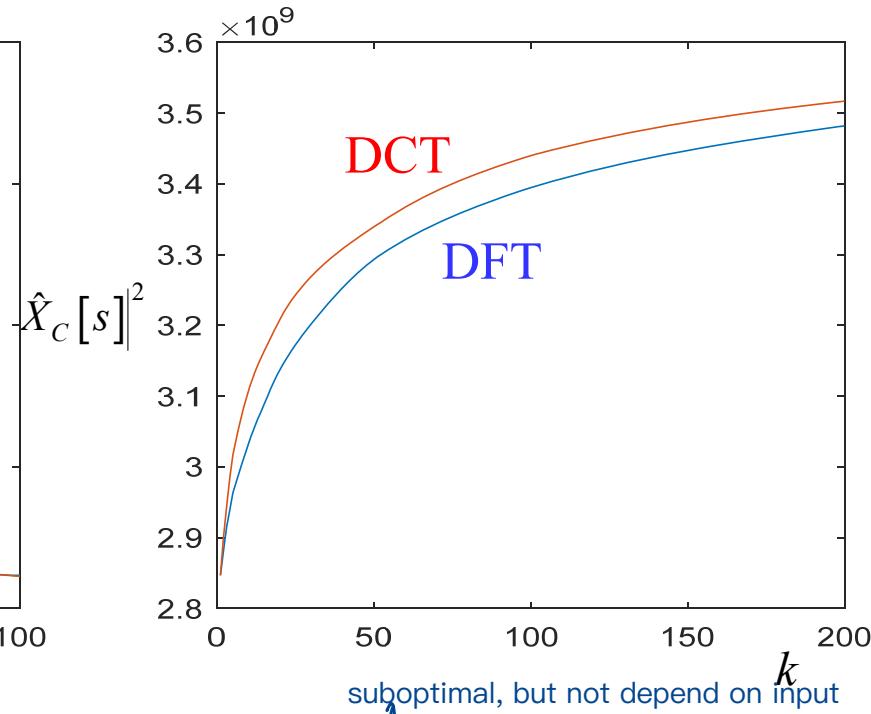
右圖：累積能量

$$\hat{X}_c[s] = \sum_{k=1}^s \hat{X}_c[k]$$

DCT output



$$\hat{X}_C[1] \geq \hat{X}_C[2] \geq \hat{X}_C[3] \geq \dots$$



Energy concentration at low frequencies: KLT > DCT > DFT

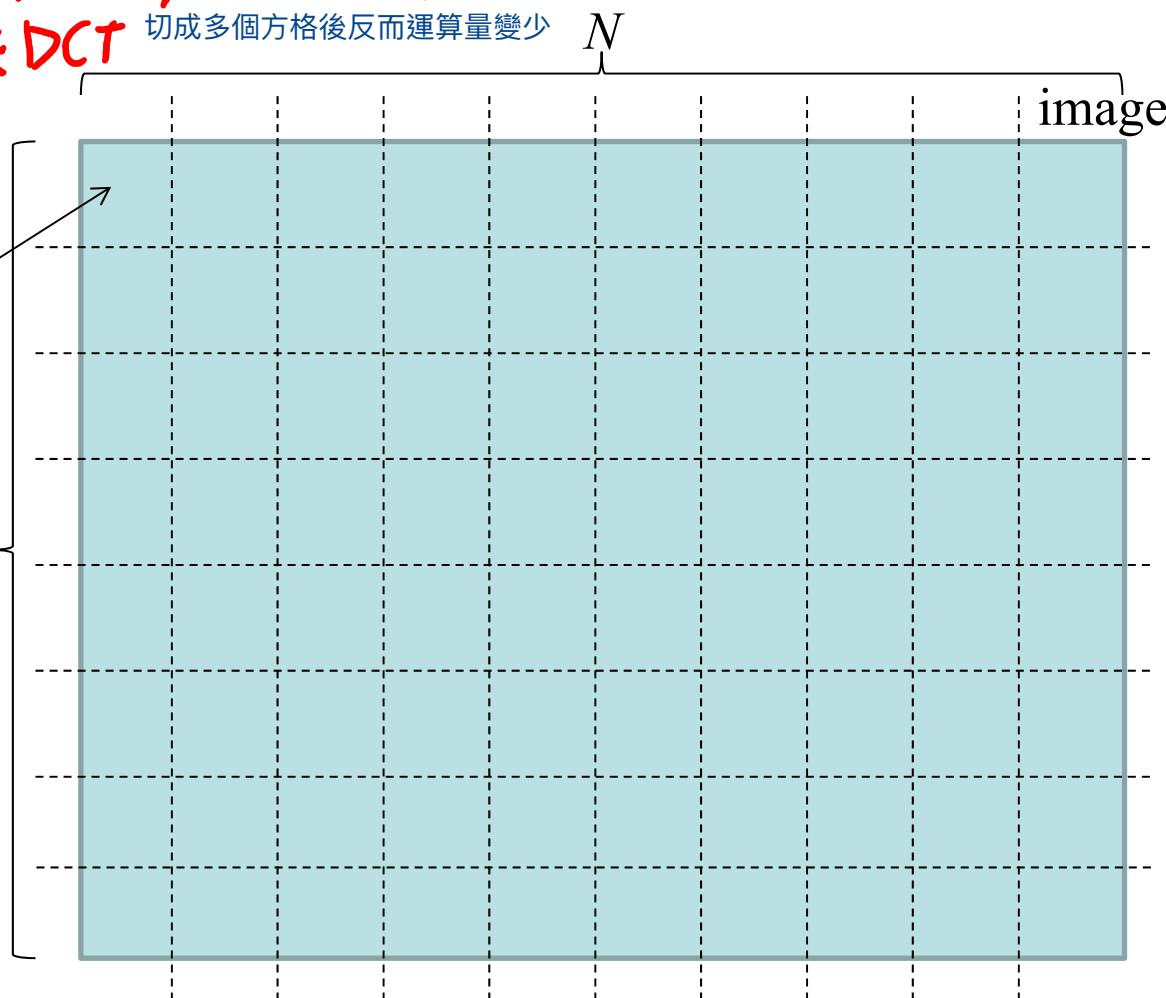
depend on input

通常，我們將影像切成  $8 \times 8$  的方格作DCT

Why:  
 (i) The frequency of an image varies with the location.  
 (ii) Saving the memory  
 (iii) less complexity  $\Theta(MN)$

complexity of  $M \times N$  point DCT

$$= MN \log(MN)$$



complexity of  $\frac{MN}{64}$  8x8 DCTs

$$= \frac{MN}{64} 64 \log 64 \quad M, N = 8 \text{ 代入}$$

$$= MN \log 64$$

complexity:  $MN$  乘上一個常數

如果壓縮是在 ex: 手機上執行 >> 影像的每個點的值都要存在暫存器中，但如果是  $8 \times 8$  暫存器就只需要放 64 個點 >>. 所以切成  $8 \times 8$  可以減少記憶體的使用量

## References

- [1] N. Ahmed, T. Natarajan, and K. R. Rao, “Discrete cosine transform,” *IEEE Trans. Comput.*, vol. C-23, pp. 90-93, Jan 1974.
- [2] K. R. Rao and P. Yip, *Discrete Cosine Transform, Algorithms, Advantage, Applications*, New York: Academic, 1990.

## 附錄十：符合人類知覺的相似度測量工具：結構相似度 Structural Similarity (SSIM)

傳統量測兩個信號 (including images, videos, and vocal signals) 之間相似度的方式：

$$(1) \text{ maximal error } \ Max(|y[m,n] - x[m,n]|)$$

$$(2) \text{ mean square error (MSE) } \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |y[m,n] - x[m,n]|^2$$

$$(3) \text{ normalized mean square error (NMSE) } \frac{\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |y[m,n] - x[m,n]|^2}{\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |x[m,n]|^2}$$

$$(4) \text{ normalized root mean square error (NRMSE) } \sqrt{\frac{\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |y[m,n] - x[m,n]|^2}{\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |x[m,n]|^2}}$$

(5)  $L_\alpha$ -Norm

$$\|y - x\|_\alpha = \left( \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |y[m, n] - x[m, n]|^\alpha \right)^{1/\alpha}$$

$$\frac{1}{MN} \|y - x\|_\alpha = \frac{1}{MN} \left( \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |y[m, n] - x[m, n]|^\alpha \right)^{1/\alpha}$$

## (6) signal to noise ratio (SNR), 信號處理常用

$$10 \log_{10} \left( \frac{\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |x[m, n]|^2}{\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |y[m, n] - x[m, n]|^2} \right)$$

## (7) peak signal to noise ratio (PSNR), 影像處理常用

$$10 \log_{10} \left( \frac{X_{Max}^2}{\frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |y[m, n] - x[m, n]|^2} \right)$$

$X_{Max}$ : the maximal possible value of  $x[m, n]$

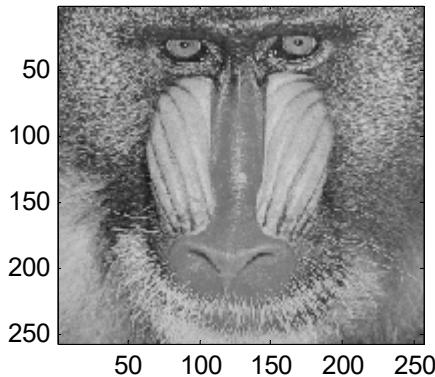
In image processing,  $X_{Max} = 255$

for color image:  $10 \log_{10} \left( \frac{X_{Max}^2}{\frac{1}{3MN} \sum_{R,G,B} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |y_{color}[m, n] - x_{color}[m, n]|^2} \right)$

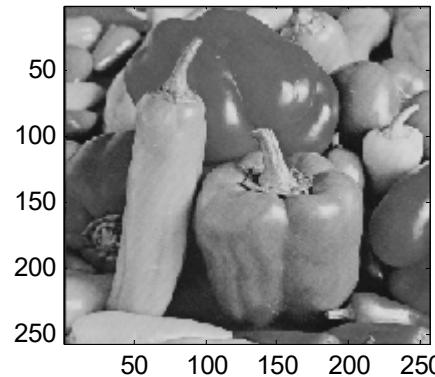
color = R, G, or B

然而，MSE 和 NRMSE 雖然在理論上是合理的，但卻無法反映出實際上兩個影像之間的相似度

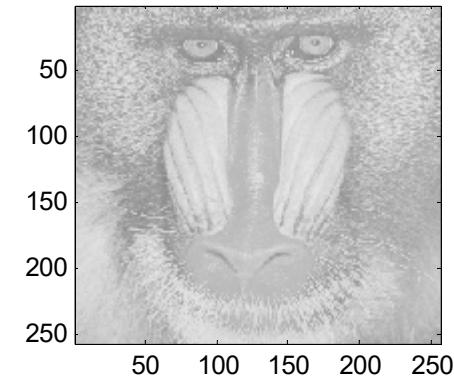
例如：以下這三張圖



圖一



圖二



圖三

$$\text{圖三} = \text{圖一} \times 0.5 + 255.5 \times 0.5$$

照理來說，圖一和圖三較相近

然而，圖一和圖二之間的 NRMSE 為 0.4411

圖一和圖三之間的 NRMSE 為 0.4460

## (8) Structural Similarity (SSIM)

有鑑於 MSE 和 PSNR 無法完全反映人類視覺上所感受的誤差，在 2004 年被提出來的新的誤差測量方法

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + (c_1L)^2)}{(\mu_x^2 + \mu_y^2 + (c_1L)^2)} \frac{(2\sigma_{xy} + (c_2L)^2)}{(\sigma_x^2 + \sigma_y^2 + (c_2L)^2)}$$

$$DSSIM(x, y) = 1 - SSIM(x, y)$$

$\mu_x, \mu_y$ : means of  $x$  and  $y$        $\sigma_x^2, \sigma_y^2$ : variances of  $x$  and  $y$

$\sigma_{xy}$ : covariance of  $x$  and  $y$        $c_1, c_2$ : adjustable constants

$$\sigma_{xy} = \frac{1}{MN} \sum_{m=1}^M \sum_{n=1}^N (x[m, n] - \mu_x)(y[m, n] - \mu_y) \quad \text{where } \mu_x = \frac{1}{MN} \sum_{m=1}^M \sum_{n=1}^N x[m, n]$$

$L$ : the maximal possible value of  $x$  – the minimal possible value of  $x$

若使用 SSIM，且前頁的  $c_1, c_2$  皆選為  $1/\sqrt{L}$

圖一、圖二之間的 SSIM 為 0.1040

圖一、圖三之間的 SSIM 為 0.7720

反映出了圖一、圖三之間確實有很高的相似度

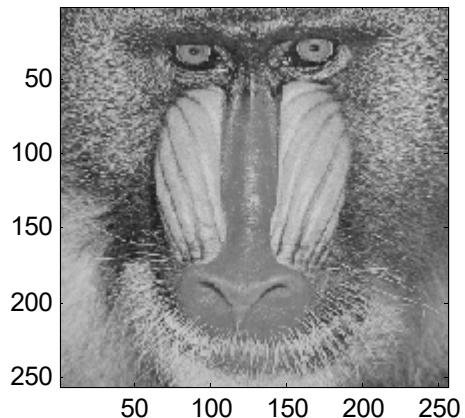
比較：機率上關於相關度 (correlation) 的定義

$$\text{corr}(x, y) = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

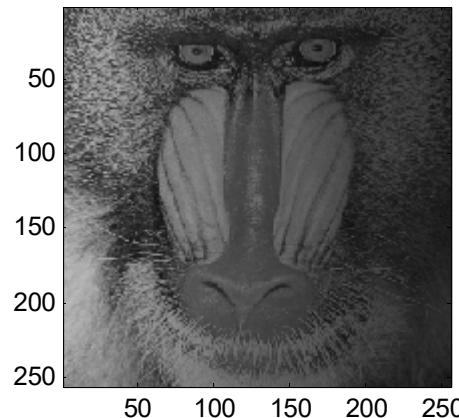
Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Trans. Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

其他幾個用 MSE 和 NRMSE 無法看出相似度，但是可以用 SSIM 看出相似度的情形

影子 shadow



圖四

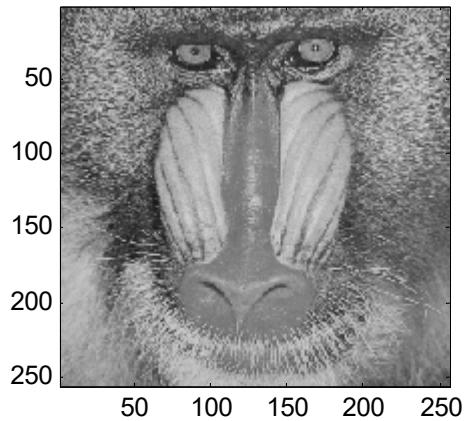


圖五

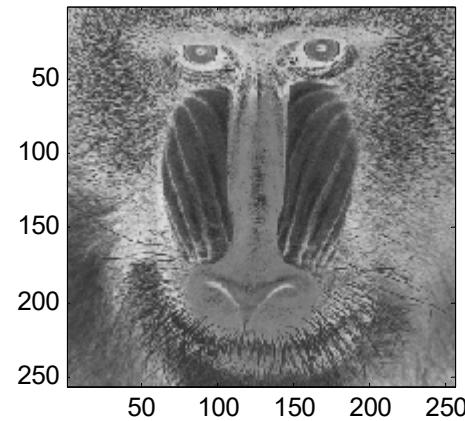
$\text{NRMSE} = 0.4521$  (大於圖一、圖二之間的 NRMSE)

$\text{SSIM} = 0.6010$

底片 the negative of a photo



圖六



圖七

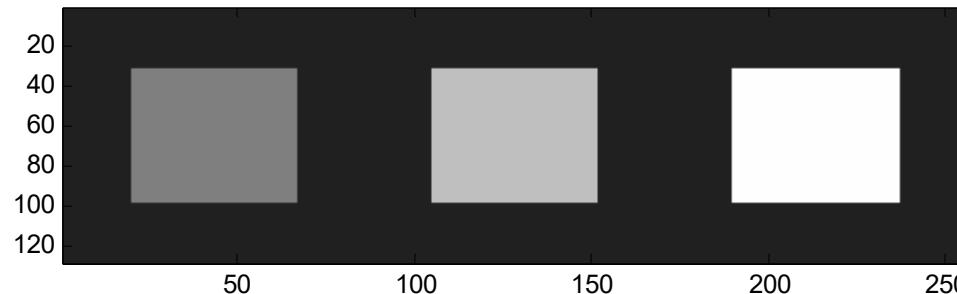
圖七 = 255 – 圖六

NRMSE = 0.5616 (大於圖一、圖二之間的 NRMSE)

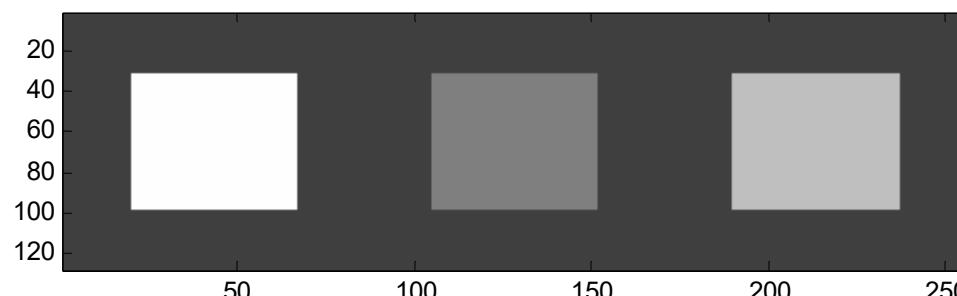
SSIM = -0.8367 (高度負相關)

同形，但亮度不同 (Same shape but different intensity)

圖八



圖九



NRMSE = 0.4978 (大於圖一、圖二之間的 NRMSE)

SSIM = 0.7333

思考：對於 vocal signal (聲音信號而言)

MSE 和 NRMSE 是否真的能反映出兩個信號的相似度？

為什麼？

### Perceptual Evaluation of Speech Quality (PESQ)

A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, “Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs,” in *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 2, pp. 749-752, May 2001.

## VIII. Data Compression (B)

### ◎ 8-A Differential Coding for DC Terms, Zigzag for AC Terms

這兩者可視為 JPEG Huffman coding 的前置工作

$$Q(0,0) = 16$$

Differential Coding (差分編碼)

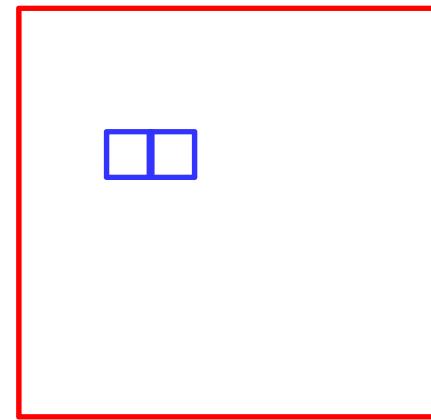
If the DC term of the  $(i, j)^{\text{th}}$  block is denoted by  $DC[i, j]$ , then

$$\text{round}\left(\frac{\text{mean} \times 8}{16}\right) = \text{round}\left(\frac{\text{mean}}{2}\right)$$

encode  $DC[i, j] - DC[i, j-1]$

這兩個值相減 = 0 或接近 0 的機率非常大

instead of  $DC[i, j]$



(也是運用 space domain 上的一致性)

## Zigzag scanning

將 2D 的 8x8 DCT outputs 變成 1D 的型態

但按照 “zigzag” 的順序 (能量可能較大的在前面)

用 zigzag 排列，越前面的值 = 0 的機率變低，越後面的值 = 0 的機率就會變高

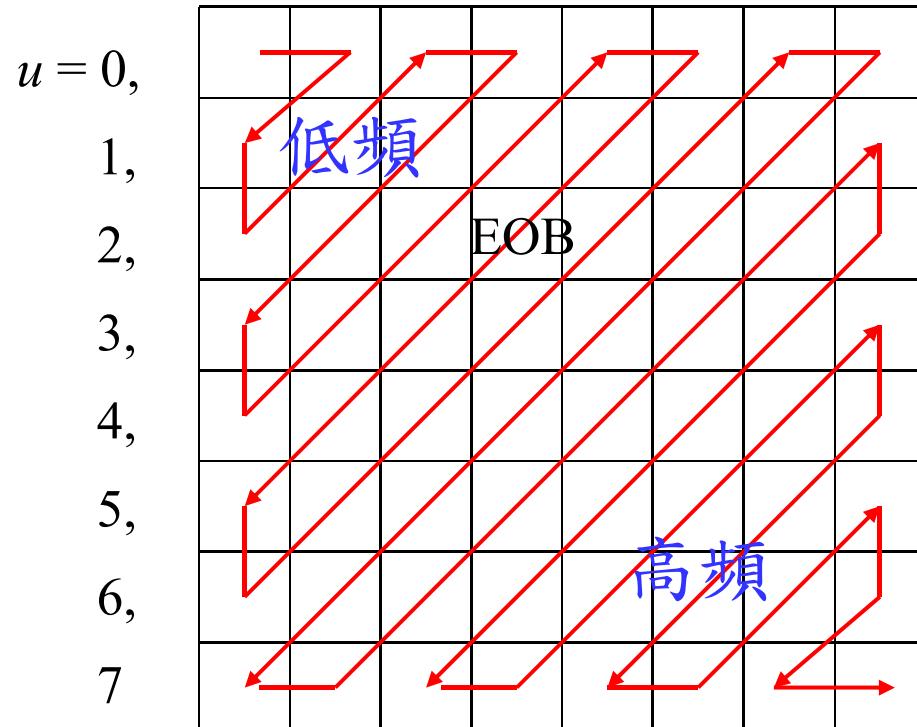
$$v = 0, 1, 2, 3, 4, 5, 6, 7$$

8x8 DCT output:

$C[u, v]$

$u = 0, 1, \dots, 7$

$v = 0, 1, \dots, 7$



EOB (end of block):  
指後面的高頻的部分經過 quantization 之後皆為 0

(也是運用 frequency domain 上的一致性)

- 4: 2: 0 或量子化都是有損壓縮 (lossy)

311

## ◎ 8-B Lossless Coding

**Lossless Coding:** The original data can be perfectly recovered

Example:

direct coding method

Huffman coding

Arithmetic coding

Shannon–Fano Coding, Golomb coding, Lempel–Ziv, .....

## ◎ 8-C Lossless Coding: Huffman Coding

- Huffman Coding 的編碼原則: (Greedy Algorithm)

leaf node

- (1) 所有的碼皆在 Coding Tree 的端點，再下去沒有分枝  
(滿足一致解碼和瞬間解碼)
- (2) 機率越大的， code length 越短；機率越小的， code length 越長
- (3) 假設  $S_1$  是第  $L$  層的 node， $S_2$  是第  $L+1$  層的 node  
則  $P(S_1) \geq P(S_2)$  必需滿足

不滿足以上的條件則將  $S_2$  往上推一層

原始的編碼方式：

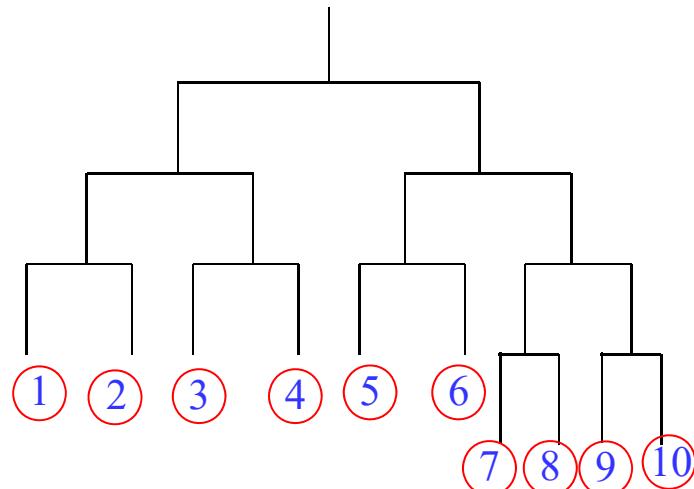
若  $\text{data}$  有  $M$  個可能的值，使用  $k$  進位的編碼，  
則每一個可能的值使用  $\text{floor}(\log_k M)$  或  $\text{ceil}(\log_k M)$  個 bits 來編碼

$\text{floor}$ : 無條件捨去

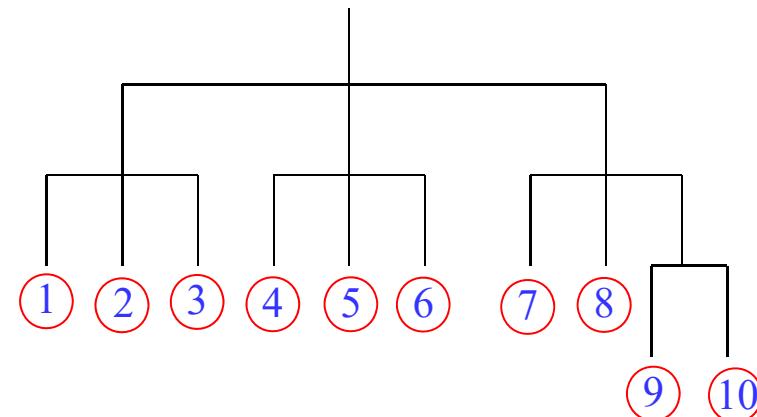
$\text{ceil}$ : 無條件進位

Example:

若有 8 個可能的值，在 2 進位的情形下，需要 3 個 bits



若有 10 個可能的值，在 3 進位的情形下，需要 2 個或 3 個 bits



因為  $10 > 2^3$  所以有些值要用四個 bit

Example: ㄉ

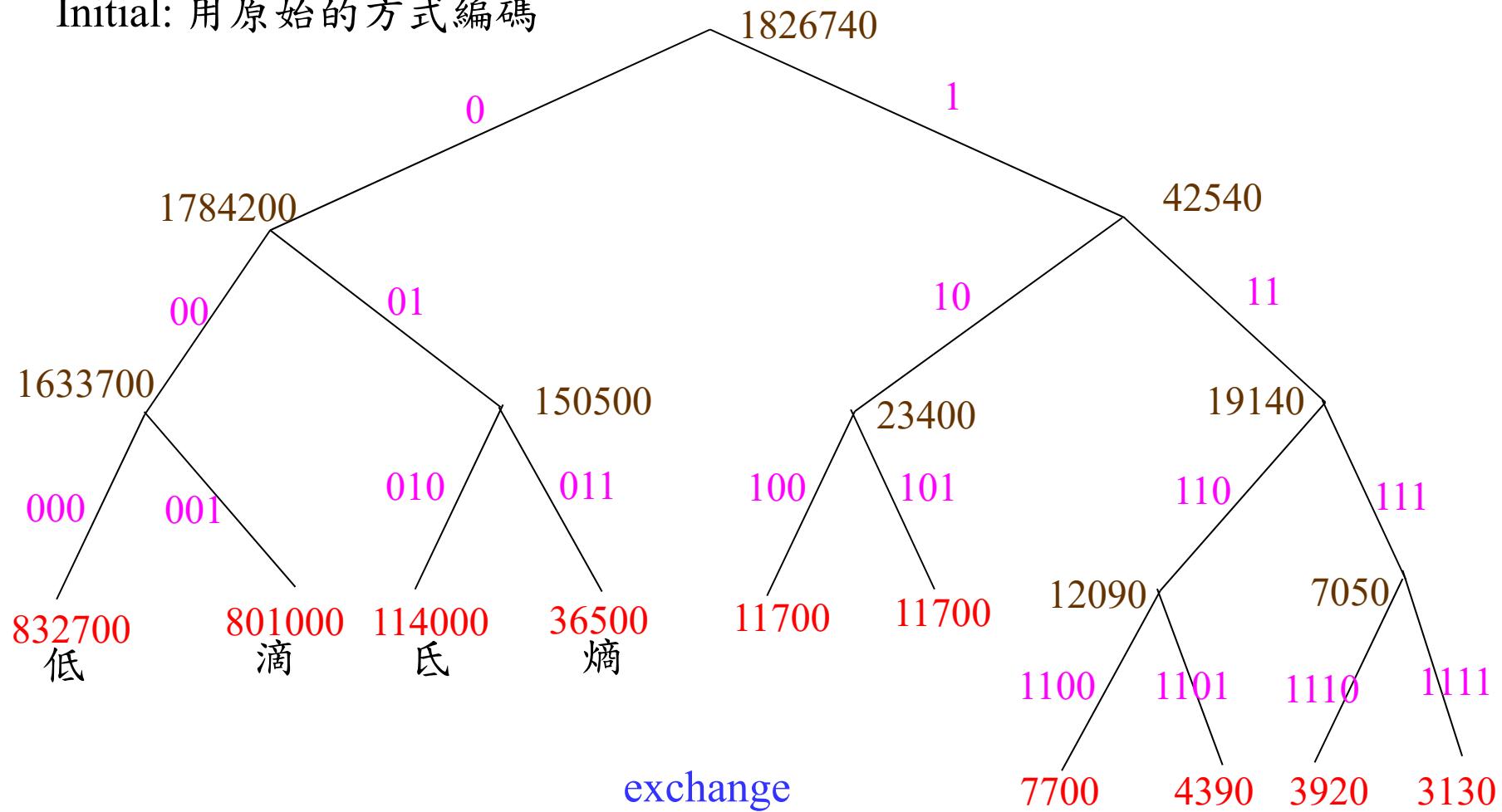
低	滴	氐	羝	鞮
ㄉ 832700	ㄉ 801000	ㄉ 114000	ㄉ 7700	ㄉ 4390
碑	祇	菂	墻	熵 <u>ㄉ</u>
ㄉ 3920	ㄉ 11700	ㄉ 11700	ㄉ 3130	ㄉ 36500

這些數字代表當初老師查詢時  
現在多少個網頁

他們 2進位的Huffman Code 該如何編

## Huffman coding

Initial: 用原始的方式編碼



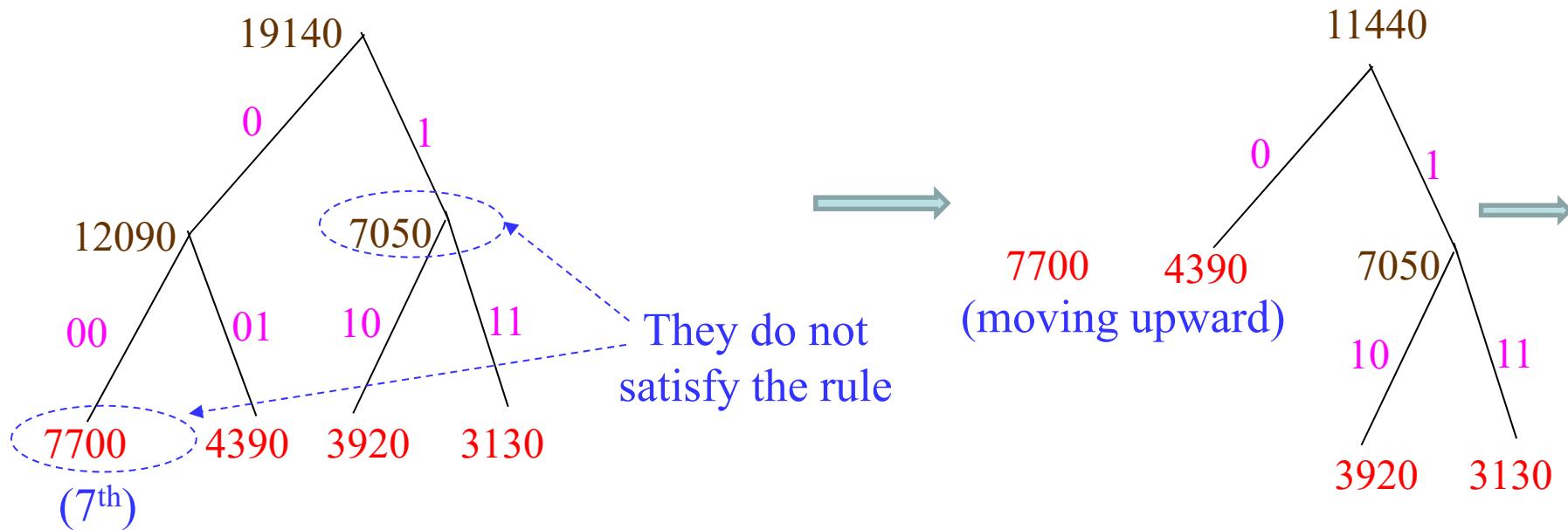
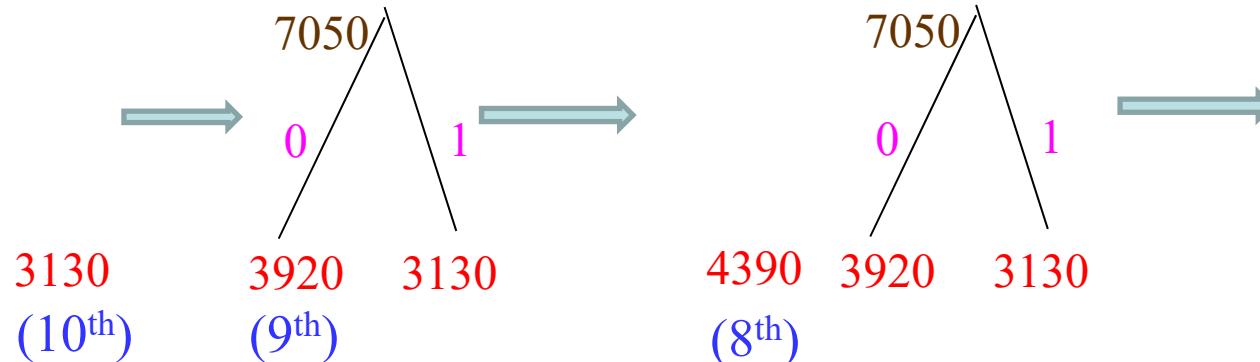
average code length = 3.0105

### The rules of the Huffman coding process.

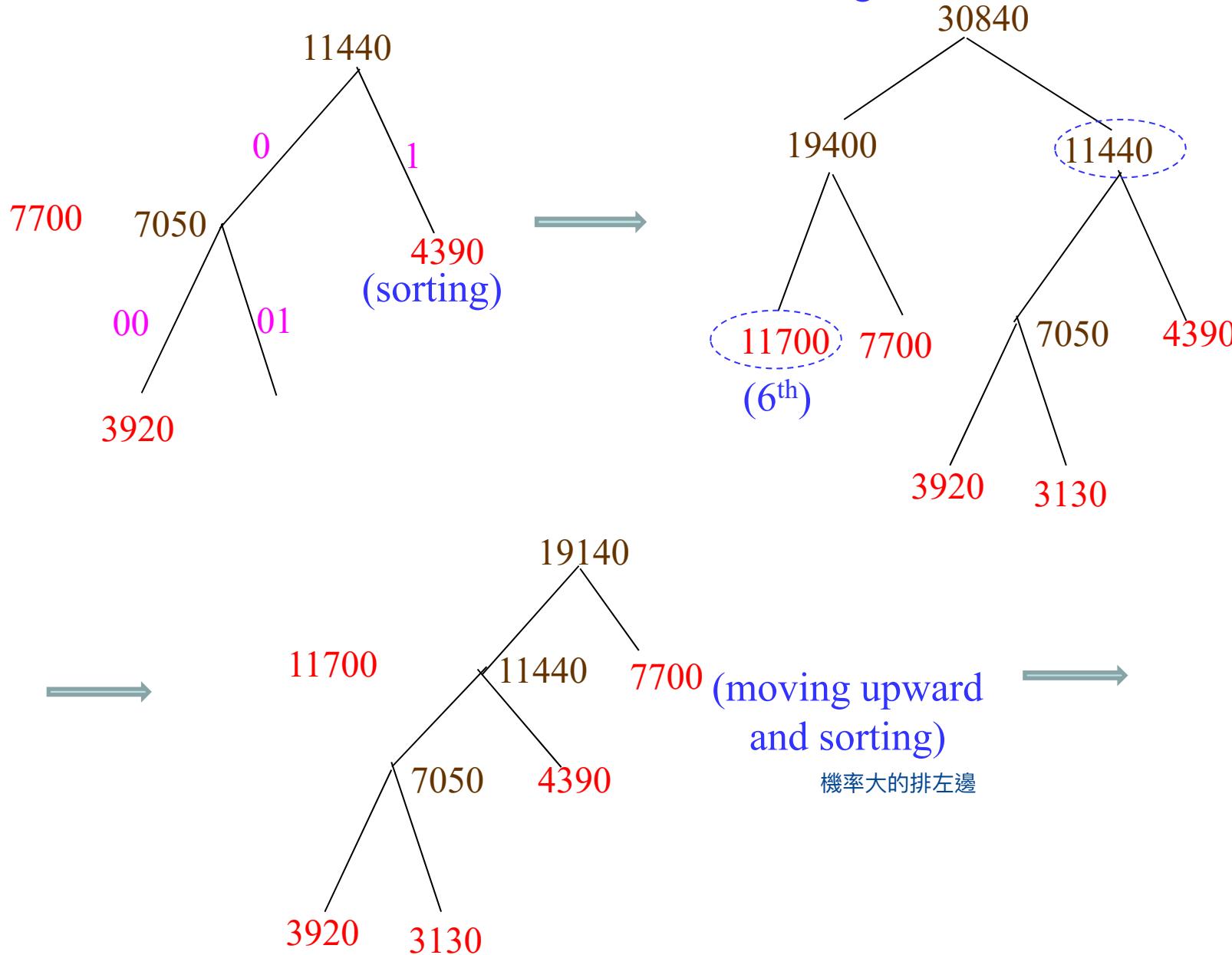
- (1) Process the case with lower probability first
- (2) If the node in the lower layer has higher probability, then move it upward.
- (3) For the node to be moved upward, if it has a partner, then move the partner, too.
- (4) Re-sort after moving upward.

## Huffman coding

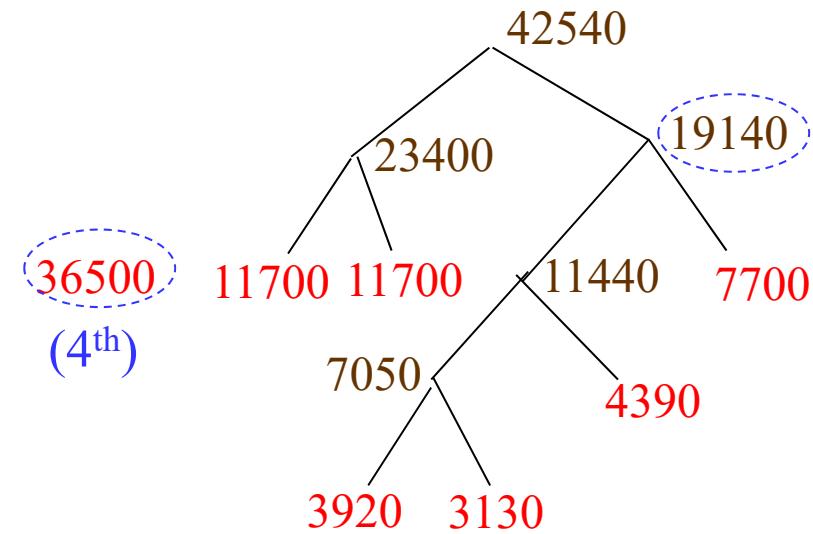
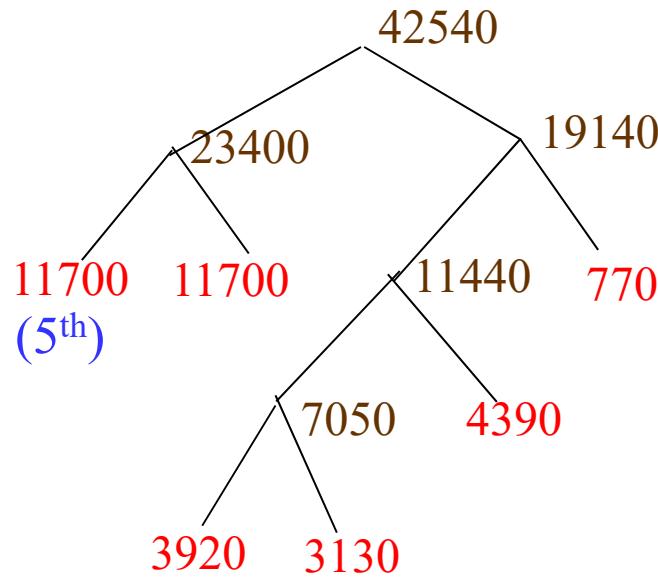
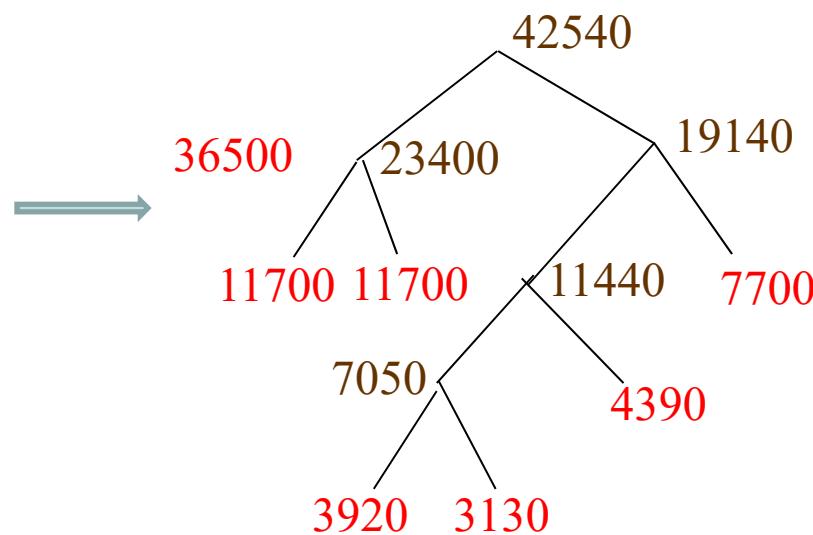
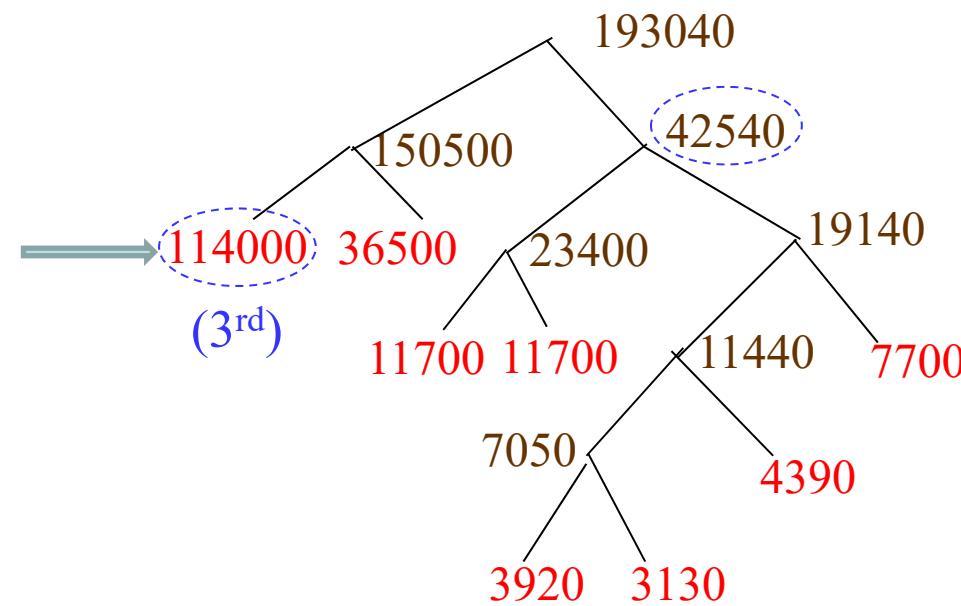
由機率低的開始編碼，一步一步加進機率高的



### Huffman coding

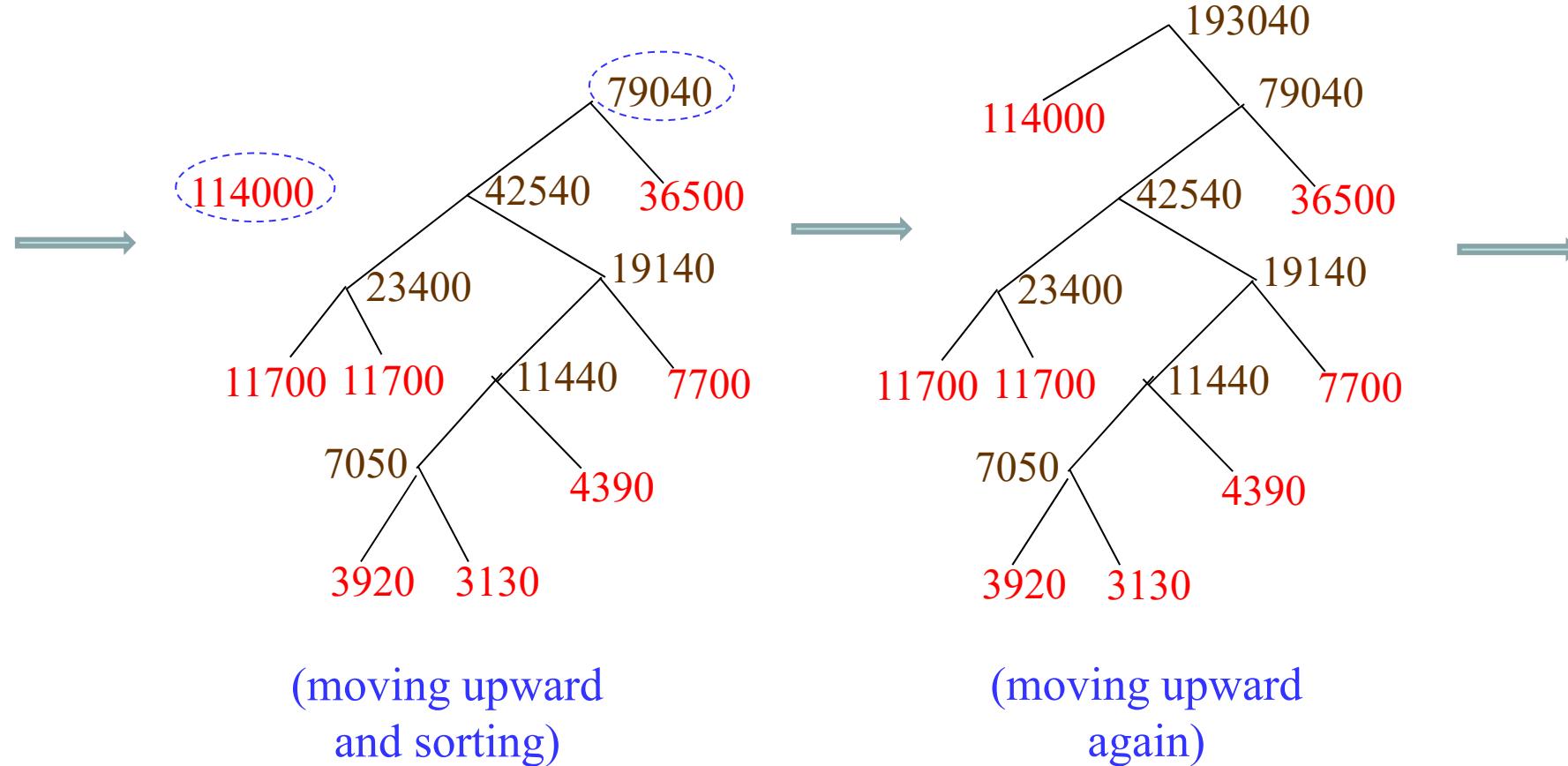


### Huffman coding

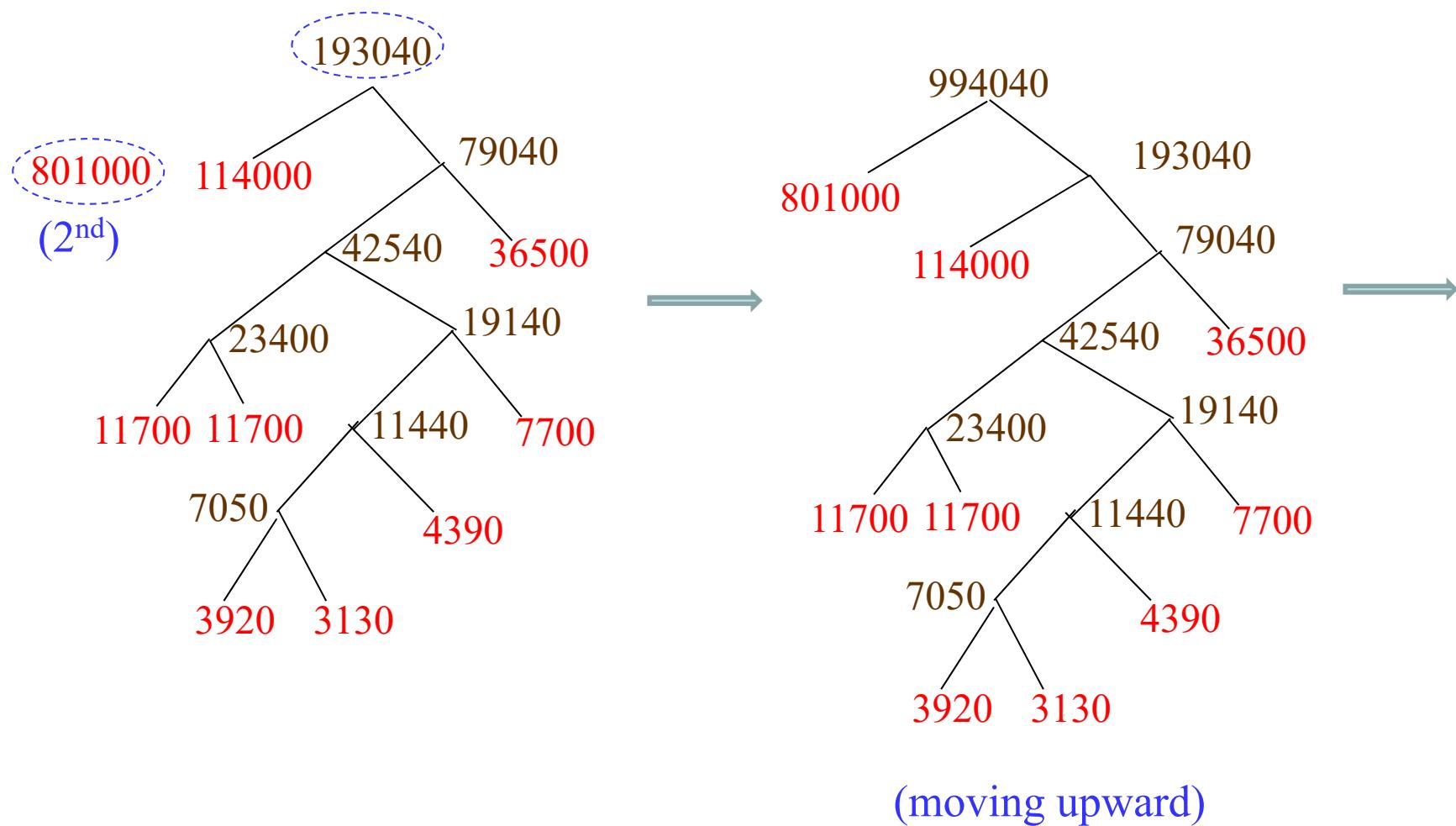
(4<sup>th</sup>)(3<sup>rd</sup>)

## Huffman coding

320

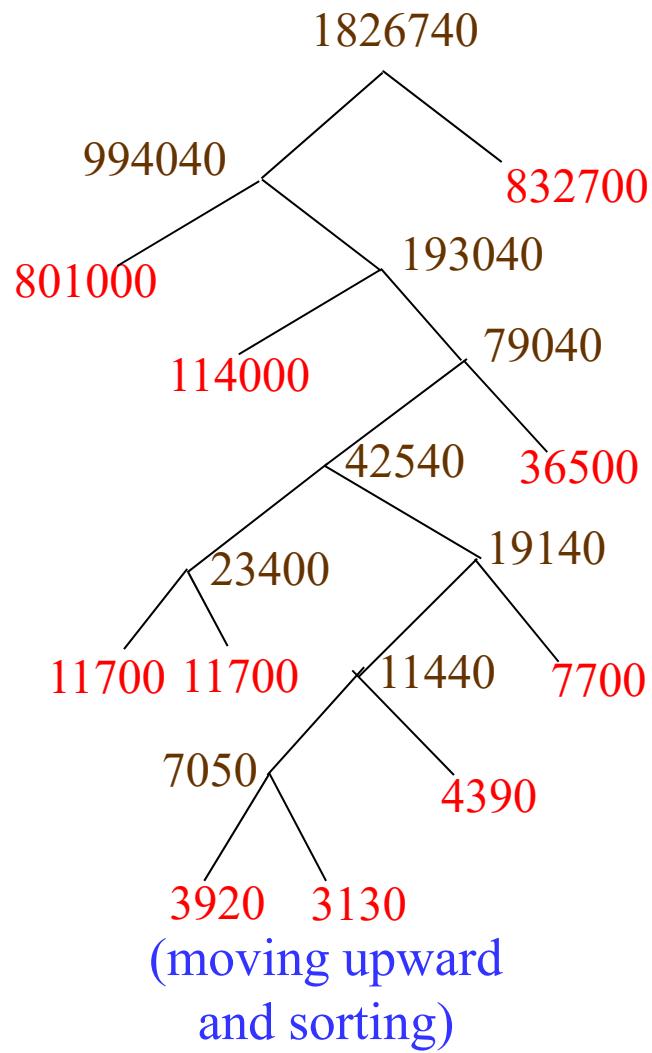
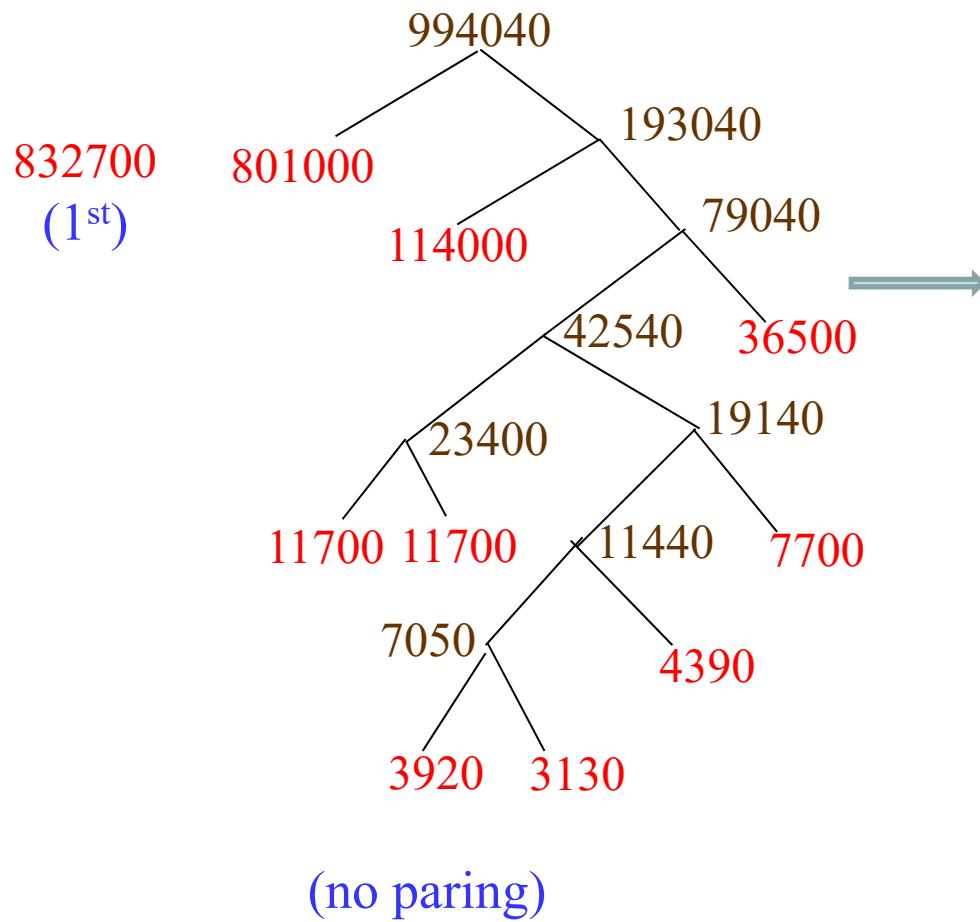


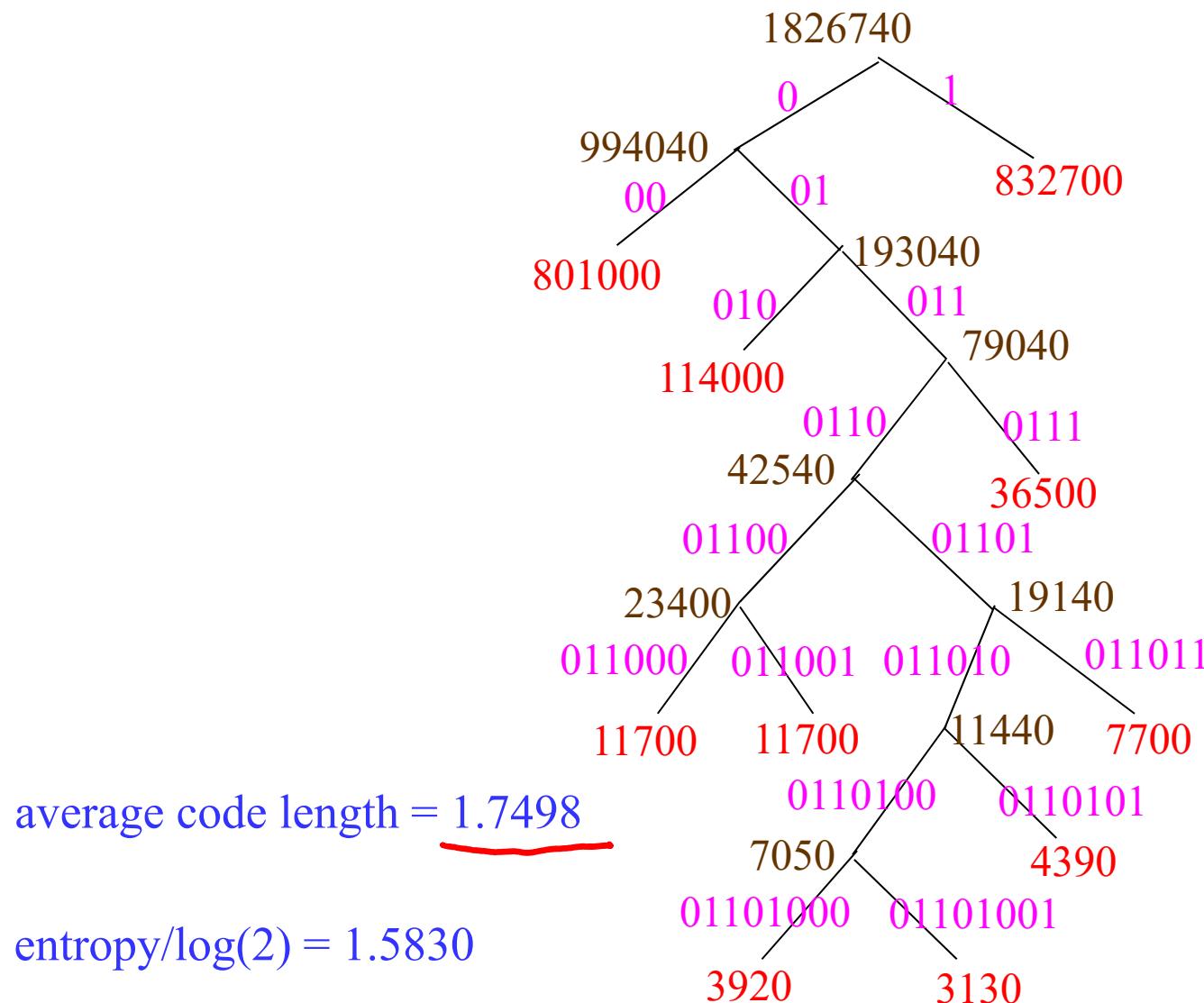
## Huffman coding



322

### Huffman coding





思考：郵遞區號是多少進位的編碼？

電話號碼的區域碼是多少進位的編碼？

中文輸入法是多少進位的編碼？

如何用 Huffman coding 來處理類似問題？

## ◎ 8-D Entropy and Coding Length

- Entropy 熵；亂度 (Information Theory)

註：此處 log 即 ln  
和  $\log_{10}$  不同

$$\text{entropy} = \sum_{j=1}^J P(S_j) \ln \frac{1}{P(S_j)}$$

P: probability

$$P(S_0) = 1, \text{ entropy} = 0$$

$$\log 2$$

$$P(S_0) = P(S_1) = 0.5, \text{ entropy} = 0.6931$$

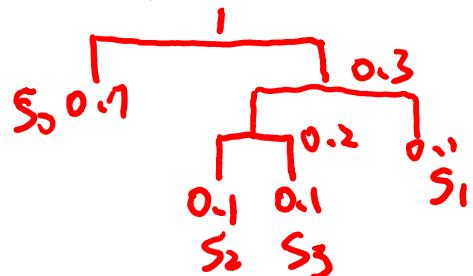
$$4 \times \left( \frac{1}{4} \log 4 \right) = \log 4 = 2 \log 2$$

$$P(S_0) = P(S_1) = P(S_2) = P(S_3) = 1/4, \text{ entropy} = 1.3863$$

$$\text{mean}(L) \geq \frac{2 \log 2}{\log 2} = 2$$

$$P(S_0) = 0.7, P(S_1) = P(S_2) = P(S_3) = 0.1, \text{ entropy} = 0.9404$$

同樣是有 4 種組合，機率分佈越集中，亂度越少



$$0.7 \times 1 + 0.1 \times 2 + 0.2 \times 3 \approx 1.5$$

$$\frac{\text{entropy}}{\log 2} = \frac{0.9404}{0.6931} \approx 1.4$$

- Huffman Coding 的平均長度

$$mean(L) = \sum_{j=1}^J P(S_j)L(S_j)$$

$P(S_j)$ :  $S_j$  發生的機率， $L(S_j)$ :  $S_j$  的編碼長度



- Shannon 編碼定理：

$$\frac{entropy}{\ln k} \leq mean(L) \leq \frac{entropy}{\ln k} + 1$$

若使用  $k$  進位的編碼

- Huffman Coding 的 total coding length  $b = mean(L)N$        $N$ : data length

$$ceil\left(N \frac{entropy}{\ln k}\right) \leq b \leq floor\left(N \frac{entropy}{\ln k} + N\right)$$

都和 entropy 有密切關係

因為編碼長度為整數，所以取 ceil / floor

ceil: 無條件進位， floor: 無條件捨去

Entropy: 估計 coding length 的重要工具

$$\frac{\text{entropy}}{\ln k} \cong \text{average bit length} \quad (\text{lower bound})$$

$$N \frac{\text{entropy}}{\ln k} \cong \text{total bit length}$$