

VIII. Data Compression (B)

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

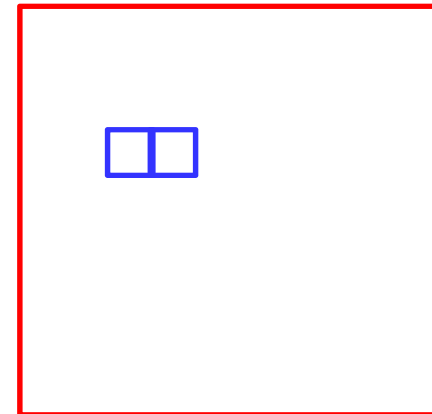
這兩者可視為 JPEG Huffman coding 的前置工作

Differential Coding (差分編碼)

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

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

Instead of $DC[i, j]$

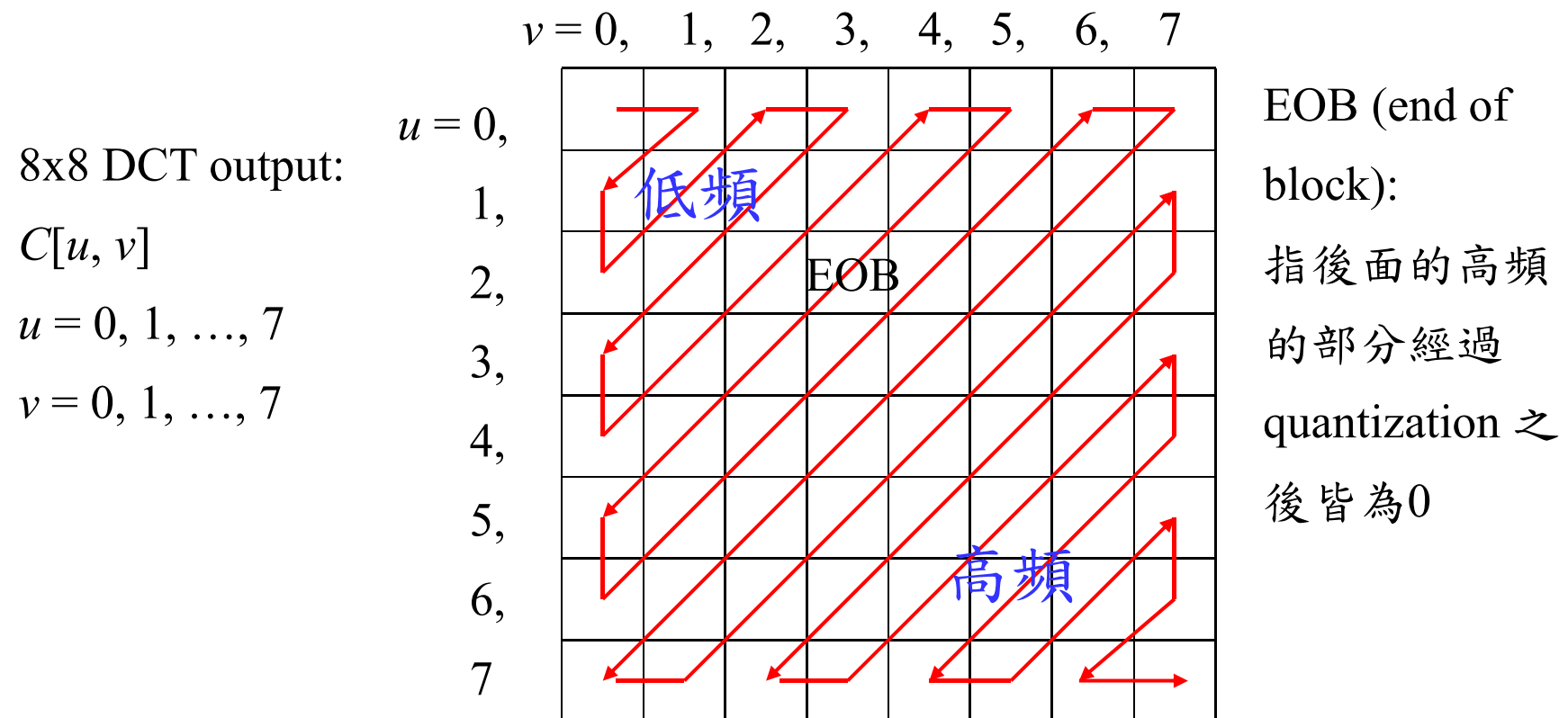


(也是運用 space domain 上的一致性)

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)



(也是運用 frequency domain 上的一致性)

© 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)

- (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_a 是第 L 層的 node， S_b 是第 $L+1$ 層的 node
則 $P(S_a) \geq P(S_b)$ 必需滿足

不滿足以上的條件則往上推一層

原始的編碼方式：

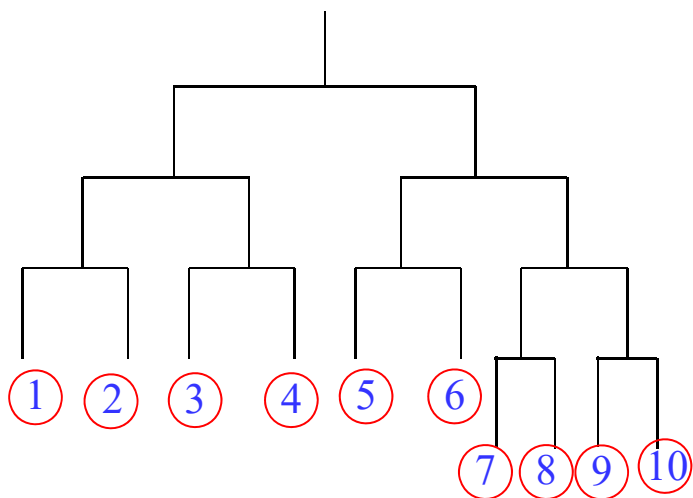
若 data 有 M 個可能的值，使用 k 進位的編碼，
則每一個可能的值使用 $\text{floor}(\log_k M)$ 或 $\text{ceil}(\log_k M)$ 個 bits 來編碼

floor: 無條件捨去

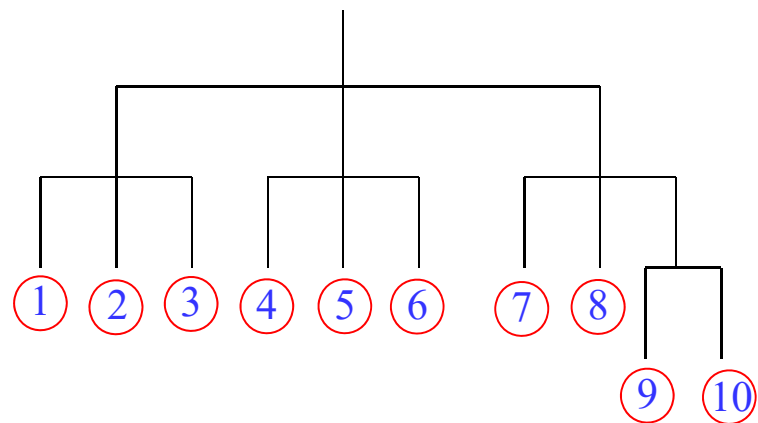
ceil: 無條件進位

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



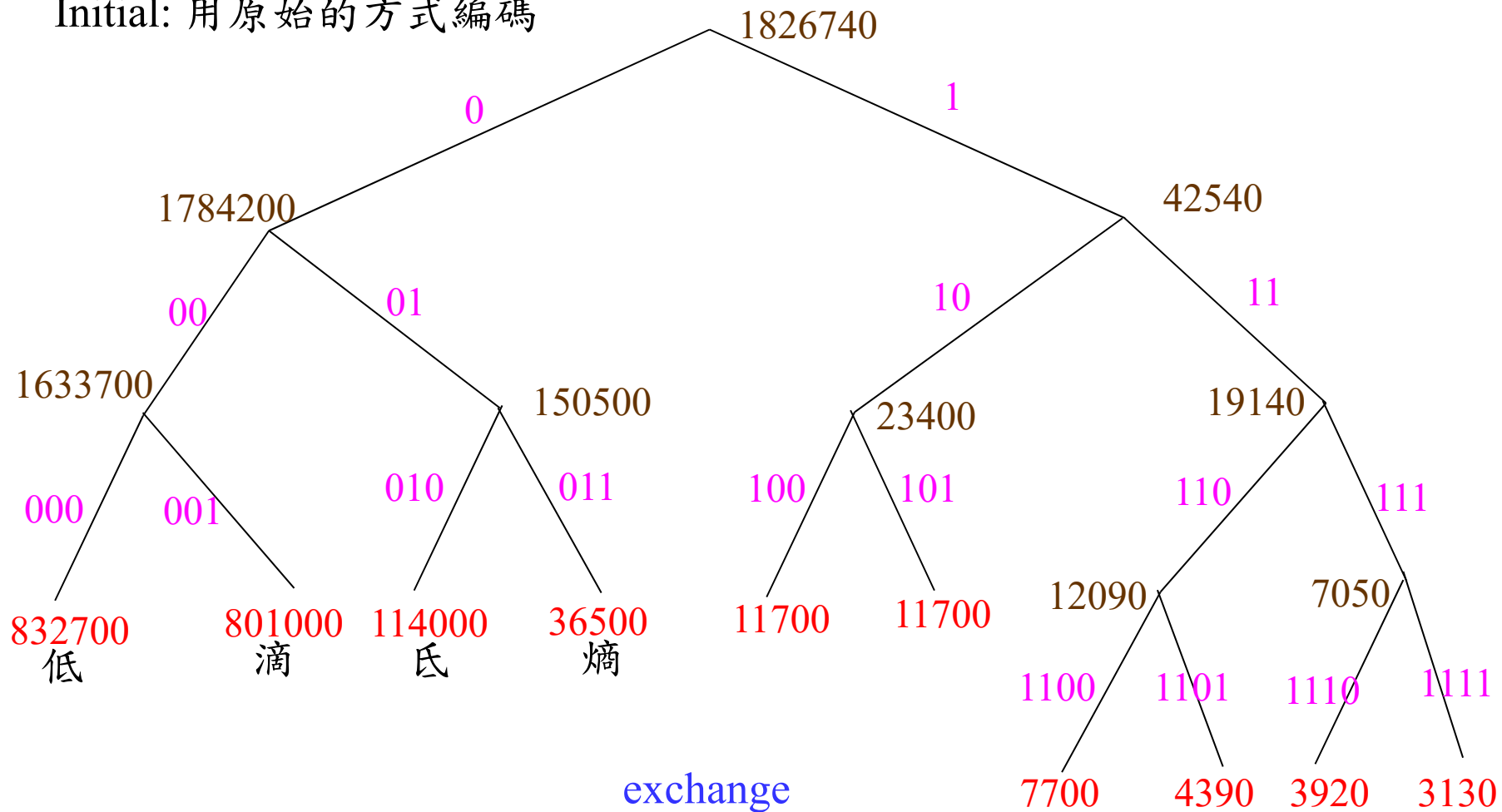
Example:

低	滴	氏	羝	鞞
832700	801000	114000	7700	4390
磳	祗	蒟	埒	熵
3920	11700	11700	3130	36500

他們 2 進位的 Huffman Code 該如何編

Huffman coding

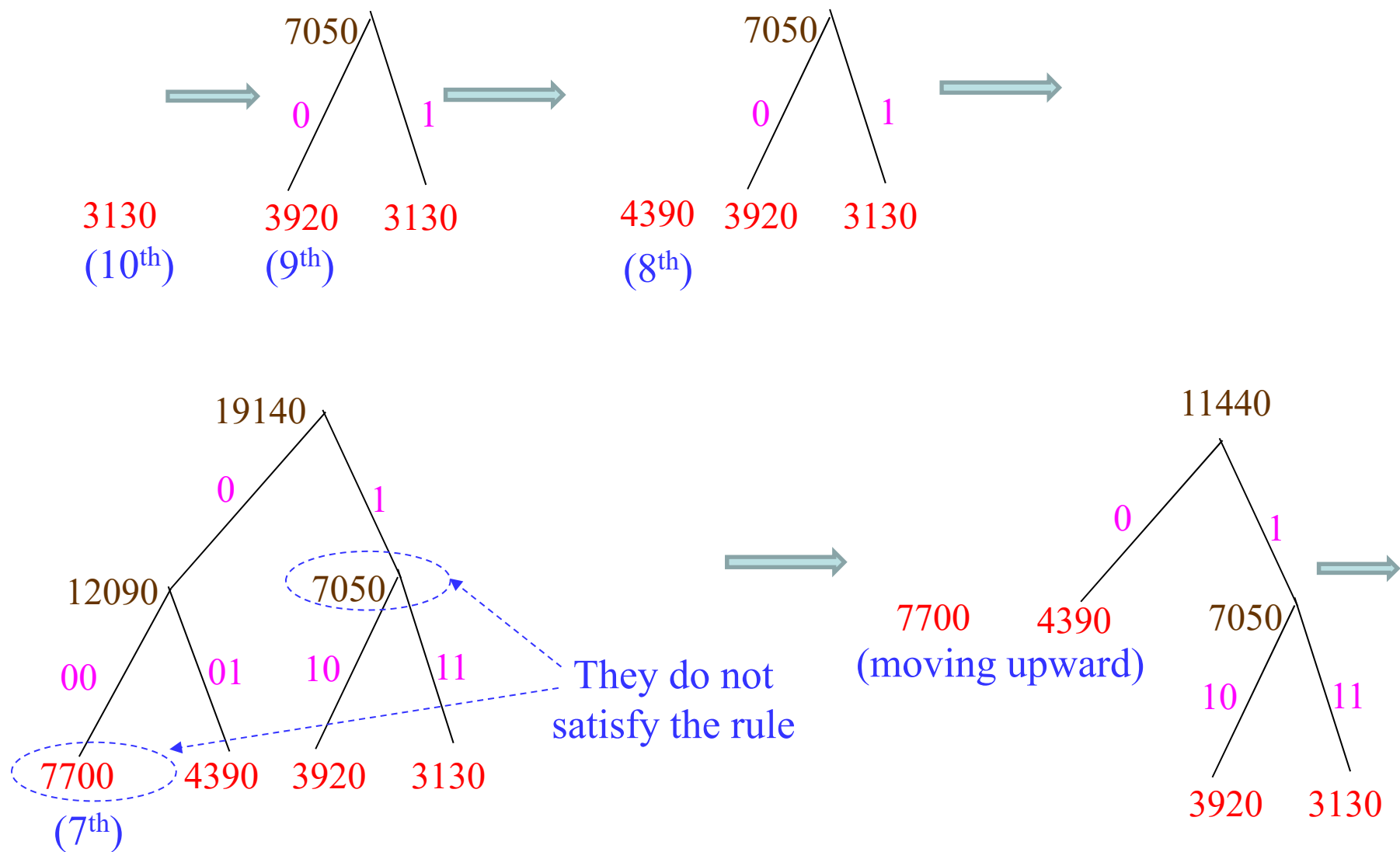
Initial: 用原始的方式編碼



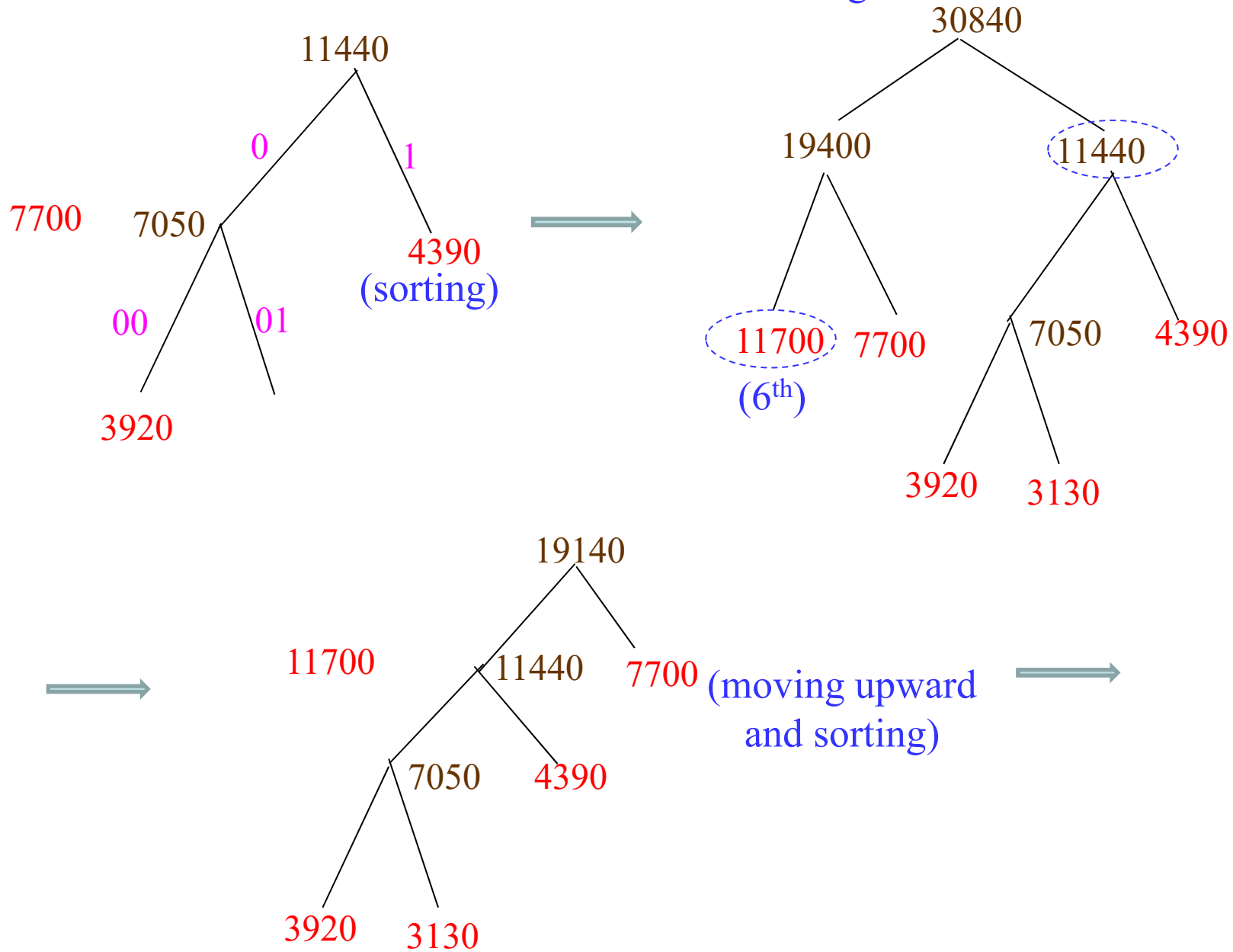
average code length = 3.0105

Huffman coding

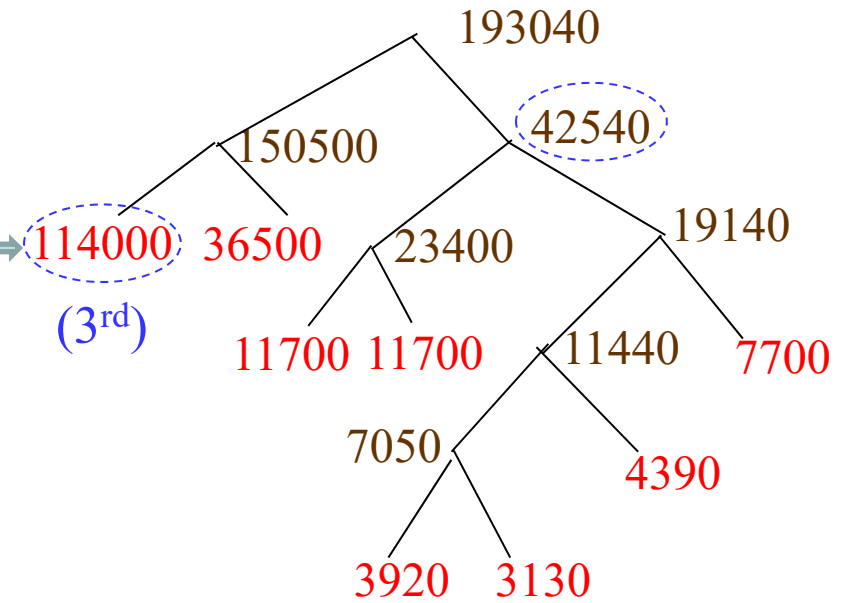
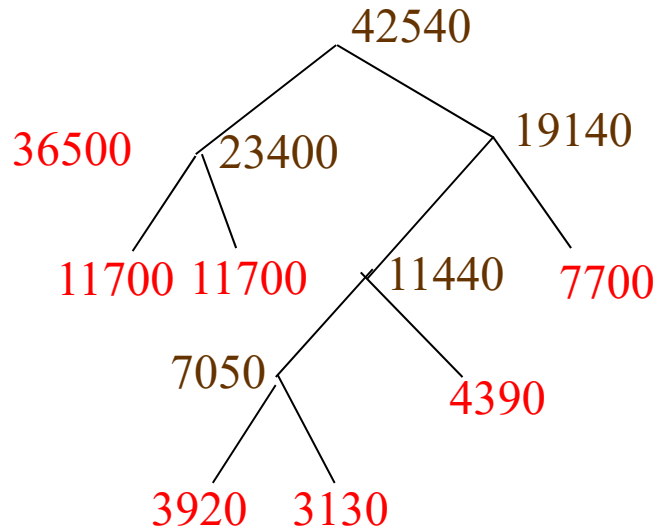
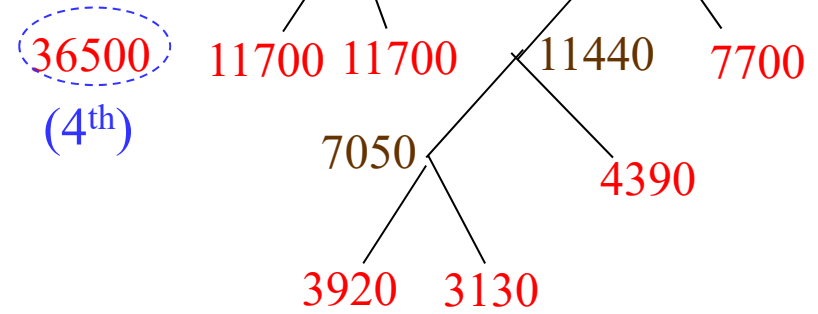
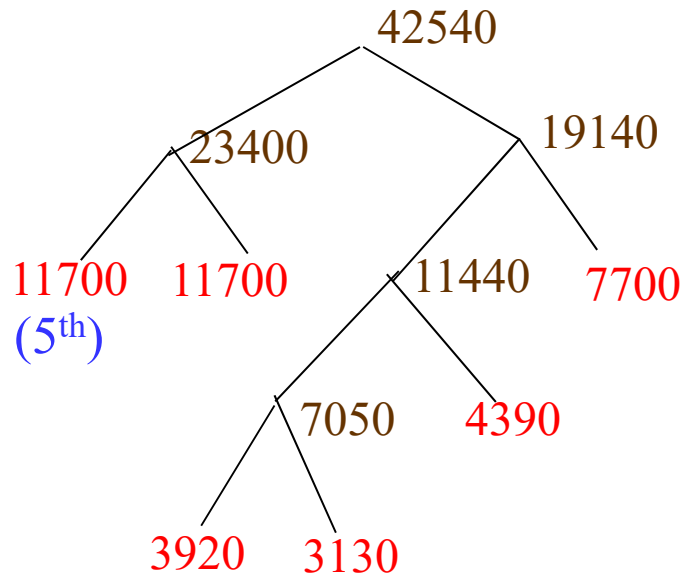
由機率低的開始編碼，一步一步加進機率高的



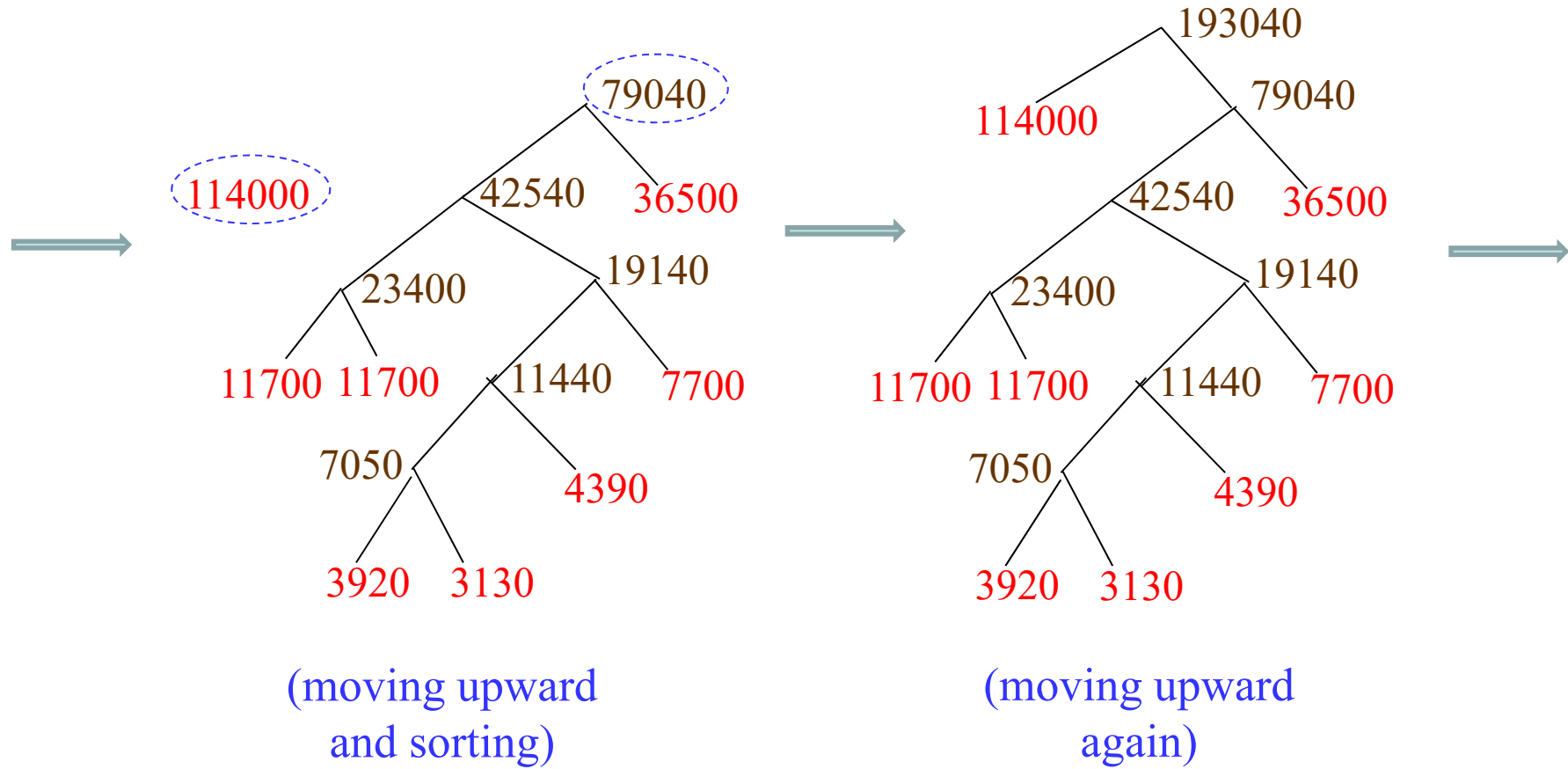
Huffman coding



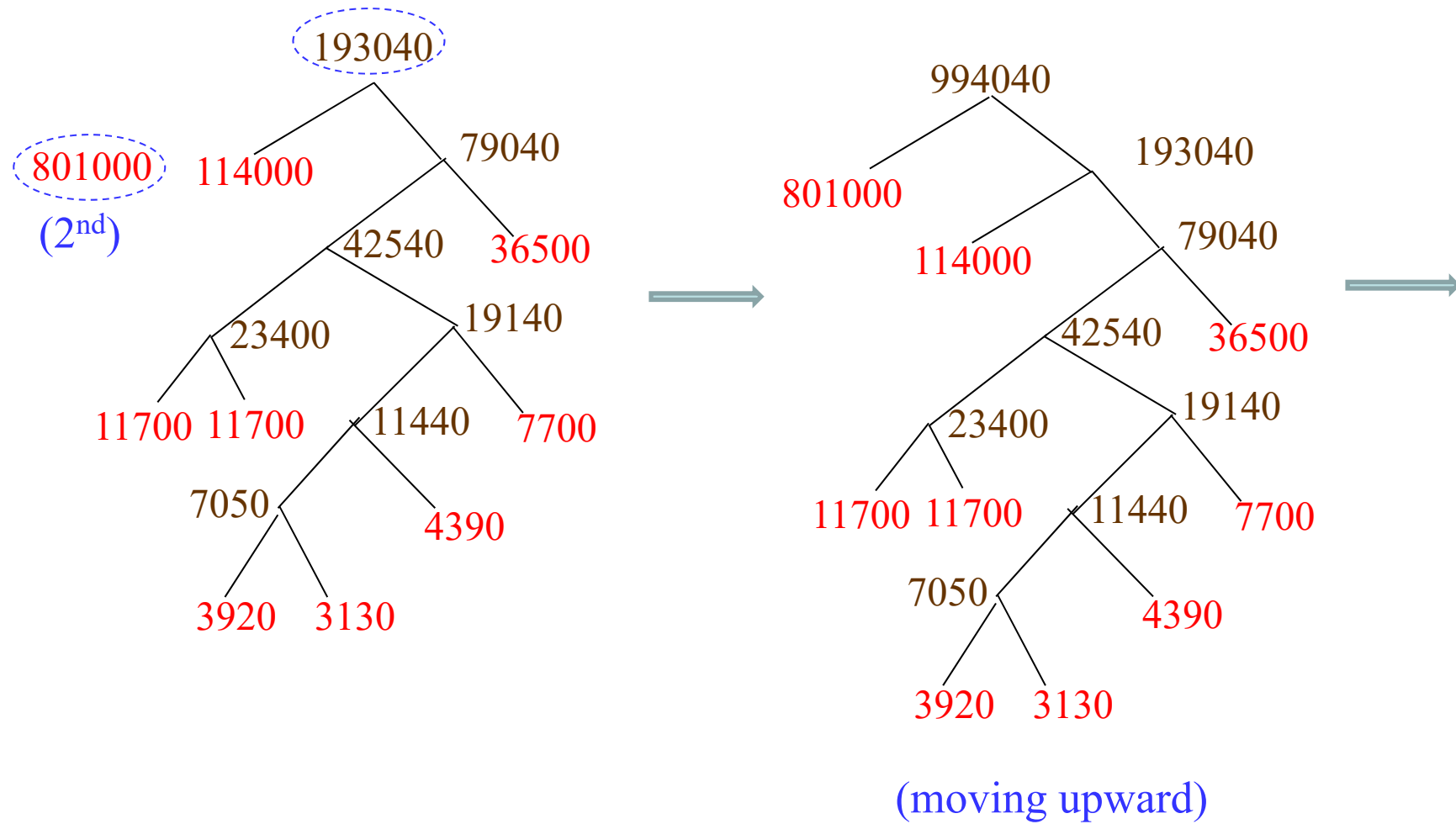
Huffman coding



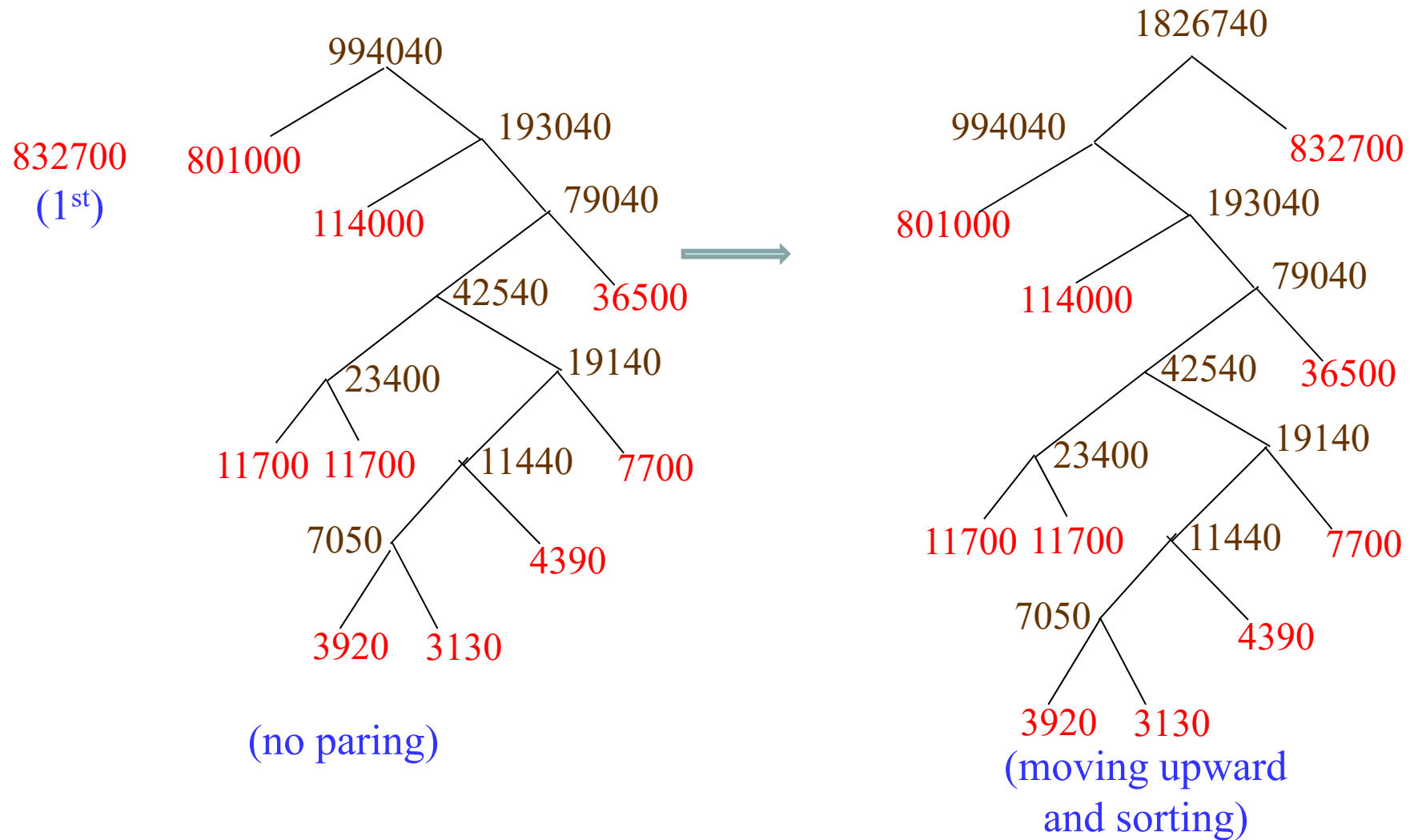
Huffman coding



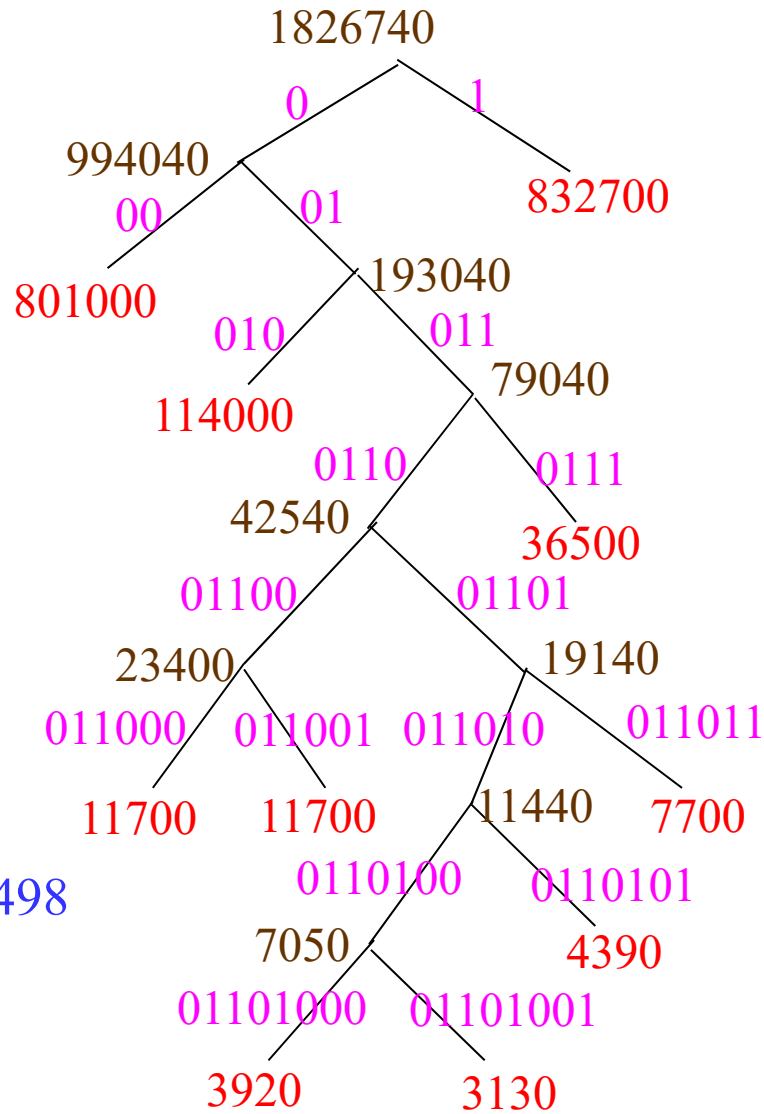
Huffman coding



Huffman coding



Huffman coding by the greedy algorithm



average code length = 1.7498

entropy/log(2) = 1.5830

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

$$\text{entropy} = \sum_{j=1}^J P(S_j) \log \frac{1}{P(S_j)} \quad P: \text{probability}$$

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

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

$$P(S_0) = P(S_1) = P(S_2) = P(S_3) = P(S_4) = 1/5, \text{ entropy} = 1.6094$$

$$P(S_0) = P(S_1) = P(S_2) = P(S_3) = 0.1, P(S_4) = 0.6, \text{ entropy} = 1.2275$$

同樣是有 5 種組合，機率分佈越集中，亂度越少

- Huffman Coding 的平均長度

$$mean(L) = \sum_{j=1}^J P(S_j) L(S_j) \quad P(S_j): S_j \text{ 發生的機率, } L(S_j): S_j \text{ 的編碼長度}$$

- ★ ★ ★ • Shannon 編碼定理:

$$\frac{entropy}{\log k} \leq mean(L) \leq \frac{entropy}{\log k} + 1 \quad \text{若使用 } k \text{ 進位的編碼}$$

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

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

都和 entropy 有密切關係

ceil: 無條件進位, floor: 無條件捨去

Entropy: 估計 coding length 的重要工具

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

◎ 8-E Arithmetic Coding

- Arithmetic Coding (算術編碼)

Huffman coding 是將每一筆資料分開編碼

Arithmetic coding 則是將多筆資料一起編碼，因此壓縮效率比 Huffman coding 更高，近年來的資料壓縮技術大多使用 arithmetic coding

K. Sayood, *Introduction to Data Compression*, Chapter 4: Arithmetic coding, 3rd ed., Amsterdam, Elsevier, 2006

編碼

若 data X 有 M 個可能的值 ($X[i] = 1, 2, \dots, \text{or } M$)，使用 k 進位的編碼，且

P_n : the probability of $x = n$ (from prediction)

$$S_0 = 0, \quad S_n = \sum_{j=1}^n P_j$$

現在要對 data X 做編碼，假設 $\text{length}(X) = N$

Algorithm for arithmetic encoding

initiation: $lower = S_{X[1]-1}$ $upper = S_{X[1]}$

for $i = 2 : N$

$$lower = lower + S_{X[i]-1} \times (upper - lower)$$

$$upper = lower + S_{X[i]} \times (upper - lower)$$

end

(continue)...

Suppose that

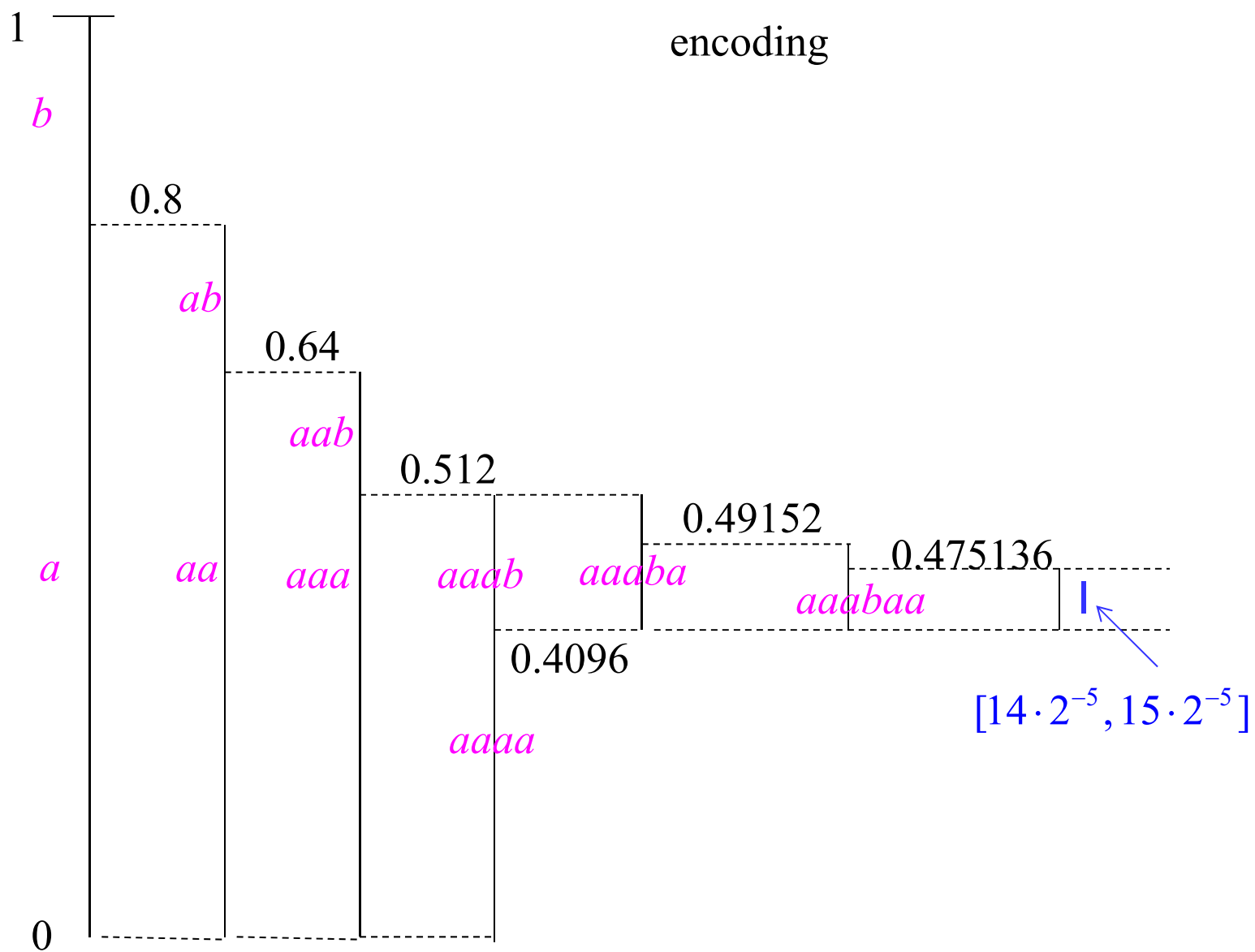
$$lower \leq C \cdot k^{-b} < (C + 1) \cdot k^{-b} \leq upper$$

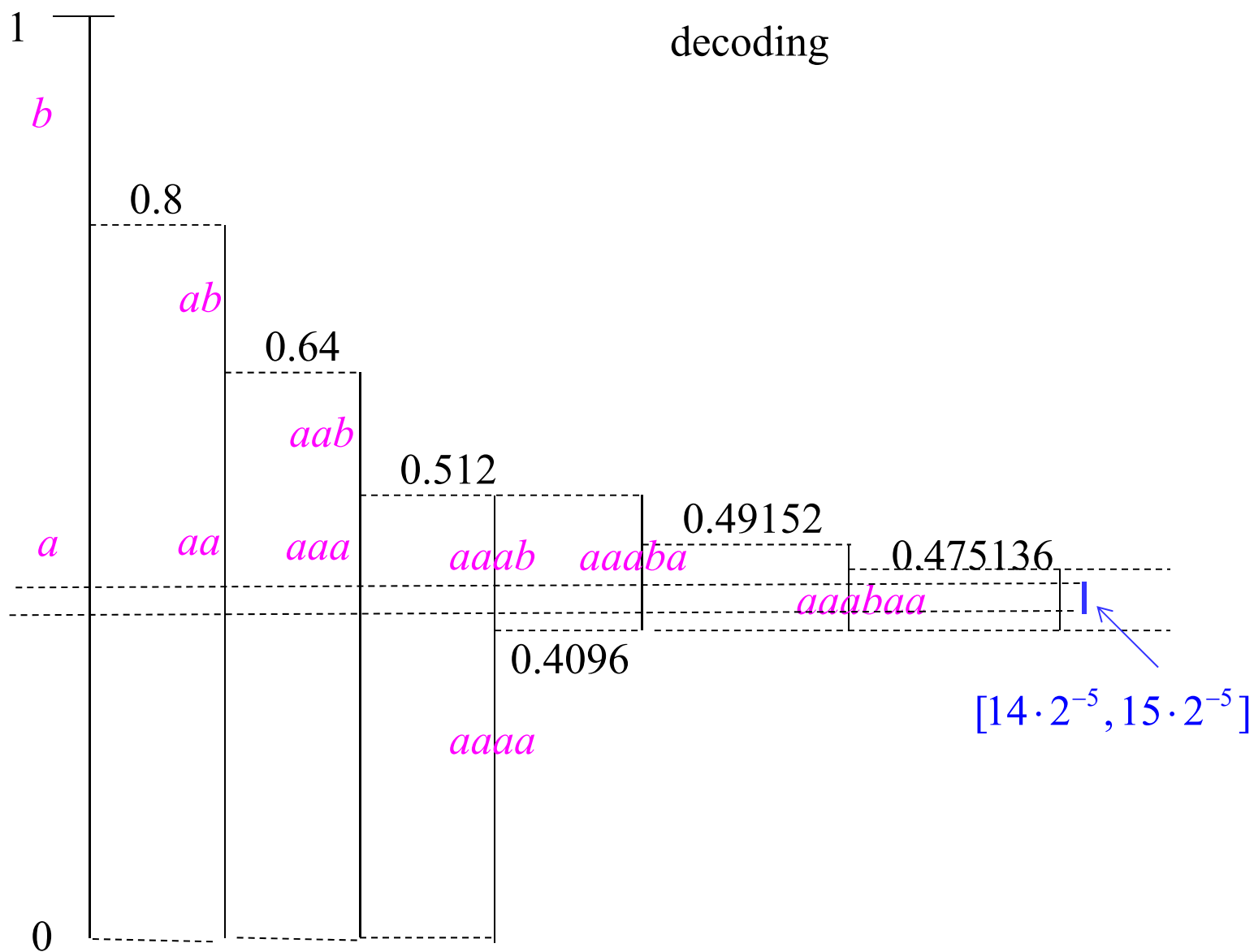
where C and b are integers (b is as small as possible), then the data X can be encoded by

$$C_{(k,b)}$$

where $C_{(k,b)}$ means that using k -ary (k 進位) and b bits to express C .

(註：Arithmetic coding 還有其他不同的方式，以上是使用其中一個較簡單的 range encoding 的方式)





Example:

假設要對 X 來做二進位 ($k=2$) 的編碼

且經由事先的估計， $X[i] = a$ 的機率為 0.8, $X[i] = b$ 的機率為 0.2

$$\longrightarrow P_1 = 0.8, \quad P_2 = 0.2, \quad S_0 = 0, \quad S_1 = 0.8, \quad S_2 = 1$$

若實際上輸入的資料為 $X = a a a b a a$

Initiation ($X[1] = a$): $lower = 0, \quad upper = 0.8$

When $i = 2$ ($X[2] = a$): $lower = 0, \quad upper = 0.64$

When $i = 3$ ($X[3] = a$): $lower = 0, \quad upper = 0.512$

When $i = 4$ ($X[4] = b$): $lower = 0.4096, \quad upper = 0.512$

When $i = 5$ ($X[5] = a$): $lower = 0.4096, \quad upper = 0.49152$

When $i = 6$ ($X[6] = a$): $lower = 0.4096, \quad upper = 0.475136$

由於 $lower = 0.4096$, $upper = 0.475136$

$$lower \leq 14 \cdot 2^{-5} < 15 \cdot 2^{-5} \leq upper$$

0.4375 0.46875

所以編碼的結果為

$$14_{(2,5)} = 01110$$

2 進位 5 個 bits

解碼

假設編碼的結果為 Y , $\text{length}(Y) = b$

其他的假設，和編碼 (see page 292) 相同 使用 k 進位的編碼

Algorithm for arithmetic decoding

```

initiation:       $lower = 0$             $upper = 1$             $j = 1$ 
                   $lower\ 1 = 0$         $upper\ 1 = 1$ 

for  $i = 1 : N$       % loop 1
    check = 1;
    while check = 1    % loop 2
        if there exists an  $n$  such that
             $lower + (upper - lower)S_{n-1} \leq lower\ 1$    and
             $lower + (upper - lower)S_n \geq upper\ 1$    are both satisfied,
            then
                check = 0;
                (continue)....

```

```

else
     $upper\ 1 = lower\ 1 + (Y[j] + 1)k^{-j}$ 
     $lower\ 1 = lower\ 1 + Y[j]k^{-j}$ 
     $j = j + 1$ 
end
end                                     % end of loop 2
 $X(i) = n;$ 
 $lower = lower + (upper - lower)S_{n-1}$ 
 $upper = lower + (upper - lower)S_n$ 
end                                     % end of loop 1

```

Coding Length for Arithmetic Coding

假設 P_n 是預測的 $X[i] = n$ 的機率

Q_n 是實際上的 $X[i] = n$ 的機率

(也就是說，若 $\text{length}(X) = N$, X 當中會有 $Q_n N$ 個 elements 等於 n)

則

$$\text{upper} - \text{lower} = \prod_{m=1}^M P_m^{Q_m N} \quad \prod : \text{連乘符號}$$

另一方面，由於 (from page 293)

$$k^{-b} \leq \text{upper} - \text{lower} < (2k)k^{-b}$$

$$-\log_k (\text{upper} - \text{lower}) \leq b < -\log_k (\text{upper} - \text{lower}) + 1 + \log_k 2$$

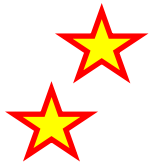
$$\text{ceil} \left(-N \sum_{m=1}^M Q_m \log_k P_m \right) \leq b \leq \text{floor} \left(-N \sum_{m=1}^M Q_m \log_k P_m + \log_k 2 \right) + 1$$

$$\text{ceil}\left(-N \sum_{m=1}^M Q_m \log_k P_m\right) \leq b \leq \text{floor}\left(-N \sum_{m=1}^M Q_m \log_k P_m + \log_k 2\right) + 1$$

在機率的預測完全準確的情形下， $Q_m = P_m$

Total coding length b 的範圍是

$$\text{ceil}\left(-N \sum_{m=1}^M P_m \log_k P_m\right) \leq b \leq \text{floor}\left(-N \sum_{m=1}^M P_m \log_k P_m + \log_k 2\right) + 1$$



$$\text{ceil}\left(N \cdot \frac{\text{entropy}}{\log k}\right) \leq b \leq \text{floor}\left(N \cdot \frac{\text{entropy}}{\log k} + \log_k 2 + 1\right)$$

Arithmetic coding 的 total coding length 的上限比 Huffman coding 更低

◎ 8-F MPEG

MPEG：動態影像編碼的國際標準 全名：Moving Picture Experts Group

MPEG standard： <http://www.iso.org/iso/prods-services/popstds/mpeg.html>

MPEG 官方網站： <http://mpeg.chiariglione.org/>

人類的視覺暫留： 1/24 second

一個動態影像，每秒有 30個或 60個畫格 (frames)



例子：

Pepsi 的廣告

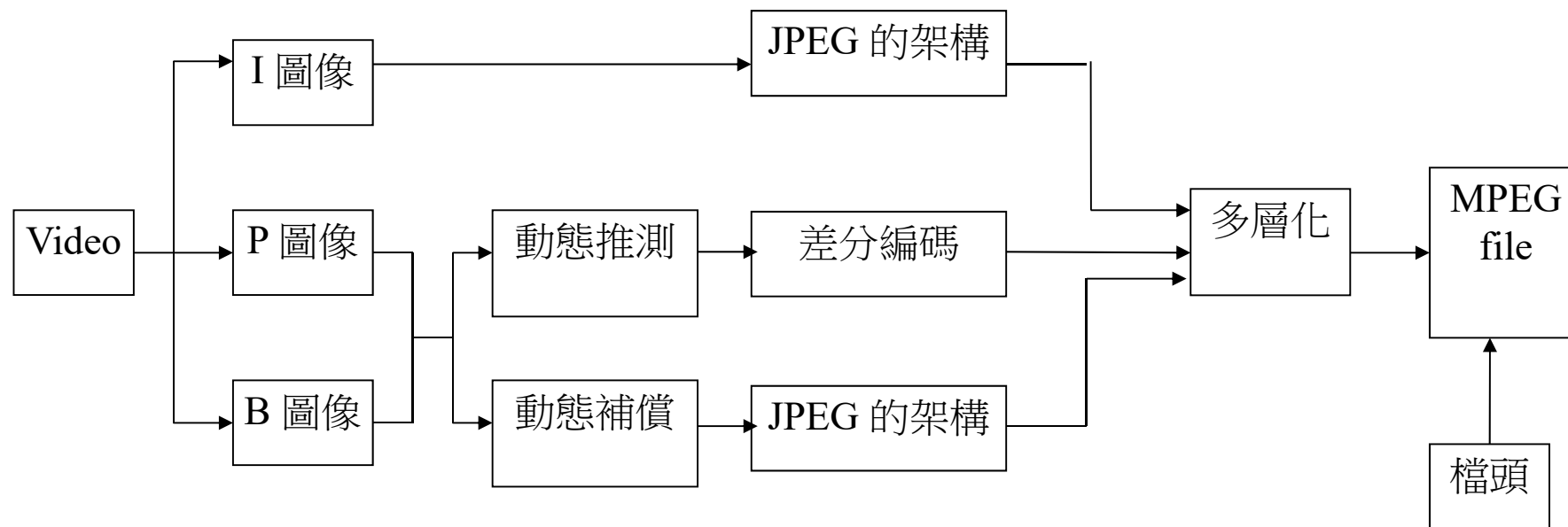
Size: 160×120 Time: 29 sec 一秒 30 個 frames

若不作壓縮： $160 \times 120 \times 29 \times 30 \times 3 = 50112000 = 47.79$ M bytes。

經過 MPEG 壓縮： $1140740 = 1.09$ M bytes。

只有原來的 2.276%。

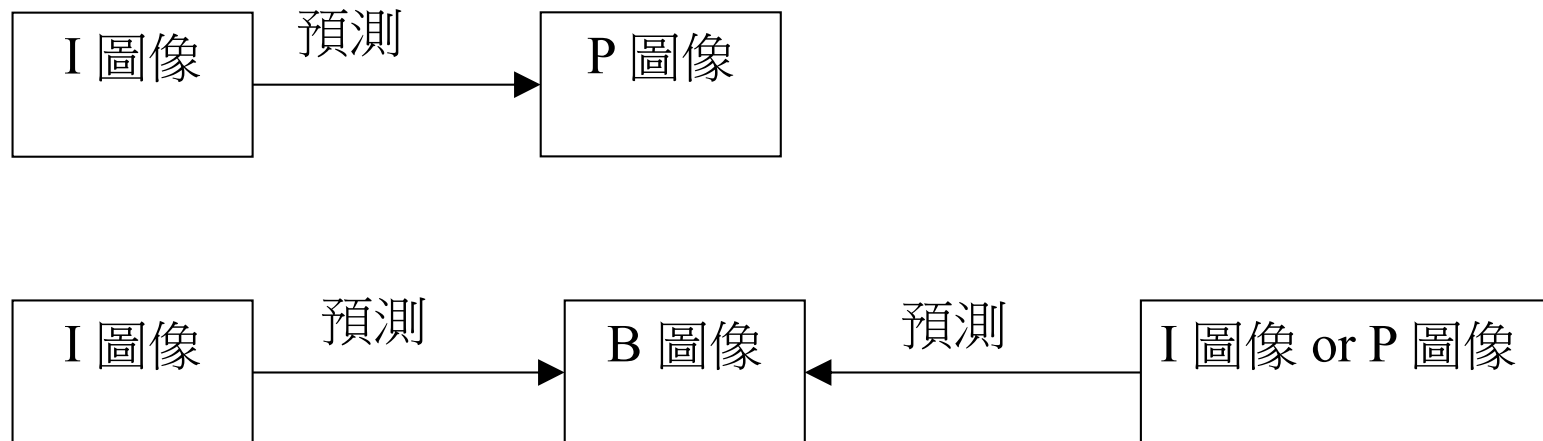
- Flowchart of MPEG Compression



I 圖像 (Intra-coded picture): 作為參考的畫格

P 圖像 (Predictive-coded picture): 由之前的畫格來做預測

B 圖像 (Bi-directionally predictive-coded picture): 由之前及之後的畫格來做預測

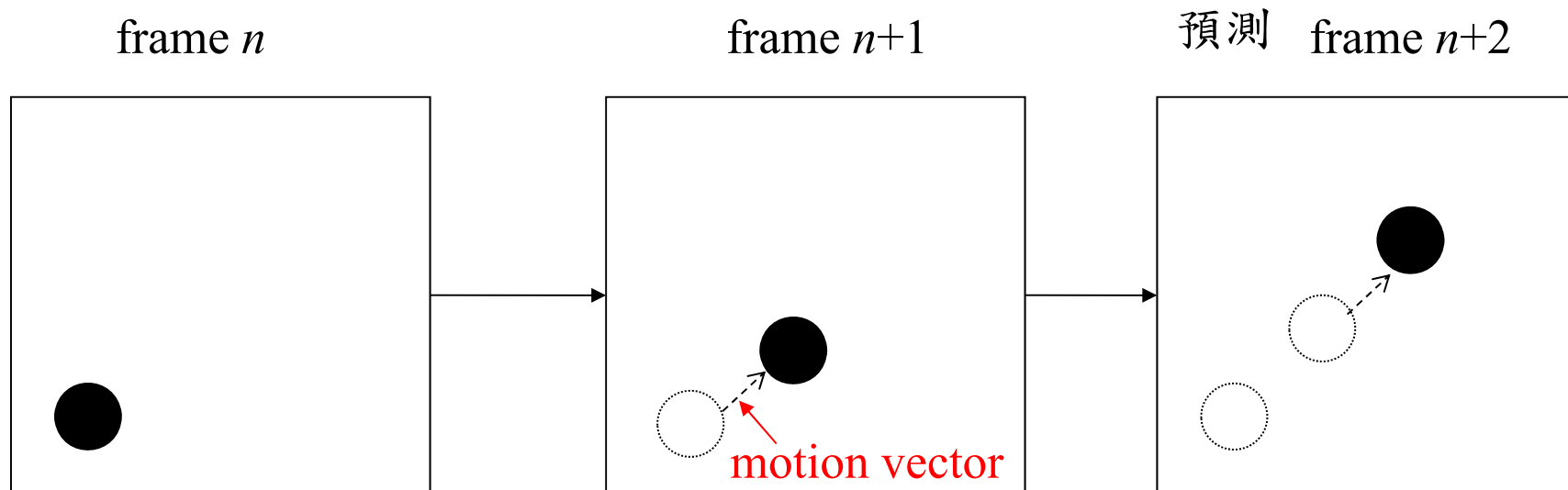


- 動態影像之編碼

原理：不同時間，同一個 pixel 之間的相關度通常極高
只需對有移動的 objects 記錄“motion vector”

- 動態補償 (Motion Compensation)

時間相近的影像，彼此間的相關度極高



$F[m, n, t]$: 時間為 t 的影像

如何由 $F[m, n, t]$, $F[m, n, t+\Delta]$ 來預測 $F[m, n, t+2\Delta]$?

(1) 移動向量 $V_x(m, n), V_y(m, n)$

(2) 預測 $F[m, n, t+2\Delta]$:

$$F_p[m, n, t+2\Delta] = F[m - V_x(m, n), n - V_y(m, n), t+\Delta]$$

(3) 計算「預測誤差」

$$E[m, n, t+2\Delta] = F[m, n, t+2\Delta] - F_p[m, n, t+2\Delta]$$

對預測誤差 $E[m, n, t+2\Delta]$ 做編碼

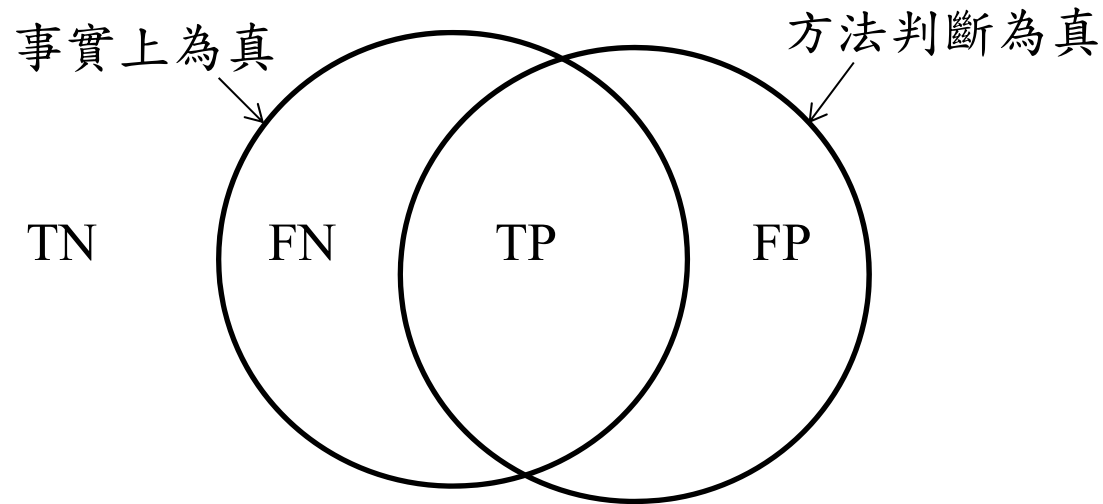
◎ 8-G Data Compression 未來發展的方向

Two important issues:

Q1: How to further improve the compression rate

Q2: How to develop a compression algorithm whose compression rate is acceptable and the buffer size / hardware cost is limited

附錄八：量測方法的精確度常用的指標



TP (true positive): 事實上為真，而且被我們的方法判斷為真的情形

FN (false negative): 事實上為真，卻未我們的方法被判斷為真的情形

FP (false positive): 事實上不為真，卻被我們的方法誤判為真的情形

TN (true negative): 事實上不為真，而且被我們的方法判斷成不為真的情形

$$precision = \frac{TP}{TP + FP} = +P \text{ (positive prediction rate)}$$

$$recall = \frac{TP}{TP + FN}$$

$$specificity = \frac{TN}{TN + FP}$$

$$sensitivity = \frac{TP}{TP + FN} = recall$$

以抓犯人為例，TP 是有罪而且被抓到的情形，FP 是無罪但被誤抓的情形，FN 是有罪但沒被抓到的情形，TN 是無罪且未被誤逮的情形

寧可錯抓一百，也不可放過一個

—————> recall 高，但 precision 低

寧可錯放一百，也不可冤枉一個

—————> precision 高，但 recall 低

$$\text{Accuracy} \quad \frac{TP + TN}{TP + FP + TN + FN}$$

$$\text{Detection error rate} \quad \frac{FP + FN}{TP + FN}$$

$$\text{F-score} \quad 2 \frac{\textit{precision} \times \textit{recall}}{\textit{precision} + \textit{recall}}$$

$$\text{General form of the F-score} \quad \frac{(1 + \beta^2) \textit{precision} \times \textit{recall}}{\beta^2 \textit{precision} + \textit{recall}}$$