Algorithms

Chapter 6
Algorithms Involving Sequences & Sets
Part 3
(pp. 143~158)

Order Statistics

Order Statistics

- Given a sequence S= $x_1, x_2, \ldots, x_n$
  - x_i has the rank of k if x_i is the k-th element in S
- Problem of Order Statistics
  - finding the maximum & minimum elements
    - sorting & finding: O(nlogn)
    - brute force: O(2n-3)
    - Improve: O(3n/2)
  - finding the k-th smallest element
    - sorting & finding: O(nlogn)
    - divide & conquer: O(n)

Finding Maximum Elements

- Finding maximum
  - **Base: one element**
  - **Hypothesis: finding maximum of n-1 element**
  - **Induction on n element**
    - **compare n-th element with max(n-1)**
    - **#(comparisons)=n-1**

Finding Both Maximum & Minimum Elements

- Approach 1
  - solve independently
  - O(n-1)+O(n-2) = O(2n-3)
- Approach 2
  - First attempt
    - hypothesis: finding both maximum & minimum of n-1 elements
    - induction: compare the n-th element to maximum(n-1) & minimum (n-1)
    - O(2n-3)

請構思O(3n/2)演算法同時找出最大與最小 (hint: induction from n-2 to n)

Finding Both Maximum & Minimum Elements (cont.)

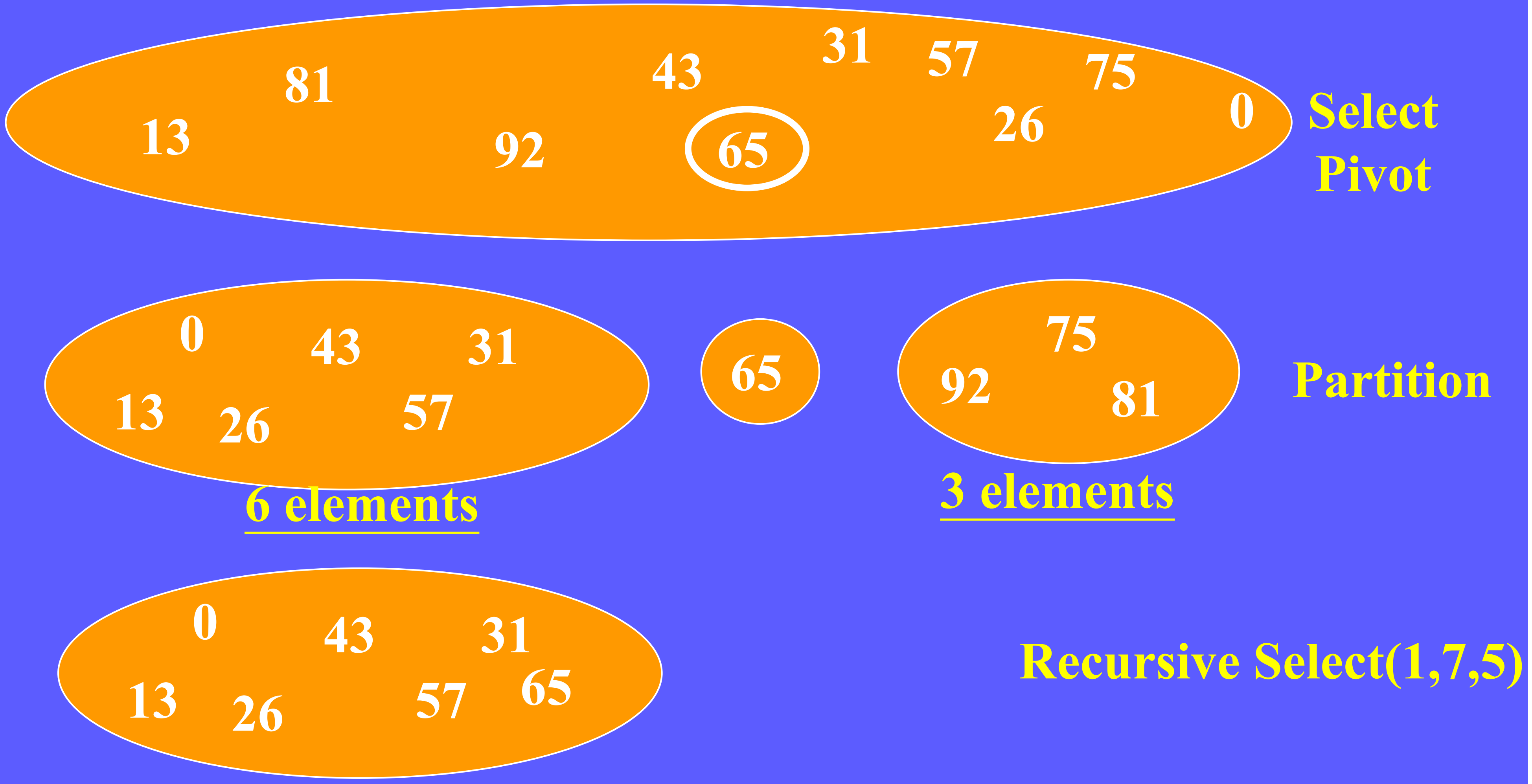
- Approach 2
  - **second attempt**
    - **base: n=1, n=2**
    - **hypothesis: finding both maximum & minimum of n-2 elements**
    - **induction**
      - **compare the n-th element with (n-1) element**
      - **compare the larger to maximum(n-2)**
      - **compare the smaller to minimum(n-2)**
    - **O(3n/2)**

請構思2個演算法來找出k-th element 並分析其時間複雜度

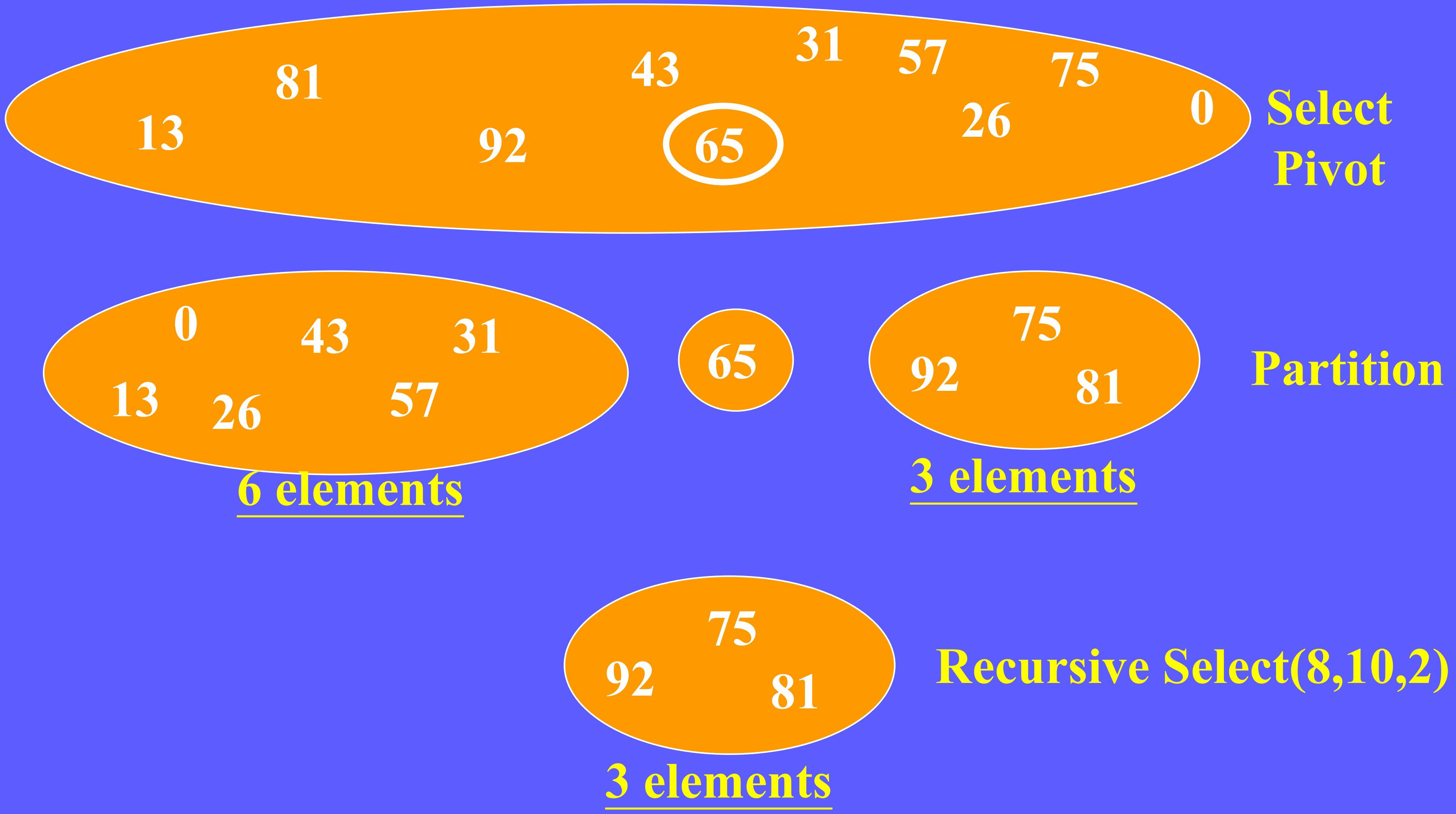
Finding the k-th Smallest Element

- Approach 1 (if k is close to 1 or n)
  - finding minimum or maximum k times
  - O(kn)
- Approach 2
  - sorting & finding
  - O(nlogn)
- Approach 3
  - divide & conquer
  - partition & finding subparts
  - similar to Quicksort
    except that only one subproblem has to be solved
  - O(n)

Finding the 5-th Smallest Element

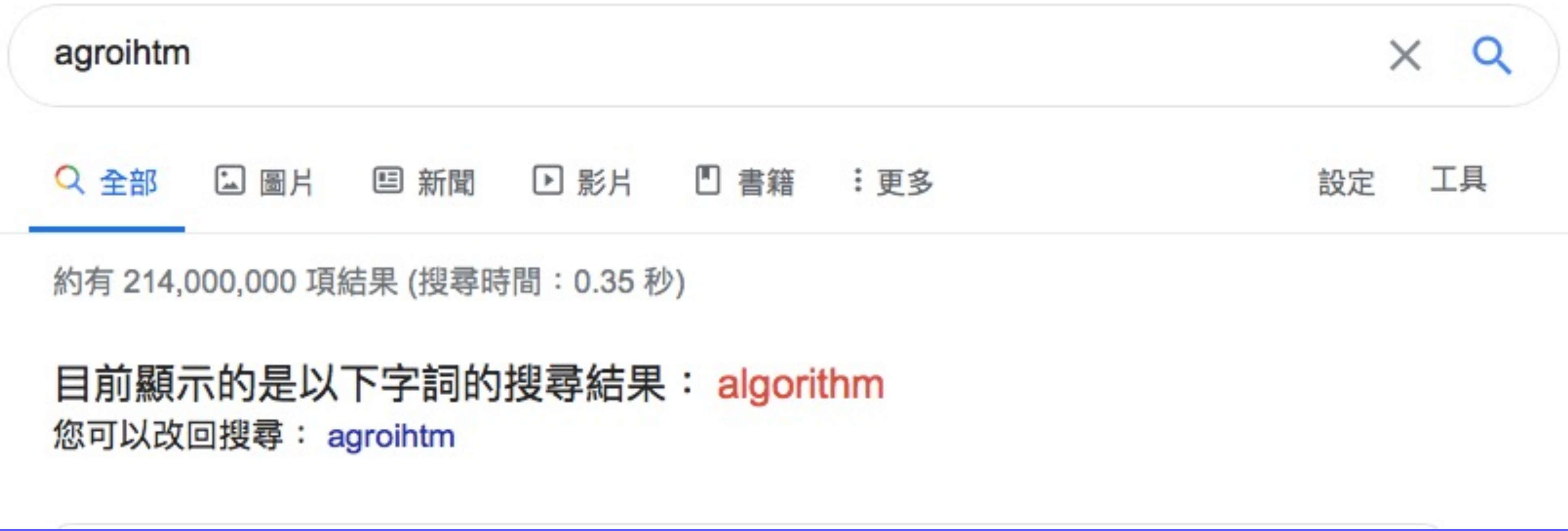


Finding the 9-th Smallest Element



```
Algorithm Selection (X, n, k)
Input: X, k
Output: S
begin
    if (k<1) or (k > n) then print “error”
    else S: Select(1,n,k)
end
procedure Select (left, Right, k)
begin
    if Left=Right then
       Select:=Left
    else
        Partition(X, Left, Right)
        Let Middle be the output of Partition;
        if Middle-left+1 >= k then
            Select(left, Middle,k)
        else
            Select(Middle+1, Right, k-(Middle-Left+1))
end
```

Sequence Comparison



agroihtm

全部 圖片 新聞 影片 書籍 更多 設定 工具

約有 214,000,000 項結果 (搜尋時間：0.35 秒)

目前顯示的是以下字詞的搜尋結果： algorithm
您可以改回搜尋： agroihtm

Sequence Comparison

■ Given two strings $A=a_1a_2\ldots a_n$, $B=b_1b_2\ldots b_m$

how to measure the dissimilarity between A and B

$\Rightarrow$ <u>Given</u> two strings $A=a_1a_2\ldots a_n$, $B=b_1b_2\ldots b_m$

<u>how to</u> change A to B character by character

using three types of edit operation

(1) insert a character

(2) delete a character

(3) replace one character with a different character

<u>such that</u> the number of edit steps is minimum

$\Rightarrow$ Dissimilarity = minimum number of edit steps

Example of Sequence Comparison

<e.g.> Given two strings A=abbc, B=babb,

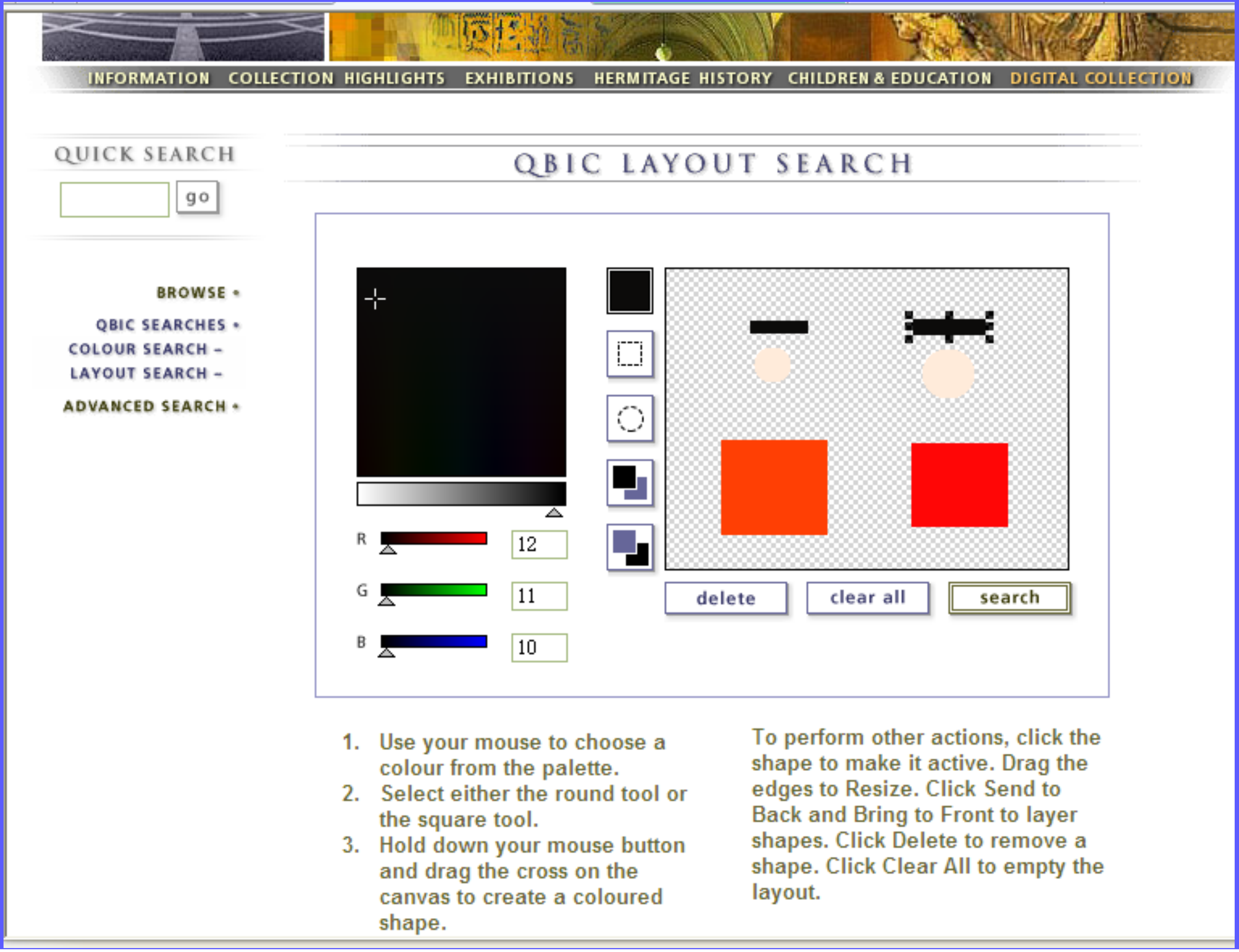
possible editing steps

- abbc $\xrightarrow{\text{delete a}}$ bbc $\xrightarrow{\text{insert a}}$ babc $\xrightarrow{\text{replace}}$ babb
- abbc $\xrightarrow{\text{insert b}}$ babbc $\xrightarrow{\text{delete c}}$ babb

Applications of String Editing Distance

- Dictionary
- Image retrieval (shape retrieval)
- Music retrieval
- Biological sequence search
- …

Image Retrieval (cont.)

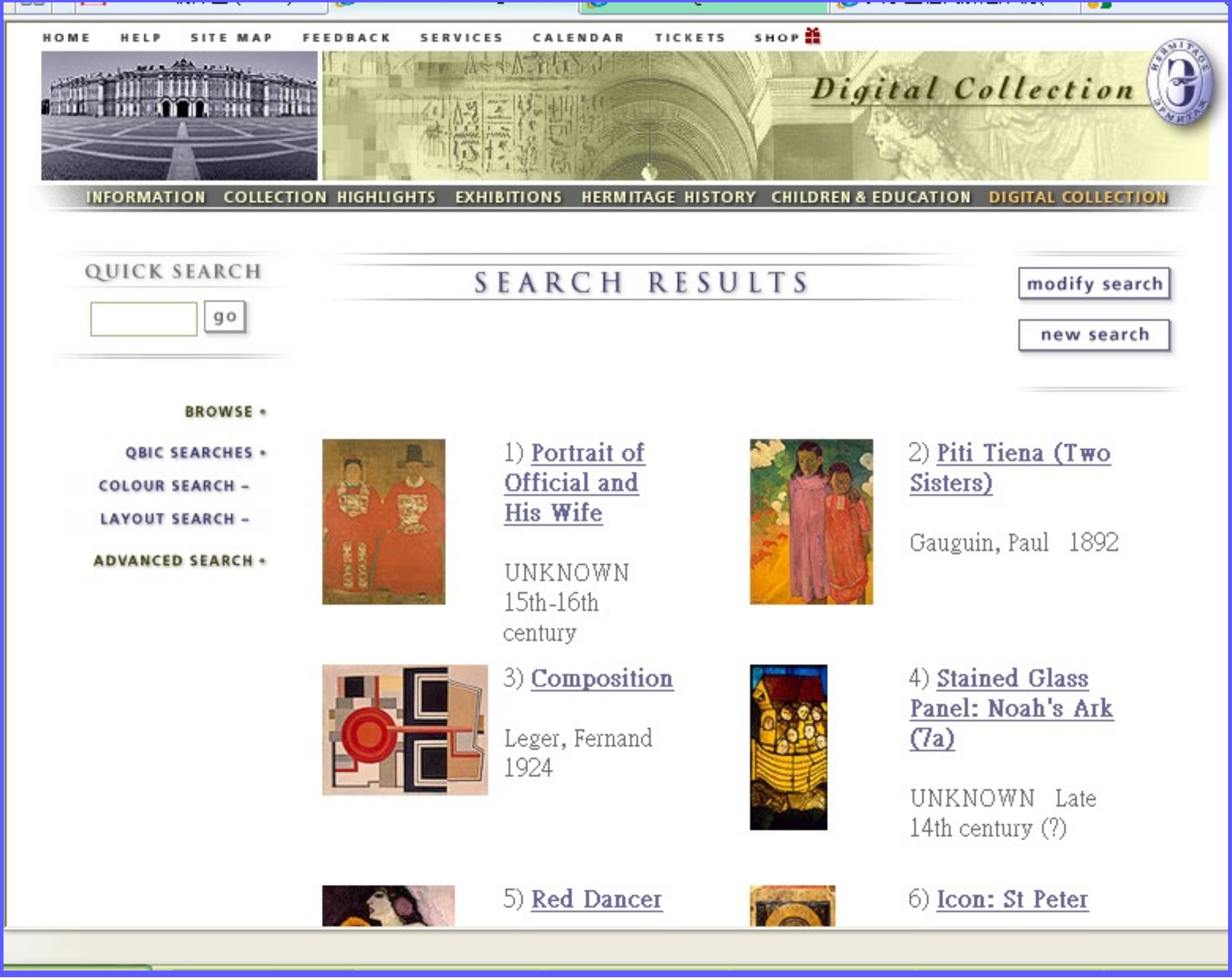


QBIC LAYOUT SEARCH

1. Use your mouse to choose a colour from the palette.
2. Select either the round tool or the square tool.
3. Hold down your mouse button and drag the cross on the canvas to create a coloured shape.

To perform other actions, click the shape to make it active. Drag the edges to Resize. Click Send to Back and Bring to Front to layer shapes. Click Delete to remove a shape. Click Clear All to empty the layout.

Image Retrieval (cont.)



SEARCH RESULTS

1) Portrait of Official and His Wife

UNKNOWN 15th-16th century

2) Piti Tiena (Two Sisters)

Gauguin, Paul 1892

3) Composition

Leger, Fernand 1924

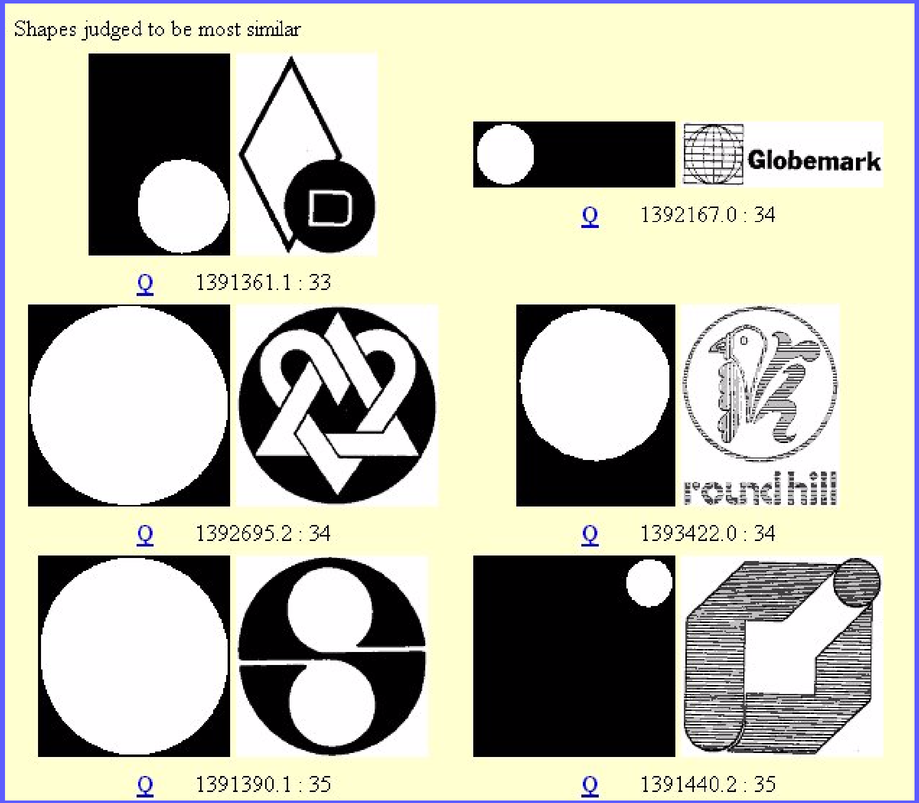
4) Stained Glass Panel: Noah's Ark (7a)

UNKNOWN Late 14th century (?)

5) Red Dancer

6) Icon: St Peter

Image Retrieval (cont.)



Shapes judged to be most similar

Q 1391361.1 : 33

Q 1392167.0 : 34

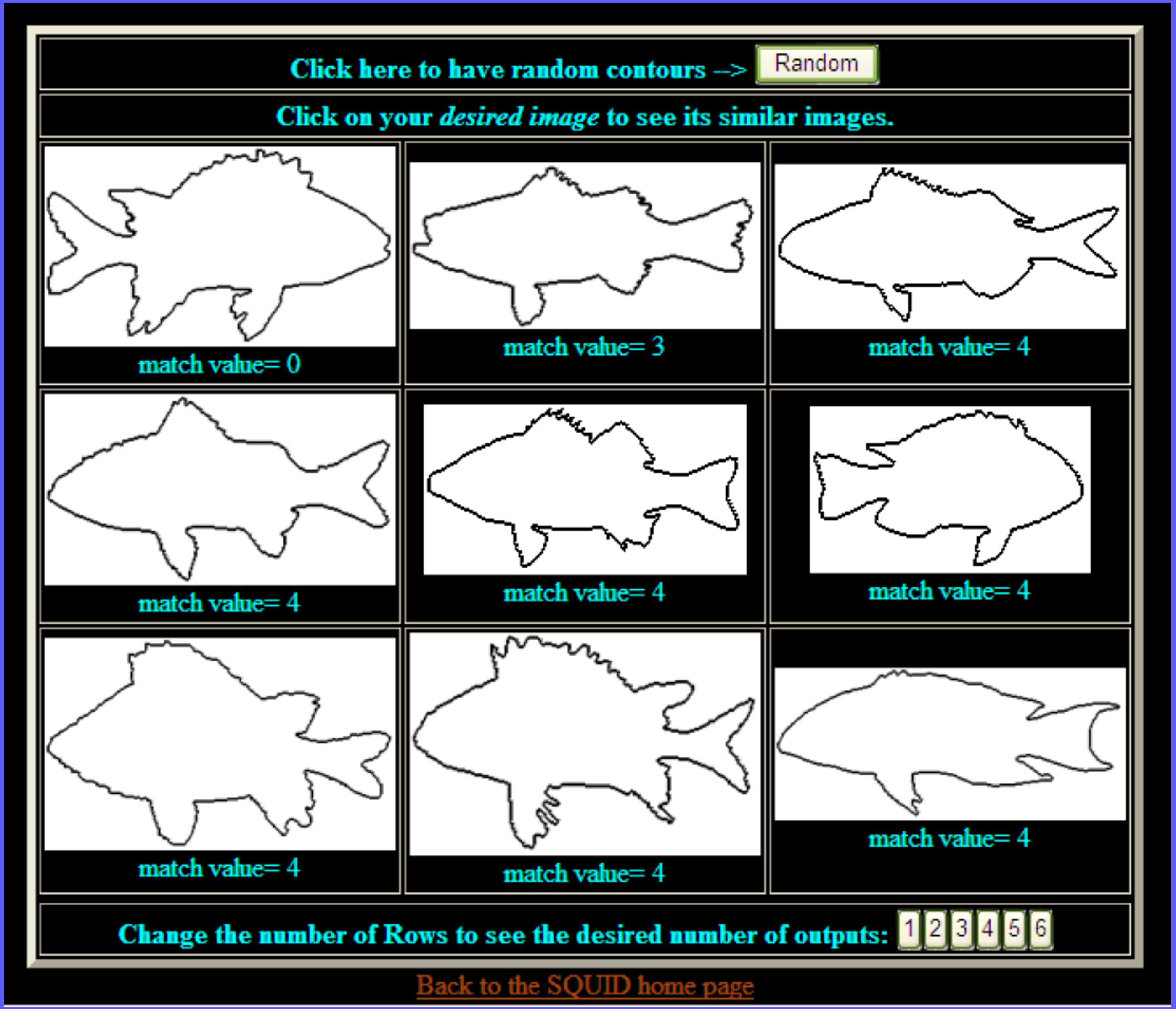
Q 1392695.2 : 34

Q 1393422.0 : 34

Q 1391390.1 : 35

Q 1391440.2 : 35

Image Retrieval (cont.)



Click here to have random contours -->

Click on your *desired image* to see its similar images.

match value= 0

match value= 3

match value= 4

match value= 4

match value= 4

match value= 4

match value= 4

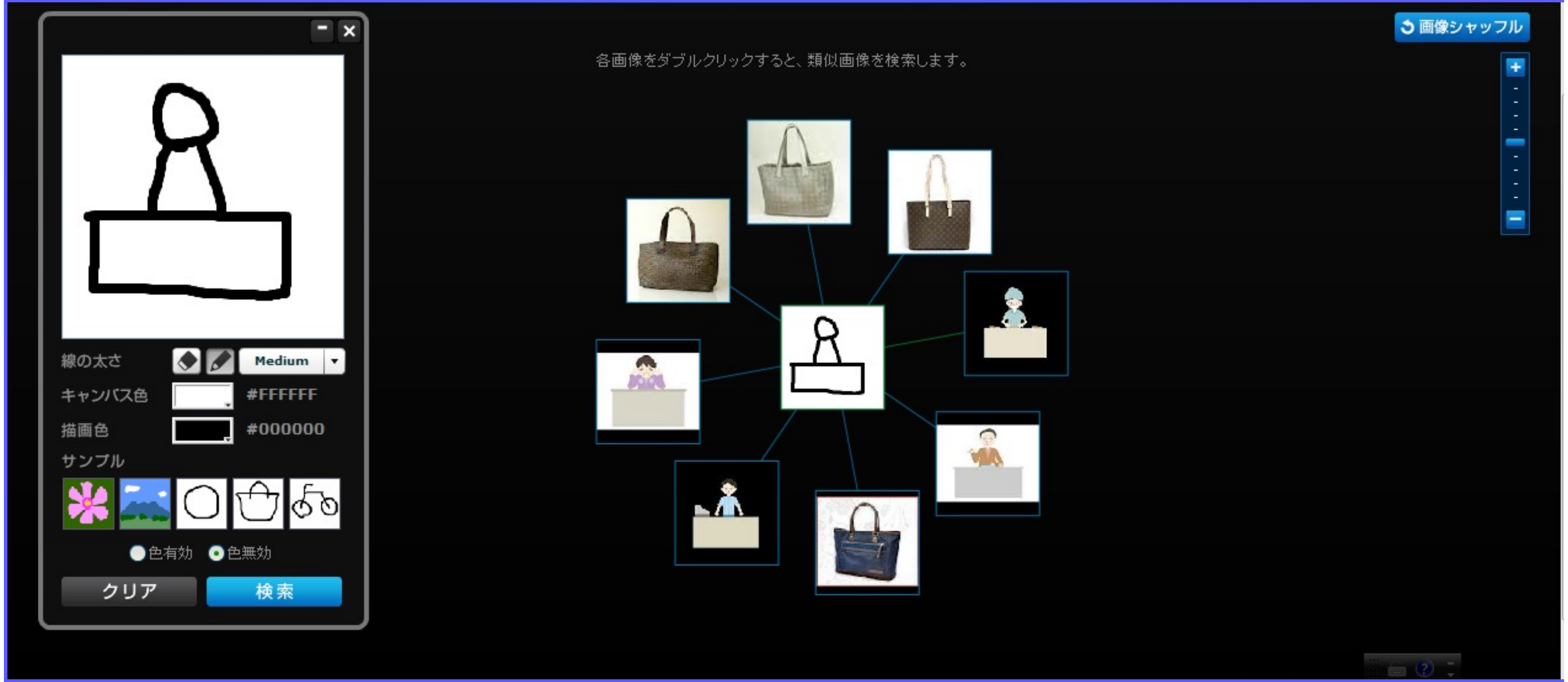
match value= 4

match value= 4

Change the number of Rows to see the desired number of outputs:

Back to the SQUID home page

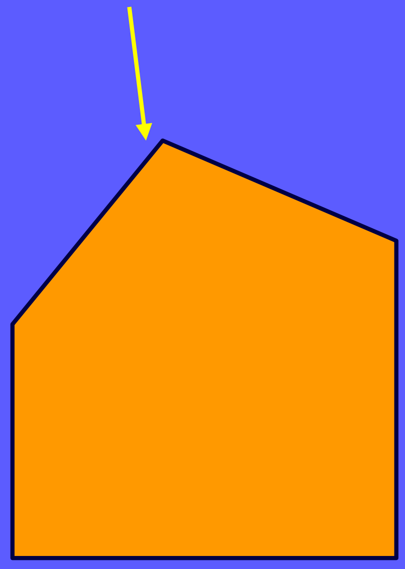
Image Retrieval (cont.)



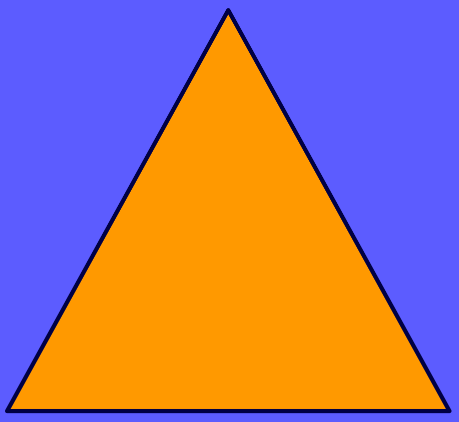
各画像をダブルクリックすると、類似画像を検索します。

線の太さ Medium
キャンバス色 #FFFFFF
描画色 #000000
色有効 [✓ 色無効]

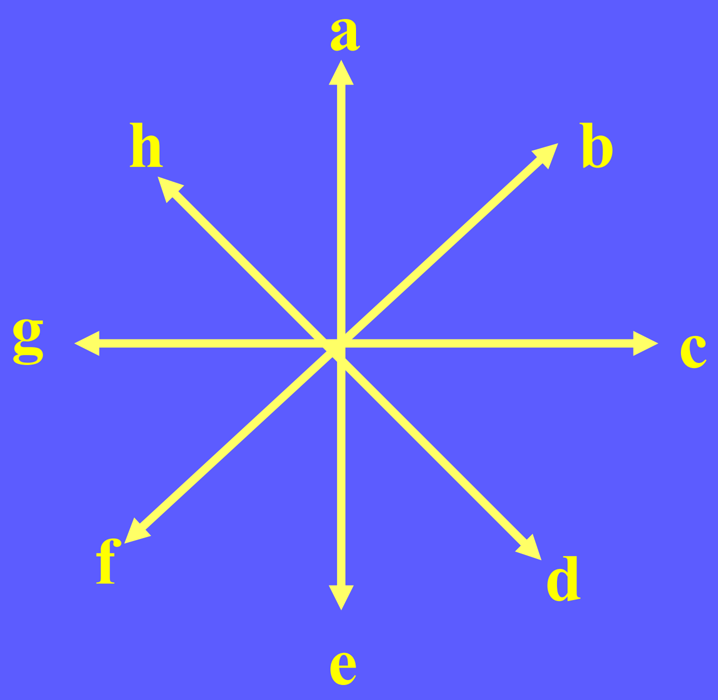
Chain Code



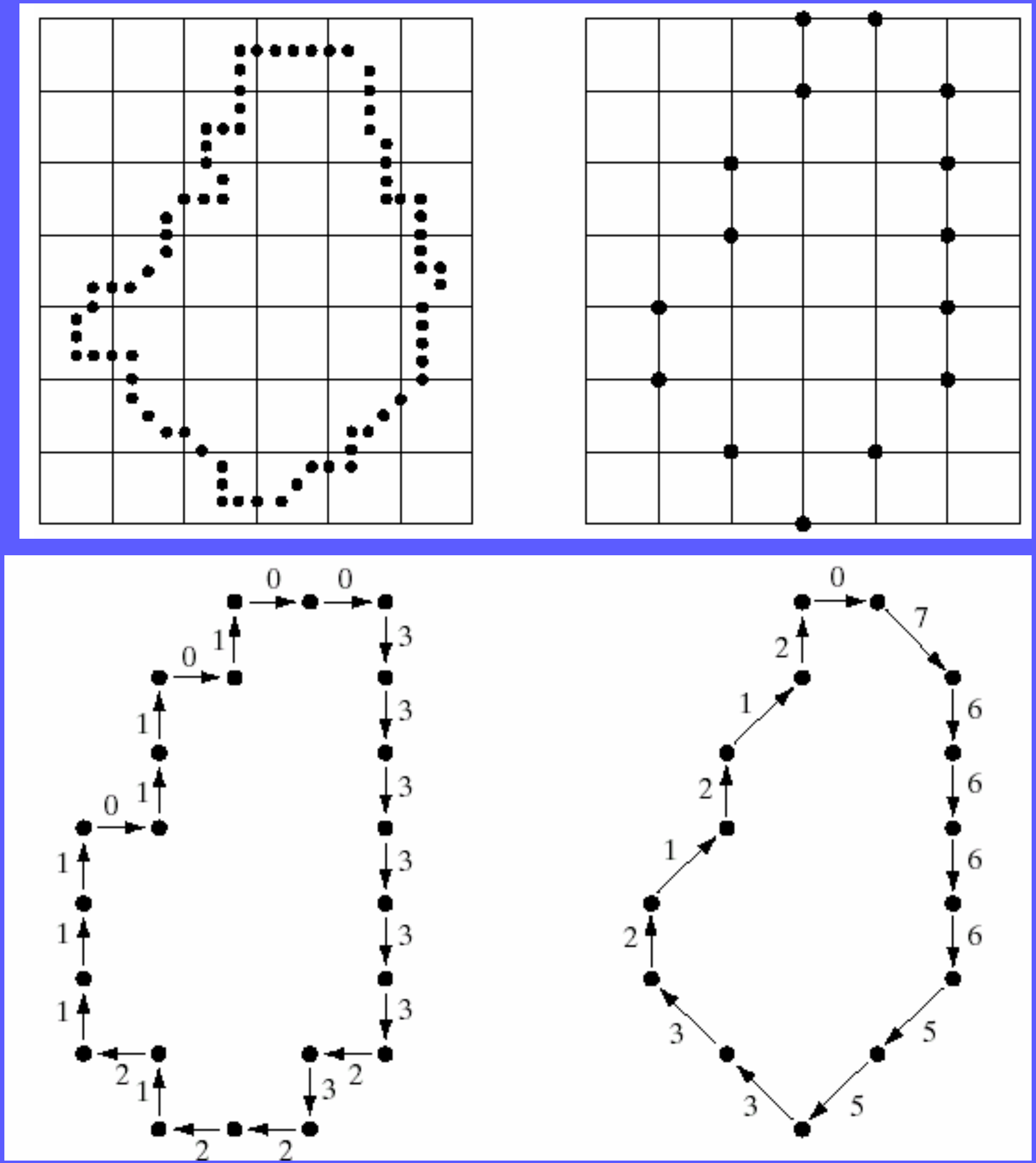
ddeeeegggaabb



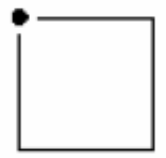
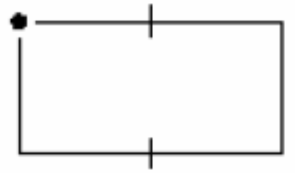
dddgggbbb



Chain Code (cont.)

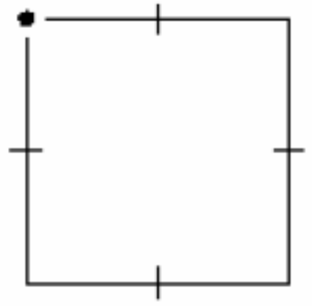
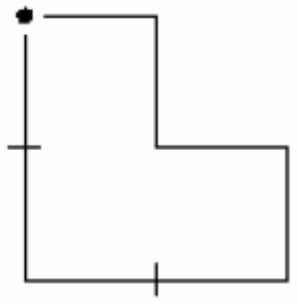
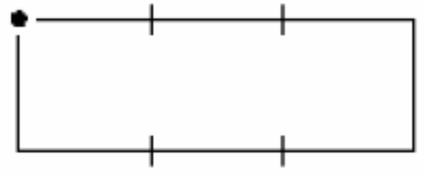


Chain Code (cont.)

	Order 4	Order 6
Chain code:	0 3 2 1	0 0 3 2 2 1
Difference:	3 3 3 3	3 0 3 3 0 3
Shape no.:	3 3 3 3	0 3 3 0 3 3

Order 8

Chain code:	0 0 3 3 2 2 1 1	0 3 0 3 2 2 1 1	0 0 0 3 2 2 2 1
Difference:	3 0 3 0 3 0 3 0	3 3 1 3 3 0 3 0	3 0 0 3 3 0 0 3
Shape no.:	0 3 0 3 0 3 0 3	0 3 0 3 3 1 3 3	0 0 3 3 0 0 3 3

Algorithm of Sequence Comparisons

- C[i, j]: minimum cost of changing

  $a_1a_2\ldots a_i$ to $b_1b_2\ldots b_j$

- Hypothesis:

  we know C[x, y], for all $0 \leq x \leq i$, $0 \leq y \leq j$

- e.g.
  - abb -> ba, min. cost = 2, abb -> bab, min. cost = ?
  - ab -> bab, min. cost = 1, abb -> bab, min. cost = ?
  - ab -> ba, min. cost = 2, abb -> bab, min. cost = ?

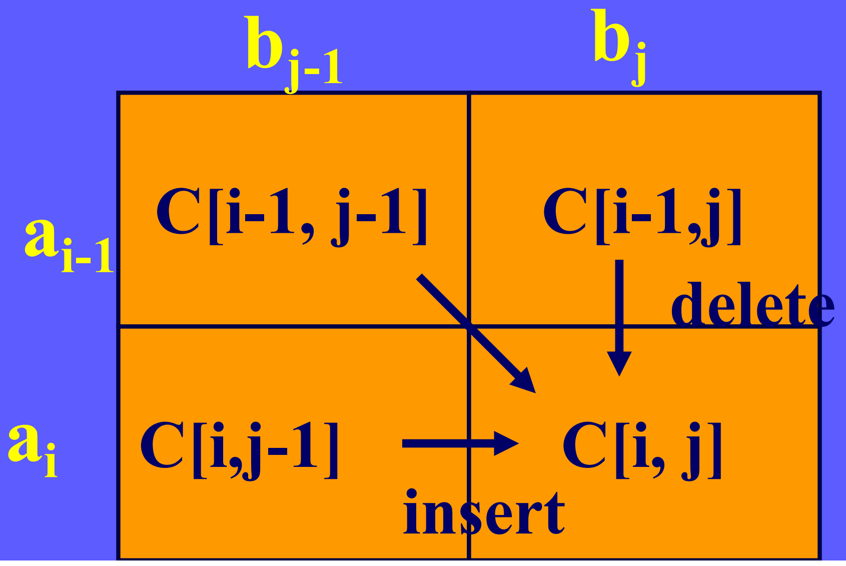
Algorithm of Sequence Comparisons

- C[i, j]: minimum cost of changing

  $a_1a_2\ldots a_i$ to $b_1b_2\ldots b_j$

$$C(i, j) = \mathbf{min}\begin{cases} C(i-1, j)+\mathbf{1} & \text{deleting } a_i \\ C(i, j-1)+\mathbf{1} & \text{inserting } b_j \\ C(i-1, j-1)+d(i, j) & \text{replacing} \end{cases}$$

$$d(i, j) = \begin{cases} \mathbf{0} & \text{if } a_i = b_j \\ \mathbf{1} & \text{if } a_i \neq b_j \end{cases}$$



Example of Algorithm of Sequence Comparisons

		b	a	b	b
	0	1	2	3	4
a	1	1	1	2	3
b	2	1	2	1	2
b	3	2	2	2	1
c	4	3	3	3	2

Example of Algorithm of Sequence Comparisons

		b	a	b	b
	0	1	2	3	4
a	1	1	1	2	3
b	2	1	2	1	2
b	3	2	② →	3	1
c	4	3	3	3	2

Example of Algorithm of Sequence Comparisons

		b	a	b	b
	0	1	2	3	4
a	1	1	1	2	3
b	2	1	2	1	2
b	3	2	2	2	1
c	4	3	3	3	2

Example of Algorithm of Sequence Comparisons

	b	a	b	b
⓪	①	2	3	4
a	1	1	①	2	3
b	2	1	2	①	2
b	3	2	2	2	①
c	4	3	3	3	②

Algorithm

```
Input: A, B
Output: C
Begin
    for i:=0 to n do C[i, 0]:=i;
    for j:=o to m do C[0, j]:=j;
    for i:=1 to n do
        for j:=1 to m do
            x:=C[i-1, j]+1;
            y:=C[i, j-1]+1;
            if $a_i = b_j$ then
                z:=C[i-1, j-1]
            else
                z:=C[i-1, j-1]+1
            C[i, j]:=min(x,y,z)
End
```

如果insertion cost為2, deletion cost為3,
演算法要如何修改？

如果不允許replacement，演算法要如何修改？

如果除了insertion, deletion, replacement, 還允許swap adjacent characters, 演算法要如何修改？

Longest Common Subsequence (LCS)

論文防抄襲！政大、清大、交大　想畢業得先過「比對系統」

2019/08/05 13:29:00

讚 72

生活中心 / 綜合報導

學術倫理不容挑戰！日前陽明大學生物藥學研究所所長蕭崇瑋因5篇升等論文被發現造假，遭校方降級處分，陽明大學當時強調，學術倫理是學者基本，要師生務必守住。如今，為防止論文抄襲，清華大學、交通大學以及政治大學3校，公布自108學年度起，碩士與博士生的論文都要先經過「比對系統」，完成「論文相似度比對」確認沒有出現抄襲情形後，才能進到口試階段。

畢業論文改二
作者: win7

畢業論文最新改版
作者: win7

畢業論文最新改版二
作者: win7

畢業論文超級無敵最終版本
作者: win7

畢業論文超級絕對一定無敵最終版本
作者: win7

我與老師

▲為防抄襲，清大、交大、政大公布，往後碩博生論文得先經過「論文相似度比對系統」審核，才能進入口試。（圖 / 資料照）

據《聯合新聞網》報導，清華大學教務長戴念華表示，送文章到國外期刊發表時，都會被要求進行比對，如果相似度過高，會直接被退件，而國際期刊的標準較嚴格，相似度必須在20％左右。

Longest Common Subsequence

- Subsequence

String $S = s_1 s_2 \ldots s_n$ is a subsequence of string $A = a_1 a_2 \ldots a_m$
if $\exists$ monotic function $F:\{1,2,\ldots,n\} \rightarrow \{1,2,\ldots,m\}$
such that $F(i) = k$ only if $s_i = a_k$

monotonic function : if $F(i) = k$, $F(j) = l$ and $i < j$, then $k < l$

<e.g.>

string "lort" is a subsequence of string "a$\underline{l}$g$\underline{or}$i$\underline{t}$hm"

Longest Common Subsequence (cont.)

- Common Subsequence

  string S is a common subsequence of string A and B

  if S is a subsequence of A and

  also a subsequence of string B

<e.g.>

common subsequence of “algorithm” and “belowart”

“lo”, “lr”, lt”, “or”, “ot”,”rt”,”ar”,

’lor”,”lot”,”lrt”,”art”,

”lort”

Longest Common Subsequence (cont.)

- **Longest Common Subsequence (LCS)**

string S is longest common subsequence of string A and B if S is common subsequence of A and B of maximal length

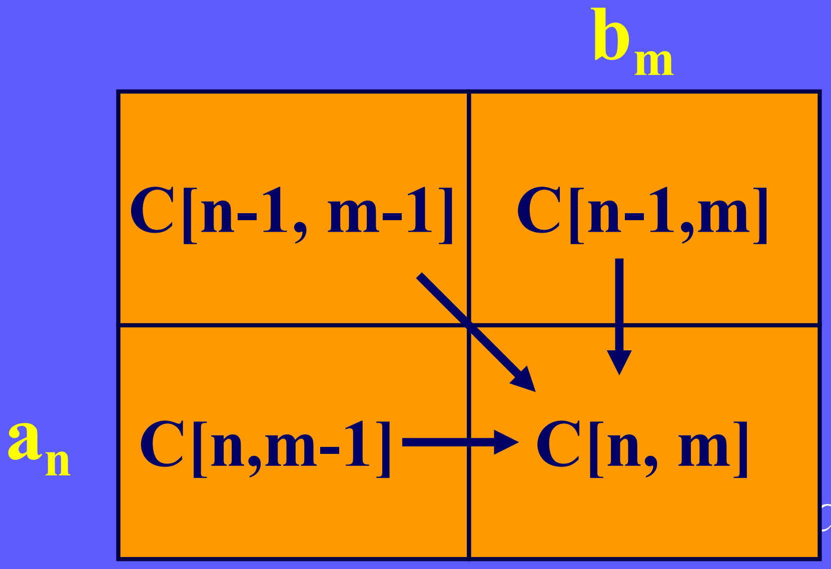
<e.g.>

The LCS of "algorithm" and "belowart" is "lort"

Longest Common Subsequence (cont.)

$$L(n,m) = \max \begin{cases} L(n-1,m) \\ L(n,m-1) \\ L(n-1,m-1) + c(n,m) \end{cases}$$

$$c(i,j) = \begin{cases} 1 & \text{if } a_i = b_j \\ 0 & \text{if } a_i \neq b_j \end{cases}$$



		a	l	g	o	r	i	t	h	m
	0	0	0	0	0	0	0	0	0	0
b	0	0	0	0	0	0	0	0	0	0
e	0	0	0	0	0	0	0	0	0	0
l	0	0	1	1	1	1	1	1	1	1
o	0	0	1	1	2	2	2	2	2	2
w	0	0	1	1	2	2	2	2	2	2
a	0	1	1	1	2	2	2	2	2	2
r	0	1	1	1	2	3	3	3	3	3
t	0	1	1	1	2	3	3	4	4	4

Algorithm

```
Input: A, B
Output: C
Begin
    for i:=0 to n do C[i, 0]:=0;
    for j:=o to m do C[0, j]:=0;
    for i:=1 to n do
        for j:=1 to m do
            x:=C[i-1, j]+1;
            y:=C[i, j-1]+1;
            if $a_i = b_j$ then
                z:=C[i-1, j-1]+1
            else
                z:=C[i-1, j-1]
            C[i, j]:=max(x,y,z)
End
```

Global Alignment

- Given two sequences,

alignment try to match as many symbols as possible, while insertion of blank is allowed.

- e.g. to align CTTGACTAGA with CTACTGTGA

```
 C T T G A C T – A G A
 C T – – A C T G T G A
score = 7*2+1*(-1)+3*(-1) =10
…
 C T T G A C T – A G A
 C T A C T G T – – G A
 score = 5*2+4*(-1)+3*(-1) = 3
 …
```

$\Rightarrow$ alignment score = 10

Global Alignment (cont.)

- Approach: dynamic programming

  Let S(i, j) be maximum score of alignment

  between $a_1 a_2 \ldots a_i$ to $b_1 b_2 \ldots b_j$

$$S(i, j) = \max \begin{cases} S(i-1, j) + w(a_i, -) & \text{match } a_i \text{ with blank} \\ S(i, j-1) + w(-, b_j) & \text{match } b_j \text{ with blank} \\ S(i-1, j-1) + w(a_i, b_j) & \end{cases}$$

while $w(a_i, -) = -1, w(-, b_j) = -1,$

$$w(a_i, b_j) = \begin{cases} 2 & \text{if } a_i = b_j \\ -1 & \text{if } a_i \neq b_j \end{cases}$$

Global Alignment (cont.)

	a	b	b	c	a	d
	$\textcircled{0}$	-1	-2	-3	-4	-5	-6
e	$\textcircled{-1}$	-1	-2	-3	-4	-5	-6
a	-2	$\textcircled{1}$	0	-1	-2	-2	-3
c	-3	0	$\textcircled{0}$	-1	1	0	-1
b	-4	-1	2	$\textcircled{2}$	$\textcircled{1}$	$\textcircled{0}$	$\textcircled{-1}$

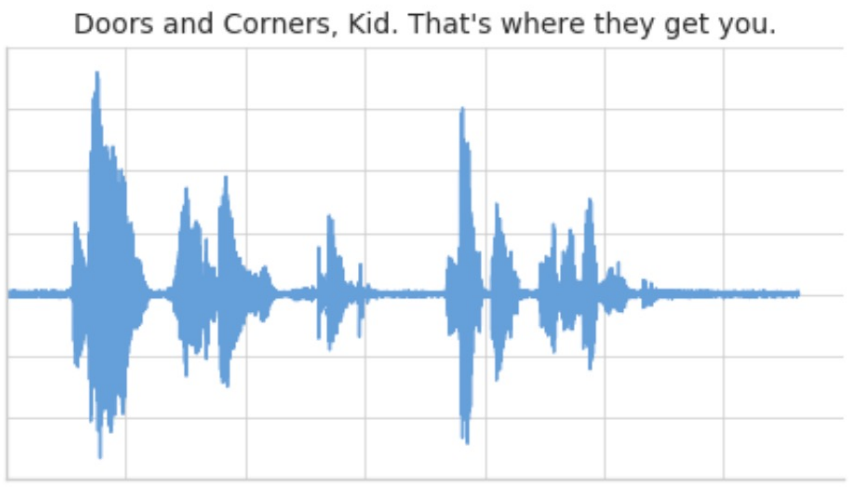
_ a b b c a d

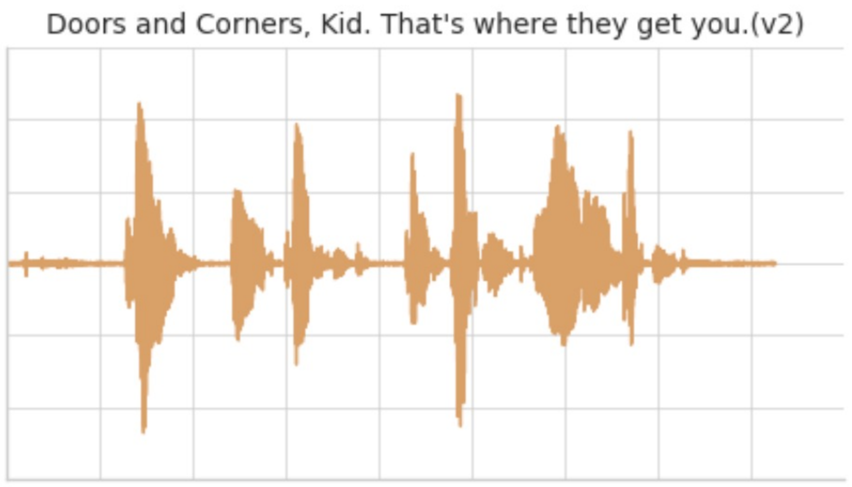
e a c b _ _ _

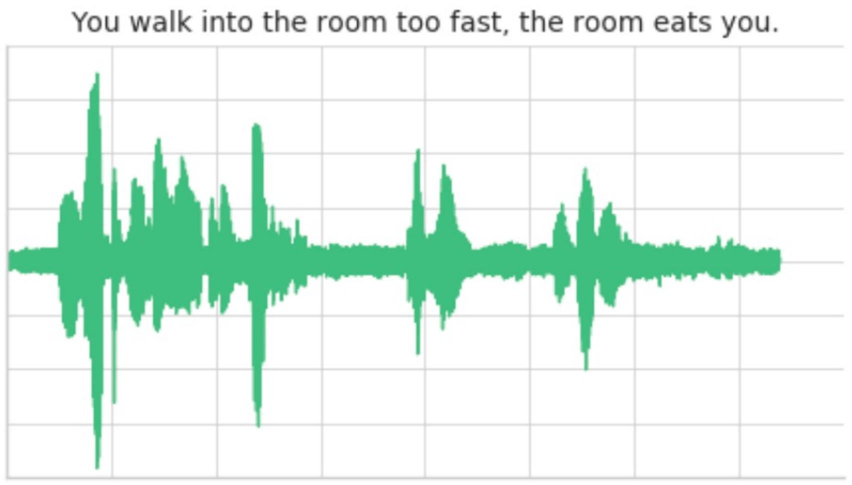
score = 2*2+1*(-1)+4*(-1) = -1

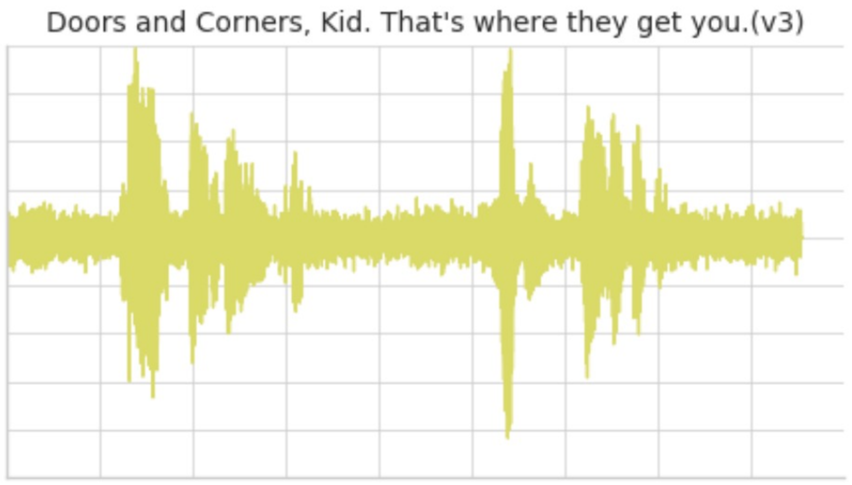
目前我們介紹的都是符號序列 (Symbol Sequence)，如何 Measure 數值序列 (Time Series) 的相似度？

Dynamic Time Warping (DTW)

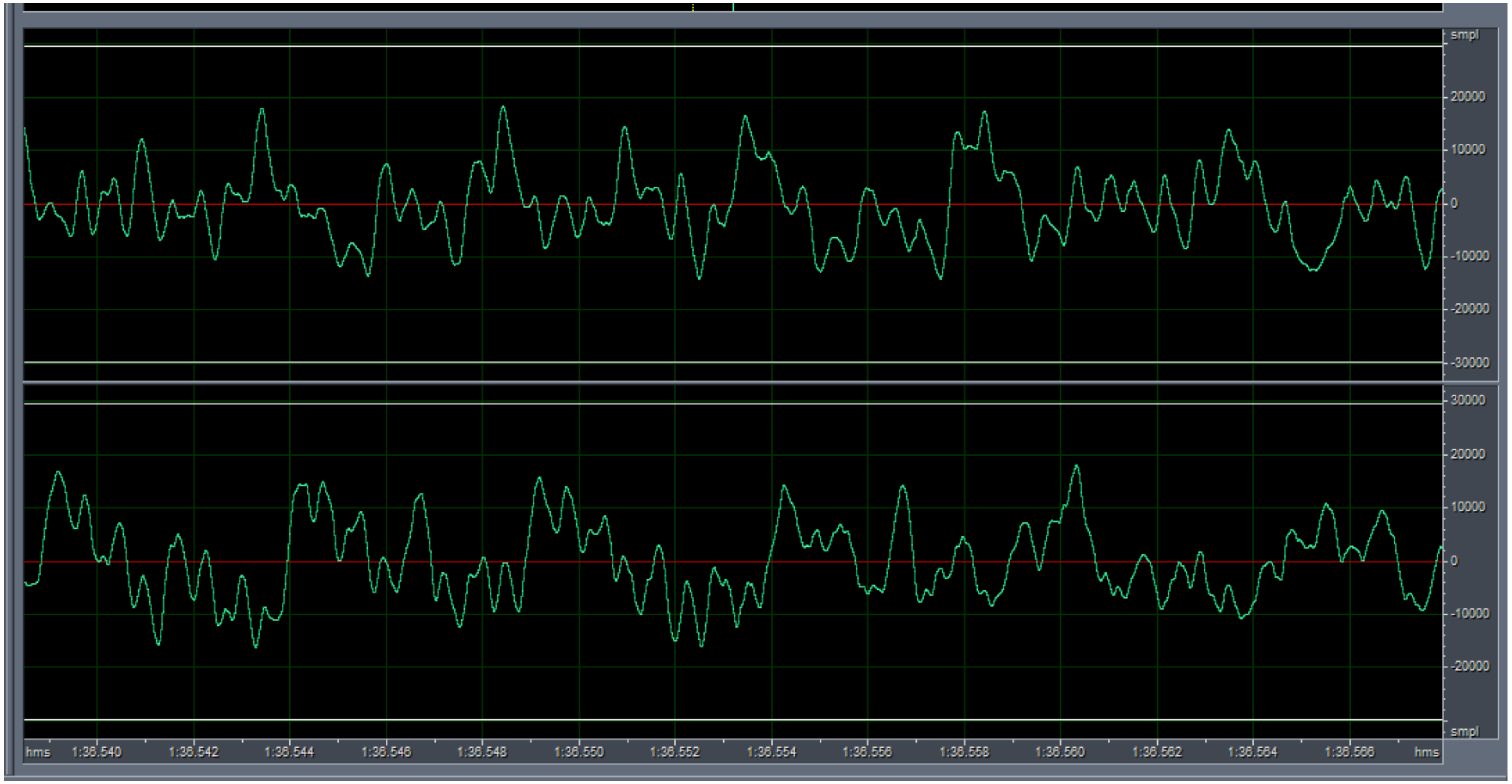




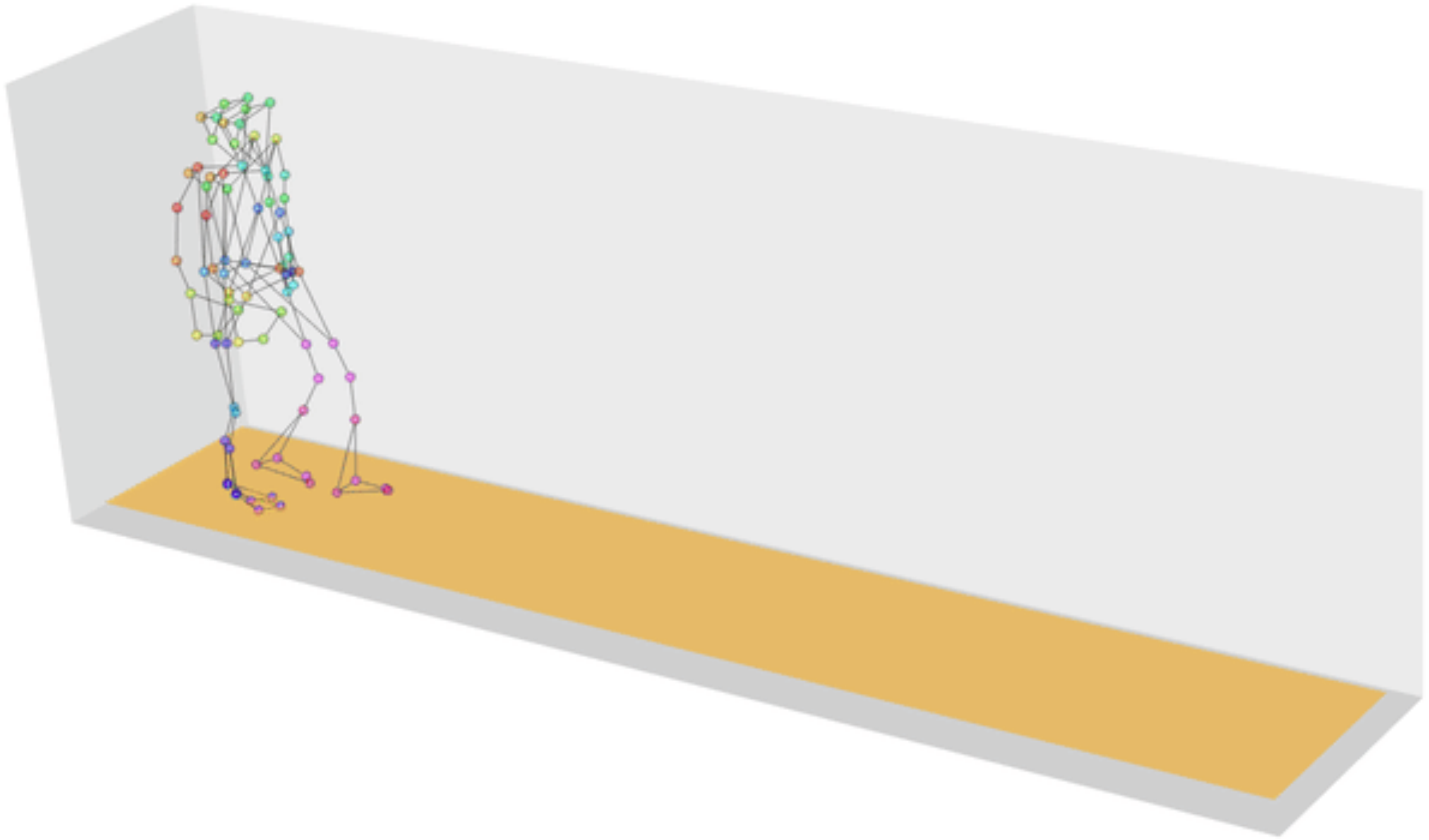




Dynamic Time Warping (cont.)

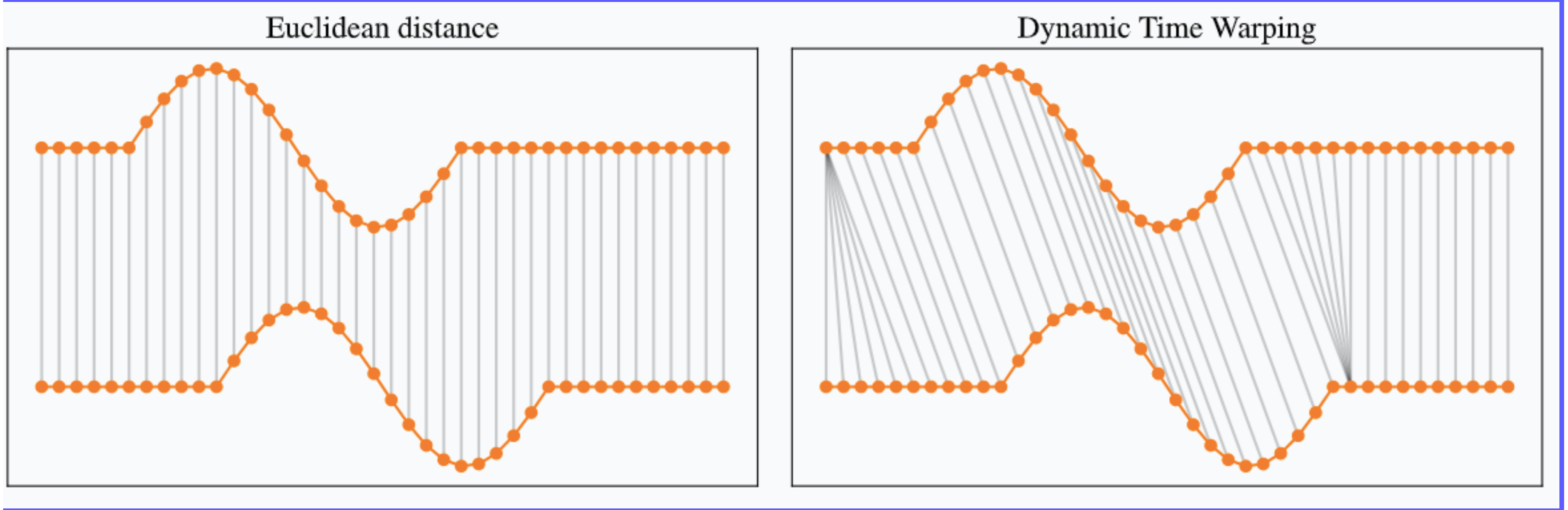


Dynamic Time Warping (cont.)

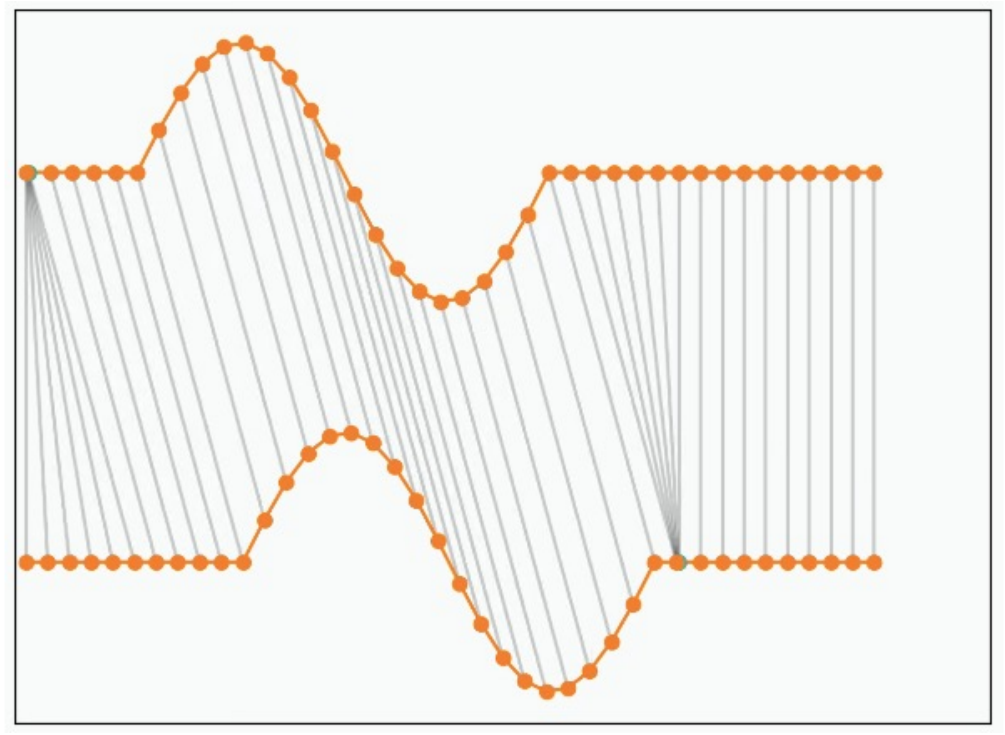


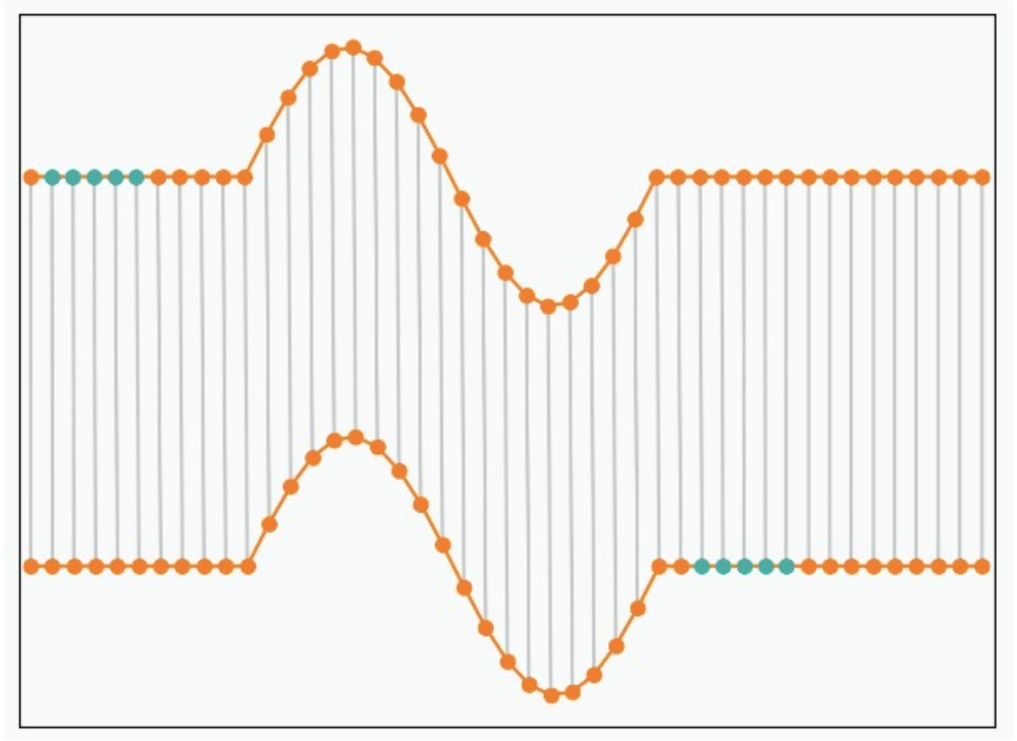
https://en.wikipedia.org/wiki/Dynamic_time_warping

Dynamic Time Warping (cont.)

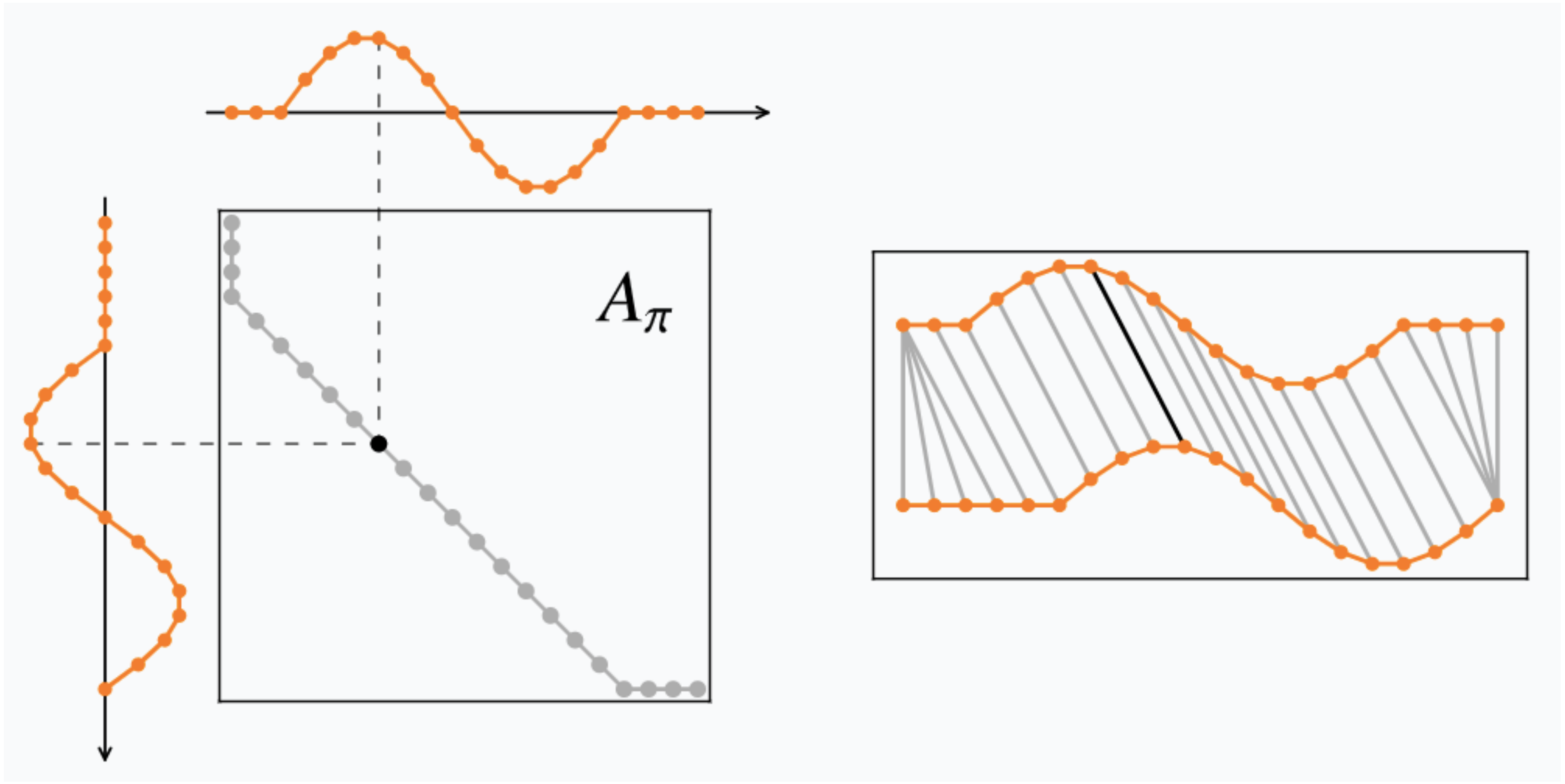


https://rtavenar.github.io/blog/dtw.html



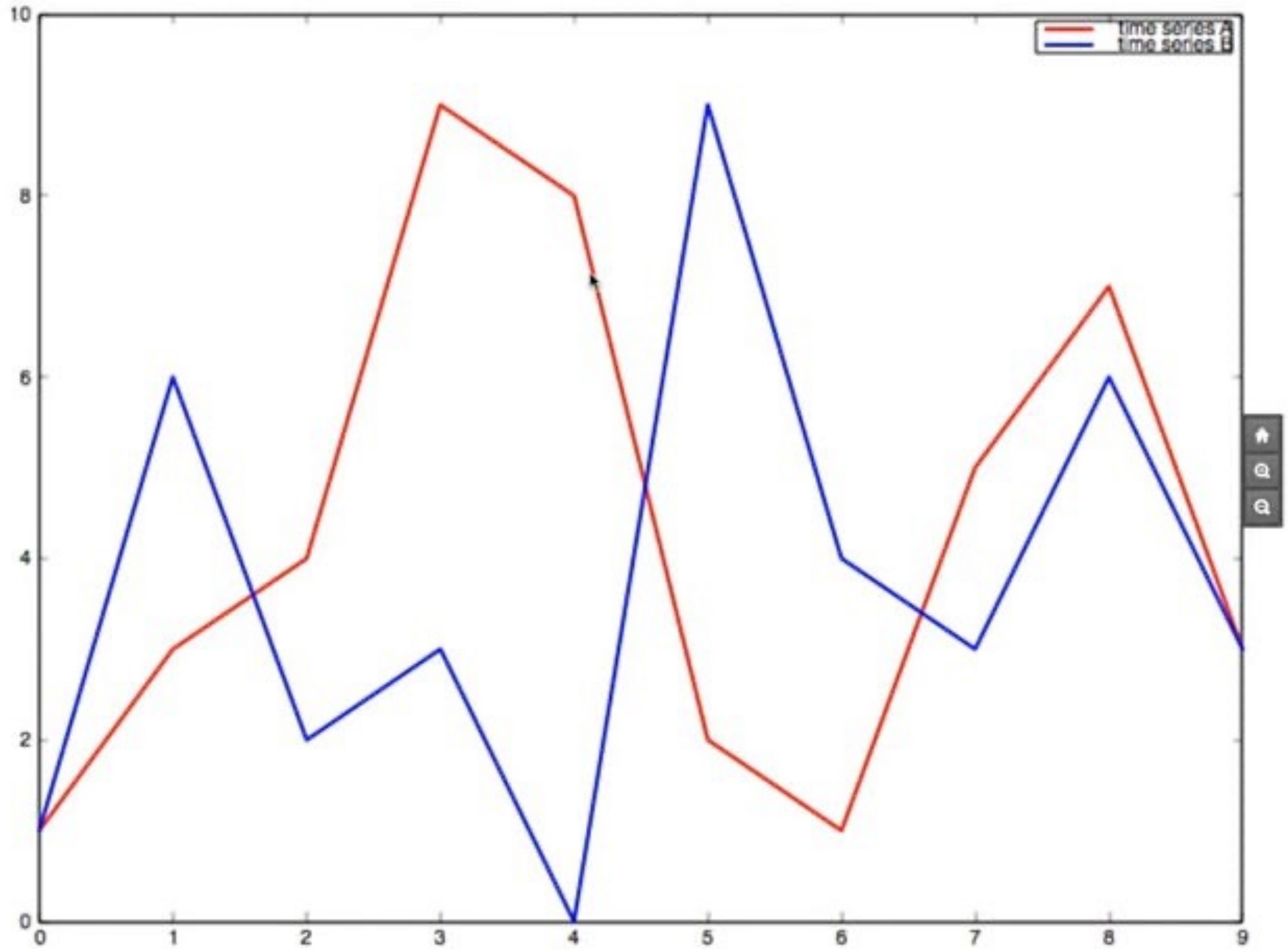


Dynamic Time Warping (cont.)



https://rtavenar.github.io/blog/dtw.html

如何以Dynamic Programming 求解 Dynamic Time Warping？



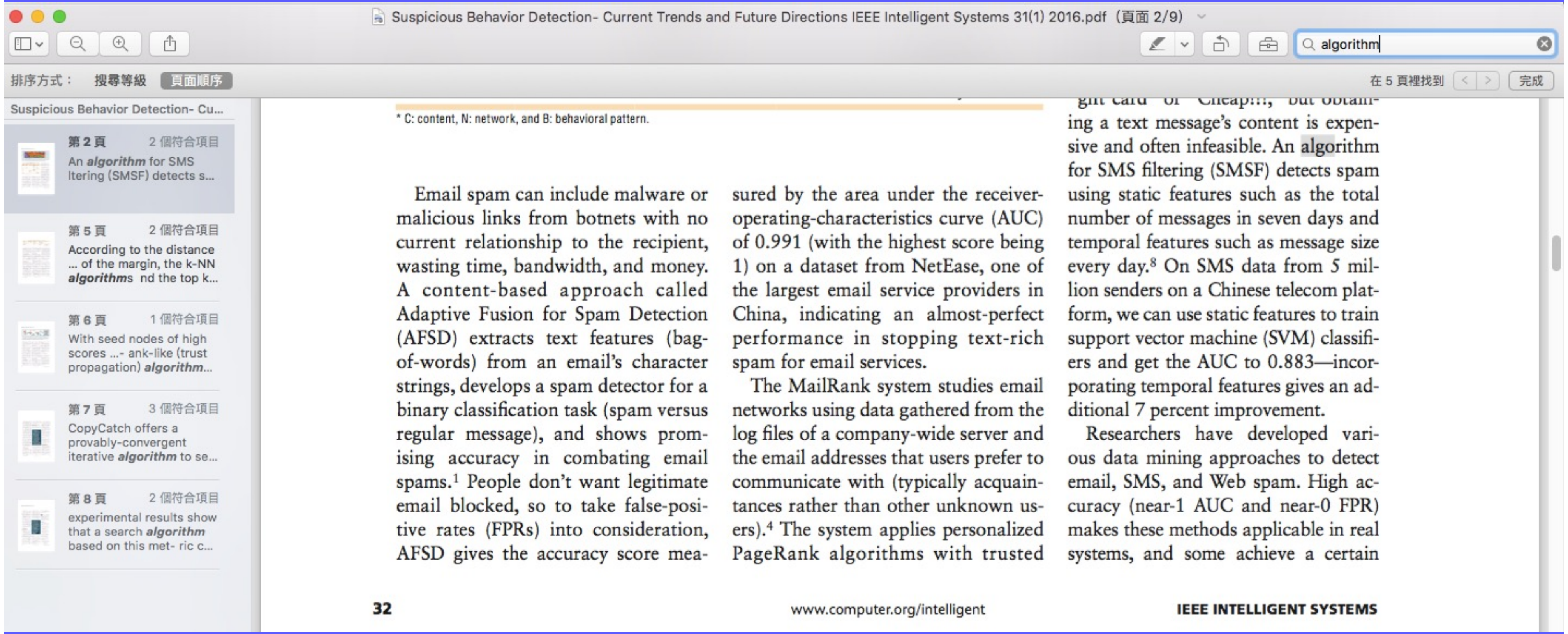
1 6 2 3 0 9 4 3 6 3

1 3 4 9 8 2 1 5 7 3

3	33	23	19	16	19	23	18	17	18	15
7	31	20	18	16	19	17	17	18	15	18
5	25	19	13	12	16	15	14	15	14	16
1	21	18	10	11	11	19	14	13	17	18
2	21	13	9	10	12	16	11	12	16	17
8	20	9	13	16	19	9	12	17	18	21
9	13	7	11	11	14	8	13	18	16	21
4	5	4	5	5	8	12	12	13	15	16
3	2	3	4	4	7	13	14	14	17	17
1	0	5	6	8	9	17	20	22	27	29
	1	6	2	3	0	9	4	3	6	3

Ref: https://medium.com/mlearning-ai/what-is-dynamic-time-warping-253a6880ad12

String Matching


Suspicious Behavior Detection- Current Trends and Future Directions IEEE Intelligent Systems 31(1) 2016.pdf（頁面 2/9）

algorithm

排序方式： 搜尋等級 頁面順序

在 5 頁裡找到 完成

Suspicious Behavior Detection- Cu...

第 2 頁 2 個符合項目
An *algorithm* for SMS lltering (SMSF) detects s...

第 5 頁 2 個符合項目
According to the distance ... of the margin, the k-NN *algorithms* nd the top k...

第 6 頁 1 個符合項目
With seed nodes of high scores ...- ank-like (trust propagation) *algorithm*...

第 7 頁 3 個符合項目
CopyCatch offers a provably-convergent iterative *algorithm* to se...

第 8 頁 2 個符合項目
experimental results show that a search *algorithm* based on this met- ric c...

* C: content, N: network, and B: behavioral pattern.

Email spam can include malware or malicious links from botnets with no current relationship to the recipient, wasting time, bandwidth, and money. A content-based approach called Adaptive Fusion for Spam Detection (AFSD) extracts text features (bag-of-words) from an email's character strings, develops a spam detector for a binary classification task (spam versus regular message), and shows promising accuracy in combating email spams.[1] People don't want legitimate email blocked, so to take false-positive rates (FPRs) into consideration, AFSD gives the accuracy score measured by the area under the receiver-operating-characteristics curve (AUC) of 0.991 (with the highest score being 1) on a dataset from NetEase, one of the largest email service providers in China, indicating an almost-perfect performance in stopping text-rich spam for email services.

The MailRank system studies email networks using data gathered from the log files of a company-wide server and the email addresses that users prefer to communicate with (typically acquaintances rather than other unknown users).[4] The system applies personalized PageRank algorithms with trusted

ing a text message's content is expensive and often infeasible. An algorithm for SMS filtering (SMSF) detects spam using static features such as the total number of messages in seven days and temporal features such as message size every day.[8] On SMS data from 5 million senders on a Chinese telecom platform, we can use static features to train support vector machine (SVM) classifiers and get the AUC to 0.883—incorporating temporal features gives an additional 7 percent improvement.

Researchers have developed various data mining approaches to detect email, SMS, and Web spam. High accuracy (near-1 AUC and near-0 FPR) makes these methods applicable in real systems, and some achieve a certain

32 www.computer.org/intelligent IEEE INTELLIGENT SYSTEMS

String Matching

- **Given a text** $A=a_1a_2\ldots a_n$

  a pattern $B=b_1b_2\ldots b_m$

  find the first occurrence of B in A

  = find the smallest k

  such that $\forall$ i, $1 \le i \le m$, $a_{k+i} = b_i$
- **Substring of a string A**

  a consecutive sequence of characters $a_ia_{i+1}\ldots a_j$

* $A(i)= a_1a_2\ldots a_i$

String Matching (cont.)

- Applications:
  - **text editor**
  - **molecular biology (RND, DNA sequence)**
- Solutions
  - **brute force: O(m × n)**
  - **KMP algorithm: O(n)**
  - **Boyer & Moore algorithm: O(n)**
  - **Finite state machine**
  - **…**

Brute Force: An Example

A= xyxxyxyxyyxyxyxyyxyxyxx, B=xyxyyxyxyxx

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
x	y	x	x	y	x	y	x	y	y	x	y	x	y	x	y	y	x	y	x	y	x	x
x	y	x	y																			
	x																					
		x	y																			
			x	y	x	y	y															
				x																		
					x	y	x	y	y	x	y	x	y	x	x							
						x																
							x	y	x													
								x														
									x													
										x	y	x	y	y								
											x											
												x	y	x	y	y	x	y	x	y	x	x

A Worst Case

A= yyyyyyyyyyyyx, B=yyyyx

1	2	3	4	5	6	7	8	9	10	11	12	13
y	y	y	y	y	y	y	y	y	y	y	y	x
y	y	y	y	x								
	y	y	y	y	x							
		y	y	y	y	x						
			y	y	y	y	x					
				y	y	y	y	x				
					y	y	y	y	x			
						y	y	y	y	x		
							y	y	y	y	x	
								y	y	y	y	x

A Better Case

A= yyyyyyyyyyyyx, B=xyyyyy

1	2	3	4	5	6	7	8	9	10	11	12	13
y	y	y	y	y	y	y	y	y	y	y	y	x
x												
	x											
		x										
			x									
				x								
					x							
						x						
							x					
								x				

Brute Force的String Matching
在Worst Case時間複雜度是多少？

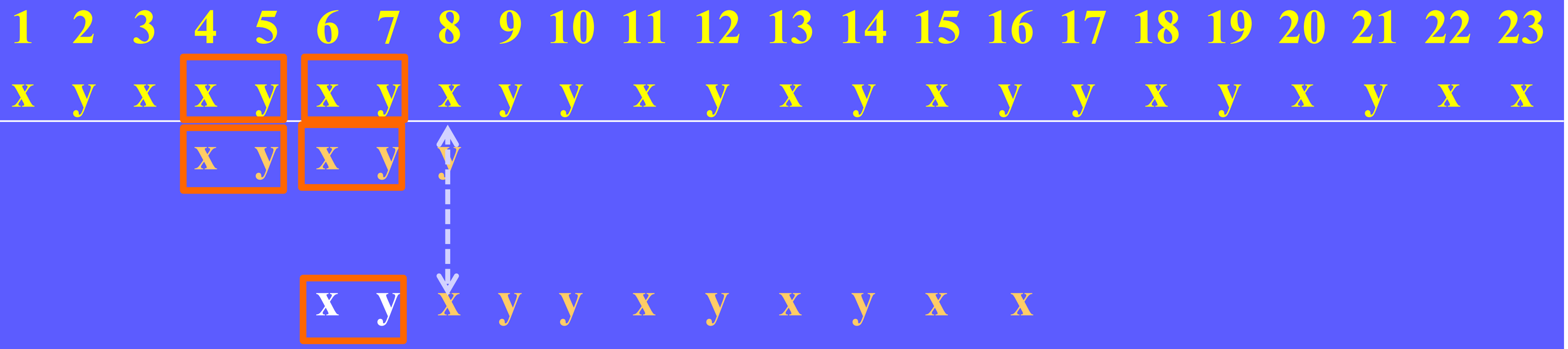
Observation

- Worst case: O(m x (n-m+1))
- Improvement:
  - **shift matching to avoid redundant matching**
  - **preprocessing of B to construct the NEXT table**
  - **NEXT table = how far to shift**

  *** independent of A**

Observation of An Example

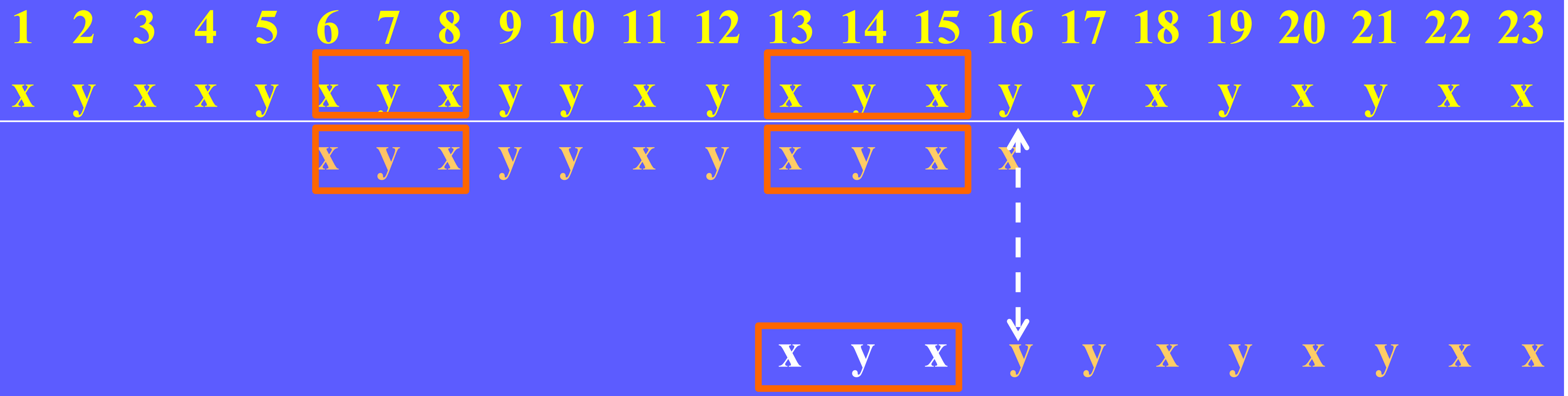
A= xyxxyxyxyyxyxyxyyxyxyxx, B=xyxyyxyxyxx

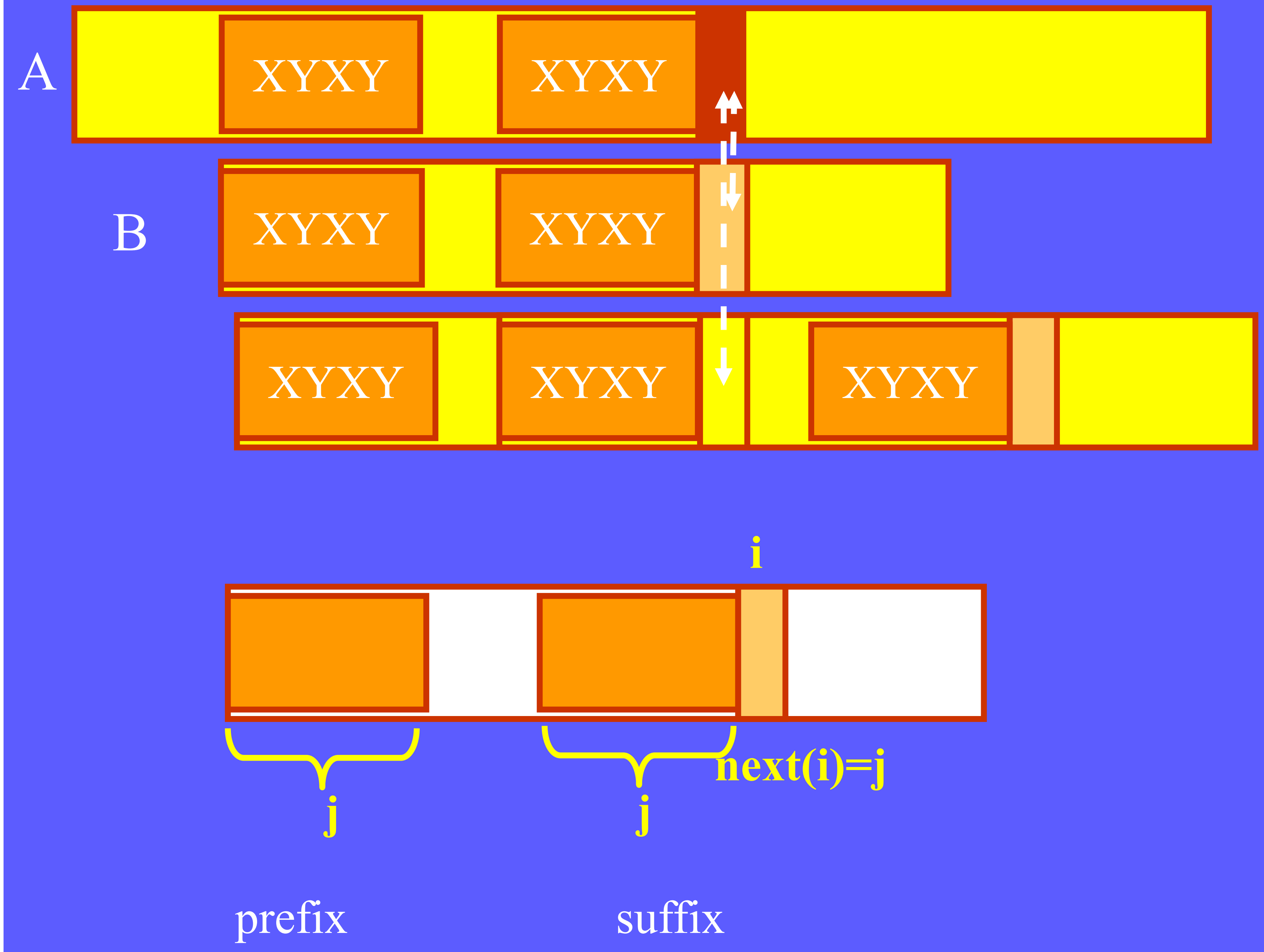


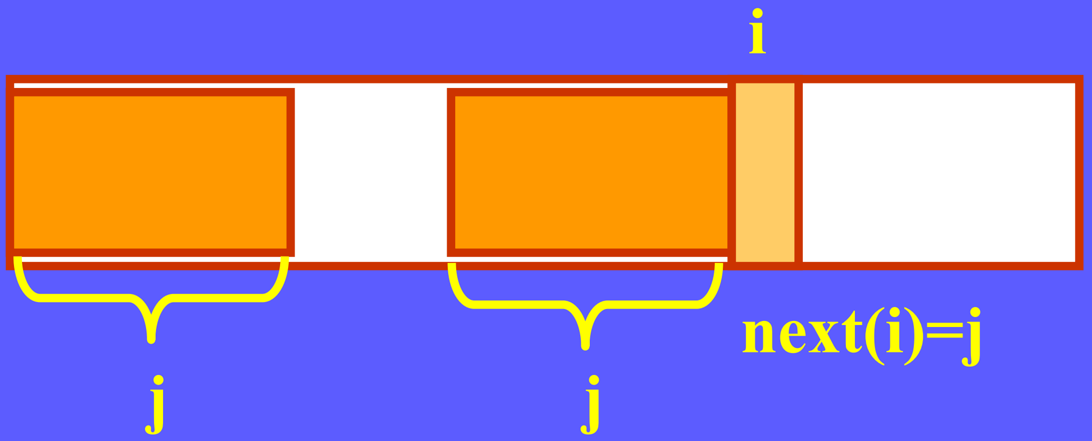
M. K. Shan, CS, NCCU

Observation of An Example

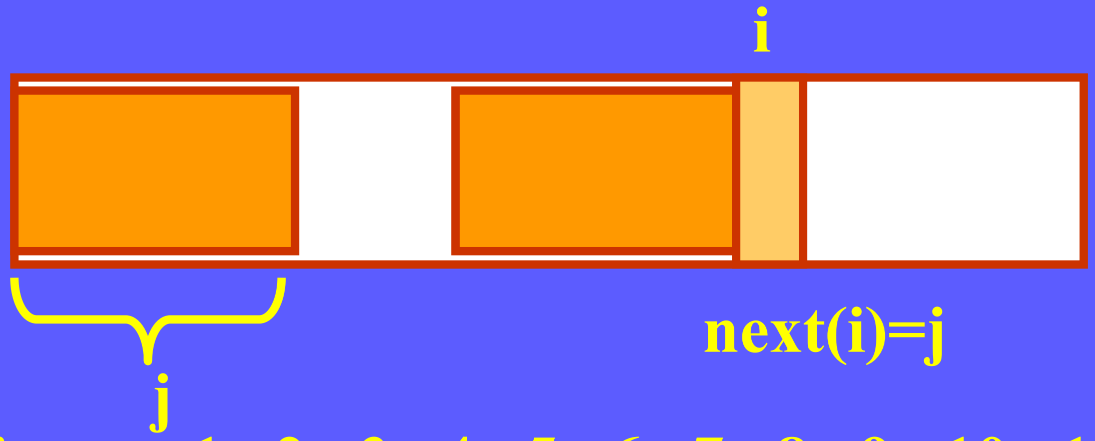
A= xyxxyxyxyyxyxyxyyxyxyxx, B=xyxyyxyxyxx




A
XYXY
XYXY
B
XYXY
XYXY
XYXY
XYXY
XYXY
i
next(i)=j
j
j
prefix
suffix



i=	1	2	3	4	5	6	7	8	9	10	11
B=	x	y	x	y	y	x	y	x	y	x	x
next=	-1	0	0	1	2	0	1	2	3	$\textcircled{4}$	3



i=	1	2	3	4	5	6	7	8	9	10	11
B=	x	y	x	y	y	x	y	x	y	x	x
next=	-1	0	0	1	2	0	1	2	3	4	3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
x	y	x	x	y	x	y	x	y	y	x	y	x	y	x	y	y	x	y	x	y	x	x
					x	y	x	y	y	x	y	x	y	x	x							
						x																
							x	y	x													
								x														
									x													
										x	y	x	y	y								
											x											
												x	y	x	y	y	x	y	x	y	x	x

Observation of An Example

-1 0 0 1 2 0 1 2 3 4 3

A= xyxxyxyxyyxyxyxyyxyxyxx, B=xyxyyxyxyxx

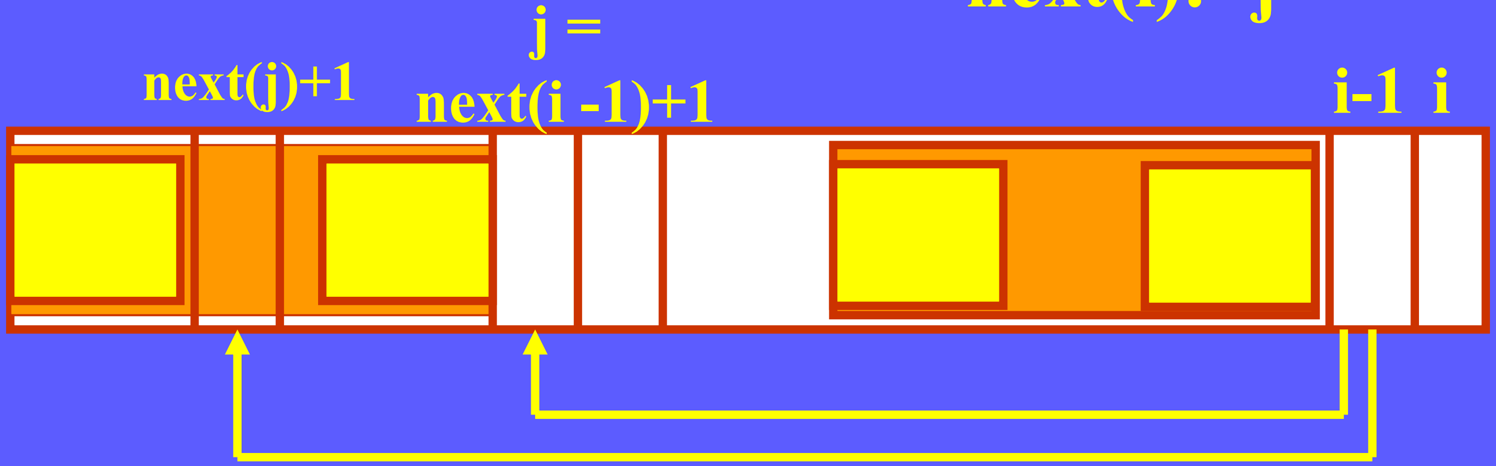
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
x	y	x	x	y	x	y	x	y	y	x	y	x	y	x	y	y	x	y	x	y	x	x
x	y	x	y																			
	x																					
		x	y																			
			x	y	x	y	y															
				x																		
					x	y	x	y	y	x	y	x	y	x	x							
						x																
							x	y	x													
								x														
									x													
										x	y	x	y	y								
											x											
												x	y	x	y	y	x	y	x	y	x	x

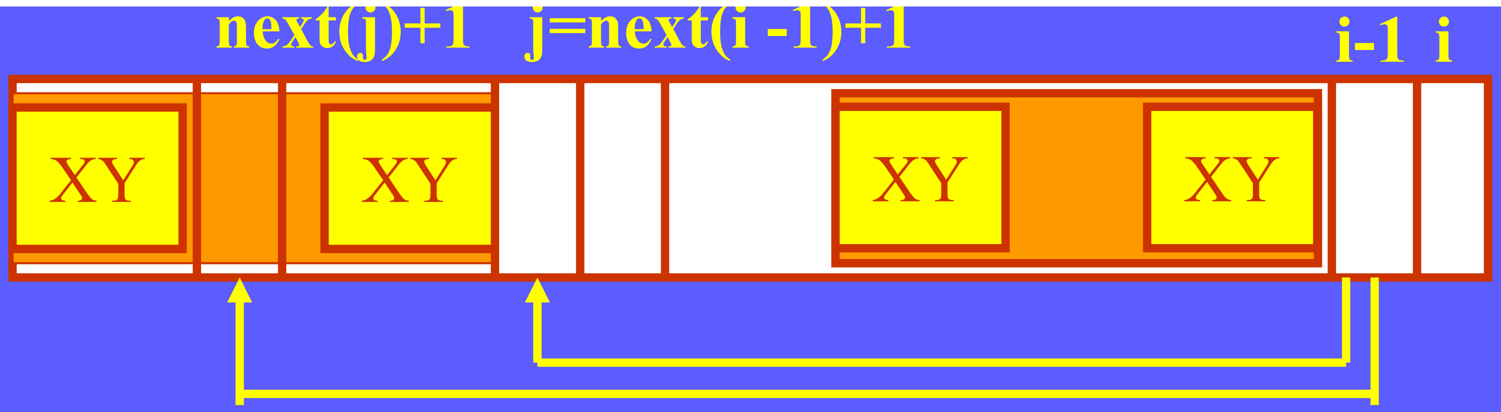
```
Algorithm String_Match(A,n,B,m);
Input: A (a string of size n), B (a string of size m)
Output: Start
begin
    j:=1; i:=1;
   Start:=0;
   While Start = 0 and i <= n do
          if B[j] = A[i] then
              j := j+1;
              i := i+1
          else
              j := next[j] + 1;                    /slide position
              if  j=0 then
                  j := 1;
                  i :=  i+1;
          if j=m+1 then Start := i-m       /found
end
```

Computing Next

- Base: next(1)=-1, next(2)=0
- Hypothesis: next(j), $1 \le j \le i-1$ have been computed
- Induction on next(i)
  - case 1 ($b_{i-1} = b_{next(i-1)+1}$): next(i)=next(i-1)+1
  - case 2 ($b_{i-1} \neq b_{next(i-1)+1}$):

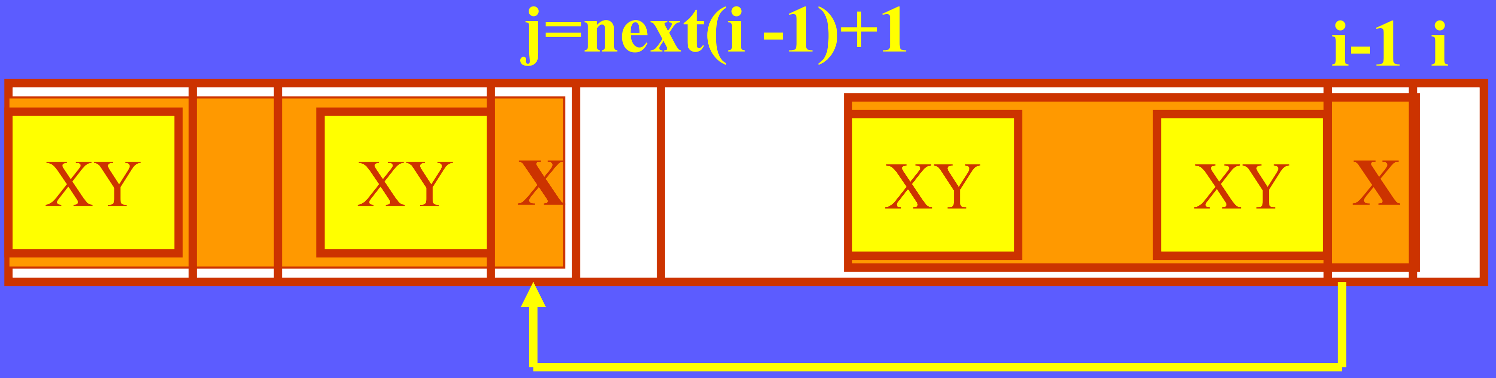
```
while b_{i-1} <> b_j and j >0 do
        j:=next(j)+1;
next(i):=j
```





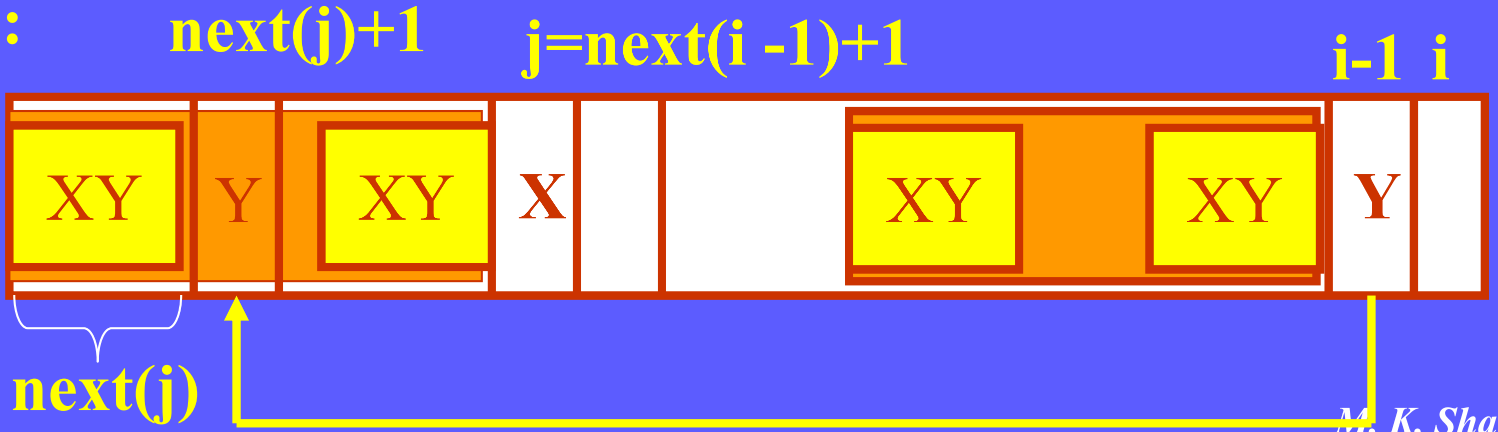
next(j)+1
j=next(i -1)+1
i-1
i
XY
XY
XY
XY

Case 1:



j=next(i -1)+1
i-1
i
XY
XY
X
XY
XY
X

Case 2:



next(j)+1
j=next(i -1)+1
i-1
i
XY
Y
XY
X
XY
XY
Y
next(j)

case 1 ($b_{i-1} = b_{next(i-1)+1}$):

next(i)=next(i-1)+1

i=	1	2	3	4	5	6	7	8	9	10	11
B=	x	y	x	$\textcircled{y}$	y	x	y	x	$\textcircled{y}$	x	x
next=	-1	0	0	1	2	0	1	2	3	4	

case 2 ($b_{i-1} \neq b_{next(i-1)+1}$):

```
while $b_{i-1}$ <> $b_j$ and j >0 do
j:=next(j)+1;
next(i):=j
```

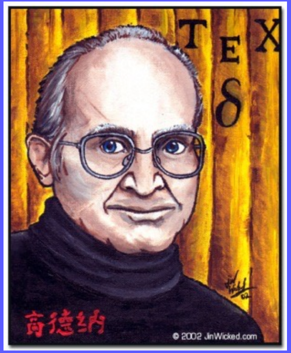
i=	1	2	3	4	5	6	7	8	9	10	11
B=	x	y	x	y	$\textcircled{y}$	x	y	x	y	$\textcircled{x}$	x
next=	-1	0	0	1	2	0	1	2	3	4	

i=	1	2	3	4	5	6	7	8	9	10	11
B=	x	y	$\textcircled{x}$	y	y	x	y	x	y	$\textcircled{x}$	x
next=	-1	0	0	1	2	0	1	2	3	4	3

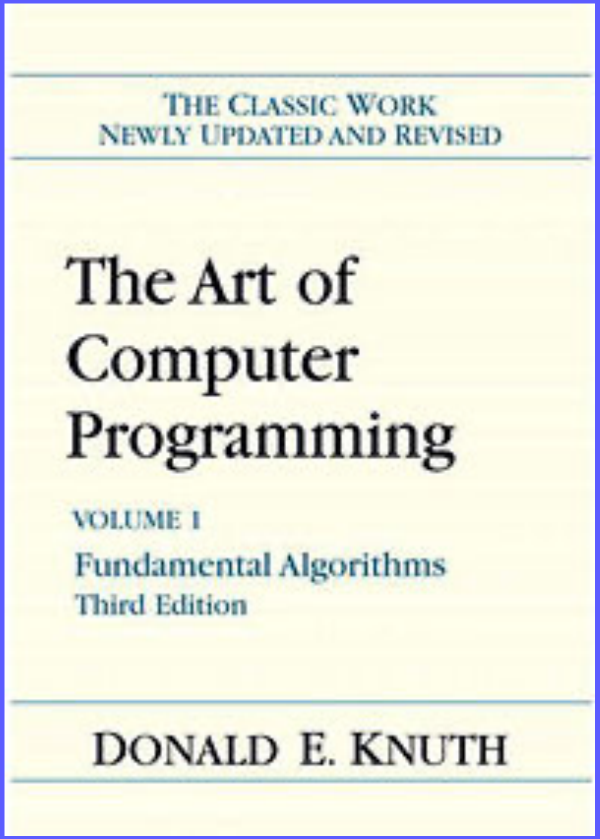
```
Algorithm Compute_Next(B, m);
Input: B (a string of size m)
Output: next
begin
    next(1):=-1;
    next(2):=0;
    for i:=3 to m do
        j:=next(i-1)+1;
        while $b_{i-1}$ <> $b_j$ and j >0 do
            j:=next(j)+1;
        next(i):=j
end
```

Donald E. Knuth 高德納

A person does not really understand something until after teaching it to a computer



The Art of Computer Programming



- among the best 12 physical-science monographs of the 20th century (*American Scientist)*
- If you think you're a really good programmer … read (Knuth's) Art of Computer Programming …

  You should definitely send me a resume if you can read the whole thing.“ (Bill Gates)

The Art of Computer Programming

- Completed
  - Volume 1 – Fundamental Algorithms (1968)
    - Chapter 1 – Basic concepts
    - Chapter 2 – Information structures
  - Volume 2 – Seminumerical Algorithms (1969)
    - Chapter 3 – Random numbers
    - Chapter 4 – Arithmetic
  - Volume 3 – Sorting and Searching (1973)
    - Chapter 5 – Sorting
    - Chapter 6 – Searching
  - Volume 4A – Combinatorial Algorithms (2001~2019)
    - Chapter 7 – Combinatorial searching (part 1)

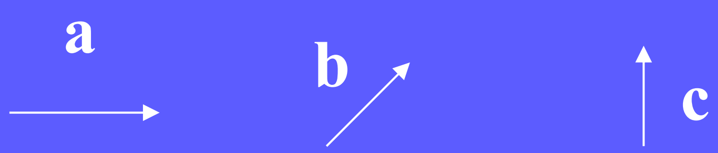
The Art of Computer Programming

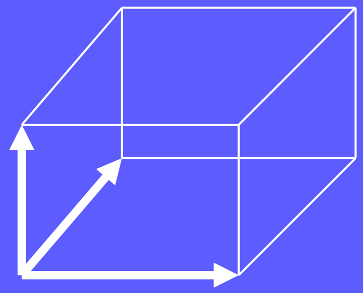
- Planned
  - Volume 4B... – Combinatorial Algorithms (chapters 7 & 8 released in several subvolumes)
    - Chapter 7 – Combinatorial searching (continued)
    - Chapter 8 – Recursion
  - Volume 5 – Syntactic Algorithms (as of 2017, estimated for release in 2025)
    - Chapter 9 – Lexical scanning (also includes string search and data compression)
    - Chapter 10 – Parsing techniques
  - Volume 6 – The Theory of Context-Free Languages
  - Volume 7 – Compiler Techniques

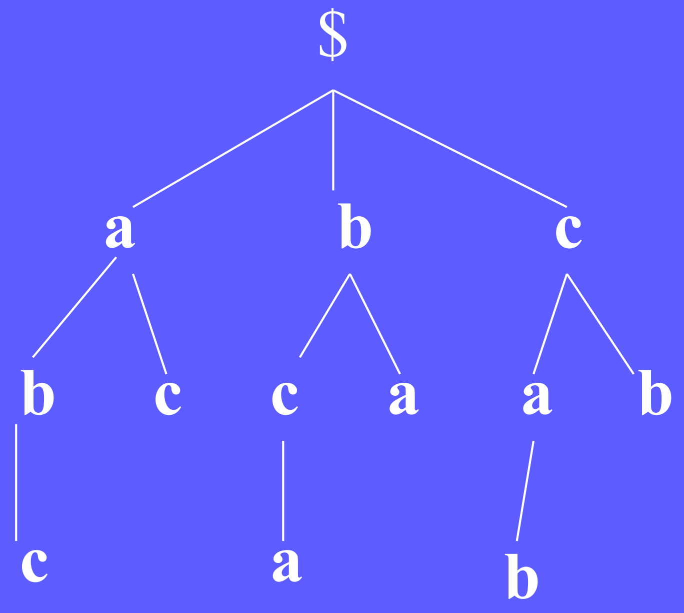
Pattern Matching

- Exact String matching
- Approximate string matching
- Single pattern matching
- Multiple patterns matching
- Two-dimensional pattern matching
- Tree matching
- Graph matching

Tree Matching



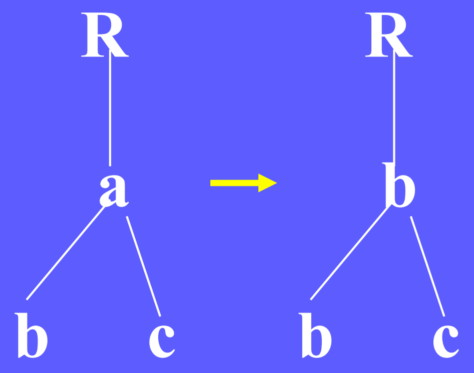




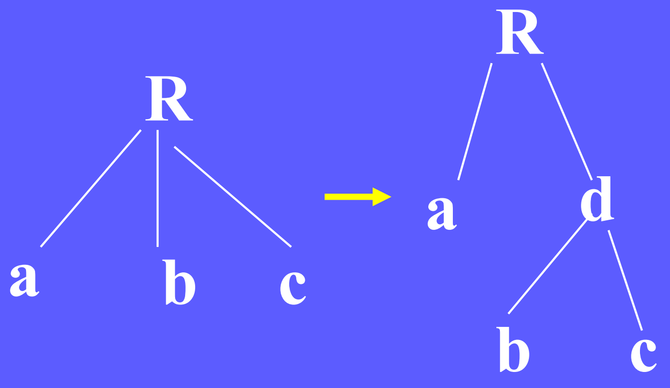
M. K. Shan, CS, NCCU

Tree Matching (cont.)

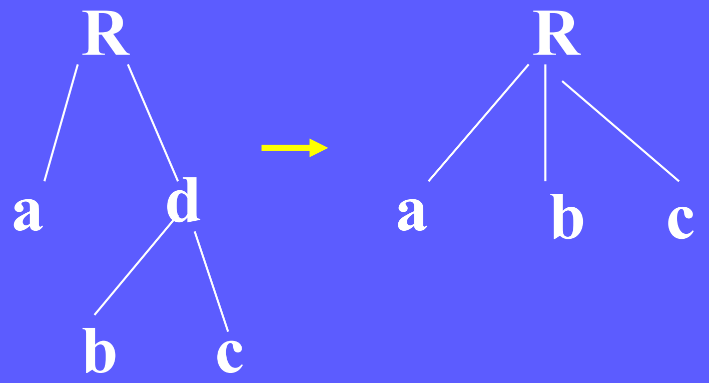
Replace:



Insert



Delete



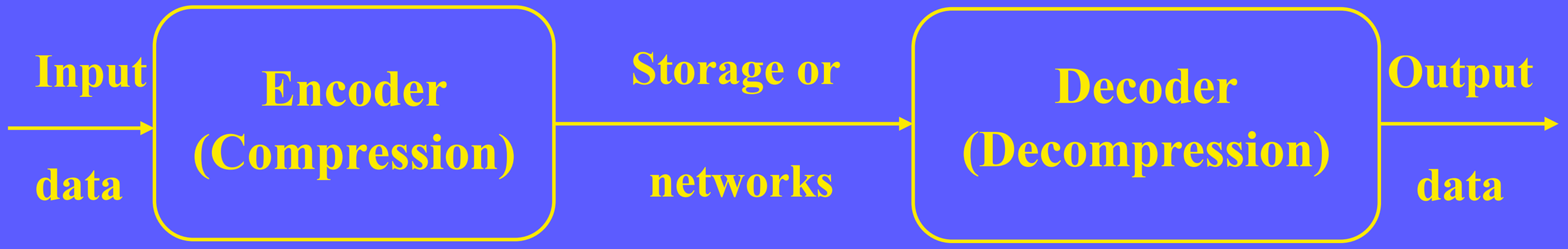
Huffman Coding

Fundamentals of Data Compression

- Why compressions
  - storage requirements
  - bandwidth requirements (storage, network)
- Objective of Compression
  - reduce the amount of data necessary to reproduce the original data
  - while maintaining
    - required levels of coded signal quality,
    - processing delay
    - implementation complexity

Fundamentals of Data Compression (cont.)

- General data compression scheme



Fundamentals of Data Compression (Cont.)

- ■Principle of Compression: remove redundancy
  - ☐Statistical redundancy
  - ☐Perceptional redundancy
- ■Considerations of compression techniques
  - ☐ Lossless or lossy
  - ☐ Entropy encoding or source encoding
    - Entropy: statistical properties of data
    - Source: content of source material, semantic and special characteristics
  - ☐ Symmetrical or asymmetrical (codec, encoder/decoder)
  - ☐ Software or hardware

Basic Compression Techniques

- Lossless
  - Run-length coding
  - Variable length coding: Huffman, Arithmetic, LZW
- Lossy
  - Predictive encoding: motion compensation, ADPCM
  - Frequency oriented coding
    - Transform coding: DCT, FFT+ quantization
    - Sub-band coding
  - Importance oriented coding
    - Quantization: scalar, vector quantization

Run Length Coding

- Basic idea
  - Encode continuous groups (run) of symbols as symbol and respective length

<e.g.> $\underline{a}$ $\underline{b}$ $\underline{aaaa}$ $\underline{bbbbbbb}$

a1 b1 a4 b7

Huffman Coding

- Huffman coding
  - Frequently occurring symbols with shorter codes

字元	次數	編碼
a	10	000
e	15	001
i	12	010
o	13	011
s	3	100
t	4	101
z	1	110

字元	次數	編碼
a	10	001
e	15	01
i	12	10
o	13	11
s	3	00000
t	4	0001
z	1	00001

(10+15+12+13+3+4+1)*3
=174 bits

10*3+(15+12+13)*2+(3+1)*5+4*4
= 146 bits

Requirement of Huffman Coding

字元	次數	編碼
a	10	01
e	12	00
i	15	1
s	4	0001
t	3	00011

Decode 00011 = ?

$\underline{00}\ \underline{01}\ \underline{1} = \text{eai}$

$\underline{0001}\ \underline{1} = \text{si}$

$\underline{00011} = \text{t}$

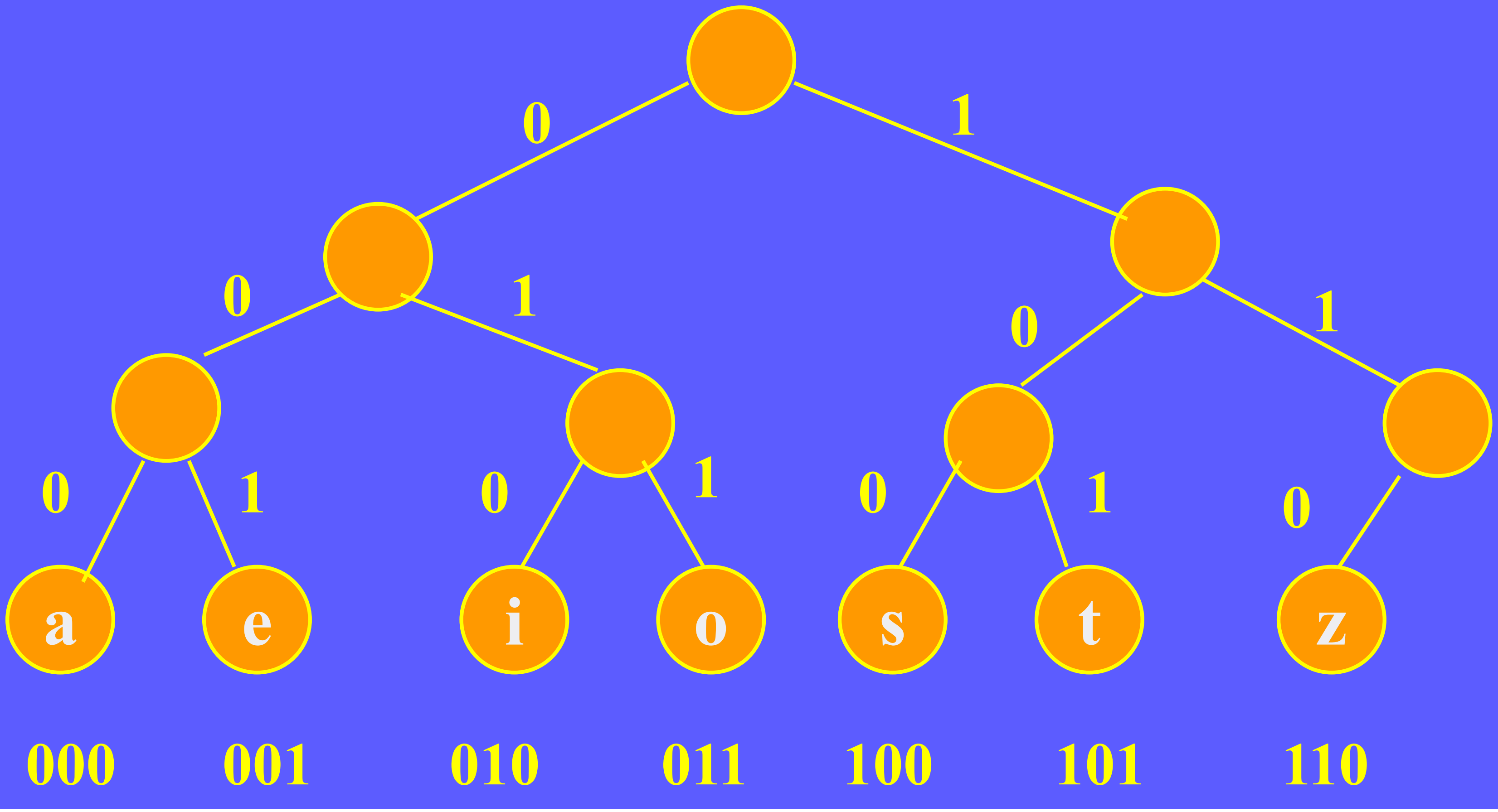
Requirement of Huffman Coding

- Unique prefix property
  - Precludes any ambiguity in decoding
  - No Huffman code is a prefix of any other Huffman code

字元	次數	編碼
a	10	01
e	12	00
i	15	1
s	4	0001
t	3	00011

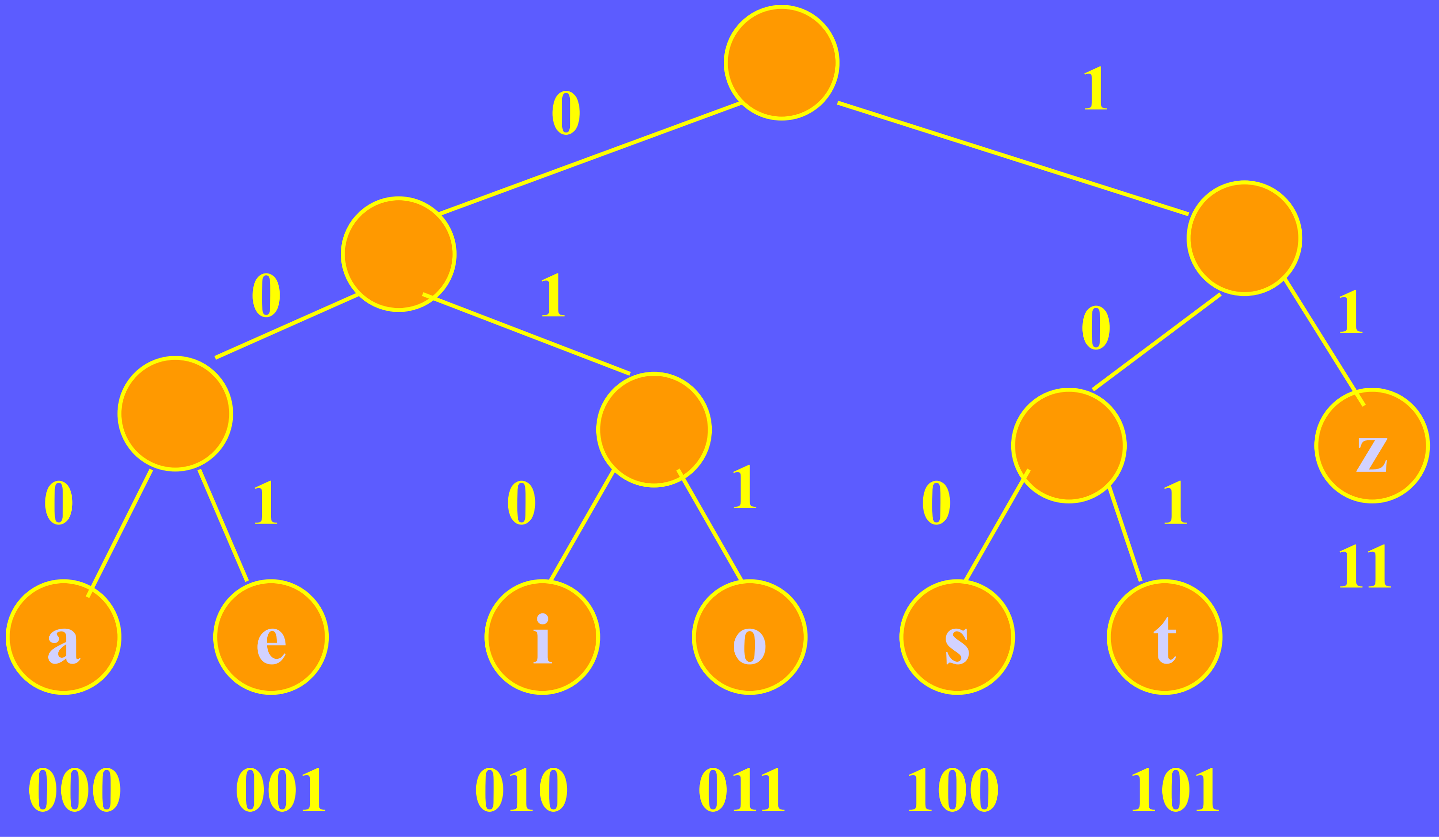
Huffman Coding (Cont.)

- Fixed length coding for a set of 7 symbols



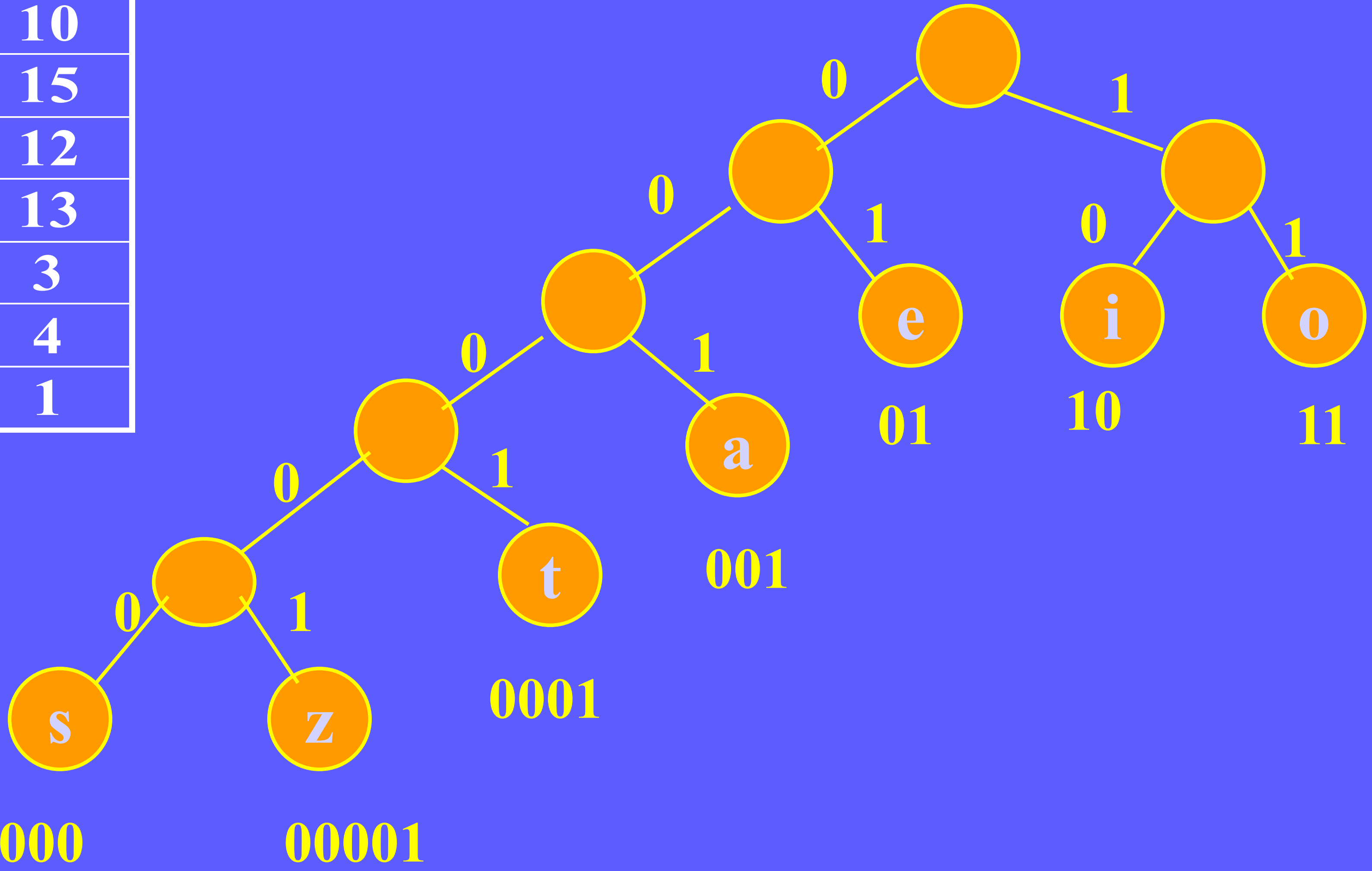
Huffman Coding (Cont.)

- Improvement

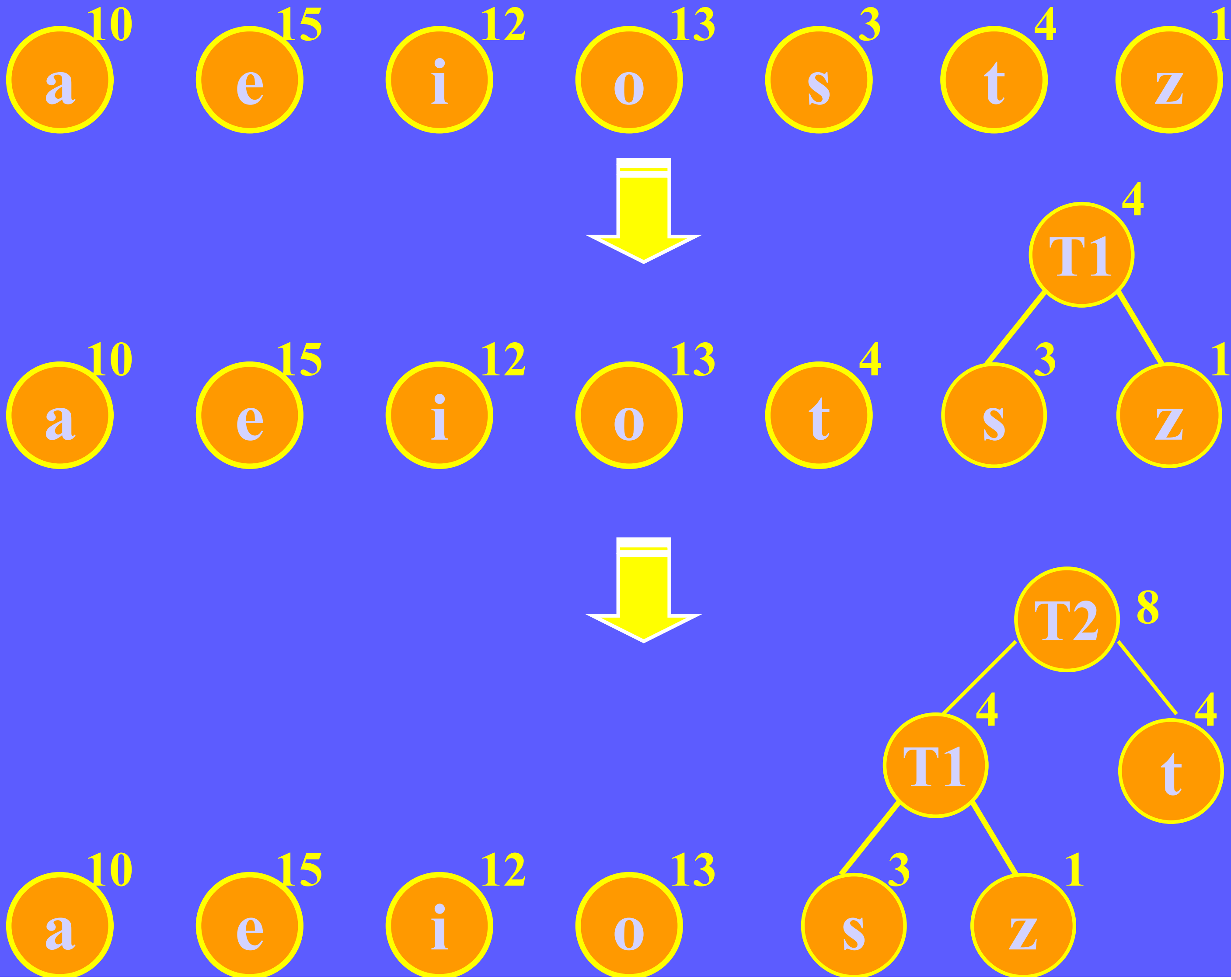


Basic Idea of Huffman Coding (cont.)

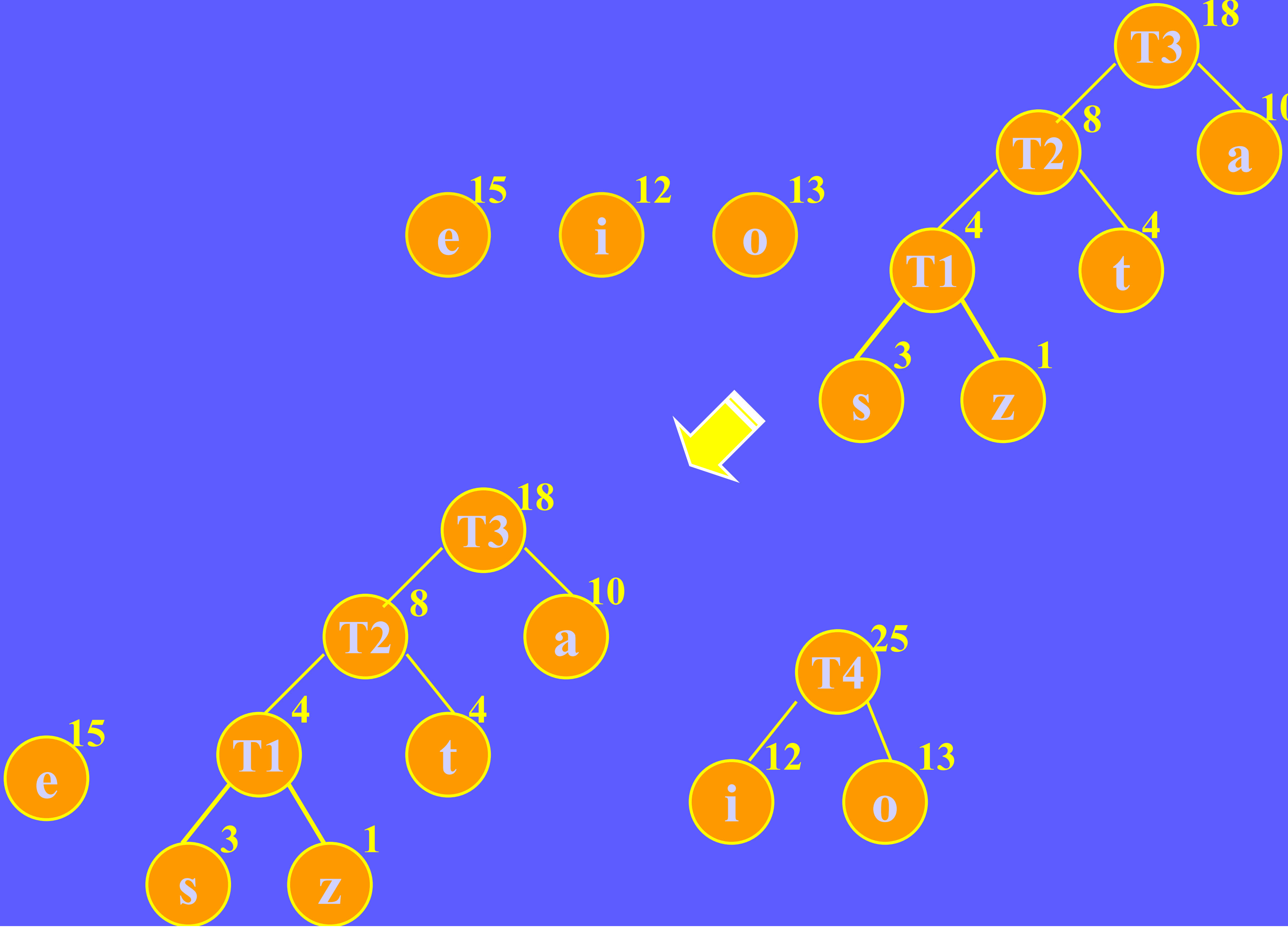
a	10
e	15
i	12
o	13
s	3
t	4
z	1



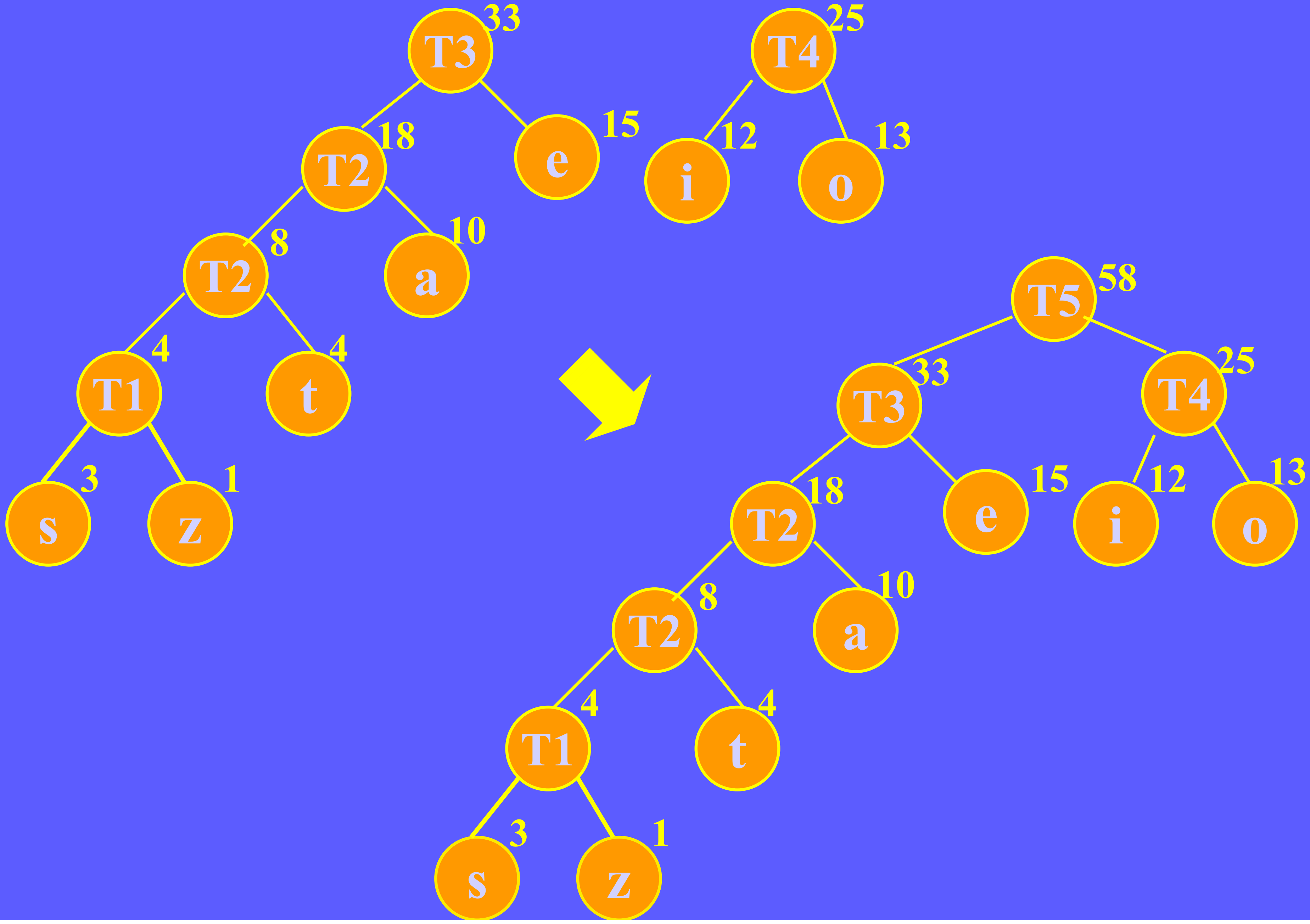
Example of Huffman Coding



Example of Huffman Coding (Cont.)



Example of Huffman Coding (Cont.)



Huffman Coding

- Proposed by David A. Huffman in 1952

$$\min \sum_i P(i)B(i)$$

- Bottom-Up Greedy Algorithm
  1. Sort all the symbols according to the frequency count
  2. Repeat
     a) From the sorted list, pick two symbols with the lowest frequency counts
     b) Form a Huffman subtree that has these two symbols as child nodes & create a parent node for them
     c) Assign the sum of the children's frequency to the parent & insert it into the list, such that the order is maintained
     d) Delete the children from the list

     Until the sorted list has only one symbol left
- Complexity: O(nlogn) using heap insertion & deletion

Basics of Information Theory

- Entropy
  - Proposed by Claude E. Shannon of Bell Labs.
  - Entropy of message M with alphabet $\{m_1, m_2, \ldots, m_n\}$

$$E(M) = -\sum_i p(m_i) \log_2 p(m_i)$$

where $p(m_i)$ is the probability that m_i in M will occur

<e.g.> Given p(male)=1/2, p(female)=1/2

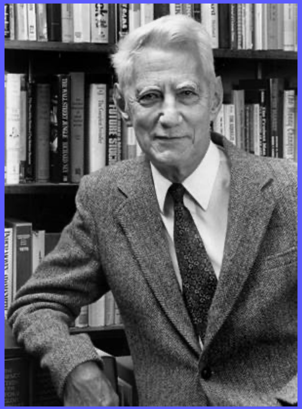
$$E(M) = -(\frac{1}{2} * (\log \frac{1}{2}) + \frac{1}{2} * (\log \frac{1}{2})) = \frac{1}{2} + \frac{1}{2} = 1$$

But if p(male)=2/3, p(female)=1/3 ?

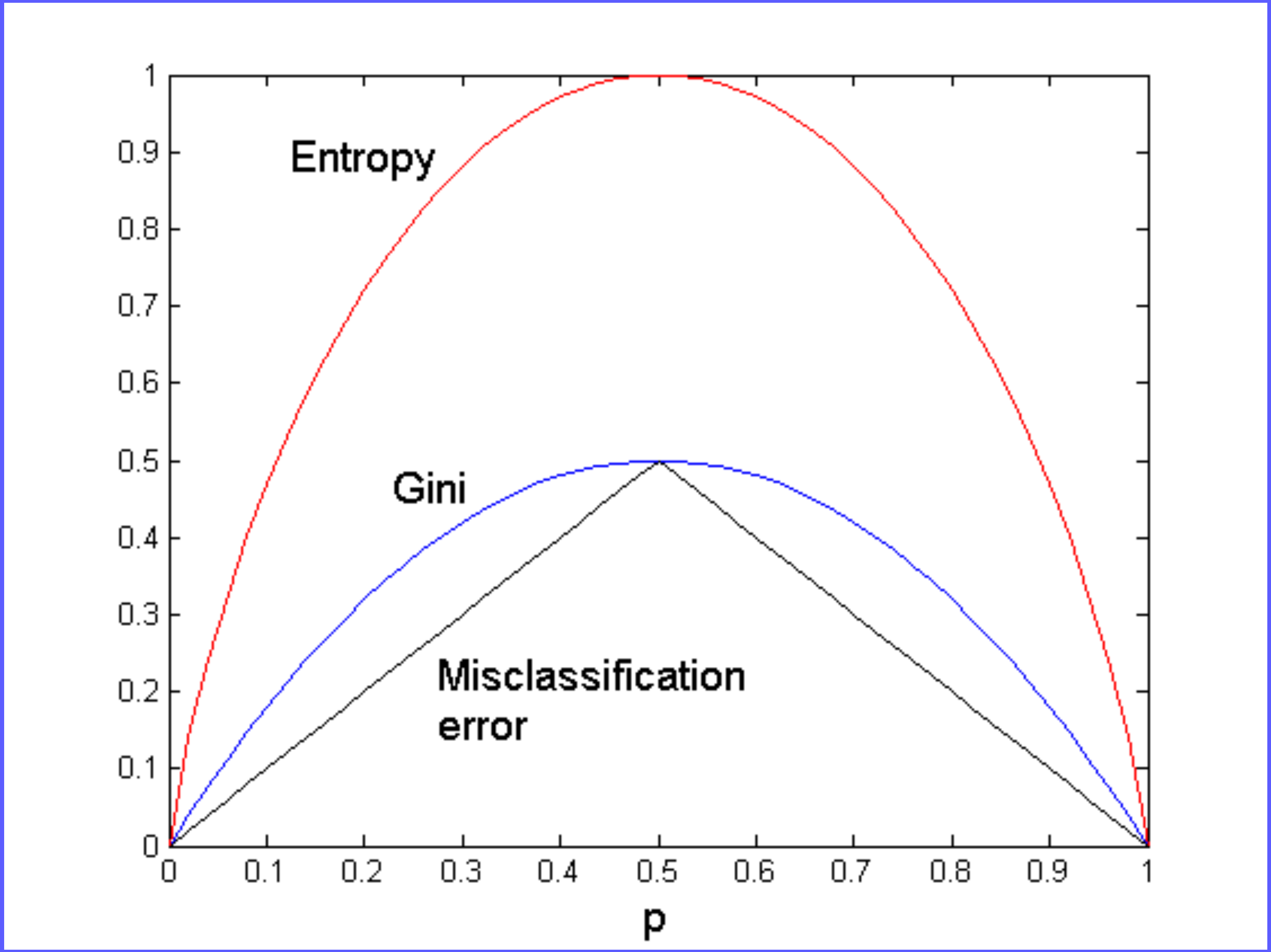
$$E(M) = -(\frac{2}{3} * (\log \frac{2}{3}) + \frac{1}{3} * (\log \frac{1}{3}))$$
$$= -(0.39 + 0.53) = 0.92$$

Claude Elwood Shannon (wiki)

- **Claude Elwood Shannon** (April 30, 1916 – February 24, 2001), an American engineer and mathematician, has been called "the father of information theory".
- Shannon is famous for having founded information theory and both digital computer and digital circuit design theory when he was 21 years-old by way of a master's thesis published in 1937, wherein he articulated that electrical application of Boolean algebra could construct and resolve any logical, numerical relationship. It has been claimed that this was the most important master's thesis of all time.
- Alfred Noble Prize, 1940



Basics of Information Theory (cont.)



Conclusions

- Finding maximum & minimum O(3/2n)
- Finding the k-th element O(n)
- String editing distance O(n x m)
- Longest common subsequence O(n x m)
- Sequence alignment O(n x m)
- Dynamic Time Warping O(n x m)
- String matching (KMP) O(n)
- Huffman coding O(nlogn)