Longest Common Subsequence (LCS)

張智星 (Roger Jang)

jang@mirlab.org

http://mirlab.org/jang

多媒體資訊檢索實驗室

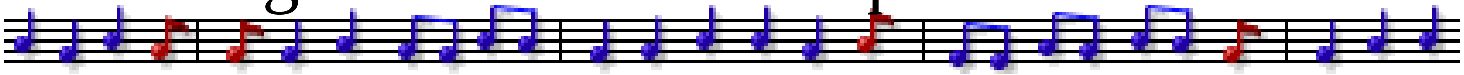
台灣大學 資訊工程系

Longest Common Subsequence

- Subsequence
  - Given a string, we can delete some elements to form a subsequence:
    - s1=uvwxyz ➔ s2=uwyz (after deleting v and x)
    - s2 is a subsequence of s1.
- Longest common subsequence (LCS)

For word suggestion!

  - The similarity of two string can be define as the length of the LCS between them.
  - Example: abcdefg and xzackdfwgh have acdfg as a longest common subsequence



Brute-Force Approach to LCS

- A brute-force solution
  - Enumerate all subsequences of X
  - Test which ones are also subsequences of Y
  - Pick the longest one.
- Analysis:
  - If X is of length n, then it has 2^n subsequences
  - This is an exponential-time algorithm!

DP for LCS: 3-step Formula

Quiz!

Three-step DP formula for computing $lcs\left(\vec{A},\vec{B}\right)$

1. Optimum-value function

$lcs\left(\underset{\vec{a}x}{\vec{p}},\underset{\vec{b}y}{\vec{q}}\right)$ is the length of LCS between string $\vec{p}$ and $\vec{q}$.

2. Recurrent formula

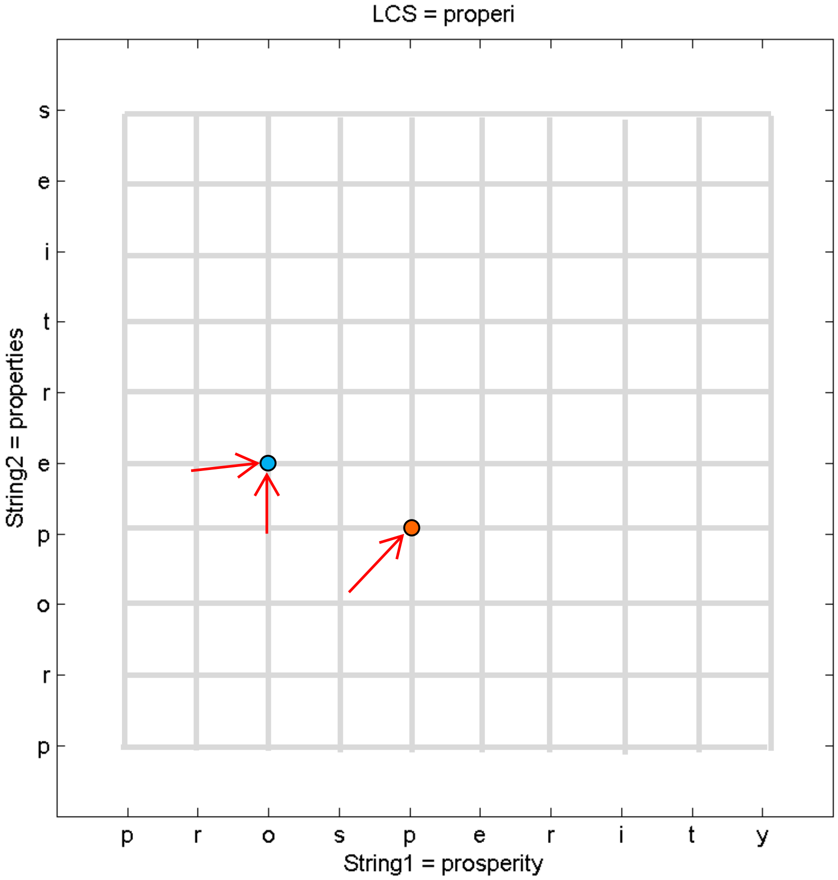
ex: $\underbrace{de}_{\vec{a}}\,\underset{x}{\underline{fg}}$

$$lcs\left(\underset{m}{\vec{a}x},\underset{n}{\vec{b}y}\right)=\begin{cases} lcs\left(\overset{m-1}{\vec{a}},\overset{n-1}{\vec{b}}\right)+1, \text{ if } x=y \\ \max\begin{cases} lcs\left(\overset{m}{\vec{a}x},\overset{n-1}{\vec{b}}\right) \\ lcs\left(\underset{m-1}{\vec{a}},\underset{n}{\vec{b}y}\right) \end{cases}, \text{ if } x\neq y \end{cases}$$

Boundary condition : $lcs\left(\vec{a},[]\right)=lcs\left([],\vec{b}\right)=0.$

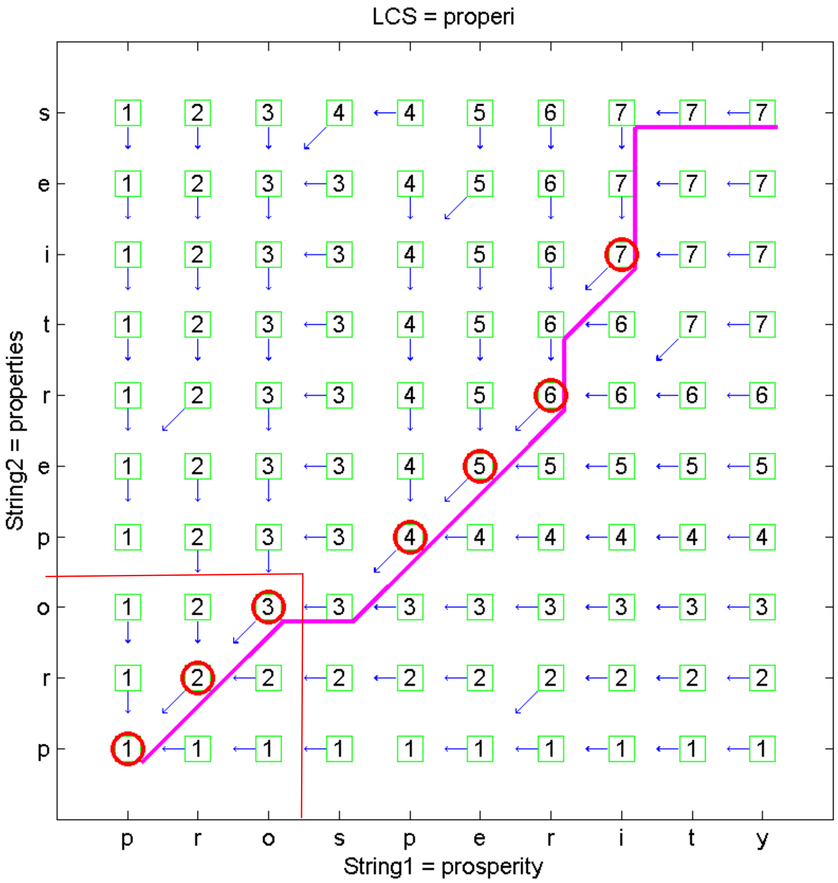
3. Answer : $lcs\left(\vec{A},\vec{B}\right)$

DP for LCS: Table Filling (1/2)



- ● $lcs(prosp, prop) = lcs(pros, prop) + 1$
- ● $lcs(pro, prope) = \max\begin{cases} lcs(pro, prop) \\ lcs(pr, prope) \end{cases}$

DP for LCS: Table Filling (2/2)

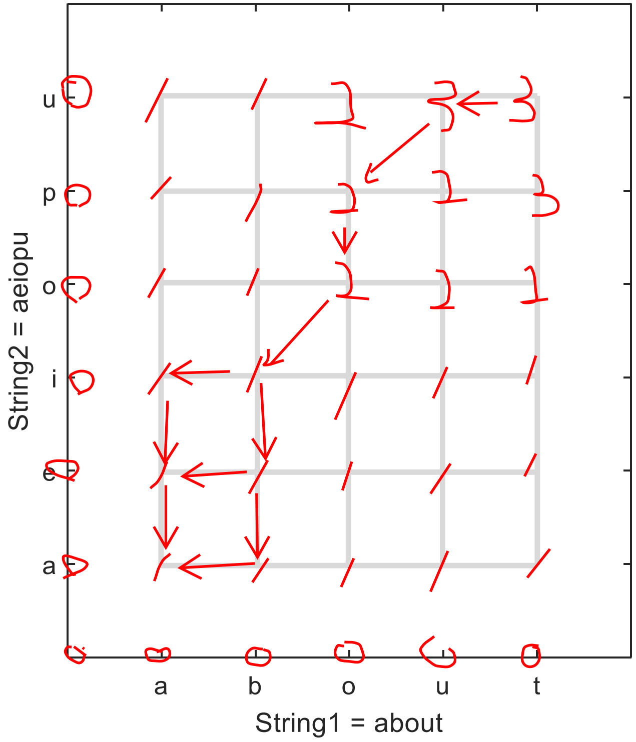


	p	r	o	s	p	e	r	i	t	y
s	1	2	3	4	4	5	6	7	7	7
e	1	2	3	3	4	5	6	7	7	7
i	1	2	3	3	4	5	6	7	7	7
t	1	2	3	3	4	5	6	6	7	7
r	1	2	3	3	4	5	6	6	6	6
e	1	2	3	3	4	5	5	5	5	5
p	1	2	3	3	4	4	4	4	4	4
o	1	2	3	3	3	3	3	3	3	3
r	1	2	2	2	2	2	2	2	2	2
p	1	1	1	1	1	1	1	1	1	1

- Observations
  - LCS=‘properi’ or ‘propert’ (which is obtained by keeping multiple back-tracking paths)
  - A match occurs when the node has a 45-degree back-tracking path

DP for LCS: Quiz for Table Filling

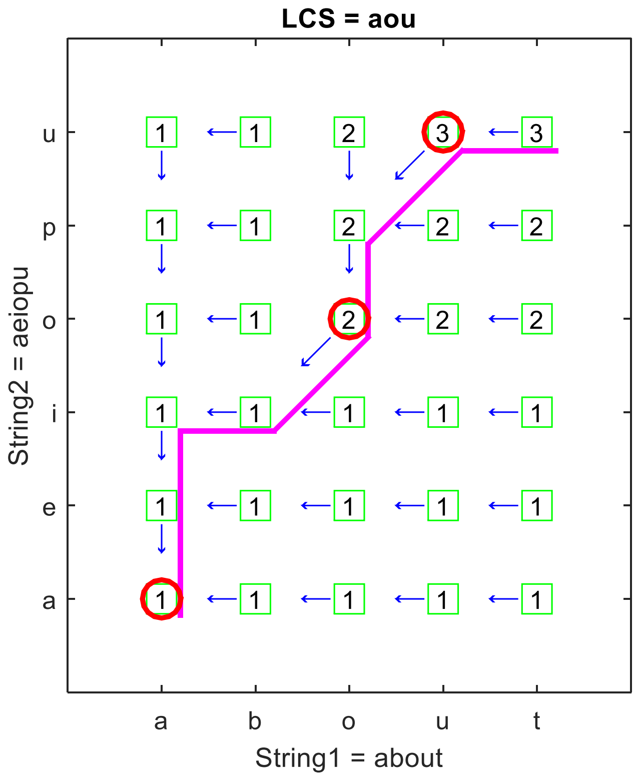
Quiz!



⌘ Hints

- Create a (m+1)*(n+1) matrix for table filling
- Fill row 0 and column 0 with 0 first to establish the base cases of boundary conditions
- Fill all the other elements in a layer-by-layer manner.

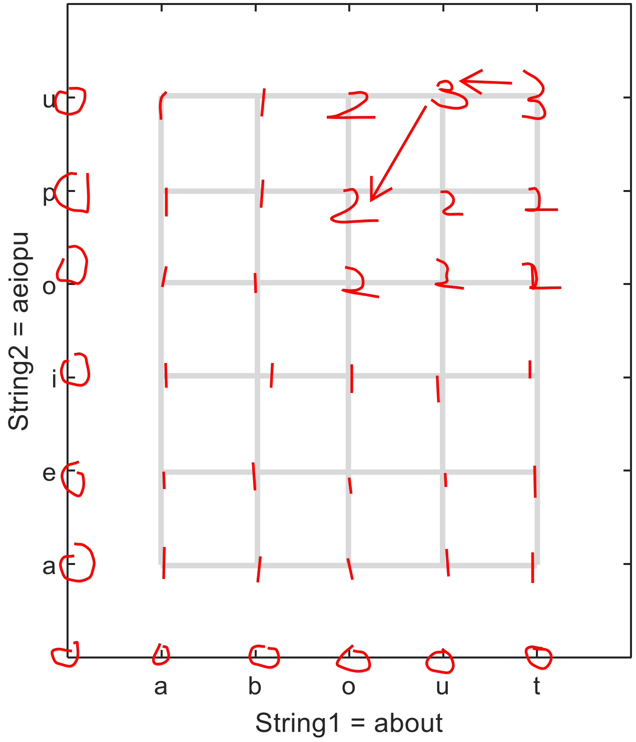
Quiz Solution



	a	b	o	u	t
u	1	1	2	3	3
p	1	1	2	2	2
o	1	1	2	2	2
i	1	1	1	1	1
e	1	1	1	1	1
a	1	1	1	1	1

- To create this plot
  - Download Machine Learning Toolbox
  - Run lcs('about', 'aeiopu', 1) under MATLAB

LCS In Terms of Path Finding



⌘ Note that all DP problem can be visualized as path finding…