15.4 Longest Common Subsequence (LCS) problem

- Given a sequence $X = \langle x_1, x_2, \ldots, x_m \rangle$, another sequence $Z = \langle z_1, z_2, \ldots, z_k \rangle$ is a subsequence of X
  if there exists a strictly increasing sequence $\langle i_1, i_2, \ldots, i_k \rangle$ of indices of X such that $x_{i_j} = z_j$ for all $j = 1, 2, \ldots, k$.
  For example: the sequence $\langle A_2, C_3, B_4, B_7, C_8, W_{11} \rangle$ is a subsequence of $\langle T, A, C, B, B, W, B, C, W, T, W \rangle$.

- Given two sequences X and Y, a sequence Z is called the common subsequence of X and Y if Z is a subsequence of both X and Y.

  For example: the sequence $\langle A, C, B, B, C, W \rangle$ is a common subsequence of $\langle T, A, C, B, B, W, B, C, W, T, W \rangle$ and $\langle A, A, B, C, B, W, B, C, A, A, W, T \rangle$.

15.4 Longest common subsequence problem (continued)

The longest-common-subsequence problem is to find a common subsequence $Z = LCS(X, Y)$ with the maximum length from both sequences X and Y.

Denote by

- $X_m = \langle x_1, x_2, \ldots, x_m \rangle$
- $Y_n = \langle y_1, y_2, \ldots, y_n \rangle$

Our task then to find an LCS(X_m, Y_n) between X_n and Y_n.

15.4 Longest common subsequence problem (continued)

The four steps to develop a DP algorithm for the LCN is as follows.

1. Characterize the structure of an optimal solution (the key observation)

- If $x_m = y_n$, then the LCS between X_n and Y_m includes x_m (= y_n).
  It thus consists of an LCS of $\langle x_1, x_2, \ldots, x_{m-1} \rangle$ and $\langle y_1, y_2, \ldots, y_{n-1} \rangle$, i.e.,
  $LCS(X_m, Y_n) = LCS(X_{m-1}, Y_{n-1}), x_m$.

- If $x_m \neq y_n$, then the LCS cannot include both x_m and y_n.
  So, at least one of them is not in the LCS of
  - (I) $X_{m-1} = \langle x_1, x_2, \ldots, x_{m-1} \rangle$ and $Y_n = \langle y_1, y_2, \ldots, y_n \rangle$, or
  - (II) $X_m = \langle x_1, x_2, \ldots, x_m \rangle$ and $Y_{n-1} = \langle y_1, y_2, \ldots, y_{n-1} \rangle$, i.e.,
    $LCS(X_m, Y_n)$ is either $LCS(X_{m-1}, Y_n)$ or $LCS(X_m, Y_{n-1})$.

15.4 Longest common subsequence problem (continued)

2. Recurrence of the optimal solution

Consider X_i and Y_j with $0 \leq i \leq n$ and $0 \leq j \leq m$. Let $c[i,j]$ be the length of an LCS of $X_i = \langle x_1, x_2, \ldots, x_i \rangle$ and $Y_j = \langle y_1, y_2, \ldots, y_j \rangle$.

Using the key observation,

$$c[i,j] = \begin{cases} 0, & \text{if } i = 0 \text{ or } j = 0 \\ c[i-1,j-1]+1, & \text{if } i,j > 0 \text{ and } x_i = y_j \\ \max\{c[i,j-1], c[i-1,j]\}, & \text{if } i,j > 0 \text{ and } x_i \neq y_j \end{cases}$$

Compute using increasing i, j until $c[m,n]$ is obtained. Row by row or column by column orders are ok.

As usual, keep tracking which option to provide the optimum at each step. This allows us to work backwards from the answer to find the actual subsequence with length $c[m,n]$. An array b can be used to record which case yielded the optimum at each step.

15.4 Longest common subsequence problem (continued)

3. Algorithm for calculating the defined recurrence

LCS_Length(X, Y)

```
1   for i ← 1 to m do c[i,0] ← 0 /* boundary condition */
2   for j ← 1 to n do c[0, j] ← 0 /* boundary condition */
3   for i ← 1 to m do
4   for     j ← 1 to n do
5           if x_i = y_j
6                   then   c[i, j] ← c[i−1, j−1] + 1;  b[i, j] ← " ↖ "
7                   else   if  c[i−1, j] ≥ c[i, j − 1]
8                                   then   c[i, j] ← c[i−1, j];  b[i, j] ← " ↑ "
9                                   else   c[i, j] ← c[i, j−1];  b[i, j] ← " ← "
```

15.4 Longest common subsequence problem (continued)

To find the LCS, it follows the path in b back from $b[m,n]$, where

- $\nwarrow$ implies that $x_i = y_j$
- $\uparrow$ implies the two strings are X_{i-1} and Y_j
- $\leftarrow$ implies the two strings are X_i and Y_{j-1}.

15.4 Longest common subsequence problem (continued)

For example: $X = \langle A,B,C,B,D,A,B \rangle$, $Y = \langle B,D,C,A,B,A \rangle$.

i		j	0	1	2	3	4	5	6
		y_j		B	D	C	A	B	A
0	x_i		0	0	0	0	0	0	0
1	A		0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B		0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C		0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B		0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D		0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A		0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B		0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

Issues:

(1) This is a special case of a longest path in acyclic digraph.

(2) How much space is required?

(3) What's running time of the algorithm?

The smart use of the LCS algorithm

- **Question 1:** Given a sequence of n positive integers, the problem is to find a longest increasing (or decreasing) subsequence of the sequence.

  Someone devised the following algorithm for this problem:

  - Let X be the original sequence;
  - Let Y be the sorted sequence of X in increasing order;
  - The longest increasing subsequence of X then is $LCS(X, Y)$.

  Is this algorithm correct?

  If yes, prove it, otherwise, disprove it by giving a counter-example.

The smart use of the LCS algorithm (cont.)

Answer: Let $x_1, x_2, \ldots, x_n$ be the sequence and Y the sorted sequence $y_1, y_2, \ldots, y_n$ of X in increasing order.

Let a subsequence $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ be the LCS of X and Y, where k is the length of the LCS. Following the definition of the LCS, the subsequence is also a subsequence of Y, then $x_{i_1} \leq x_{i_2} \leq \ldots \leq x_{i_k}$, which means that the subsequence is an increasing subsequence.

On the other hand, let k' be the length of a longest increasing subsequence of X that is also a common subsequence between X and Y with $k' > k$, this contracts with the initial assumption that $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ is a LCS of X and Y. Thus, the algorithm is correct.

The smart use of the LCS algorithm (cont.)

- **Question 2:** Let $X = \langle x_1, x_2, \ldots, x_n \rangle$, the question is to find the longest length palindrome of X, where a palindrome is a subsequence which is identical, no matter whether you read the sequence from its left side or from its right side.

  Someone devised the following algorithm for this problem:

  - Let X be the original sequence;
  - Let $X' = \langle x_n, x_{n-1}, \ldots, x_2, x_1 \rangle$ be *the reverse of* X;
  - The longest common subsequence $LCS(X, X')$ then is the longest length palindrome.

  Is this algorithm correct? If yes, prove it, otherwise, disprove it by giving a counter-example.

The smart use of the LCS algorithm (cont.)

Answer: Incorrect. For example, consider a sequence $X = A, L, F, A, L, F, A$, its reverse $X' = A, F, L, A, F, L, A$.

- One LCS between X and X' is A, L, A, F, A, which is not palindrome.
- Another LCS between X and X' is A, F, A, F, A, which is palindrome

An Exercise the LCS algorithm

Question: Given three sequences X, Y and Z with length n_X, n_Y and n_Z, respectively, find a longest common subsequence W among the three sequences, what is the running time of the proposed algorithm?