Longest Common Subsequence

Prepared by Suk Jin Lee

Biological Applications

- A strand of DNA consists of a string of molecules
  - Adnine (A), guanine (C), cytosine (G), and thymine (T)
  - For example
    - X = ACCGGTCGAGTGCGCGGAAGCCGGCCGAA
      Y = GTCGTTCGGAATGCCGTTGCTCTGTAAA
    - Similarity of strands X and Y
      Z = GTCGTCGGAAGCCGGCCGAA
  - Formalize this similarity
    - Longest common subsequence problem

Longest Common Subsequence

- 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 for all $j = 1, 2, \ldots, k$, we have $x_{i_j} = z_j$

  For example, $Z = \langle B, C, D, B \rangle$ is a subsequence of $X = \langle A, B, C, B, D, A, B \rangle$ with corresponding index sequence $\langle 2, 3, 5, 7 \rangle$

Longest Common Subsequence

- Given two sequences X and Y , we say that a sequence Z is a ***common subsequence*** of X and Y if Z is a subsequence of both X and Y.

  For example, if $X = \langle A, B, C, B, D, A, B \rangle$ and $Y = \langle B, D, C, A, B, A \rangle$, the sequence $\langle B, C, A \rangle$ is a common subsequence of both X and Y

  What is a *longest common subsequence* of both X and Y?

Solve the LCS problem

- Brute-force approach
  - Exhaustive search
  - Enumerate all subsequences of X and check each subsequence to see whether it is also a subsequence of Y
  - If $|X| = m$, $|Y| = n$, then there are 2^m subsequences of X
  - Must compare each with Y (n comparisons)
  - Running time: **$O(n\ 2^m)$**

LCS Algorithm

- Theorem. *Optimal substructure of an LCS*
  Let $X = \langle x_1, x_2, \ldots, x_m \rangle$ and $Y = \langle y_1, y_2, \ldots, y_n \rangle$ be sequences, and let $Z = \langle z_1, z_2, \ldots, z_k \rangle$ be any LCS of X and Y
  - If $x_m = y_n$, then $z_k = x_m = y_n$ and Z_{k-1} is an LCS of X_{m-1} and Y_{n-1}
  - If $x_m \neq y_n$, then $z_k \neq x_m$ implies that Z is an LCS of X_{m-1} and Y
  - If $x_m \neq y_n$, then $z_k \neq y_n$ implies that Z is an LCS of X and Y_{n-1}

LCS Algorithm

- Given a sequence $X = \langle x_1, x_2, \ldots, x_m \rangle$, define the i^{th} ***prefix*** of X, for $i = 0, 1, \ldots, m$, as $X_i = \langle x_1, x_2, \ldots, x_i \rangle$

  For example, if $X = \langle A, B, C, B, D, A, B \rangle$, then $X_4 = \langle A, B, C, B \rangle$ and X_0 is the empty sequence

- Define $c[i, j]$ as the length of LCS of X_i and Y_j
- Then the length of LCS of X and Y will be $c[m, n]$

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

LCS recursive solution

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

- We start with $i = j = 0$ (empty substrings of X and Y)
- Since X_0 and Y_0 are empty strings, their LCS is always empty (*i.e.* $c[0, 0] = 0$)
- LCS of empty string and any other string is empty, so for every i and j, $c[0, j] = c[i, 0] = 0$

LCS recursive solution

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

- **First case**: $x_i = y_i$
  - One more symbol in strings X and Y matches, so the length of LCS X_i and Y_j equals to the length of LCS of smaller strings X_{i-1} and Y_{i-1} , plus 1

LCS recursive solution

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

- **Second case**: $x_i \neq y_i$
  - As symbols don't match, our solution is not improved, and the length of LCS(X_i , Y_j) is the same as before (i.e. maximum of LCS(X_i, Y_{j-1}) and LCS(X_{i-1}, Y_j)

LCS Length Algorithm

LCS-LENGTH(X, Y)

1. $m = X.length$
2. $n = Y.length$
3. Let $b[1 \ldots m, 1 \ldots n]$ and $c[0 \ldots m, 0 \ldots n]$ be new tables
4. for $i = 1$ to m $c[i, 0] = 0$ // special case: Y_0
5. for $j = 1$ to n $c[0, j] = 0$ // special case: X_0
6. for $i = 1$ to m // for all x_i
7. for $j = 1$ to n // for all y_j
8. if ($x_i == y_j$) // two subsequences are equal?
9. $c[i, j] = c[i-1, j-1] + 1$
10. $b[i, j] =$ "↖" // points to the optimal subproblem solution chosen
11. elseif $c[i-1, j] \geq c[i, j-1]$
12. $c[i, j] = c[i-1, j]$ // $i > j$, LCS length determined by j
13. $b[i, j] =$ "↑"
14. else $c[i, j] = c[i, j-1]$ // $i < j$, LCS length determined by i
15. $b[i, j] =$ "←"
16. return c and b

Running time?

LCS Example

- LCS algorithm on the following example:
  - $X = \langle A, B, C, B, D, A, B \rangle$
  - $Y = \langle B, D, C, A, B, A \rangle$

LCS Example

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

$m = |X| = 7$, $n = |Y| = 6$

Allocate array $c[7,6]$

LCS Example

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

for $i = 1$ to m $\quad c[i, 0] = 0$
for $j = 1$ to n $\quad c[0, j] = 0$

LCS Example

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

if ($x_i == y_j$) $c[i, j] = c[i-1, j-1] + 1$; $b[i, j] =$ "↖"

elseif $c[i-1, j] \geq c[i, j-1]$ $c[i, j] = c[i-1, j]$; $b[i, j] =$ "↑"

else $c[i, j] = c[i, j-1]$; $b[i, j] =$ "←"

LCS Example

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

if ($x_i == y_j$)
elseif $c[i-1, j] \geq c[i, j-1]$
else

$c[i, j] = c[i-1, j-1] + 1;$
$c[i, j] = c[i-1, j];$
$c[i, j] = c[i, j-1];$

$b[i, j] =$ "↖"
$b[i, j] =$ "↑"
$b[i, j] =$ "←"

LCS Example

	j	0	**1**	2	3	4	5	6
i		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**		
2	***B***	0						
3	***C***	0						
4	***B***	0						
5	***D***	0						
6	***A***	0						
7	***B***	0						

if ($x_i == y_j$) — **$c[i, j] = c[i-1, j-1] + 1$;** — **$b[i, j] =$ "↖"**

elseif $c[i-1, j] \geq c[i, j-1]$ — $c[i, j] = c[i-1, j]$; — $b[i, j] =$ "↑"

else — $c[i, j] = c[i, j-1]$; — $b[i, j] =$ "←"

LCS Example

	j	0	**1**	2	3	4	5	6
i		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**	
2	***B***	0						
3	***C***	0						
4	***B***	0						
5	***D***	0						
6	***A***	0						
7	***B***	0						

if ($x_i == y_j$)
elseif $c[i-1, j] \geq c[i, j-1]$
else

$c[i, j] = c[i-1, j-1] + 1;$
$c[i, j] = c[i-1, j];$
$c[i, j] = c[i, j-1];$

$b[i, j] =$ " ↖ "
$b[i, j] =$ "↑"
$b[i, j] =$ "←"

LCS Example

	j	0	**1**	2	3	4	5	6
i		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						
3	***C***	0						
4	***B***	0						
5	***D***	0						
6	***A***	0						
7	***B***	0						

if ($x_i == y_j$) — **$c[i, j] = c[i-1, j-1] + 1$;** — **$b[i, j] =$ "↖"**

elseif $c[i-1, j] \geq c[i, j-1]$ — $c[i, j] = c[i-1, j]$; — $b[i, j] =$ "↑"

else — $c[i, j] = c[i, j-1]$; — $b[i, j] =$ "←"

LCS Example

	j	0	**1**	2	3	4	5	6
i		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**					
3	***C***	0						
4	***B***	0						
5	***D***	0						
6	***A***	0						
7	***B***	0						

if ($x_i == y_j$) — **$c[i, j] = c[i-1, j-1] + 1$;** — **$b[i, j] =$ "↖"**

elseif $c[i-1, j] \geq c[i, j-1]$ — $c[i, j] = c[i-1, j]$; — $b[i, j] =$ "↑"

else — $c[i, j] = c[i, j-1]$; — $b[i, j] =$ "←"

LCS Example

	j	0	**1**	2	3	4	5	6
i		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**		
3	***C***	0						
4	***B***	0						
5	***D***	0						
6	***A***	0						
7	***B***	0						

if ($x_i == y_j$)
elseif $c[i-1, j] \geq c[i, j-1]$
else

$c[i, j] = c[i-1, j-1] + 1;$
$c[i, j] = c[i-1, j];$
$c[i, j] = c[i, j-1];$

$b[i, j] =$ " ↖ "
$b[i, j] =$ "↑"
$b[i, j] =$ "←"

LCS Example

	j	0	**1**	2	3	4	5	6
i		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						
4	***B***	0						
5	***D***	0						
6	***A***	0						
7	***B***	0						

if ($x_i == y_j$) — **$c[i, j] = c[i-1, j-1] + 1$;** — **$b[i, j] =$ "↖"**

elseif $c[i-1, j] \geq c[i, j-1]$ — $c[i, j] = c[i-1, j]$; — $b[i, j] =$ "↑"

else — $c[i, j] = c[i, j-1]$; — $b[i, j] =$ "←"

LCS Example

	j	0	**1**	2	3	4	5	6
i		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**			
4	***B***	0						
5	***D***	0						
6	***A***	0						
7	***B***	0						

if ($x_i == y_j$) — **$c[i, j] = c[i-1, j-1] + 1$;** — **$b[i, j] =$ "↖"**

elseif $c[i-1, j] \geq c[i, j-1]$ — $c[i, j] = c[i-1, j]$; — $b[i, j] =$ "↑"

else — $c[i, j] = c[i, j-1]$; — $b[i, j] =$ "←"

LCS Example

	j	0	**1**	2	3	4	5	6
i		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						
5	***D***	0						
6	***A***	0						
7	***B***	0						

if ($x_i == y_j$)
elseif $c[i-1, j] \geq c[i, j-1]$
else

$c[i, j] = c[i-1, j-1] + 1;$
$c[i, j] = c[i-1, j];$
$c[i, j] = c[i, j-1];$

$b[i, j] =$ "↖"
$b[i, j] =$ "↑"
$b[i, j] =$ "←"

LCS Example

	j	0	**1**	2	3	4	5	6
i		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						
6	***A***	0						
7	***B***	0						

if ($x_i == y_j$)
elseif $c[i-1, j] \geq c[i, j-1]$
else

$c[i, j] = c[i-1, j-1] + 1$;
$c[i, j] = c[i-1, j]$;
$c[i, j] = c[i, j-1]$;

$b[i, j]$ = "↖"
$b[i, j]$ = "↑"
$b[i, j]$ = "←"

LCS Example

	j	0	**1**	2	3	4	5	6
i		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						
7	***B***	0						

if ($x_i == y_j$) — **$c[i, j] = c[i-1, j-1] + 1$;** — **$b[i, j] =$ "↖"**

elseif $c[i-1, j] \geq c[i, j-1]$ — $c[i, j] = c[i-1, j]$; — $b[i, j] =$ "↑"

else — $c[i, j] = c[i, j-1]$; — $b[i, j] =$ "←"

LCS Example

	j	0	**1**	2	3	4	5	6
i		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						

if ($x_i == y_j$) — **$c[i, j] = c[i-1, j-1] + 1$;** — **$b[i, j] =$ "↖"**

elseif $c[i-1, j] \geq c[i, j-1]$ — $c[i, j] = c[i-1, j]$; — $b[i, j] =$ "↑"

else — $c[i, j] = c[i, j-1]$; — $b[i, j] =$ "←"

LCS Example

	j	0	**1**	2	3	4	5	6
i		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**

if ($x_i == y_j$) — **$c[i, j] = c[i-1, j-1] + 1$;** — **$b[i, j] =$ "↖"**

elseif $c[i-1, j] \geq c[i, j-1]$ — $c[i, j] = c[i-1, j]$; — $b[i, j] =$ "↑"

else — $c[i, j] = c[i, j-1]$; — $b[i, j] =$ "←"

LCS Algorithm Running Time

- LCS algorithm calculates the values of each entry of the array $c[m, n]$
- So what is the running time?

$O(m \times n)$

since each $c[i, j]$ is calculated in constant time, and there are $m \times n$ elements in the array

Constructing an LCS

- Simply begin at $b[m, n]$ and trace through the table by following the arrows
- If encountering a "↖" in entry $b[i, j]$, it implies that $x_i = y_i$ is an element of the LCS

```
PRINT-LCS(b, X, i, j)
1.  if i == 0 or j == 0
2.      return
3.  If b[i , j] == "↖"
4.      PRINT-LCS(b, X, i – 1, j – 1)
5.      print x_i
6.  elseif b[i , j] == " ↑ "
7.      PRINT-LCS(b, X, i – 1, j)        // move up
8.  else PRINT-LCS(b, X, i, j – 1)       // move left
```

Constructing an LCS

	j	0	**1**	2	3	4	5	6
i		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**

Constructing an LCS

	j	0	**1**	2	3	4	5	6
i		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

LCS reversed order: *A*

Constructing an LCS

	j	0	**1**	2	3	4	5	6
i		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

LCS reversed order: *A B*

Constructing an LCS

	j	0	**1**	2	3	4	5	6
i		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

LCS reversed order: *A B C*

Constructing an LCS

	j	0	**1**	2	3	4	5	6
i		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

LCS reversed order: *A B C B*

LCS straight order: *B C B A*

Running Time

- From the *b* table, it decrements at least one of *i* and *j* in each recursive call

$$\Rightarrow O(m + n)$$

Demonstration

- What do `spanking` and `amputation` have in common? [show only $c[i, j]$]

Demonstration

- What do `spanking` and `amputation` have in common? [show only $c[i, j]$]

	j	0	**1**	2	3	4	5	6	7	8	9	10
i		y_j	**a**	**m**	**p**	**u**	**t**	**a**	**t**	**i**	**o**	**n**
0	x_i	0	0	0	0	0	0	0	0	0	0	0
1	**s**	0	0	0	0	0	0	0	0	0	0	0
2	**p**	0	0	0	1	1	1	1	1	1	1	1
3	**a**	0	1	1	1	1	1	2	2	2	2	2
4	**n**	0	1	1	1	1	1	2	2	2	2	3
5	**k**	0	1	1	1	1	1	2	2	2	2	3
6	**i**	0	1	1	1	1	1	2	2	3	3	3
7	**n**	0	1	1	1	1	1	2	2	3	3	4
8	**g**	0	1	1	1	1	1	2	2	3	3	4