Dynamic Programming

Longest Common Subsequence

Dr. G P Gupta

Longest Common Subsequence

What is Subsequences?

- Suppose you have a sequence $X = \langle x_1, x_2, ..., x_m \rangle$ of elements over a finite set S.
- A sequence Z = < z₁, z₂,...,z_k> over S is called a subsequence of X if and only if it can be obtained from X by deleting elements.
- Put differently, there exist indices $i_1{<}i_2{<}...{<}i_k$ such that

$$z_a = x_{i_a}$$

for all a in the range $1 \le a \le k$.

What is Subsequences? Cont..

A subsequence of a string S, is a set of characters that appear in left-to-right order, but not necessarily consecutively.

Example

ACTTGCG

- \bullet ACT , $\ ATTC$, $\ T$, $\ ACTTGC$ are all subsequences.
- \bullet TTA is not a subequence

What is Common Subsequences?

- Suppose that X and Y are two sequences over a set S.
- We say that \boldsymbol{Z} is a common subsequence of X and Y if and only if
- Z is a subsequence of X
- Z is a subsequence of Y

What is Longest common subsequence?

- · Subsequence:
 - A subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements.
- · Longest common subsequence:
 - Longest common subsequence (LCS) of 2 sequences is a subsequence, with maximal length, which is common to both the sequences.

What is Longest common subsequence?

A common subsequence of two strings is a subsequence that appears in both strings. A longest common subsequence is a common subsequence of maximal length.

Example

$$\begin{split} S_1 &= AAACCGTGAGTTATTCGTTCTAGAA \\ S_2 &= CACCCCTAAGGTACCTTTGGTTC \end{split}$$

The Longest Common Subsequence Problem

Given two sequences X and Y over a set S, the longest common subsequence problem asks to find a common subsequence of X and Y that is of maximal length.

Longest Common Subsequence

- Biologists need to *measure how similar strands of DNA are* to determine how closely related an organism is to another.
- considering DNA as strings of letters A,C,G,T and then comparing similarities in the strings.
- Formally , researchers look at common subsequences in the strings.
- Example: X = AGTCAACGTT, Y=GTTCGACTGTG
- Both **S** = AGTG and **S'**=GTCACGT are subsequences
- · How to do find these efficiently?

What is Longest common subsequence?

 $S_1 = AAACCGTGAGTTATTCGTTCTAGAA$ $S_2 = CACCCCTAAGGTACCTTTGGTTC$

LCS is

ACCTAGTACTTTG

Has applications in many areas including biology.

Brute Force solution

- if |X| = m, |Y| = n, then there are 2^m subsequences of x; we must compare each with Y (n comparisons)
- So the running time of the brute-force algorithm is $O(n \ 2^{\text{m}})$
- Notice that the LCS problem has optimal substructure:
 - solutions of subproblems are parts of the final solution.
- Subproblems: "find LCS of pairs of prefixes of X and Y"

Dynamic Programming

Let us try to develop a dynamic programming solution to the LCS problem.

ith prefix of X

- Let $X = \langle x_1, x_2, ..., x_m \rangle$ be a sequence.
- ith prefix of X:

We denote by X_i the sequence $X_i = \langle x_1, x_2, ..., x_i \rangle$ and call it the **i**th **prefix of X**.

- · For example:
- if X = <A; B; C; B; D; A; B>, then 4th prefix: X4 = <A; B; C; B> and
- X0 is the empty sequence.

LCS Notation

Let X and Y be sequences.

LCS(X, Y) represent:

- the set of longest common subsequences of X and Y.

Optimal Substructure

- Let $X = < x_1, x_2, ..., x_m >$ and $Y = < y_1, y_2, ..., y_n >$ be two sequences.
- Let $Z = \langle z_1, z_2, ..., z_k \rangle$ is any LCS of X and Y.
- a) Case1: If $x_m = y_n$ then certainly $x_m = y_n = z_k$ and Z_{k-1} is $in LCS(X_{m-1}, Y_{n-1})$

Optimal Substructure cont..

Let $X = \langle x_1, x_2, ..., x_m \rangle$ and $Y = \langle y_1, y_2, ..., y_n \rangle$ be two sequences. Let $Z = \langle z_1, z_2, ..., z_k \rangle$ is any LCS of X and Y.

- b) Case2: If $x_m \neq y_n$ then $x_m \neq z_k$ implies that **Z** is in LCS(X_{m-1} , Y)
- c) Case3: If $x_m \neq y_n$ then $y_n \neq z_k$ implies that **Z** is in LCS(X, Y_{n-1})

Overlapping Subproblems

- If $x_m = y_n$ then we solve the subproblem to find an element in LCS(X_{m-1} , Y_{n-1}) and append x_m
- If $x_m \neq y_n$ then we solve the two subproblems of finding elements in $LCS(X_{m-1}\,,\,Y_{n-1}\,)$ and $LCS(X_{m-1}\,,\,Y_{n-1}\,)$ and choose the longer one.

Recursive Solution Let X and Y be sequences. Let $\mathbf{c}[\mathbf{i},\mathbf{j}]$ be the length of an LCS of the sequences Xi and Yj $\mathbf{c}[\mathbf{i},\mathbf{j}] = \begin{bmatrix} 0 & \bullet & \text{if } \mathbf{i} = \mathbf{0} & \text{or } \mathbf{j} = \mathbf{0} \\ & \bullet & \text{if } \mathbf{i} = \mathbf{0} & \text{or } \mathbf{j} = \mathbf{0} \\ & & \mathbf{c}[\mathbf{i},\mathbf{j}] = \mathbf{0} & \bullet & \text{if } \mathbf{i},\mathbf{j} > \mathbf{0} \text{ and } \mathbf{x}_i = \mathbf{y}_j \\ & & & \mathbf{max}(\mathbf{c}[\mathbf{i},\mathbf{j}-1],\mathbf{c}[\mathbf{i}-1,\mathbf{j}]) & \bullet & \text{if } \mathbf{i},\mathbf{j} > \mathbf{0} \text{ and } \mathbf{x}_i \neq \mathbf{y}_j \end{bmatrix}$

optimal substructure of the LCS problem

The optimal substructure of the LCS problem gives the recursive formula

$$c[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0, \\ c[i-1,j-1]+1 & \text{if } i,j > 0 \text{ and } x_i = y_j, \\ \max(c[i,j-1],c[i-1,j]) & \text{if } i,j > 0 \text{ and } x_i \neq y_j. \end{cases}$$
 (15.9)

Dynamic Programming Solution

•To compute length of an element in LCS(X,Y) with X of length m and Y of length n,

we do the following:

- Initialize first row and first column of the array c with 0.
- Calculate: c[1,j] for $1 \le j \le n$, $c[2,j] \text{ for } 1 \le j \le n$
- Return c[m, n]
- Complexity O(mn).

Dynamic Programming Solution cont..

- How can we get an actual longest common subsequence?
- Store in addition to the array c an array b pointing to the optimal subproblem chosen when computing c[i,j].

Example

LCS Example-1

- Consider the two sequences
- $X = \langle A, B, C, B, A \rangle$
- $Y = \langle B, D, C, A, B \rangle$

LCS Example-1

$$x_{i} = y_{j} \Rightarrow c[i,j] = c[i-1,j-1]+1$$

$$x_{i} \neq y_{j} \Rightarrow c[i-1,j] \geqslant c[i,j-1]$$

$$c[i,j] = c[i-1,j] \qquad \uparrow$$

$$c[i-1,j] < c[i,j-1]$$

$$c[i,j] = c[i,j-1] \qquad \bullet$$

LCS Example-1

	j	→	1	2	3	4	5
i		Уj	В	D	C	A	В
ļ	x_{i}	0	0	0	0	0	0
1	A	0	0 ↑	0 ↑	0 ↑	1 ^	1 ←
2	В	0	1 5	1 ←	1 ←	1 ↑	2 5
3	C	0	1 ↑	1 ↑	2 5	2 ←	2 ↑
4	В	0	1 \	1 ↑	2 ↑	2 ↑	3 5
5	A	0	1 ↑	1 ↑	2 ↑	3 <	3 ↑

Thus the optimal LCS length is c[m,n] = 3.

LCS Algorithm

- Computing the length of an LCS LCS-LENGTH (X, Y)
 - stores the c[i,j] values in a table c[0...m,0...n], and it computes the entries in row-major order.
- · Constructing an LCS
 - PRINT-LCS(b, X, i, j)
 - maintains the table b[1...m; 1...n] to help us construct an optimal solution

b[i,j] points to the table entry corresponding to the optimal subproblem solution chosen when computing c[i,j].

```
LCS-LENGTH(X, Y)

1  m = X.length

2  n = Y.length

3  let b[1...m, 1...n] and c[0...m, 0...n] be new tables

4  for i = 1 to m

5  c[i, 0] = 0

6  for j = 0 to n

7  c[0, j] = 0

8  for i = 1 to m

9  for j = 1 to n

10  if x_i = y_j

11  c[i, j] = c[i - 1, j - 1] + 1

12  b[i, j] = x_j

13  elseif c[i - 1, j] \ge c[i, j - 1]

14  c[i, j] = c[i - 1, j]

15  b[i, j] = x_j

16  else c[i, j] = c[i, j - 1]

17  b[i, j] = x_j

18  return c and b
```

PRINT-LCS

```
PRINT-LCS((b, X, i, j))

1 if i = 0 or j = 0

2 return

3 if b[i, j] = \text{```}

4 PRINT-LCS((b, X, i - 1, j - 1))

5 print x_i

6 elseif b[i, j] = \text{``}

7 PRINT-LCS((b, X, i - 1, j))

8 else PRINT-LCS((b, X, i, j - 1))
```

Analysis

- since each table entry takes O(1) time to compute.
- The running time of the procedure is Θ (mn).

LCS Example

We'll see how LCS algorithm works on the following example:

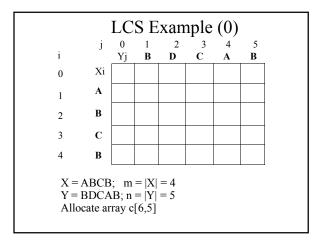
- X = ABCB
- Y = BDCAB

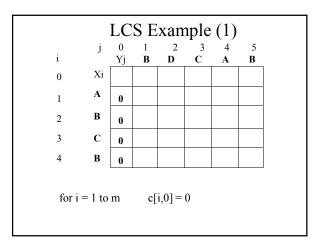
What is the Longest Common Subsequence of X and Y?

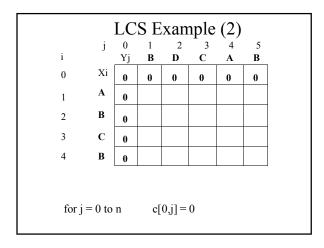
$$LCS(X, Y) = BCB$$

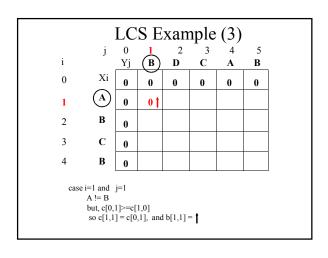
$$X = A B C B$$

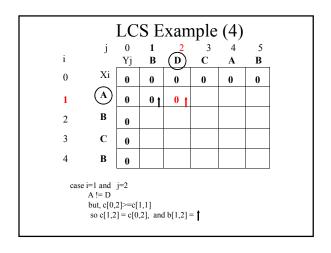
$$Y = B D C A B$$

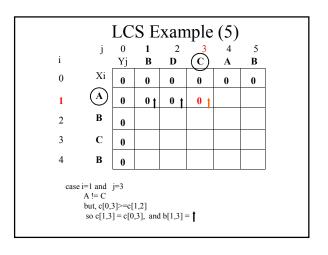


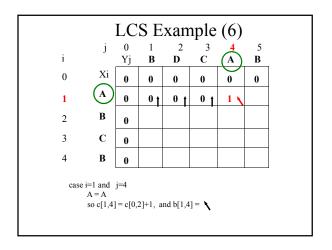


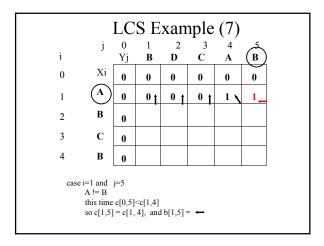


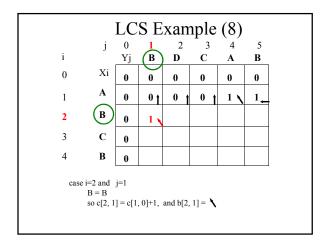


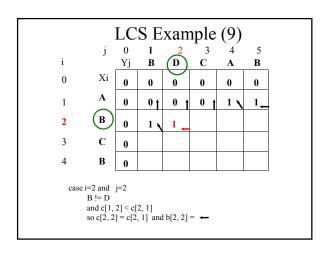


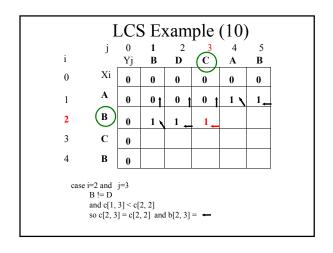


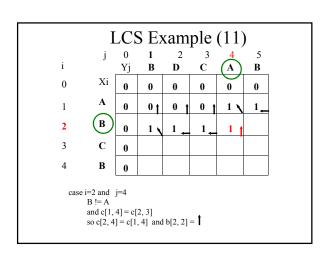


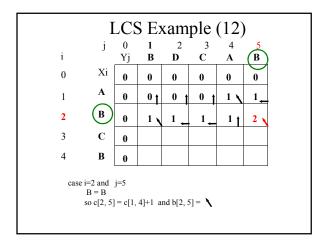


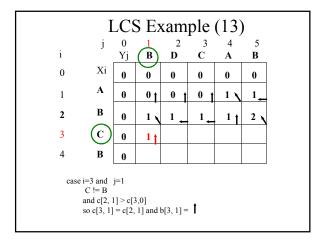


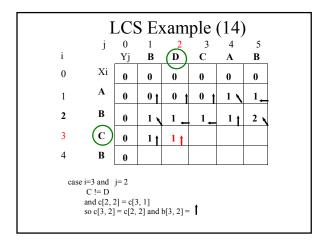


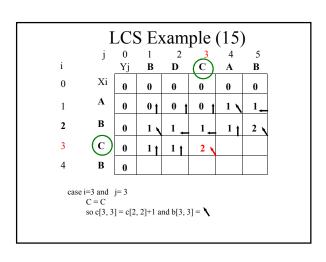


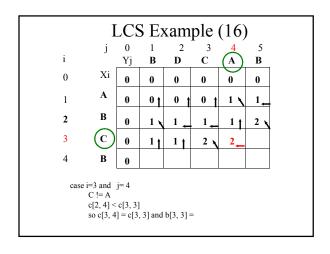


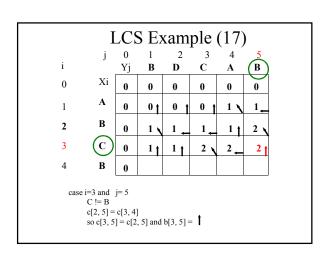


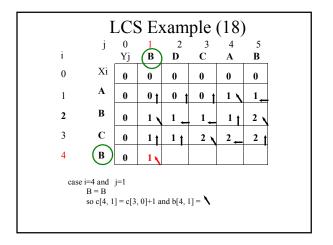


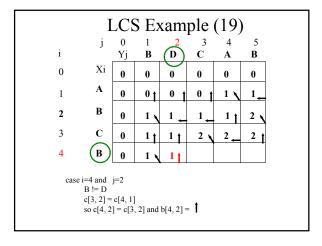


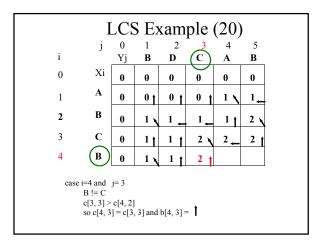


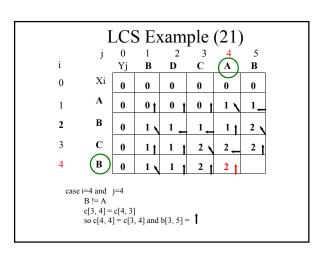


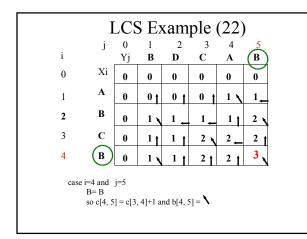






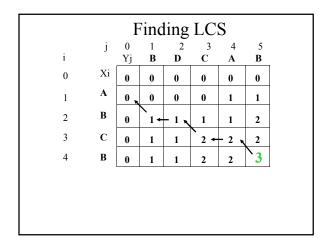


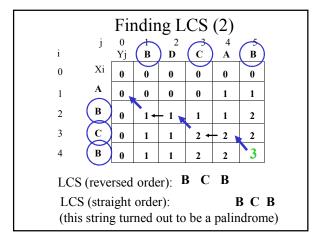




LCS Algorithm Running Time

- LCS algorithm calculates the values of each entry of the array c[m,n]
- So the running time is clearly O(mn) as each entry is done in 3 steps.
- Now how to get at the solution?
- · We use the arrows we created to guide us.
- We simply follow arrows back to base case 0





```
LCS-Length(X, Y)
m = length(X), n = length(Y)
for i = 1 to m
    do c[i, 0] = 0
for j = 0 to n
    do c[0, j] = 0
for i = 1 to m
       do for j = 1 to n
              do if (x_i = y_j)
                     then c[i, j] = c[i - 1, j - 1] + 1
                              b\left[i\,,\;j\,\right] = " \leftarrow \uparrow "
              else if c[i - 1, j] \ge c[i, j - 1]
                                then c[i, j] = c[i - 1, j]

b[i, j] = "\uparrow"
                                 else c[i, j] = c[i, j - 1]
                                      b[i, j] = "\leftarrow"
return c and b
```

End