CSCE 411: Design and Analysis of Algorithms

Lecture 4: More Dynamic Programming: LCS

Date: January 28, 2025 *Nate Veldt, updated by Samson Zhou*

Course Logistics

- Read chapter 15 over the course of the next few lectures (focus: 15.1, 15.2, 15.3).
- Homework 2 due this Friday

1 The Longest Common Subsequence Problem

Preliminaries A sequence is an ordered list of elements:

$$X = \langle x_1, x_2, \dots, x_m \rangle$$

A subsequence of X is a sequence you obtain by deleting elements of X.

A sequence $Z = \langle z_1, z_2, \dots, z_k \rangle$ is a subsequence of X if there are indices $i_1 < i_2 < \cdots < i_k$ such that:

$$z_1 = x_{i_1}, z_2 = x_{i_2}, \dots, z_k = x_{i_k}.$$

Example

1.1 Defining the LCS Problem

Given two sequences

$$X = \langle x_1, x_2, \ldots, x_m \rangle$$
$$Y = \langle y_1, y_2, \ldots, y_n \rangle$$

A *longest common subsequence* (LCS) of (X, Y) is a

Brute Force Algorithm: Consider every subsequence of X and check if it's a subsequence of Y. Return the longest sequence of X where the answer is "yes".

Question 1. How many subsequences of X are there?

A $\Theta(m)$

B $\Theta(m^2)$

C $\Theta(2^m)$

Question 2. How long does it take to check whether a sequence Z of length $k < n$ is a subsequence of $Y = \langle y_1, y_2, \ldots, y_n \rangle$

A $\Theta(1)$

B $\Theta(k)$

C $\Theta(n)$

D $\Theta(nk)$

E $\Theta(2^n)$

1.2 LCS subproblems

Given an instance of LCS (X, Y) with sequences

$$\begin{aligned} X &= \langle x_1, x_2, \ldots, x_m \rangle \\ Y &= \langle y_1, y_2, \ldots, y_n \rangle \end{aligned}$$

Define

$$\begin{aligned} X_i &= \langle x_1, x_2, \ldots, x_i \rangle \\ Y_j &= \langle y_1, y_2, \ldots, y_j \rangle \end{aligned}$$

The subproblems of (X, Y) we will consider are of the form:

Question 3. How many different subproblems of this form are there?

A $\Theta(n + m)$

B $\Theta(nm)$

C $\Theta(2^n + 2^m)$

D $\Theta(2^{n+m})$

1.3 Proving optimal substructure

Theorem 1.1. *Let $Z = \langle z_1, z_2, \ldots, z_k \rangle$ be an LCS of (X, Y).*

1. *If $x_m = y_n$, then $z_k = x_m = y_n$ and Z_{k-1} is an LCS of (X_{m-1}, Y_{n-1})*
2. *If $x_m \neq y_n$ and $z_k \neq x_m$, then Z is an LCS of (X_{m-1}, Y)*
3. *If $x_m \neq y_n$ and $z_k \neq y_n$, then Z is an LCS of (X, Y_{n-1}).*

This means: *if $x_m \neq y_n$, the LCS of (X, Y) is contained within* (X, Y_{n-1}) or (X_{m-1}, Y).

Proof. See textbook. You will prove a small part of it for homework.
Illustration.

1.4 Defining a recursive approach

The theorem gives the following strategy for recursively finding the LCS of (X, Y)

- If $x_m = y_n$,

- If $x_m \neq y_n$,

Recurrence relations

$$c[i,j] = \text{ the length of an LCS of } (X_i, Y_j)$$

This satisfies the following recurrence:

Are subproblems overlapping?

2 Runtime Analysis

Recall that we have

$$c[i,j] = \text{ the length of an LCS of } (X_i, Y_j)$$

This satisfies the following recurrence:

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

Question 4. *Assuming we carefully solve subproblems only once, what should be the overall runtime of finding the LCS of (X, Y) using dynamic programming? Hint: how many different subproblems are there?*

A $\Theta(mn)$

B $\Theta(mn^2)$

C $\Theta(m^2 n)$

D $\Theta(m^2 n^2)$

2.1 Presenting an Actual Algorithm

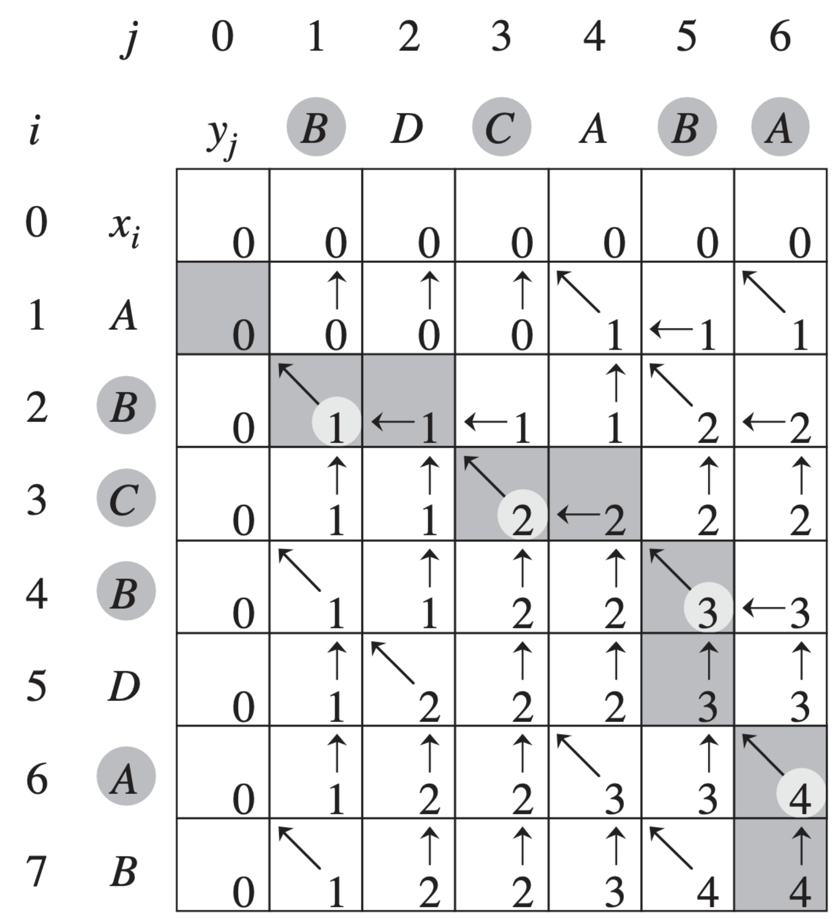
Think of the $c[i, j]$ values as being entries of a table or matrix.

```
LCS-LENGTH($X, Y$)
  $m$ = length($X$)
  $n$ = length($Y$)
  Let $c[0..m, 0..n]$ be a new table
  **for** $i = 1$ to $m$ **do**
      $c[i, 0] = 0$
  **end for**
  **for** $j = 1$ to $n$ **do**
      $c[0, j] = 0$
  **end for**
  **for** $i = 1$ to $m$ **do**
      **for** $j = 1$ to $n$ **do**
          **if** $x_i == y_j$ **then**


          **else**


          **end if**
      **end for**
  **end for**
  Return $c[m, n]$
```

2.2 Finding the longest common subsequence

The algorithm on the previous page gives the length of the longest possible subsequence, but not the subsequence itself. We can obtain the subsequence by inspecting the entries of c, starting with $c[m, n]$.

The textbook solution suggests storing arrows to tell you how to build up the subsequence.



	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