San Skulrattanakulchai
Gustavus Adolphus College
Oct 16, 2015

Longest Common Subsequence

GT: Ch 12.5

The topic of this handout concerns sequences from some fixed alphabet Σ. A sequence $S = s_1 s_2 \dots s_k$ is a *subsequence* of another sequence $T = t_1 t_2 \dots t_\ell$ if there exists a strictly increasing function $\phi : \{1, 2, \dots, k\} \to \{1, 2, \dots, \ell\}$ such that $s_i = t_{\phi(i)}$ for all $i = 1, 2, \dots, k$.

A sequence S is a *common subsequence* of sequences T and T' if S is a subsequence of both T and T'. A *longest common subsequence* (LCS) of sequences T and T' is a common subsequence of T and T' of maximum length.

Examples: `grim` is a subsequence of `algorithm` with $\phi(1) = 3$, $\phi(2) = 5$, $\phi(3) = 6$, and $\phi(4) = 9$. `dicor` is an LCS of `dynamicprogramming` and `divideandconquer`.

Problem

Let two sequences $X = x_1 x_2 \dots x_m$ and $Y = y_1 y_2 \dots y_n$ be given. We want to find an LCS of X and Y.

Dynamic Programming Solution

For $1 \le i \le m$ and $1 \le j \le n$, let $c(i, j)$ be the length of an LCS of $x_1 x_2 \dots x_i$ and $y_1 y_2 \dots y_j$.

We seek $c(m, n)$.

Optimal Substructure Property

Suppose $Z = z_1 z_2 \dots z_k$ is an LCS of $x_1 x_2 \dots x_i$ and $y_1 y_2 \dots y_j$.

If $x_i = y_j$, then we can infer that $x_i = z_k$, and that $z_1 z_2 \dots z_{k-1}$ is an LCS of $x_1 x_2 \dots x_{i-1}$ and $y_1 y_2 \dots y_{j-1}$.

If $x_i \ne y_j$, then $x_i \ne z_k$ or $y_j \ne z_k$. If $x_i \ne z_k$, we can show that Z is an LCS of $x_1 x_2 \dots x_{i-1}$ and $y_1 y_2 \dots y_j$. If $y_j \ne z_k$, we can show that Z is an LCS of $x_1 x_2 \dots x_i$ and $y_1 y_2 \dots y_{j-1}$.

We know that one of the above cases must occur. This gives us the following recurrence.

Recurrence

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

Question Explain how the artificial base case greatly helps simplify the recurrence.

Answer Without the artificial base case, we end up with this more complicated recurrence:

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

Example dynamic programming table

		s	a	i	n	t
	0	0	0	0	0	0
s	0	1	1	1	1	1
a	0	1	2	2	2	2
t	0	1	2	2	2	3
a	0	1	2	2	2	3
n	0	1	2	2	3	3

Longest Common Subsequence Algorithm

Step 1. Fill in a table of $c(\cdot,\cdot)$ values, plus a companion table of maximizers. We can fill in the table row-by-row, column-by-column, or diagonal-by-diagonal.

Step 2. Find the LCS by following maximizer pointers, starting from $c(m,n)$.

Running Time

- Step 1 fills in each table entry in $O(1)$ time, $O(mn)$ time total.

- Step 2 follows each pointer in $O(1)$ time, $O(m+n)$ time total.

Notes

(i) If we start by comparing X and Y from their ends, we get a similar optimal substructure and a corresponding right-to-left recurrence.

(ii) This problem illustrates 2D-dynamic programming where $c(i, j)$ depends on $O(1)$ "smaller" values and the time is $O(mn)$.