Dynamic Programming

A bottom-up solution technique to optimization problems.

The optimal solution is computed from optimal solutions to sub-problems.

Overlapping subproblems.

Longest Common Subsequence

Given a sequence $X = x_1, x_2, ..., x_m$, another sequence $Z = z_1, ..., z_k$ is a **subsequence** of X if there are indices $i_1 < i_2 < i_3.... < i_k$ such that for all $j = 1, ..., k$, $x_{i_j} = z_j$.

Given two sequences X and Y, a sequence Z is a **common subsequence** of X and Y if it is a subsequence of both X and Y.

The **longest common sequence (LCS) problem:** Given two sequences X and Y, find the longest common subsequence of both X and Y.

Optimal Substructure

Let $X = x_1, ..., x_k$, the i-th **prefix** of X is the sequence $X_i = x_1, ..., x_i$.

Theorem 1. *Let $Z = z_1, ..., z_k$ be the LCS of $X = x_1, ..., x_m$ and $Y = y_1, ..., y_n$,*

1. *If $x_m = y_n$ then $z_k = x_m = y_n$ and Z_{k-1} is the LCS of X_{m-1} and Y_{n-1}.*

2. *If $x_m \neq y_m$ then $z_k \neq x_m$ implies that Z is the LCS of X_{m-1} and Y.*

3. *If $x_m \neq y_m$ then $z_k \neq y_n$ implies that Z is the LCS of X and Y_{n-1}.*

Let $c[i,j]$ be the length of the LCS of X_i and Y_j.

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

A top-down solution can be exponential.

A bottom-up approach takes $O(nm)$ time, since there are only nm "subproblems" and each can be computed in $O(1)$ time if the smallest subproblems have already been computed.

Theorem 2. *The LCS-Length algorithm terminates in $O(nm)$ time and computes the correct LCS value.*

Elements of Dynamic Programming

Optimal substructures: A k-stage optimal solution is computed from $k-1$-stage optimal solutions.

Overlapping substructures: the same $k-1$-stage substructure is used in the computation of a number of k-stage substructures.