Lecture 25

Dynamic Programming: LCS (contd.)

Source: Introduction to Algorithms, CLRS

Optimal Substructure in LCS

Optimal Substructure in LCS

Claim: Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be two sequences such that $x_m = y_n$. Then,

Optimal Substructure in LCS

Claim: Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be two sequences such that $x_m = y_n$. Then, $Z = \textbf{LCS}(x_1 x_2 \ldots x_{m-1}, y_1 y_2 \ldots y_{n-1}) + x_m$ will be an LCS of X and Y.

Optimal Substructure in LCS

Claim: Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be two sequences such that $x_m = y_n$. Then, $Z = \mathbf{LCS}(x_1 x_2 \ldots x_{m-1}, y_1 y_2 \ldots y_{n-1}) + x_m$ will be an LCS of X and Y.

LCS of $x_1 x_2 \ldots x_{m-1}$ and $y_1 y_2 \ldots y_{n-1}$ followed by x_m

Optimal Substructure in LCS

Claim: Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be two sequences such that $x_m = y_n$. Then, $Z = \textbf{LCS}(x_1 x_2 \ldots x_{m-1}, y_1 y_2 \ldots y_{n-1}) + x_m$ will be an LCS of X and Y.

Proof:

Optimal Substructure in LCS

Claim: Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be two sequences such that $x_m = y_n$. Then, $Z = \mathbf{LCS}(x_1 x_2 \ldots x_{m-1}, y_1 y_2 \ldots y_{n-1}) + x_m$ will be an LCS of X and Y.

Proof: Suppose length of Z is k

Optimal Substructure in LCS

Claim: Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be two sequences such that $x_m = y_n$. Then, $Z = \mathbf{LCS}(x_1 x_2 \ldots x_{m-1}, y_1 y_2 \ldots y_{n-1}) + x_m$ will be an LCS of X and Y.

Proof: Suppose length of Z is k and, hence, length of $\mathbf{LCS}(x_1 x_2 \ldots x_{m-1}, y_1 y_2 \ldots y_{n-1})$ is $k - 1$.

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be two sequences such that $x_m = y_n$. Then, $Z = \mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1}) + x_m$ will be an LCS of X and Y.

Proof: Suppose length of Z is k and, hence, length of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1})$ is $k-1$.

Suppose $W = w_1w_2\ldots w_l$, not Z, is an LCS of X and Y, and $l > k$.

Optimal Substructure in LCS

Claim: Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be two sequences such that $x_m = y_n$. Then, $Z = \mathbf{LCS}(x_1 x_2 \ldots x_{m-1}, y_1 y_2 \ldots y_{n-1}) + x_m$ will be an LCS of X and Y.

Proof: Suppose length of Z is k and, hence, length of $\mathbf{LCS}(x_1 x_2 \ldots x_{m-1}, y_1 y_2 \ldots y_{n-1})$ is $k - 1$.

Suppose $W = w_1 w_2 \ldots w_l$, not Z, is an LCS of X and Y, and $l > k$.

Clearly, W ends with x_m (or y_n).

Optimal Substructure in LCS

Claim: Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be two sequences such that $x_m = y_n$. Then, $Z = \mathbf{LCS}(x_1 x_2 \ldots x_{m-1}, y_1 y_2 \ldots y_{n-1}) + x_m$ will be an LCS of X and Y.

Proof: Suppose length of Z is k and, hence, length of $\mathbf{LCS}(x_1 x_2 \ldots x_{m-1}, y_1 y_2 \ldots y_{n-1})$ is $k - 1$.

Suppose $W = w_1 w_2 \ldots w_l$, not Z, is an LCS of X and Y, and $l > k$.

Clearly, W ends with x_m (or y_n). (Why?)

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be two sequences such that $x_m = y_n$. Then, $Z = \mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1}) + x_m$ will be an LCS of X and Y.

Proof: Suppose length of Z is k and, hence, length of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1})$ is $k-1$.

Suppose $W = w_1w_2\ldots w_l$, not Z, is an LCS of X and Y, and $l > k$.

Clearly, W ends with x_m (or y_n). (Why?)

$$X = x_1x_2x_3\ldots x_{m-1}x_m$$

$$Y = y_1y_2y_3\ldots y_{n-1}y_n$$

$$W = w_1w_2w_3\ldots w_{l-1}w_l$$

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be two sequences such that $x_m = y_n$. Then, $Z = \mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1}) + x_m$ will be an LCS of X and Y.

Proof: Suppose length of Z is k and, hence, length of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1})$ is $k-1$.

Suppose $W = w_1w_2\ldots w_l$, not Z, is an LCS of X and Y, and $l > k$.

Clearly, W ends with x_m (or y_n). (Why?)

$$X = x_1x_2x_3\ldots x_{m-1}\cancel{x_m}$$

$$Y = y_1y_2y_3\ldots y_{n-1}\cancel{y_n}$$

$$W = w_1w_2w_3\ldots w_{l-1}\cancel{w_l}$$

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be two sequences such that $x_m = y_n$. Then, $Z = \mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1}) + x_m$ will be an LCS of X and Y.

Proof: Suppose length of Z is k and, hence, length of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1})$ is $k-1$.

Suppose $W = w_1w_2\ldots w_l$, not Z, is an LCS of X and Y, and $l > k$.

Clearly, W ends with x_m (or y_n). (Why?)

$X = x_1x_2x_3\ldots x_{m-1}\cancel{x_m}$

$Y = y_1y_2y_3\ldots y_{n-1}\cancel{y_n}$

$W = w_1w_2w_3\ldots w_{l-1}\cancel{w_l}$

$w_1w_2w_3\ldots w_{l-1}$ must be a common subsequence of $x_1x_2x_3\ldots x_{m-1}$ and $y_1y_2y_3\ldots y_{n-1}$

Optimal Substructure in LCS

Claim: Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be two sequences such that $x_m = y_n$. Then, $Z = \mathbf{LCS}(x_1 x_2 \ldots x_{m-1}, y_1 y_2 \ldots y_{n-1}) + x_m$ will be an LCS of X and Y.

Proof: Suppose length of Z is k and, hence, length of $\mathbf{LCS}(x_1 x_2 \ldots x_{m-1}, y_1 y_2 \ldots y_{n-1})$ is $k - 1$.

Suppose $W = w_1 w_2 \ldots w_l$, not Z, is an LCS of X and Y, and $l > k$.

Clearly, W ends with x_m (or y_n). (Why?)

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be two sequences such that $x_m = y_n$. Then, $Z = \mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1}) + x_m$ will be an LCS of X and Y.

Proof: Suppose length of Z is k and, hence, length of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1})$ is $k-1$.

Suppose $W = w_1w_2\ldots w_l$, not Z, is an LCS of X and Y, and $l > k$.

Clearly, W ends with x_m (or y_n). (Why?)

Then $w_1w_2\ldots w_{l-1}$ will be a **CS** of $x_1x_2\ldots x_{m-1}$ and $y_1y_2\ldots y_{n-1}$.

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be two sequences such that $x_m = y_n$. Then, $Z = \mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1}) + x_m$ will be an LCS of X and Y.

Proof: Suppose length of Z is k and, hence, length of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1})$ is $k-1$.

Suppose $W = w_1w_2\ldots w_l$, not Z, is an LCS of X and Y, and $l > k$.

Clearly, W ends with x_m (or y_n). (Why?)

Then $w_1w_2\ldots w_{l-1}$ will be a **CS** of $x_1x_2\ldots x_{m-1}$ and $y_1y_2\ldots y_{n-1}$.

This is a contradiction as $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1})$ is shorter than $w_1w_2\ldots w_{l-1}$.

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be two sequences such that $x_m = y_n$. Then, $Z = \mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1}) + x_m$ will be an LCS of X and Y.

Proof: Suppose length of Z is k and, hence, length of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1})$ is $k-1$.

Suppose $W = w_1w_2\ldots w_l$, not Z, is an LCS of X and Y, and $l > k$.

Clearly, W ends with x_m (or y_n). (Why?)

Then $w_1w_2\ldots w_{l-1}$ will be a **CS** of $x_1x_2\ldots x_{m-1}$ and $y_1y_2\ldots y_{n-1}$.

This is a contradiction as $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1})$ is shorter than $w_1w_2\ldots w_{l-1}$.

length $k-1$

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be two sequences such that $x_m = y_n$. Then, $Z = \mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1}) + x_m$ will be an LCS of X and Y.

Proof: Suppose length of Z is k and, hence, length of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1})$ is $k-1$.

Suppose $W = w_1w_2\ldots w_l$, not Z, is an LCS of X and Y, and $l > k$.

Clearly, W ends with x_m (or y_n). (Why?)

Then $w_1w_2\ldots w_{l-1}$ will be a **CS** of $x_1x_2\ldots x_{m-1}$ and $y_1y_2\ldots y_{n-1}$.

This is a contradiction as $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1})$ is shorter than $w_1w_2\ldots w_{l-1}$.

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be two sequences such that $x_m = y_n$. Then, $Z = \mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1}) + x_m$ will be an LCS of X and Y.

Proof: Suppose length of Z is k and, hence, length of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1})$ is $k-1$.

Suppose $W = w_1w_2\ldots w_l$, not Z, is an LCS of X and Y, and $l > k$.

Clearly, W ends with x_m (or y_n). (Why?)

Then $w_1w_2\ldots w_{l-1}$ will be a **CS** of $x_1x_2\ldots x_{m-1}$ and $y_1y_2\ldots y_{n-1}$.

This is a contradiction as $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_{n-1})$ is shorter than $w_1w_2\ldots w_{l-1}$.

■

Optimal Substructure in LCS

Optimal Substructure in LCS

Claim: Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be sequences such that $x_m \neq y_n$. Then, at least

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be sequences such that $x_m \neq y_n$. Then, at least one out of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ and $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ will be an $\mathbf{LCS}(X, Y)$.

Optimal Substructure in LCS

Claim: Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be sequences such that $x_m \neq y_n$. Then, at least one out of $\mathbf{LCS}(x_1 x_2 \ldots x_{m-1}, y_1 y_2 \ldots y_n)$ and $\mathbf{LCS}(x_1 x_2 \ldots x_m, y_1 y_2 \ldots y_{n-1})$ will be an $\mathbf{LCS}(X, Y)$.

Proof:

Optimal Substructure in LCS

Claim: Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be sequences such that $x_m \neq y_n$. Then, at least one out of $\mathbf{LCS}(x_1 x_2 \ldots x_{m-1}, y_1 y_2 \ldots y_n)$ and $\mathbf{LCS}(x_1 x_2 \ldots x_m, y_1 y_2 \ldots y_{n-1})$ will be an $\mathbf{LCS}(X, Y)$.

Proof: Suppose neither $\mathbf{LCS}(x_1 x_2 \ldots x_{m-1}, y_1 y_2 \ldots y_n)$ nor $\mathbf{LCS}(x_1 x_2 \ldots x_m, y_1 y_2 \ldots y_{n-1})$ is an $\mathbf{LCS}(X, Y)$.

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be sequences such that $x_m \neq y_n$. Then, at least one out of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ and $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ will be an $\mathbf{LCS}(X, Y)$.

Proof: Suppose neither $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ nor $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ is an $\mathbf{LCS}(X, Y)$.

Let $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ be of length j.

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be sequences such that $x_m \neq y_n$. Then, at least one out of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ and $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ will be an $\mathbf{LCS}(X, Y)$.

Proof: Suppose neither $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ nor $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ is an $\mathbf{LCS}(X, Y)$.

Let $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ be of length j.

Let $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ be of length k.

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be sequences such that $x_m \neq y_n$. Then, at least one out of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ and $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ will be an $\mathbf{LCS}(X, Y)$.

Proof: Suppose neither $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ nor $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ is an $\mathbf{LCS}(X, Y)$.

Let $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ be of length j.

Let $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ be of length k.

Let W be an $\mathbf{LCS}(X, Y)$ of length $l > \mathbf{Max}(j, k)$.

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be sequences such that $x_m \neq y_n$. Then, at least one out of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ and $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ will be an $\mathbf{LCS}(X, Y)$.

Proof: Suppose neither $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ nor $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ is an $\mathbf{LCS}(X, Y)$.

Let $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ be of length j.

Let $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ be of length k.

Let W be an $\mathbf{LCS}(X, Y)$ of length $l > \mathbf{Max}(j, k)$.

Since $x_m \neq y_n$, W cannot end with both x_m and y_n.

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be sequences such that $x_m \neq y_n$. Then, at least one out of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ and $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ will be an $\mathbf{LCS}(X, Y)$.

Proof: Suppose neither $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ nor $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ is an $\mathbf{LCS}(X, Y)$.

Let $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ be of length j.

Let $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ be of length k.

Let W be an $\mathbf{LCS}(X, Y)$ of length $l > \mathbf{Max}(j, k)$.

Since $x_m \neq y_n$, W cannot end with both x_m and y_n.

$$X = x_1x_2x_3\ldots x_{m-1}x_m$$

$$Y = y_1y_2y_3\ldots y_{n-1}y_n$$

$$W = w_1w_2w_3\ldots w_{l-1}w_l$$

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be sequences such that $x_m \neq y_n$. Then, at least one out of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ and $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ will be an $\mathbf{LCS}(X, Y)$.

Proof: Suppose neither $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ nor $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ is an $\mathbf{LCS}(X, Y)$.

Let $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ be of length j.

Let $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ be of length k.

Let W be an $\mathbf{LCS}(X, Y)$ of length $l > \mathbf{Max}(j, k)$.

Since $x_m \neq y_n$, W cannot end with both x_m and y_n.

$$X = x_1x_2x_3\ldots x_{m-1}\cancel{x_m}$$

$$Y = y_1y_2y_3\ldots y_{n-1}y_n$$

$$W = w_1w_2w_3\ldots w_{l-1}w_l$$

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be sequences such that $x_m \neq y_n$. Then, at least one out of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ and $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ will be an $\mathbf{LCS}(X, Y)$.

Proof: Suppose neither $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ nor $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ is an $\mathbf{LCS}(X, Y)$.

Let $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ be of length j.

Let $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ be of length k.

Let W be an $\mathbf{LCS}(X, Y)$ of length $l > \mathbf{Max}(j, k)$.

Since $x_m \neq y_n$, W cannot end with both x_m and y_n.

If W doesn't end with x_m (WLOG), then W will be an **CS** of $x_1x_2\ldots x_{m-1}$ and $y_1y_2\ldots y_n$.

$$X = x_1x_2x_3\ldots x_{m-1}\cancel{x_m}$$

$$Y = y_1y_2y_3\ldots y_{n-1}y_n$$

$$W = w_1w_2w_3\ldots w_{l-1}w_l$$

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be sequences such that $x_m \neq y_n$. Then, at least one out of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ and $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ will be an $\mathbf{LCS}(X, Y)$.

Proof: Suppose neither $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ nor $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ is an $\mathbf{LCS}(X, Y)$.

Let $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ be of length j.

Let $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ be of length k.

Let W be an $\mathbf{LCS}(X, Y)$ of length $l > \mathbf{Max}(j, k)$.

Since $x_m \neq y_n$, W cannot end with both x_m and y_n.

$$X = x_1x_2x_3\ldots x_{m-1}\cancel{x_m}$$

$$Y = y_1y_2y_3\ldots y_{n-1}y_n$$

$$W = w_1w_2w_3\ldots w_{l-1}w_l$$

If W doesn't end with x_m (WLOG), then W will be an $\mathbf{CS}$ of $x_1x_2\ldots x_{m-1}$ and $y_1y_2\ldots y_n$.

This is a contradiction as W is longer than $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$.

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be sequences such that $x_m \neq y_n$. Then, at least one out of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ and $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ will be an $\mathbf{LCS}(X, Y)$.

Proof: Suppose neither $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ nor $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ is an $\mathbf{LCS}(X, Y)$.

Let $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ be of length j.

Let $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ be of length k.

Let W be an $\mathbf{LCS}(X, Y)$ of length $l > \mathbf{Max}(j, k)$.

Since $x_m \neq y_n$, W cannot end with both x_m and y_n.

$$X = x_1x_2x_3\ldots x_{m-1}\cancel{x_m}$$

$$Y = y_1y_2y_3\ldots y_{n-1}y_n$$

$$W = w_1w_2w_3\ldots w_{l-1}w_l$$

If W doesn't end with x_m (WLOG), then W will be an $\mathbf{CS}$ of $x_1x_2\ldots x_{m-1}$ and $y_1y_2\ldots y_n$.

This is a contradiction as W is longer than $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$.

length l

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be sequences such that $x_m \neq y_n$. Then, at least one out of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ and $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ will be an $\mathbf{LCS}(X, Y)$.

Proof: Suppose neither $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ nor $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ is an $\mathbf{LCS}(X, Y)$.

Let $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ be of length j.

Let $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ be of length k.

Let W be an $\mathbf{LCS}(X, Y)$ of length $l > \mathbf{Max}(j, k)$.

$$X = x_1x_2x_3\ldots x_{m-1}\cancel{x_m}$$

$$Y = y_1y_2y_3\ldots y_{n-1}y_n$$

$$W = w_1w_2w_3\ldots w_{l-1}w_l$$

Since $x_m \neq y_n$, W cannot end with both x_m and y_n.

If W doesn't end with x_m (WLOG), then W will be an $\mathbf{CS}$ of $x_1x_2\ldots x_{m-1}$ and $y_1y_2\ldots y_n$.

This is a contradiction as W is longer than $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$.

length l

length j

Optimal Substructure in LCS

Claim: Let $X = x_1x_2\ldots x_m$ and $Y = y_1y_2\ldots y_n$ be sequences such that $x_m \neq y_n$. Then, at least one out of $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ and $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ will be an $\mathbf{LCS}(X, Y)$.

Proof: Suppose neither $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ nor $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ is an $\mathbf{LCS}(X, Y)$.

Let $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$ be of length j.

Let $\mathbf{LCS}(x_1x_2\ldots x_m, y_1y_2\ldots y_{n-1})$ be of length k.

Let W be an $\mathbf{LCS}(X, Y)$ of length $l > \mathbf{Max}(j, k)$.

Since $x_m \neq y_n$, W cannot end with both x_m and y_n.

$$X = x_1x_2x_3\ldots x_{m-1}\cancel{x_m}$$

$$Y = y_1y_2y_3\ldots y_{n-1}y_n$$

$$W = w_1w_2w_3\ldots w_{l-1}w_l$$

If W doesn't end with x_m (WLOG), then W will be an $\mathbf{CS}$ of $x_1x_2\ldots x_{m-1}$ and $y_1y_2\ldots y_n$.

This is a contradiction as W is longer than $\mathbf{LCS}(x_1x_2\ldots x_{m-1}, y_1y_2\ldots y_n)$.

length l

length j

■

Recurrence for LCS

Recurrence for LCS

Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be two sequences.

Recurrence for LCS

Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be two sequences.

Let $LCS(i, j)$ = length of the **LCS** of $x_1 x_2 \ldots x_i$ and $y_1 y_2 \ldots y_j$.

Recurrence for LCS

Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be two sequences.

Let $LCS(i, j)$ = length of the **LCS** of $x_1 x_2 \ldots x_i$ and $y_1 y_2 \ldots y_j$. Then,

Recurrence for LCS

Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be two sequences.

Let $LCS(i, j)$ = length of the **LCS** of $x_1 x_2 \ldots x_i$ and $y_1 y_2 \ldots y_j$. Then,

$$LCS(i, j) =$$

Recurrence for LCS

Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be two sequences.

Let $LCS(i, j)$ = length of the **LCS** of $x_1 x_2 \ldots x_i$ and $y_1 y_2 \ldots y_j$. Then,

$$LCS(i, j) = \begin{cases} \end{cases}$$

Recurrence for LCS

Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be two sequences.

Let $LCS(i, j)$ = length of the **LCS** of $x_1 x_2 \ldots x_i$ and $y_1 y_2 \ldots y_j$. Then,

$$LCS(i,j) = \begin{cases} 0, & \text{if } i = 0 \text{ or } j = 0 \\ \\ \\ \end{cases}$$

Recurrence for LCS

Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be two sequences.

Let $LCS(i, j)$ = length of the **LCS** of $x_1 x_2 \ldots x_i$ and $y_1 y_2 \ldots y_j$. Then,

$$LCS(i,j) = \begin{cases} 0, & \text{if } i = 0 \text{ or } j = 0 \\ LCS(i-1, j-1) + 1, & \text{if } x_i = y_j \end{cases}$$

Recurrence for LCS

Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be two sequences.

Let $LCS(i, j)$ = length of the **LCS** of $x_1 x_2 \ldots x_i$ and $y_1 y_2 \ldots y_j$. Then,

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

Overlapping Subproblems in LCS

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

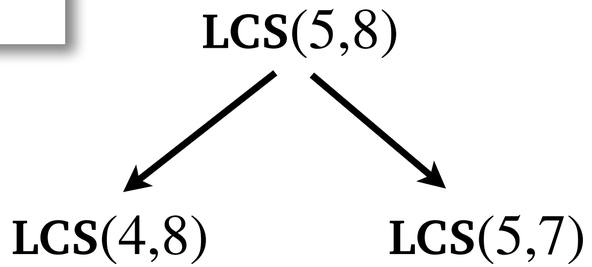
Overlapping Subproblems in LCS

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

LCS(5,8)

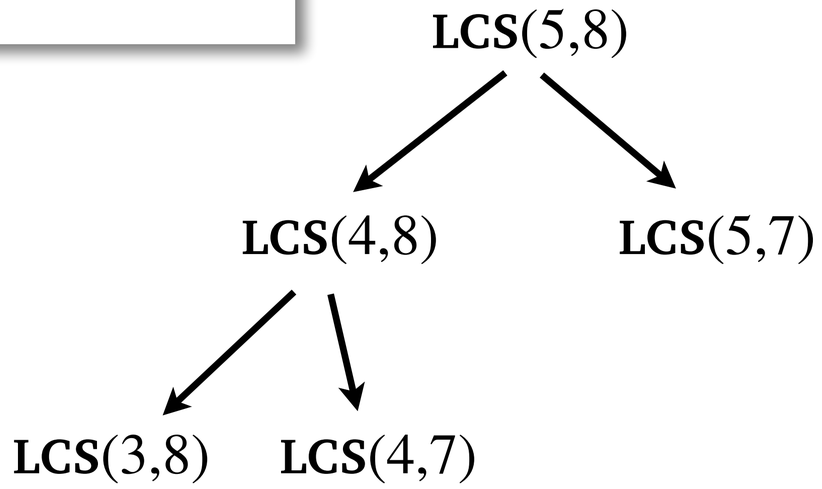
Overlapping Subproblems in LCS

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



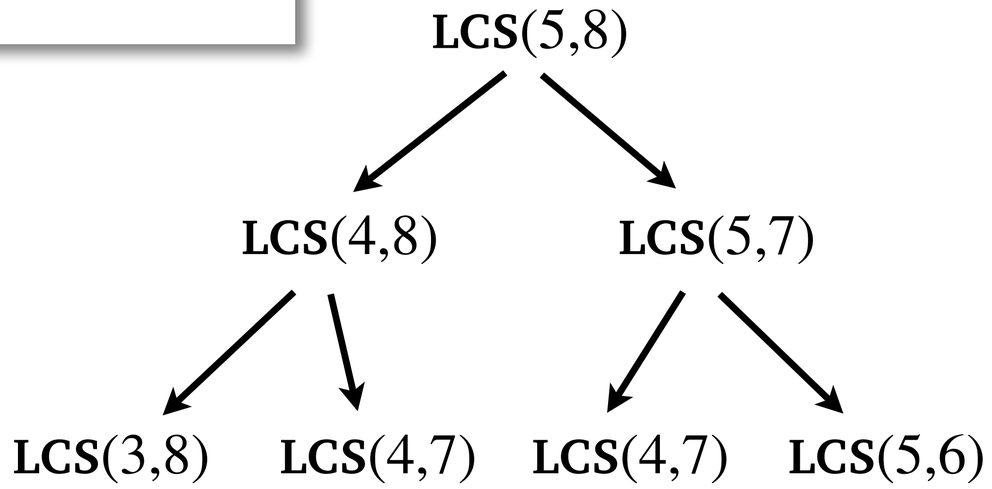
Overlapping Subproblems in LCS

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



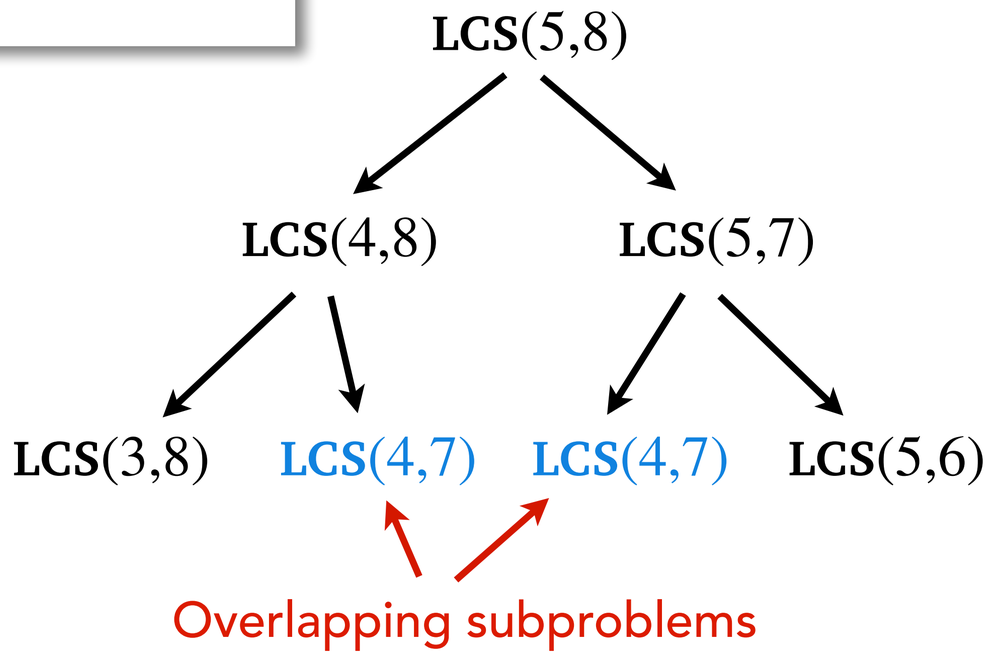
Overlapping Subproblems in LCS

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



Overlapping Subproblems in LCS

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



Top-Down DP for LCS

Top-Down DP for LCS

$dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$

Top-Down DP for LCS

$dp[i][j]$ stores $LCS(i, j)$

$dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$

Top-Down DP for LCS

$dp[i][j]$ stores $LCS(i, j)$

$dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$

$dp[0][0:m] = 0, dp[0:n][0] = 0$

Top-Down DP for LCS

$dp[i][j]$ stores $LCS(i, j)$

$dp[0 : n][0 : m] = \{-1, -1, \ldots, -1\}$

$dp[0][0 : m] = 0, dp[0 : n][0] = 0$

dp

	0	0	0	0	0	0	0
0	0	-1	-1	-1	-1	-1	-1
n 0	0	-1	-1	-1	-1	-1	-1
0	0	-1	-1	-1	-1	-1	-1
0	0	-1	-1	-1	-1	-1	-1
0	0	-1	-1	-1	-1	-1	-1

m

Top-Down DP for LCS

$dp[i][j]$ stores $LCS(i, j)$

$dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$

$dp[0][0:m] = 0, dp[0:n][0] = 0$

Top-Down DP for LCS

$dp[i][j]$ stores $LCS(i, j)$

$dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$

$dp[0][0:m] = 0, dp[0:n][0] = 0$

LCS(i, j):

Top-Down DP for LCS

$dp[i][j]$ stores $LCS(i, j)$

$dp[0 : n][0 : m] = \{-1, -1, \ldots, -1\}$

$dp[0][0 : m] = 0, dp[0 : n][0] = 0$

LCS(i, j):

Call with **LCS**$(|X|, |Y|)$

Top-Down DP for LCS

$dp[i][j]$ stores $LCS(i, j)$

$dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$

$dp[0][0:m] = 0, dp[0:n][0] = 0$

LCS(i, j):

Top-Down DP for LCS

$dp[i][j]$ stores $LCS(i, j)$

$dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$

$dp[0][0:m] = 0, dp[0:n][0] = 0$

LCS(i, j):

1. **if** $(dp[i][j] \neq -1)$

Top-Down DP for LCS

$dp[i][j]$ stores $LCS(i, j)$

$dp[0 : n][0 : m] = \{-1, -1, \ldots, -1\}$

$dp[0][0 : m] = 0, dp[0 : n][0] = 0$

LCS(i, j):

```
1.  if (dp[i][j] ≠ −1)
2.      return dp[i][j]
```

Top-Down DP for LCS

$dp[i][j]$ stores $LCS(i, j)$

$dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$

$dp[0][0:m] = 0, dp[0:n][0] = 0$

LCS(i, j):

1. **if** $(dp[i][j] \neq -1)$
2. **return** $dp[i][j]$
3. **if** $x_i == y_j$

Top-Down DP for LCS

$dp[i][j]$ stores $LCS(i, j)$

$dp[0 : n][0 : m] = \{-1, -1, \ldots, -1\}$

$dp[0][0 : m] = 0, dp[0 : n][0] = 0$

LCS(i, j):

```
1.  if (dp[i][j] ≠ −1)
2.      return dp[i][j]
3.  if x_i == y_j
4.      dp[i][j] = LCS(i − 1, j − 1) + 1
```

Top-Down DP for LCS

$dp[i][j]$ stores $LCS(i, j)$

$dp[0 : n][0 : m] = \{-1, -1, \ldots, -1\}$

$dp[0][0 : m] = 0, dp[0 : n][0] = 0$

LCS(i, j):

```
1.  if (dp[i][j] ≠ −1)
2.      return dp[i][j]
3.  if x_i == y_j
4.      dp[i][j] = LCS(i − 1, j − 1) + 1
5.  else
```

Top-Down DP for LCS

$dp[i][j]$ stores $LCS(i, j)$

$dp[0 : n][0 : m] = \{-1, -1, \ldots, -1\}$

$dp[0][0 : m] = 0, dp[0 : n][0] = 0$

LCS(i, j):

```
1.  if (dp[i][j] ≠ −1)
2.      return dp[i][j]
3.  if x_i == y_j
4.      dp[i][j] = LCS(i − 1, j − 1) + 1
5.  else
6.      dp[i][j] = Max(LCS(i − 1, j), LCS(i, j − 1))
```

Top-Down DP for LCS

$dp[i][j]$ stores $LCS(i, j)$

$dp[0 : n][0 : m] = \{-1, -1, \ldots, -1\}$

$dp[0][0 : m] = 0, dp[0 : n][0] = 0$

LCS(i, j):

```
1.  if (dp[i][j] ≠ −1)
2.      return dp[i][j]
3.  if x_i == y_j
4.      dp[i][j] = LCS(i − 1, j − 1) + 1
5.  else
6.      dp[i][j] = Max(LCS(i − 1, j), LCS(i, j − 1))
7.  return dp[i][j]
```

Top-Down DP for LCS

$dp[i][j]$ stores $LCS(i, j)$

$dp[0 : n][0 : m] = \{-1, -1, \ldots, -1\}$

$dp[0][0 : m] = 0, dp[0 : n][0] = 0$

LCS(i, j):

```
1.  if (dp[i][j] ≠ −1)
2.      return dp[i][j]
3.  if x_i == y_j
4.      dp[i][j] = LCS(i − 1, j − 1) + 1
5.  else
6.      dp[i][j] = Max(LCS(i − 1, j), LCS(i, j − 1))
7.  return dp[i][j]
```

Time Complexity: $O(nm)$

Top-Down DP for LCS

$dp[i][j]$ stores $LCS(i, j)$

$dp[0 : n][0 : m] = \{-1, -1, \ldots, -1\}$

$dp[0][0 : m] = 0, dp[0 : n][0] = 0$

LCS(i, j):

```
1.  if (dp[i][j] ≠ −1)
2.      return dp[i][j]
3.  if x_i == y_j
4.      dp[i][j] = LCS(i − 1, j − 1) + 1
5.  else
6.      dp[i][j] = Max(LCS(i − 1, j), LCS(i, j − 1))
7.  return dp[i][j]
```

Time Complexity: $O(nm)$

Why?

Bottom-Up DP for LCS

Bottom-Up DP for LCS

LCS(X, Y)

Bottom-Up DP for LCS

LCS(X, Y)

1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$

Bottom-Up DP for LCS

LCS(X, Y)

1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0$

Bottom-Up DP for LCS

LCS(X, Y)

1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0$

dp

	0	0	0	0	0	0	0
	0	-1	-1	-1	-1	-1	-1
n	0	-1	-1	-1	-1	-1	-1
	0	-1	-1	-1	-1	-1	-1
	0	-1	-1	-1	-1	-1	-1
	0	-1	-1	-1	-1	-1	-1

m

Bottom-Up DP for LCS

LCS(X, Y)

1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0$
3. **for** $i = 1$ **to** n

dp

0	0	0	0	0	0	0
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1

n

m

Bottom-Up DP for LCS

LCS(X, Y)

1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0$
3. **for** $i = 1$ **to** n
4. **for** $j = 1$ **to** m

dp

0	0	0	0	0	0	0
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1

n

m

Bottom-Up DP for LCS

LCS(X, Y)

1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0$
3. **for** $i = 1$ **to** n
4. **for** $j = 1$ **to** m
5. **if** $x_i == y_j$

dp

0	0	0	0	0	0	0
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1

n

m

Bottom-Up DP for LCS

LCS(X, Y)

1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0$
3. **for** $i = 1$ **to** n
4. **for** $j = 1$ **to** m
5. **if** $x_i == y_j$
6. $dp[i][j] = dp[i-1][j-1] + 1$

dp

0	0	0	0	0	0	0
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1

n

m

Bottom-Up DP for LCS

LCS(X, Y)

1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0$
3. **for** $i = 1$ **to** n
4. **for** $j = 1$ **to** m
5. **if** $x_i == y_j$
6. $dp[i][j] = dp[i-1][j-1] + 1$
7. **else**

dp

0	0	0	0	0	0	0
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1

n

m

Bottom-Up DP for LCS

LCS(X, Y)

1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0$
3. **for** $i = 1$ **to** n
4. **for** $j = 1$ **to** m
5. **if** $x_i == y_j$
6. $dp[i][j] = dp[i-1][j-1] + 1$
7. **else**
8. $dp[i][j] = \textbf{Max}(dp[i-1][j], dp[i][j-1])$

dp

0	0	0	0	0	0	0
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1

n

m

Bottom-Up DP for LCS

LCS(X, Y)

1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0$
3. **for** $i = 1$ **to** n
4. **for** $j = 1$ **to** m
5. **if** $x_i == y_j$
6. $dp[i][j] = dp[i-1][j-1] + 1$
7. **else**
8. $dp[i][j] = \textbf{Max}(dp[i-1][j], dp[i][j-1])$
9. **return** $dp[n][m]$

dp

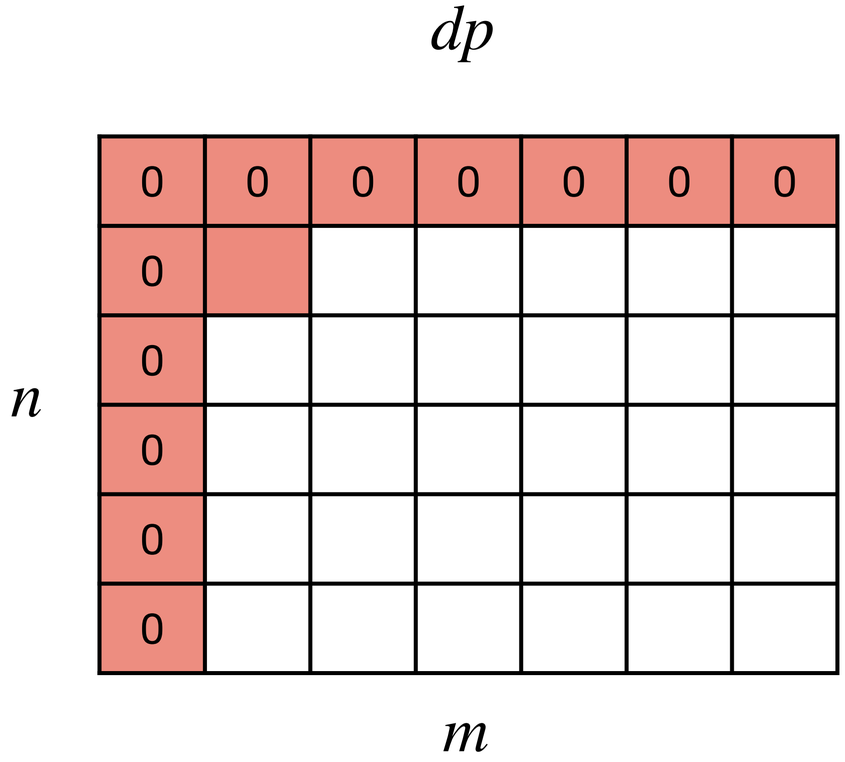
n

0	0	0	0	0	0	0
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	-1	-1	-1

m

Bottom-Up DP for LCS

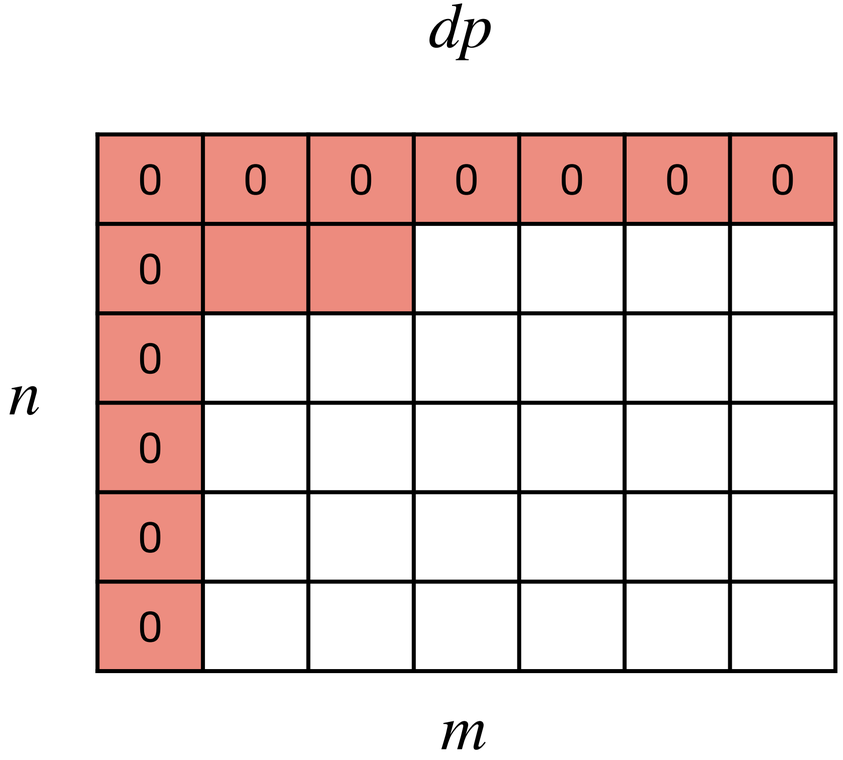
```
LCS(X, Y)
1.  dp[0 : n][0 : m] = {−1, − 1,..., − 1}
2.  dp[0][0 : m] = 0, dp[0 : n][0] = 0
3.  for i = 1 to n
4.      for j = 1 to m
5.          if x_i == y_j
6.              dp[i][j] = dp[i − 1][j − 1] + 1
7.          else
8.              dp[i][j] = Max(dp[i − 1][j], dp[i][j − 1])
9.  return dp[n][m]
```



Bottom-Up DP for LCS

LCS(X, Y)

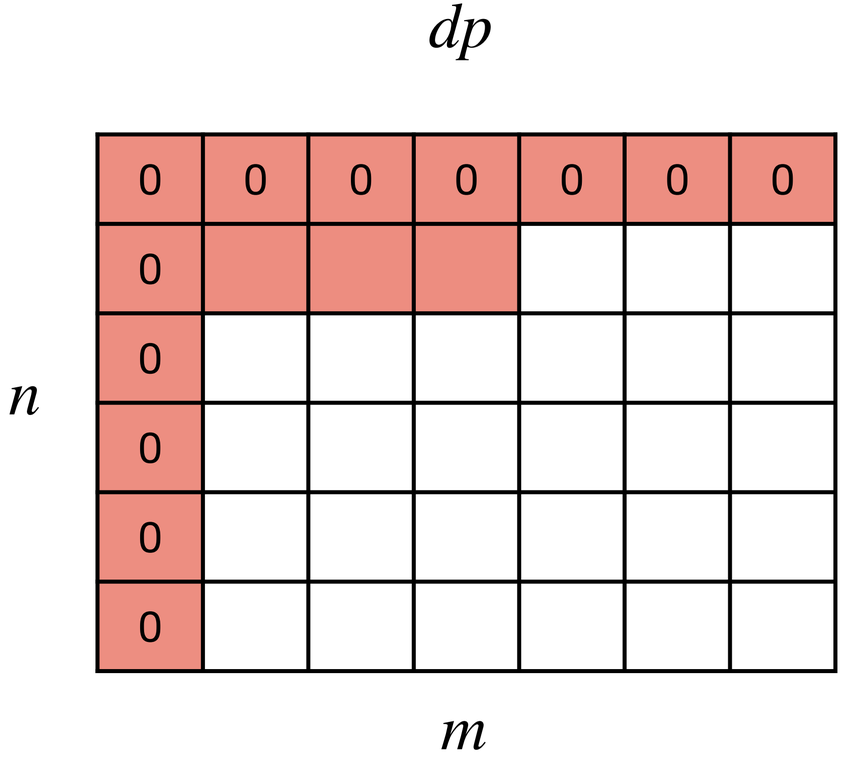
1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0$
3. **for** $i = 1$ **to** n
4. **for** $j = 1$ **to** m
5. **if** $x_i == y_j$
6. $dp[i][j] = dp[i-1][j-1] + 1$
7. **else**
8. $dp[i][j] = \textbf{Max}(dp[i-1][j], dp[i][j-1])$
9. **return** $dp[n][m]$



Bottom-Up DP for LCS

LCS(X, Y)

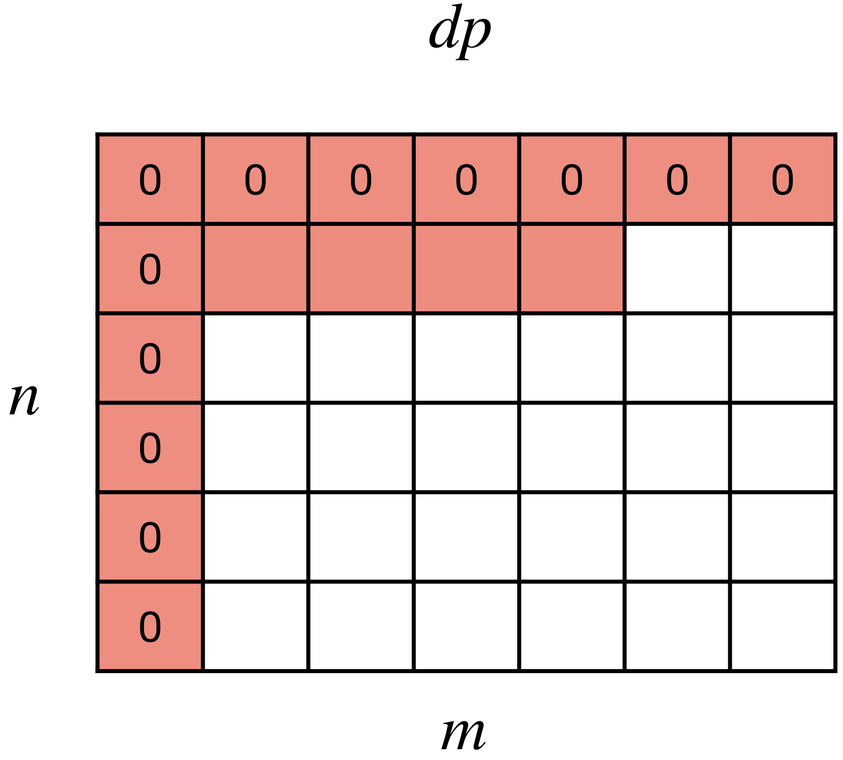
1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0$
3. **for** $i = 1$ **to** n
4. **for** $j = 1$ **to** m
5. **if** $x_i == y_j$
6. $dp[i][j] = dp[i-1][j-1] + 1$
7. **else**
8. $dp[i][j] = \textbf{Max}(dp[i-1][j], dp[i][j-1])$
9. **return** $dp[n][m]$



Bottom-Up DP for LCS

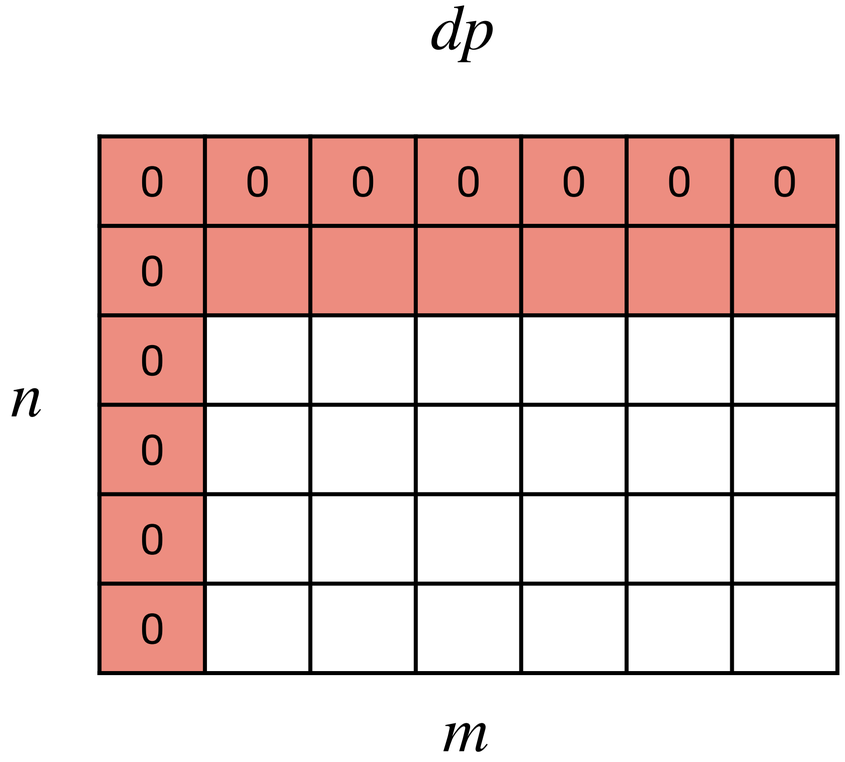
LCS(X, Y)

1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0$
3. **for** $i = 1$ **to** n
4. **for** $j = 1$ **to** m
5. **if** $x_i == y_j$
6. $dp[i][j] = dp[i-1][j-1] + 1$
7. **else**
8. $dp[i][j] = \textbf{Max}(dp[i-1][j], dp[i][j-1])$
9. **return** $dp[n][m]$



Bottom-Up DP for LCS

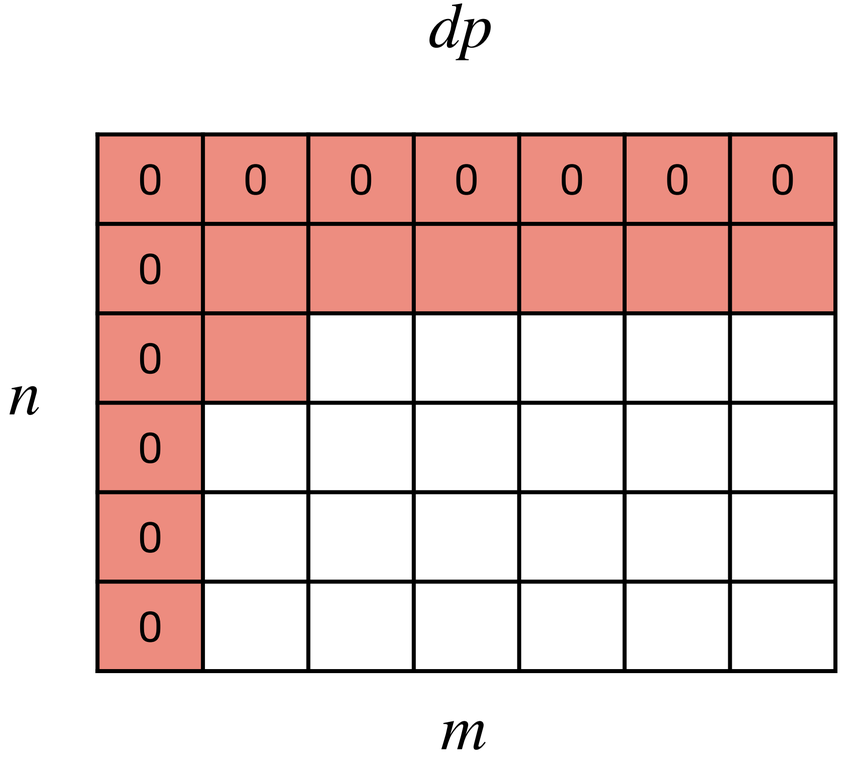
```
LCS(X, Y)
1.  dp[0 : n][0 : m] = {−1, − 1,..., − 1}
2.  dp[0][0 : m] = 0, dp[0 : n][0] = 0
3.  for i = 1 to n
4.      for j = 1 to m
5.          if x_i == y_j
6.              dp[i][j] = dp[i − 1][j − 1] + 1
7.          else
8.              dp[i][j] = Max(dp[i − 1][j], dp[i][j − 1])
9.  return dp[n][m]
```



Bottom-Up DP for LCS

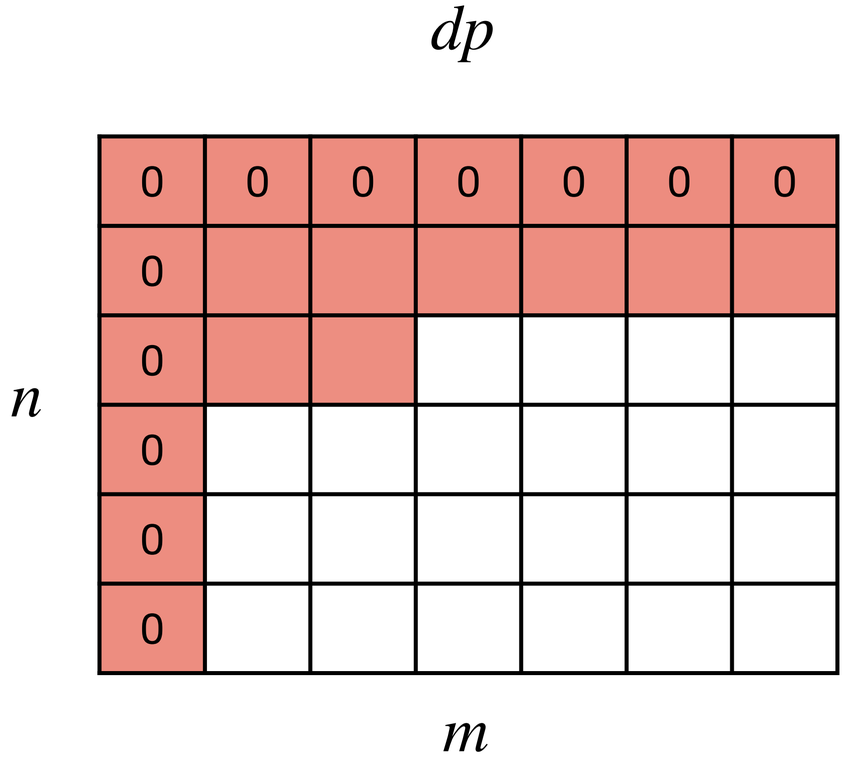
LCS(X, Y)

1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0$
3. **for** $i = 1$ **to** n
4. **for** $j = 1$ **to** m
5. **if** $x_i == y_j$
6. $dp[i][j] = dp[i-1][j-1] + 1$
7. **else**
8. $dp[i][j] = \textbf{Max}(dp[i-1][j], dp[i][j-1])$
9. **return** $dp[n][m]$



Bottom-Up DP for LCS

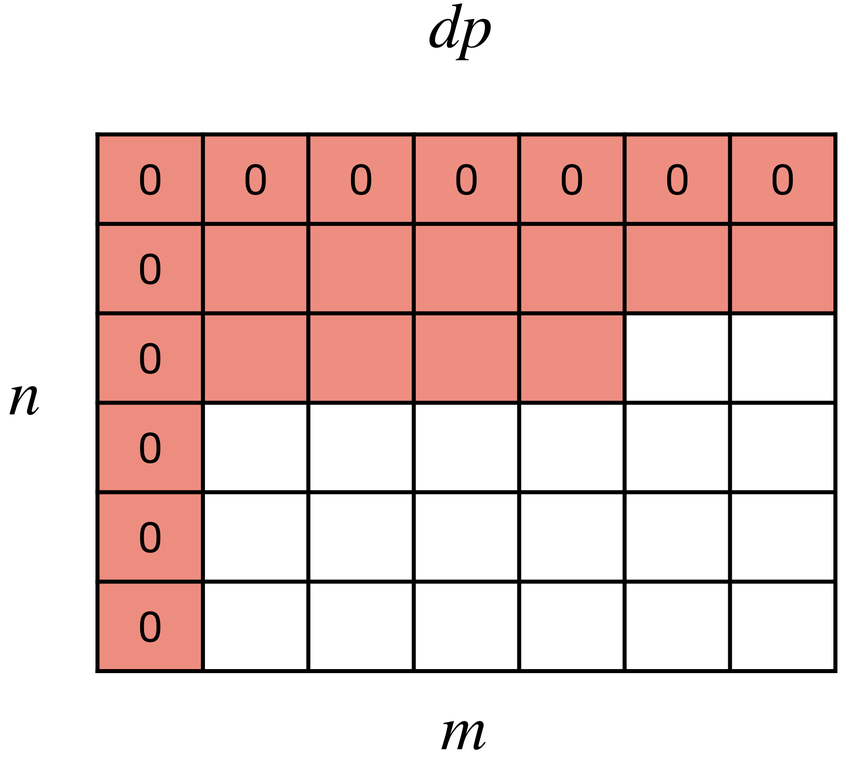
```
LCS(X, Y)
1.  dp[0 : n][0 : m] = {−1, − 1,..., − 1}
2.  dp[0][0 : m] = 0, dp[0 : n][0] = 0
3.  for i = 1 to n
4.      for j = 1 to m
5.          if x_i == y_j
6.              dp[i][j] = dp[i − 1][j − 1] + 1
7.          else
8.              dp[i][j] = Max(dp[i − 1][j], dp[i][j − 1])
9.  return dp[n][m]
```



Bottom-Up DP for LCS

LCS(X, Y)

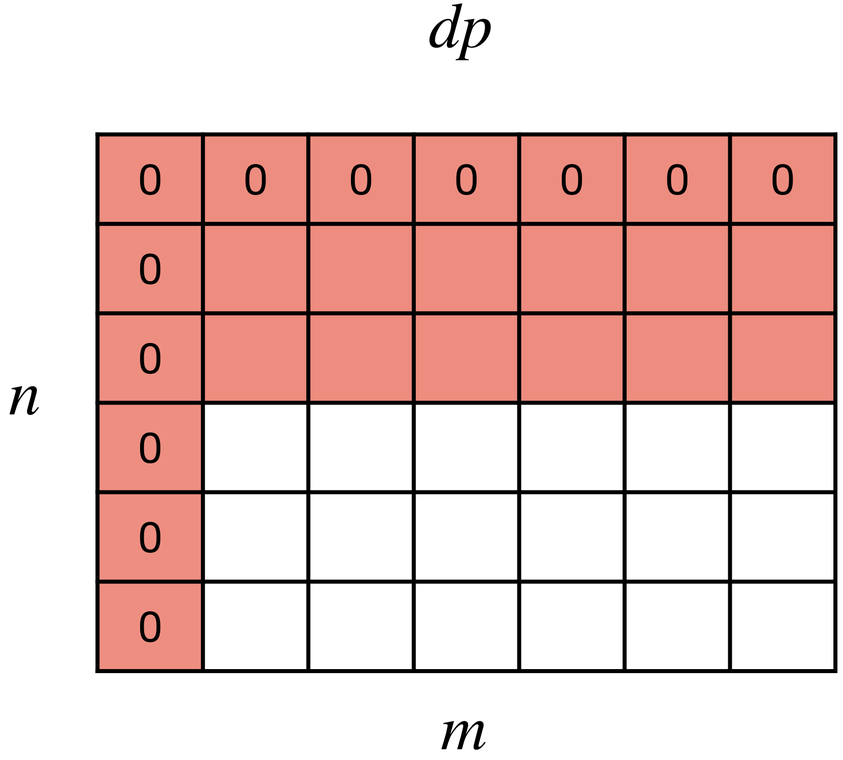
1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0$
3. **for** $i = 1$ **to** n
4. **for** $j = 1$ **to** m
5. **if** $x_i == y_j$
6. $dp[i][j] = dp[i-1][j-1] + 1$
7. **else**
8. $dp[i][j] = \textbf{Max}(dp[i-1][j], dp[i][j-1])$
9. **return** $dp[n][m]$



Bottom-Up DP for LCS

LCS(X, Y)

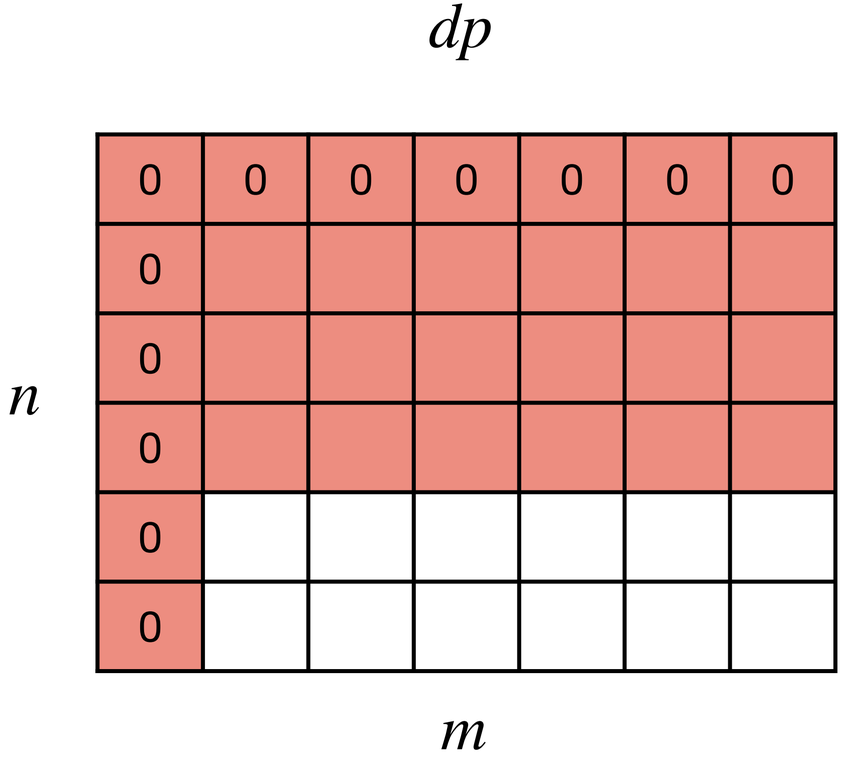
1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0$
3. **for** $i = 1$ **to** n
4. **for** $j = 1$ **to** m
5. **if** $x_i == y_j$
6. $dp[i][j] = dp[i-1][j-1] + 1$
7. **else**
8. $dp[i][j] = \textbf{Max}(dp[i-1][j], dp[i][j-1])$
9. **return** $dp[n][m]$



Bottom-Up DP for LCS

LCS(X, Y)

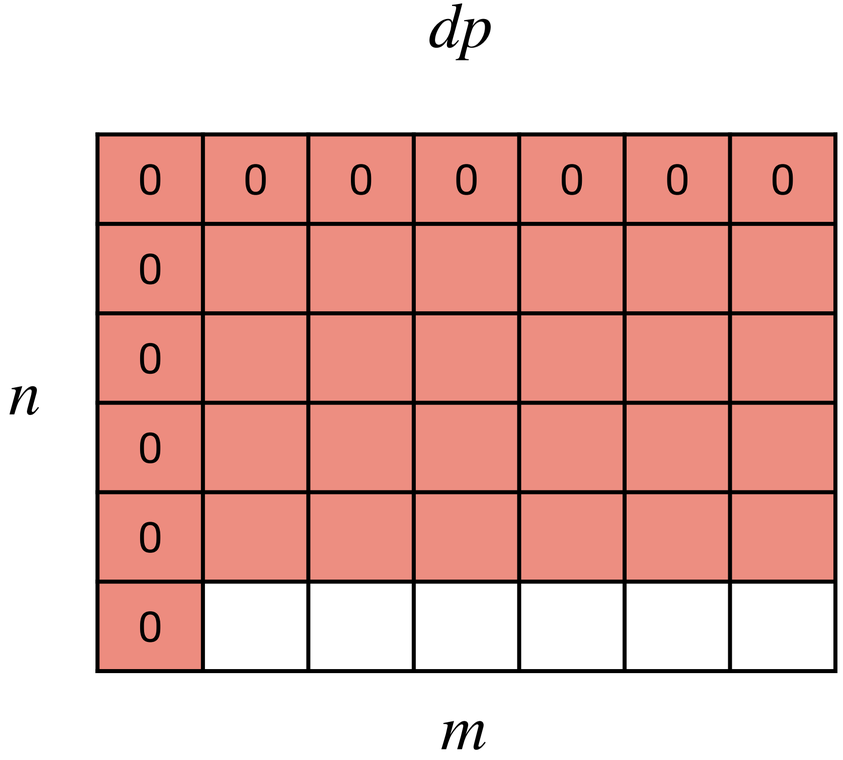
1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0$
3. **for** $i = 1$ **to** n
4. **for** $j = 1$ **to** m
5. **if** $x_i == y_j$
6. $dp[i][j] = dp[i-1][j-1] + 1$
7. **else**
8. $dp[i][j] = \textbf{Max}(dp[i-1][j], dp[i][j-1])$
9. **return** $dp[n][m]$



Bottom-Up DP for LCS

LCS(X, Y)

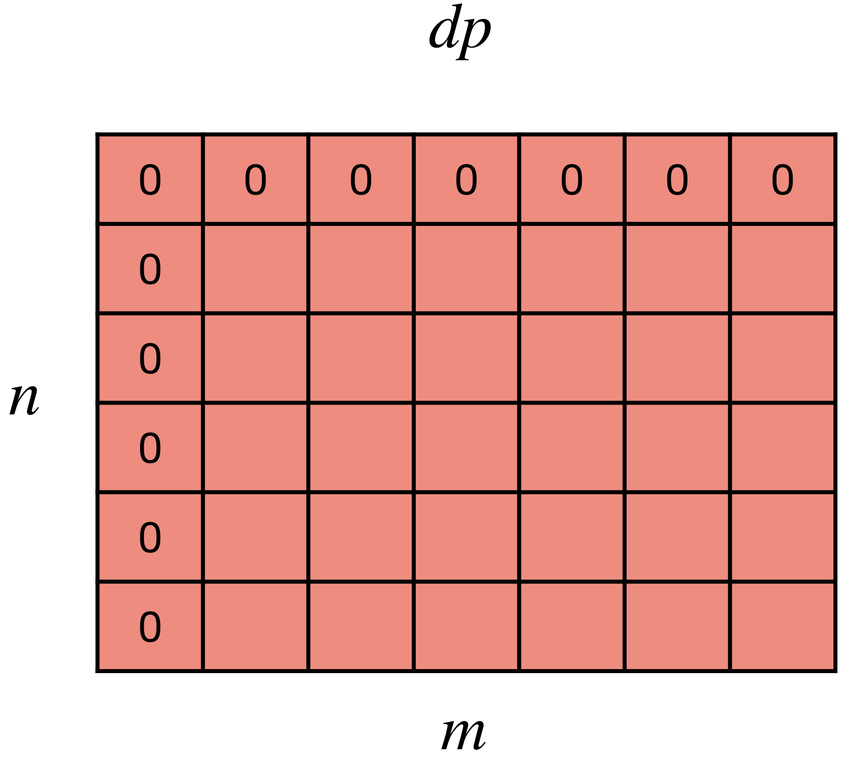
1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0$
3. **for** $i = 1$ **to** n
4. **for** $j = 1$ **to** m
5. **if** $x_i == y_j$
6. $dp[i][j] = dp[i-1][j-1] + 1$
7. **else**
8. $dp[i][j] = \textbf{Max}(dp[i-1][j], dp[i][j-1])$
9. **return** $dp[n][m]$



Bottom-Up DP for LCS

LCS(X, Y)

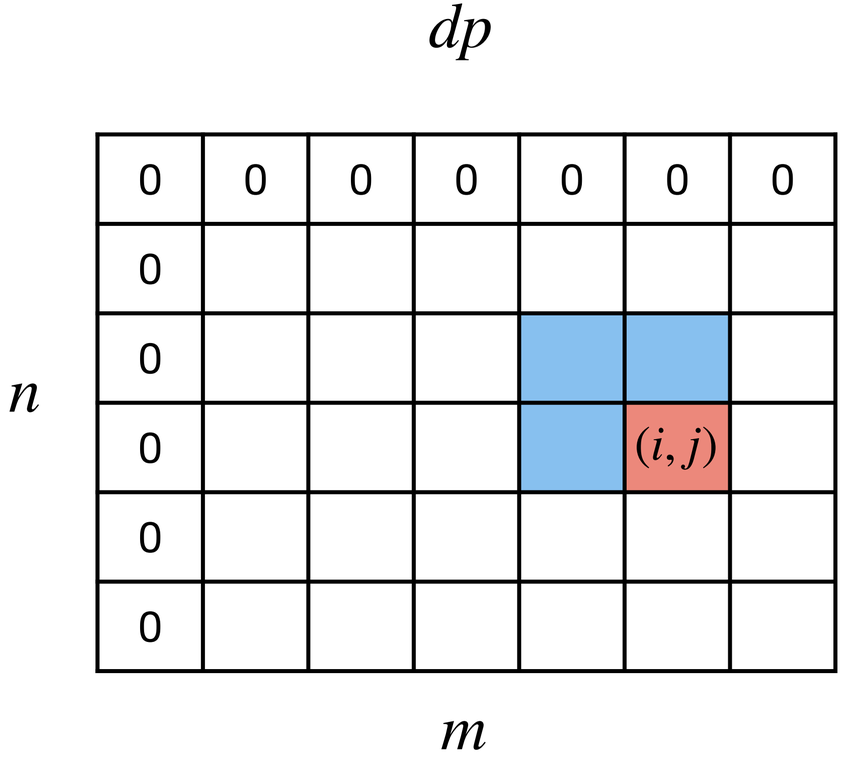
1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0$
3. **for** $i = 1$ **to** n
4. **for** $j = 1$ **to** m
5. **if** $x_i == y_j$
6. $dp[i][j] = dp[i-1][j-1] + 1$
7. **else**
8. $dp[i][j] = \textbf{Max}(dp[i-1][j], dp[i][j-1])$
9. **return** $dp[n][m]$



Bottom-Up DP for LCS

LCS(X, Y)

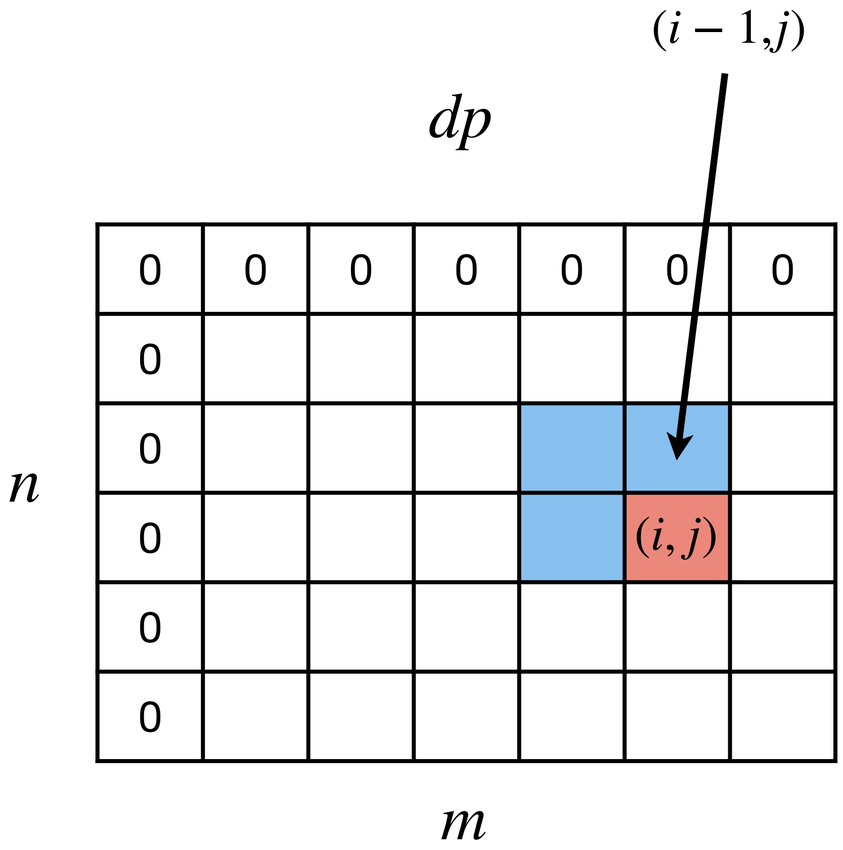
1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0$
3. **for** $i = 1$ **to** n
4. **for** $j = 1$ **to** m
5. **if** $x_i == y_j$
6. $dp[i][j] = dp[i-1][j-1] + 1$
7. **else**
8. $dp[i][j] = \textbf{Max}(dp[i-1][j], dp[i][j-1])$
9. **return** $dp[n][m]$



Bottom-Up DP for LCS

LCS(X, Y)

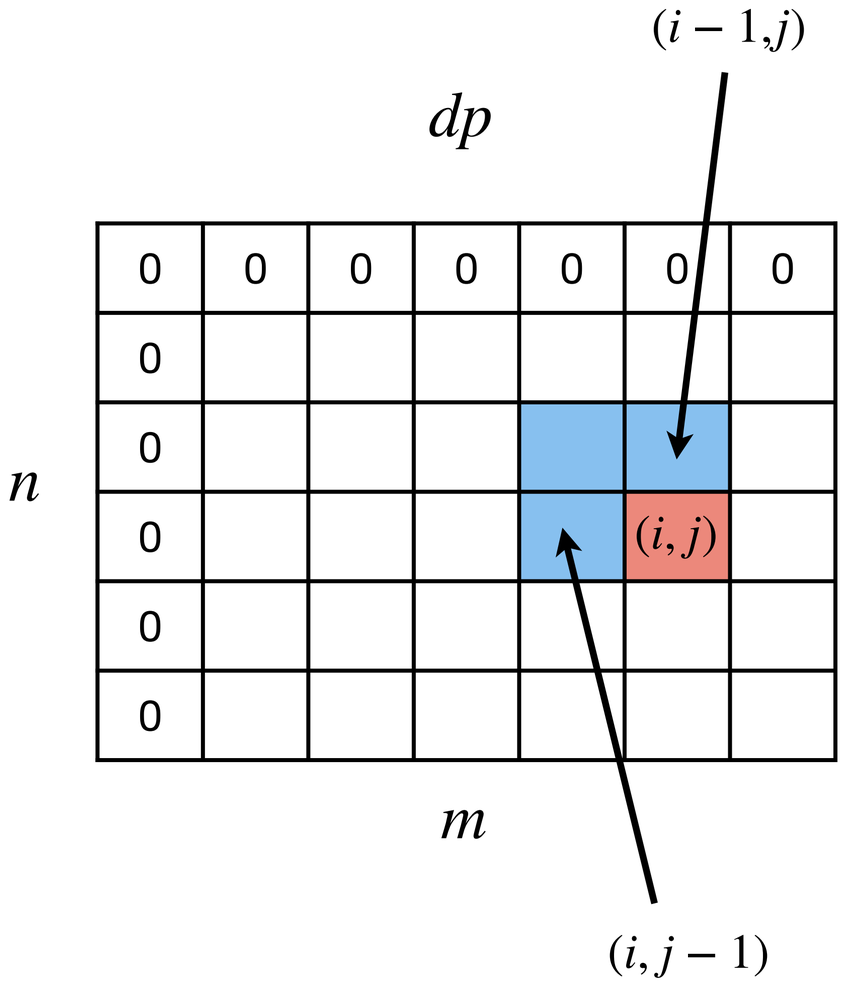
1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0$
3. **for** $i = 1$ **to** n
4. **for** $j = 1$ **to** m
5. **if** $x_i == y_j$
6. $dp[i][j] = dp[i-1][j-1] + 1$
7. **else**
8. $dp[i][j] = \textbf{Max}(dp[i-1][j], dp[i][j-1])$
9. **return** $dp[n][m]$



Bottom-Up DP for LCS

LCS(X, Y)

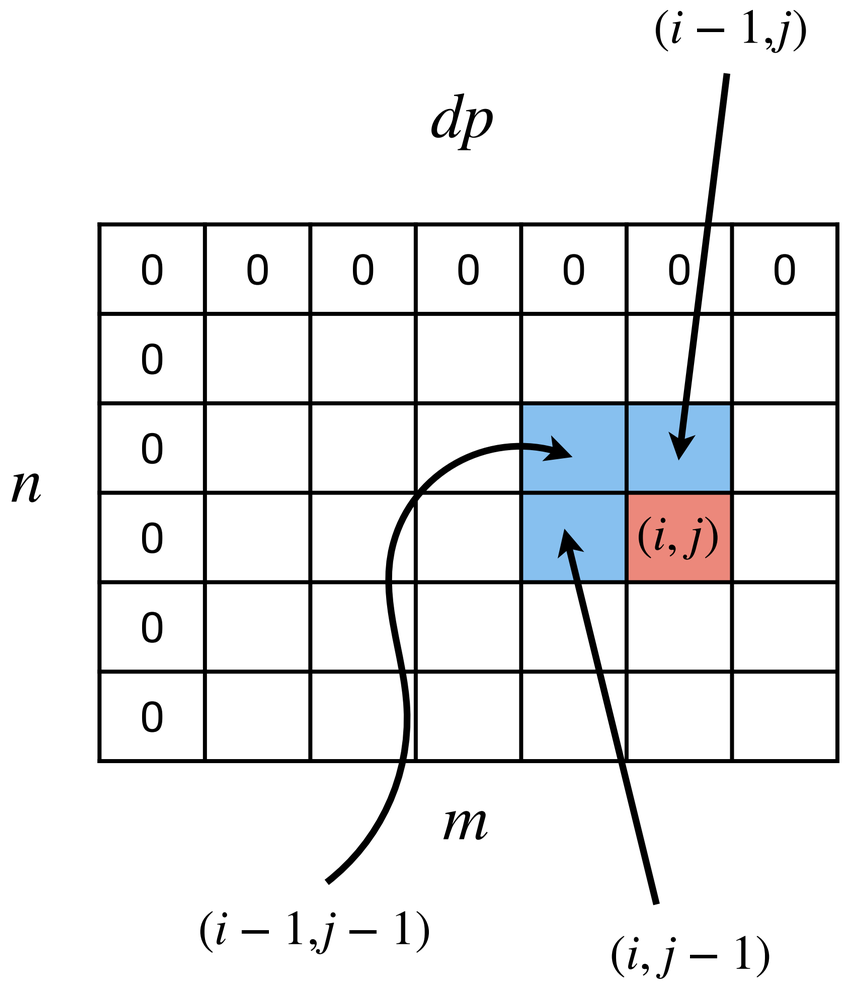
1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0,\ dp[0:n][0] = 0$
3. **for** $i = 1$ **to** n
4. **for** $j = 1$ **to** m
5. **if** $x_i == y_j$
6. $dp[i][j] = dp[i-1][j-1] + 1$
7. **else**
8. $dp[i][j] = \textbf{Max}(dp[i-1][j], dp[i][j-1])$
9. **return** $dp[n][m]$



Bottom-Up DP for LCS

LCS(X, Y)

1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0,\ dp[0:n][0] = 0$
3. **for** $i = 1$ **to** n
4. **for** $j = 1$ **to** m
5. **if** $x_i == y_j$
6. $dp[i][j] = dp[i-1][j-1] + 1$
7. **else**
8. $dp[i][j] = \textbf{Max}(dp[i-1][j], dp[i][j-1])$
9. **return** $dp[n][m]$



Bottom-Up DP for LCS

LCS(X, Y)

1. $dp[0:n][0:m] = \{-1, -1, \ldots, -1\}$
2. $dp[0][0:m] = 0, dp[0:n][0] = 0.$
3. **for** $i = 1$ **to** n
4. **for** $j = 1$ **to** m
5. **if** $x_i == y_j$
6. $dp[i][j] = dp[i-1][j-1] + 1$
7. **else**
8. $dp[i][j] = \textbf{Max}(dp[i-1][j], dp[i][j-1])$
9. **return** $dp[n][m]$

Time Complexity: $\Theta(nm)$

Bottom-Up vs Top-Down DP

Bottom-Up vs Top-Down DP

Let $X =$ "aaa…a" and $Y =$ "aaa…a" be two sequences of n many as.

Bottom-Up vs Top-Down DP

Let $X =$ "aaa…a" and $Y =$ "aaa…a" be two sequences of n many as.

How will top-down LCS run on X and Y?

Bottom-Up vs Top-Down DP

Let $X =$ "aaa…a" and $Y =$ "aaa…a" be two sequences of n many as.

How will top-down LCS run on X and Y?

$$LCS(n, n)$$

Bottom-Up vs Top-Down DP

Let $X =$ "aaa…a" and $Y =$ "aaa…a" be two sequences of n many as.

How will top-down LCS run on X and Y?

$$LCS(n, n) \longrightarrow LCS(n-1, n-1)$$

Bottom-Up vs Top-Down DP

Let $X =$ "aaa…a" and $Y =$ "aaa…a" be two sequences of n many as.

How will top-down LCS run on X and Y?

$$LCS(n, n) \longrightarrow LCS(n-1, n-1) \longrightarrow LCS(n-2, n-2)$$

Bottom-Up vs Top-Down DP

Let $X =$ "aaa…a" and $Y =$ "aaa…a" be two sequences of n many as.

How will top-down LCS run on X and Y?

$$LCS(n,n) \longrightarrow LCS(n-1,n-1) \longrightarrow LCS(n-2,n-2) \longrightarrow \cdots \longrightarrow LCS(0,0)$$

Bottom-Up vs Top-Down DP

Let $X =$ "aaa…a" and $Y =$ "aaa…a" be two sequences of n many as.

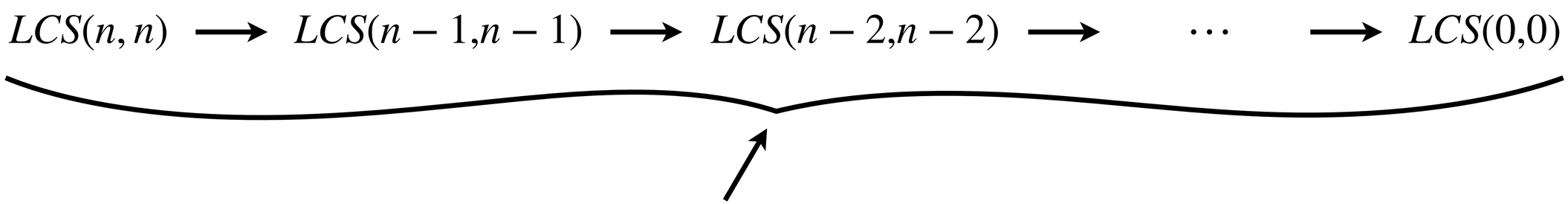
How will top-down LCS run on X and Y?

$$LCS(n,n) \longrightarrow LCS(n-1,n-1) \longrightarrow LCS(n-2,n-2) \longrightarrow \cdots \longrightarrow LCS(0,0)$$

Bottom-Up vs Top-Down DP

Let $X =$ "aaa…a" and $Y =$ "aaa…a" be two sequences of n many as.

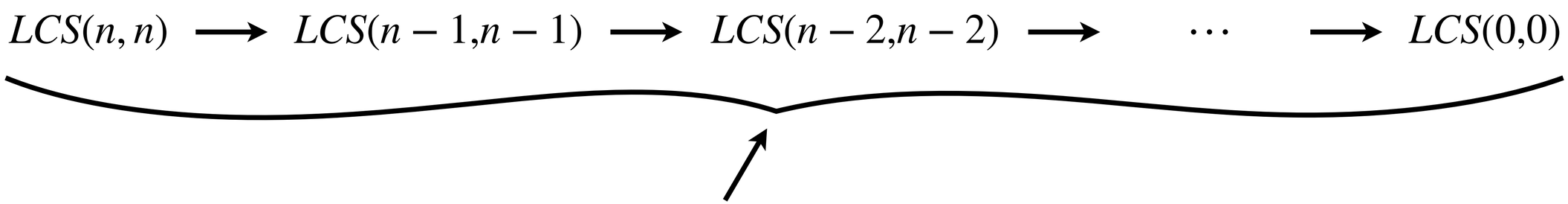
How will top-down LCS run on X and Y?



Bottom-Up vs Top-Down DP

Let $X =$ "aaa…a" and $Y =$ "aaa…a" be two sequences of n many as.

How will top-down LCS run on X and Y?



Top-down LCS requires n many recursive calls. Hence, runtime would be $\Theta(n)$.

Bottom-Up vs Top-Down DP

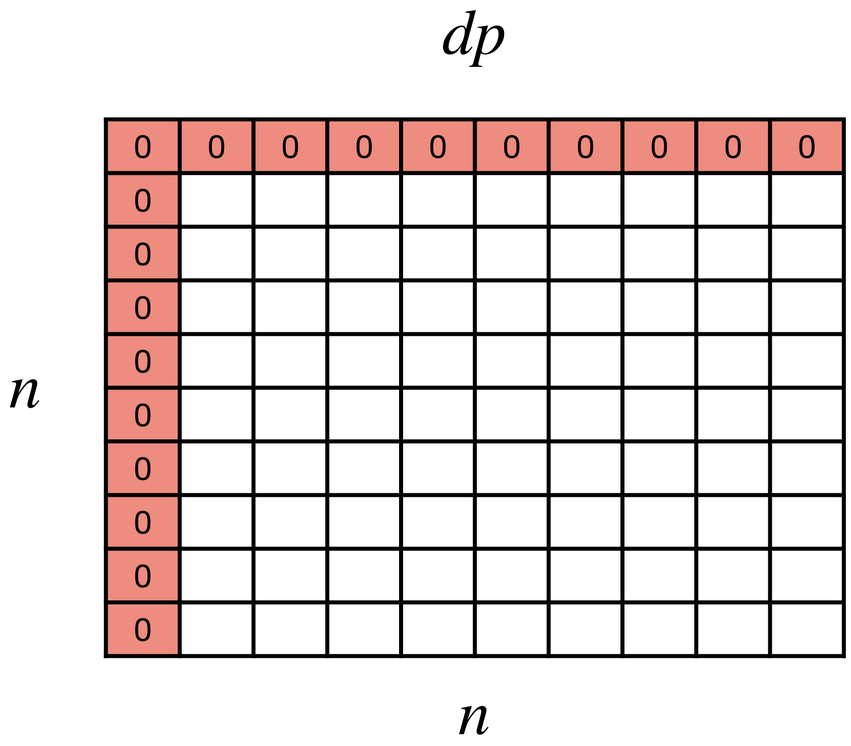
Let $X =$ "aaa…a" and $Y =$ "aaa…a" be two sequences of n many as.

How will bottom-up LCS run on X and Y?

Bottom-Up vs Top-Down DP

Let $X =$ "aaa…a" and $Y =$ "aaa…a" be two sequences of n many as.

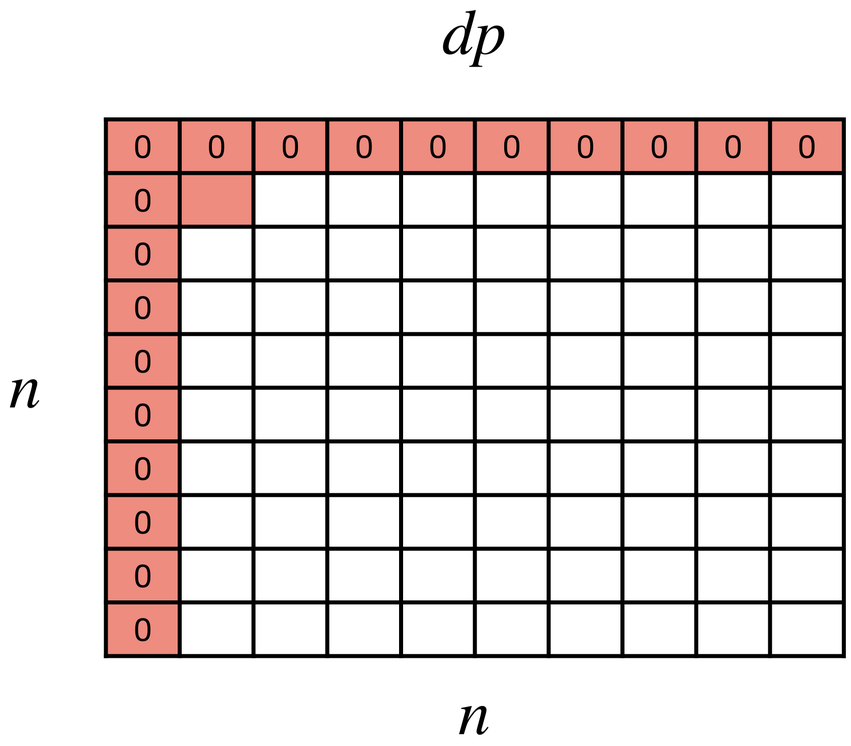
How will bottom-up LCS run on X and Y?



Bottom-Up vs Top-Down DP

Let $X =$ "aaa…a" and $Y =$ "aaa…a" be two sequences of n many as.

How will bottom-up LCS run on X and Y?

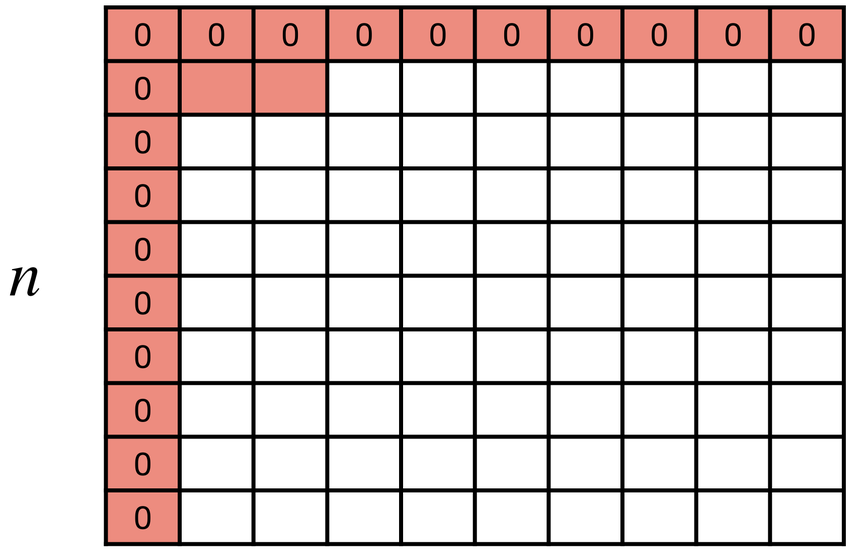


Bottom-Up vs Top-Down DP

Let $X =$ "aaa…a" and $Y =$ "aaa…a" be two sequences of n many as.

How will bottom-up LCS run on X and Y?

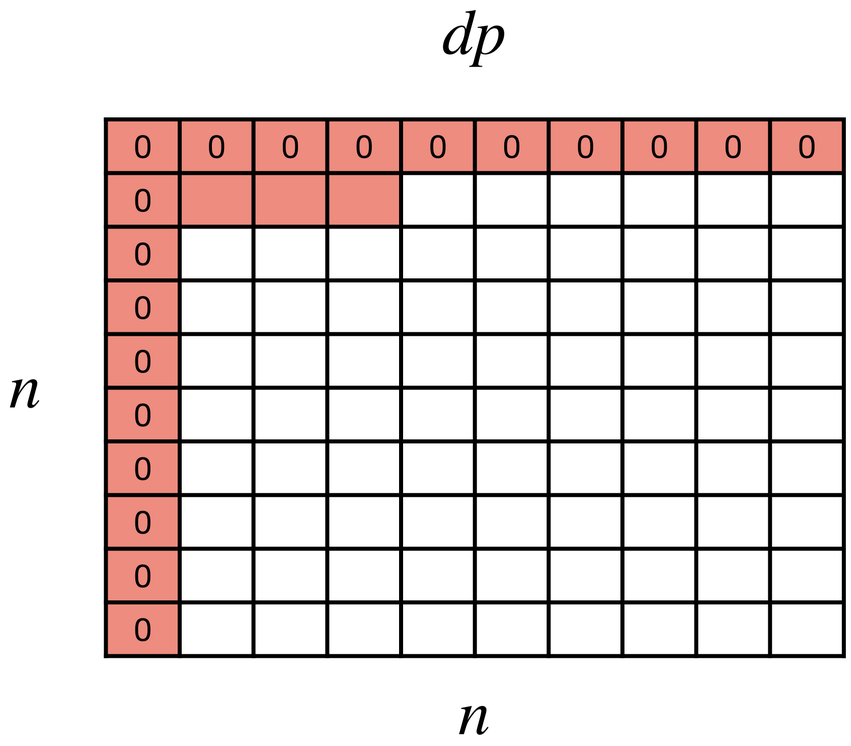
dp



Bottom-Up vs Top-Down DP

Let $X =$ "aaa…a" and $Y =$ "aaa…a" be two sequences of n many as.

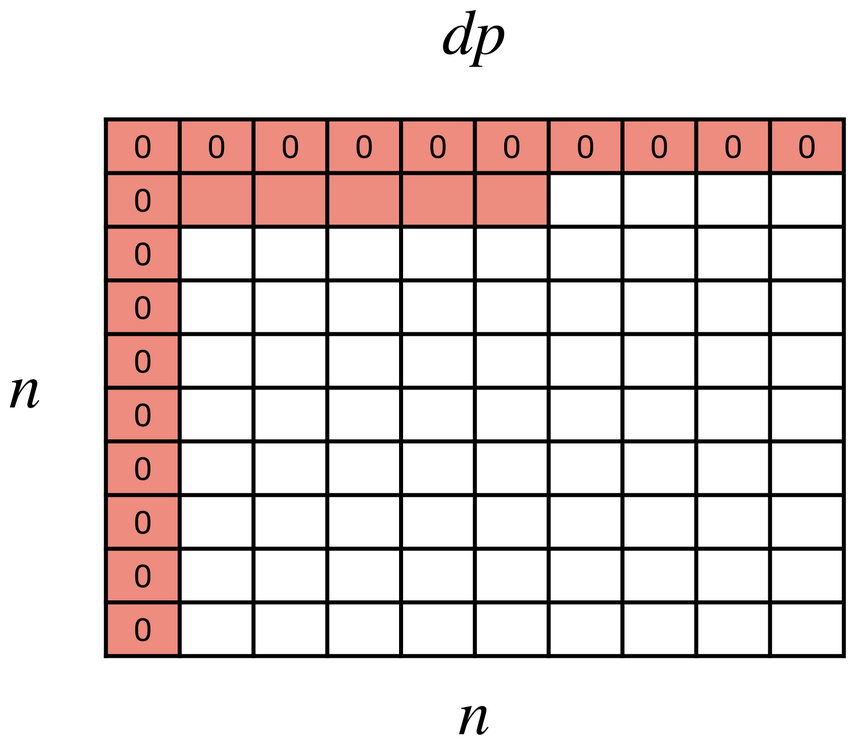
How will bottom-up LCS run on X and Y?



Bottom-Up vs Top-Down DP

Let $X =$ "aaa…a" and $Y =$ "aaa…a" be two sequences of n many as.

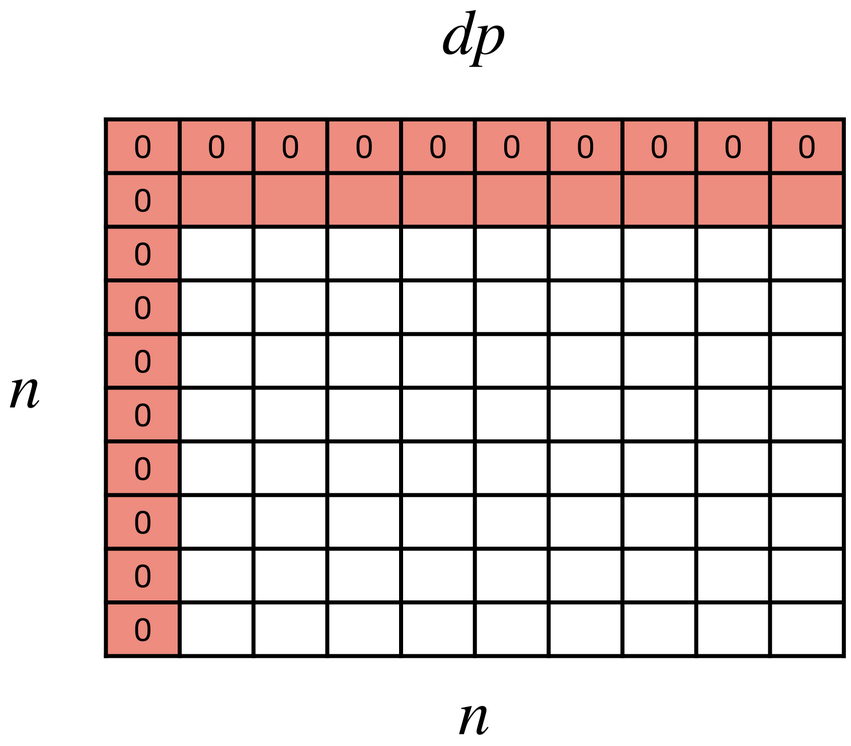
How will bottom-up LCS run on X and Y?



Bottom-Up vs Top-Down DP

Let $X =$ "aaa…a" and $Y =$ "aaa…a" be two sequences of n many as.

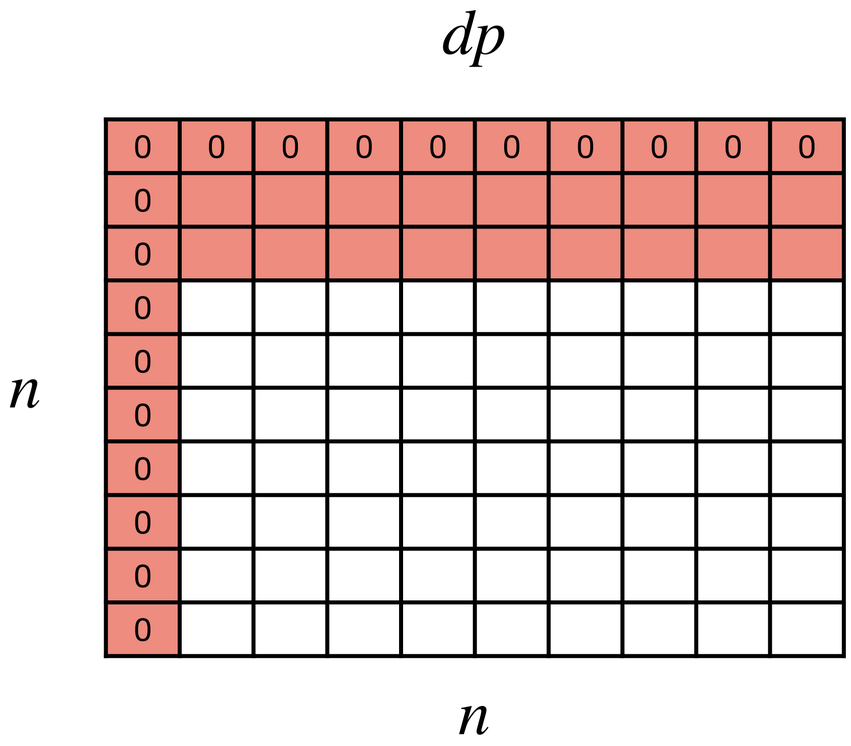
How will bottom-up LCS run on X and Y?



Bottom-Up vs Top-Down DP

Let $X =$ "aaa…a" and $Y =$ "aaa…a" be two sequences of n many as.

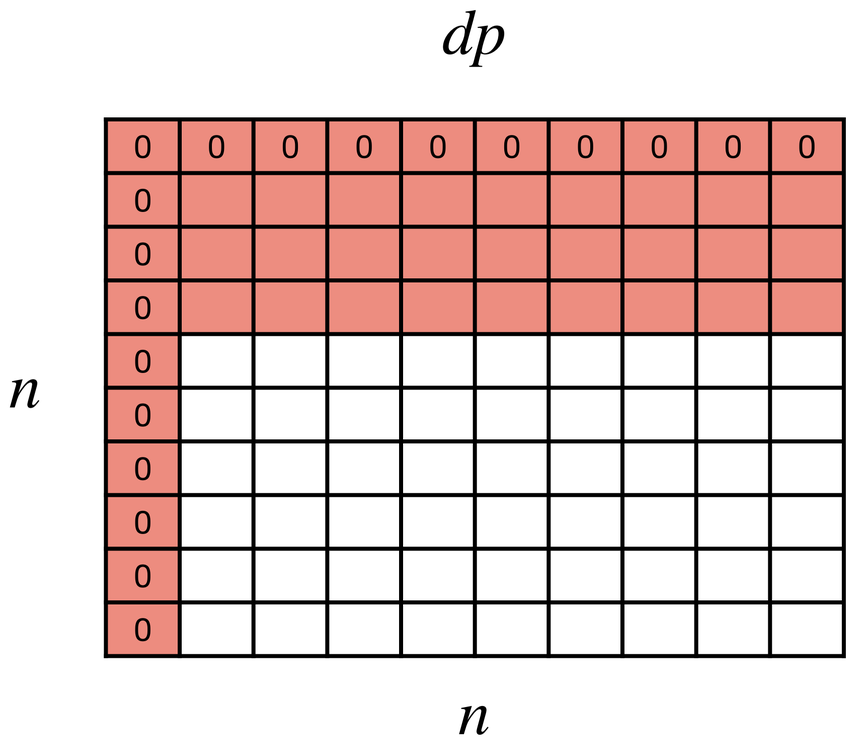
How will bottom-up LCS run on X and Y?



Bottom-Up vs Top-Down DP

Let $X =$ "aaa…a" and $Y =$ "aaa…a" be two sequences of n many as.

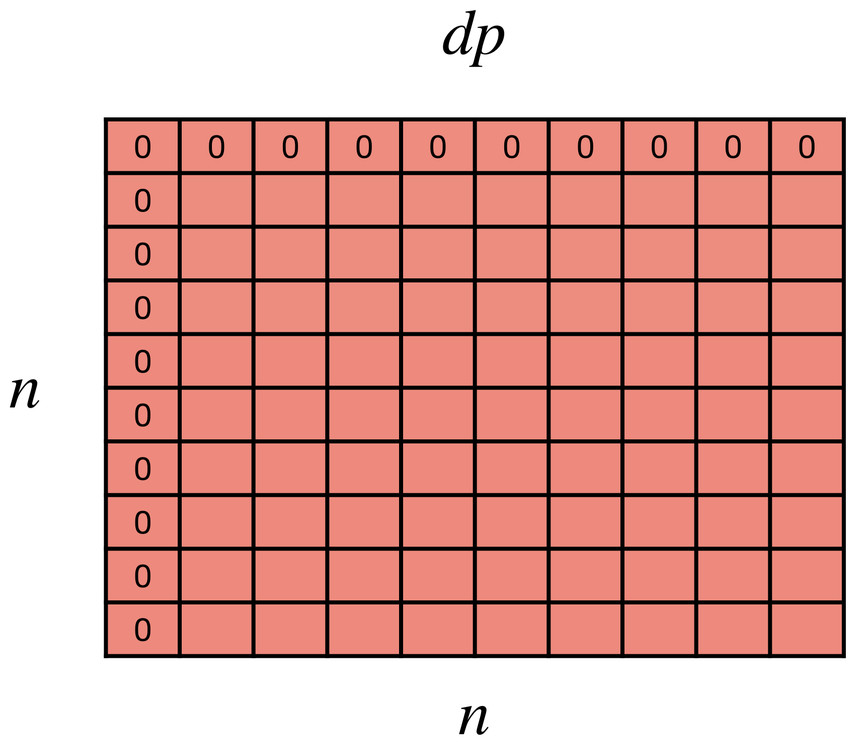
How will bottom-up LCS run on X and Y?



Bottom-Up vs Top-Down DP

Let $X =$ "aaa…a" and $Y =$ "aaa…a" be two sequences of n many as.

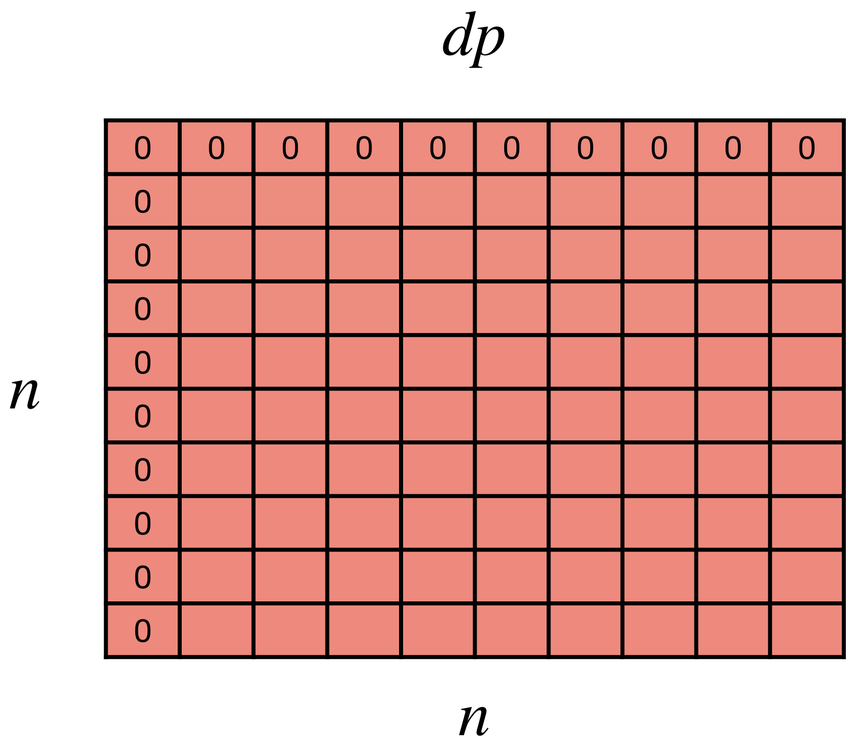
How will bottom-up LCS run on X and Y?



Bottom-Up vs Top-Down DP

Let $X =$ "aaa…a" and $Y =$ "aaa…a" be two sequences of n many as.

How will bottom-up LCS run on X and Y?



Bottom-up LCS will still fill all $n * n$ entries of dp array in $\Theta(n^2)$ time.

Bottom-Up vs Top-Down DP

Bottom-Up vs Top-Down DP

- Top-down solves only those subproblems which are necessary.

Bottom-Up vs Top-Down DP

- Top-down solves only those subproblems which are necessary.
- Bottom-up solves all the subproblems.

Bottom-Up vs Top-Down DP

- Top-down solves only those subproblems which are necessary.
- Bottom-up solves all the subproblems.
- If all subproblems need to be solved, then bottom-up is better due to no recursion overhead.

Bottom-Up vs Top-Down DP

- Top-down solves only those subproblems which are necessary.
- Bottom-up solves all the subproblems.
- If all subproblems need to be solved, then bottom-up is better due to no recursion overhead.
- Otherwise, top-down might be better.