Longest common subsequence (LCS):

Input: strings \( X = x_1 x_2 \ldots x_n \) & \( Y = y_1 y_2 \ldots y_m \)

Goal: length of the longest string that is a subsequence in both \( X \) & \( Y \).

Example: \( X = B C D B C D A \)

\( Y = A B E C B A \)

answer = 4 for BCBA

Application: used in Unix diff for comparing files.

First step: define subproblem.

Look at prefixes, and in this case it's natural to use a 2-dimensional table.

For \( 0 \leq i \leq n \& 0 \leq j \leq m \) let

\( L(i, j) = \text{length of LCS of } X_i \ldots X_i \text{ with } Y_j \ldots Y_j \)
Base cases: \( L(i,0) = 0 \)
\( L(0,j) = 0 \)

For recurrence, two cases: \( X_i = Y_j \) & \( X_i \neq Y_j \)

Suppose \( X_i \neq Y_j \):

either \( X_i \) is not included in the common string
and/or \( Y_j \) is not included.

So we can drop \( X_i \) or \( Y_j \).

If \( X_i \) is dropped then
\( L(i,j) = L(i-1,j) \)

If \( Y_j \) is dropped then
\( L(i,j) = L(i,j-1) \)

Therefore, if \( X_i \neq Y_j \) then:
\( L(i,j) = \max\{ L(i,j-1), L(i-1,j) \} \)
Suppose \( X_i = Y_j \):

Either the LCS ends with \( X_i = Y_j \),
in which case \( L(i,j) = 1 + L(i-1,j-1) \)
or the LCS does not include \( X_i = Y_j \)
but it can always be added to make it longer, so it must end with \( X_i = Y_j \).

Therefore, we have:

\[
L(i,j) = 1 + L(i-1,j-1)
\]

In general,

\[
L(i,j) = \begin{cases} 
0 & \text{if } i = 0 \text{ or } j = 0 \\
1 + L(i-1,j-1) & \text{if } X_i = Y_j \\
\max[L(i-1,j), L(i,j-1)] & \text{if } X_i \neq Y_j 
\end{cases}
\]

To fill the table \( L \) row-by-row:

To get \( L(i,j) \) look at 3 previous neighbors:

\( L(i-1,j), L(i-1,j-1), L(i,j-1) \).
LCS(X, Y):

for i = 0 \rightarrow n, \quad L(i, 0) = 0

for j = 0 \rightarrow m, \quad L(0, j) = 0

for i = 1 \rightarrow n

\quad for j = 1 \rightarrow m

\quad if X_i = Y_j

\quad \quad then \quad L(i, j) = 1 + L(i-1, j-1)

\quad else \quad L(i, j) = \max\{L(i-1, j), L(i, j-1)\}

return \(L(n, m)\)

Running time: \(O(nm)\)
Earlier example:

\[ X = BCDBCDA \]
\[ Y = ABECBA \]

\[ L = \]

\[
\begin{array}{ccccc}
A & B & E & C & B & A \\
\phi & 0 & 0 & 0 & 0 & 0 \\
B & 0 & 0 & 1 & 1 & 1 \\
C & 0 & 0 & 1 & 2 & 2 \\
D & 0 & 0 & 1 & 2 & 2 \\
E & 0 & 0 & 1 & 2 & 2 \\
\end{array}
\]
Closely related:

**Edit distance:**

input: strings \( X = x_1 \ldots x_n \) & \( Y = y_1 \ldots y_m \)

Goal: Min # of edits (insertion, deletion, substitution)

to go between \( X \) & \( Y \)

Example:

\( X = \text{AAGCTG} \text{CCTAA} \)

\( Y = \text{AACCGCAATA} \)

\[
\begin{align*}
\text{AAGCTG} & \text{CCTAA} \\
\text{AACCGCAATA} & \\
\end{align*}
\]

edit distance = 5

Variant used in BLAST - used in Biology to align DNA sequences
In Biology add a 5x5 scoring matrix:

\[
\delta = \begin{bmatrix}
A & G & C & T \\
A & - & - & - \\
G & - & - & - \\
C & - & - & - \\
T & - & - & - \\
\end{bmatrix}
\]

\(- \) = gap

for a mismatch, i.e., when \( X_i \neq Y_j \) the cost of aligning them is \( \delta(X_i, Y_j) \)
Manhattan Tourist Problem:

For an $n \times n$ grid, find the max weight path from $(0,0)$ to $(n,n)$—but only moving N & E on the path.

Input:

$N(i,j) =$ weight of edge from $(i,j)$ to $(i,j+1)$

$E(i,j) =$ weight of edge from $(i,j)$ to $(i+1,j)$

Weight of a path = sum of edge weights

Assume all edge weights are $\geq 0.$
Example:

Let

\[ L(i,j) = \max \text{ weight for a valid path from } (0,0) \text{ to } (i,j) \]

For \( i > 0, j > 0 \),

\[ L(i,j) = \max \left\{ L(i-1,j) + E(i-1,j), \quad L(i,j-1) + N(i,j-1) \right\} \]

For \( i = 0 \), \( j > 0 \),

\[ L(0,j) = L(0,j-1) + N(0,j-1) \]

For \( i > 0, j = 0 \),

\[ L(i,0) = L(i-1,0) + E(i-1,0) \]

\[ L(0,0) = 0 \]
MTP:

\[ f(0,0) = 0 \]

for \( i = 1 \rightarrow n \), \( L(i,0) = L(i-1,0) + E(i-1,0) \)

for \( j = 1 \rightarrow n \), \( L(0,j) = L(0,j-1) + N(0,j-1) \)

for \( i = 1 \rightarrow n \)

for \( j = 1 \rightarrow n \)

\[ L(i,j) = \max \left( L(i-1,j) + E(i-1,j), L(i,j-1) + N(i,j-1) \right) \]

Return \( L(n,n) \)

Running time: \( O(n^2) \)