CSE 202: Design and Analysis of Algorithms

Lecture 6

Instructor: Kamalika Chaudhuri
Announcements

• Homework due in lecture today!
• My office hours moving to Room 4110
Last Class: Dynamic Programming

- String Reconstruction
- Longest Common Subsequence
Three steps of Dynamic Programming

Main Steps:

1. Divide the problem into **subtasks**
2. Define the subtasks **recursively** (express larger subtasks in terms of smaller ones)
3. Find the **right order** for solving the subtasks (but do not solve them recursively!)
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**Example:**

$x = A, C, G, T, A, G$
$y = G, T, C, C, A, C$

$LCS(x, y) = G, T, A$
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**Example:**

$x = A,C,G,T,A,G$

$y = G,T,C,C,A,C$

$LCS(x, y) = G,T,A$

**Structure:**

$x = A,C,G,T$

$y = G,T$
Longest Common Subsequence (LCS)

Problem: Given two sequences \(x[1..m]\) and \(y[1..n]\), find their longest common subsequence.

Example:

\[x = A,C,G,T,A,G\]
\[y = G,T,C,C,A,C\]

\(LCS(x, y) = G,T,A\)

Structure:

\[x = A,C,G, T,\]
\[y = G, T,\]

If \(x[i] = y[j]\), then

\[LCS(x[1..i], y[1..j]) = LCS(x[1..i-1], y[1..j-1]) + x[i]\]
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**Example:**

$x = A,C,G,T,A,G$
$y = G,T,C,C,A,C$

$LCS(x, y) = G,T,A$

**Structure:**

If $x[i] = y[j]$, then

$LCS(x[1..i], y[1..j]) = LCS(x[1..i-1], y[1..j-1]) + x[i]$
Longest Common Subsequence (LCS)

**Problem:** Given two sequences \( x[1..m] \) and \( y[1..n] \), find their longest common subsequence

**Example:**

\[
\begin{align*}
x &= A, C, G, T, A, G \\
y &= G, T, C, C, A, C \\
\text{LCS}(x, y) &= G, T, A
\end{align*}
\]

**Structure:**

\[
\begin{align*}
x &= A, C, G, T, A \\
y &= G, T, \\
\text{If } x[i] = y[j], \text{ then } \\
\text{LCS}(x[1..i], y[1..j]) &= \text{LCS}(x[1..i-1], y[1..j-1]) + x[i] \\
x &= A, C, G, T, A \\
y &= G, T, \\
\text{Otherwise,} \\
\text{LCS}(x[1..i], y[1..j]) &= \max(\text{LCS}(x[1..i-1], y[1..j]), \text{LCS}(x[1..i], y[1..j-1]))
\end{align*}
\]
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**Example:**

$x = A, C, G, T, A, G$
$y = G, T, C, C, A, C$

$LCS(x, y) = G, T, A$

**STEP 1: Define subtasks**

$S(i,j) =$ Length of LCS of $x[1..i]$ and $y[1..j]$

Output of algorithm = $S(m,n)$
Longest Common Subsequence (LCS)

Problem: Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**Example:**

$x = A, C, G, T, A, G$
$y = G, T, C, C, A, C$

$LCS(x, y) = G, T, A$

**STEP 1: Define subtasks**

$S(i,j) = \text{Length of LCS of } x[1..i] \text{ and } y[1..j]$

Output of algorithm = $S(m,n)$

**STEP 2: Express recursively**

$S(i,j) = S(i-1,j-1) + 1,$ if $x[i] = y[j]$

$= \max(S(i-1,j), S(i,j-1)),$ ow
Longest Common Subsequence (LCS)

**Problem:** Given two sequences \(x[1..m]\) and \(y[1..n]\), find their longest common subsequence

**Example:**

\(x = A, C, G, T, A, G\)
\(y = G, T, C, C, A, C\)

\(\text{LCS}(x, y) = G, T, A\)

**STEP 1: Define subtasks**

\(S(i,j) = \text{Length of LCS of } x[1..i]\)
\(\text{and } y[1..j]\)

Output of algorithm = \(S(m,n)\)

**STEP 2: Express recursively**

\(S(i,j) = S(i-1,j-1) + 1, \text{ if } x[i] = y[j]\)
\(= \max(S(i-1,j), S(i,j-1)), \text{ ow}\)

**STEP 3: Order of subtasks**

Row by row, left to right
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**Example:**

$x = A, C, G, T, A, G$

$y = G, T, C, C, A, C$

$LCS(x, y) = G, T, A$

**STEP 1: Define subtasks**

$S(i,j) =$ Length of LCS of $x[1..i]$ and $y[1..j]$

Output of algorithm = $S(m,n)$

**STEP 2: Express recursively**

$S(i,j) = S(i-1,j-1) + 1$, if $x[i] = y[j]$

$= \max(S(i-1,j), S(i,j-1))$, otherwise

**STEP 3: Order of subtasks**

Row by row, left to right

---

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:** Row 0, Column 0
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**Example:**

$x = A,C,G,T,A,G$

$y = G,T,C,C,A,C$

$LCS(x, y) = G,T,A$

**STEP 1: Define subtasks**

$S(i,j) =$ Length of LCS of $x[1..i]$ and $y[1..j]$

Output of algorithm = $S(m,n)$

**STEP 2: Express recursively**

$S(i,j) = S(i-1,j-1) + 1$, if $x[i] = y[j]$

$= \max(S(i-1,j), S(i,j-1))$, ow

**STEP 3: Order of subtasks**

Row by row, left to right

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:** Row 0, Column 0
Longest Common Subsequence (LCS)

**Problem:** Given two sequences \(x[1..m]\) and \(y[1..n]\), find their longest common subsequence

**Example:**

\[ x = A,C,G,T,A,G \]
\[ y = G,T,C,C,A,C \]

\[ \text{LCS}(x, y) = G,T,A \]

**STEP 1: Define subtasks**

\[ S(i,j) = \text{Length of LCS of } x[1..i] \text{ and } y[1..j] \]

Output of algorithm = \(S(m,n)\)

**STEP 2: Express recursively**

\[ S(i,j) = S(i-1,j-1) + 1, \text{ if } x[i] = y[j] \]
\[ = \text{max}(S(i-1,j), S(i,j-1)), \text{ ow} \]

**STEP 3: Order of subtasks**

Row by row, left to right

---

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:** Row 0, Column 0
Longest Common Subsequence (LCS)

**Problem:** Given two sequences x[1..m] and y[1..n], find their longest common subsequence

**Example:**

\[ x = A, C, G, T, A, G \]
\[ y = G, T, C, C, A, C \]

\[ \text{LCS}(x, y) = G, T, A \]

**STEP 1: Define subtasks**

\[ S(i,j) = \text{Length of LCS of } x[1..i] \]
\[ \text{and } y[1..j] \]

Output of algorithm = \( S(m,n) \)

**STEP 2: Express recursively**

\[ S(i,j) = \begin{cases} 
S(i-1,j-1) + 1, & \text{if } x[i] = y[j] \\
\max(S(i-1,j), S(i,j-1)), & \text{otherwise}
\end{cases} \]

**STEP 3: Order of subtasks**

Row by row, left to right

---

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:** Row 0, Column 0
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**Example:**

$x = A, C, G, T, A, G$

$y = G, T, C, C, A, C$

$LCS(x, y) = G, T, A$

**STEP 1: Define subtasks**

$S(i,j)$ = Length of LCS of $x[1..i]$ and $y[1..j]$

Output of algorithm = $S(m,n)$

**STEP 2: Express recursively**

$S(i,j) = S(i-1,j-1) + 1$, if $x[i] = y[j]$

$= \max(S(i-1,j), S(i,j-1))$, ow

**STEP 3: Order of subtasks**

Row by row, left to right

**Base Case:** Row 0, Column 0
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**Example:**
$x = A,C,G,T,A,G$
y = $G,T,C,C,A,C$
$LCS(x, y) = G,T,A$

**STEP 1: Define subtasks**
$S(i,j) =$ Length of LCS of $x[1..i]$ and $y[1..j]$
Output of algorithm = $S(m,n)$

**STEP 2: Express recursively**
$S(i,j) = S(i-1,j-1) + 1$, if $x[i] = y[j]$
$= \max(S(i-1,j), S(i,j-1))$, ow

**STEP 3: Order of subtasks**
Row by row, left to right

**Base Case:** Row 0, Column 0
Longest Common Subsequence (LCS)

**Problem:** Given two sequences x[1..m] and y[1..n], find their longest common subsequence

**Example:**

x = A,C,G,T,A,G
y = G,T,C,C,A,C

LCS(x, y) = G,T,A

**STEP 1: Define subtasks**

S(i,j) = Length of LCS of x[1..i] and y[1..j]

Output of algorithm = S(m,n)

**STEP 2: Express recursively**

\[
S(i,j) = S(i-1,j-1) + 1, \text{ if } x[i] = y[j]
\]

\[
= \max(S(i-1,j), S(i,j-1)), \text{ otherwise}
\]

**STEP 3: Order of subtasks**

Row by row, left to right

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:** Row 0, Column 0
Longest Common Subsequence (LCS)

**Problem:** Given two sequences x[1..m] and y[1..n], find their longest common subsequence

**Example:**

\[ x = A, C, G, T, A, G \]
\[ y = G, T, C, C, A, C \]

\[ \text{LCS}(x, y) = G, T, A \]

**STEP 1: Define subtasks**

\[ S(i, j) = \text{Length of LCS of } x[1..i] \]
\[ \text{and } y[1..j] \]

Output of algorithm = \( S(m,n) \)

**STEP 2: Express recursively**

\[ S(i, j) = S(i-1, j-1) + 1, \text{ if } x[i] = y[j] \]
\[ = \max(S(i-1, j), S(i, j-1)), \text{ otherwise} \]

**STEP 3: Order of subtasks**

Row by row, left to right

**Base Case:** Row 0, Column 0

\[
\begin{array}{ccccccccc}
A & C & T & G & G & C & T & A & G \\
\hline
G & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
T & 2 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
G & 3 & & & & & & & \\
A & 4 & & & & & & & \\
C & 5 & & & & & & & \\
A & 6 & & & & & & & \\
G & 7 & & & & & & & \\
T & 8 & & & & & & & \\
T & 9 & & & & & & & \\
\end{array}
\]
**Longest Common Subsequence (LCS)**

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence.

**Example:**

$x = A,C,G,T,A,G$
$y = G,T,C,C,A,C$

$LCS(x, y) = G,T,A$

**STEP 1: Define subtasks**

$S(i,j) = \text{Length of LCS of } x[1..i]$

and $y[1..j]$

Output of algorithm = $S(m,n)$

**STEP 2: Express recursively**

$S(i,j) = S(i-1,j-1) + 1, \text{ if } x[i] = y[j]$

$= \max(S(i-1,j), S(i,j-1)), \text{ ow}$

**STEP 3: Order of subtasks**

Row by row, left to right

**Base Case:** Row 0, Column 0
Longest Common Subsequence (LCS)

**Problem:** Given two sequences x[1..m] and y[1..n], find their longest common subsequence

**Example:**

x = A,C,G,T,A,G  
y = G,T,C,C,A,C  
LCS(x, y) = G,T,A

**STEP 1: Define subtasks**

S(i,j) = Length of LCS of x[1..i] and y[1..j]  
Output of algorithm = S(m,n)

**STEP 2: Express recursively**

\[ S(i,j) = \begin{cases} 
S(i-1,j-1) + 1, & \text{if } x[i] = y[j] \\
\max(S(i-1,j), S(i,j-1)), & \text{otherwise} 
\end{cases} \]

**STEP 3: Order of subtasks**

Row by row, left to right

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:** Row 0, Column 0
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence.

**Example:**

$x = A, C, G, T, A, G$

$y = G, T, C, C, A, C$

$LCS(x, y) = G, T, A$

**STEP 1: Define subtasks**

$S(i,j) = \text{Length of LCS of } x[1..i]$

and $y[1..j]$

Output of algorithm $= S(m,n)$

**STEP 2: Express recursively**

$S(i,j) = S(i-1,j-1) + 1$, if $x[i] = y[j]$

$= \max(S(i-1,j), S(i,j-1))$, otherwise

**STEP 3: Order of subtasks**

Row by row, left to right

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:** Row 0, Column 0
Longest Common Subsequence (LCS)

**Problem:** Given two sequences x[1..m] and y[1..n], find their longest common subsequence

**Example:**

x = A,C,G,T,A,G  
y = G,T,C,C,A,C  
LCS(x, y) = G,T,A

**STEP 1: Define subtasks**

S(i,j) = Length of LCS of x[1..i] and y[1..j]

Output of algorithm = S(m,n)

**STEP 2: Express recursively**

S(i,j) = S(i-1,j-1) + 1, if x[i] = y[j]

= max(S(i-1,j), S(i,j-1)), o.w.

**STEP 3: Order of subtasks**

Row by row, left to right

---

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:** Row 0, Column 0
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**Example:**

$x = A,C,G,T,A,G$

$y = G,T,C,C,A,C$

$LCS(x, y) = G,T,A$

**STEP 1: Define subtasks**

$S(i,j)$ = Length of LCS of $x[1..i]$ and $y[1..j]$

Output of algorithm = $S(m,n)$

**STEP 2: Express recursively**

$S(i,j) = S(i-1,j-1) + 1$, if $x[i] = y[j]$

$= \max(S(i-1,j), S(i,j-1))$, otherwise

**STEP 3: Order of subtasks**

Row by row, left to right

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:** Row 0, Column 0
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**Example:**

$x = A, C, G, T, A, G$

$y = G, T, C, C, A, C$

$LCS(x, y) = G, T, A$

**STEP 1:** Define subtasks

$S(i, j) =$ Length of LCS of $x[1..i]$ and $y[1..j]$

Output of algorithm $= S(m, n)$

**STEP 2:** Express recursively

$S(i, j) = S(i-1, j-1) + 1$, if $x[i] = y[j]$

$= \max(S(i-1, j), S(i, j-1))$, otherwise

**STEP 3:** Order of subtasks

Row by row, left to right

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:** Row 0, Column 0
Longest Common Subsequence (LCS)

**Problem:** Given two sequences x[1..m] and y[1..n], find their longest common subsequence

**Example:**

x = A,C,G,T,A,G  
y = G,T,C,C,A,C  
LCS(x, y) = G,T,A

**STEP 1: Define subtasks**

S(i,j) = Length of LCS of x[1..i] and y[1..j]

Output of algorithm = S(m,n)

**STEP 2: Express recursively**

S(i,j) = S(i-1,j-1) + 1, if x[i] = y[j]

= max(S(i-1,j), S(i,j-1)), otherwise

**STEP 3: Order of subtasks**

Row by row, left to right

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:** Row 0, Column 0
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**Example:**

$x = A,C,G,T,A,G$
$y = G,T,C,C,A,C$

$LCS(x, y) = G,T,A$

**STEP 1: Define subtasks**

$S(i,j) =$ Length of LCS of $x[1..i]$ and $y[1..j]$

Output of algorithm = $S(m,n)$

**STEP 2: Express recursively**

$S(i,j) = S(i-1,j-1) + 1$, if $x[i] = y[j]$

$= \max(S(i-1,j), S(i,j-1))$, otherwise

**STEP 3: Order of subtasks**

Row by row, left to right

### Base Case:

Row 0, Column 0
Longest Common Subsequence (LCS)

**Problem:** Given two sequences \(x[1..m]\) and \(y[1..n]\), find their longest common subsequence

**Example:**
\[x = \text{A,C,G,T,A,G}\]
\[y = \text{G,T,C,C,A,C}\]
\[\text{LCS}(x, y) = \text{G,T,A}\]

**STEP 1: Define subtasks**
\[S(i,j) = \text{Length of LCS of } x[1..i] \text{ and } y[1..j]\]
Output of algorithm = \(S(m,n)\)

**STEP 2: Express recursively**
\[S(i,j) = S(i-1,j-1) + 1, \text{ if } x[i] = y[j]\]
\[= \max(S(i-1,j), S(i,j-1)), \text{ otherwise}\]

**STEP 3: Order of subtasks**
Row by row, left to right

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:** Row 0, Column 0
Longest Common Subsequence (LCS)

**Problem:** Given two sequences x[1..m] and y[1..n], find their longest common subsequence

**Example:**

\[
x = A, C, G, T, A, G \\
y = G, T, C, C, A, C \\
\]

LCS(x, y) = G, T, A

**STEP 1: Define subtasks**

\[S(i,j) = \text{Length of LCS of } x[1..i] \text{ and } y[1..j]\]

Output of algorithm = S(m,n)

**STEP 2: Express recursively**

\[S(i,j) = \begin{cases} 
S(i-1,j-1) + 1, & \text{if } x[i] = y[j] \\
\max(S(i-1,j), S(i,j-1)), & \text{otherwise}
\end{cases}\]

**Base Case:** Row 0, Column 0

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Longest Common Subsequence (LCS)**

**Problem:** Given two sequences \( x[1..m] \) and \( y[1..n] \), find their longest common subsequence

**Example:**

\[
x = \text{A,C,G,T,A,G} \\
y = \text{G,T,C,C,A,C}
\]

LCS(x, y) = G,T,A

**STEP 1: Define subtasks**

\[ S(i,j) = \text{Length of LCS of } x[1..i] \text{ and } y[1..j] \]

Output of algorithm = \( S(m,n) \)

**STEP 2: Express recursively**

\[
S(i,j) = \begin{cases} 
S(i-1,j-1) + 1, & \text{if } x[i] = y[j] \\
\max(S(i-1,j), S(i,j-1)), & \text{otherwise}
\end{cases}
\]

**STEP 3: Order of subtasks**

Row by row, left to right

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:** Row 0, Column 0
Longest Common Subsequence (LCS)

**Problem:** Given two sequences x[1..m] and y[1..n], find their longest common subsequence

**Example:**

\[ x = \text{A, C, G, T, A, G} \]
\[ y = \text{G, T, C, C, A, C} \]

LCS(x, y) = G, T, A

**STEP 1: Define subtasks**

\[ S(i, j) = \text{Length of LCS of } x[1..i] \]
\[ \text{and } y[1..j] \]

Output of algorithm = \( S(m, n) \)

**STEP 2: Express recursively**

\[ S(i, j) = S(i-1, j-1) + 1, \text{ if } x[i] = y[j] \]
\[ = \max(S(i-1, j), S(i, j-1)), \text{ otherwise} \]

**STEP 3: Order of subtasks**

Row by row, left to right

---

**Base Case:** Row 0, Column 0

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**Example:**

$x = A,C,G,T,A,G$

$y = G,T,C,C,A,C$

LCS($x$, $y$) = $G,T,A$

**STEP 1: Define subtasks**

$S(i,j)$ = Length of LCS of $x[1..i]$ and $y[1..j]$

Output of algorithm = $S(m,n)$

**STEP 2: Express recursively**

$S(i,j) = S(i-1,j-1) + 1$, if $x[i] = y[j]$

$= \max(S(i-1,j), S(i,j-1))$, otherwise

**STEP 3: Order of subtasks**

Row by row, left to right

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:** Row 0, Column 0
**Longest Common Subsequence (LCS)**

**Problem:** Given two sequences \(x[1..m]\) and \(y[1..n]\), find their longest common subsequence

**Example:**

\[x = A,C,G,T,A,G\]
\[y = G,T,C,C,A,C\]

\[LCS(x, y) = G,T,A\]

**STEP 1: Define subtasks**

\[S(i,j) = \text{Length of LCS of } x[1..i] \text{ and } y[1..j]\]

Output of algorithm = \(S(m,n)\)

**STEP 2: Express recursively**

\[S(i,j) = S(i-1,j-1) + 1, \text{ if } x[i] = y[j]\]

\[= \max(S(i-1,j), S(i,j-1)), \text{ otherwise} \]

**STEP 3: Order of subtasks**

Row by row, left to right

---

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:** Row 0, Column 0
**Longest Common Subsequence (LCS)**

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**Example:**

$x = A, C, G, T, A, G$
$y = G, T, C, C, A, C$

$LCS(x, y) = G, T, A$

**STEP 1: Define subtasks**

$S(i,j) =$ Length of LCS of $x[1..i]$ and $y[1..j]$

Output of algorithm $= S(m,n)$

**STEP 2: Express recursively**

$S(i,j) = S(i-1,j-1) + 1$, if $x[i] = y[j]$

$= \max(S(i-1,j), S(i,j-1))$, otherwise

**STEP 3: Order of subtasks**

Row by row, left to right

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:** Row 0, Column 0
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**Example:**
$x = A,C,G,T,A,G$
$y = G,T,C,C,A,C$

$LCS(x, y) = G,T,A$

**STEP 1: Define subtasks**

$S(i,j) =$ Length of LCS of $x[1..i]$ and $y[1..j]$

Output of algorithm = $S(m,n)$

**STEP 2: Express recursively**

$S(i,j)$ = $S(i-1,j-1) + 1$, if $x[i] = y[j]$

$= \max(S(i-1,j), S(i,j-1))$, ow

**STEP 3: Order of subtasks**

Row by row, left to right

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:** Row 0, Column 0
Longest Common Subsequence (LCS)

Problem: Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence.

Example:

$x = A, C, G, T, A, G$
$y = G, T, C, C, A, C$

$LCS(x, y) = G, T, A$

STEP 1: Define subtasks

$S(i, j) =$ Length of LCS of $x[1..i]$ and $y[1..j]$

Output of algorithm = $S(m,n)$

STEP 2: Express recursively

$S(i, j) = S(i-1, j-1) + 1$, if $x[i] = y[j]$

$= \max(S(i-1, j), S(i, j-1))$, otherwise

STEP 3: Order of subtasks

Row by row, left to right

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Base Case: Row 0, Column 0
**Longest Common Subsequence (LCS)**

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**Example:**

$x = A,C,G,T,A,G$

$y = G,T,C,C,A,C$

LCS($x$, $y$) = $G,T,A$

**STEP 1: Define subtasks**

$S(i,j) = $ Length of LCS of $x[1..i]$ and $y[1..j]$

Output of algorithm = $S(m,n)$

**STEP 2: Express recursively**

$S(i,j) = S(i-1,j-1) + 1$, if $x[i] = y[j]$

$= \max(S(i-1,j), S(i,j-1))$, otherwise

**Base Case:** Row 0, Column 0

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence.

**Example:**

$x = A, C, G, T, A, G$
$y = G, T, C, C, A, C$

$LCS(x, y) = G, T, A$

**STEP 1: Define subtasks**

$S(i, j) = \text{Length of LCS of } x[1..i] \text{ and } y[1..j]$

Output of algorithm = $S(m, n)$

**STEP 2: Express recursively**

$S(i, j) = S(i-1, j-1) + 1$, if $x[i] = y[j]$

$= \max(S(i-1, j), S(i, j-1))$, otherwise

**STEP 3: Order of subtasks**

Row by row, left to right

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:** Row 0, Column 0

---

1. **Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence.
2. **Example:**
   - $x = A, C, G, T, A, G$
   - $y = G, T, C, C, A, C$
   - $LCS(x, y) = G, T, A$
3. **STEP 1: Define subtasks**
   - $S(i, j) = \text{Length of LCS of } x[1..i] \text{ and } y[1..j]$
   - Output of algorithm = $S(m, n)$
4. **STEP 2: Express recursively**
   - $S(i, j) = S(i-1, j-1) + 1$, if $x[i] = y[j]$
   - $= \max(S(i-1, j), S(i, j-1))$, otherwise
5. **STEP 3: Order of subtasks**
   - Row by row, left to right

---

**Base Case:** Row 0, Column 0
**Longest Common Subsequence (LCS)**

**Problem:** Given two sequences \(x[1..m]\) and \(y[1..n]\), find their longest common subsequence.

**Example:**

\[x = A, C, G, T, A, G\]

\[y = G, T, C, C, A, C\]

\[\text{LCS}(x, y) = G, T, A\]

**STEP 1: Define subtasks**

\[S(i, j) = \text{Length of LCS of } x[1..i] \text{ and } y[1..j]\]

Output of algorithm = \(S(m, n)\)

**STEP 2: Express recursively**

\[S(i, j) = S(i-1, j-1) + 1, \text{ if } x[i] = y[j]\]

\[= \max(S(i-1, j), S(i, j-1)), \text{ otherwise}\]

**STEP 3: Order of subtasks**

Row by row, left to right

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:** Row 0, Column 0
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**STEP 1: Define subtasks**

$S(i,j) =$ Length of LCS of $x[1..i]$ and $y[1..j]$

Output of algorithm = $S(m,n)$

**STEP 2: Express recursively**

$S(i,j) =$ $S(i-1,j-1) + 1$, if $x[i] = y[j]$

$= \max(S(i-1,j), S(i,j-1))$, otherwise

**STEP 3: Order of subtasks**

Row by row, left to right

**Algorithm:**

for $i = 0$ to $n$: $S[i,0] = 0$

for $j = 0$ to $m$: $S[0,j] = 0$

for $i = 1$ to $n$:

for $j = 1$ to $m$:

if $x[i] = y[j]$:

$S[i,j] = S[i-1,j-1] + 1$

else:

$S[i,j] = \max\{S[i-1,j], S[i,j-1]\}$

return $S[n,m]$
**Longest Common Subsequence (LCS)**

**Problem:** Given two sequences x[1..m] and y[1..n], find their longest common subsequence

**Algorithm:**

```python
for i = 0 to n: S[i,0] = 0
for j = 0 to m: S[0,j] = 0
for i = 1 to n:
    for j = 1 to m:
        if x[i] = y[j]:
            S[i,j] = S[i-1,j-1] + 1
        else:
            S[i,j] = max{S[i-1,j], S[i,j-1]}
return S[n,m]
```

**STEP 1: Define subtasks**

S(i,j) = Length of LCS of x[1..i] and y[1..j]

Output of algorithm = S(m,n)

**STEP 2: Express recursively**

S(i,j) = S(i-1,j-1) + 1, if x[i] = y[j]

= max(S(i-1,j), S(i,j-1)), otherwise

**STEP 3: Order of subtasks**

Row by row, left to right

**Running Time:** O(mn)
Longest Common Subsequence (LCS)

**Problem:** Given two sequences \( x[1..m] \) and \( y[1..n] \), find their longest common subsequence

**Algorithm:**

```plaintext
for i = 0 to n: S[i,0] = 0
for j = 0 to m: S[0,j] = 0
for i = 1 to n:
    for j = 1 to m:
        if x[i] = y[j]:
            S[i,j] = S[i-1,j-1] + 1
        else:
            S[i,j] = max{S[i-1,j], S[i,j-1]}
return S[n,m]
```

**STEP 1: Define subtasks**

\( S(i,j) = \text{Length of LCS of } x[1..i] \)
and \( y[1..j] \)

Output of algorithm = \( S(m,n) \)

**STEP 2: Express recursively**

\( S(i,j) = S(i-1,j-1) + 1, \text{ if } x[i] = y[j] \)

\( = \max(S(i-1,j), S(i,j-1)), \text{ otherwise} \)

**STEP 3: Order of subtasks**

Row by row, left to right

**Running Time:** \( O(mn) \)

How to reconstruct the actual subsequence?
**Longest Common Subsequence (LCS)**

**Problem:** Given two sequences \(x[1..m]\) and \(y[1..n]\), find their longest common subsequence.

**STEP 1: Define subtasks**

\[ S(i,j) = \text{Length of LCS of } x[1..i] \text{ and } y[1..j] \]

**STEP 2: Express recursively**

\[ S(i,j) = \begin{cases} S(i-1,j-1) + 1, & \text{if } x[i] = y[j] \\ \max(S(i-1,j), S(i,j-1)), & \text{ow} \end{cases} \]

**To reconstruct LCS:**

Define \( L(i,j) \):

\[ L(i,j) = \begin{cases} (i-1,j-1), & \text{if } x[i] = y[j] \\ (i-1,j), & \text{ow if } S(i-1,j) > S(i,j-1) \\ (i,j-1), & \text{ow} \end{cases} \]

Reconstruct LCS by following the \( L(i,j) \) pointers, starting with \( L(m,n) \)

Recall: Row 0 and column 0: Base Case
Longest Common Subsequence (LCS)

**Problem:** Given two sequences \(x[1..m]\) and \(y[1..n]\), find their longest common subsequence.

**STEP 1: Define subtasks**

\(S(i,j) = \text{Length of LCS of } x[1..i] \text{ and } y[1..j]\)

**STEP 2: Express recursively**

\[S(i,j) = \begin{cases} S(i-1,j-1) + 1, & \text{if } x[i] = y[j] \\ \max(S(i-1,j), S(i,j-1)), & \text{otherwise} \end{cases}\]

To reconstruct LCS:

Define \(L(i,j)\):

\[L(i, j) = \begin{cases} (i - 1, j - 1), & \text{if } x[i] = y[j] \\ (i - 1, j), & \text{if } S(i-1,j) > S(i, j-1) \\ (i, j - 1), & \text{otherwise} \end{cases}\]

Recall: Row 0 and column 0: Base Case

Reconstruct LCS by following the \(L(i,j)\) pointers, starting with \(L(m,n)\)

---

**Table:**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

**Diagram:**

```
  0 1 2 3 4 5 6 7 8 9
  2 0 0 1 1 1 1 2 2 2
  3 0 0 1 2 2 2 2 2 3
  4 1 1 1 2 2 2 2 2 3
```

---

Recall: Row 0 and column 0: Base Case
Longest Common Subsequence (LCS)

**Problem:** Given two sequences x[1..m] and y[1..n], find their longest common subsequence

**STEP 1: Define subtasks**

\[ S(i,j) = \text{Length of LCS of } x[1..i] \]
\[ \quad \text{and } y[1..j] \]

**STEP 2: Express recursively**

\[ S(i,j) = S(i-1,j-1) + 1, \text{ if } x[i] = y[j] \]
\[ = \max(S(i-1,j), S(i,j-1)), \text{ ow} \]

To reconstruct LCS:

Define \( L(i,j) \):

\[ L(i,j) = (i - 1, j - 1), \text{ if } x[i] = y[j] \]
\[ = (i - 1, j), \quad \text{ow if } S(i-1,j) > S(i, j-1) \]
\[ = (i, j - 1), \quad \text{ow} \]

Reconstruct LCS by following the \( L(i,j) \) pointers, starting with \( L(m,n) \)

Recall: Row 0 and column 0: Base Case
**Longest Common Subsequence (LCS)**

**Problem:** Given two sequences \(x[1..m]\) and \(y[1..n]\), find their longest common subsequence

**STEP 1: Define subtasks**

\[ S(i,j) = \text{Length of LCS of } x[1..i] \text{ and } y[1..j] \]

**STEP 2: Express recursively**

\[ S(i,j) = S(i-1,j-1) + 1, \text{ if } x[i] = y[j] \]
\[ = \max(S(i-1,j), S(i,j-1)), \text{ otherwise} \]

**To reconstruct LCS:**

Define \(L(i,j)\):
\[ L(i,j) = (i - 1, j - 1), \text{ if } x[i] = y[j] \]
\[ = (i - 1, j), \text{ if } S(i-1,j) > S(i, j-1) \]
\[ = (i, j - 1), \text{ otherwise} \]

Reconstruct LCS by following the \(L(i,j)\) pointers, starting with \(L(m,n)\)

Recall: Row 0 and column 0: Base Case
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**STEP 1: Define subtasks**

$S(i, j) =$ Length of LCS of $x[1..i]$ and $y[1..j]$

**STEP 2: Express recursively**

$S(i, j) = S(i-1, j-1) + 1$, if $x[i] = y[j]$

$= \max(S(i-1, j), S(i, j-1))$, ow

To reconstruct LCS:

Define $L(i, j)$:

$L(i, j) = (i - 1, j - 1)$, if $x[i] = y[j]$

$= (i - 1, j)$, $ow$ if $S(i-1, j) > S(i, j-1)$

$= (i, j - 1)$, $ow$

Reconstruct LCS by following the $L(i, j)$ pointers, starting with $L(m, n)$

Recall: Row 0 and column 0: Base Case
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence.

**STEP 1: Define subtasks**

$S(i,j) =$ Length of LCS of $x[1..i]$
and $y[1..j]$

**STEP 2: Express recursively**

$S(i,j) = S(i-1,j-1) + 1$, if $x[i] = y[j]$

= max($S(i-1,j)$, $S(i,j-1)$), ow

**To reconstruct LCS:**

Define $L(i,j)$:

$L(i,j) = (i - 1, j - 1)$, if $x[i] = y[j]$

= $(i - 1, j)$, ow if $S(i-1,j) > S(i,j-1)$

= $(i, j - 1)$, ow

Reconstruct LCS by following the $L(i,j)$ pointers, starting with $L(m,n)$

Recall: Row 0 and column 0: Base Case
Longest Common Subsequence (LCS)

**Problem:** Given two sequences x[1..m] and y[1..n], find their longest common subsequence

**STEP 1: Define subtasks**

\[ S(i,j) = \text{Length of LCS of } x[1..i] \text{ and } y[1..j] \]

**STEP 2: Express recursively**

\[ S(i,j) = S(i-1,j-1) + 1, \text{ if } x[i] = y[j] \]

\[ = \max(S(i-1,j), S(i,j-1)), \text{ otherwise} \]

To reconstruct LCS:

Define \( L(i,j) \):

\[ L(i,j) = \begin{cases} (i - 1, j - 1), & \text{if } x[i] = y[j] \\ (i - 1, j), & \text{otherwise if } S(i-1,j) > S(i, j-1) \\ (i, j - 1), & \text{otherwise} \end{cases} \]

Recall: Row 0 and column 0: Base Case

**Matrix S**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

**Matrix L**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
Longest Common Subsequence (LCS)

**Problem:** Given two sequences \( x[1..m] \) and \( y[1..n] \), find their longest common subsequence

**STEP 1: Define subtasks**

\[ S(i,j) = \text{Length of LCS of } x[1..i] \]
and \( y[1..j] \)

**STEP 2: Express recursively**

\[ S(i,j) = \begin{cases} 
S(i-1,j-1) + 1, & \text{if } x[i] = y[j] \\
\max(S(i-1,j), S(i,j-1)), & \text{otherwise}
\end{cases} \]

To reconstruct LCS:

Define \( L(i, j) \):

\[ L(i, j) = \begin{cases} 
(i - 1, j - 1), & \text{if } x[i] = y[j] \\
(i - 1, j), & \text{if } S(i-1,j) > S(i, j-1) \\
(i, j - 1), & \text{otherwise}
\end{cases} \]

Reconstruct LCS by following the \( L(i,j) \) pointers, starting with \( L(m,n) \)

Recall: Row 0 and column 0: Base Case
**Longest Common Subsequence (LCS)**

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**STEP 1: Define subtasks**

$S(i,j)$ = Length of LCS of $x[1..i]$ and $y[1..j]$

**STEP 2: Express recursively**

$S(i,j)$ = $S(i-1,j-1) + 1$, if $x[i] = y[j]$

= $\max(S(i-1,j), S(i,j-1))$, ow

To reconstruct LCS:

Define $L(i,j)$:

$L(i,j) = (i - 1, j - 1)$, if $x[i] = y[j]$

= $(i - 1, j)$, ow if $S(i-1,j) > S(i, j-1)$

= $(i, j - 1)$, ow

Reconstruct LCS by following the $L(i,j)$ pointers, starting with $L(m,n)$

Recall: Row 0 and column 0: Base Case
**Longest Common Subsequence (LCS)**

**Problem:** Given two sequences \( x[1..m] \) and \( y[1..n] \), find their longest common subsequence

**STEP 1: Define subtasks**

\[
S(i, j) = \text{Length of LCS of } x[1..i] \text{ and } y[1..j]
\]

**STEP 2: Express recursively**

\[
S(i, j) = S(i-1, j-1) + 1, \text{ if } x[i] = y[j]
\]

\[
= \max(S(i-1, j), S(i, j-1)), \text{ ow}
\]

To reconstruct LCS:

Define \( L(i, j) \):

\[
L(i, j) = (i - 1, j - 1), \text{ if } x[i] = y[j]
\]

\[
= (i - 1, j), \quad \text{ ow if } S(i-1, j) > S(i, j-1)
\]

\[
= (i, j - 1), \quad \text{ ow}
\]

Reconstruct LCS by following the \( L(i,j) \) pointers, starting with \( L(m,n) \)

Recall: Row 0 and column 0: Base Case
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence.

**STEP 1: Define subtasks**

$S(i,j) =$ Length of LCS of $x[1..i]$ and $y[1..j]$

**STEP 2: Express recursively**

$S(i,j) = S(i-1,j-1) + 1$, if $x[i] = y[j]$

$= \max(S(i-1,j), S(i,j-1))$, ow

To reconstruct LCS:

Define $L(i, j)$:

$L(i,j) = (i - 1, j - 1)$, if $x[i] = y[j]$

$= (i - 1, j)$, ow if $S(i-1,j) > S(i, j-1)$

$= (i, j - 1)$, ow

Reconstruct LCS by following the $L(i,j)$ pointers, starting with $L(m,n)$

Recall: Row 0 and column 0: Base Case
Longest Common Subsequence (LCS)

**Problem:** Given two sequences x[1..m] and y[1..n], find their longest common subsequence

**STEP 1: Define subtasks**
S(i,j) = Length of LCS of x[1..i] and y[1..j]

**STEP 2: Express recursively**
S(i,j) = S(i-1,j-1) + 1, if x[i] = y[j]
= max(S(i-1,j), S(i,j-1)), ow

To reconstruct LCS:
Define L(i, j):
L(i, j) = (i - 1, j - 1), if x[i] = y[j]
= (i - 1, j), ow if S(i-1,j) > S(i, j-1)
= (i, j - 1), ow

Reconstruct LCS by following the L(i,j) pointers, starting with L(m,n)
Recall: Row 0 and column 0: Base Case
Longest Common Subsequence (LCS)

**Problem:** Given two sequences x[1..m] and y[1..n], find their longest common subsequence

**STEP 1: Define subtasks**

\[ S(i,j) = \text{Length of LCS of } x[1..i] \]
\[ \text{and } y[1..j] \]

**STEP 2: Express recursively**

\[ S(i,j) = S(i-1,j-1) + 1, \text{ if } x[i] = y[j] \]
\[ = \max(S(i-1,j), S(i,j-1)), \text{ otherwise} \]

To reconstruct LCS:

Define \( L(i,j) \):

\[ L(i, j) = (i - 1, j - 1), \text{ if } x[i] = y[j] \]
\[ = (i - 1, j), \text{ if } S(i-1,j) > S(i, j-1) \]
\[ = (i, j - 1), \text{ otherwise} \]

Reconstruct LCS by following the \( L(i,j) \) pointers, starting with \( L(m,n) \)

Recall: Row 0 and column 0: Base Case
Longest Common Subsequence (LCS)

**Problem:** Given two sequences x[1..m] and y[1..n], find their longest common subsequence

**STEP 1: Define subtasks**

\[ S(i,j) = \text{Length of LCS of } x[1..i] \text{ and } y[1..j] \]

**STEP 2: Express recursively**

\[ S(i,j) = S(i-1,j-1) + 1, \text{ if } x[i] = y[j] \]

\[ = \max(S(i-1,j), S(i,j-1)), \text{ ow} \]

To reconstruct LCS:

Define \( L(i,j) \):

\[ L(i,j) = (i-1,j-1), \text{ if } x[i] = y[j] \]

\[ = (i-1,j), \text{ ow if } S(i-1,j) > S(i,j-1) \]

\[ = (i,j-1), \text{ ow} \]

Reconstruct LCS by following the \( L(i,j) \) pointers, starting with \( L(m,n) \)

Recall: Row 0 and column 0: Base Case
Longest Common Subsequence (LCS)

**Problem:** Given two sequences x[1..m] and y[1..n], find their longest common subsequence

**STEP 1: Define subtasks**

\[ S(i,j) = \text{Length of LCS of } x[1..i] \]
\[ \text{and } y[1..j] \]

**STEP 2: Express recursively**

\[ S(i,j) = \begin{cases} 
S(i-1,j-1) + 1, & \text{if } x[i] = y[j] \\
\max(S(i-1,j), S(i,j-1)), & \text{otherwise} 
\end{cases} \]

To reconstruct LCS:

Define \( L(i, j) \):

\[ L(i, j) = \begin{cases} 
(i - 1, j - 1), & \text{if } x[i] = y[j] \\
(i - 1, j), & \text{if } S(i-1,j) > S(i, j-1) \\
(i, j - 1), & \text{otherwise} 
\end{cases} \]

Reconstruct LCS by following the \( L(i,j) \) pointers, starting with \( L(m,n) \)

Recall: Row 0 and column 0: Base Case
**Longest Common Subsequence (LCS)**

**Problem:** Given two sequences x[1..m] and y[1..n], find their longest common subsequence

---

**STEP 1: Define subtasks**

\[ S(i,j) = \text{Length of LCS of } x[1..i] \]

and \( y[1..j] \)

**STEP 2: Express recursively**

\[ S(i,j) = S(i-1,j-1) + 1, \text{ if } x[i] = y[j] \]

\[ = \max(S(i-1,j), S(i,j-1)), \text{ ow} \]

---

**To reconstruct LCS:**

Define \( L(i, j) \):

\[ L(i, j) = (i - 1, j - 1), \text{ if } x[i] = y[j] \]

\[ = (i - 1, j), \text{ ow if } S(i-1,j) > S(i, j-1) \]

\[ = (i, j - 1), \text{ ow} \]

Reconstruct LCS by following the \( L(i,j) \) pointers, starting with \( L(m,n) \)

Recall: Row 0 and column 0: Base Case
**Longest Common Subsequence (LCS)**

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**STEP 1: Define subtasks**

$S(i,j) = $ Length of LCS of $x[1..i]$ and $y[1..j]$

**STEP 2: Express recursively**

$S(i,j) = S(i-1,j-1) + 1$, if $x[i] = y[j]$

$= \max(S(i-1,j), S(i,j-1))$, ow

To reconstruct LCS:

Define $L(i,j)$:

$L(i, j) = (i - 1, j - 1)$, if $x[i] = y[j]$

$= (i - 1, j)$, ow if $S(i-1,j) > S(i, j-1)$

$= (i, j - 1)$, ow

Reconstruct LCS by following the $L(i,j)$ pointers, starting with $L(m,n)$

Recall: Row 0 and column 0: Base Case
Longest Common Subsequence (LCS)

Problem: Given two sequences x[1..m] and y[1..n], find their longest common subsequence

STEP 1: Define subtasks
S(i,j) = Length of LCS of x[1..i] and y[1..j]

STEP 2: Express recursively
S(i,j) = S(i-1,j-1) + 1, if x[i] = y[j]
= max(S(i-1,j), S(i,j-1)), ow

To reconstruct LCS:
Define L(i, j):
L(i, j) = (i - 1, j - 1), if x[i] = y[j]
= (i - 1, j), ow if S(i-1,j) > S(i, j-1)
= (i, j - 1), ow

Reconstruct LCS by following the L(i,j) pointers, starting with L(m,n)
Recall: Row 0 and column 0: Base Case
Longest Common Subsequence (LCS)

**Problem:** Given two sequences \( x[1..m] \) and \( y[1..n] \), find their longest common subsequence

**STEP 1: Define subtasks**

\[
S(i,j) = \text{Length of LCS of } x[1..i] \quad \text{and } y[1..j]
\]

**STEP 2: Express recursively**

\[
S(i,j) = \begin{cases} 
S(i-1,j-1) + 1, & \text{if } x[i] = y[j] \\
\max(S(i-1,j), S(i,j-1)), & \text{ow}
\end{cases}
\]

To reconstruct LCS:

Define \( L(i, j) \):

\[
L(i, j) = \begin{cases} 
(i - 1, j - 1), & \text{if } x[i] = y[j] \\
(i - 1, j), & \text{ow if } S(i-1,j) > S(i, j-1) \\
(i, j - 1), & \text{ow}
\end{cases}
\]

Reconstruct LCS by following the \( L(i,j) \) pointers, starting with \( L(m,n) \)

Recall: Row 0 and column 0: Base Case
**Longest Common Subsequence (LCS)**

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**STEP 1: Define subtasks**

$S(i,j) = \text{Length of LCS of } x[1..i]$

and $y[1..j]$

**STEP 2: Express recursively**

$S(i,j) = S(i-1,j-1) + 1$, if $x[i] = y[j]$

$= \max(S(i-1,j), S(i,j-1))$, ow

**To reconstruct LCS:**

Define $L(i,j)$:

$L(i,j) = (i - 1, j - 1)$, if $x[i] = y[j]$

$= (i - 1, j)$, ow if $S(i-1,j) > S(i,j-1)$

$= (i, j - 1)$, ow

Reconstruct LCS by following the $L(i,j)$ pointers, starting with $L(m,n)$

Recall: Row 0 and column 0: Base Case
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**STEP 1: Define subtasks**

$S(i,j) =$ Length of LCS of $x[1..i]$ and $y[1..j]$

**STEP 2: Express recursively**

$S(i,j)$  
= $S(i-1,j-1) + 1$, if $x[i] = y[j]$  
= max($S(i-1,j)$, $S(i,j-1)$), ow

To reconstruct LCS:

Define $L(i, j)$:

$L(i, j)$  
= $(i - 1, j - 1)$, if $x[i] = y[j]$  
= $(i - 1, j)$, ow if $S(i-1,j) > S(i, j-1)$  
= $(i, j - 1)$, ow

Reconstruct LCS by following the $L(i,j)$ pointers, starting with $L(m,n)$

Recall: Row 0 and column 0: Base Case
Longest Common Subsequence (LCS)

**Problem:** Given two sequences x[1..m] and y[1..n], find their longest common subsequence.

**STEP 1: Define subtasks**

\( S(i, j) = \) Length of LCS of \( x[1..i] \)

\( \) and \( y[1..j] \)

**STEP 2: Express recursively**

\[ S(i, j) = \begin{cases} 
S(i-1, j-1) + 1, & \text{if } x[i] = y[j] \\
\max(S(i-1, j), S(i, j-1)), & \text{otherwise}
\end{cases} \]

**To reconstruct LCS:**

Define \( L(i, j) \):

\( L(i, j) = \begin{cases} 
(i - 1, j - 1), & \text{if } x[i] = y[j] \\
(i - 1, j), & \text{if } S(i-1, j) > S(i, j-1) \\
(i, j - 1), & \text{otherwise}
\end{cases} \)

Reconstruct LCS by following the \( L(i, j) \) pointers, starting with \( L(m, n) \)

Recall: Row 0 and column 0: Base Case
Longest Common Subsequence (LCS)

**Problem:** Given two sequences x[1..m] and y[1..n], find their longest common subsequence

**STEP 1: Define subtasks**

\[ S(i,j) = \text{Length of LCS of } x[1..i] \]
and \[ y[1..j] \]

**STEP 2: Express recursively**

\[ S(i,j) = S(i-1,j-1) + 1, \text{ if } x[i] = y[j] \]
\[ = \max(S(i-1,j), S(i,j-1)), \text{ ow} \]

To reconstruct LCS:

Define \( L(i,j) \):

\[ L(i,j) = (i - 1, j - 1), \text{ if } x[i] = y[j] \]
\[ = (i - 1, j), \text{ ow if } S(i-1,j) > S(i, j-1) \]
\[ = (i, j - 1), \text{ ow} \]

Reconstruct LCS by following the \( L(i,j) \) pointers, starting with \( L(m,n) \)
Recall: Row 0 and column 0: Base Case
Longest Common Subsequence (LCS)

**Problem:** Given two sequences \(x[1..m]\) and \(y[1..n]\), find their longest common subsequence

**STEP 1: Define subtasks**

\[ S(i,j) = \text{Length of LCS of } x[1..i] \text{ and } y[1..j] \]

**STEP 2: Express recursively**

\[ S(i,j) = S(i-1,j-1) + 1, \text{ if } x[i] = y[j] \]
\[ = \max(S(i-1,j), S(i,j-1)), \text{ otherwise} \]

To reconstruct LCS:

Define \(L(i,j)\):

\[ L(i, j) = (i - 1, j - 1), \text{ if } x[i] = y[j] \]
\[ = (i - 1, j), \text{ if } S(i-1,j) > S(i,j-1) \]
\[ = (i, j - 1), \text{ otherwise} \]

Reconstruct LCS by following the \(L(i,j)\) pointers, starting with \(L(m,n)\)

Recall: Row 0 and column 0: Base Case
Longest Common Subsequence (LCS)

**Problem:** Given two sequences \(x[1..m]\) and \(y[1..n]\), find their longest common subsequence

**STEP 1: Define subtasks**

\(S(i,j) = \text{Length of LCS of } x[1..i] \text{ and } y[1..j]\)

**STEP 2: Express recursively**

\(S(i,j) = S(i-1,j-1) + 1, \text{ if } x[i] = y[j]\)

\(= \max(S(i-1,j), S(i,j-1)), \text{ ow}

To reconstruct LCS:

Define \(L(i, j)\):

\(L(i, j) = (i - 1, j - 1), \text{ if } x[i] = y[j]\)

\(= (i - 1, j), \text{ ow if } S(i-1,j) > S(i, j-1)\)

\(= (i, j - 1), \text{ ow}

Reconstruct LCS by following the \(L(i,j)\) pointers, starting with \(L(m,n)\)

Recall: Row 0 and column 0: Base Case

\(\text{LCS} = T, G, A\)
Longest Common Subsequence (LCS)

**Problem:** Given two sequences $x[1..m]$ and $y[1..n]$, find their longest common subsequence

**STEP 1: Define subtasks**

$S(i,j) = \text{Length of LCS of } x[1..i] \text{ and } y[1..j]$

Output of algorithm = $S(m,n)$

**STEP 2: Express recursively**

$S(i,j) = \begin{cases} 
S(i-1,j-1) + 1, & \text{if } x[i] = y[j] \\
\max(S(i-1,j), S(i,j-1)), & \text{ow}
\end{cases}$

**STEP 3: Order of subtasks**

Row by row, left to right

**To reconstruct LCS:**

Define $L(i,j)$:

$L(i,j) = \begin{cases} 
(i-1, j-1), & \text{if } x[i] = y[j] \\
(i-1, j), & \text{ow if } S(i-1,j) > S(i,j-1) \\
(i, j-1), & \text{ow}
\end{cases}$

Reconstruct LCS by following the $L(i,j)$ pointers, starting with $L(m,n)$

**Running Time:** $O(mn)$
Dynamic Programming

• String Reconstruction
• Longest Common Subsequence
• ...

<table>
<thead>
<tr>
<th>Dynamic Programming vs Divide and Conquer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Divide-and-conquer</strong></td>
</tr>
<tr>
<td>A problem of size $n$ is decomposed into several subproblems which are significantly smaller (e.g. $n/2$, $3n/4$, ...).</td>
</tr>
<tr>
<td>Therefore, size of subproblems decreases geometrically.</td>
</tr>
<tr>
<td>eg. $n$, $n/2$, $n/4$, $n/8$, etc</td>
</tr>
<tr>
<td>Use a recursive algorithm.</td>
</tr>
<tr>
<td><strong>Dynamic programming</strong></td>
</tr>
<tr>
<td>A problem of size $n$ is expressed in terms of subproblems that are not much smaller (e.g. $n-1$, $n-2$, ...).</td>
</tr>
<tr>
<td>A recursive algorithm would take exponential time.</td>
</tr>
<tr>
<td>Saving grace: in total, there are only polynomially many subproblems.</td>
</tr>
<tr>
<td>Avoid recursion and instead solve the subproblems one-by-one, saving the answers in a table, in a clever explicit order.</td>
</tr>
</tbody>
</table>
**Case 1:** Input: \( x_1, x_2, \ldots, x_n \) Subproblem: \( x_1, \ldots, x_i \).
Case 1: Input: $x_1, x_2, \ldots, x_n$ Subproblem: $x_1, \ldots, x_i$. 

Case 2: Input: $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_m$ Subproblem: $x_1, \ldots, x_i$ and $y_1, y_2, \ldots, y_j$. 

\[ \begin{array}{ccccccccccc} 
  x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} \\
\end{array} \]

\[ \begin{array}{ccccccccccc} 
  y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 \\
\end{array} \]
DP: Common Subtasks

**Case 1:** Input: $x_1, x_2, \ldots, x_n$ Subproblem: $x_1, \ldots, x_i$.

```
X_1 X_2 X_3 X_4 X_5 X_6 X_7 X_8 X_9 X_{10}
```

**Case 2:** Input: $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_m$ Subproblem: $x_1, \ldots, x_i$ and $y_1, y_2, \ldots, y_j$.

```
X_1 X_2 X_3 X_4 X_5 X_6 X_7 X_8 X_9 X_{10}
Y_1 Y_2 Y_3 Y_4 Y_5 Y_6 Y_7 Y_8
```

**Case 3:** Input: $x_1, x_2, \ldots, x_n$. Subproblem: $x_i, \ldots, x_j$.

```
X_1 X_2 X_3 X_4 X_5 X_6 X_7 X_8 X_9 X_{10}
```
**Case 1:** Input: $x_1, x_2, ..., x_n$ Subproblem: $x_1, ..., x_i$.

**Case 2:** Input: $x_1, x_2, ..., x_n$ and $y_1, y_2, ..., y_m$ Subproblem: $x_1, ..., x_i$ and $y_1, y_2, ..., y_j$.

**Case 3:** Input: $x_1, x_2, ..., x_n$. Subproblem: $x_i, ..., x_j$.

**Case 4:** Input: a rooted tree. Subproblem: a subtree.
Dynamic Programming

- String Reconstruction
- Longest Common Subsequence
- Edit Distance
Edit Distance: String Alignment

**Alignment:** Convert one string to another using insertions, deletions and substitutions.

$$\begin{array}{cccc}
S & U & N & Y \\
S & - & N & O \\
\end{array}$$

Alignment 1
Cost = 3
Edit Distance: String Alignment

**Alignment:** Convert one string to another using insertions, deletions and substitutions.

**Alignment 1**
- Cost = 3

**Alignment 2**
- Cost = 5
Edit Distance: String Alignment

Alignment: Convert one string to another using insertions, deletions and substitutions.

Alignment 1
Cost = 3

Alignment 2
Cost = 5

Edit Distance \((x, y)\): minimum # of insertions, deletions and substitutions required to convert \(x\) to \(y\)
Edit Distance: String Alignment

**Alignment:** Convert one string to another using insertions, deletions and substitutions.

```
S U N N Y
S - N O W Y
```

Alignment 1
Cost = 3

```
S U N N Y
S - N O W Y
```

Alignment 2
Cost = 5

**Edit Distance**\((x, y)\): minimum # of insertions, deletions and substitutions required to convert \(x\) to \(y\)

\(\text{Edit Distance}(\text{SUNNY}, \text{SNOWY}) = 3\)
Edit Distance: String Alignment

**Alignment:** Convert one string to another using insertions, deletions and substitutions.

S U N N Y
S - N O W Y

Alignment 1
Cost = 3

S U N Y
S - N O W

Alignment 2
Cost = 5

**Edit Distance**($x, y$): minimum # of insertions, deletions and substitutions required to convert $x$ to $y$

Edit Distance(SUNNY, SNOWY) = 3

Is Edit Distance($x, y$) = Edit Distance($y, x$)?
Edit Distance

**Problem:** Given two strings \( x[1..n] \) and \( y[1..m] \), compute edit-distance\((x, y)\)

Example:

\[
\begin{array}{cccc}
S & U & N & Y \\
S & - & N & O & Y \\
\end{array}
\]

Cost = 3

**Structure:**
Three cases in the last column of alignment between \( x[1..i] \) and \( y[1..j] \):

- **Case 1.** Align \( x[1..i-1] \) and \( y[1..j] \), delete \( x[i] \)
- **Case 2.** Align \( x[1..i] \) and \( y[1..j-1] \), insert \( y[j] \)
- **Case 3.** Align \( x[1..i-1] \) and \( y[1..j-1] \). Substitute \( x[i] \), \( y[j] \) if different.
Problem: Given two strings $x[1..n]$ and $y[1..m]$, compute edit-distance($x, y$)

Example:

$x = $ SUNNY  
$y = $ SNOWY  
$E(x, y) = 3$

STEP 1: Define subtasks
$E(i,j) = $ Edit-distance($x[1..i], y[1..j]$)
Output of algorithm = $E(n,m)$

STEP 2: Express recursively
$E(i,j) = \min(E(i-1,j) + 1, E(i, j-1) + 1, E(i-1,j-1) + \text{diff}(x[i], y[j]))$

STEP 3: Order of subtasks
Row by row, left to right
Problem: Given two strings $x[1..n]$ and $y[1..m]$, compute edit-distance($x, y$)

Example:

$x = \text{SUNNY}$

$y = \text{SNOWY}$

Edit-distance($x, y$) = 3

**STEP 1: Define subtasks**

$E(i,j) = \text{Edit-distance}(x[1..i], y[1..j])$

Output of algorithm = $E(n,m)$

**STEP 2: Express recursively**

$E(i,j) = \min(E(i-1,j) + 1, E(i, j-1) + 1, E(i-1,j-1) + \text{diff}(x[i], y[j]))$

**STEP 3: Order of subtasks**

Row by row, left to right
**Problem:** Given two strings $x[1..n]$ and $y[1..m]$, compute $edit-distance(x, y)$

**STEP 1: Define subtasks**  
$E(i,j) = Edit-distance(x[1..i], y[1..j])$  
Output of algorithm = $E(n,m)$

**STEP 2: Express recursively**  
$E(i,j) = \min(E(i-1,j) + 1, E(i, j-1) + 1, E(i-1,j-1) + \text{diff}(x[i], y[j]))$

**STEP 3: Order of subtasks**  
Row by row, left to right

**Example:**  
$x = \text{SUNNY}$  
$y = \text{SNOWY}$  
$edit-distance(x, y) = 3$

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>U</th>
<th>N</th>
<th>N</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Edit Distance

Problem: Given two strings $x[1..n]$ and $y[1..m]$, compute edit-distance$(x, y)$

STEP 1: Define subtasks
$E(i,j) = \text{Edit-distance}(x[1..i], y[1..j])$
Output of algorithm = $E(n,m)$

STEP 2: Express recursively
$E(i,j) = \min(E(i-1,j) + 1, E(i, j-1) + 1, E(i-1,j-1) + \text{diff}(x[i], y[j]))$

STEP 3: Order of subtasks
Row by row, left to right

Example:
$x = \text{SUNNY}$
$y = \text{SNOWY}$
Edit-distance$(x, y) = 3$

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>U</th>
<th>N</th>
<th>N</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem: Given two strings $x[1..n]$ and $y[1..m]$, compute edit-distance($x, y$)

**STEP 1: Define subtasks**

$E(i,j) = \text{Edit-distance}(x[1..i], y[1..j])$

Output of algorithm = $E(n,m)$

**STEP 2: Express recursively**

$E(i,j) = \min(E(i-1,j) + 1, E(i, j-1) + 1,$
          $E(i-1,j-1) + \text{diff}(x[i], y[j]))$

**STEP 3: Order of subtasks**

Row by row, left to right

Example:

$x = \text{SUNNY}$

$y = \text{SNOWY}$

Edit-distance($x, y$) = 3

```
<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>U</th>
<th>N</th>
<th>N</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>N</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
**Edit Distance**

**Problem:** Given two strings $x[1..n]$ and $y[1..m]$, compute $\text{edit-distance}(x, y)$

**Example:**
$x = \text{SUNNY}$  
$y = \text{SNOWY}$  
Edit-distance($x, y$) = 3

**STEP 1: Define subtasks**
$E(i,j) = \text{Edit-distance}(x[1..i], y[1..j])$
Output of algorithm = $E(n,m)$

**STEP 2: Express recursively**
$E(i,j) = \min(E(i-1,j) + 1, E(i, j-1) + 1,$
$E(i-1,j-1) + \text{diff}(x[i], y[j]))$

**STEP 3: Order of subtasks**
Row by row, left to right
Edit Distance

**Problem:** Given two strings $x[1..n]$ and $y[1..m]$, compute $\text{edit-distance}(x, y)$

**Example:**

$x = \text{SUNNY}$
$y = \text{SNOWY}$

Edit-distance($x, y$) = 3

**STEP 1: Define subtasks**

$E(i,j) = \text{Edit-distance}(x[1..i], y[1..j])$

Output of algorithm = $E(n,m)$

**STEP 2: Express recursively**

$E(i,j) = \min(E(i-1,j) + 1, E(i, j-1) + 1, E(i-1,j-1) + \text{diff}(x[i], y[j]))$

**STEP 3: Order of subtasks**

Row by row, left to right

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>U</th>
<th>N</th>
<th>N</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>N</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Edit Distance**

**Problem:** Given two strings $x[1..n]$ and $y[1..m]$, compute edit-distance$(x, y)$

**Example:**

$x = \text{SUNNY}$  
$y = \text{SNOWY}$  
Edit-distance$(x, y) = 3$

**STEP 1: Define subtasks**

$E(i,j) = \text{Edit-distance}(x[1..i], y[1..j])$

Output of algorithm = $E(n,m)$

**STEP 2: Express recursively**

$E(i,j) = \min(E(i-1,j) + 1, E(i, j-1) + 1,$

$E(i-1,j-1) + \text{diff}(x[i], y[j]))$

**STEP 3: Order of subtasks**

Row by row, left to right
Edit Distance

Problem: Given two strings $x[1..n]$ and $y[1..m]$, compute edit-distance($x, y$)

**Example:**

$x =$ SUNNY  
$y =$ SNOWY  
Edit-distance($x, y$) = 3

**STEP 1: Define subtasks**

$E(i,j) =$ Edit-distance($x[1..i], y[1..j]$)

Output of algorithm = $E(n,m)$

**STEP 2: Express recursively**

$E(i,j) = \min(E(i-1,j) + 1, E(i, j-1) + 1, E(i-1,j-1) + \text{diff}(x[i], y[j]))$

**STEP 3: Order of subtasks**

Row by row, left to right
**Edit Distance**

**Problem:** Given two strings $x[1..n]$ and $y[1..m]$, compute edit-distance($x, y$)

**Example:**

$x = \text{SUNNY}$  
$y = \text{SNOWY}$  
$\text{Edit-distance}(x, y) = 3$

**STEP 1: Define subtasks**

$E(i,j) = \text{Edit-distance}(x[1..i], y[1..j])$

Output of algorithm = $E(n,m)$

**STEP 2: Express recursively**

$E(i,j) = \min(E(i-1,j) + 1, E(i, j-1) + 1, E(i-1,j-1) + \text{diff}(x[i], y[j]))$

**STEP 3: Order of subtasks**

Row by row, left to right
**Edit Distance**

**Problem:** Given two strings $x[1..n]$ and $y[1..m]$, compute $\text{edit-distance}(x, y)$

**Example:**
\[
x = \text{SUNNY} \quad \text{Edit-distance}(x, y) = 3
\]
\[
y = \text{SNOWY}
\]

**STEP 1: Define subtasks**
\[
E(i,j) = \text{Edit-distance}(x[1..i], y[1..j])
\]
Output of algorithm = $E(n,m)$

**STEP 2: Express recursively**
\[
E(i,j) = \min(E(i-1,j) + 1, E(i, j-1) + 1, E(i-1,j-1) + \text{diff}(x[i], y[j]))
\]

**STEP 3: Order of subtasks**
Row by row, left to right

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>U</th>
<th>N</th>
<th>N</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>N</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>O</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Edit Distance

**Problem:** Given two strings $x[1..n]$ and $y[1..m]$, compute $\text{edit-distance}(x, y)$

## Example:

- $x = \text{SUNNY}$
- $y = \text{SNOWY}$

$\text{edit-distance}(x, y) = 3$

## STEP 1: Define subtasks

$E(i,j) = \text{Edit-distance}(x[1..i], y[1..j])$

Output of algorithm = $E(n,m)$

## STEP 2: Express recursively

$E(i,j) = \min(E(i-1,j) + 1, E(i, j-1) + 1, E(i-1,j-1) + \text{diff}(x[i], y[j]))$

## STEP 3: Order of subtasks

Row by row, left to right

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>U</th>
<th>N</th>
<th>N</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>N</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>O</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Edit Distance

**Problem:** Given two strings \( x[1..n] \) and \( y[1..m] \), compute edit-distance\((x, y)\)

**Example:**
\[
x = \text{SUNNY} \\
y = \text{SNOWY}
\]

**STEP 1: Define subtasks**
\[
E(i,j) = \text{Edit-distance}(x[1..i], y[1..j])
\]
Output of algorithm = \( E(n,m) \)

**STEP 2: Express recursively**
\[
E(i,j) = \min(E(i-1,j) + 1, E(i, j-1) + 1, E(i-1,j-1) + \text{diff}(x[i], y[j]))
\]

**STEP 3: Order of subtasks**
Row by row, left to right

**Running Time** = \( O(mn) \)

How to reconstruct the best alignment?
Dynamic Programming

- String Reconstruction
- Longest Common Subsequence
- Edit Distance
- Subset Sum
**Problem:** Given a list of positive integers $a[1..n]$ and an integer $t$, is there some subset of $a$ that sums to exactly $t$?

**Example:** $a = [ 12, 1, 3, 8, 20, 50 ]$
Subset Sum

**Problem:** Given a list of positive integers $a[1..n]$ and an integer $t$, is there some subset of $a$ that sums to exactly $t$?

**Example:**

- $a = [12, 1, 3, 8, 20, 50]$  
  - $t = 44$  
  - $t = 14$
Subset Sum

Problem: Given a list of positive integers $a[1..n]$ and an integer $t$, is there some subset of $a$ that sums to exactly $t$?

Example: $a = [12, 1, 3, 8, 20, 50]$  $t = 44$  True  $t = 14$  False
Subset Sum

**Problem:** Given a list of positive integers $a[1..n]$ and an integer $t$, is there some subset of $a$ that sums to exactly $t$?

**Example:** $a = [12, 1, 3, 8, 20, 50]$  
$t = 44$  True  
$t = 14$  False

**Structure:**
If $a[1...i]$ has a subset $T$ that sums to $s$, then:
- Either $a[i]$ is not in $T$: $a[1...i-1]$ has a subset that sums to $s$
- Or $a[i]$ is in $T$: $a[1...i-1]$ has a subset that sums to $s - a[i]$
**Subset Sum**

**Problem:** Given a list of positive integers $a[1..n]$ and an integer $t$, is there some subset of $a$ that sums to exactly $t$?

**Example:** $a = [12, 1, 3, 8, 20, 50]$

**STEP 1: Define subtasks**

For $i=1..n$, $s=1..t$,

$S(i,s) = True$, if some subset of $S[1..i]$ adds to $s$

$= False$, ow

Output = $S(n, t)$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$t = 44$  True  $t = 14$  False
Subset Sum

**Problem:** Given a list of positive integers \( a[1..n] \) and an integer \( t \), is there some subset of \( a \) that sums to exactly \( t \)?

**Example:** \( a = [12, 1, 3, 8, 20, 50] \)

**STEP 1: Define subtasks**

For \( i=1..n \), \( s=1..t \),

\[
S(i, s) = \begin{cases} 
\text{True, if some subset of } S[1..i] \text{ adds to } s \\
\text{False, otherwise}
\end{cases}
\]

Output = \( S(n, t) \)

**STEP 2: Express recursively**

If \( a[i] \leq s \),

\[ S(i, s) = S(i - 1, s - a[i]) \text{ OR } S(i - 1, s) \]

Else: \( S(i, s) = S(i - 1, s) \)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( t = 44 \quad \text{True} \quad t = 14 \quad \text{False} \)
Problem: Given a list of positive integers \( a[1..n] \) and an integer \( t \), is there some subset of \( a \) that sums to exactly \( t \)?

Example: \( a = [ 12, 1, 3, 8, 20, 50 ] \)

STEP 1: Define subtasks
For \( i=1..n \), \( s=1..t \),
\( S(i,s) = \begin{cases} \text{True, if some subset of } S[1..i] \\ \text{adds to } s \\ \text{False, otherwise} \end{cases} \)
Output = \( S(n, t) \)

STEP 2: Express recursively
If \( a[i] \leq s \),
\( S(i,s) = S(i - 1, s - a[i]) \) OR \( S(i - 1, s) \)
Else: \( S(i,s) = S(i - 1, s) \)

STEP 3: Order of subtasks
Row by row, increasing column
Subset Sum

**Problem:** Given a list of positive integers $a[1..n]$ and an integer $t$, is there some subset of $a$ that sums to exactly $t$?

**Example:** $a = [12, 1, 3, 8, 20, 50]$

**STEP 1: Define subtasks**
For $i=1..n$, $s=1..t$,
$S(i,s) = \text{True, if some subset of } S[1..i]\text{ adds to } s$
$= \text{False, otherwise}$
Output $= S(n, t)$

**STEP 2: Express recursively**
If $a[i] \leq s$,
$S(i,s) = S(i - 1, s - a[i]) \text{ OR } S(i - 1, s)$
Else: $S(i, s) = S(i - 1, s)$

**STEP 3: Order of subtasks**
Row by row, increasing column

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:**
Column 0: all True
Row 0: all False except $(0, 0)$ entry

$t = 44$ True  $t = 14$ False
Subset Sum

**Problem:** Given a list of positive integers \( a[1..n] \) and an integer \( t \), is there some subset of \( a \) that sums to exactly \( t \)?

**Example:** \( a = [ 12, 1, 3, 8, 20, 50 ] \)

**STEP 1: Define subtasks**
For \( i=1..n, \ s=1..t \),
\[
S(i,s) = \begin{cases} 
    \text{True, if some subset of } S[1..i] \\
    \text{adds to } s \\
    \text{False, otherwise}
\end{cases}
\]
Output = \( S(n, t) \)

**STEP 2: Express recursively**
If \( a[i] \leq s \),
\[
S(i,s) = S(i - 1, s - a[i]) \text{ OR } S(i - 1, s)
\]
Else: \( S(i, s) = S(i - 1, s) \)

**STEP 3: Order of subtasks**
Row by row, increasing column

---

<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

**Base Case:**
Column 0: all True
Row 0: all False except (0, 0) entry
Subset Sum

**Problem:** Given a list of positive integers $a[1..n]$ and an integer $t$, is there some subset of $a$ that sums to exactly $t$?

**Example:** $a = [12, 1, 3, 8, 20, 50]$  

$$t = 44 \quad \text{True} \quad t = 14 \quad \text{False}$$

**STEP 1: Define subtasks**

For $i=1..n$, $s=1..t$,

$S(i,s) = \begin{cases} 
\text{True, if some subset of } S[1..i] \\
\text{False, otherwise}
\end{cases}$

Output $= S(n, t)$

**STEP 2: Express recursively**

If $a[i] \leq s$,

$S(i,s) = S(i - 1, s - a[i]) \text{ OR } S(i - 1, s)$

Else: $S(i, s) = S(i - 1, s)$

**STEP 3: Order of subtasks**

Row by row, increasing column

Base Case:

Column 0: all True  
Row 0: all False except (0, 0) entry
Subset Sum

**Problem:** Given a list of positive integers \(a[1..n]\) and an integer \(t\), is there some subset of \(a\) that sums to exactly \(t\)?

**Example:** \(a = [12, 1, 3, 8, 20, 50]\)

**STEP 1: Define subtasks**

For \(i=1..n\), \(s=1..t\),

\[ S(i,s) = \begin{cases} \text{True}, & \text{if some subset of } S[1..i] \text{ adds to } s \\ \text{False}, & \text{otherwise} \end{cases} \]

Output = \(S(n, t)\)

**STEP 2: Express recursively**

If \(a[i] \leq s\),

\[ S(i,s) = S(i - 1, s - a[i]) \text{ OR } S(i - 1, s) \]

Else: \(S(i, s) = S(i - 1, s)\)

**STEP 3: Order of subtasks**

Row by row, increasing column

Base Case:

Column 0: all True
Row 0: all False except \((0, 0)\) entry
Subset Sum

**Problem:** Given a list of positive integers $a[1..n]$ and an integer $t$, is there some subset of $a$ that sums to exactly $t$?

**Example:** $a = [12, 1, 3, 8, 20, 50]$

**STEP 1: Define subtasks**
For $i=1..n$, $s=1..t$,

- $S(i,s) = \text{True}$, if some subset of $S[1..i]$ adds to $s$
- $= \text{False}$, otherwise

Output = $S(n, t)$

**STEP 2: Express recursively**
If $a[i] \leq s$,

- $S(i,s) = S(i-1, s-a[i]) \text{ OR } S(i-1, s)$

Else: $S(i,s) = S(i-1, s)$

**STEP 3: Order of subtasks**
Row by row, increasing column

- $t = 44$ True
- $t = 14$ False

<table>
<thead>
<tr>
<th>$i$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:**
- Column 0: all True
- Row 0: all False except (0, 0) entry
Subset Sum

**Problem:** Given a list of positive integers a[1..n] and an integer t, is there some subset of a that sums to exactly t?

**Example:** a = [ 12, 1, 3, 8, 20, 50 ]

**STEP 1: Define subtasks**
For i=1..n, s=1..t,
S(i,s) = True, if some subset of S[1..i] adds to s
    = False, ow
Output = S(n, t)

**STEP 2: Express recursively**
If a[i] ≤ s,
    S(i,s) = S(i - 1, s - a[i]) OR S(i - 1, s)
Else: S(i, s) = S(i - 1, s)

**STEP 3: Order of subtasks**
Row by row, increasing column

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:**
Column 0: all True
Row 0: all False except (0, 0) entry

t = 44  True  t = 14  False
**Subset Sum**

**Problem:** Given a list of positive integers \( a[1..n] \) and an integer \( t \), is there some subset of \( a \) that sums to exactly \( t \)?

**Example:** \( a = [12, 1, 3, 8, 20, 50] \)

**STEP 1: Define subtasks**

For \( i=1..n \) and \( s=1..t \),

\[
S(i,s) = \begin{cases} 
  \text{True, if some subset of } S[1..i] \\
  \text{adds to } s \\
  \text{False, otherwise}
\end{cases}
\]

Output = \( S(n, t) \)

**STEP 2: Express recursively**

If \( a[i] \leq s \),

\[
S(i,s) = S(i-1, s-a[i]) \text{ OR } S(i-1, s)
\]

Else: \( S(i, s) = S(i-1, s) \)

**STEP 3: Order of subtasks**

Row by row, increasing column

---

**Base Case:**

Column 0: all True
Row 0: all False except (0, 0) entry

---

<table>
<thead>
<tr>
<th>( i )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>5</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>6</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

\( t = 44 \) True \( t = 14 \) False
Subset Sum

Problem: Given a list of positive integers $a[1..n]$ and an integer $t$, is there some subset of $a$ that sums to exactly $t$?

Example: $a = [12, 1, 3, 8, 20, 50]$

STEP 1: Define subtasks
For $i=1..n$, $s=1..t$,

$S(i,s) =$ True, if some subset of $a[1..i]$ adds to $s$

= False, otherwise

Output = $S(n, t)$

STEP 2: Express recursively
If $a[i] \leq s$,

$S(i,s) = S(i - 1, s - a[i])$ OR $S(i - 1, s)$

Else: $S(i, s) = S(i - 1, s)$

STEP 3: Order of subtasks
Row by row, increasing column

Base Case:
Column 0: all True
Row 0: all False except (0, 0) entry

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$t = 44 \quad$ True  \quad t = 14 \quad$ False
Subset Sum

**Problem:** Given a list of positive integers \( a[1..n] \) and an integer \( t \), is there some subset of \( a \) that sums to exactly \( t \)?

**Example:** \( a = [12, 1, 3, 8, 20, 50] \)

**STEP 1: Define subtasks**
For \( i=1..n \), \( s=1..t \),

\[
S(i,s) = \text{True, if some subset of } S[1..i] \text{ adds to } s
\]

= \text{ False, otherwise}

Output = \( S(n, t) \)

**STEP 2: Express recursively**
If \( a[i] \leq s \),

\[
S(i,s) = S(i - 1, s - a[i]) \text{ OR } S(i - 1, s)
\]
Else: \( S(i, s) = S(i - 1, s) \)

**STEP 3: Order of subtasks**
Row by row, increasing column

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:**
Column 0: all True
Row 0: all False except \((0, 0)\) entry
Subset Sum

**Problem:** Given a list of positive integers \(a[1..n]\) and an integer \(t\), is there some subset of \(a\) that sums to exactly \(t\)?

**Example:** \(a = [12, 1, 3, 8, 20, 50]\)

**STEP 1: Define subtasks**
For \(i=1..n, s=1..t\),

\[S(i,s) = \begin{cases} 
\text{True, if some subset of } S[1..i] \\
\text{adds to } s \\
\text{False, otherwise} 
\end{cases}\]

Output = \(S(n, t)\)

**STEP 2: Express recursively**
If \(a[i] \leq s\),

\[S(i,s) = S(i-1, s-a[i]) \text{ OR } S(i-1, s)\]

Else: \(S(i, s) = S(i-1, s)\)

**STEP 3: Order of subtasks**
Row by row, increasing column

<table>
<thead>
<tr>
<th>(i)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

**Base Case:**
Column 0: all True
Row 0: all False except \((0, 0)\) entry

\(t = 44\) \ True \ \(t = 14\) \ False
Subset Sum

**Problem:** Given a list of positive integers \( a[1..n] \) and an integer \( t \), is there some subset of \( a \) that sums to exactly \( t \)?

**Example:** \( a = [12, 1, 3, 8, 20, 50] \)

**STEP 1: Define subtasks**
For \( i=1..n \), \( s=1..t \),
\( S(i,s) = \) True, if some subset of \( S[1..i] \) adds to \( s \)  
= False, ow
Output = \( S(n, t) \)

**STEP 2: Express recursively**
If \( a[i] \leq s \),  
\( S(i,s) = S(i - 1, s - a[i]) \) OR \( S(i - 1, s) \)
Else: \( S(i, s) = S(i - 1, s) \)

**STEP 3: Order of subtasks**
Row by row, increasing column

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Base Case:**
Column 0: all True  
Row 0: all False except (0, 0) entry

<table>
<thead>
<tr>
<th></th>
<th>t = 44</th>
<th>True</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>t = 14</td>
<td>False</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Subset Sum

**Problem:** Given a list of positive integers \( a[1..n] \) and an integer \( t \), is there some subset of \( a \) that sums to exactly \( t \)?

**Example:** \( a = [12, 1, 3, 8, 20, 50] \)

**STEP 1: Define subtasks**

For \( i=1..n \), \( s=1..t \),

\[
S(i,s) = \begin{cases} 
  \text{True, if some subset of } S[1..i] \\
  \text{adds to } s \\
  \text{False, otherwise}
\end{cases}
\]

Output = \( S(n, t) \)

**STEP 2: Express recursively**

If \( a[i] \leq s \),

\[
S(i,s) = S(i - 1, s - a[i]) \text{ OR } S(i - 1, s)
\]

Else: \( S(i, s) = S(i - 1, s) \)

**STEP 3: Order of subtasks**

Row by row, increasing column

### Running Time

\( O(nt) \)

How to reconstruct the subset?

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Subset Sum

**Problem:** Given a list of positive integers \( a[1..n] \) and an integer \( t \), is there some subset of \( a \) that sums to exactly \( t \)?

**STEP 1: Define subtasks**
For \( i=1..n \), \( s=1..t \),
\[ S(i,s) = \begin{cases} \text{True, if some subset of } S[1..i] \text{ adds to } s \\ \text{False, ow} \end{cases} \]
Output = \( S(n, t) \)

**STEP 2: Express recursively**
If \( a[i] \leq s \),
\[ S(i,s) = S(i - 1, s - a[i]) \text{ OR } S(i - 1, s) \]
Else: \( S(i, s) = S(i - 1, s) \)

**STEP 3: Order of subtasks**
Row by row, increasing column

**Reconstructing the subset:**
Define an array \( D(i, s) \).
If \( S(i, s) = \text{True} \), and \( S(i-1, s-a[i]) = \text{True} \)
\[ D(i, s) = (i - 1, s - a[i]) \]
Else: \( D(i, s) = (i - 1, s) \) ow.

Reconstruct the subset by following the pointers from \( D(n,t) \)

**Running Time** = \( O(nt) \)
Dynamic Programming

- String Reconstruction
- Longest Common Subsequence
- Edit Distance
- Subset Sum
- Independent Set in a Tree
Independent Set

**Independent Set:** Given a graph $G = (V, E)$, a subset of vertices $S$ is an independent set if there are no edges between them.

**Max Independent Set Problem:** Given a graph $G = (V, E)$, find the largest independent set in $G$.

**Max Independent Set** is a notoriously hard problem! We will look at a restricted case, when $G$ is a tree.
Max. Independent Set in a Tree

A set of nodes is an **independent set** if there are no edges between the nodes.

**Two Cases at node u:**
1. Don’t include u
2. Include u, and don’t include its children
Max. Independent Set in a Tree

A set of nodes is an **independent set** if there are no edges between the nodes.

**STEP 1: Define subtask**

\[ I(u) = \text{size of largest independent set in subtree rooted at } u \]

We want \( I(r) \), where \( r = \text{root} \)

**STEP 2: Express recursively**

\[
I(u) = \max\left\{ \sum_{\text{children } w \text{ of } u} I(w), 1 + \sum_{\text{grandchildren } w \text{ of } u} I(w) \right\}
\]

Base case: for leaf nodes, \( I(u) = 1 \)

**STEP 3: Order of subtasks**

Reverse order of distance from root; use BFS!

Two Cases at node \( u \):
1. Don't include \( u \)
2. Include \( u \), and don’t include its children
Max. Independent Set in a Tree

A set of nodes is an **independent set** if there are no edges between the nodes.

**STEP 1: Define subtask**

\[ l(u) = \text{size of largest independent set in subtree rooted at } u \]

We want \( l(r) \), where \( r = \text{root} \)

**STEP 2: Express recursively**

\[
l(u) = \max \left\{ \sum_{\text{children } w \text{ of } u} I(w), \ 1 + \sum_{\text{grandchildren } w \text{ of } u} I(w) \right\}
\]

Base case: for leaf nodes, \( l(u) = 1 \)

**STEP 3: Order of subtasks**

Reverse order of distance from root; use BFS!

**Running Time: \( O(n) \)**

Edge \((u, v)\) is examined in Step 2 at most twice:

1. \( v \) is a child of \( u \)
2. \( v \) is a grandchild of \( u \)’s parent

There are \( n-1 \) edges in a tree on \( n \) nodes.