**Longest Common Subsequence**

Much overlap b/w CS + Biology

Compare DNA of 2 organisms:

\[ S_1 = \text{ACCGGTCACCT} \]
\[ S_2 = \text{TTCGGAATCTG} \]

Many ways to measure similarity:

- \( \# \) of identical chars in the same index
- \( \# \) of substring in common
- longest subsequence (string of sequential, not nec adjacent characters) in common

For \( x \):

\[ S_1 = \text{ACCGGTCACCT} \]
\[ S_2 = \text{TTCGGAATCTG} \]

\[ \text{LCS} = \text{CCGTCT} \]

Formally:

Sequence \( X = \langle x_1, x_2, \ldots, x_m \rangle \)

\( Z \) is a length- \( k \) subsequence of \( X \) if (\( Z \) a strictly increasing sequence of indices) there exists \( i_1 < i_2 < \ldots < i_k \) s.t. \( Z = \langle x_{i_1}, x_{i_2}, \ldots, x_{i_k} \rangle \).

Intuitively, \( Z \) is a subsequence of \( X \) if \( Z \) is made up of characters from \( X \) in some order as in \( X \).
Which are subsequences of $S$? $A$, $GCC$, $CIO$, $GA$

Longest Common Subsequence (LCS) Problem

Given two strings $X = \langle x_1, x_2, \ldots, x_m \rangle$ and $Y = \langle y_1, y_2, \ldots, y_n \rangle$

Find length (subsequences later) of max-length LCS of $XY$.

Denote LCS

Greedy? unclear what a greedy approach would be. Ideas?

Brute Force:

1. Enumerate all subsequences of $X$ and all subsequences of $Y$.
   
2. For each character, either include
   
3. Hash one list into table. Hash the other while storing collisions
   $O(2^m + 2^n)$ collisions indicate common subsequences.

Dynamic Programming

Good indication that DP works? Optimal Substructure property

$\Rightarrow$ LCS of entire string depends on LCS of substrings

Many ways to consider substrings:

- divide $X$, $Y$ arbitrarily $\Leftarrow$ not clear how to then "combine"
- grow $X$, $Y$ from start (or end) $\Leftarrow$ we will use start
- prefix - any leading (starting from beginning) contiguous (adjacent) substring

Example:

$X = BACDB$

$Y = BDEEB$
Denote:
\[ \overline{X}_i \text{: length-}i\text{ prefix of } X \quad \text{ex: } \overline{X}_3 = BAC \]
\[ \overline{Y}_i \text{: length-}i\text{ prefix of } Y \quad \text{ex: } \overline{Y}_4 = BDEB \]

Need to use solution for shorter prefixes to find solutions for longer prefixes.

Initial idea: compare same-length prefixes of \( X, Y \). OK?

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

\[ Y = B D E B \uparrow \]

\[ \uparrow \uparrow \uparrow \uparrow \uparrow \]

LCS: B B B B B But LCS is BDB!

So need to compare each prefix of \( X \) with each prefix of \( Y \).

\( (X) \): B BA BAC ... BACDB ... BACD ... BACDB

\( (Y) \): B B B B B BD BDEB

\[ \uparrow \uparrow \uparrow \uparrow \uparrow \]

LCS: B B B B B BD BDB.
Now...

\[ X = \text{BACDB} \]
\[ Y = \text{BD} \overline{EB} \]

Suppose you knew \( |LCS| \) of these prefixes:
\( |LCS(\overline{X_4}, \overline{Y_3})| = 2 \) (BD)

How to use next chars in \( X, Y \) to find \( LCS(\overline{X_5}, \overline{Y_4}) \)?

2 cases based on next chars \( X_5, Y_4 \)

Case 1: If \( x_i = y_j \)

\[ X = \overline{X_{i-1}} \overline{B} \quad ? \quad \text{Can match } x_i + y_j \text{ to get a longer LCS.} \]
\[ Y = \overline{Y_{j-1}} \overline{B} \]

\[ |LCS(\overline{X_i}, \overline{Y_j})| = LCS(\overline{X_{i-1}}, \overline{Y_{j-1}}) + 1 \]

Ques: What if \( x_i \) also matches with a previous char in \( Y \)?

Ex: \( X = \text{BACD} \overline{N} \Rightarrow \text{Better to match} \quad X = \overline{BA} \overline{CD} \overline{N} \overline{E} \quad x_i (\overline{N}) \text{ with previous} \quad Y = \overline{BN} \overline{E} \overline{N} \overline{N} \text{ in } Y \text{ only if} \overline{N} \) in \( X \) only if

Would match \( x_i (\overline{N}) \text{ with previous char in } Y \) only if there is a future character in \( X \) that can be matched with another character in \( Y \).
But we will discover this in a future computation.

\[ \bar{x}_5 \]

\[ \bar{y}_3 \]

\[ \begin{array}{c}
\text{ex: For } (\bar{x}_6 = \overline{BA C D N E}) \\
(\bar{y}_3 = \overline{B N E N}) \\
\uparrow y_4
\end{array} \]

| LCS1 = 1, new LCS = 2 |

\[ \text{LCS}(\bar{x}_6, \bar{y}_3) \text{ computed after LCS}(\bar{x}_5, \bar{y}_4) \]

Would match current X (x₅) with a previous N (y₂) only if there is some future char in X that can be matched with a char in Y (e.g. E in the example above).

But we would discover this in a future computation (e.g. when computing for the E's in the example above).

**Case 2: \( x_i \neq y_j \)**

\[ x = \overline{B A C D E} \]

\[ \overline{B D E B} \]

\[ \text{LCS won't improve!} \]
LCS \( (x_i, y_j) \) may end in \( x_i \); LCS \( (x_i, y_j) \) may end in \( y_j \); LCS \( (x_i, y_j) \) can't end in both \( x_i, y_j \).

\[
\begin{align*}
X &= \text{BACDE} \\
Y &= \text{BDEB} \\
\text{LCS} &= \text{BDE} \\
\end{align*}
\]

\[
\begin{align*}
X &= \text{BACDE} \\
Y &= \text{BCFD} \\
\text{LCS} &= \text{BCD} \\
\end{align*}
\]

\[
\begin{align*}
X &= \text{BACDE} \\
Y &= \text{BDEC} \\
\text{LCS} &= \text{BDE} \\
\end{align*}
\]

But these have already been found in a previous computation!

Or, LCS ends in neither.

\[
\begin{align*}
X &= \text{BACDK} \\
Y &= \text{BDEJ} \\
\text{LCS} &= \text{BDE} \\
\end{align*}
\]

\[
|\text{LCS}(\overline{x_i}, \overline{y_j})| = \max \left\{ |\text{LCS}(\overline{x_i}, \overline{y_{j-1}})|, |\text{LCS}(\overline{x_{i-1}}, \overline{y_j})|, |\text{LCS}(\overline{x_{i-1}}, \overline{y_{j-1}})| \right\}
\]

will always be ≤ (A, B)
Dynamic Programming Formulation:

Recall X is length m, Y is length n

What size array to store values? $m \times n = (m+1) \times (n+1)$

For every prefix of X, find LCS with every prefix of Y.

$C[i][j] = |LCS(\overline{X_i}, \overline{Y_j})|$ (length)

\[
C[i][j] = \begin{cases} 
10^{-1} & \text{if } i = 0 \text{ or } j = 0; \\
C[i-1][j-1]+1 & \text{if } x_i = y_j \\
\max(C[i-1][j], C[i][j-1]) & \text{if } x_i \neq y_j 
\end{cases}
\]

Base Case? $i = 0$ or $j = 0$.

To fill matrix:

$C[0][0...m] = 0$, $C[0...n][0] = 0$

for ($i = 1$ to $m$) // every prefix of X

for ($j = 1$ to $n$) // "" // Y

$C[i][j] = \langle$based on D.P. formulation$\rangle$

Final LCS length? In $C[m][n]$

Run Time? $O(mn)$. 
How to find actual LCS?

Quick example: ...
Quick example will show how to find subsequence.

**Ex. 1:** \( X = B A C D B \quad m = 5 \quad n = 4 \)
\[ Y = B D E B \]

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>D</th>
<th>E</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

arbitrarily break ties between \( \leftarrow \) and \( \uparrow \) by choosing \( \leftarrow \).

Find subsequence?

Keep 2D array of arrows that indicate where best LCS was from.

Notice: Only indices with \( \uparrow \) indicate that corresponding char is in LCS. So store these chars.

### LCS: BDP
Get Subsequence()

1. Start at [Em][en].
2. Follow arrows until -
3. If arrow is ▲, output character.
4. Return reverse of outputted string.
LCS Problem

another example:

\[ X = BC\text{AAD} \quad m = 5 \quad n = 4 \]
\[ Y = A\text{AACD} \]

\[
\begin{array}{cccc}
  & A & A & C & D \\
\hline
  O & O & O & O & O \\
  B & O & O & O & O \\
  C & O & O & O & * \\
  A & O & O & O & * \\
  D & 0 & 1 & 2 & 2 \\
\end{array}
\]

"incorrect" match

Find sequence:
Start at \([m][n]\)
For every \( \downarrow \) output character (in reverse)

\[ \text{LCS}(X_5, Y_4) = \text{AAD} \]
Find actual subsequences? Quick ex!

ex 0: \( X = BCEAD \) \( m = 5, n = 3 \)

\( Y = ADA \)

1. *Note: First A's in X, Y matched
2. *although "wrong" A's matched later
3. *where we correctly decide that matching the first A in X yields longer LCS

How to find actual LCS?

- Keep arrows that indicate where best LCS was from.
- Notice: Only indices with \( \wedge \) indicate that corresponding char is in LCS. So store these chars.

1. Start at \( C [m|J|e|n|] \)

2. Follow arrows.

3. If see "\( \wedge \)" output character

4. LCS is reverse of outputted string
Quick example will show how to find subsequence.

\[ X = BCEAD \quad m = 5 \quad n = 4 \]
\[ Y = BDEB \]

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>