14.2.6
Longest Common Subsequence Problem

LCS Problem

Definition 14.7.

LCS between two strings $\boldsymbol{X}$ and $\boldsymbol{Y}$ is the length of longest common subsequence between $\boldsymbol{X}$ and $\boldsymbol{Y}$.

ABAZDC
BACBAD

ABAZDC
BACBAD

Example 14.8.

LCS between ABAZDC and BACBAD is 4 via ABAD

Derive a dynamic programming algorithm for the problem.

LCS Problem

Definition 14.7.

LCS between two strings $\boldsymbol{X}$ and $\boldsymbol{Y}$ is the length of longest common subsequence between $\boldsymbol{X}$ and $\boldsymbol{Y}$.

$$\boldsymbol{ABAZDC} \qquad\qquad \boldsymbol{ABAZDC}$$
$$\boldsymbol{BACBAD} \qquad\qquad \boldsymbol{BACBAD}$$

Example 14.8.

LCS between ABAZDC and BACBAD is 4 via ABAD

Derive a dynamic programming algorithm for the problem.

LCS Problem

Definition 14.7.

LCS between two strings X and Y is the length of longest common subsequence between X and Y.

ABAZDC ***ABAZDC***
BACBAD ***BACBAD***

Example 14.8.

LCS between ABAZDC and BACBAD is 4 via ABAD

Derive a dynamic programming algorithm for the problem.

LCS recursive definition

$A[1..n]$, $B[1..m]$: Input strings.

$$LCS(i,j) = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ \max\begin{pmatrix} LCS(i-1,j), \\ LCS(i,j-1) \end{pmatrix} & A[i] \neq B[j] \\ \max\begin{pmatrix} LCS(i-1,j), \\ LCS(i,j-1), \\ 1 + LCS(i-1,j-1) \end{pmatrix} & A[i] = B[j] \end{cases}$$

Similar to edit distance... $O(nm)$ time algorithm $O(m)$ space.

LCS recursive definition

$A[1..n]$, $B[1..m]$: Input strings.

$$LCS(i,j) = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ \max\begin{pmatrix} LCS(i-1,j), \\ LCS(i,j-1) \end{pmatrix} & A[i] \neq B[j] \\ \max\begin{pmatrix} LCS(i-1,j), \\ LCS(i,j-1), \\ 1 + LCS(i-1,j-1) \end{pmatrix} & A[i] = B[j] \end{cases}$$

Similar to edit distance... $O(nm)$ time algorithm $O(m)$ space.

Longest common subsequence is just edit distance for the two sequences...

$\boldsymbol{A}, \boldsymbol{B}$: input sequences
Σ: "alphabet" all the different values in $\boldsymbol{A}$ and $\boldsymbol{B}$

$$\forall \boldsymbol{b}, \boldsymbol{c} \in \Sigma : \boldsymbol{b} \neq \boldsymbol{c} \qquad \boldsymbol{COST}[\boldsymbol{b}][\boldsymbol{c}] = +\infty.$$
$$\forall \boldsymbol{b} \in \Sigma \qquad \boldsymbol{COST}[\boldsymbol{b}][\boldsymbol{b}] = \mathbf{1}$$

$\mathbf{1}$: price of deletion of insertion of a single character

Length of longest common subsequence $= \boldsymbol{m} + \boldsymbol{n} - \mathrm{ed}(\boldsymbol{A}, \boldsymbol{B})$

Longest common subsequence is just edit distance for the two sequences...

$\boldsymbol{A}, \boldsymbol{B}$: input sequences
Σ: "alphabet" all the different values in $\boldsymbol{A}$ and $\boldsymbol{B}$

$$\begin{array}{ll} \forall \boldsymbol{b}, \boldsymbol{c} \in \Sigma : \boldsymbol{b} \neq \boldsymbol{c} & \boldsymbol{COST}[\boldsymbol{b}][\boldsymbol{c}] = +\infty. \\ \forall \boldsymbol{b} \in \Sigma & \boldsymbol{COST}[\boldsymbol{b}][\boldsymbol{b}] = \mathbf{1} \end{array}$$

$\mathbf{1}$: price of deletion of insertion of a single character

Length of longest common subsequence $= \boldsymbol{m} + \boldsymbol{n} - \mathrm{ed}(\boldsymbol{A}, \boldsymbol{B})$

THE END

...

(for now)