Matrix Chain Order

\(P_0, \ldots, P_k\)

find optimal order of multiplying

\(A_1 \ldots A_k\)

Table \(m[1..k, 1..k]\)

\(m[i..j]\) is optimal number of scalar mults to compute

\(A_i \ldots A_j\) \((i \leq j)\)

\(m[i..i] = 0\)

if \(i < j\) then

\(m[i..j] = \min_{i \leq \ell < j} \left[ m[i..\ell] + m[\ell+1..j] + P_{i-1}P_{\ell}P_j \right] \)

\((A_i \ldots A_\ell) (A_{\ell+1} \ldots A_j)\)

\(m[i..j]\) incrementally in order of increasing \(j-i = 0, 1, 2, \ldots\)
How to find the actual multiplication order: Remember the value of $\ell$ that achieves the minimum in each case:

$$m[i,j] = \min_{\ell} \text{ minimizing}$$

Store value of $\ell$ in $m[j,i]$ ($j > i$) (otherwise unused).

**Ex:** Find $\alpha$-values for Knapsack.

**Longest Common Subseq**

- **hence**

**LCS**

Given two sequences

$$X = \langle x_1, \ldots, x_k \rangle$$

$$Y = \langle y_1, \ldots, y_l \rangle$$

where $x_i$ and $y_j$ are symbols from same fixed alphabet.

A subsequence of $X$
is some sequence
\[ <x_{i_1}, x_{i_2}, \ldots, x_{i_n} > \]
where
\[ 1 \leq i_1 < i_2 < \ldots < i_n \leq k \]
Similarly for \( Y \).

A common subsequence of \( X, Y \) is a sequence that is a subseq. of both \( X \) and \( Y \).

A LCS of \( X \) & \( Y \) is a common subseq of max length.

Recursive Rules for the length of an LCS of \( X \) & \( Y \):
If either \( X \) or \( Y \) is the empty seq.
then length of LCS = 0.
Assume \( X, Y \) both nonempty.
\[ X = <x_1, \ldots, x_{k-1}, x_k> \]
\[ Y = <y_1, \ldots, y_{e-1}, y_e> \]
Case 1: $x_k = y_k$.
Then any LCS of $xy$ must include this last symbol.

$LCS = \langle x_1, \ldots, x_m \rangle$

and

$m = x_k = y_k$

$LCS = \langle x_1, \ldots, x_{m-1} \rangle$ must be an LCS of

$LCS = \langle x_1, \ldots, x_{k-1} \rangle$ and

$LCS = \langle y_1, \ldots, y_{e-1} \rangle$

So, $LCS(xy)$ has length

$\text{length}(LCS(\langle x_1, \ldots, x_{k-1} \rangle, \langle y_1, \ldots, y_{e-1} \rangle)) + 1$

Case 2: $x_k \neq y_k$.
Any LCS of $x$ & $y$ is either a subseq of

$LCS = \langle x_1, \ldots, x_{k-1} \rangle = X$

or of $LCS = \langle y_1, \ldots, y_{e-1} \rangle = Y$

An LCS of $x$ & $y$ is either a common subseq of $X$ & $Y$ or of $x$ and $y$. 
The length of an LCS of \( X \) & \( Y \) is
the longer of:

- LCS length of \( X', Y \)
- LCS length of \( X, Y' \)

\( c[0..k, 0..l] \) such that

\[
c[i, j] = \text{length of an LCS for } \langle x_1, \ldots, x_i \rangle \text{ and } \langle y_1, \ldots, y_j \rangle.
\]

\[
c[i, j] = \begin{cases} 
0 & \text{if } i = 0 \text{ or } j = 0, \text{ else} \\
\max\{c[i-1, j-1] + 1 \text{ if } x_i = y_j, c[i-1, j], c[i, j-1]\} & \text{otherwise}
\end{cases}
\]

Time = \( \Theta(kl) \) = Space
Space = \( \Theta(\min(k, l)) \)

Greedy Algorithms:
Finding a global optimum by repeatedly finding local optima
Huffman Codes

Data compression

"variable-length prefix code"

\[ a = 00 \quad \text{no codeword} \]

\[ b = 010 \quad \text{is a proper prefix of any other codeword} \]

\[ c = 011 \]

\[ d = 100 \]

\[ e = 11 \]

\[ 0 1 1 0 1 0 0 1 0 0 \]

Describe a (binary) prefix code with a binary tree:

leaves labeled with distinct letters, each internal node has 2 children
Assume:
alphabet \( C = \{c_1, \ldots, c_n\} \)
each letter \( c_i \) in \( C \) has
a frequency \( f[c_i] \)
(nonnegative integer)
= \# of occurrences of
\( c_i \) in the file to be
compressed.

<table>
<thead>
<tr>
<th>letter</th>
<th>freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>500</td>
</tr>
<tr>
<td>b</td>
<td>125</td>
</tr>
<tr>
<td>c</td>
<td>250</td>
</tr>
<tr>
<td>d</td>
<td>125</td>
</tr>
</tbody>
</table>

Fixed-length encoding:
2 bits per letter
file is 2000 bits.

variable-length prefix
code:

file length
= 500 \cdot 1 + 250 \cdot 2
+ 125 \cdot 3 + 125 \cdot 3
= 1750 < 2000
A Huffman code is an optimal binary prefix code, given symbols and their frequencies.

Given a tree \( T \) for a prefix code for \( C \), the cost of \( T \) is defined as

\[
B(T) = \sum_{i=1}^{n} f[c_i] d(c_i)
\]

where \( d(c_i) \) is the depth of \( c_i \) in \( T \)

\( = \# \) of bits for \( c_i \)

\( B(T) = \) total length of encoded file.

Huffman tree = tree with min cost.