An input sensitive online algorithm for LCS computation

Heikki Hyyrö

Department of Computer Sciences
University of Tampere, Finland
An input sensitive online algorithm for LCS computation

The basic problem setting:
An input sensitive online algorithm for LCS computation

The basic problem setting:

- Input: strings $A$ and $B$ from an alphabet of size $\sigma$
  - The lengths are $n$ and $m$ with $n \geq m$
An input sensitive online algorithm for LCS computation

The basic problem setting:

- Input: strings $A$ and $B$ from an alphabet of size $\sigma$
  - The lengths are $n$ and $m$ with $n \geq m$
- The task: compute the length of a longest common subsequence (LCS) of $A$ and $B$
The basic problem setting:

- Input: strings $A$ and $B$ from an alphabet of size $\sigma$
  - The lengths are $n$ and $m$ with $n \geq m$
- The task: compute the length of a longest common subsequence (LCS) of $A$ and $B$
  - We denote the length by $L$
  - E.g. if $A =$ “Prague” and $B =$ “charge”
An input sensitive online algorithm for LCS computation

The basic problem setting:

- Input: strings $A$ and $B$ from an alphabet of size $\sigma$
  - The lengths are $n$ and $m$ with $n \geq m$
- The task: compute the length of a longest common subsequence (LCS) of $A$ and $B$
  - We denote the length by $L$
  - E.g. if $A = \text{“Prague”}$ and $B = \text{“charge”}$, then $L = \text{LLCS}(A,B) = 3$
An input sensitive online algorithm for LCS computation

The basic problem setting:

- Input: strings $A$ and $B$ from an alphabet of size $\sigma$
  - The lengths are $n$ and $m$ with $n \geq m$
- The task: compute the length of a longest common subsequence (LCS) of $A$ and $B$
  - We denote the length by $L$
  - E.g. if $A = \text{“Prague”}$ and $B = \text{“charge”}$, then $L = LLCS(A,B) = 3$

LLCS is a dual of indel edit distance:

- $ed_{id}(A,B) = n + m - 2LLCS(A,B)$
An input sensitive online algorithm for LCS computation

An online algorithm:
An input sensitive online algorithm for LCS computation

An online algorithm:

- Preprocesses only \( A \), and can then read \( B \) one character at a time
An input sensitive online algorithm for LCS computation

An online algorithm:

- Preprocesses only $A$, and can then read $B$ one character at a time
- Useful e.g. in one-against-many comparison or neighborhood generation
An input sensitive online algorithm for LCS computation

An online algorithm:

- Preprocesses only $A$, and can then read $B$ one character at a time
- Useful e.g. in one-against-many comparison or neighborhood generation
An input sensitive online algorithm for LCS computation

Many input-sensitive algorithms exist
An input sensitive online algorithm for LCS computation

Many input-sensitive algorithms exist, e.g.:

An input sensitive online algorithm for LCS computation

Many input-sensitive algorithms exist, e.g.:


An input sensitive online algorithm for LCS computation

Many input-sensitive algorithms exist, e.g.:


- $O(\sigma n + \min\{mL, L(n - L)\})$: Rick: A New Flexible Algorithm for the Longest Common Subsequence Problem, *CPM 1995*
An input sensitive online algorithm for LCS computation

Many input-sensitive algorithms exist, e.g.:

- $O(\sigma n + \min\{mL, L(n - L)\})$: Rick: A New Flexible Algorithm for the Longest Common Subsequence Problem, *CPM 1995*
- $O(\sigma n + \min\{mL, L(n - L)\})$: Goeman & Clausen: A New Practical Linear Space Algorithm for the Longest Common Subsequence Problem, *Kybernetika* (2002)
An input sensitive online algorithm for LCS computation
Classic solution: $\mathcal{O}(mn)$ dynamic programming
An input sensitive online algorithm for LCS computation

Classic solution: $O(mn)$ dynamic programming

- A table $D$, where $D[i, j] = \text{LLCS}(A_{1..i}, B_{1..j})$ and
  
  $D[i, j] = 1 + D[i-1, j-1]$, if $A_i = B_j$,
  
  else $\max\{D[i, j-1], D[i-1, j]\}$

<table>
<thead>
<tr>
<th></th>
<th>c</th>
<th>h</th>
<th>a</th>
<th>r</th>
<th>g</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>g</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>u</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
An input sensitive online algorithm for LCS computation
Incremental encoding of columns of $D$
An input sensitive online algorithm for LCS computation

Incremental encoding of columns of \( D \)

\[
\begin{array}{cccc}
\text{A} & \text{T} & \text{C} \\
\hline
\text{Regular:} & 0 & 0 & 0 & 0 \\
\text{T} & 0 & 0 & 1 & 1 \\
\text{A} & 0 & 1 & 1 & 1 \\
\text{C} & 0 & 1 & 1 & 2 \\
\end{array}
\quad
\begin{array}{cccc}
\text{A} & \text{T} & \text{C} \\
\hline
\text{Incremental:} & 0 & 0 & 0 & 0 \\
\text{T} & 0 & 0 & 1 & 1 \\
\text{A} & 0 & 1 & 0 & 0 \\
\text{C} & 0 & 0 & 0 & 1 \\
\end{array}
\]

\( \Delta[i,j] = D[i,j] - D[i-1,j] \)

\( D[i,j] = \sum_{k=1}^{i} \Delta[k,j] \)

\[
> \{|\Delta[k,j] : 1 \leq k \leq j \land \Delta[k,j] = 1\} |
\]
An input sensitive online algorithm for LCS computation

Incremental encoding of columns of $D$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>T</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular:</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>T</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incremental:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>T</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

- $\Delta[i, j] = D[i, j] - D[i-1, j]$
- $D[i, j] = \sum_{k=1}^{i} \Delta[k, j]$
- $\supset \{\Delta[k, j] : 1 \leq k \leq j \land \Delta[k, j] = 1\}$
- Store only increment points $i$ where $\Delta[i, j] = 1$
An input sensitive online algorithm for LCS computation

Incremental encoding of columns of $D$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>T</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular:</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>T</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incremental:</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

- $\Delta[i, j] = D[i, j] - D[i-1, j]$
- $D[i, j] = \sum_{k=1}^{i} \Delta[k, j]$

\[ \triangleright \left\{ \Delta[k, j] : 1 \leq k \leq j \land \Delta[k, j] = 1 \right\} \]

- Store only increment points $i$ where $\Delta[i, j] = 1 \Rightarrow$ each column $j$ takes $D[n, j] \leq L = \text{LLCS}(A, B)$ space
An input sensitive online algorithm for LCS computation

Incremental encoding of columns of $D$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>T</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular:</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>T</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incremental:</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

- $\Delta[i, j] = D[i, j] - D[i-1, j]$
- $D[i, j] = \Sigma_{k=1}^{i} \Delta[k, j]$
  
  \[ |\{\Delta[k, j] : 1 \leq k \leq j \land \Delta[k, j] = 1}\}| 

- Store only increment points $i$ where $\Delta[i, j] = 1 \Rightarrow$ each column $j$ takes $D[n, j] \leq L = \text{LLCS}(A, B)$ space
- let $I_x[j]$ denote the $x$th increment point in column $j$
An input sensitive online algorithm for LCS computation

How to compute increment points for column $j$?
An input sensitive online algorithm for LCS computation

How to compute increment points for column $j$?
An input sensitive online algorithm for LCS computation

How to compute increment points for column $j$?

<table>
<thead>
<tr>
<th></th>
<th>$j-1$</th>
<th>$j$</th>
<th>$j-1$</th>
<th>$j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x-2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x-1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x-1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x-1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Prague Stringology Conference '09
An input sensitive online algorithm for LCS computation

How to compute increment points for column $j$?

$I_x[j] = \min\{i : i > I_{x-1}[j-1] \land (A_i = B_j \lor i = I_x[j-1])\}$

Prague Stringology Conference '09
An input sensitive online algorithm for LCS computation

\[ I_x[j] = \min\{i : i > I_{x-1}[j-1] \land (A_i = B_j \lor i = I_x[j-1]) \} \]
An input sensitive online algorithm for LCS computation

\[ I_x[j] = \min\{i : i > I_{x-1}[j-1] \land (A_i = B_j \lor i = I_x[j-1]) \} \]

How to locate the relevant match \( A_i = B_j \) quickly?
An input sensitive online algorithm for LCS computation

\[ I_x[j] = \min\{ i : i > I_{x-1}[j-1] \land (A_i = B_j \lor i = I_x[j-1]) \} \]

How to locate the relevant match \( A_i = B_j \) quickly?

- Precompute a \( \sigma \times n \) table \( NM \), where

\[
NM[\lambda, k] = \min\{ i : i > k \land (A_i = \lambda \lor i = n + 1) \}
\]
An input sensitive online algorithm for LCS computation

\[ I_x[j] = \min \{ i : i > I_{x-1}[j-1] \land (A_i = B_j \lor i = I_x[j-1]) \} \]

How to locate the relevant match \( A_i = B_j \) quickly?

- Precompute a \( \sigma \times n \) table \( NM \), where

\[ NM[\lambda, k] = \min \{ i : i > k \land (A_i = \lambda \lor i = n + 1) \} \]

▷ E.g. if \( A = \text{“oklahoma”} \), then \( NM[\text{‘a’,1}] = 4 \) and \( NM[\text{‘h’,5}] = 9 \)

Prague Stringology Conference ’09
An input sensitive online algorithm for LCS computation

\[
I_x[j] = \min \{ i : i > I_{x-1}[j-1] \land (A_i = B_j \lor i = I_x[j-1]) \}
\]

How to locate the relevant match \( A_i = B_j \) quickly?

- Precompute a \( \sigma \times n \) table \( NM \), where

\[
NM[\lambda, k] = \min \{ i : i > k \land (A_i = \lambda \lor i = n + 1) \}
\]

▷ E.g. if \( A = "oklahoma" \), then \( NM[\\text{‘a’}, 1] = 4 \) and \( NM[\\text{‘h’}, 5] = 9 \)

Resulting time to compute \( L = LLCS(A, B) : \mathcal{O}(\sigma n + m L) \)
An input sensitive online algorithm for LCS computation

Consider again the computation of the increment points

- The changes occur only to the first increment points among blocks of consecutive increments.
Consider again the computation of the increment points

- The changes occur only to the first increment points among blocks of consecutive increments

A “block” encoding:
Consider again the computation of the increment points

- The changes occur only to the first increment points among blocks of consecutive increments

A “block” encoding: let $S_y[j]$ and $E_y[j]$ be the positions of the first and last points in the $y$th maximal segment of consecutive increment points

Prague Stringology Conference ’09
Consider again the computation of the increment points.

- The changes occur only to the first increment points among blocks of consecutive increments.

A “block” encoding: let $S_y[j]$ and $E_y[j]$ be the positions of the first and last points in the $y$th maximal segment of consecutive increment points.

- $\Delta[k, j] = 1$ for $k = S_y[j]..E_y[j]$
- $\Delta[k, j] \neq 0$ for $k = S_y[j]-1$ and $k = E_y[j] + 1$
An input sensitive online algorithm for LCS computation

Create the list of increment blocks for column $j$
incrementally from column $j - 1$
An input sensitive online algorithm for LCS computation

Create the list of increment blocks for column $j$ incrementally from column $j - 1$
An input sensitive online algorithm for LCS computation
Create the list of increment blocks for column $j$
incrementally from column $j - 1$

```
j-1   j       j-1   j
  x-1  |       x-1  |
x-1   |       x-1  |  x
  x    |       x    |  x
 x+1   |       x+1  |
x+2   |       x+2  |
x+2   |       x+2  |
```

if $i = NM[B_j, E_{y-1}[j-1]] < S_y[j-1]$
An input sensitive online algorithm for LCS computation

Create the list of increment blocks for column $j$ incrementally from column $j - 1$

<table>
<thead>
<tr>
<th></th>
<th>$j-1$</th>
<th>$j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x-1$</td>
<td>$x-1$</td>
<td>$x-1$</td>
</tr>
<tr>
<td>$x$</td>
<td>$x$</td>
<td>$x$</td>
</tr>
<tr>
<td>$x+1$</td>
<td>$x+1$</td>
<td>$x+1$</td>
</tr>
<tr>
<td>$x+2$</td>
<td>$x+2$</td>
<td>$x+2$</td>
</tr>
<tr>
<td>$x+2$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

if $i = NM[B_j, E_{y-1}[j-1]] < S_y[j-1]$, add increment point $i$ and increment block $S_y[j-1] + 1..E_y[j-1]$ to column $j$ (possibly merging point $i$ into a block)
An input sensitive online algorithm for LCS computation

Prague Stringology Conference '09
An input sensitive online algorithm for LCS computation

\[
\begin{array}{cccc}
  j-1 & j & j-1 & j \\
  x-1 & x & x-1 & x-1 \\
  x-1 & x+1 & x-1 & x-1 \\
  x & x+1 & x & x \\
  x+1 & x+2 & x+1 & x+2 \\
  x+2 & x+2 & x+2 & x+2 \\
\end{array}
\]

if \( i = NM[B_j, E_{y-1}[j-1]] \geq S_y[j-1] \)
An input sensitive online algorithm for LCS computation

if $i = N M[B_{j}, E_{y-1}[j-1]] \geq S_{y}[j-1]$, add increment block $S_{y}[j-1]..E_{y}[j-1]$ as such to column $j$
An input sensitive online algorithm for LCS computation

\[
\begin{array}{cccc}
  j-1 & j & j-1 & j \\
  x-1 &  & x-1 &  \\
  x-1 &  & x-1 &  \\
  x & x & x & x \\
  x+1 & x+1 & x+1 & x+1 \\
  x+2 & x+2 & x+2 & x+2 \\
  x+2 &  &  &  \\
\end{array}
\]

if \( i = NM[B_j, E_{y-1}[j-1]] \geq S_y[j-1] \), add increment block \( S_y[j-1]..E_y[j-1] \) as such to column \( j \)

Amount of work

Prague Stringology Conference '09
An input sensitive online algorithm for LCS computation

if \( i = NM[B_j, E_{y-1}[j-1]] \geq S_y[j-1] \), add increment block \( S_y[j-1..E_y[j-1]] \) as such to column \( j \)

Amount of work \( \approx \) the number of increment blocks
An input sensitive online algorithm for LCS computation
Analysis of the encoding
An input sensitive online algorithm for LCS computation

Analysis of the encoding

- Consider a column of size $l$ in the table $\Delta$
Consider a column of size \( l \) in the table \( \Delta \)

Let \( 0\# \) be the number of non-increment points
An input sensitive online algorithm for LCS computation

Analysis of the encoding

- Consider a column of size $l$ in the table $\Delta$
- Let $0#$ be the number of non-increment points and $1#$ the number of increment points
An input sensitive online algorithm for LCS computation

Analysis of the encoding

- Consider a column of size $l$ in the table $\Delta$
- Let $0#$ be the number of non-increment points and $1#$ the number of increment points

\[ l = 0# + 1# \]
An input sensitive online algorithm for LCS computation

Analysis of the encoding

- Consider a column of size $l$ in the table $\Delta$
- Let $0\#$ be the number of non-increment points and $1\#$ the number of increment points
  \[ l = 0\# + 1\# \]
- Also let $block\#$ be the number of maximal increment blocks
An input sensitive online algorithm for LCS computation

Analysis of the encoding

- Consider a column of size $l$ in the table $\Delta$
- Let $0#$ be the number of non-increment points and $1#$ the number of increment points
  \[ l = 0# + 1# \]
- Also let $\text{block}#$ be the number of maximal increment blocks
- Now it holds that
  \[ \text{block}# \leq 1# \]
Consider a column of size $l$ in the table $\Delta$

Let $0#$ be the number of non-increment points and $1#$ the number of increment points

\[ l = 0# + 1# \]

Also let $\text{block}#$ be the number of maximal increment blocks

Now it holds that

\[ \text{block}# \leq 1# \]

\[ \text{block}# \leq 1 + 0# = l - 1# + 1 \]
An input sensitive online algorithm for LCS computation

Consider the figure

\[ m-L = z \]
An input sensitive online algorithm for LCS computation

Consider the figure

- Each of the first \( z = m - L \) columns
An input sensitive online algorithm for LCS computation

Consider the figure

- Each of the first $z = m - L$ columns has at most $z$ increment points (because $D[n, j] \leq j$)
An input sensitive online algorithm for LCS computation

Consider the figure

- Each of the first $z = m - L$ columns has at most $z$ increment points (because $D[n, j] \leq j$)

▷ Work for this part: $O(z^2)$
An input sensitive online algorithm for LCS computation

\[ m - L = z \]

- In each column \( z + i \):
An input sensitive online algorithm for LCS computation

$m-L = z$

- In each column $z+i$: there are at most $z+i$ increment points
An input sensitive online algorithm for LCS computation

$m - L = z$

- In each column $z + i$: there are at most $z + i$ increment points and the first $n - L + i$ rows hold at least $i$ increment points.
An input sensitive online algorithm for LCS computation

$m-L = z$

- In each column $z+i$: there are at most $z+i$ increment points and the first $n-L+i$ rows hold at least $i$ increment points

$\triangleright \textit{block} \# \text{ for first } n-L+i \text{ rows } \leq n-L+1$
An input sensitive online algorithm for LCS computation

\[
m-L = z
\]

- In each column \( z + i \): there are at most \( z + i \) increment points and the first \( n-L+i \) rows hold at least \( i \) increment points

- \( \text{block}_\# \) for first \( n-L+i \) rows \( \leq n-L+1 \)

- \( \text{block}_\# \) for remaining rows \( \leq z+i-i = z \)
An input sensitive online algorithm for LCS computation

\[ m - L = z \]

\[
\text{Total work} \approx z^2 + (m - z)(n - L + z + 1) = \mathcal{O}(m(n - L))
\]
An input sensitive online algorithm for LCS computation

\[ m - L = z \]

\[
\text{Total work} \approx z^2 + (m - z)(n - L + z + 1) = \mathcal{O}(m(n - L)),
\]

and also bounded by \( \mathcal{O}(mL) \)
An input sensitive online algorithm for LCS computation

$m-L = z$

Total work $\approx z^2 + (m - z)(n - L + z + 1) = O(m(n - L))$, and also bounded by $O(mL)$

Total time complexity $O(\sigma n + \min\{mL, L(n - L)\})$
An input sensitive online algorithm for LCS computation

$m-L = z$

Total work $\approx z^2 + (m - z)(n - L + z + 1) = O(m(n - L))$, and also bounded by $O(mL)$

Total time complexity $O(\sigma n + \min\{mL, L(n - L)\})$

- If $L < \frac{m}{2}$, then $O(mL) = O(L(n - L))$
An input sensitive online algorithm for LCS computation

Total work \( \approx z^2 + (m - z)(n - L + z + 1) = \mathcal{O}(m(n - L)) \), and also bounded by \( \mathcal{O}(mL) \)

Total time complexity \( \mathcal{O}(\sigma n + \min\{mL, L(n - L)\}) \)

- If \( L < \frac{m}{2} \), then \( \mathcal{O}(mL) = \mathcal{O}(L(n - L)) \)
- If \( L \geq \frac{m}{2} \), then \( \mathcal{O}(m(n - L)) = \mathcal{O}(L(n - L)) \)