Algorithms for Computing the Longest Parameterized Common Subsequence

Costas S. Iliopoulos[1], **Marcin Kubica**[2],
M. Sohel Rahman[1] and Tomasz Waleń[2]

[1]Algorithm Design Group
Department of Computer Science, Kings College London

[2]Faculty of Mathematics, Informatics and Applied Mathematics
Warsaw University, Poland

CPM, 2007-07-11

The *LPCS* Problem

The *LPCS* (*longest parameterized common subsequence*) problem is a generalization of a well known *LCS* problem, containing *gap-constraints*.

Definition

In $LPCS(X, Y, K_1, K_2, D)$ we look for such longest increasing sequences of indices $P[1, .., l]$ and $Q[1, .., l]$, that:

- $X[P[i]] = Y[P[i]]$
  Common subsequence.
- $K_1 \leq P[i+1] - P[i], Q[i+1] - Q[i] \leq K_2$
  Gaps between consecutive matches are not shorter than K_1 and not longer than K_2.
- $|(P[i+1] - P[i]) - (Q[i+1] - Q[i])| \leq D$
  The corresponding gaps in both sequences cannot differ more than D.

The *LPCS* Problem

The *LPCS* (*longest parameterized common subsequence*) problem is a generalization of a well known *LCS* problem, containing *gap-constraints*.

Definition

In $LPCS(X, Y, K_1, K_2, D)$ we look for such longest increasing sequences of indices $P[1, .., l]$ and $Q[1, .., l]$, that:

- $X[P[i]] = Y[P[i]]$
  Common subsequence.
- $K_1 \leq P[i+1] - P[i], Q[i+1] - Q[i] \leq K_2$
  Gaps between consecutive matches are not shorter than K_1 and not longer than K_2.
- $|(P[i+1] - P[i]) - (Q[i+1] - Q[i])| \leq D$
  The corresponding gaps in both sequences cannot differ more than D.

The *LPCS* Problem

The *LPCS* (*longest parameterized common subsequence*) problem is a generalization of a well known *LCS* problem, containing *gap-constraints*.

Definition

In $LPCS(X, Y, K_1, K_2, D)$ we look for such longest increasing sequences of indices $P[1, .., l]$ and $Q[1, .., l]$, that:

- $X[P[i]] = Y[P[i]]$
  Common subsequence.
- $K_1 \leq P[i+1] - P[i], Q[i+1] - Q[i] \leq K_2$
  Gaps between consecutive matches are not shorter than K_1 and not longer than K_2.
- $|(P[i+1] - P[i]) - (Q[i+1] - Q[i])| \leq D$
  The corresponding gaps in both sequences cannot differ more than D.

The *LPCS* Problem

The *LPCS* (*longest parameterized common subsequence*) problem is a generalization of a well known *LCS* problem, containing *gap-constraints*.

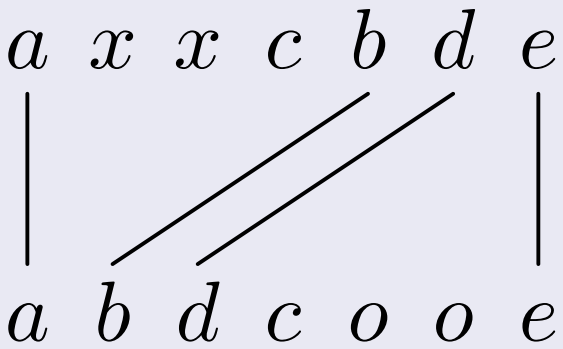
Definition

In $LPCS(X, Y, K_1, K_2, D)$ we look for such longest increasing sequences of indices $P[1, .., l]$ and $Q[1, .., l]$, that:

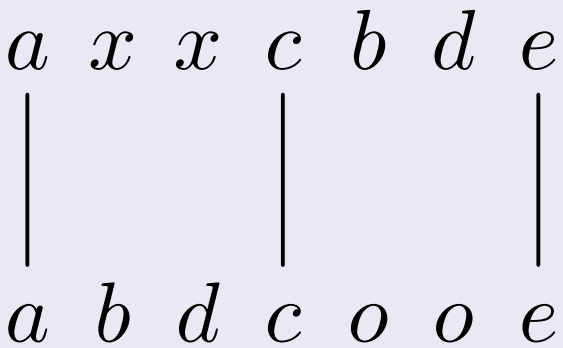
- $X[P[i]] = Y[P[i]]$
  Common subsequence.
- $K_1 \leq P[i+1] - P[i], Q[i+1] - Q[i] \leq K_2$
  Gaps between consecutive matches are not shorter than K_1 and not longer than K_2.
- $|(P[i+1] - P[i]) - (Q[i+1] - Q[i])| \leq D$
  The corresponding gaps in both sequences cannot differ more than D.

The LCS and *LPCS* Problems

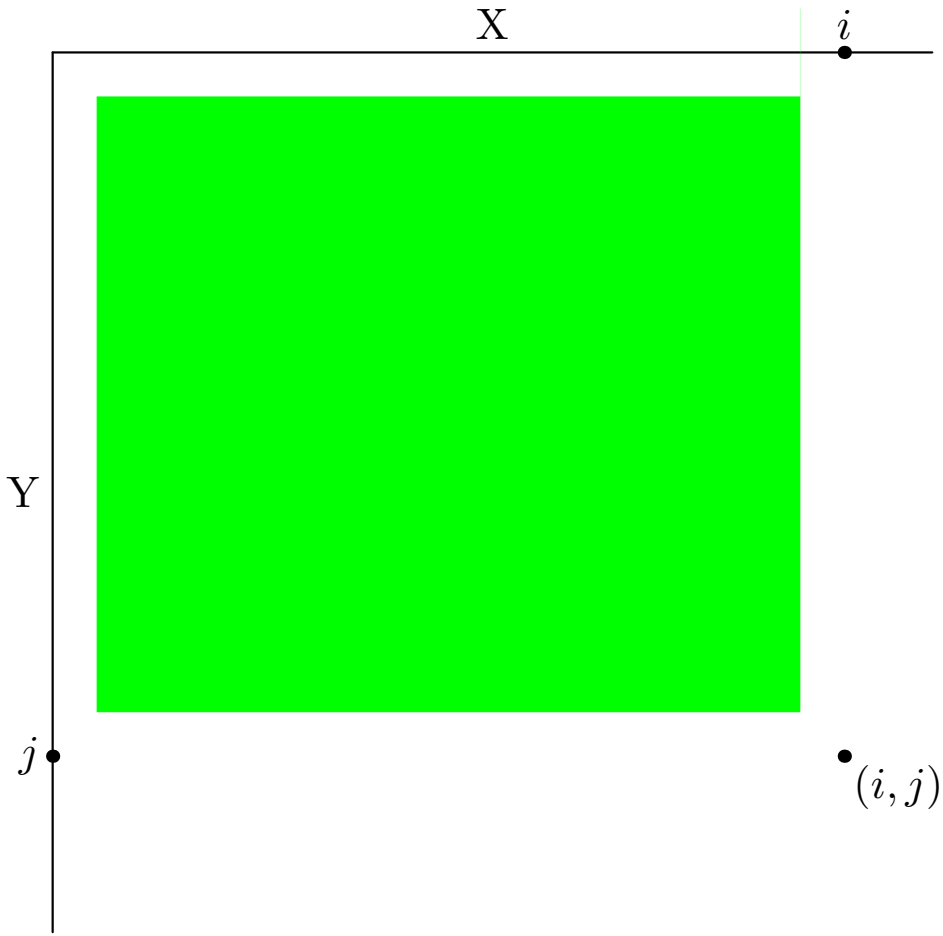
LCS



LPCS, $K_1 = 1$, $K_2 = 3$, $D = 1$

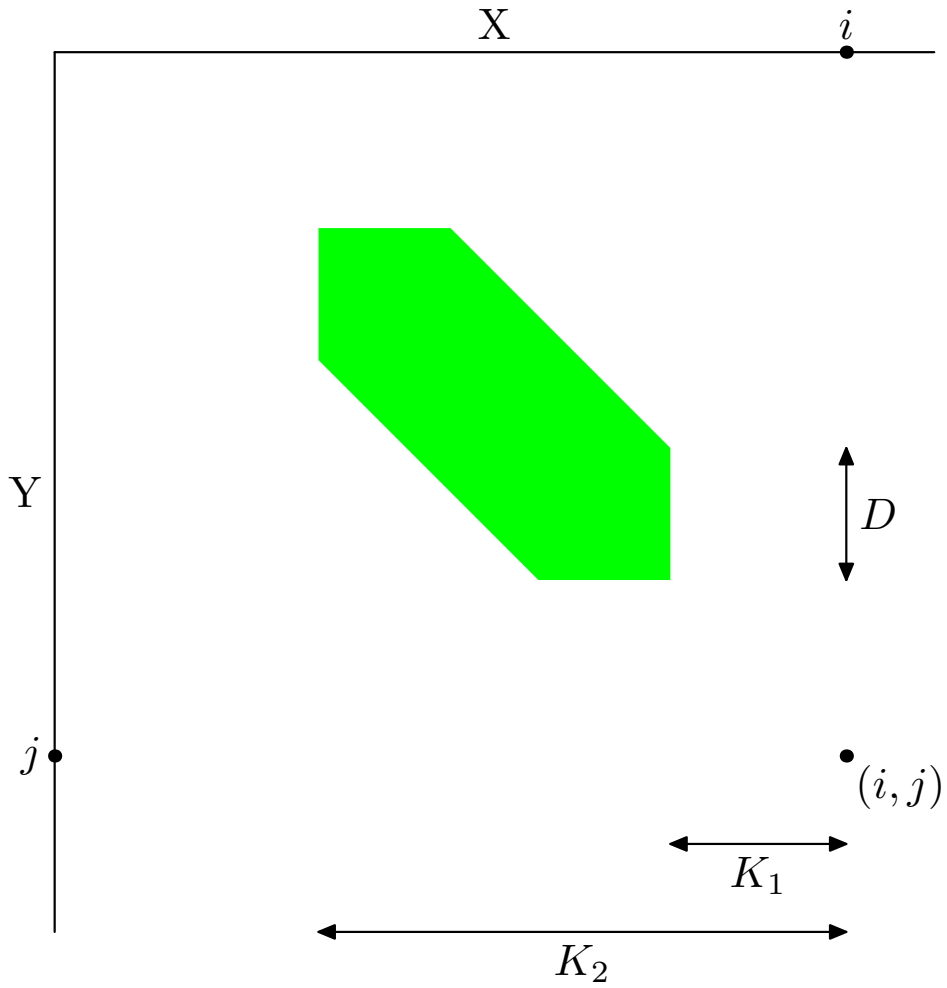


The LCS and *LPCS* Problems



$$LCS(i,j) = 1 + \max\{LCS(x,y) : 1 \leq x < i, 1 \leq y < j\}$$

The LCS and *LPCS* Problems



$$PLCS(i,j) = 1 + \max\left\{ PLCS(x,y) \quad : \quad \begin{array}{l} K_1 \leq i-x, j-y \leq K_2, \\ |(i-x)-(j-y)| \leq D \end{array} \right\}$$

The *FIG*, *ELAG*, *RIFIG* and *RELAG* problems

The *LPCS* problem is a generalization of four problems introduced by C. S. Iliopoulos and M. S. Rahman (ISAAC 2006):

Definition

- $FIG(X, Y, K) = LPCS(X, Y, 1, K, n)$
  LCS problem with fixed gaps.
- $ELAG(X, Y, K_1, K_2) = LPCS(X, Y, K_1, K_2, n)$
  LCS problem with elastic gaps.
- $RIFIG(X, Y, K) = LPCS(X, Y, 1, K, 0)$
  LCS problem with rigid fixed gaps.
- $RELAG(X, Y, K_1, K_2) = LPCS(X, Y, K_1, K_2, 0)$
  LCS problem with rigid elastic gaps.

The *FIG*, *ELAG*, *RIFIG* and *RELAG* problems

The *LPCS* problem is a generalization of four problems introduced by C. S. Iliopoulos and M. S. Rahman (ISAAC 2006):

Definition

- $FIG(X, Y, K) = LPCS(X, Y, 1, K, n)$
  LCS problem with fixed gaps.
- $ELAG(X, Y, K_1, K_2) = LPCS(X, Y, K_1, K_2, n)$
  LCS problem with elastic gaps.
- $RIFIG(X, Y, K) = LPCS(X, Y, 1, K, 0)$
  LCS problem with rigid fixed gaps.
- $RELAG(X, Y, K_1, K_2) = LPCS(X, Y, K_1, K_2, 0)$
  LCS problem with rigid elastic gaps.

The *FIG*, *ELAG*, *RIFIG* and *RELAG* problems

The *LPCS* problem is a generalization of four problems introduced by C. S. Iliopoulos and M. S. Rahman (ISAAC 2006):

Definition

- $FIG(X, Y, K) = LPCS(X, Y, 1, K, n)$
  LCS problem with fixed gaps.
- $ELAG(X, Y, K_1, K_2) = LPCS(X, Y, K_1, K_2, n)$
  LCS problem with elastic gaps.
- $RIFIG(X, Y, K) = LPCS(X, Y, 1, K, 0)$
  LCS problem with rigid fixed gaps.
- $RELAG(X, Y, K_1, K_2) = LPCS(X, Y, K_1, K_2, 0)$
  LCS problem with rigid elastic gaps.

The *FIG*, *ELAG*, *RIFIG* and *RELAG* problems

The *LPCS* problem is a generalization of four problems introduced by C. S. Iliopoulos and M. S. Rahman (ISAAC 2006):

Definition

- $FIG(X, Y, K) = LPCS(X, Y, 1, K, n)$
  LCS problem with fixed gaps.
- $ELAG(X, Y, K_1, K_2) = LPCS(X, Y, K_1, K_2, n)$
  LCS problem with elastic gaps.
- $RIFIG(X, Y, K) = LPCS(X, Y, 1, K, 0)$
  LCS problem with rigid fixed gaps.
- $RELAG(X, Y, K_1, K_2) = LPCS(X, Y, K_1, K_2, 0)$
  LCS problem with rigid elastic gaps.

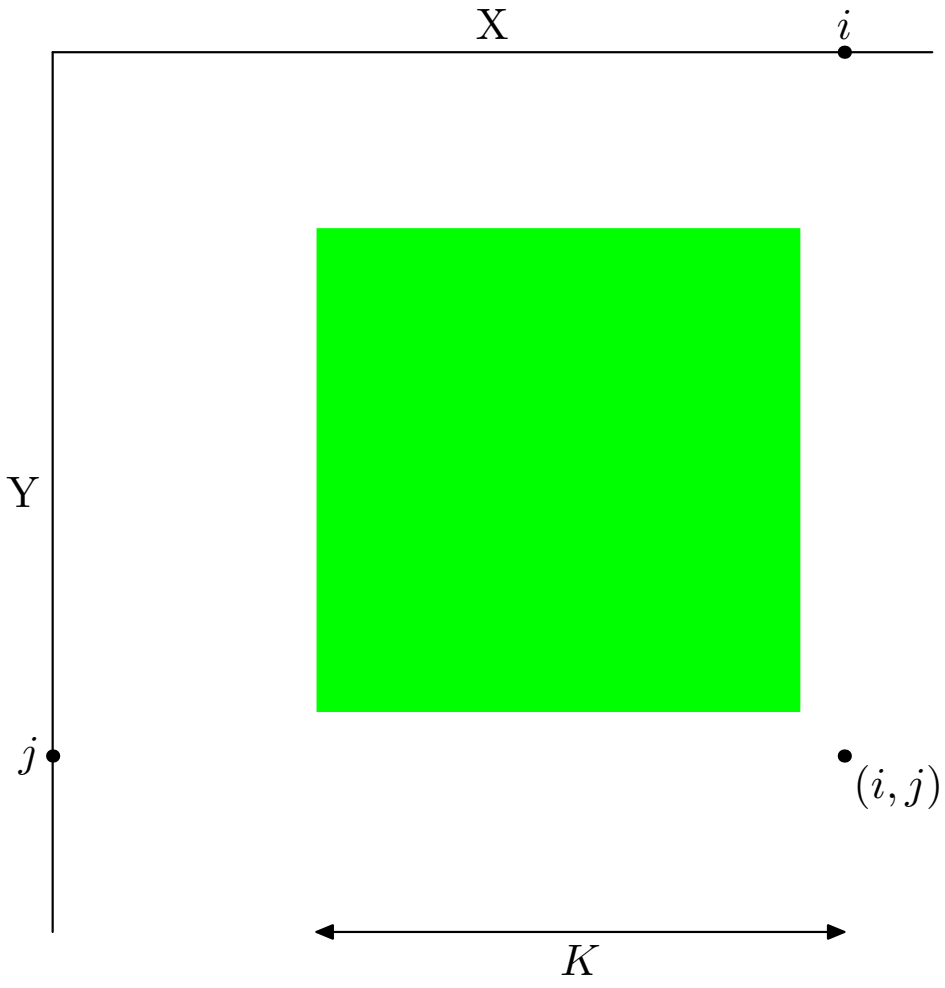
The *FIG*, *ELAG*, *RIFIG* and *RELAG* problems

The *LPCS* problem is a generalization of four problems introduced by C. S. Iliopoulos and M. S. Rahman (ISAAC 2006):

Definition

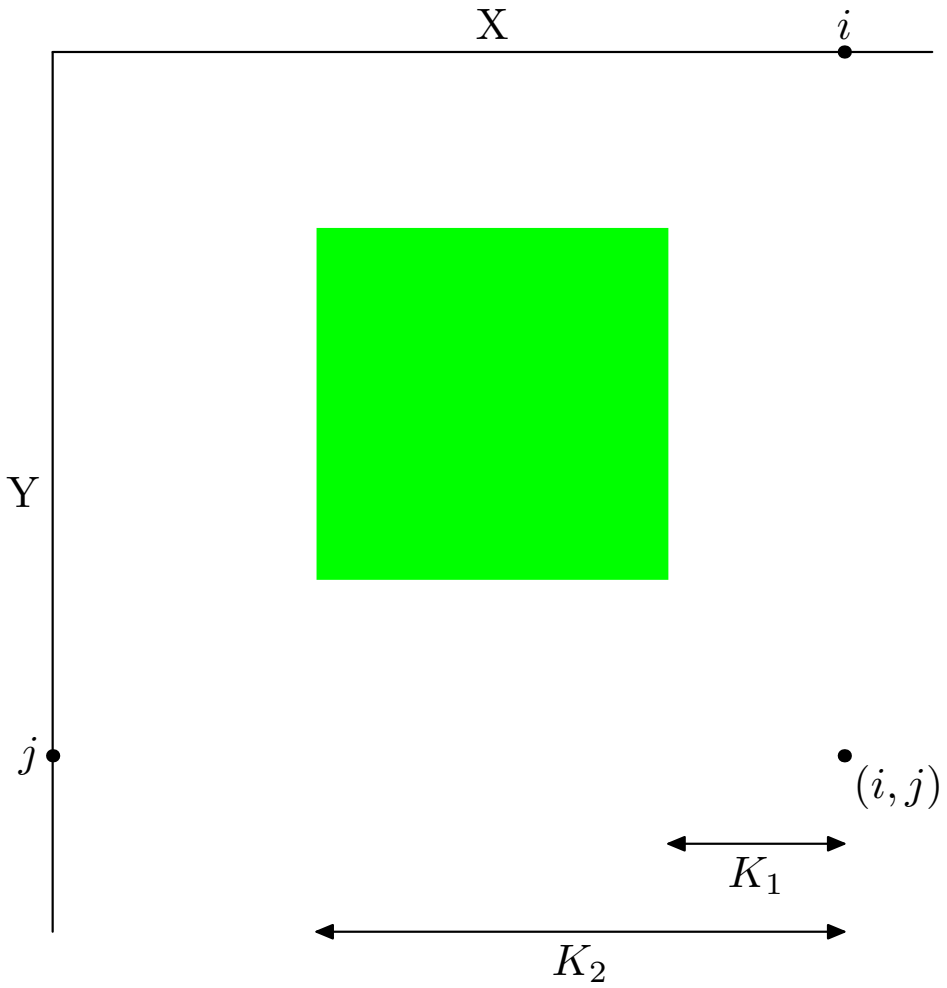
- $FIG(X, Y, K) = LPCS(X, Y, 1, K, n)$
  LCS problem with fixed gaps.
- $ELAG(X, Y, K_1, K_2) = LPCS(X, Y, K_1, K_2, n)$
  LCS problem with elastic gaps.
- $RIFIG(X, Y, K) = LPCS(X, Y, 1, K, 0)$
  LCS problem with rigid fixed gaps.
- $RELAG(X, Y, K_1, K_2) = LPCS(X, Y, K_1, K_2, 0)$
  LCS problem with rigid elastic gaps.

The *LPCS* Problem



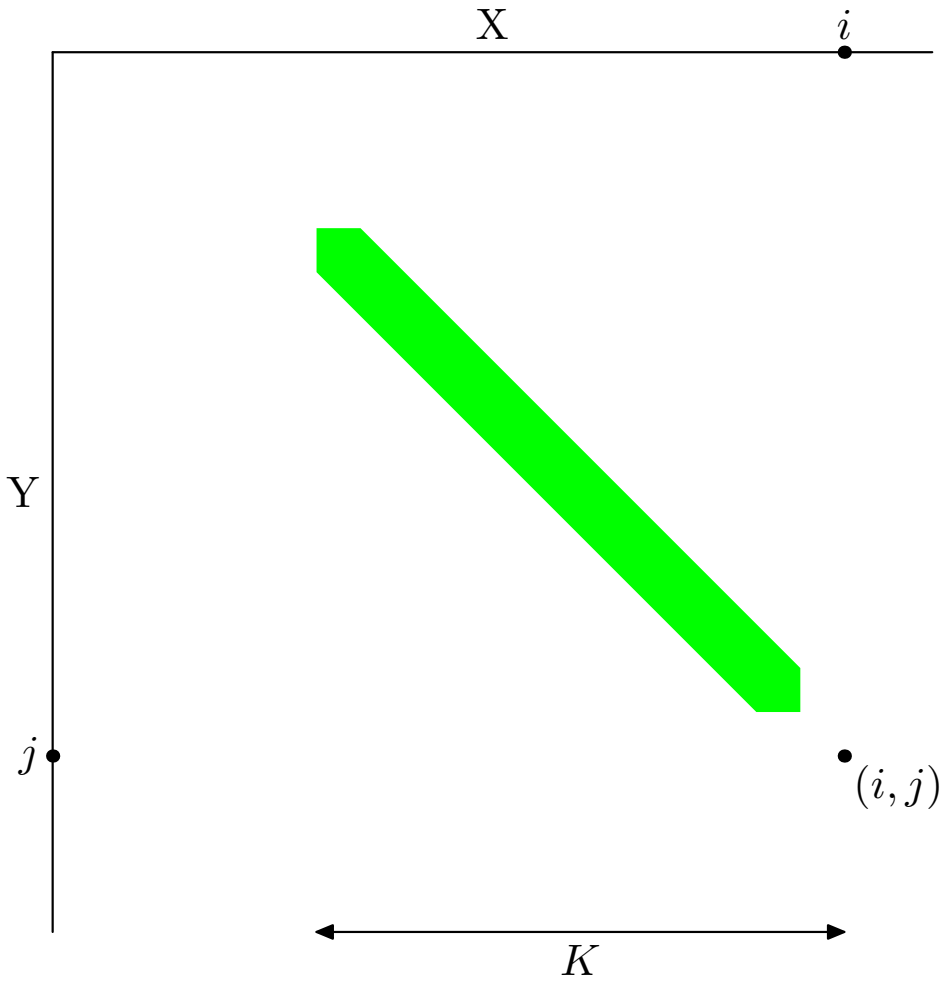
$$FIG(i,j) = 1 + \max\{FIG(x,y) : i - x, j - y \leq K\}$$

The *LPCS* Problem



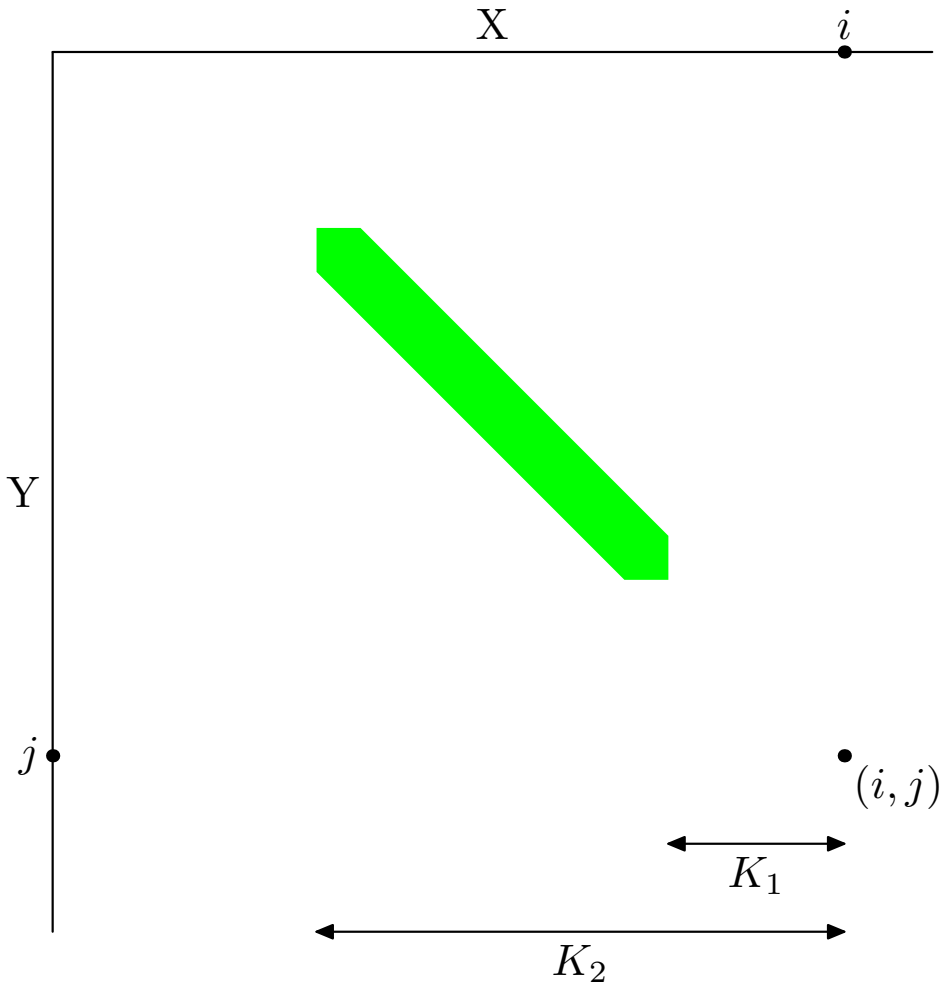
$$ELAG(i,j) = 1 + \max\{ELAG(x,y) : K_1 \leq i - x, j - y \leq K_2\}$$

The *LPCS* Problem



$$RIFIG(i,j) = 1 + \max\{RIFIG(x,y) : i - x = j - y \leq K\}$$

The *LPCS* Problem



$$RELAG(i,j) = 1 + \max\{RELAG(x,y) : K_1 \leq i - x = j - y \leq K_2\}$$

Previous Results

Summary of previously known results

PROBLEM	Previous Results	Our Results
LPCS	–	$O(\min(n^2, n + \mathcal{R} \log n))$
FIG	$O(n^2 + \mathcal{R} \log \log n)$	
ELAG	$O(n^2 + \mathcal{R} \log \log n)$	
RIFIG	$O(n^2)$	$O(n + \mathcal{R})$
RELAG	$O(n^2 + \mathcal{R}(K_2 - K_1))$	

Where $\mathcal{R}$ is the total number of matches.

Max-queue data structure

The max-queue data structure can be used to calculate maximum of last L elements inserted into the queue.

Operations

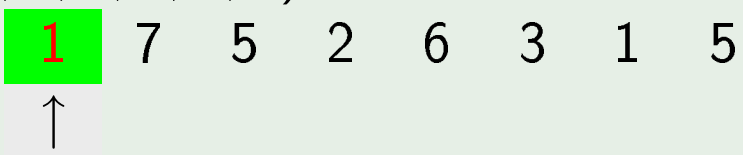
- `init`(Q, L) — initialize and set the history length
- `insert`(Q, x)
- `max`(Q) — returns maximum of the last L inserted elements.

All operations run in $O(1)$ (amortized) time.

Max-queue data structure example

Example

For the sequence $(1, 7, 5, 2, 6, 3, 1)$ and $L = 4$

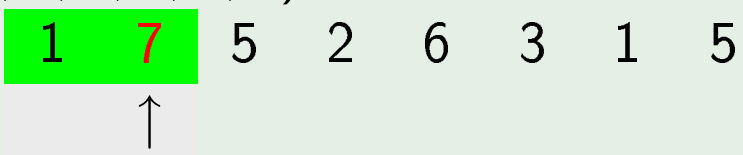


$$MaxQueue = (1)$$

Max-queue data structure example

Example

For the sequence $(1, 7, 5, 2, 6, 3, 1)$ and $L = 4$



$$MaxQueue = (7)$$

Max-queue data structure example

Example

For the sequence $(1, 7, 5, 2, 6, 3, 1)$ and $L = 4$

$$1 \quad 7 \quad 5 \quad 2 \quad 6 \quad 3 \quad 1 \quad 5$$

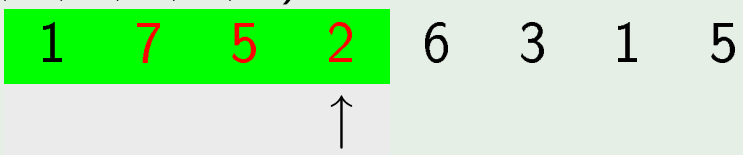
$$\uparrow$$

$$MaxQueue = (7, 5)$$

Max-queue data structure example

Example

For the sequence $(1, 7, 5, 2, 6, 3, 1)$ and $L = 4$

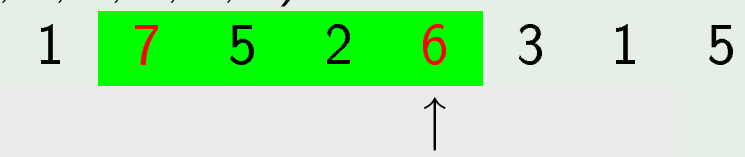


$$MaxQueue = (7, 5, 2)$$

Max-queue data structure example

Example

For the sequence $(1, 7, 5, 2, 6, 3, 1)$ and $L = 4$



$$MaxQueue = (7, 6)$$

Max-queue data structure example

Example

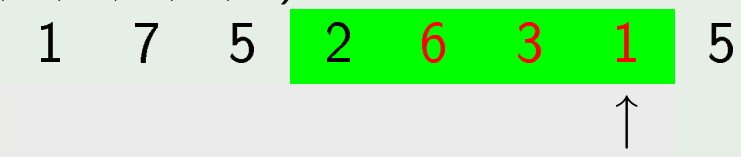
For the sequence $(1, 7, 5, 2, 6, 3, 1)$ and $L = 4$

$$MaxQueue = (6, 3)$$

Max-queue data structure example

Example

For the sequence $(1, 7, 5, 2, 6, 3, 1)$ and $L = 4$

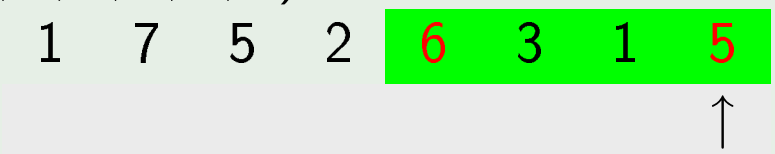


$$MaxQueue = (6, 3, 1)$$

Max-queue data structure example

Example

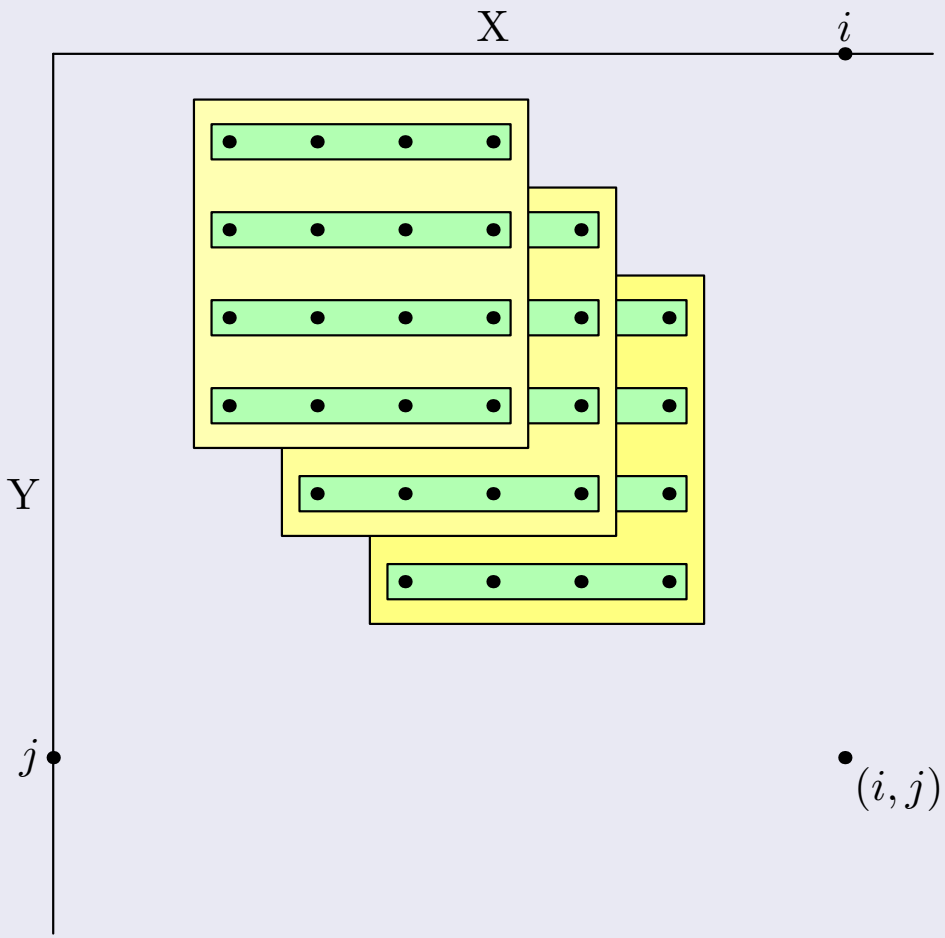
For the sequence $(1, 7, 5, 2, 6, 3, 1)$ and $L = 4$



$$MaxQueue = (6, 5)$$

The Algorithm for *LPCS*

Algorithm



- Dynamic programming.
- Three-level Max-queue.
- Time complexity $O(n^2)$.

The Algorithm for *FIG* and *ELAG*

Algorithm

For $\mathcal{R} = o(n^2/\log n)$:

- Dynamic programming.
- Using dictionary data-structure providing: insertion, removal and max-range queries.
- $O(\mathcal{R})$ steps (for matches only), each in $O(\log n)$ time.
- Time complexity: $O(n + \mathcal{R} \log n)$.

Can be extended to solve *LPCS* in $O(n + \mathcal{R} \cdot \log n)$ time.

The Algorithm for *RIFIG* and *RELAG*.

Algorithm

- Dynamic programming.
- Each diagonal is processed separately.
- $O(\mathcal{R})$ steps (for matches only).
- Each step in $O(1)$ amortised time (using max-queue).
- Time complexity: $O(n + \mathcal{R})$.

Conclusions

Conclusions

- New problem *LPCS* which generalizes *FIG*, *ELAG*, *RIFIG*, *RELAG*.
- Simplified and faster, the $O(n^2)$ algorithm for *LPCS* problem.
- The $O(n + \mathcal{R} \log n)$ algorithm for *ELAG* and *LPCS* problem.
- The $O(n + \mathcal{R})$ algorithm for *RIFIG*, *RELAG*.

The End

Thank you for your attention!