2-Dimensional Range Minimum Queries

Amihood Amir\textsuperscript{1}, Johannes Fischer\textsuperscript{2}, and Moshe Lewenstein\textsuperscript{1}

\textsuperscript{1}Computer Science Department
Bar Ilan University

\textsuperscript{2}Institut für Informatik
Ludwig-Maximilians-Universität München

July 2007, London (Ontario)
Outline

1 Introduction to RMQs
   - Formal Problem Definition
   - Previous Results

2 Solution Methods
   - Overview
   - Preprocessing of First Level
   - Other Levels and Microblock-Queries
Outline

1. Introduction to RMQs
   - Formal Problem Definition
   - Previous Results

2. Solution Methods
   - Overview
   - Preprocessing of First Level
   - Other Levels and Microblock-Queries
The Problem

given: matrix $A[1..m][1..n]$ (totally ordered objects)
task: preprocess $A$ to answer “efficiently”

$$\text{RMQ}(y_1, y_2, x_1, x_2) = \arg\min_{(y,x)\in[y_1:y_2] \times [x_1:x_2]} A[y][x]$$
Main Result

The Problem

given: matrix $A[1..m][1..n]$ (totally ordered objects)
task: preprocess $A$ to answer “efficiently”

$$\text{RMQ}(y_1, y_2, x_1, x_2) = \arg\min_{(y,x)\in[y_1:y_2] \times [x_1:x_2]} A[y][x]$$

Theorem (2-Dimensional RMQs)

$O(nm(k + \log^{[k+1]}(mn)))$-preprocessing using $O(kmn)$
space for $O(1)$-RMQs, for any $k > 1$.
Converges towards $O(mn \log^*(mn))$ preprocessing time
and space, and $O(1)$ query time.
Our Result in Context

\[ N := mn \]

<table>
<thead>
<tr>
<th></th>
<th>preprocessing</th>
<th>space</th>
<th>query time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gabow et al.’84</td>
<td>( O(N \log N) )</td>
<td>( O(N \log N) )</td>
<td>( O(\log N) )</td>
</tr>
<tr>
<td>Chazelle-Rosenberg’89</td>
<td>( O(\alpha^2(RN, N)) )</td>
<td>( O(\alpha^2(RN, N)) )</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>(with ( R = \text{const} ))</td>
<td>( O(N) )</td>
<td>( O(N) )</td>
<td></td>
</tr>
<tr>
<td>Mäkinen’03</td>
<td>( O(N \log m) )</td>
<td>( O(N \log m) )</td>
<td>( O(\alpha^2(RN, N)) )</td>
</tr>
<tr>
<td>this paper</td>
<td>( O(N(k + \log^{[k+1]} N)) )</td>
<td>( O(kN) )</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>(with ( k = 2 ))</td>
<td>( O(N \log \log \log N) )</td>
<td>( O(N) )</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>this paper</td>
<td>( O(N \log^* N) )</td>
<td>( O(N \log^* N) )</td>
<td>( O(1) )</td>
</tr>
</tbody>
</table>

\( R \geq 144, k \geq 2 \) (both not necessarily constants!)
Results for 1-Dimensional RMQs

- Results for $O(n)$ preprocessing time and $O(1)$ query time:

<table>
<thead>
<tr>
<th></th>
<th>space (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>final</td>
</tr>
<tr>
<td>Berkman-Vishkin’93</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>Bender-Farach-Colton’00</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>Alstrup et al.’02</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>Fischer-Heun’06</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>Sadakane’02</td>
<td>$4n + o(n)$</td>
</tr>
<tr>
<td>Fischer-Heun’07</td>
<td>$2n + o(n)$</td>
</tr>
</tbody>
</table>

- Cartesian Trees (=treaps) important tool for all!
Outline

1. Introduction to RMQs
   - Formal Problem Definition
   - Previous Results

2. Solution Methods
   - Overview
   - Preprocessing of First Level
   - Other Levels and Microblock-Queries
Overview of the Algorithm

- impose grids on \( A \) (widths \( s_1 \geq s_2 \geq \cdots \geq s_k \))
- preprocess each “layer” separately
- level \( i \): queries crossing grid \( s_i \), but no grid \( s_j \) for \( j < i \)
- other queries: precompute all possible!

Time:

\[
O\left(\frac{kN}{k \text{ levels}}\right) + N \log^{[k+1]} N \text{ sorting on level } k
\]
Assume $A$ is square ($\Rightarrow n = m, N = n^2$)

$s_1 = \log n$: grid-width on level 1

Decompose $\text{RMQ}(y_1, y_2, x_1, x_2)$ into nine sub-queries:

1. Several blocks in both directions
2-5. Several blocks in 1 direction
6-9. In-block, but "touches" boundary
Precomputation of Queries 1–9

- Precompute only queries that span $2^k \times 2^l$ blocks

- Answer all queries by selecting at most four overlapping “power-of-two”-queries

- same idea as for 1D-RMQ!
Recursive Partitioning

- perform \textit{same} preprocessing for grid-widths
  \[ s_2 = \log \log n, \ s_3 = \log \log \log n, \ldots, \ s_k = \log^{[k]} n \]

- either stopping at some fixed \( k > 1 \ldots \)
- \( \ldots \) or until \( s_k = O(1) \) (\( \iff \) \( k = \Theta(\log^* n) \))
Recursive Partitioning

- perform **same** preprocessing for grid-widths
  
  \[ s_2 = \log \log n, \ s_3 = \log \log \log n, \ldots, \ s_k = \log^{[k]} n \]

- either stopping at some fixed \( k > 1 \ldots \)
- \( \ldots \) or until \( s_k = O(1) \) (\( \iff k = \Theta(\log^* n) \))

- naive query-answering would cost \( O(k) \) or \( O(\log^* n) \) time!

\[ \Rightarrow \text{store additional structures of size } o(n) \text{ for selecting the right grid in } O(1) \text{ time} \]

- still left with **microblock-queries**
What to do with microblock-queries?

### Key-property for 1D-RMQs

2 blocks have \textit{same answers} to all RMQs $\iff$ they have equal Cartesian Trees

$$A = \begin{array}{cccccccccccccccccccccc} 3 & 0 & 2 & 1 & 1 & 1 & 2 & 0 & 2 & 1 & 1 & 2 & 3 & 2 & 1 & 1 & 0 & 2 & 0 & 2 & 3 \end{array}$$

Problem: nothing similar for 2D

### Weaker Property

2 blocks have same answers to all RMQs $\iff$ elements have \textit{same relative order}
What to do with microblock-queries (2)?

**Weaker Property**

2 blocks have same answers to all RMQs ⇐ elements have same relative order

⇒ sort microblocks (e.g. row-wise) to get permutation of $[1 : S]$ (=relative order), $S = s_k^2$

- precompute microblock-queries only for different permutations and **not** for all occurring microblocks
What to do with microblock-queries (2)?

**Weaker Property**

2 blocks have same answers to all RMQs $\iff$ elements have same relative order

$\Rightarrow$ sort microblocks (e.g. row-wise) to get permutation of $[1 : S]$ (=relative order), $S = s_k^2$

- precompute microblock-queries only for different permutations and not for all occurring microblocks
- space is $O(S^2 \times S!) = \cdots = O(N)$ (if $k > 1$)

- time is $O\left(\frac{N}{S} \times S \log S\right) = O(N \log^{[k+1]} n)$
Summary

- preprocessing-scheme for 2D-RMQs ($N$: size of input)
  - space $O(N)$
  - preprocessing time $\tilde{O}(N)$
  - query time $O(1)$

- generalizes to higher dimensions $d$: query time $O(C^d)$, $C = \text{const}$

- open question: can we achieve $O(N)$ preprocessing time and $O(1)$ query time?
  - impossible for slightly more general operations! (Chazelle-Rosenberg’89)