Time-Space Trade-Offs for 2D Range Minimum Queries

Gerth Stølting Brodal
Aarhus University

Join work with Pooya Davoodi and S. Srinivasa Rao

Dagstuhl Seminar on Data Structures, February 28 - March 5, 2010
The 2D Range Minimum Problem

Preprocess an \( m \times n \)-matrix of size \( N = n \cdot m \), \( m \leq n \), to efficiently support range minimum queries

\[
\text{RMQ}([i_1, i_2] \times [j_1, j_2]) = (i', j')
\]

\[
A_{i', j'} = \min\{ A_{i'', j''} \mid (i'', j'') \in [i_1, i_2] \times [j_1, j_2] \}, \quad (i', j') \in [i_1, i_2] \times [j_1, j_2]
\]
Models

Encoding model
- Queries can access data structure but not input matrix

Indexing model
- Queries can access data structure and read input matrix
## Some Trivial Examples...

<table>
<thead>
<tr>
<th>Solution</th>
<th>Additional space (bits)</th>
<th>Query time</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>No data structure</td>
<td>0</td>
<td>$O(N)$</td>
<td>Indexing</td>
</tr>
<tr>
<td>Tabulate answers</td>
<td>$O(N^2 \log N)$</td>
<td>$O(1)$</td>
<td>Encoding</td>
</tr>
<tr>
<td>Store permutation</td>
<td>$O(N \log N)$</td>
<td>$O(N)$</td>
<td>Encoding</td>
</tr>
</tbody>
</table>

#### Table:

<table>
<thead>
<tr>
<th>$i_1$</th>
<th>$j_1$</th>
<th>10</th>
<th>4</th>
<th>13</th>
<th>9</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>14</td>
<td>6</td>
<td>11</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>28</td>
<td>9</td>
<td>16</td>
<td>52</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>48</td>
<td>19</td>
<td>2</td>
<td>23</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Minimum $j$'s index is $j_2$.
Results
1D Range Minimum Queries

Indexing

- Upper Bound
  - Fischer and Heun (2007)
  - \( \text{Time} = O(1) \)
  - \( \text{Space} = 2n + o(n) + |A| \) bits

- NEW
  - Lower Bound
    - (matching upper bound)
    - \( \text{Time} = \Omega(c) \)
    - \( \text{Space} = O(n/c) + |A| \) bits

Encoding

- Upper Bound
  - Fischer (Latin 2010)
  - \( \text{Time} = O(1) \)
  - \( \text{Space} = 2n + o(n) \) bits

- Lower Bound:
  - \( \text{Space} = 2n - \Theta(\log n) \) bits
2D Range Minimum Queries

Indexing

\[ \text{Upper Bound} \]
\[ \begin{align*}
\text{Time} &= O(1) \\
\text{Space} &= O(N) + |A| \text{ bits}
\end{align*} \]

\[ \begin{align*}
\text{Time} &= O(c \log^2 c) \\
\text{Space} &= O(N/c) + |A| \text{ bits}
\end{align*} \]

\[ \text{NEW} \]

\[ \text{Lower Bound} \]
\[ \begin{align*}
\text{Time} &= \Omega(c) \\
\text{Space} &= O(N/c) + |A| \text{ bits}
\end{align*} \]

Encoding

\[ \text{Upper Bound} \]
\[ \begin{align*}
\text{Time} &= O(1) \\
\text{Space} &= O(N \log n) \text{ bits}
\end{align*} \]

\[ \text{NEW Proof} \]

\[ \text{Lower Bound:} \]
\[ \begin{align*}
\text{Space} &= \Omega(N \log m) \text{ bits}
\end{align*} \]

Demain et al. (2009)
1D Encoding model
Index model
Upper bound
Lower bound

<table>
<thead>
<tr>
<th>1D</th>
<th>Encoding model</th>
<th>Index model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Lower Bound (1D, Encoding)

- For each input array consider the Cartesian tree
- Each binary tree is a possible Cartesian tree
- RMQ queries can reconstruct the Cartesian tree
- \# Cartesian trees is $\binom{2n}{n} / (n+1)$
- \# bits $\geq \log\left(\binom{2n}{n} / (n+1)\right) = 2n - \Theta(\log n)$
For an input array consider the Cartesian tree.

- Succinct representation using $4n+o(n)$ bits and $O(1)$ query time (Sadakane 2007)
- Improved to $2n+o(n)$ (Fischer 2010)
Upper Bounds (1D, Indexing)

- Build encoding \(O(N/c)\) bit structure for block minimums
- RMQ = query to encoding structure + 3\(c\) elements, i.e. query time \(O(c)\)
Lower Bounds (1D, Indexing)

Thm  Space $\frac{N}{c}$ bits implies $\Omega(c)$ query time

- Consider $\frac{N}{C}$ queries for $c^{\frac{N}{c}}$ different \{0,1\} inputs with exactly one zero in each block
- $c^{\frac{N}{c}} / 2^{\frac{N}{c}}$ inputs share some data structure
- Every query is a decision tree of height $\leq d$
Lower Bounds (1D, Indexing) cont.

- Combine queries to decision tree identifying input
- Prune non-reachable branches

\[ \# \text{ zeroes on any path} \leq \frac{N}{c} \]

\[ \frac{c^{N/c}}{2^{N/c}} \leq \# \text{ inputs} = \# \text{ leaves} \leq \binom{d \cdot N/c}{N/c} \]

query time \( d = \Omega(c) \)
Upper Bounds (2D, Indexing)

$O(1)$ time using $O(N)$ words

Atallah and Yuan (SODA 2010)

- Using two-levels of recursion, tabulating micro-blocks of size $\log \log m \times \log \log n$

$O(1)$ time using $O(N)$ bits
Upper Bounds (2D, Indexing) \textit{cont.}

\textbf{Thm} \( O(N/c \cdot \log c) \) bits and \( O(c \log c) \) query time

- Build \( \log c \) indexing structures for compressed matrices for block sizes \( 2^i \times c/2^i \), each using \( O(N/c) \) bits and can locate \( O(1) \) blocks with minimum key in \( O(1) \) time

- **Query**: \( O(1) \) blocks for each block size in time \( O(c) + \) elements not covered by blocks in time \( O(c \log c) \)
Lower Bounds (2D, Indexing)

- As for 1D consider \(\{0,1\}\) matrices and partition the array into blocks of \(c\) elements each containing exactly one zero.

- As for 1D an algorithm being able to identify the zero in each block using \(N/c\) bits will require time \(\Omega(c)\).
### Upper Bounds (2D, Encoding)

- **Translate input matrix into rank matrix using** \( O(N \log N) \) **bits**
- **Apply index structure to rank matrix using** \( O(N) \) **bits achieving** \( O(1) \) **query time**

<table>
<thead>
<tr>
<th></th>
<th>29</th>
<th>-14</th>
<th>10</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>7</td>
<td>0</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>-4</td>
<td>-5</td>
<td>-1</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>-17</td>
<td>32</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>15</th>
<th>2</th>
<th>10</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>9</td>
<td>6</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>5</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>13</td>
<td>1</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

**input matrix**

**rank matrix**
NEW Proof

Lower Bound (2D, Encoding)
Demaine et al. 2009

- Define a set of
  \[ \left( \frac{m}{2}! \right)^{\frac{n}{2}} - \frac{m}{4} \]
  matrices where the RMQ answers differ among all matrices

- Bits required is at least
  \[ \log \left( \frac{m}{2}! \right)^{\frac{n}{2}} - \frac{m}{4} = \Omega(N \log m) \]
Conclusion
1D Range Minimum Queries

**Indexing**

- **Upper Bound**
  - Fischer and Heun (2007)
  - Time: $O(1)$
  - Space: $2n + o(n) + |A|$ bits

- **Lower Bound**
  - Matching upper bound
  - Time: $\Omega(c)$
  - Space: $O(n/c) + |A|$ bits

**Encoding**

- **Upper Bound**
  - Fischer (Latin 2010)
  - Time: $O(1)$
  - Space: $2n + o(n)$ bits

- **Lower Bound**
  - Space: $2n - \Theta(\log n)$ bits
2D Range Minimum Queries

Indexing

- NEW Upper Bound
  - Time $= O(1)$
  - Space $= O(N) + |A|$ bits
  - Time $= O(c \log^2 c)$
  - Space $= O(N/c) + |A|$ bits

- NEW Lower Bound
  - Time $= \Omega(c)$
  - Space $= O(N/c) + |A|$ bits

Encoding

- Upper Bound
  - Time $= O(1)$
  - Space $= O(N \log n)$ bits

- Lower Bound:
  - Demain et al. (2009)
  - Space $= \Omega(N \log m)$ bits

NEW Proof
Thank You