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

Gerth Stølting Brodal
Pooya Davoodi

Aarhus University

S. Srinivasa Rao
Seoul National University
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

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

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

![Matrix showing data structure and minimum values](image)
Results
1D Range Minimum Queries

Indexing

Upper Bound
Fischer and Heun (2007)

\[ \text{Time} = O(1) \]
\[ \text{Space} = 2n + o(n) + |A| \text{ bits} \]

Lower Bound
(matching upper bound)

\[ \text{Time} = \Omega(c) \]
\[ \text{Space} = O(n/c) + |A| \text{ bits} \]

Encoding

Upper Bound
Fischer (Latin 2010)

\[ \text{Time} = O(1) \]
\[ \text{Space} = 2n + o(n) \text{ bits} \]

Lower Bound:
\[ \text{Space} = 2n - \Theta(\log n) \text{ bits} \]
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

Lower Bound

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

Encoding

NEW Upper Bound

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

NEW Proof

Lower Bound:

Space = $\Omega(N \log m)$ bits

Demain et al. (2009)
1D Encoding model
Upper bound
Lower bound
1D Index model
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** \( \frac{2^n}{n} / (n+1) \)
- **# bits ≥** \( \log \left( \frac{2^n}{n} \right) / (n+1) = 2n - \Theta(\log n) \)
Upper Bound (1D, Encoding)

- For an input array consider the Cartesian tree
- Succint 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 \( n/c \) bits implies \( \Omega(c) \) query time

- Consider \( n/C \) queries for \( c^{n/c} \) different \( \{0, 1\} \) inputs with exactly one zero in each block
- \( c^{n/c} / 2^{n/c} \) 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

- # zeroes on any path \( \leq n/c \)

- \( \frac{c^{n/c}}{2^{n/c}} \leq \) inputs = leaves \( \leq \binom{d \cdot n/c}{n/c} \)

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

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

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

Atallah and Yuan (SODA 2010)

$O(1)$ time using $O(N)$ words
Upper Bounds (2D, Indexing) cont.

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
NEW Proof

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

- Define a set of 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
(new, 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

NEW
Lower Bound

\( Time = \Omega(c) \)
\( Space = O(N/c) + |A| \) bits

Encoding

NEW Proof
Lower Bound:

Demain et al. (2009)

\( Time = O(1) \)
\( Space = \Omega(N \log n) \) bits

\( Time = \Omega(c) \)
\( Space = \Omega(N \log m) \) bits
Thank You