Succinct Data Structures for the Range Minimum Query Problems

S. Srinivasa Rao
Seoul National University

Joint work with
Gerth Brodal, Pooya Davoodi, Mordecai Golin, Roberto Grossi
John Iacono, Danny Kryzanc, Moshe Lewenstein,
Gonzalo Navarro, Rajeev Raman and Sunil Shende
Range Minimum Query (RMQ) Problem

• Input: an array $A$ of size $n$.
• Preprocess the array s.t. range minimum queries (RMQs) are supported efficiently.
• $\text{RMQ}(i,j)$ – returns the position of the smallest element in the sub-array $A[i,j]$

• Eg. $\text{RMQ}(4,7) = 6$
1D RMQ

- One of the classical data structure problems with lots of applications.
- Has been studied extensively.
- Linear-space data structures have been designed that achieve constant query-time by
  - [Harel and Tarjan, ’84]
  - [Schieber and Vishkin, ’88]
  - [Berkman et al., ’89]
  - [Bender and Farach-Colton, ’00]
  - [Alstrup et al., ’02]
  -...
1D RMQ

- All these data structures take linear space – $O(n \log n)$ bits.

- [Sadakane, ’03] improved the space to $4n+o(n)$ bits.

- [Fischer, ’10] further improved it to $2n+o(n)$ bits.

- Most of these structures are based on Cartesian tree.
Cartesian tree

- A binary tree with nodes labeled by the indices in the array
- The root is labeled by the position, \( p \), of min. of \( A \).
- The left and right subtrees are the Cartesian trees for the sub-arrays \( A[1, p-1] \) and \( A[p+1, n] \).
- \( RMQ(i, j) = LCA(i, j) \).
Models

Encoding model

- Queries can access **data structure** but not input matrix

Indexing model

- Queries can access **data structure** and read **input matrix**
1D Range Minimum Queries

Indexing

Upper Bound

Time = \(O(1)\)
Space = \(2n + o(n) + |A|\) bits

Lower Bound

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

Encoding

Upper Bound

Time = \(O(1)\)
Space = \(2n + o(n)\) bits

Lower Bound: Space = \(2n - \Theta(\log n)\) bits

Fischer (2010)
Fischer and Heun (2007)
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 $\geq \log \left( \frac{2n}{n} / (n+1) \right) = 2n - \Theta(\log n)$
Represent the **Cartesian tree** of the input array.

Succint representation using $4n + o(n)$ bits and $O(1)$ query time \([Sadakane ~'07]\)

Improved to $2n + o(n)$ bits \([Fischer ~'10]\)
Upper Bounds (1D, Indexing)

- Build encoding $O(n/c)$ bit structure for block minimums
- RMQ = query to encoding structure + $3c$ elements, i.e. $O(c)$ query time
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 $\leq N/c$

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

query time $d = \Omega(c)$
The 2D Range Minimum Problem

Introduced by Amir et al. (2007) as a generalization of the 1D RMQ problem.

- Input: an $m \times n$-matrix of size $N = m \cdot n$, $m \leq n$.
- Preprocess the matrix s.t. range minimum queries are efficiently supported.

```
\begin{array}{cccccc}
  i_1 & i' & i_2 \\
  10 & 4 & 13 & 9 & 12 \\
  65 & 14 & 6 & 11 & 30 \\
  7 & 28 & 9 & 16 & 52 \\
  17 & 48 & 19 & 2 & 23 \\
\end{array}
```
### Some Obvious Bounds...

<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 diagram]

Minimum
### 2D RMQ

[Brodal et al., ’10]

#### Indexing

**Upper Bound**
- Time = $O(1)$
- Space = $O(N) + |A|$ bits

**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:**
- Space = $\Omega(N \log m)$ bits

Demaine et al. (2009)
### Lower Bounds (2D, Indexing)

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

- Any algorithm being able to identify the zero in each block using \(N/c\) bits will require \(\Omega(c)\) time.

<table>
<thead>
<tr>
<th>2D Encoding model</th>
<th>Index model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper bound</td>
<td></td>
</tr>
<tr>
<td>Lower bound</td>
<td>(\bullet)</td>
</tr>
</tbody>
</table>
Upper Bounds (2D, Indexing)

- \(O(1)\) time using \(O(N)\) words \([\text{Atallah and Yuan (SODA 2010)}]\) using \(O(N)\) preprocessing time.

- 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) 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 $O(c \log c)$
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
Upper Bounds (2D, Encoding)

- Store a Cartesian tree for every column
  - Space: $O(N)$ bits
- For every pair of rows $(i,j)$, consider the array $A_{(i,j)}$ where $A_{(i,j)}[k] = \min\{ A[r,k] \mid i \leq r \leq j \}$.
- Store a Cartesian tree for each $A_{(i,j)}$
  - Total space: $O(n m^2) = O(N m)$ bits
- Queries can be answered in $O(1)$ time.
Define a set of matrices where the RMQ answers differ for all the matrices.

Bits required is at least

$$\log \left( \frac{m}{2}! \right) \frac{n}{2} - \frac{m}{4} = \Omega(N \log m)$$
2D RMQ

Indexing

Upper Bound

Time = \( O(1) \)
Space = \( O(N) + |A| \) bits

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

Lower Bound

Encoding

Upper Bound

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

Lower Bound:

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

Demaine et al. (2009)
Effective Entropy

- “information content of the data structure”
  - Given a set of objects $S$,
  - a set of queries $Q$,
  - let $C$ be the set of equivalence classes of $S$ induced by $Q$ ($x, y \in S$ are equivalent iff they cannot be distinguished by queries in $Q$).
  - We want to store $x$ in $\lceil \lg |C| \rceil$ bits.

- Want encoding size to equal the effective entropy (exact constant if possible).
Effective Entropy: example

- Consider the 1D RMQ problem:
- We can use a Cartesian tree of A[1..n] to encode all the answers to RMQ queries.
  - Cartesian tree can be represented in $2n - O(lg n)$ bits.
  - Cartesian tree completely characterizes 1D case → effective entropy of 1D RMQ is $2n - O(lgn)$ bits.
- The low effective entropy of 1D RMQ is used in many space-efficient data structures.
2D RMQ

- Effective entropy for the 2D RMQ problem:

<table>
<thead>
<tr>
<th></th>
<th>( \Omega(n^2 \lg n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m = n )</td>
<td>( \Omega(nm \lg n) )</td>
</tr>
<tr>
<td>General</td>
<td>( \Omega(nm \lg m) )</td>
</tr>
<tr>
<td></td>
<td>( O(nm^2) )</td>
</tr>
</tbody>
</table>

- Closing the gap between the upper and lower bounds for the general case is an interesting open problem.
2D RMQ
Special cases
Results

- Random input matrix:

<table>
<thead>
<tr>
<th></th>
<th>1-sided</th>
<th>2-sided</th>
<th>3-sided</th>
<th>4-sided</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-sided</td>
<td>$\Theta((\log n)^2)$</td>
<td>$\Theta((\log n)^2 \log m)$</td>
<td>$\Theta(n(\log m)^2)$</td>
<td>$\Theta(nm)$</td>
</tr>
</tbody>
</table>

- Small values of $m$:

| $m = 2$  | $5n - O(\log n)$ | $\leq 8.32n$ | Lower bound for $m = 3$: $8n - O(\log n)$ | $\leq 7n - O(\log n)$ [Brodal et al. 2010] | $\leq 14.32n - O(\log n)$ |

- Random array: $\approx (1.736..)n$
2D RMQ for $m = 2$

- Simple solution [Brodal et al. 2010]:
  - Store CTs for T, B and for TB --- $6n$ bits
  - Store $n$ bits giving location of column-wise min.

- Approach based on merging CTs:
  - Store CTs for T and B as before.
  - For TB:
    - Use T, B to get row minima i and j.
    - Recurse on [1..j – 1] and [j + 1..n].
  - Total space: $5n$ bits.
2D RMQ encoding

- For \( m = 2 \):
  - We can also show a lower bound of \( 5n - O(\lg n) \) bits – “any \( n \)-bit sequence can be used to merge two CTs”.
  - We can answer 2D RMQ queries in \( (5+\varepsilon)n \) bits and \( O(1/\varepsilon) \) query time, for any \( \varepsilon > 0 \).

- For \( m = 3 \):
  - Lower bound: \( 8n - O(\lg n) \)
  - Upper bound: \( 8.32n \)
  - Data structure?
2D RMQ

Indexing

Upper Bound

Time $= O(1)$
Space $= O(N) + |A|$ bits

Upper Bound

Time $= O(c \log^2 c)$
Space $= O(N/c) + |A|$ bits

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: $\Omega(N \log m)$ bits

Demaine et al. (2009)
2D RMQ index

- Space – $O(N/c)$ bits
- Time – $O(c \log c \ (\log \log c)^2)$

- Uses
  - Fibonacci point set [Fiat and Shamir, ‘00]
  - Succinct index for 2-sided geometric RMQ [Farzan et al. ‘12]
Encoding 1D range top-k queries

- Generalization of 1D RMQ problem: encode a 1D array to support range top-k queries. [Indexing problem can be solved (efficiently?) using CT.]

- Special case: prefix top-k queries.
  - Optimal bounds are known.
Prefix top-k queries

- prefix-top-3-positions(5) = {1, 3, 5}
- prefix-top-3-values(5) = {2, 7, 8}
- prefix-3\textsuperscript{rd}-position(5) = 5
- prefix-3\textsuperscript{rd}-value(5) = 8
Prefix top-k results

- For an array $A$ of size $n$, we can support:
  - prefix-kth-value$(i)$ in $O(1)$ time using $\Theta(n)$ bits (assuming the values of $A$ are polynomial in $n$).
  - prefix-kth-position$(i)$ in $O(1)$ time, and
  - prefix-top-k-values$(i)$ and prefix-top-k-positions in $O(k)$ time using
    - $\Theta(n \log k)$ bits.
2-sided top-k queries

- Given an array of length n, top-k-positions(i, j) and k-th-smallest-position(i, j) can be supported in $O(k)$ time, using an encoding of size $O(n \log k)$ bits.
  - Imply the results for prefix-top-k queries.
  - More complex data structures.
Conclusions

- Various time-space trade-offs for the RMQ problem in the indexing and encoding models.
  - Encodings for specific (small) values of the parameter.
  - Encodings for random inputs.
- Top-k encodings
  - Prefix queries: almost tight bounds.
  - 2-sided queries: some trade-offs.
Open problems

- **2D RMQ:**
  - Closing the gap between the upper and lower bounds in the indexing model for 2D RMQ.
  - Closing the gap(s) in the encoding model.
  - Improving the bounds for small m.
  - Higher dimensions.

- **Top-k encodings:**
  - Better bounds (and data structures ?) for the 2-sided kth-smallest queries.
Thank you