Practical Range Minimum Queries Revisited

Niklas Baumstark, Simon Gog, Tobias Heuer, Julian Labeit | August 21, 2019
Range Minimum Query Problem

Definition

Given an array $A[1..n]$ of $n$ numbers. The range minimum query (RMQ) problem is to find an index structure that returns for any range $A[i..j]$ the position of the leftmost minimum.

$A = \begin{array}{cccccccccc}
5 & 2 & 3 & 1 & 4 & 3 & 2 & 4 & 2 & 3 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\end{array}$

$RMQ_A(3, 8) = 4$
Range Minimum Query Problem

Definition

Given an array $A[1..n]$ of $n$ numbers. The range minimum query (RMQ) problem is to find an index structure that returns for any range $A[i..j]$ the position of the leftmost minimum.

$A = \begin{array}{cccccccccc}
5 & 2 & 3 & 1 & 4 & 3 & 2 & 4 & 2 & 3 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10
\end{array}$

$RMQ_A(7, 9) = 7$
RMQs have a wide range of applications [FH11]

- Text-Indexing, Pattern Matching
- String mining
- Text compression
- Document Retrieval
- Tree, Graphs, Bioinformatics, etc.
## Applications

### Document Listing

<table>
<thead>
<tr>
<th>Document 1</th>
<th>Document 2</th>
<th>Search Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>ananas</td>
<td>bananas</td>
<td>ana</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Doc</th>
<th>Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>2</td>
<td>#$</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>#bananas#$</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>$</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>ananas#$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>ananas#bananas#$</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>anas#$</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>anas#bananas#$</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>as#$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>as#bananas#$</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>bananas#$</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>nanas#$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>nanas#bananas#$</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>nas#$</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>nas#bananas#$</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>s#$</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>s#bananas#$</td>
</tr>
</tbody>
</table>
Applications

Suffix Tree Traversal

Introduction
Related Work
A Optimized Recursive Solution
Experiments
References

Baumstark, Gog, Heuer, Labeit – Practical Range Minimum Queries Revisited

August 21, 2019 5/25
### Overview

<table>
<thead>
<tr>
<th>Preprocessing</th>
<th>Space</th>
<th>Query Time</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRECOMPUTE</td>
<td>$O(n^2)$</td>
<td>$O(n^2)$   words</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>SCAN</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td></td>
</tr>
<tr>
<td>SPARSE TABLE</td>
<td>$O(n \log n)$</td>
<td>$O(n \log n)$ words</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Bender &amp; Farach-Colton</td>
<td>$O(n)$</td>
<td>$O(n)$ words</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Sadakane</td>
<td>$O(n)$</td>
<td>$4n + o(n)$ bits</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Fischer &amp; Heun</td>
<td>$O(n)$</td>
<td>$2n + o(n)$ bits</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Ferrada &amp; Navarro</td>
<td>$O(n)$</td>
<td>$2n + o(n)$ bits</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>

Note, succinct solution are only constant in theory.

In practice **highly engineered logarithmic solutions** are used to lower the $o(n)$ space term.

⇒ Time-Space Tradeoff
## Overview

<table>
<thead>
<tr>
<th></th>
<th>Preprocessing</th>
<th>Space</th>
<th>Query Time</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRECOMPUTE</td>
<td>$O(n^2)$</td>
<td>$O(n^2)$ words</td>
<td>$O(1)$</td>
<td></td>
</tr>
<tr>
<td>SCAN</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
<td></td>
</tr>
<tr>
<td>SPARSETABLE</td>
<td>$O(n \log n)$</td>
<td>$O(n \log n)$ words</td>
<td>$O(1)$</td>
<td>[BF00]</td>
</tr>
<tr>
<td>Bender &amp; Farach-Colton</td>
<td>$O(n)$</td>
<td>$O(n)$ words</td>
<td>$O(1)$</td>
<td>[BF00]</td>
</tr>
<tr>
<td>Sadakane</td>
<td>$O(n)$</td>
<td>$4n + o(n)$ bits</td>
<td>$O(1)$</td>
<td>[Sad07]</td>
</tr>
<tr>
<td>Fischer &amp; Heun</td>
<td>$O(n)$</td>
<td>$2n + o(n)$ bits</td>
<td>$O(1)$</td>
<td>[FH11]</td>
</tr>
<tr>
<td>Ferrada &amp; Navarro</td>
<td>$O(n)$</td>
<td>$2n + o(n)$ bits</td>
<td>$O(1)$</td>
<td>[FN16]</td>
</tr>
</tbody>
</table>

Note, succinct solution are only **constant in theory**.

In practice **highly engineered logarithmic solutions** are used to lower the $o(n)$ space term.

$\Rightarrow$ **Time-Space Tradeoff**
Contributions

\[ |A| = n \]

\[ \pm 1 \text{RMQ} \]

\[ |E'| = \lceil \frac{n}{b} \rceil \]

\[ |E'| \log n \notin o(n) \quad |E'| \log n \in o(n) \]

\[ \Rightarrow \text{SparseTable} \]

1. Using a recursive data layout
2. New SELECT\(_1\) implementation on special bit vectors
3. Height minimization strategy for Generalized Cartesian Trees

⇒ Factor 3 faster than the current fastest solution

---

Introduction

Related Work

A Optimized Recursive Solution

Experiments

References

Baumstark, Gog, Heuer, Labeit – Practical Range Minimum Queries Revisited

August 21, 2019
The Cartesian Tree [Vui80]

\[ A = \begin{bmatrix}
5 & 2 & 3 & 1 & 4 & 3 & 2 & 4 & 2 & 3 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10
\end{bmatrix} \]
The Cartesian Tree [Vui80]

\[ A = \begin{bmatrix} 5 & 2 & 3 & 1 & 4 & 3 & 2 & 4 & 2 & 3 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{bmatrix} \]

Introduction

Related Work

A Optimized Recursive Solution

Experiments

References

Baumstark, Gog, Heuer, Labeit – Practical Range Minimum Queries Revisited

August 21, 2019
The Cartesian Tree [Vui80]

A =

\[
\begin{array}{cccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
5 & 2 & 3 & 1 & 4 & 3 & 2 & 4 & 2 & 3
\end{array}
\]
The Cartesian Tree [Vui80]
Reducing RMQ to LCA [GBT84]

\[ A = \begin{array}{cccccccc}
5 & 2 & 3 & 1 & 4 & 3 & 2 & 4 & 2 & 3 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10
\end{array} \]

\[ \text{RMQ}_A(5, 10) = 7 \]

\[ \text{LCA}_C(A)(5, 10) = 7 \]
Balanced Parentheses [MR01]

\[BP = \begin{pmatrix}
1 & 5 & 2 & 3 & 1 & 4 & 3 & 2 & 4 & 2 & 3
\end{pmatrix}\]

A =

\[
\begin{array}{cccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
5 & 2 & 3 & 1 & 4 & 3 & 2 & 4 & 2 & 3
\end{array}
\]
Balanced Parentheses [MR01]

\[ BP = \left(\right) \]

\[ A = \begin{array}{cccccccccc}
5 & 2 & 3 & 1 & 4 & 3 & 2 & 4 & 2 & 3 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10
\end{array} \]
Balanced Parentheses [MR01]

\[ BP = \left( \left( \left( \right) \right) \right) \]

\[ A = \begin{bmatrix}
5 & 2 & 3 & 1 & 4 & 3 & 2 & 4 & 2 & 3 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10
\end{bmatrix} \]
Balanced Parentheses [MR01]

\[ BP = \left( \left( \left( \right) \right) \right) \]

\[ A = \begin{array}{cccccccc} 5 & 2 & 3 & 1 & 4 & 3 & 2 & 4 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \]
Balanced Parentheses [MR01]

\[ BP = \left( \left( \left( \right) \right) \right) \]

\[ A = \begin{array}{cccccccccc}
5 & 2 & 3 & 1 & 4 & 3 & 2 & 4 & 2 & 3 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10
\end{array} \]
Balanced Parentheses [MR01]

\[ BP = ( ( ( ) ) ( ) ) \]

\[ A = \begin{bmatrix} 5 & 2 & 3 & 1 & 4 & 3 & 2 & 4 & 2 & 3 \end{bmatrix} \]
Balanced Parentheses [MR01]

\[ BP = \left( \left( \left( \left( \left( \right) \right) \right) \right) \right) \]

\[ A = \begin{array}{cccccccccc}
5 & 2 & 3 & 1 & 4 & 3 & 2 & 4 & 2 & 3 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 
\end{array} \]
Balanced Parentheses [MR01]

\[ BP = \left( \left( \left( \left( \left( \right) \right) \right) \right) \right) \]

\[ A = \begin{bmatrix} 5 & 2 & 3 & 1 & 4 & 3 & 2 & 4 & 2 & 3 \end{bmatrix} \]
Balanced Parentheses [MR01]

\[ BP = 1 \left( 2 \left( 3 \left( \right) \right) \right) \left( 5 \left( 6 \left( 7 \left( \right) \right) \right) \right) \left( 8 \left( 9 \left( \right) \right) \right) \left( 10 \right) \right) \]

\[ A = \begin{array}{cccccccccc}
5 & 2 & 3 & 1 & 4 & 3 & 2 & 4 & 2 & 3 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\end{array} \]
A node with INORDER $i$, which is equal to its index in $A$, is mapped to its PREORDER position in the $BP$ sequence.
Given a bitvector $B[1..n]$ (where $B[i] \in \{0, 1\} \ \forall [1, n]$)

- $\text{RANK}_1(i, B) =$ counts the number of 1’s in $B[1..i]$
- $\text{SELECT}_1(i, B) =$ return the index of the $i$-th 1 in $B$

$\Rightarrow$ solvable in $O(1)$ time and $o(n)$ bits space [Jac88]
Given a bitvector $B[1..n]$ (where $B[i] \in \{0, 1\} \forall [1, n]$)

- $\text{RANK}_1(i, B) =$ counts the number of 1’s in $B[1..i]$
- $\text{SELECT}_1(i, B) =$ return the index of the $i$-th 1 in $B$

⇒ solvable in $O(1)$ time and $o(n)$ bits space [Jac88]
Reducing LCA to ±1RMQ [BV93]

A =

\[
L[i] = \text{Depth}(v_i)
\]

\[
L = 
\begin{pmatrix}
1 & 2 & 3 & 2 & 3 & 2 & 1 & 2 & 3 & 4 & 3 & 2 & 3 & 4 & 3 & 4 & 3 & 2 & 1 & 0
\end{pmatrix}
\]

\[
BP = ( ( ( ) ) ( ) ) ( ( ( ) ) ) ( ( ) ( ) ) ( ) ( ) ( ) )
\]
Reducing LCA to ±1RMQ [BV93]

\[
A = \begin{array}{cccccccccc}
5 & 2 & 3 & 1 & 4 & 3 & 2 & 4 & 2 & 3 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10
\end{array}
\]

\[\text{RMQ}_A(5, 10) = 7\]

\[\text{LCA}_C(A)(5, 10) = 7\]

\[\Rightarrow L[i] =: \text{EXCESS}(i, BP) = 2\text{RANK}_1(i, BP) - i\]
Generalized Cartesian Tree [MR01]

Leftmost Path Mapping
- Add a pseudo root to the tree
- Connect the pseudo root with all nodes on the leftmost path of the root of $C(A)$
- Continue recursively with the childs of the pseudo root

\[ BP = ( ( ( ) ) ( ) ( ) ( ) ( ) ( ) ( ) ) ) \]
\[ A = \begin{array}{cccccccccccc}
0 & 5 & 2 & 3 & 1 & 4 & 3 & 2 & 4 & 2 & 3 & 5 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 
\end{array} \]

$A$ node with $\text{INORDER } i$ in $C(A)$ corresponds to a node with $\text{PREORDER } i$ in $C_L(A)$.

How to solve the RMQ problem? [FN16]

\[ \text{RMQ}_A(i, j) = \text{RANK}_1(\text{RRMQ}_{BP(CL)}^{\pm}(\text{SELECT}_1(i + 1) - 1, \text{SELECT}_1(j + 1))) \]

\textbf{Note:} $\pm 1 \text{RRMQ}$ has to return the rightmost minimum EXCESS value in $BP$.
Generalized Cartesian Tree [MR01]

Leftmost Path Mapping

Rightmost Path Mapping

Introduction

Relation Work

A Optimized Recursive Solution

Experiments

References

Baumstark, Gog, Heuer, Labeit – Practical Range Minimum Queries Revisited

August 21, 2019

13/25
Solving $\pm 1$RMQ in practice - *rmM-Tree*

Nodes are enhanced with additional informations to solve *EXCESS* and *SELECT* [FN16] $\Rightarrow o(n)$ extra space and $O(\log n)$ running time.
Solving $\pm 1$RMQ - Recursive Approach

Recursion or terminate with SparseTable

Build $BP$

Build $C^L(E)$

$E =$

Recursion or terminate with SparseTable

$b$ bit block $b$ bit block $b$ bit block $b$ bit block

Introduction
Related Work
A Optimized Recursive Solution
Experiments
References

Baumstark, Gog, Heuer, Labeit – Practical Range Minimum Queries Revisited
Lemma

The position of the representing opening parenthesis in BP of a node \( v \in T \) with \( \text{PREORDER}(v) = i \) is \( \text{SELECT}_1(i, BP) = 2i - \text{DEPTH}(v_i) \), where \( \text{DEPTH}(v) \geq 0 \) denotes the distance of \( v \) to the root node.

\( BP = \)
**Lemma**

The position of the representing opening parenthesis in $BP$ of a node $v \in T$ with $\text{PREORDER}(v) = i$ is $\text{SELECT}_1(i, BP) = 2i - \text{DEPTH}(v_i)$, where $\text{DEPTH}(v) \geq 0$ denotes the distance of $v$ to the root node.

$$BP = (0$$

$$\text{SELECT}_1(0, BP) = 2 \cdot 0 - 0 = 0$$
Lemma

The position of the representing opening parenthesis in $BP$ of a node $v \in T$ with $\text{PREORDER}(v) = i$ is

$$\text{SELECT}_1(i, BP) = 2i - \text{DEPTH}(v_i),$$

where $\text{DEPTH}(v) \geq 0$ denotes the distance of $v$ to the root node.

$BP = (0)$

$\text{SELECT}_1(1, BP) = 2 \cdot 1 - 1 = 1$
Lemma

The position of the representing opening parenthesis in $BP$ of a node $v \in T$ with $\text{PREORDER}(v) = i$ is

$$\text{SELECT}_1(i, BP) = 2i - \text{DEPTH}(v_i),$$

where $\text{DEPTH}(v) \geq 0$ denotes the distance of $v$ to the root node.

$$BP = \begin{pmatrix} 0 & 1 & 3 \end{pmatrix}$$

$$\text{SELECT}_1(2, BP) = 2 \cdot 2 - 1 = 3$$
**Lemma**

The position of the representing opening parenthesis in $BP$ of a node $v \in T$ with $\text{PREORDER}(v) = i$ is $\text{SELECT}_1(i, BP) = 2i - \text{DEPTH}(v_i)$, where $\text{DEPTH}(v) \geq 0$ denotes the distance of $v$ to the root node.

$BP = (0 \ 1 \ 3 \ 4)$

$\text{SELECT}_1(3, BP) = 2 \cdot 3 - 2 = 4$
Lemma

The position of the representing opening parenthesis in BP of a node $v \in T$ with $\text{PREORDER}(v) = i$ is $\text{SELECT}_1(i, BP) = 2i - \text{DEPTH}(v_i)$, where $\text{DEPTH}(v) \geq 0$ denotes the distance of $v$ to the root node.

$BP = 0(1)347$

$\text{SELECT}_1(4, BP) = 2 \cdot 4 - 1 = 7$
Lemma

The position of the representing opening parenthesis in BP of a node \( v \in T \) with \( \text{PREORDER}(v) = i \) is \( \text{SELECT}_1(i, BP) = 2i - \text{DEPTH}(v_i) \), where \( \text{DEPTH}(v) \geq 0 \) denotes the distance of \( v \) to the root node.

Consequences

- Solve \( \text{SELECT}_1(i, BP) \) with binary search in interval 
  \( [2i - \text{MAXDEPTH}, 2i] \) with \( \text{RANK}_1(i, BP) \)
Minimize the depth of the Cartesian Tree

- Cartesian Tree over $A$ with \textbf{leftmost} minimum as root
- Use \textbf{rightmost} path mapping for Generalized Cartesian Tree

$A = \begin{bmatrix} 5 & 2 & 3 & 1 & 4 & 3 & 2 & 4 & 2 & 3 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{bmatrix}$

$\overleftarrow{A} = \begin{bmatrix} 3 & 2 & 4 & 2 & 3 & 4 & 1 & 3 & 2 & 5 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{bmatrix}$

$\Rightarrow C^R_L(A)$ and $C^L_R(\overleftarrow{A})$ are isomorphic.
Minimize the depth of the Cartesian Tree

- Build $C_L^L(A)$ and $C_R^L(\overleftarrow{A}) \Rightarrow$ Continue with Cartesian Tree with minimal height
- Query logic does not change for $\pm 1$RMQ $\Rightarrow$ only mirror result of a RMQ call on $C_R^L(\overleftarrow{A})$
- Accelerates SELECT

$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}$
$\overleftarrow{A} = \begin{bmatrix} 5 & 4 & 3 & 2 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}$
Parameter Tuning

Query Time

$n = 10^9$

Block size $s$
- 1024
- 2048
- 4096

Max. number of recursion $\Lambda$
- 1
- 2
- 3

Size of query range [$10^x$]

Time per query [$\mu s$]

Introduction

Related Work

A Optimized Recursive Solution

Experiments

References

Baumstark, Gog, Heuer, Labeit – Practical Range Minimum Queries Revisited

August 21, 2019
## Parameter Tuning

### Space

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Space in bits per element with varying $n$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$n = 10^6$</td>
</tr>
<tr>
<td>$s = 1024$, $\Lambda = 1$</td>
<td>2.37</td>
</tr>
<tr>
<td>$s = 2048$, $\Lambda = 1$</td>
<td>2.24</td>
</tr>
<tr>
<td>$s = 4096$, $\Lambda = 1$</td>
<td>2.18</td>
</tr>
<tr>
<td>$s = 1024$, $\Lambda = 2$</td>
<td>2.16</td>
</tr>
<tr>
<td>$s = 2048$, $\Lambda = 2$</td>
<td>2.14</td>
</tr>
<tr>
<td>$s = 4096$, $\Lambda = 2$</td>
<td>2.14</td>
</tr>
<tr>
<td>$s = 1024$, $\Lambda = 3$</td>
<td>2.16</td>
</tr>
<tr>
<td>$s = 2048$, $\Lambda = 3$</td>
<td>2.14</td>
</tr>
<tr>
<td>$s = 4096$, $\Lambda = 3$</td>
<td>2.14</td>
</tr>
</tbody>
</table>
Comparison with other Implementations

Random Input

<table>
<thead>
<tr>
<th>Size of query range [10^x]</th>
<th>n=10^8</th>
<th>n=10^9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time [μs]</td>
<td>Time [μs]</td>
</tr>
<tr>
<td>1</td>
<td>SDSL</td>
<td>NEWRMQ</td>
</tr>
<tr>
<td>2</td>
<td>Succinct</td>
<td>F&amp;N'16</td>
</tr>
</tbody>
</table>

Introduction

Related Work

A Optimized Recursive Solution

Experiments

References

Baumstark, Gog, Heuer, Labeit – Practical Range Minimum Queries Revisited

August 21, 2019
Comparison with other Implementations

Random Input

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Space in bits per element with varying $n$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$n = 10^6$</td>
</tr>
<tr>
<td>SDSL</td>
<td>2.61</td>
</tr>
<tr>
<td>SUCCINCT</td>
<td>2.70</td>
</tr>
<tr>
<td>F&amp;N'16</td>
<td>2.10</td>
</tr>
<tr>
<td>NEWRMQ</td>
<td>2.16</td>
</tr>
</tbody>
</table>
Comparison with other Implementations

Increasing Input

<table>
<thead>
<tr>
<th>Size of query range [$10^x$]</th>
<th>n=10^8</th>
<th>n=10^9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time [$\mu s$]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **SDSL**
- **SUCCINCT**
- **F&N’16**
- **NEWRMQ**

Introduction

Related Work

A Optimized Recursive Solution

Experiments

References

Baumstark, Gog, Heuer, Labeit – Practical Range Minimum Queries Revisited

August 21, 2019
Comparison with other Implementations

Decreasing Input

Size of query range $[10^x]$
Traversing the *Suffix Tree* with RMQ

### Time per query [ns]

<table>
<thead>
<tr>
<th></th>
<th>dblp</th>
<th>dna</th>
<th>english</th>
<th>sources</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SDSL</strong></td>
<td>160</td>
<td>155</td>
<td>154</td>
<td>165</td>
</tr>
<tr>
<td><strong>SUCCINCT</strong></td>
<td>237</td>
<td>217</td>
<td>221</td>
<td>235</td>
</tr>
<tr>
<td><strong>F&amp;N’16</strong></td>
<td>284</td>
<td>270</td>
<td>259</td>
<td>278</td>
</tr>
<tr>
<td><strong>NEWRMQ</strong></td>
<td>127</td>
<td>133</td>
<td>136</td>
<td>139</td>
</tr>
</tbody>
</table>

**Legend:**
- SDSL
- SUCCINCT
- F&N’16
- NEWRMQ

---

**Introduction**

**Related Work**

**A Optimized Recursive Solution**

**Experiments**

**References**

Baumstark, Gog, Heuer, Labeit – Practical Range Minimum Queries Revisited

August 21, 2019


Additional Implementations Details

\[ \text{RMQ}_A(i, j) = \text{RANK}_1(\text{RRMQ}^{\pm}_{BP(C^L)}(\text{SELECT}_1(i + 1) - 1, \text{SELECT}_1(j + 1))) \]

- If \( j - i \leq \log n \), then try to avoid second \text{SELECT} by simple scan of the next \( 2 \log n \) bits starting at position \( \text{SELECT}_1(i + 1) \)
- For large tree sizes use sampling scheme (\( o(n) \) space) for \text{SELECT} to prevent \( \mathcal{O}(\log n) \) running time
Parameter Tuning

Query Time

$n = 10^9$

Block size $s$
- 1024
- 2048
- 4096

Max. number of recursion $\Lambda$
- 1
- 2
- 3

Size of query range $[10^x]$
Query Time Breakdown

The figure shows the time per query in nanoseconds (ns) for different sizes of query range, denoted as $10^x$, for $n = 10^9$ and $n = 10^{10}$. The graph is divided into two panels, each showing the distribution of time for basic operations such as PARSE, TABLE, RANK, ACCESS, E & I, SCAN, and SELECT.

- For $n = 10^9$, the time increases significantly with the size of the query range.
- For $n = 10^{10}$, the time further increases, with a notable increase in the time spent on SELECT operations.

The x-axis represents the size of the query range in $10^x$ format, and the y-axis shows the time per query in nanoseconds.
Comparison with other Implementations

Worst Case Input

<table>
<thead>
<tr>
<th>Size of query range $[10^x]$</th>
<th>Time [$\mu$s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n=10^8</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

- **SDL**
- **SUCCINCT**
- **F&N’16**
- **NEWRMQ**

Baumstark, Gog, Heuer, Labeit – Practical Range Minimum Queries Revisited

August 21, 2019 4/6
Comparison with other Implementations

Increasing Input

<table>
<thead>
<tr>
<th>δ = 0</th>
<th>δ = 100</th>
<th>δ = 10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDSL</td>
<td>SUCCINCT</td>
<td>F&amp;N’16</td>
</tr>
<tr>
<td>NEWRMQ</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Size of query range [10^x]

Time [µs]
Comparison with other Implementations

Decreasing Input

<table>
<thead>
<tr>
<th>$\delta = 0$</th>
<th>$\delta = 100$</th>
<th>$\delta = 10000$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time [$\mu s$]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size of query range [$10^x$]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- SDSL
- Succinct
- F&N’16
- NewRMQ