Advanced Algorithms – COMS31900

2014/2015

Range Minimum Queries

Benjamin Sach

Range minimum query

\[ i = 5 \quad j = 11 \]

Preprocess an integer array \( A \) (length \( n \)) to answer range minimum queries.

After preprocessing, a range minimum query is given by \( \text{RMQ}(i,j) \) the output is the location of the smallest element in \( A[i,j] \).

\[ \text{RMQ}(3,7) = 6, \text{ RMQ}(5,11) = 8 \]

- We will discuss several algorithms which give trade-offs between space used, prep. time and query time.
- Ideally we would like \( O(n) \) space, \( O(n) \) prep. time and \( O(1) \) query time.

Block decomposition

1. Find the largest block which is completely contained within the query interval but doesn’t overlap a block you chose before.
2. The minimum is the smallest in all these blocks.

Repeat:

- Find the largest block which is completely contained within the query interval but doesn’t overlap a block you chose before.
- The minimum is the smallest in all these blocks because they cover the query.
How do we find RMQ(i, j)?

**Key Idea** precompute the answers for every interval of length 2, 4, 8, 16, ...

- The array \( R_2 \) stores RMQ(i, i + 1) for all i
- The array \( R_4 \) stores RMQ(i, i + 3) for all i
- The array \( R_8 \) stores RMQ(i, i + 7) for all i
- The array \( R_k \) stores RMQ(i, i + k − 1) for all i

We build \( R_2 \) from \( A \) in \( O(n) \) time

We build \( R_k \) for \( k = 2, 4, 8, 16, \ldots \leq n \)

How do we compute RMQ(i, j)?

- If the interval length, \( k = (j - i + 1) \), is a power of two, just look up the answer.
- Otherwise, find the \( k = 2, 4, 8, 16, \ldots \) such that \( k \leq \ell < 2k \)

This takes \( O(1) \) time but why does it work?

---

Range minimum query (intermediate) summary

Preprocess an integer array \( A \) (length \( n \)) to answer range minimum queries...

<table>
<thead>
<tr>
<th>( i )</th>
<th>( j )</th>
<th>RMQ(i, j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>6</td>
</tr>
</tbody>
</table>

After preprocessing, a range minimum query is given by \( RMQ(i, j) \)

- the output is the location of the smallest element in \( A[i, j] \)

**Solution 1**

- \( O(n) \) space
- \( O(n) \) prep time
- \( O(\log n) \) query time

**Solution 2**

- \( O(n \log n) \) space
- \( O(n \log n) \) prep time
- \( O(1) \) query time

**Solution 3**

- \( O(n \log \log n) \) space
- \( O(n \log \log n) \) prep time
- \( O(1) \) query time

---

Low-resolution RMQ

Key Idea replace \( A \) with a smaller, 'low resolution' array \( H \)

- and many small arrays \( L_0, L_1, L_2, \ldots \) for the details

<table>
<thead>
<tr>
<th>( i )</th>
<th>( j )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_0 )</td>
<td>( L_1 )</td>
</tr>
<tr>
<td>( L_2 )</td>
<td>( \ldots )</td>
</tr>
</tbody>
</table>

How do we answer a query in \( A \)?

- Do at most one query in \( H \)...
- and one query in at most two different \( L_j \) (here we query \( L_1 \) and \( L_2 \))
- then take the smallest

This takes \( O(1) \) total query time

**Solution 4**

- \( O(n \log \log n) \) space
- \( O(n \log \log n) \) prep time
- \( O(1) \) query time
Range minimum query summary

Preprocess an integer array $A$ (length $n$) to answer range minimum queries...

$A = [21, 7, 13, 12, 7, 17, 14, 16, 9, 27, 14]$

After preprocessing, a range minimum query is given by $RMQ(i, j)$

the output is the location of the smallest element in $A[i, j]$

<table>
<thead>
<tr>
<th>Solution 1</th>
<th>Solution 2</th>
<th>Solution 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n)$ space</td>
<td>$O(n \log n)$ space</td>
<td>$O(n \log \log n)$ space</td>
</tr>
<tr>
<td>$O(n)$ prep time</td>
<td>$O(n \log n)$ prep time</td>
<td>$O(n \log n)$ prep time</td>
</tr>
<tr>
<td>$O(\log n)$ query time</td>
<td>$O(1)$ query time</td>
<td>$O(1)$ query time</td>
</tr>
</tbody>
</table>

Can we do $O(n)$ space and $O(1)$ query time? 

Yes... but not until next lecture