Advanced Algorithms – COMS31900

2014/2015

Range Minimum Queries

Benjamin Sach
Range minimum query

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

<p>| | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>8</td>
<td>73</td>
<td>51</td>
<td>82</td>
<td>19</td>
<td>32</td>
<td>5</td>
<td>67</td>
<td>91</td>
<td>14</td>
<td>46</td>
<td>9</td>
</tr>
</tbody>
</table>

$n$
Range minimum query

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

After preprocessing, a **range minimum query** is given by $\text{RMQ}(i,j)$.
Range minimum query

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 $A[i, j]$
Range minimum query

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 $A[i, j]$
Range minimum query

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 $A[i, j]$
Range minimum query

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 $A[i, j]$
Range minimum query

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 $A[i, j]$.

e.g. $\text{RMQ}(3, 7) = 6$, which is the location of the smallest element in $A[3, 7]$.
Range minimum query

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

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

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

e.g. $RMQ(3, 7) = 6$, which is the location of the smallest element in $A[3, 7]$

e.g. $RMQ(5, 11) = 8$, which is the location of the smallest element in $A[5, 11]$
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 $A[i, j]$

- e.g. $\text{RMQ}(3, 7) = 6$, which is the location of the smallest element in $A[3, 7]$
- e.g. $\text{RMQ}(5, 11) = 8$, which is the location of the smallest element in $A[5, 11]$

- We will discuss several algorithms which give trade-offs between
  space used, prep. time and query time
Range minimum query

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 $A[i, j]$

e.g. $\text{RMQ}(3, 7) = 6$, which is the location of the smallest element in $A[3, 7]$

e.g. $\text{RMQ}(5, 11) = 8$, which is the location of the smallest element in $A[5, 11]$

- 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

\[
A = \begin{bmatrix}
23 & 17 & 8 & 73 & 51 & 82 & 19 & 32 & 5 & 67 & 91 & 14 & 46 & 9 & 21 & 54 \\
\end{bmatrix}
\]
Block decomposition

\[
A = \begin{array}{cccccccccccc}
\end{array}
\]
Block decomposition

A

<table>
<thead>
<tr>
<th>23</th>
<th>17</th>
<th>8</th>
<th>73</th>
<th>51</th>
<th>82</th>
<th>19</th>
<th>32</th>
<th>5</th>
<th>67</th>
<th>91</th>
<th>14</th>
<th>46</th>
<th>9</th>
<th>21</th>
<th>54</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

smallest from each pair
Block decomposition

<table>
<thead>
<tr>
<th>A</th>
<th>23</th>
<th>17</th>
<th>8</th>
<th>73</th>
<th>51</th>
<th>82</th>
<th>19</th>
<th>32</th>
<th>5</th>
<th>67</th>
<th>91</th>
<th>14</th>
<th>46</th>
<th>9</th>
<th>21</th>
<th>54</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

smallest from each pair
Block decomposition

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>17</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>17</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>51</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>82</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>19</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>19</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>19</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>32</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>11</td>
<td>67</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>91</td>
<td>11</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>12</td>
</tr>
<tr>
<td>14</td>
<td>46</td>
<td>13</td>
</tr>
<tr>
<td>15</td>
<td>9</td>
<td>14</td>
</tr>
<tr>
<td>16</td>
<td>21</td>
<td>15</td>
</tr>
<tr>
<td>17</td>
<td>54</td>
<td></td>
</tr>
</tbody>
</table>

The smallest number from each pair is highlighted.
### Block decomposition

Block decomposition involves dividing a matrix into smaller blocks. In this example, the matrix $A$ is divided into smaller blocks, each containing a specific set of numbers.

\[ A = \begin{bmatrix}
17 & 8 & 51 & 19 & 5 & 14 & 9 & 21 & 23 \\
8 & 51 & 19 & 32 & 5 & 67 & 91 & 14 & 23 \\
73 & 19 & 32 & 5 & 67 & 91 & 14 & 46 & 23 \\
51 & 51 & 5 & 67 & 91 & 14 & 46 & 9 & 23 \\
19 & 19 & 51 & 51 & 51 & 51 & 51 & 51 & 51 \\
32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 \\
5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\
14 & 14 & 14 & 14 & 14 & 14 & 14 & 14 & 14 \\
9 & 9 & 9 & 9 & 9 & 9 & 9 & 9 & 9 \\
21 & 21 & 21 & 21 & 21 & 21 & 21 & 21 & 21 \\
23 & 23 & 23 & 23 & 23 & 23 & 23 & 23 & 23 \\
\end{bmatrix} \]

Each block is highlighted with a dashed line, and the numbers within each block are shown. This representation helps in understanding the structure and properties of the matrix $A$ in a more organized manner.
Block decomposition

\[ A = \begin{bmatrix}
8 & 19 & 5 & 9 \\
17 & 51 & 51 & 9 \\
23 & 17 & 8 & 73 & 51 & 82 & 19 & 32 & 5 & 67 & 91 & 14 & 46 & 9 & 21 & 54
\end{bmatrix} \]
Block decomposition

smallest from each four

A

\[
\begin{array}{cccccccccccc}
23 & 17 & 8 & 73 & 51 & 82 & 19 & 32 & 5 & 67 & 91 & 14 & 46 & 9 & 21 & 54 \\
\hline
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15
\end{array}
\]
Block decomposition

\[
A = \begin{bmatrix}
8 & 17 & 51 & 19 & 5 & 14 & 9 & 21 & 54 \\
19 & 51 & 82 & 19 & 32 & 5 & 67 & 91 & 14 & 46 & 9 & 21 & 54 \\
17 & 8 & 51 & 19 & 5 & 14 & 9 & 21 & 54
\end{bmatrix}
\]

smallest from each four
Block decomposition

<table>
<thead>
<tr>
<th></th>
<th>8</th>
<th>17</th>
<th>51</th>
<th>19</th>
<th>5</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td>8</td>
<td>19</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>19</td>
<td>51</td>
<td>19</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>19</td>
<td>51</td>
<td>19</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>21</td>
<td>21</td>
<td>21</td>
<td>54</td>
<td></td>
</tr>
</tbody>
</table>

\[ A = \begin{bmatrix}
23 & 17 & 8 & 73 & 51 & 82 & 19 & 32 & 5 & 67 & 91 & 14 & 46 & 9 & 21 & 54 \\
\end{bmatrix} \]
Block decomposition
Block decomposition

The diagram illustrates the process of selecting the smallest numbers from each block of eight. The blocks are divided into smaller sections, and the smallest number from each section is highlighted. The diagram includes the matrix A with numbers arranged in a grid, and the process of selecting the smallest numbers is indicated by arrows pointing to the chosen numbers.
Block decomposition

23 17 8 73 51 82 19 32 5 67 91 14 46 9 21 54

\( n \)
Block decomposition
Block decomposition

\[
\begin{bmatrix}
8 & 2 \\
5 & 8 \\
5 & 8 \\
8 & 2 \\
19 & 5 \\
9 & 13 \\
17 & 8 & 51 & 19 & 5 & 14 & 9 & 21 & 54 \\
23 & 17 & 8 & 73 & 19 & 32 & 5 & 67 & 91 & 14 & 46 & 9 & 21 & 54
\end{bmatrix}
\]
Block decomposition

<table>
<thead>
<tr>
<th>A</th>
<th>23</th>
<th>17</th>
<th>873</th>
<th>51</th>
<th>82</th>
<th>19</th>
<th>32</th>
<th>5</th>
<th>67</th>
<th>91</th>
<th>14</th>
<th>46</th>
<th>9</th>
<th>21</th>
<th>54</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2</td>
<td>17</td>
<td>8</td>
<td>51</td>
<td>19</td>
<td>32</td>
<td>5</td>
<td>67</td>
<td>91</td>
<td>14</td>
<td>46</td>
<td>9</td>
<td>21</td>
<td>54</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td>8</td>
<td>2</td>
<td>19</td>
<td>5</td>
<td>8</td>
<td>2</td>
<td>19</td>
<td>5</td>
<td>8</td>
<td>2</td>
<td>19</td>
<td>5</td>
<td>8</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>A8</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>8</td>
<td>2</td>
<td>19</td>
<td>5</td>
<td>8</td>
<td>2</td>
<td>19</td>
<td>5</td>
<td>8</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A16</td>
<td>5</td>
<td>8</td>
<td>5</td>
<td>8</td>
<td>5</td>
<td>8</td>
<td>5</td>
<td>8</td>
<td>5</td>
<td>8</td>
<td>5</td>
<td>8</td>
<td>5</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

n
Block decomposition

$A_k$ is an array of length $\frac{n}{k}$ so that for all $i$: $A_k[i] = (x, v)$

where $v$ is the minimum in $A[i k, (i + 1) k]$ and $x$ is its location in $A$. 

\[
\begin{array}{c|c|c|c|c|c|c|c}
A_{16} & 5 \\
A_8 & 8 & 5 \\
A_4 & 8 & 19 & 5 \\
A_2 & 17 & 8 & 51 & 19 & 5 \\
A & 23 & 17 & 8 & 73 & 51 & 82 & 19 & 32 & 5 & 67 & 91 & 14 & 46 & 9 & 21 & 54 \\
\end{array}
\]
Block decomposition

\( A_k \) is an array of length \( \frac{n}{k} \) so that for all \( i \): \( A_k[i] = (x, v) \)

where \( v \) is the minimum in \( A[ik, (i + 1)k] \) and \( x \) is its location in \( A \).

We store \( A_k \) for all \( k = 1, 2, 4, 8 \ldots \leq n \)
Block decomposition

$A_k$ is an array of length $\frac{n}{k}$ so that for all $i$: $A_k[i] = (x, v)$

where $v$ is the minimum in $A[ik, (i + 1)k]$ and $x$ is its location in $A$.

We store $A_k$ for all $k = 1, 2, 4, 8 \ldots \leq n$

How much space is this?
Block decomposition

\(A_k\) is an array of length \(\frac{n}{k}\) so that for all \(i\): \(A_k[i] = (x, v)\)

where \(v\) is the minimum in \(A[i_k, (i + 1)k]\) and \(x\) is its location in \(A\).

We store \(A_k\) for all \(k = 1, 2, 4, 8 \ldots \leq n\)

How much space is this? \(O(n)\) in total
Block decomposition

$A_k$ is an array of length $\frac{n}{k}$ so that for all $i$: $A_k[i] = (x, v)$

where $v$ is the minimum in $A[i \cdot k, (i + 1)k]$ and $x$ is its location in $A$.

We store $A_k$ for all $k = 1, 2, 4, 8 \ldots \leq n$

How much space is this? $O(n)$ in total

<table>
<thead>
<tr>
<th>$A_{16}$</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_8$</td>
<td>8 5</td>
</tr>
<tr>
<td>$A_4$</td>
<td>8 19 5</td>
</tr>
<tr>
<td>$A_2$</td>
<td>17 8 51 19 5</td>
</tr>
<tr>
<td>$A$</td>
<td>23 17 8 73 51 82 19 32 5 67 91 14 46 9 21 54</td>
</tr>
</tbody>
</table>

$O(n)$ in total
Block decomposition

$A_k$ is an array of length $\frac{n}{k}$ so that for all $i$: $A_k[i] = (x, v)$

where $v$ is the minimum in $A[ik, (i + 1)k]$ and $x$ is its location in $A$.

We store $A_k$ for all $k = 1, 2, 4, 8 \ldots \leq n$

How much space is this? $O(n)$ in total

\[
\begin{array}{|c|c|}
\hline
A_{16} & 5 \\
\hline
A_8 & 8 & 5 \\
\hline
A_4 & 8 & 19 & 5 & 9 \\
\hline
A_2 & 17 & 8 & 51 & 19 & 14 & 9 & 21 \\
\hline
A & 23 & 17 & 8 & 73 & 51 & 82 & 19 & 32 & 5 & 67 & 91 & 14 & 46 & 9 & 21 & 54 \\
\hline
\end{array}
\]

$+$ \frac{n}{16}$ + \frac{n}{8}$ $+$ \frac{n}{4}$ $+$ \frac{n}{2}$ $O(n)$
**Block decomposition**

$A_k$ is an array of length $\frac{n}{k}$ so that for all $i$: $A_k[i] = (x, v)$

where $v$ is the minimum in $A[ik, (i + 1)k]$ and $x$ is its location in $A$.

We store $A_k$ for all $k = 1, 2, 4, 8 \ldots \leq n$

How much space is this? $O(n)$ in total

<table>
<thead>
<tr>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_4$</th>
<th>$A_8$</th>
<th>$A_{16}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>8</td>
<td>73</td>
<td>51</td>
</tr>
<tr>
<td>17</td>
<td>8</td>
<td>51</td>
<td>82</td>
<td>19</td>
</tr>
<tr>
<td>8</td>
<td>73</td>
<td>51</td>
<td>82</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>67</td>
<td>91</td>
<td>14</td>
<td>46</td>
</tr>
<tr>
<td>67</td>
<td>91</td>
<td>14</td>
<td>46</td>
<td>9</td>
</tr>
<tr>
<td>91</td>
<td>14</td>
<td>46</td>
<td>9</td>
<td>21</td>
</tr>
<tr>
<td>14</td>
<td>46</td>
<td>9</td>
<td>21</td>
<td>54</td>
</tr>
</tbody>
</table>

$n$ is the length of the array.
Block decomposition

\( A_k \) is an array of length \( \frac{n}{k} \) so that for all \( i \): \( A_k[i] = (x, v) \)

where \( v \) is the minimum in \( A[ik, (i + 1)k] \) and \( x \) is its location in \( A \).

We store \( A_k \) for all \( k = 1, 2, 4, 8 \ldots \leq n \)

How much space is this? \( O(n) \) in total

How quickly can we build them?

\[ \begin{array}{cccc}
A_{16} & & & 5 \\
& 8 & & 5 \\
A_{8} & & & 9 \\
& 8 & & 19 \\
A_{4} & & & 5 \\
& 17 & & 19 \\
A_{2} & & & 14 \\
& 23 & & 5 \\
A & & & 16 \\
& 23 & & 54 \\
\end{array} \]
Block decomposition

$A_k$ is an array of length $\frac{n}{k}$ so that for all $i$: $A_k[i] = (x, v)$

where $v$ is the minimum in $A[ik, (i + 1)k]$ and $x$ is its location in $A$.

We store $A_k$ for all $k = 1, 2, 4, 8 \ldots \leq n$

How much space is this? $O(n)$ in total

How quickly can we build them?

construct the $A_k$ arrays bottom-up
Block decomposition

$A_k$ is an array of length $\frac{n}{k}$ so that for all $i$: $A_k[i] = (x, v)$
where $v$ is the minimum in $A[ik, (i + 1)k]$ and $x$ is its location in $A$.

We store $A_k$ for all $k = 1, 2, 4, 8 \ldots \leq n$

How much space is this? $O(n)$ in total

How quickly can we build them?

The construction of $A_k$ arrays can be done bottom-up:

- **$A_{16}$**
- **$A_8$**
- **$A_4$**
- **$A_2$**
- **$A$**

To compute $A_k$ from $A_{\frac{n}{k}}$, these can be done in $O(1)$ time.
Block decomposition

\( A_k \) is an array of length \( \frac{n}{k} \) so that for all \( i: A_k[i] = (x, v) \)

where \( v \) is the minimum in \( A[ik, (i + 1)k] \) and \( x \) is its location in \( A \).

We store \( A_k \) for all \( k = 1, 2, 4, 8 \ldots \leq n \)

How much space is this? \( O(n) \) in total

How quickly can we build them? \( O(n) \) preprocessing time

\[
\begin{array}{|c|c|c|}
\hline
A_{16} & & 5 \\
\hline
A_8 & 8 & 5 \\
\hline
A_4 & 8 & 19 & 5 \\
\hline
A_2 & 17 & 8 & 51 & 19 & 5 \\
\hline
A & 23 & 17 & 8 & 73 & 51 & 82 & 19 & 32 & 5 & 67 & 91 & 14 & 46 & 9 & 21 & 54 \\
\hline
\end{array}
\]

construct the \( A_k \)
arrays bottom-up

compute this from
these in \( O(1) \) time
### Block decomposition

$A_k$ is an array of length $\frac{n}{k}$ so that for all $i$: $A_k[i] = (x, v)$

where $v$ is the minimum in $A[ik, (i + 1)k]$ and $x$ is its location in $A$.

We store $A_k$ for all $k = 1, 2, 4, 8 \ldots \leq n$

**How much space is this?** $O(n)$ in total

**How quickly can we build them?** $O(n)$ preprocessing time

<table>
<thead>
<tr>
<th>$A$</th>
<th>23</th>
<th>17</th>
<th>8</th>
<th>73</th>
<th>51</th>
<th>82</th>
<th>19</th>
<th>32</th>
<th>5</th>
<th>67</th>
<th>91</th>
<th>14</th>
<th>46</th>
<th>9</th>
<th>21</th>
<th>54</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_2$</td>
<td>17</td>
<td>8</td>
<td>51</td>
<td>19</td>
<td>19</td>
<td>5</td>
<td>14</td>
<td>9</td>
<td>21</td>
<td>14</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>14</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>$A_4$</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>19</td>
<td>19</td>
<td>5</td>
<td>5</td>
<td>9</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>$A_8$</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>5</td>
<td>5</td>
<td>9</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>$A_{16}$</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>5</td>
<td>5</td>
<td>9</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td></td>
</tr>
</tbody>
</table>

$A_k$ for all $k = 1, 2, 4, 8 \ldots \leq n$
Block decomposition

\[ A_{16} \]

\[ A_8 \]

\[ A_4 \]

\[ A_2 \]

\[ A \]

\[ \begin{array}{cccccccccccc}
23 & 17 & 8 & 73 & 51 & 82 & 19 & 32 & 5 & 67 & 91 & 14 & 46 & 9 & 21 & 54 \\
\end{array} \]
How do we find RMQ(i,j)?

Block decomposition
How do we find $\text{RMQ}(i,j)$?

Block decomposition
How do we find $\text{RMQ}(i,j)$?

Find the largest block which is completely contained within the query interval $\text{RMQ}(1,9)$. 

```
\[
\begin{array}{cccccccccccc}
A_{16} & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad \\
A_{12} & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad \\
A_{8} & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad \\
A_{4} & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad \\
A_{2} & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad \\
\end{array}
\]
How do we find \( \text{RMQ}(i,j) \)?

Find the largest block which is completely contained within the query interval.
How do we find **RMQ(i,j)**?

**Repeat:** Find the largest *block* which

is completely contained within the query interval

*_but doesn’t overlap a block you chose before_*

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{16}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_{8}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_{4}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_{2}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A$</td>
<td>23</td>
<td>17</td>
<td>8</td>
<td>73</td>
<td>51</td>
<td>82</td>
</tr>
</tbody>
</table>

**RMQ(1,9)**

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$n$
How do we find $\text{RMQ}(i,j)$?

**Repeat:** Find the largest block which is completely contained within the query interval but doesn’t overlap a block you chose before.

$\text{RMQ}(1,9)$

<table>
<thead>
<tr>
<th></th>
<th>$A_{16}$</th>
<th></th>
<th>$A_{8}$</th>
<th></th>
<th>$A_{4}$</th>
<th></th>
<th>$A_{2}$</th>
<th></th>
<th>$A$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td></td>
<td>5</td>
<td></td>
<td>5</td>
<td></td>
<td>5</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>$A_{16}$</td>
<td>8</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>17</td>
<td>1</td>
<td>23</td>
</tr>
<tr>
<td>$A_{8}$</td>
<td>51</td>
<td>4</td>
<td>51</td>
<td>4</td>
<td>51</td>
<td>4</td>
<td>51</td>
<td>4</td>
<td>51</td>
</tr>
<tr>
<td>$A_{4}$</td>
<td>19</td>
<td>6</td>
<td>19</td>
<td>6</td>
<td>19</td>
<td>6</td>
<td>19</td>
<td>6</td>
<td>19</td>
</tr>
<tr>
<td>$A_{2}$</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>$A$</td>
<td>23</td>
<td>8</td>
<td>73</td>
<td>19</td>
<td>82</td>
<td>19</td>
<td>32</td>
<td>5</td>
<td>67</td>
</tr>
</tbody>
</table>

$\text{n}$
How do we find $\text{RMQ}(i,j)$?

**Repeat:** Find the largest block which is completely contained within the query interval but doesn’t overlap a block you chose before

*(break ties arbitrarily)*
How do we find $\text{RMQ}(i,j)$?

**Repeat:** Find the largest block which is completely contained within the query interval but doesn’t overlap a block you chose before.
How do we find $\text{RMQ}(i,j)$?

**Repeat:** Find the largest *block* which is completely contained within the query interval  

*but doesn’t overlap a block you chose before*
How do we find $\text{RMQ}(i,j)$?

**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 \( \text{RMQ}(i,j) \)?

**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 many blocks do we pick?
How do we find $\text{RMQ}(i,j)$?

**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 $\text{RMQ}(i,j)$?

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 many blocks do we pick?

<table>
<thead>
<tr>
<th>A16</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A8</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A4</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A2</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>23</th>
<th>17</th>
<th>8</th>
<th>73</th>
<th>51</th>
<th>82</th>
<th>19</th>
<th>32</th>
<th>5</th>
<th>67</th>
<th>91</th>
<th>14</th>
<th>46</th>
<th>9</th>
<th>21</th>
<th>54</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>
How do we find $\text{RMQ}(i,j)$?

**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 many blocks do we pick?
How do we find $\text{RMQ}(i,j)$?

**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 $\text{RMQ}(i,j)$?

**Repeat:** Find the largest \textit{block} which

is completely contained within the query interval

\textit{but doesn’t overlap a block you chose before}

The minimum is the smallest in all these blocks \textit{because they cover the query}

---

**Diagram:**

- **$A_{16}$**
- **$A_{8}$**
- **$A_{4}$**
- **$A_{2}$**
- **$A$**

The diagram shows a grid with numbers and borders indicating the blocks. Each block is labeled with a number, and the minimum value within the query interval is highlighted.
How do we find RMQ(i,j)?

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)?

**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*

---

**Block decomposition**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A_2</th>
<th>A_4</th>
<th>A_8</th>
<th>A_16</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23</td>
<td>17</td>
<td>8</td>
<td>73</td>
<td>51</td>
</tr>
<tr>
<td>2</td>
<td>17</td>
<td>8</td>
<td>51</td>
<td>19</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>51</td>
<td>19</td>
<td>5</td>
<td>67</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>8</td>
<td>5</td>
<td>14</td>
<td>91</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>51</td>
<td>19</td>
<td>5</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>51</td>
<td>19</td>
<td>5</td>
<td>14</td>
<td>91</td>
</tr>
<tr>
<td>7</td>
<td>19</td>
<td>5</td>
<td>14</td>
<td>9</td>
<td>46</td>
</tr>
<tr>
<td>8</td>
<td>51</td>
<td>67</td>
<td>91</td>
<td>9</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>19</td>
<td>67</td>
<td>91</td>
<td>21</td>
<td>54</td>
</tr>
</tbody>
</table>

---

**How many blocks do we pick?**

at most 2 blocks of each size
Block decomposition

How do we find $\text{RMQ}(i,j)$?

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 many blocks do we pick?

at most 2 blocks of each size
How do we find $\text{RMQ}(i,j)$?

**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*

---

**Diagram:**

- $A_{16}$
- $A_8$
- $A_4$
- $A_2$
- $A$

**Legend:**

- $n$ denotes the size of the block.
- Each block is labeled with its minimum value.

**How many blocks do we pick?**

- at most 2 blocks of each size

---
How do we find $\text{RMQ}(i,j)$?

**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*

---

<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_{16}$</td>
<td></td>
<td></td>
<td>$A_{8}$</td>
<td></td>
<td></td>
<td>$A_{4}$</td>
<td></td>
<td></td>
<td>$A_{2}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

How many blocks do we pick?

at most 2 blocks of each size
How do we find $RMQ(i,j)$?

**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.

10,000 foot view
- $A_{16}$
- $A_{8}$
- $A_{4}$
- $A_{2}$
- $A$

No valleys
- $A_{16}$
- $A_{8}$

No plateaus
- $A_{4}$
- $A_{2}$
- $A$

How many blocks do we pick? at most 2 blocks of each size

<table>
<thead>
<tr>
<th>32</th>
<th>5</th>
<th>67</th>
<th>91</th>
<th>14</th>
<th>46</th>
<th>9</th>
<th>21</th>
<th>54</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>
How do we find RMQ(i,j)?

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 many blocks do we pick? at most 2 blocks of each size.
How do we find $\text{RMQ}(i,j)$?

**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*

<table>
<thead>
<tr>
<th></th>
<th>$\text{RMQ}(1,9)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$</td>
<td>23 17 8 73 51 82 19 32 5 67 91 14 46 9 21 54</td>
</tr>
<tr>
<td>$A_2$</td>
<td>17 8 51 19 5 14</td>
</tr>
<tr>
<td>$A_4$</td>
<td>8 19 5</td>
</tr>
<tr>
<td>$A_8$</td>
<td>8</td>
</tr>
<tr>
<td>$A_{16}$</td>
<td>A</td>
</tr>
</tbody>
</table>
Block decomposition

How do we find $\text{RMQ}(i,j)$?

**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*

---

<table>
<thead>
<tr>
<th>$A_{16}$</th>
<th></th>
<th></th>
<th>$A_{16}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_{8}$</td>
<td>8</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_{4}$</td>
<td>8</td>
<td>19</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A_{2}$</td>
<td>17</td>
<td>8</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A$</td>
<td>23</td>
<td>17</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

How many blocks do we pick?

*at most 2 blocks of each size*

There are $O(\log n)$ sizes
Block decomposition

How do we find $RMQ(i,j)$?

**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 many blocks do we pick?
at most 2 blocks of each size

There are $O(\log n)$ sizes

Picking the next block
takes $O(1)$ time
Block decomposition

How do we find $\text{RMQ}(i,j)$?

**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 many blocks do we pick?

at most 2 blocks of each size

There are $O(\log n)$ sizes

Picking the next block takes $O(1)$ time

So we have … $O(n)$ space,

$O(n)$ prep time

$O(\log n)$ query time
More space, faster queries

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$.
More space, faster queries

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

The array $R_2$ stores $RMQ(i, i + 1)$ for all $i$
More space, faster queries

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

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$. 

\[
\begin{array}{c}
A \\
\hline
2 \\
\end{array}
\]
More space, faster queries

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

The array $A$ stores $\text{RMQ}(i, i + 1)$ for all $i$

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$
More space, faster queries

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$
More space, faster queries

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$. 
More space, faster queries

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

![Diagram](image)

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$. 
More space, faster queries

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$. 

![Diagram of array A with interval of length 2]
More space, faster queries

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

\[ A \]

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$
**Key Idea** precompute the answers for every interval of length 2, 4, 8, 16 . . .

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$. 

More space, faster queries
More space, faster queries

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$
More space, faster queries

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$
More space, faster queries

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$
More space, faster queries

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$

$R_4$ stores $\text{RMQ}(i, i + 3)$ for all $i$
More space, faster queries

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$

$R_4$ stores $\text{RMQ}(i, i + 3)$ for all $i$
More space, faster queries

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$

$R_4$ stores $\text{RMQ}(i, i + 3)$ for all $i$

$R_8$ stores $\text{RMQ}(i, i + 7)$ for all $i$
More space, faster queries

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$

$R_4$ stores $\text{RMQ}(i, i + 3)$ for all $i$

$R_8$ stores $\text{RMQ}(i, i + 7)$ for all $i$
More space, faster queries

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$

$R_4$ stores $\text{RMQ}(i, i + 3)$ for all $i$

$R_8$ stores $\text{RMQ}(i, i + 7)$ for all $i$

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$
More space, faster queries

**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 $i$

$R_4$ stores RMQ$(i, i + 3)$ for all $i$

$R_8$ stores RMQ$(i, i + 7)$ for all $i$

$R_k$ stores RMQ$(i, i + k - 1)$ for all $i$

We build $R_k$ for $k = 2, 4, 8, 16 . . . \leq n$
More space, faster queries

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$

$R_4$ stores $\text{RMQ}(i, i + 3)$ for all $i$

$R_8$ stores $\text{RMQ}(i, i + 7)$ for all $i$

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$

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

each of the $O(\log n)$ arrays uses $O(n)$ space
More space, faster queries

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$

$R_4$ stores $\text{RMQ}(i, i + 3)$ for all $i$

$R_8$ stores $\text{RMQ}(i, i + 7)$ for all $i$

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$

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

each of the $O(\log n)$ arrays uses $O(n)$ space

so $O(n \log n)$ total space
More space, faster queries

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$

$R_4$ stores $\text{RMQ}(i, i + 3)$ for all $i$

$R_8$ stores $\text{RMQ}(i, i + 7)$ for all $i$

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$

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

each of the $O(\log n)$ arrays uses $O(n)$ space

so $O(n \log n)$ total space
**More space, faster queries**

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$

$R_4$ stores $\text{RMQ}(i, i + 3)$ for all $i$

$R_8$ stores $\text{RMQ}(i, i + 7)$ for all $i$

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$

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

We build $R_k$ from $R_k$ in $O(n)$ time

We build $R_{2k}$ for $k = 2, 4, 8, 16 \ldots \leq n$

each of the $O(\log n)$ arrays uses $O(n)$ space

so $O(n \log n)$ total space
More space, faster queries

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$

$R_4$ stores $\text{RMQ}(i, i + 3)$ for all $i$

$R_8$ stores $\text{RMQ}(i, i + 7)$ for all $i$

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$

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

We build $R_{2k}$ from $R_k$ in $O(n)$ time

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

each of the $O(\log n)$ arrays uses $O(n)$ space

so $O(n \log n)$ total space
More space, faster queries

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$

$R_4$ stores $\text{RMQ}(i, i + 3)$ for all $i$

$R_8$ stores $\text{RMQ}(i, i + 7)$ for all $i$

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$

We build $R_{2k}$ from $R_k$ in $O(n)$ time

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

each of the $O(\log n)$ arrays uses $O(n)$ space

so $O(n \log n)$ total space

We build $R_2$ from $A$ in $O(n)$ time
More space, faster queries

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

![Diagram showing intervals](image)

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$

$R_4$ stores $\text{RMQ}(i, i + 3)$ for all $i$

$R_8$ stores $\text{RMQ}(i, i + 7)$ for all $i$

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$

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

We build $R_{2k}$ from $R_k$ in $O(n)$ time

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

each of the $O(\log n)$ arrays uses $O(n)$ space

so $O(n \log n)$ total space
More space, faster queries

Key Idea: precompute the answers for every interval of length $2, 4, 8, 16 \ldots$

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$
$R_4$ stores $\text{RMQ}(i, i + 3)$ for all $i$
$R_8$ stores $\text{RMQ}(i, i + 7)$ for all $i$

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$

We build $R_2$ from $A$ in $O(n)$ time
We build $R_{2k}$ from $R_k$ in $O(n)$ time

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

each of the $O(\log n)$ arrays uses $O(n)$ space

so $O(n \log n)$ total space
More space, faster queries

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$

$R_4$ stores $\text{RMQ}(i, i + 3)$ for all $i$

$R_8$ stores $\text{RMQ}(i, i + 7)$ for all $i$

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$

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

each of the $O(\log n)$ arrays uses $O(n)$ space

so $O(n \log n)$ total space

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

We build $R_{2k}$ from $R_k$ in $O(n)$ time
More space, faster queries

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$

$R_4$ stores $\text{RMQ}(i, i + 3)$ for all $i$

$R_8$ stores $\text{RMQ}(i, i + 7)$ for all $i$

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$

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

We build $R_{2k}$ from $R_k$ in $O(n)$ time

We build $R_k$ for $k = 2, 4, 8, 16 . . . \leq n$

each of the $O(\log n)$ arrays uses $O(n)$ space

so $O(n \log n)$ total space
More space, faster queries

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$

$R_4$ stores $\text{RMQ}(i, i + 3)$ for all $i$

$R_8$ stores $\text{RMQ}(i, i + 7)$ for all $i$

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$

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

We build $R_{2k}$ from $R_k$ in $O(n)$ time

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

each of the $O(\log n)$ arrays uses $O(n)$ space

so $O(n \log n)$ total space
**More space, faster queries**

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

The array $R_2$ stores $\text{RMQ}(i, i + 1)$ for all $i$

$R_4$ stores $\text{RMQ}(i, i + 3)$ for all $i$

$R_8$ stores $\text{RMQ}(i, i + 7)$ for all $i$

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$

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

each of the $O(\log n)$ arrays uses $O(n)$ space

so $O(n \log n)$ total space

This takes $O(n \log n)$ prep time
More space, faster queries

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$.

we build $R_k$ for $k = 2, 4, 8, 16 \ldots \leq n$

How do we compute $\text{RMQ}(i, j)$?

If the interval length, $\ell = (j - i + 1)$, is a power-of-two - just look up the answer
More space, faster queries

\( R_k \) stores \( \text{RMQ}(i, i + k - 1) \) for all \( i \),
we build \( R_k \) for \( k = 2, 4, 8, 16 \ldots \leq n \)

\[
\begin{align*}
A & \quad \text{stored in } R_2 \\
& \quad \text{stored in } R_4 \\
& \quad \text{stored in } R_8 \\
\end{align*}
\]

How do we compute \( \text{RMQ}(i, j) \)?

If the interval length, \( \ell = (j - i + 1) \), is a power-of-two - just look up the answer
More space, faster queries

\(R_k\) stores \(\text{RMQ}(i, i + k - 1)\) for all \(i\),

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

How do we compute \(\text{RMQ}(i, j)\)?

If the interval length, \(\ell = (j - i + 1)\), is a power-of-two - just look up the answer

these queries take \(O(1)\) time
More space, faster queries

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$.

we build $R_k$ for $k = 2, 4, 8, 16 \ldots \leq n$

$A$

stored in $R_2$

stored in $R_4$

stored in $R_8$

How do we compute $\text{RMQ}(i, j)$?

If the interval length, $\ell = (j - i + 1)$, is a power-of-two - just look up the answer

these queries take $O(1)$ time

Otherwise, find the $k = 2, 4, 8, 16 \ldots$ such that $k \leq \ell < 2k$
More space, faster queries

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$, we build $R_k$ for $k = 2, 4, 8, 16 \ldots \leq n$

How do we compute $\text{RMQ}(i, j)$?

If the interval length, $\ell = (j - i + 1)$, is a power-of-two - just look up the answer

these queries take $O(1)$ time

Otherwise, find the $k = 2, 4, 8, 16 \ldots$ such that $k \leq \ell < 2k$
More space, faster queries

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$.

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

How do we compute $\text{RMQ}(i, j)$?

If the interval length, $\ell = (j - i + 1)$, is a power-of-two - just look up the answer

these queries take $O(1)$ time

Otherwise, find the $k = 2, 4, 8, 16 \ldots$ such that $k \leq \ell < 2k$

Compute the minimum of $\text{RMQ}(i, i + k - 1)$ and $\text{RMQ}(j - k + 1, j)$
More space, faster queries

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$,
we build $R_k$ for $k = 2, 4, 8, 16 \ldots \leq n$

How do we compute $\text{RMQ}(i, j)$?

If the interval length, $\ell = (j - i + 1)$, is a power-of-two - just look up the answer
these queries take $O(1)$ time

Otherwise, find the $k = 2, 4, 8, 16 \ldots$ such that $k \leq \ell < 2k$

Compute the minimum of $\text{RMQ}(i, i + k - 1)$ and $\text{RMQ}(j - k + 1, j)$
(these two queries take $O(1)$ time)
More space, faster queries

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$.
we build $R_k$ for $k = 2, 4, 8, 16 \ldots \leq n$

How do we compute $\text{RMQ}(i, j)$?

If the interval length, $\ell = (j - i + 1)$, is a power-of-two - just look up the answer
these queries take $O(1)$ time

Otherwise, find the $k = 2, 4, 8, 16 \ldots$ such that $k \leq \ell < 2k$

Compute the minimum of $\text{RMQ}(i, i + k - 1)$ and $\text{RMQ}(j - k + 1, j)$
(these two queries take $O(1)$ time)
More space, faster queries

\( R_k \) stores \( \text{RMQ}(i, i + k - 1) \) for all \( i \),
we build \( R_k \) for \( k = 2, 4, 8, 16 \ldots \leq n \)

How do we compute \( \text{RMQ}(i, j) \)?

If the interval length, \( \ell = (j - i + 1) \), is a power-of-two - just look up the answer
these queries take \( O(1) \) time

Otherwise, find the \( k = 2, 4, 8, 16 \ldots \) such that \( k \leq \ell < 2k \)

Compute the minimum of \( \text{RMQ}(i, i + k - 1) \) and \( \text{RMQ}(j - k + 1, j) \)
(these two queries take \( O(1) \) time)

This takes \( O(1) \) time but why does it work?
More space, faster queries

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$,

we build $R_k$ for $k = 2, 4, 8, 16 \ldots \leq n$

How do we compute $\text{RMQ}(i, j)$?

If the interval length, $\ell = (j - i + 1)$, is a power-of-two - just look up the answer

these queries take $O(1)$ time

Otherwise, find the $k = 2, 4, 8, 16 \ldots$ such that $k \leq \ell < 2k$

Compute the minimum of $\text{RMQ}(i, i + k - 1)$ and $\text{RMQ}(j - k + 1, j)$

(these two queries take $O(1)$ time)

This takes $O(1)$ time but why does it work?
More space, faster queries

\( R_k \) stores \( \text{RMQ}(i, i + k - 1) \) for all \( i \),
we build \( R_k \) for \( k = 2, 4, 8, 16 \ldots \leq n \)

How do we compute \( \text{RMQ}(i, j) \)?

If the interval length, \( \ell = (j - i + 1) \), is a power-of-two - just look up the answer
these queries take \( O(1) \) time

Otherwise, find the \( k = 2, 4, 8, 16 \ldots \) such that \( k \leq \ell < 2k \)

Compute the minimum of \( \text{RMQ}(i, i + k - 1) \) and \( \text{RMQ}(j - k + 1, j) \)
(these two queries take \( O(1) \) time)

This takes \( O(1) \) time but why does it work?
More space, faster queries

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$, we build $R_k$ for $k = 2, 4, 8, 16 \ldots \leq n$

A

$\text{RMQ}(i, j)$?

If the interval length, $\ell = (j - i + 1)$, is a power-of-two - just look up the answer

these queries take $O(1)$ time

Otherwise, find the $k = 2, 4, 8, 16 \ldots$ such that $k \leq \ell < 2k$

Compute the minimum of $\text{RMQ}(i, i + k - 1)$ and $\text{RMQ}(j - k + 1, j)$

(these two queries take $O(1)$ time)

This takes $O(1)$ time but why does it work?
More space, faster queries

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$.

we build $R_k$ for $k = 2, 4, 8, 16 \ldots \leq n$

How do we compute $\text{RMQ}(i, j)$?

If the interval length, $\ell = (j - i + 1)$, is a power-of-two - just look up the answer

these queries take $O(1)$ time

Otherwise, find the $k = 2, 4, 8, 16 \ldots$ such that $k \leq \ell < 2k$

Compute the minimum of $\text{RMQ}(i, i + k - 1)$ and $\text{RMQ}(j - k + 1, j)$

(these two queries take $O(1)$ time)

This takes $O(1)$ time but why does it work?
More space, faster queries

$R_k$ stores $\text{RMQ}(i, i + k - 1)$ for all $i$, we build $R_k$ for $k = 2, 4, 8, 16 \ldots \leq n$

How do we compute $\text{RMQ}(i, j)$?

If the interval length, $\ell = (j - i + 1)$, is a power-of-two - just look up the answer

these queries take $O(1)$ time

Otherwise, find the $k = 2, 4, 8, 16 \ldots$ such that $k \leq \ell < 2k$

Compute the minimum of $\text{RMQ}(i, i + k - 1)$ and $\text{RMQ}(j - k + 1, j)$

(these two queries take $O(1)$ time)

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...

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

the output is the location of the smallest element in $A[i, j]$
Range minimum query (intermediate) summary

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 \( A[i, j] \)

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
Range minimum query (intermediate) summary

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

![Array A with indices](image)

$$A = 23, 17, 8, 73, 51, 82, 19, 32, 5, 67, 91, 14, 46, 9, 21, 54$$

$i = 3$, $j = 7$

$\text{RMQ}(3, 7) = 6$

After preprocessing, a **range minimum query** is given by $\text{RMQ}(i, j)$

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

---

**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

---

Can we do better?
Range minimum query (intermediate) summary

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 $A[i, j]$

**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

Can we do better? (yes)
Low-resolution RMQ

**Key Idea** replace $A$ with a smaller, *low resolution* array $H$
Low-resolution RMQ

**Key Idea** replace $A$ with a smaller, ‘low resolution’ array $H$
Low-resolution RMQ

**Key Idea** replace $A$ with a smaller, *low resolution* array $H$

\[
\tilde{n} = \frac{n}{\log n}
\]
Low-resolution RMQ

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

\[ \tilde{n} = \frac{n}{\log n} \]
Low-resolution RMQ

Key Idea replace $A$ with a smaller, ‘low resolution’ array $H$

$$
\tilde{n} = \frac{n}{\log n}
$$

Diagram:
- $A$ is the original array of length $n$.
- $H$ is the low-resolution array of length $\tilde{n} = \frac{n}{\log n}$.
- The smallest of the elements in $A$ is stored in $H$.
Low-resolution RMQ

**Key Idea** replace $A$ with a smaller, ‘low resolution’ array $H$

\[
\tilde{n} = \frac{n}{\log n}
\]
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’

\[
\tilde{n} = \frac{n}{\log n}
\]
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’

$\tilde{n} = \frac{n}{\log n}$
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’

$$\tilde{n} = \frac{n}{\log n}$$
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’

$$\tilde{n} = \frac{n}{\log n}$$
**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*

![Diagram of arrays](image)

Preprocess the array $H$ (which has length $\tilde{n} = \frac{n}{\log n}$) to answer RMQs...
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’

Preprocess the array $H$ (which has length $\tilde{n} = \frac{n}{\log n}$) to answer RMQs…

using Solution 2
**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’

![Diagram showing arrays $A$ and $H$ with subarrays $L_0, L_1, L_2, \ldots, L_{\tilde{n}}$.]

Preprocess the array $H$ (which has length $\tilde{n} = \frac{n}{\log n}$) to answer RMQs…

*Recall...*

**Solution 2 on $A$**

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

\[\tilde{n} = \frac{n}{\log n}\]
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’

Preprocess the array $H$ (which has length $\tilde{n} = \frac{n}{\log n}$) to answer RMQs…

Recall…

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

using **Solution 2**
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’

Preprocess the array $H$ (which has length $\tilde{n} = \frac{n}{\log n}$) to answer RMQs...

Recall...

<table>
<thead>
<tr>
<th>Solution 2 on $A$</th>
<th>Solution 2 on $H$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n \log n)$ space</td>
<td>$O(\tilde{n} \log \tilde{n})$ space</td>
</tr>
<tr>
<td>$O(n \log n)$ prep time</td>
<td>$O(\tilde{n} \log \tilde{n})$ prep time</td>
</tr>
<tr>
<td>$O(1)$ query time</td>
<td>$O(1)$ query time</td>
</tr>
</tbody>
</table>
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’

![Diagram of arrays $A$ and $H$]

Preprocess the array $H$ (which has length $\tilde{n} = \frac{n}{\log n}$) to answer RMQs...

**Recall**...

Solution 2 on $A$

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

Solution 2 on $H$

- $O(\tilde{n} \log \tilde{n})$ space $= O\left(\left(\frac{n}{\log n}\right) \log \left(\frac{n}{\log n}\right)\right)$
- $O(\tilde{n} \log \tilde{n})$ prep time
- $O(1)$ query time
Low-resolution RMQ

Key Idea replace $A$ with a smaller, ‘low resolution’ array $H$

and many small arrays $L_0, L_1, L_2$ . . . ‘for the details’

Preprocess the array $H$ (which has length $\tilde{n} = \frac{n}{\log n}$) to answer RMQs . . .

Recall . . .

Solution 2 on $A$

$O(n \log n)$ space

$O(n \log n)$ prep time

$O(1)$ query time

Solution 2 on $H$

$O(\tilde{n} \log \tilde{n})$ prep time

$O(\tilde{n} \log \tilde{n})$ query 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’

Preprocess the array \( H \) (which has length \( \tilde{n} = \frac{n}{\log n} \)) to answer RMQs…

**Recall…**

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

<table>
<thead>
<tr>
<th>Solution 2 on ( H )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( O(\tilde{n} \log \tilde{n}) ) space = ( O\left(\left(\frac{n}{\log n}\right) \log \left(\frac{n}{\log n}\right)\right) ) = ( O(n) )</td>
</tr>
<tr>
<td>( O(\tilde{n} \log \tilde{n}) ) prep time = ( O(n) )</td>
</tr>
<tr>
<td>( O(1) ) query time</td>
</tr>
</tbody>
</table>
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’

Preprocess the array $H$ (which has length $\tilde{n} = \frac{n}{\log n}$) to answer RMQs ...

Recall ...

Solution 2 on $A$

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

Solution 2 on $H$

$O(\tilde{n} \log \tilde{n})$ space $= O \left( \left( \frac{n}{\log n} \right) \log \left( \frac{n}{\log n} \right) \right) = O(n)$

$O(\tilde{n} \log \tilde{n})$ prep time $= O(n)$

$O(1)$ 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’

Preprocess the array $H$ (which has length $\tilde{n} = \frac{n}{\log n}$) to answer RMQs…

using **Solution 2** in $O(n)$ space/prep 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’

\[ \tilde{n} = \frac{n}{\log n} \]

Preprocess the array $H$ (which has length $\tilde{n} = \frac{n}{\log n}$) to answer RMQs…

using **Solution 2** in $O(n)$ space/prep time

Preprocess each array $L_i$ (which has length $\log n$) to answer RMQs…

using **Solution 2**
**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’

Preprocess the array $H$ (which has length $\tilde{n} = \frac{n}{\log n}$) to answer RMQs...

using **Solution 2** in $O(n)$ space/prep time

Preprocess each array $L_i$ (which has length $\log n$) to answer RMQs...

using **Solution 2**

**Solution 2 on $L_i$**

$O((\log n \log \log n))$ space/prep time  \hspace{1cm} $O(1)$ 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’

![Diagram showing arrays $A$, $L_0$, $L_1$, $L_2$, $L_3$, $L_4$, $L_5$, $L_{\tilde{n}}$ and $H$]

Preprocess the array $H$ (which has length $\tilde{n} = \frac{n}{\log n}$) to answer RMQs...

using **Solution 2** in $O(n)$ space/prep time

Preprocess each array $L_i$ (which has length $\log n$) to answer RMQs...

using **Solution 2** in $O(\log n \log \log n)$ space/prep time

**Solution 2 on** $L_i$

$O((\log n) \log \log n))$ space/prep time \quad $O(1)$ 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’

$\tilde{n} = \frac{n}{\log n}$

Preprocess the array $H$ (which has length $\tilde{n} = \frac{n}{\log n}$) to answer RMQs…

using **Solution 2** in $O(n)$ space/prep time

Preprocess each array $L_i$ (which has length $\log n$) to answer RMQs…

using **Solution 2** in $O(\log n \log \log n)$ space/prep time

**Total space** = $O(n) + O(\tilde{n} \log n \log \log n)$
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'

Preprocess the array $H$ (which has length $\tilde{n} = \frac{n}{\log n}$) to answer RMQs... using **Solution 2** in $O(n)$ space/prep time

Preprocess each array $L_i$ (which has length $\log n$) to answer RMQs... using **Solution 2** in $O(\log n \log \log n)$ space/prep time

**Total space** = $O(n) + O(\tilde{n} \log n \log \log n)$

space for RMQ structure for $H$  

space for RMQ structures for all the $L_i$ arrays
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’*

Preprocess the array $H$ (which has length $\tilde{n} = \frac{n}{\log n}$) to answer RMQs...

using **Solution 2** in $O(n)$ space/prep time

Preprocess each array $L_i$ (which has length $\log n$) to answer RMQs...

using **Solution 2** in $O(\log n \log \log n)$ space/prep time

**Total space** = $O(n) + O(\tilde{n} \log n \log \log n) = O(n \log \log n)$

space for RMQ structure for $H$ 

space for RMQ structures for all the $L_i$ arrays
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’

\[
\hat{n} = \frac{n}{\log n}
\]

Preprocess the array \( H \) (which has length \( \hat{n} = \frac{n}{\log n} \)) to answer RMQs…
using **Solution 2** in \( O(n) \) space/prep time

Preprocess each array \( L_i \) (which has length \( \log n \)) to answer RMQs…
using **Solution 2** in \( O(\log n \log \log n) \) space/prep time

**Total space** = \( O(n) + O(\hat{n} \log n \log \log n) = O(n \log \log n) \)
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’

Preprocess the array $H$ (which has length $\tilde{n} = \frac{n}{\log n}$) to answer RMQs...

using **Solution 2** in $O(n)$ space/prep time

Preprocess each array $L_i$ (which has length $\log n$) to answer RMQs...

using **Solution 2** in $O(\log n \log \log n)$ space/prep time

**Total space** $= O(n) + O(\tilde{n} \log n \log \log n) = O(n \log \log n)$

**Total prep. time** $= O(n \log \log n)$
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’

\[
\tilde{n} = \frac{n}{\log n}
\]
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’

\[ \tilde{n} = \frac{n}{\log n} \]

How do we answer a query in $A$?
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’

$$\tilde{n} = \frac{n}{\log n}$$

How do we answer a query in $A$?
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’

$$\tilde{n} = \frac{n}{\log n}$$

How do we answer a query in $A$?

Do at most one query in $H \ldots$
and one query in at most two different $L_i$
then take the smallest
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’

\[
\tilde{n} = \frac{n}{\log n}
\]

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

Do at most one query in \( H \) . . .

and one query in at most two different \( L_i \)

then take the smallest
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’

$$\tilde{n} = \frac{n}{\log n}$$

How do we answer a query in $A$?

Do at most one query in $H$ . . .

and one query in at most two different $L_i$

then take the smallest
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’

\[
\tilde{n} = \frac{n}{\log n}
\]

How do we answer a query in $A$?

Do at most one query in $H \ldots$

and one query in at most two different $L_i$

then take the smallest

\[
i' = \left\lceil \frac{i}{\log n} \right\rceil \quad j' = \left\lfloor \frac{j}{\log n} \right\rfloor
\]
**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’

\[
\tilde{n} = \frac{n}{\log n}
\]

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

Do at most one query in \( H \) . . . 
and one query in at most two different \( L_i \) 
then take the smallest 

\[
i' = \left\lfloor \frac{i}{\log n} \right\rfloor \quad j' = \left\lfloor \frac{j}{\log n} \right\rfloor
\]
**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’

\[
\tilde{n} = \frac{n}{\log n}
\]

How do we answer a query in $A$?

Do at most one query in $H \ldots$

and one query in at most two different $L_i$
then take the smallest

\[
i' = \left\lceil \frac{i}{\log n} \right\rceil \quad j' = \left\lfloor \frac{j}{\log n} \right\rfloor
\]
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’

$$\tilde{n} = \frac{n}{\log n}$$

How do we answer a query in $A$?

Do at most one query in $H$ . . .

and one query in at most two different $L_i$

then take the smallest

$$i' = \left\lceil \frac{i}{\log n} \right\rceil \quad j' = \left\lfloor \frac{j}{\log n} \right\rfloor$$
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’

\( \tilde{n} = \frac{n}{\log n} \)

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

Do at most one query in \( H \) . . .

and one query in at most two different \( L_i \)

then take the smallest

\[
i' = \left\lceil \frac{i}{\log n} \right\rceil \quad j' = \left\lfloor \frac{j}{\log n} \right\rfloor
\]
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’

\[ \tilde{n} = \frac{n}{\log n} \]

How do we answer a query in $A$?

Do at most one query in $H \ldots$

and one query in at most two different $L_i$ (here we query $L_1$ and $L_5$)

then take the smallest
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’

\[
\tilde{n} = \frac{n}{\log n}
\]

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

Do at most one query in \( H \) . . .

and one query in at most two different \( L_i \) (here we query \( L_1 \) and \( L_5 \))

then take the smallest

\[
i' = \left\lceil \frac{i}{\log n} \right\rceil \quad j' = \left\lfloor \frac{j}{\log n} \right\rfloor
\]

This takes \( O(1) \) total 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*.

\[ \tilde{n} = \frac{n}{\log n} \]

How do we answer a query in $A$?

Do at most one query in $H$ . . .

and one query in at most two different $L_i$ (here we query $L_1$ and $L_5$)

then take the smallest

\[ i' = \left\lceil \frac{i}{\log n} \right\rceil \quad j' = \left\lfloor \frac{j}{\log n} \right\rfloor \]

*This takes $O(1)$ total 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'*

\[ \tilde{n} = \frac{n}{\log n} \]

How do we answer a query in $A$?

Do at most one query in $H$ . . .

and one query in at most two different $L_i$ (here we query $L_1$ and $L_5$)

then take the smallest $i' = \lceil \frac{i}{\log n} \rceil$ $j' = \lfloor \frac{j}{\log n} \rfloor$

*This takes $O(1)$ total 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’

$\tilde{n} = \frac{n}{\log n}$

How do we answer a query in $A$?

Do at most one query in $H$ . . .

and one query in at most two different $L_i$ (here we query $L_1$ and $L_5$)

then take the smallest

This takes $O(1)$ total query time

Solution 4

$O(n \log \log \log n)$ space \hspace{1cm} $O(n \log \log \log n)$ prep time \hspace{1cm} $O(1)$ query time
**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’

### How do we answer a query in $A$?

Do at most one query in $H$ . . .

and one query in at most two different $L_i$ (here we query $L_1$ and $L_5$)

then take the smallest

$$i' = \left\lceil \frac{i}{\log n} \right\rceil \quad j' = \left\lfloor \frac{j}{\log n} \right\rfloor$$

This takes $O(1)$ total query time

---

### Solution 4

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

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

$A$

\[
\begin{array}{cccccccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\
23 & 17 & 8 & \textcolor{blue}{73} & 51 & 82 & \textcolor{blue}{19} & 32 & 5 & 67 & 91 & 14 & 46 & 9 & 21 & 54 \\
\end{array}
\]

$i = 3$  \hspace{1cm} $j = 7$  \hspace{1cm} RMQ(3, 7) = 6

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

**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
Range minimum query summary

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

$A = \begin{bmatrix} 23 & 17 & 8 & 73 & 51 & 82 & 19 & 32 & 5 & 67 & 91 & 14 & 46 & 9 & 21 & 54 \end{bmatrix}$

$i = 3 \quad j = 7$

$\text{RMQ}(3, 7) = 6$

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

**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

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

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 $A[i, j]$

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

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

Yes...
Range minimum query summary

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

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

Can we do $O(n)$ space and $O(1)$ query time? Yes... but not until next lecture