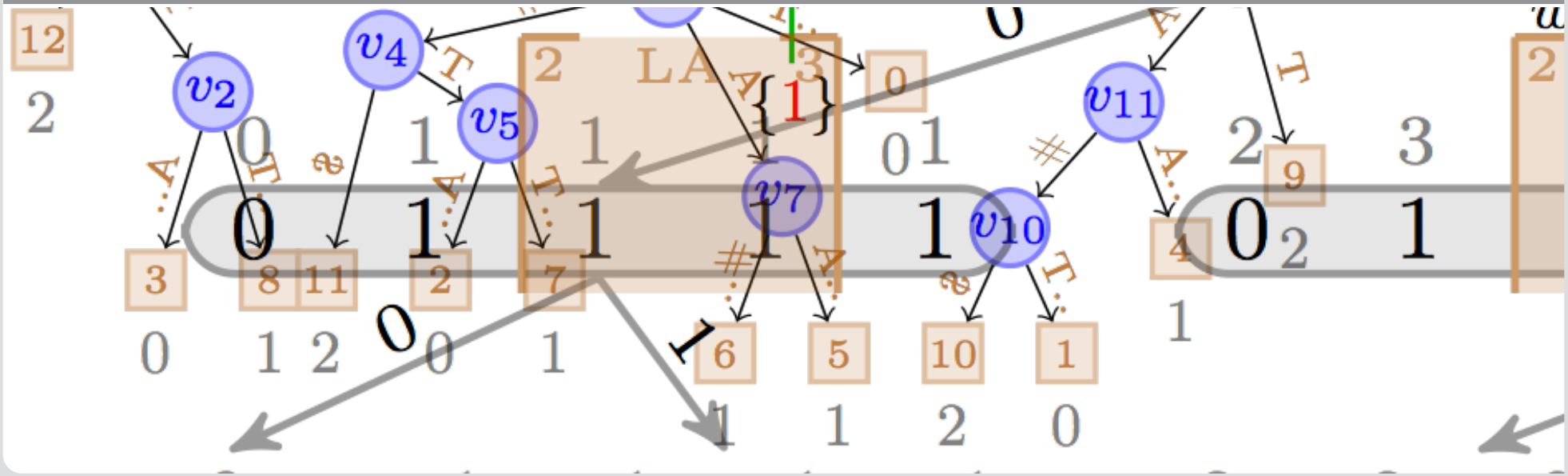
Karlsruhe Institute of Technology

Advanced Data Structures

Simon Gog – *gog@kit.edu*

Institute of Theoretical Informatics - Algorithmics



Range minimum queries (RMQs)

Karlsruhe Institute of Technology

Definition

Given an array A of length n containing elements from a totally ordered set. A range minimum query $rmq_A(\ell, r)$ returns the position of the minimal element in the sub-array $A[\ell, r]$:

$$rmq_A(\ell, r) = \arg\min_{\ell \leq k \leq r} A[k]$$

Example

$A =$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	2	4	7	1	9	3	5	7	4	6	4	3	1	4	8

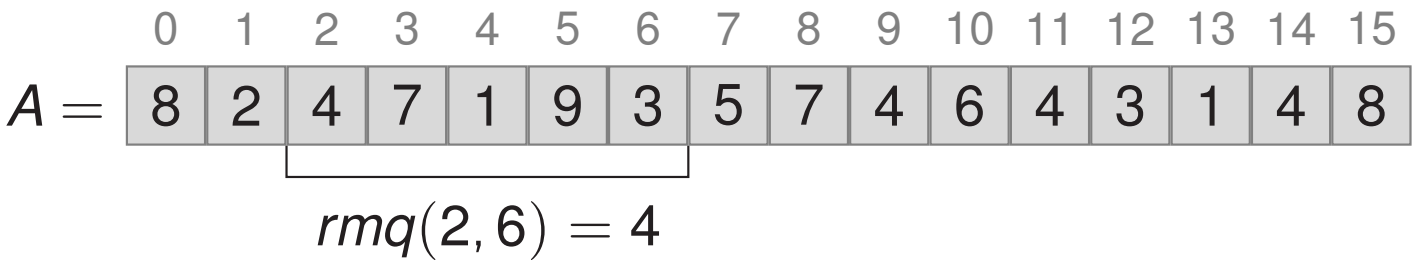
Range minimum queries (RMQs)

Definition

Given an array A of length n containing elements from a totally ordered set. A range minimum query $rmq_A(\ell, r)$ returns the position of the minimal element in the sub-array $A[\ell, r]$:

$$rmq_A(\ell, r) = \arg\min_{\ell \leq k \leq r} A[k]$$

Example



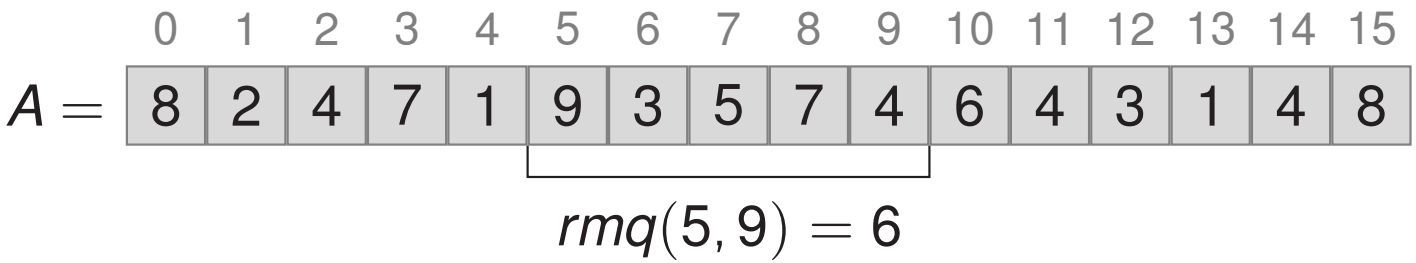
Range minimum queries (RMQs)

Definition

Given an array A of length n containing elements from a totally ordered set. A range minimum query $rmq_A(\ell, r)$ returns the position of the minimal element in the sub-array $A[\ell, r]$:

$$rmq_A(\ell, r) = \arg\min_{\ell \leq k \leq r} A[k]$$

Example



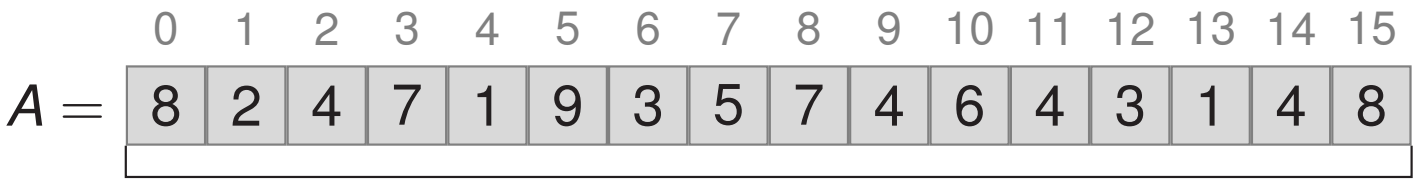
Range minimum queries (RMQs)

Definition

Given an array A of length n containing elements from a totally ordered set. A range minimum query $rmq_A(\ell, r)$ returns the position of the minimal element in the sub-array $A[\ell, r]$:

$$rmq_A(\ell, r) = \arg\min_{\ell \leq k \leq r} A[k]$$

Example



$$rmq(0, 15) = 4$$

Overview

- Notation: Complexity of an algorithm is denoted with $\langle f(n), g(n) \rangle$, where $f(n)$ is preprocessing time and $g(n)$ query time.
- Different solutions:
  - naïve approach 1: $\langle O(n^2), O(1) \rangle$ using $O(n^2)$ words of space
  - naïve approach 2: $\langle O(1), O(n) \rangle$
  - $\langle O(n), O(\log n) \rangle$ using $O(n)$ words of space
  - $\langle O(n \log n), O(1) \rangle$ using $O(n \log n)$ words of space
  - $\langle O(n \log \log n), O(1) \rangle$ using $O(n \log \log n)$ words of space
  - $\langle O(n), O(1) \rangle$ using $O(n)$ words of space
  - $\langle O(n), O(1) \rangle$ using $4n + o(n)$ bits of space
  - $\langle O(n), O(1) \rangle$ using $2n + o(n)$ bits of space

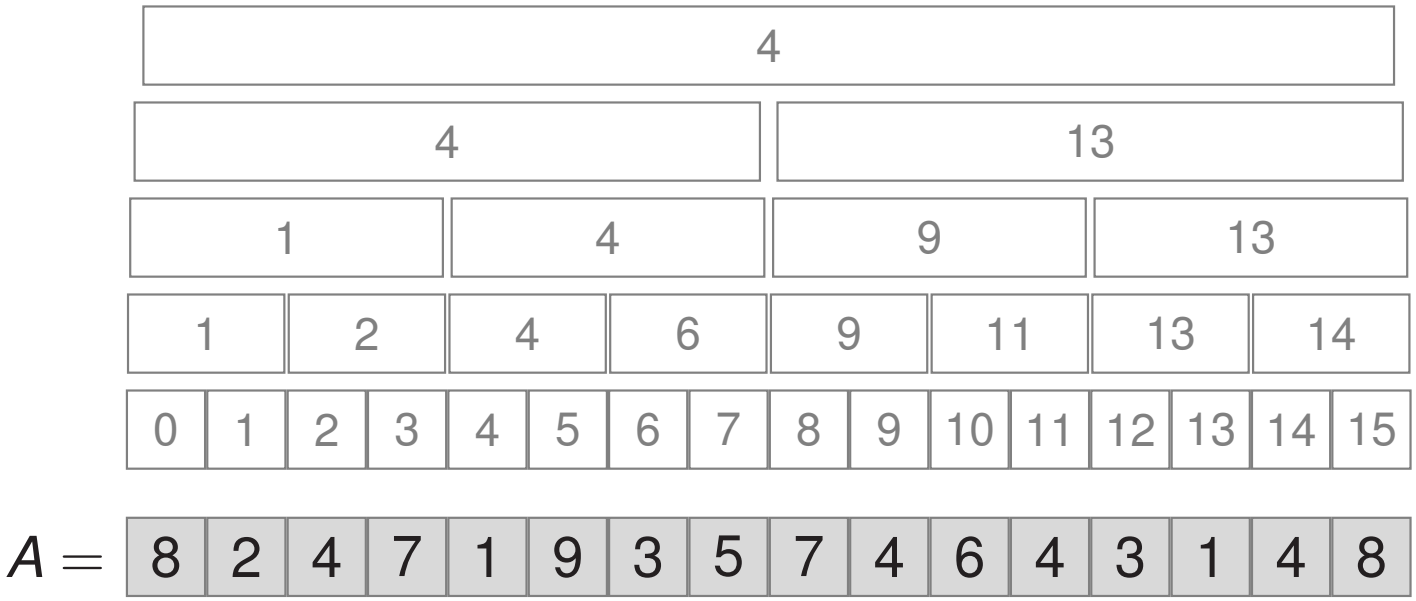
Literature

M.A. Bender, M. Farach-Colton: The LCA Problem Revisited. (LATIN 2000) K. Sadakane: Compressed Suffix Trees with Full Functionality. (TCS 2007)

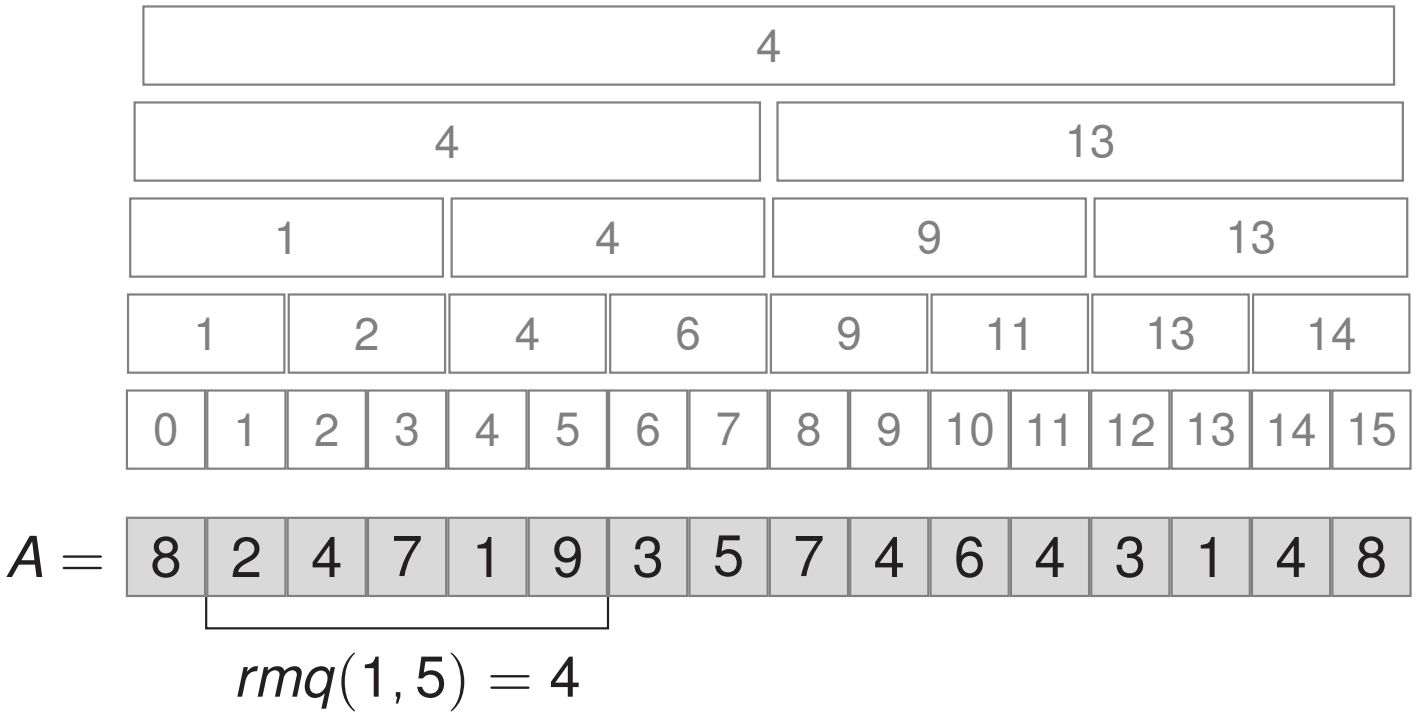
$\langle O(n), O(\log n) \rangle$ – solution #1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$A =$	8	2	4	7	1	9	3	5	7	4	6	4	3	1	4	8

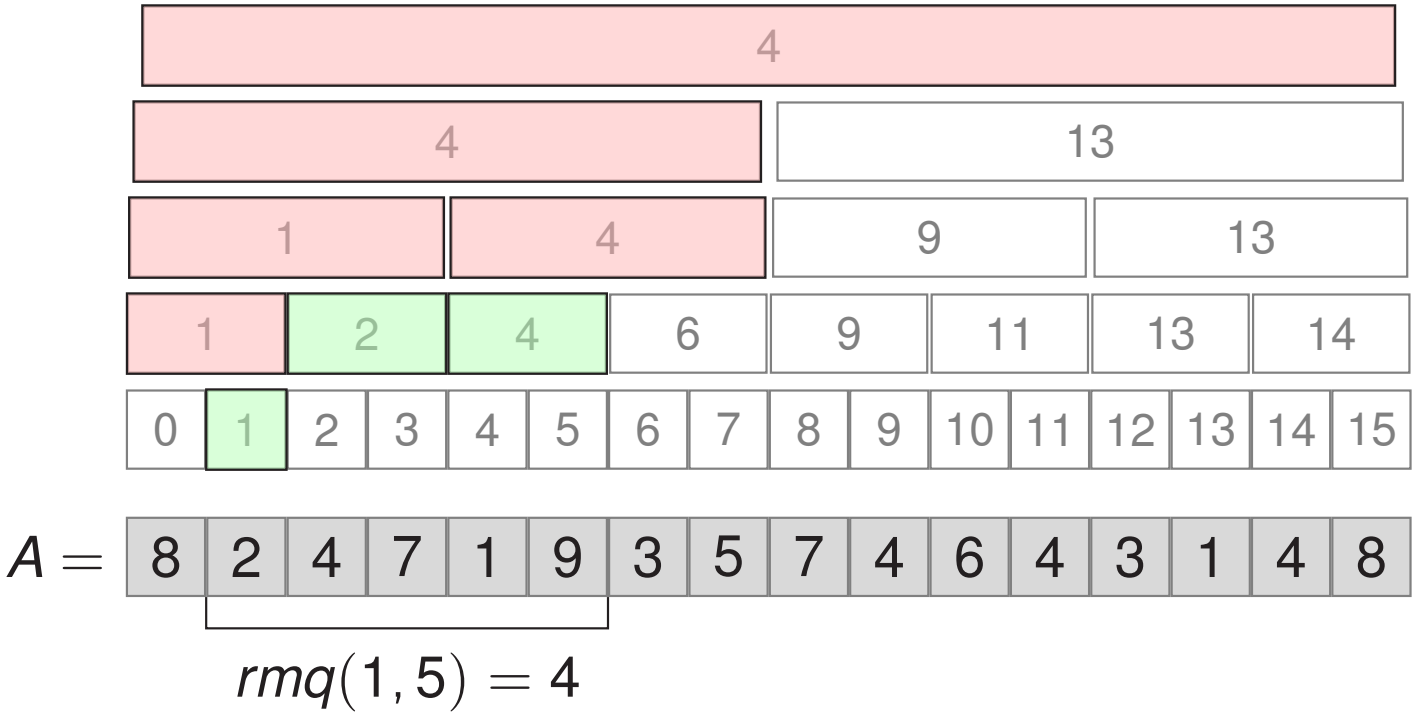
$\langle O(n), O(\log n) \rangle$ – solution #1



$\langle O(n), O(\log n) \rangle$ – solution #1



$\langle O(n), O(\log n) \rangle$ – solution #1



$\langle O(n), O(\log n) \rangle$ – solution #1

- Store index of minimum in binary interval tree.
- Tree has $O(n)$ nodes.
- Follow all nodes which overlap with the query interval but are not fully contained in it (at most 2 per level).
- So not more than $2 \log n$ such nodes in total.
- Select all children of these nodes which are fully contained in the query interval.
- From these nodes select the index with minimal value.

$\langle O(n \log n), O(1) \rangle$ – solution #2

- For each item $A[i]$ store an array $M_i[0, \log n]$.
- $M_i[j] = rmq_A(i, i + 2^j - 1)$
- Space is $O(n \log n)$ words
- How long does pre-computation take?

Querying

Find the largest k with $2^k \leq \ell - r + 1$. Then

$$rmq_A(\ell, r) = \begin{cases} M_i[k] & \text{if } A[M_i[k]] < A[M_{j-2^k+1}[k]] \\ M_{j-2^k+1}[k] & \text{otherwise} \end{cases}$$

Question: How can k be determined in constant time?

$\langle O(n \log\log n), O(1) \rangle$ solution

- Split A into $t = \frac{n}{\log n}$ blocks $B_0, ..., B_{t-1}$. B spans $O(\log n)$ items of A.
- Create an array $S[0, t-1]$ with $S[i] = min\{x \in B_i\}$
- Build rmq structure #2 for S
- For each block B_i of $O(\log n)$ elements build rmq structure #2
- Total space: $O(n) + O(n \log\log n)$

Querying

- Determine blocks $B_{\ell'}$, $B_{r'}$ which contain ℓ and r
- Calculate $m = rmq_S(\ell' + 1, r' - 1)$
- Let k_0, k_1, k_2 be the results of the RMQs in blocks ℓ', r', and m relative to A
- Return $\arg\min_{k_i} A[k_i]$ for $0 \leq k \leq 3$

$\langle O(n), O(1) \rangle$ solution

Karlsruhe Institute of Technology

Definition

The Cartesian Tree C of an array is defined as follows:

- The root of C is the minimum element of the array and is labeled with its position
- Removing the root splits the array into two pieces
- The left and right children of the root are recursively constructed Cartesian trees of the left and right subarray

Exercise: How to construct C in linear time?

$\langle O(n), O(1) \rangle$ solution

Solution overview:

- Partition the array into blocks of size s
- Each block corresponds to a Cartesian Tree of size s
- Represent the Cartesian Tree via LOUDS ($2s - 1$ bits)
- For each of the 2^{2s-1} bit pattern and the s^2 possible RMQ queries store the result in a table P („Four Fussians Trick”)
- I.e. P requires $O(2^{2s}s^2)$ words space
- For $s = \frac{\log n}{4}$ P requires $o(n)$ words of space

LCA & $\pm 1 RMQ$

Karlsruhe Institute of Technology

See chalkboard presentation.

$\langle O(n), O(1) \rangle$ solution using $4n + o(n)$ bits

See chalkboard presentation.