2-Dimensional Range Minimum Queries

Amihood Amir[1], Johannes Fischer[2], and Moshe Lewenstein[1]

[1]Computer Science Department
Bar Ilan University

[2]Institut für Informatik
Ludwig-Maximilians-Universität München

July 2007, London (Ontario)

Outline

1. **Introduction to RMQs**
   - Formal Problem Definition
   - Previous Results
2. **Solution Methods**
   - Overview
   - Preprocessing of First Level
   - Other Levels and Microblock-Queries

Outline

1. **Introduction to RMQs**
   - Formal Problem Definition
   - Previous Results

2. **Solution Methods**
   - Overview
   - Preprocessing of First Level
   - Other Levels and Microblock-Queries

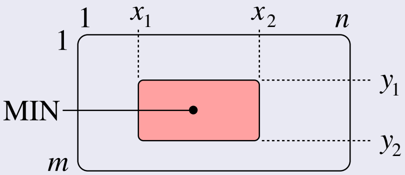
Formal Problem Definition

The Problem

given: matrix $A[1..m][1..n]$ (totally ordered objects)

task: preprocess A to answer "efficiently"

$$\mathrm{RMQ}(y_1, y_2, x_1, x_2) = \underset{(y,x)\in[y_1:y_2]\times[x_1:x_2]}{\mathrm{argmin}} A[y][x]$$



Main Result

The Problem

given: matrix $A[1..m][1..n]$ (totally ordered objects)

task: preprocess A to answer "efficiently"

$$\mathrm{RMQ}(y_1, y_2, x_1, x_2) = \underset{(y,x)\in[y_1:y_2]\times[x_1:x_2]}{\operatorname{argmin}} A[y][x]$$

Theorem (2-Dimensional RMQs)

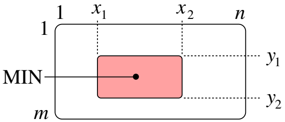
$O(nm(k + \log^{[k+1]}(mn)))$-preprocessing using $O(kmn)$ space for $O(1)$-RMQs, for any $k > 1$.
*Converges towards $O(mn \log^*(mn))$ preprocessing time **and** space, and $O(1)$ query time.*

Our Result in Context

$N := mn$

	preprocessing time	preprocessing space	query time
Gabow et al.'84	$O(N \log N)$	$O(N \log N)$	$O(\log N)$
Chazelle-Rosenberg'89	$O(RN)$	$O(RN)$	$O(\alpha^2(RN, N))$
(with R =const)	$O(N)$	$O(N)$	$O(\alpha^2(RN, N))$
Mäkinen'03	$O(N \log m)$	$O(N \log m)$	$O(1)$
this paper	$O(N(k + \log^{[k+1]} N))$	$O(kN)$	$O(1)$
(with $k = 2$)	$O(N \log \log \log N)$	$O(N)$	$O(1)$
this paper	$O(N \log^* N)$	$O(N \log^* N)$	$O(1)$

$R \geq 144$, $k \geq 2$ (both not necessarily constants!)



Results for 1-Dimensional RMQs

- Results for $O(n)$ preprocessing time and $O(1)$ query time:

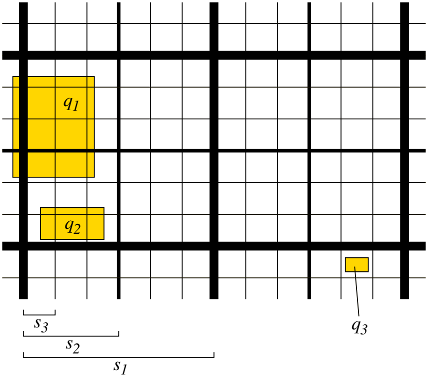
	space (bits) final	space (bits) peak
Berkman-Vishkin'93	$O(n \log n)$	$O(n \log n)$
Bender-Farach-Colton'00	$O(n \log n)$	$O(n \log n)$
Alstrup et al.'02	$O(n \log n)$	$O(n \log n)$
Fischer-Heun'06	$O(n \log n)$	$O(n \log n)$
Sadakane'02	$4n + o(n)$	$O(n \log n)$
Fischer-Heun'07	$2n + o(n)$	$2n + o(n)$

- Cartesian Trees (=treaps) important tool for all!

Outline

Overview of the Algorithm

- impose grids on A (widths $s_1 \geq s_2 \geq \cdots \geq s_k$)
- preprocess each "layer" seperately
- level i: queries crossing grid s_i, but no grid s_j for $j < i$
- other queries: precompute all possible!
- time: $$O(\underbrace{kN}_{k \text{ levels}} + \underbrace{N \log^{[k+1]} N}_{\text{sorting on level } k})$$



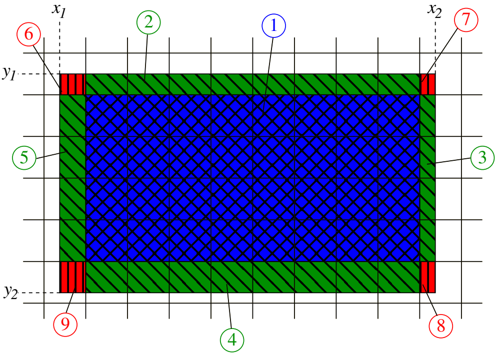
Preprocessing on Level 1

- assume A is square ($\Rightarrow n = m, N = n^2$)
- $s_1 = \log n$: grid-width on level 1
- decompose $\mathrm{RMQ}(y_1, y_2, x_1, x_2)$ into nine sub-queries:

1 several blocks in both directions

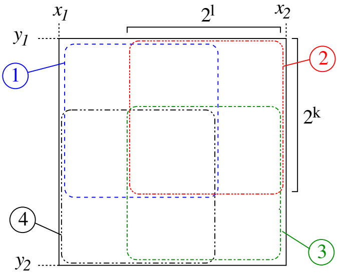
2-5 several blocks in 1 direction

6-9 in-block, but "touches" boundary



Precomputation of Queries 1–9

- Precompute only queries that span $2^k \times 2^l$ blocks



- Answer all queries by selecting at most four overlapping "power-of-two"-queries
- same idea as for 1D-RMQ!

Recursive Partitioning

- perform same preprocessing for grid-widths

$$s_2 = \log\log n, s_3 = \log\log\log n, \ldots, s_k = \log^{[k]} n$$

  - either stopping at some fixed $k > 1\ldots$
  - $\ldots$ or until $s_k = O(1)$ ($\iff k = \Theta(\log^* n)$)

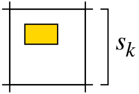
Recursive Partitioning

- perform same preprocessing for grid-widths

$$s_2 = \log\log n, s_3 = \log\log\log n, \ldots, s_k = \log^{[k]} n$$

  - either stopping at some fixed $k > 1$...
  - ...or until $s_k = O(1)$ ($\iff k = \Theta(\log^* n)$)
- naive query-answering would cost $O(k)$ or $O(\log^* n)$ time!

$\Rightarrow$ store additional structures of size $o(n)$ for selecting the right grid in $O(1)$ time

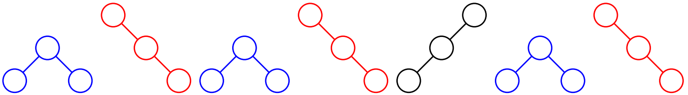


- still left with microblock-queries

What to do with microblock-queries?

Key-property for 1D-RMQs

2 blocks have same answers to all RMQs $\iff$ they have equal Cartesian Trees



$$A = \boxed{3\;0\;2} \mid \boxed{1\;1\;1} \mid \boxed{2\;0\;2} \mid \boxed{1\;1\;2} \mid \boxed{3\;2\;1} \mid \boxed{1\;0\;2} \mid \boxed{0\;2\;3}$$

Problem: nothing similar for 2D

Weaker Property

2 blocks have same answers to all RMQs $\Longleftarrow$ elements have same relative order

What to do with microblock-queries (2)?

Weaker Property

2 blocks have same answers to all RMQs $\Longleftarrow$ elements have same relative order

$\Rightarrow$ sort microblocks (e.g. row-wise) to get permutation of $[1 : S]$ (=relative order), $S = s_k^2$

- precompute microblock-queries only for different permutations and **not** for all occurring microblocks

What to do with microblock-queries (2)?

Weaker Property

2 blocks have same answers to all RMQs $\Longleftarrow$ elements have same relative order

$\Rightarrow$ sort microblocks (e.g. row-wise) to get permutation of $[1 : S]$ (=relative order), $S = s_k^2$

- precompute microblock-queries only for different permutations and **not** for all occurring microblocks
- space is

$$O(S^2 \times S!) = \cdots = O(N) \quad (\text{if } k > 1)$$

- time is $O(\frac{N}{S} \times S \log S) = O(N \log^{[k+1]} n)$

Summary

- preprocessing-scheme for 2D-RMQs (N: size of input)
  - space $O(N)$
  - preprocessing time $\widetilde{O}(N)$
  - query time $O(1)$
- generalizes to higher dimensions d: query time $O(C^d)$, $C =$const
- open question: can we achieve $O(N)$ preprocessing time **and** $O(1)$ query time?
  - impossible for slightly more general operations! (Chazelle-Rosenberg'89)