madalgo
CENTER FOR MASSIVE DATA ALGORITHMICS

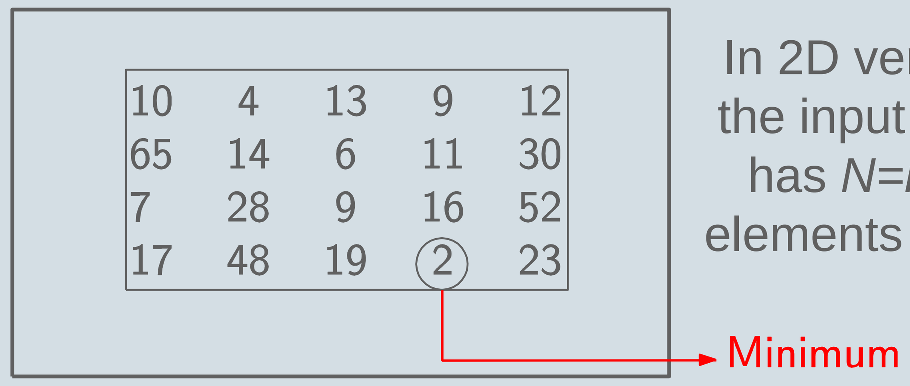
Pooya Davoodi
Aarhus University

Space Efficient Range Minimum Queries

Introduction

Problem
Preprocessing an input array such that the given query asking for the index of the minimum element in a rectangular range within the array is solved efficiently.

Succinct Models
- Indexing: Probes into the input.
- Encoding: No access to the input.



In 2D version, the input array has $N=m\times n$ elements ($m\leq n$)

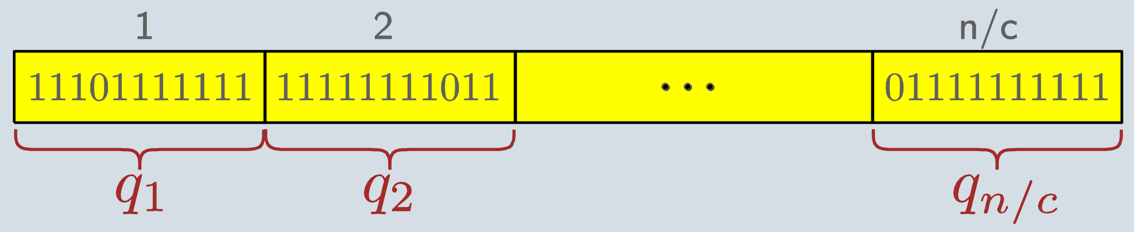
Applications
Databases, geographic information systems, graphics, computing lowest common ancestors in trees, pattern matching, document retrieval queries, maximum segment queries and more.

Indexing Lower Bound (1D and 2D)[1]

Theorem

Any RMQ algorithm using n / c bits additional space, requires $\Omega(c)$ probes into the input.

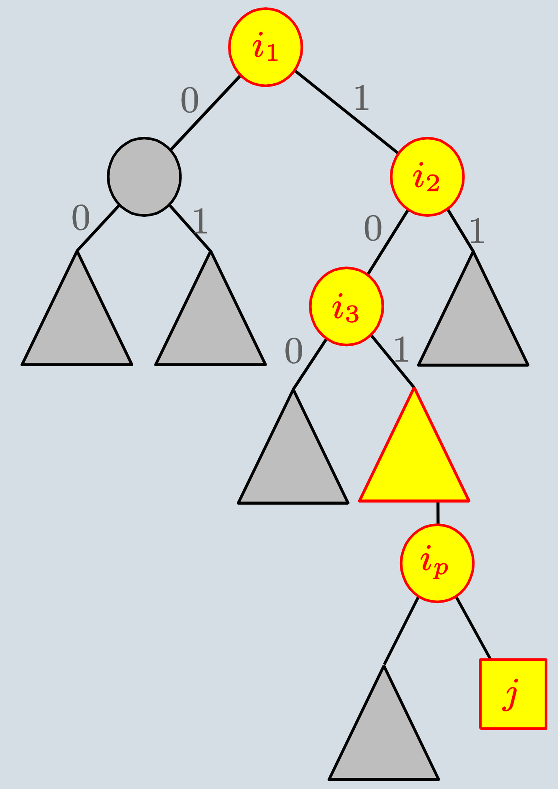
Proof
- Consider n/c queries for $c^{n/c}$ different {0,1} inputs with exactly one zero in each block.



- $c^{n/c} / 2^{n/c}$ inputs share some data structure.
- Fix the data structure.
- Every query is a decision tree of height ≤ d.

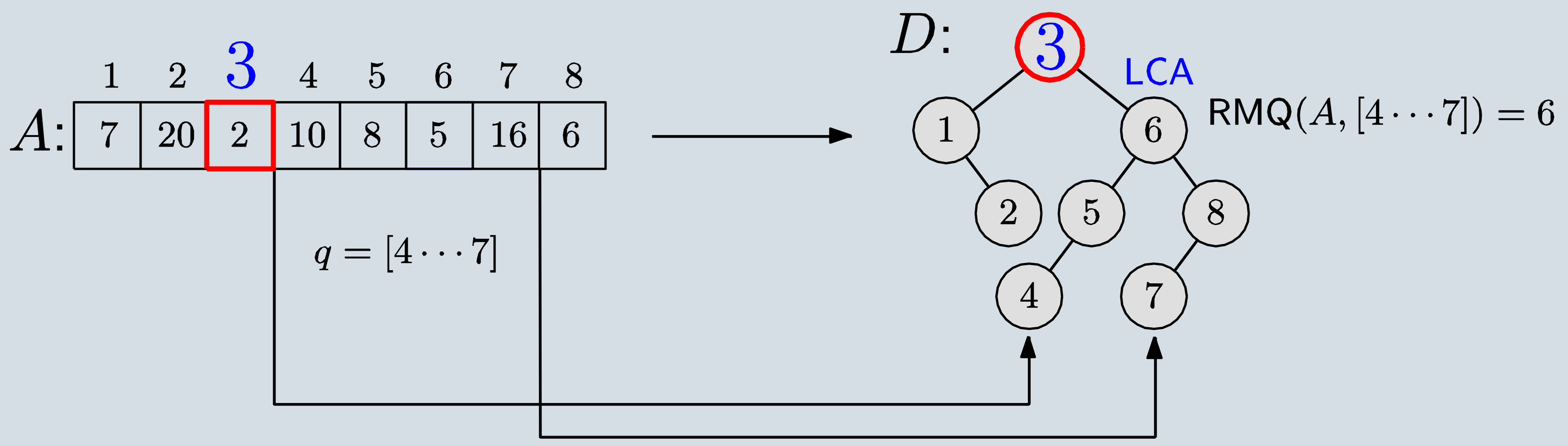
The path (1,0,1,...,1) ending at the leaf j shows that the answer of the query over the input matching with the probes of the path is j.

- Combine queries to a decision tree.
- Prune non-reachable branches for the inputs sharing the data structure.
- # zeroes on any path ≤ n/c.
- $\frac{c^{n/c}}{2^{n/c}} \leq$ #inputs = #leaves $\leq \binom{d+n/c}{n/c}$.
- Query time $d = \Omega(c)$.



One Dimensional Data Structure

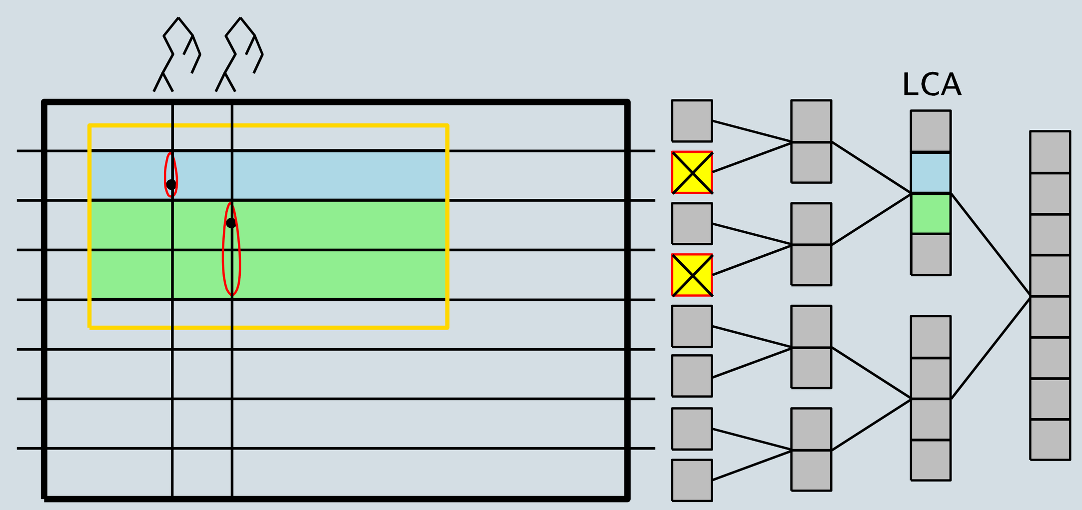
- A Cartesian Tree encodes the input 1D array of size n elements in $4n$+o(n) bits s.t. the query can be solved without accessing the input in O(1) time (Sadakane'07).



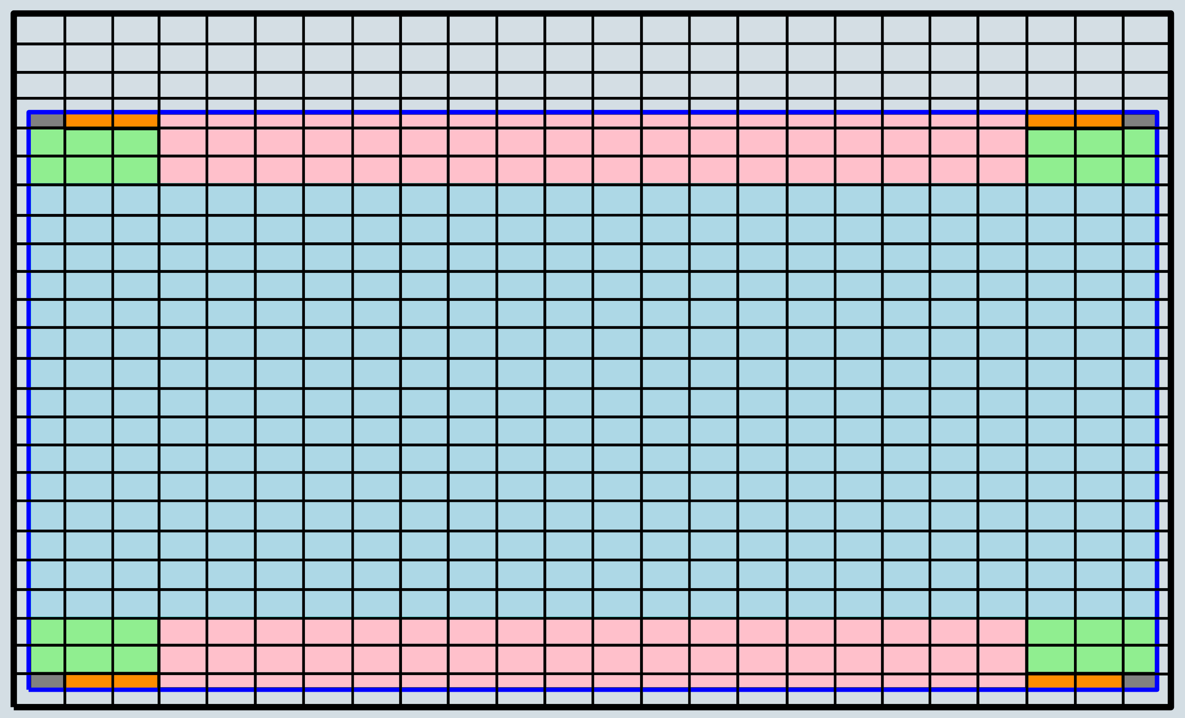
- The best indexing and encoding data structures have size $2n$+o(n) bits matching the lower bound of $2n$-Θ(log n) bits (Fischer and Heun'03, Fischer'07).

Indexing 2D Data Structure: Linear Space[2]

- Partition the input recursively.
- Construct a binary tree in each level of the recursion.
- Solve the queries spanning over the blocks.



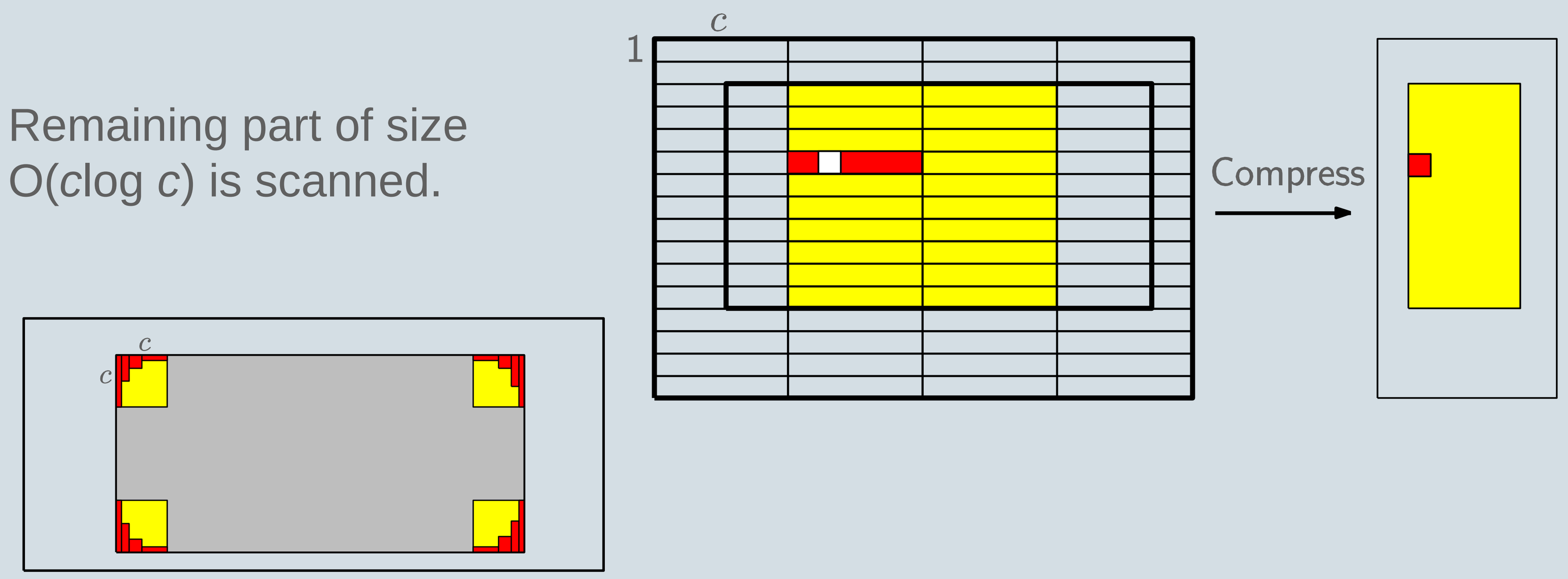
- Use Four-Russians trick in the last level where the block size is O(log N).



Indexing 2D Data Structure: Time-space Trade-off[3]

- Divide the input to blocks of size 2^i x $c/2^i$ in log c steps.
- In each step, construct an indexing data structure of size O(N/c) bits for the compressed matrix.

Remaining part of size O($c$log c) is scanned.



Results

No.	Query time	Space (bits)	Preprocessing time
1	Ω(c)	O(N/c)+|A|	-
2	O(1)	O(N)+|A|	O(N)
3	O($c\log^2 c$)	O(N/c)+|A|	O(N)
4	-	Ω(N log m)	-
5	O(1)	O(N log n)	O(N)

Encoding 2D Lower Bound[4]

- Define a set of $\left(\frac{m}{2}!\right)^{\frac{n}{2}-\frac{m}{4}}$ different matrices.

Bits required is at least:

$$\log\left(\frac{m}{2}!\right)^{\frac{n}{2}-\frac{m}{4}} = \Theta(N \log m)$$