Encoding Data Structures

Rajeev Raman

University of Leicester

ERATO-ALSIP special seminar

RMQ problem

Given a static array $A[1..n]$, pre-process A to answer queries:

RMQ(l, r) : return $\max_{l \leq i \leq r} A[i]$.

47	43	97	24	46	33	34	85	67	18	83	45

RMQ problem

Given a static array $A[1..n]$, pre-process A to answer queries:

RMQ(l, r) : return $\max_{l \leq i \leq r} A[i]$.

47	43	97	24	46	33	34	85	67	18	83	45

$$\text{RMQ}(5, 10) = 85.$$

Data Structuring Problems

This is a *data structuring* problem.

- Pre-process input data (here array A) to answer long series of queries.
- Want to minimize:
  1. Space usage of data structure.
  2. Query time.
  3. Time/space for pre-processing.
- For now, we assume the input data is *static* i.e. it does not change between queries.

Solution to RMQ Problem: Cartesian Tree

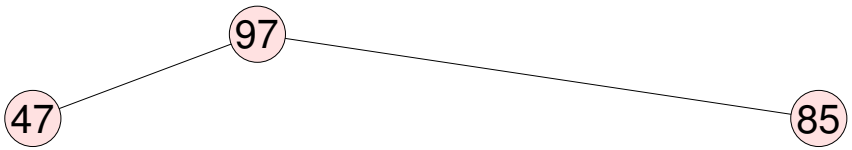
The Cartesian tree of A [Vuillemin '80] is a binary tree.

47	43	97	24	46	33	34	85	67	18	83	45

- Place largest value at root of tree.

Solution to RMQ Problem: Cartesian Tree

The Cartesian tree of A [Vuillemin '80] is a binary tree.

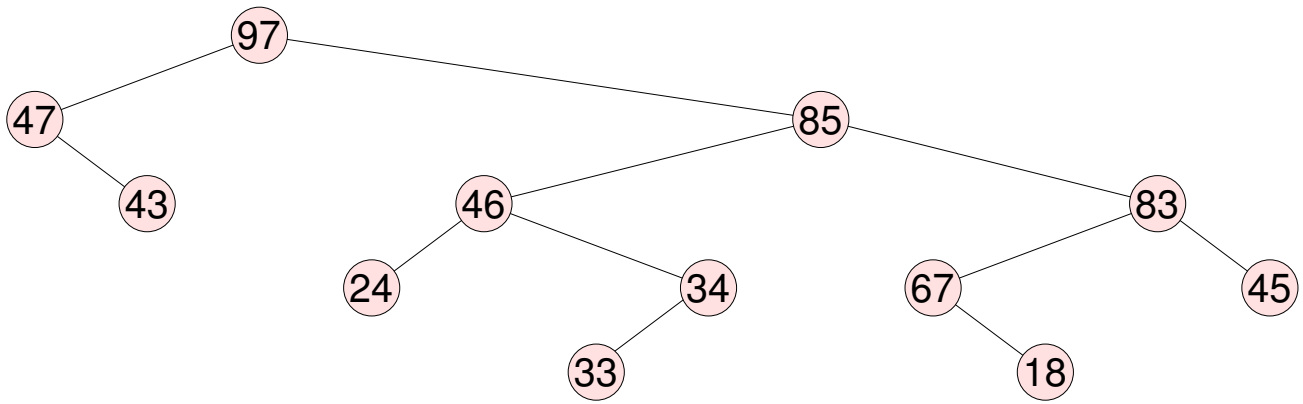


47	43	97	24	46	33	34	85	67	18	83	45

- Place largest value at root of tree.
- Recurse on sub-arrays to left and right.

Solution to RMQ Problem: Cartesian Tree

The Cartesian tree of A [Vuillemin '80] is a binary tree.

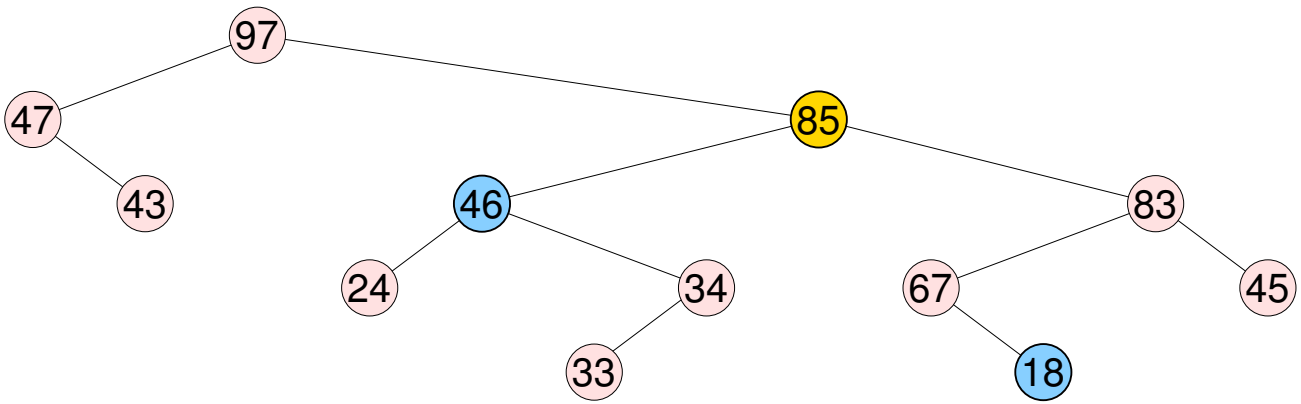


47	43	97	24	46	33	34	85	67	18	83	45

- Place largest value at root of tree.
- Recurse on sub-arrays to left and right.

Solution to RMQ Problem: Cartesian Tree

The Cartesian tree of A [Vuillemin '80] is a binary tree.



47	43	97	24	46	33	34	85	67	18	83	45

- Place largest value at root of tree.
- Recurse on sub-arrays to left and right.
- RMQ is the lowest common ancestor (LCA) of interval endpoints.
- n-node binary tree can support LCA in $O(n)$ space and $O(1)$ time. [Gabow et al.]

Compressing RMQ

- Many applications where using $O(n)$ memory words is way too much.
  - Suffix tree on a string of n bits occupies $O(n)$ words $= O(n \lg n)$ bits[1].
- The same is true for many applications of RMQ.
  - Can reconstruct A by asking RMQ(i, i) queries.
  - In general A can't be compressed below $\Omega(n \log n)$ bits.
  - In specific applications (e.g. LCP array), A can be compressed, but then accessing $A[i]$ is slow.

Can we do better?

[1] $\lg = \log_2$.

The RMQ Problem Redefined

Given a static array $A[1..n]$, pre-process A to answer queries:

$$\text{RMQ}(l, r) = \arg \max_{l \leq i \leq r} A[i]$$

.

47	43	97	24	46	33	34	85	67	18	83	45

The RMQ Problem Redefined

Given a static array $A[1..n]$, pre-process A to answer queries:

$$\text{RMQ}(l, r) = \arg \max_{l \leq i \leq r} A[i]$$

.

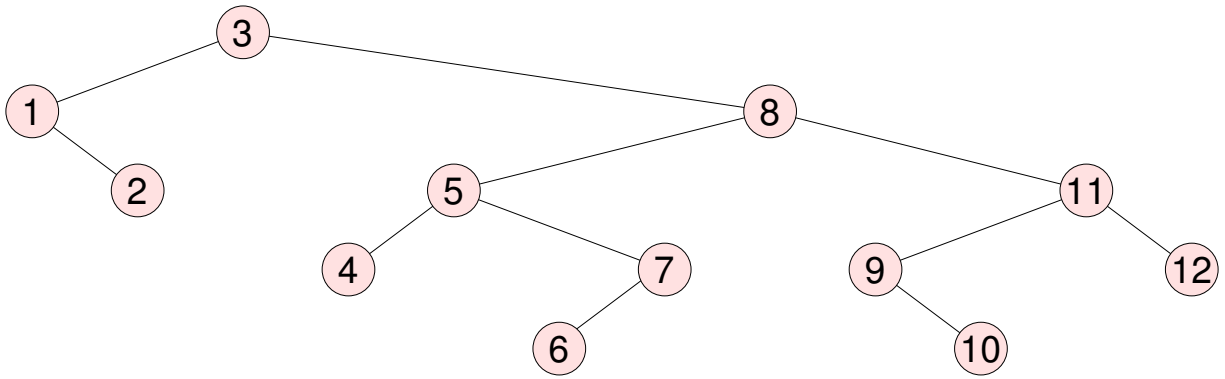
47	43	97	24	46	33	34	85	67	18	83	45

$$RMQ(5, 10) = 8.$$

Encoding RMQ

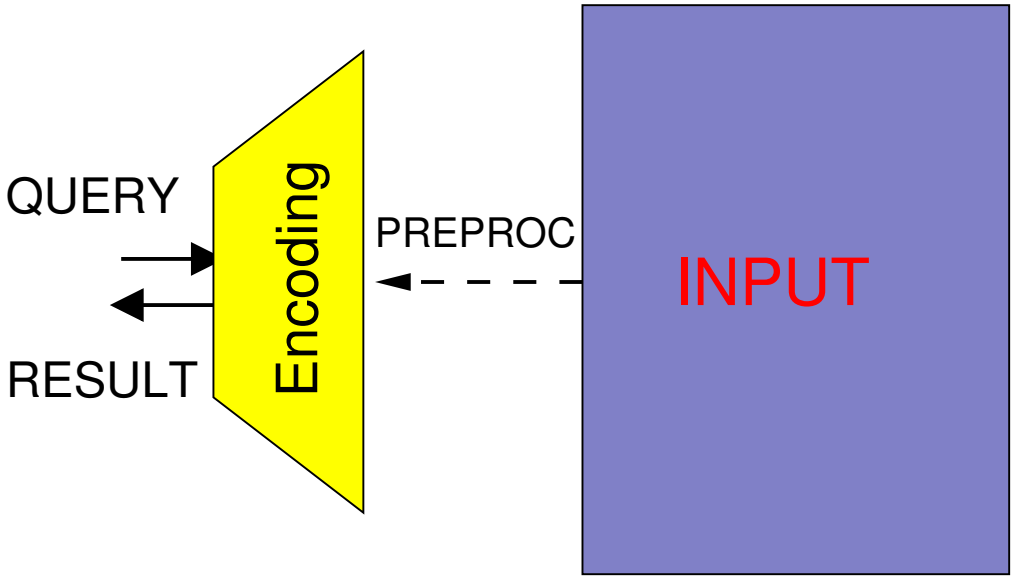
$$RMQ(l, r) = \arg \max_{l \le i \le r} A[i]$$

.



- Shape of Cartesian tree is enough to answer modified RMQ queries.
- There are $\le 4^n$ distinct binary trees on n nodes.
  - Shape can be encoded in $\le \log_2 4^n = 2n$ bits.
- Often the value of $A[i]$ is not needed (cf. LCP array).
- Data structures using $2n + o(n)$ bits, $O(1)$ query time. [Fischer/Heun SICOMP'11],[Davoodi et al. COCOON'12].

Encoding Data Structures



- Preprocess input data to answer a long series of queries.
- Preprocessing creates an *encoding* and deletes input.
- Queries *only* read encoding.
- Minimize: encoding size and query time.
- Non-trivial encodings must be smaller than original input data.

Encoding: Effective Entropy

Encoding $\equiv$ determining *effective entropy*.

- Extensive literature on *succinct and compressed data structures*.
- Entropy: "information content of data."
  - Input data instance x is from a set S.
  - To represent arbitrary x requires $\geq \lceil \lg |S| \rceil$ bits in the worst case.
- Effective Entropy is "the information content of the data *structure*" [Golin et al. TCS]:
  - Given a set of objects S, a set of queries Q.
  - Let $\mathcal{C}$ be the equivalence class on S induced by Q ($x, y \in S$ are equivalent if they cannot be distinguished by queries in Q).
  - We want to store x in $\lceil \lg |\mathcal{C}| \rceil$ bits.
  - Can define *expected* effective entropy as well.

Overview of Talk

- Overview of recent encoding results.
- Asymptotically optimal encodings
  - Range Top-k [Grossi et al. ESA'13]
  - 2D Range Maximum [Brodnik et al. ESA'13]
  - Range Majority [Navarro and Thankachan CPM'14]
  - Range Selection [Navarro et al. FSTTCS'14]
  - Range Maximum Sum Query [Nicholson and Gawrychowski,'14]
  - 2D NLVs [Jo et al. WALCOM'15]
  - Any NLV [Nicholson and R.]
- Minimal encodings
  - RMQs [Fischer, Heun, SICOMP'11][Davoodi et al. PTRS-A '14]
  - Range Second Maximum [Davoodi et al. PTRS-A '14]
  - Bidirectional NLVs [Fischer, TCS'11]
  - Range Min-Max [Gawrychowski and Nicholson,'14]
  - 2D Range Maximum, $m = 2$ [Golin et al. TCS]
  - Range Top-k [Gawrychowski and Nicholson,'14]

Encoding Range Selection

Problem Definition

Given $A[1..n]$ and κ, encode A to answer the query:

select(k, l, r): return the position of the k-th largest value in $A[l..r]$, for any $k \leq \kappa$.

- Related work by many authors including [Brodal and Jørgensen, ISAAC'09] [Jørgensen and Larsen, SODA'11], [Chan and Wilkinson, SODA'13].
- κ must be known at construction time due to lower bound on encoding size.
- Results [Navarro et al. FSTTCS'14]:
  - $O(n \lg \kappa)$ bits and $O(\lg k / \lg \lg n)$ time.
  - Matches time bound of [CW SODA'13] but uses less space. Time cannot be improved using $n(\lg n)^{O(1)}$ space [JL SODA'11].

Lower Bound on Encoding Size

Lemma

Any encoding for *one-sided* top-k queries must take $\Omega(n \lg k)$ bits.

Proof: The index can encode $(n/k) - 1$ independent permutations over k elements $\Rightarrow \Omega((n/k) \cdot k \lg k)$ bits $= \Omega(n \lg k)$ bits.

Proof by example ($k = 3$).

$A =$

3	1	2	4	6	5	8	9	7	$\cdots$

Encode A. Now:

top-k-pos$(1, 4) = \{1, 3, 4\} \Rightarrow A[2] = 1$.
top-k-pos$(1, 5) = \{1, 4, 5\} \Rightarrow A[3] = 2$.
top-k-pos$(1, 6) = \{4, 5, 6\} \Rightarrow A[1] = 3$.

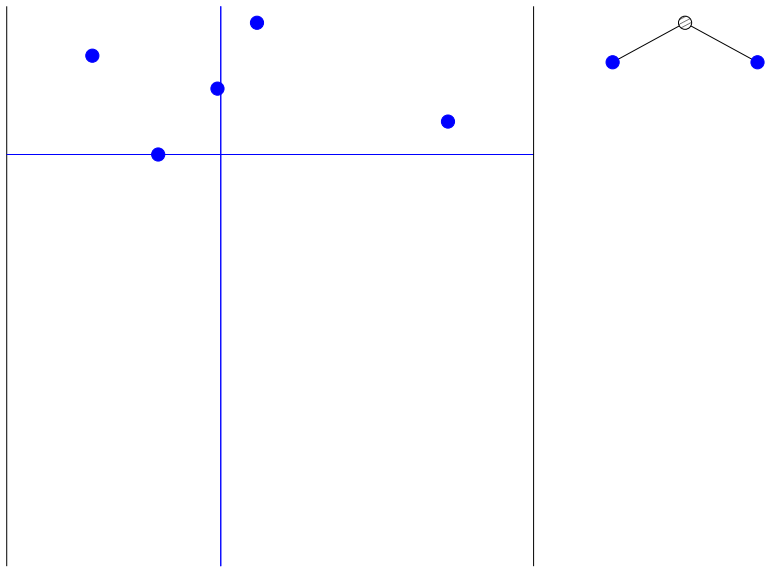
- k must be known at construction time.

Encoding Range Selection: Overview

- View A geometrically in 2D: $A[i] = y \Rightarrow (i, y)$.
- Use idea of *shallow cutting* for top-k [JL SODA'11].
- Take set of n given points and decompose into $O(n/\kappa)$ *slabs* each containing $O(\kappa)$ points such that:
  - For any 2-sided query select(l, r) $\exists$ slab such that it and two other adjacent slabs contain the top κ elements in $A[l..r]$.
  - Gives a kind of encoding: store relative order among these $O(\kappa)$ elements: $O(\kappa \lg \kappa)$ bits/slab $= O(n \lg \kappa)$ bits, optimal!

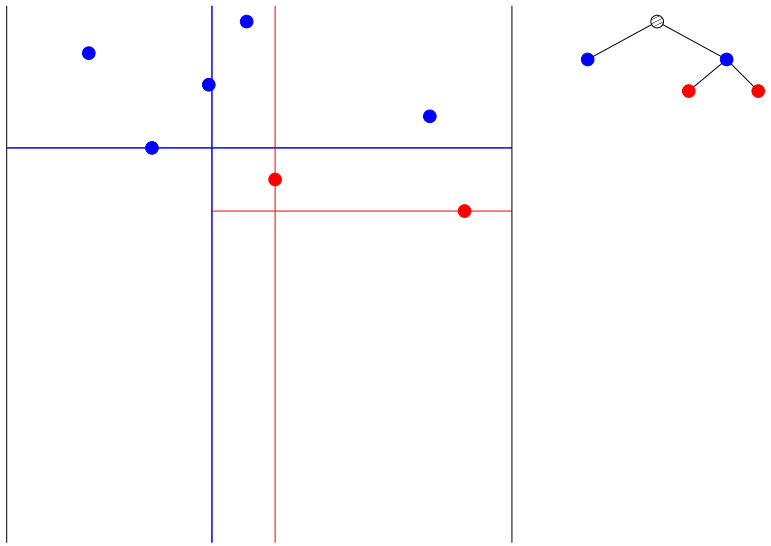
Shallow cutting (pre-processing)

- Sweep a horizontal line down from $x = +\infty$.
- Initially just one slab. Place points as encountered into their slab.
- When slab has $2\kappa - 1$ points, split and create boundaries as follows:
  - median x-coordinate as vertical boundary.
  - bottom y-coordinate as bottom boundary.
- Example: $\kappa = 3$.



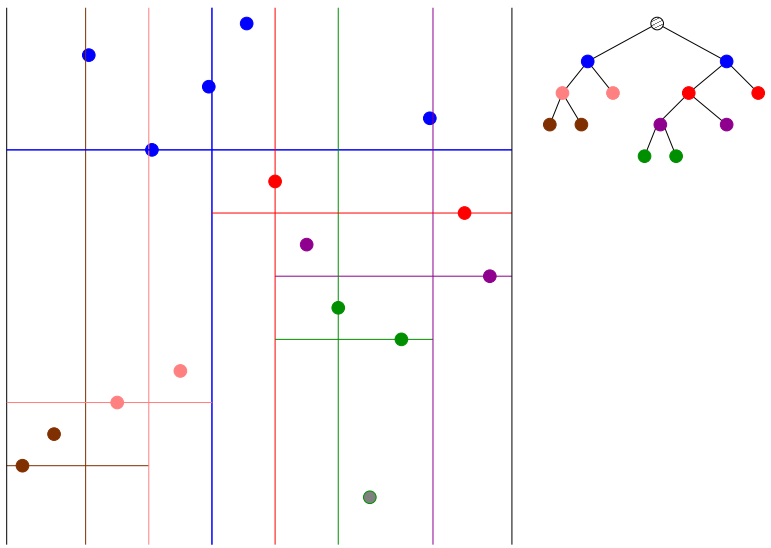
Shallow cutting (pre-processing)

- Sweep a horizontal line down from $x = +\infty$.
- Initially just one slab. Place points as encountered into their slab.
- When slab has $2\kappa - 1$ points, split and create boundaries as follows:
  - median x-coordinate as vertical boundary.
  - bottom y-coordinate as bottom boundary.
- Example: $\kappa = 3$.



Shallow cutting (pre-processing)

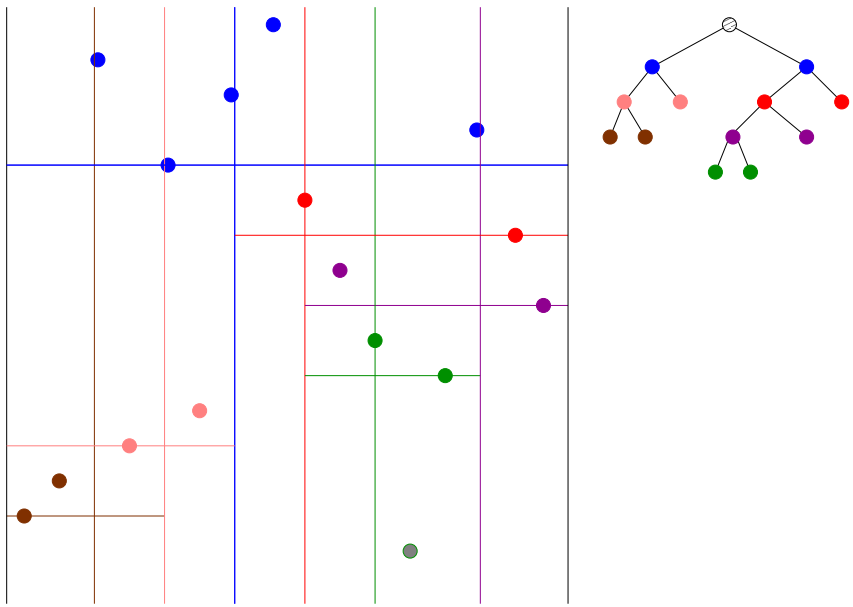
- Sweep a horizontal line down from $x = +\infty$.
- Initially just one slab. Place points as encountered into their slab.
- When slab has $2\kappa - 1$ points, split and create boundaries as follows:
  - median x-coordinate as vertical boundary.
  - bottom y-coordinate as bottom boundary.
- Example: $\kappa = 3$.



- At end: $O(n/\kappa)$ slabs each with $\Theta(\kappa)$ elements.
- Slabs naturally form full binary "tree of slabs" T_s.

Answering Selection Queries

- Create κ-shallow cutting.
- For $O(\kappa)$ points in each slab, store range selection DS: $O(\kappa \lg \kappa)$ bits, or $O(n \log \kappa)$ bits overall (optimal).
- Find resolving slab for given query.
  - LCA query + range top-2 query to find slab boundaries.
- Use slab's range selection data structure to answer query.
  - Slab's points are numbered $1..O(\kappa)$, input query and answer are in $1..n$.
    1. Map query range to range among slab's points.
    2. Convert answer back to "global" coordinates.
- Storing global coordinates of points in a slab takes $O(\kappa \log n)$ bits per slab or $O(n \log n)$ bits overall.



Representing Shallow Cutting

We need a representation of slabs which can space-efficiently:

- in $O(\lg \kappa / \lg \lg n)$ time, perform predecessor search for l and r among x coordinates in a slab.
- in $O(1)$ time, retrieve the i-th largest x-coordinate in the slab (access query).

Theorem [Navarro et al. FSTTCS'14]

There is an encoding using $O(n \lg \kappa)$ bits of space and supports range selection in $O(\lg k / \lg \lg n)$ time.

Result uses novel tree partitioning scheme and uses succinct DS™ technology extensively. Previous result [CW SODA'13]:

- $O(n \lg \kappa + \underbrace{n \lg \lg n + (n \lg n)/\kappa}_{\text{non-optimal terms}})$ bits space and $O(\lg k / \lg \lg n)$ time.

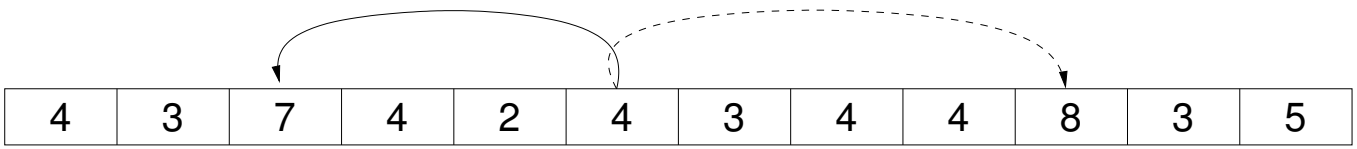
Encoding Nearest Larger Values (NLV)

- Given $A[1..n]$, encode A to answer the query:

  NLV(i): return j s.t. $A[j] > A[i]$ and $|j - i|$ is minimized (tie?).

  NLRV(i): return $j > i$ s.t. $A[j] > A[i]$ and $j - i$ is minimized.

- Non-distinct values in A.



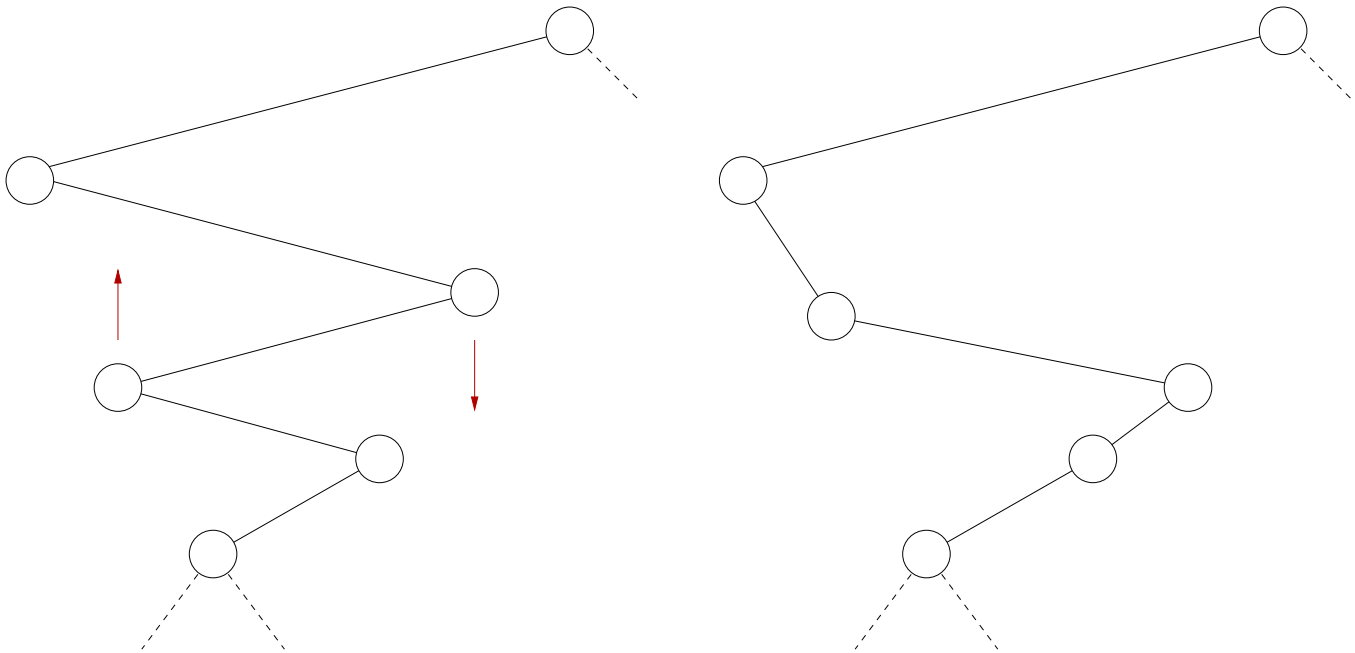
$$\text{NLV}(6) = 3$$
$$\text{NLRV}(6) = 10$$

- NLV problem related to RMQ problem.
- NLRV is basically balanced parenthesis — $2n - O(\log n)$ bits [Jayapaul et al. IWOCA'14]
- Encoding NLV in $< 1.8n$ bits, *if* tie-break arbitrary. [Nicholson, R.]

Unidirectional NLVs

Given $A[1..n]$, encode A to answer:

NLV(i): return i such that $A[j] > A[i]$ and $|j - i|$ is minimized.

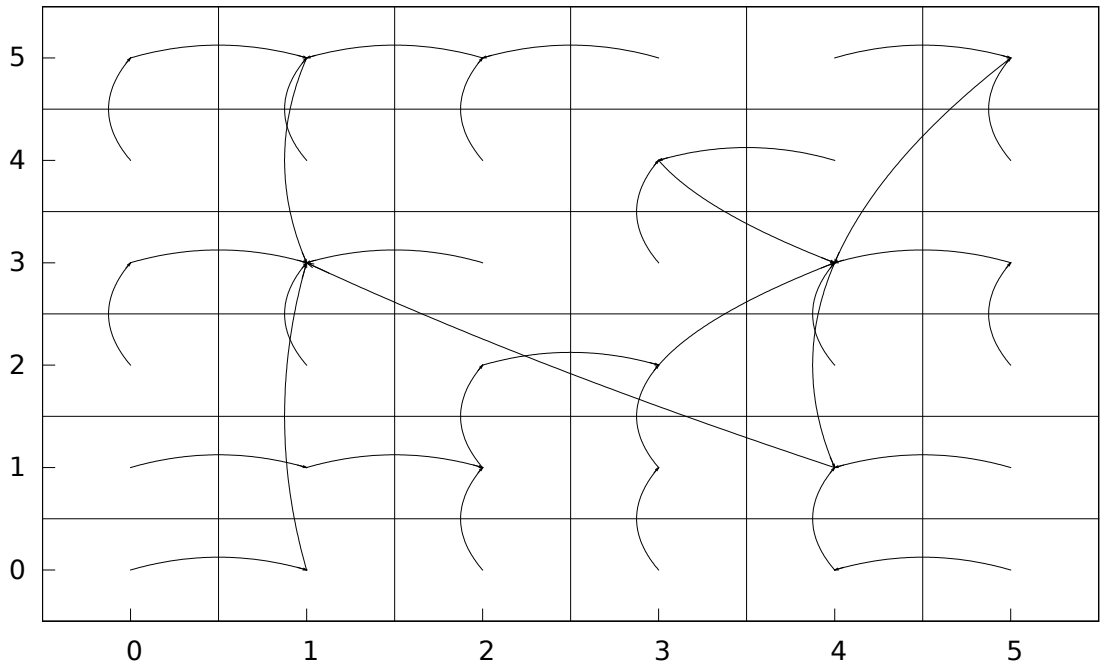


- Modify paths of degree-1 nodes in Cartesian tree.
- Upper bound $< 1.8n$ bits.
- Hard to count—sequence not in `oeis.org`.

2D NLV

- Given a 2D matrix A, and a query point $p = (i, j)$, find $q = (i', j')$ s.t. $A[q] > A[p]$ and $|p - q|_1 = |i - i'| + |j - j'|$ is minimized.
- Considered by [Jaypaul et al. IWOCA'14]:
  - $O(n^2)$-bit encoding if all items in A distinct.
  - $O(n^2 \lg \lg n)$ bit encoding in general case.
- $O(n^2)$-bit encoding, $O(1)$ time queries, in general case [Jo et al. WALCOM'15].
- Encoding 2D-RMQ requires $\Omega(n^2 \lg n)$ bits [Demaine et al. ICALP'09].

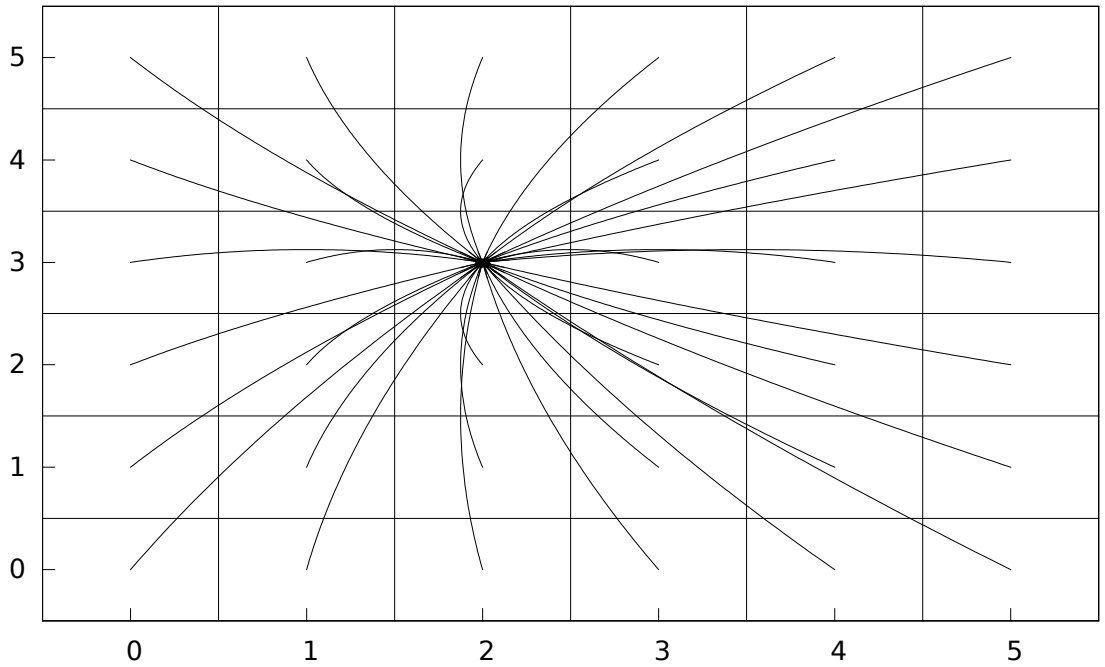
Encoding 2D NLVs



When elements of A are distinct, can explicitly store pointers using VLC (pointer of length i takes $O(\lg i)$ bits), this takes $O(n^2)$ bits.

Pictures © Pat Nicholson

Encoding 2D NLVs



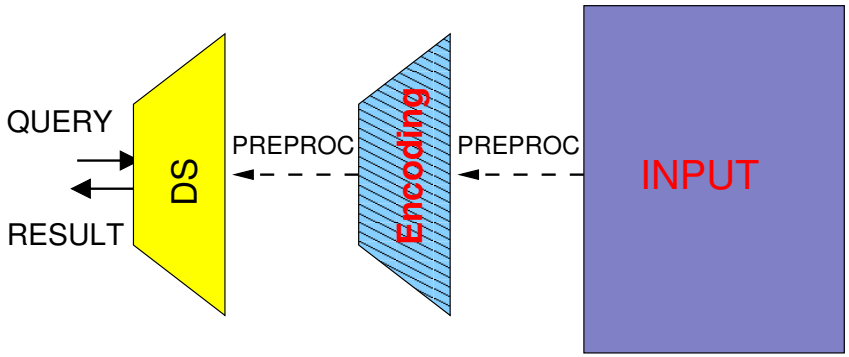
Can't do this when elements of A are non-distinct, requires $\Omega(n^2 \log n)$ bits.

[Jo et al. '15] Point to next equal value in row/column.

Pictures © Pat Nicholson

Minimal Encodings

1. Pre-process given data to obtain encoding E, discard input.
2. E should precisely characterize the query – # distinct Es should equal # distinguishable data instances using the query ($|\mathcal{C}|$).
3. Create succinct DS on E, using $\log|\mathcal{C}|(1+o(1))$ bits. Second pre-processing should not access input.



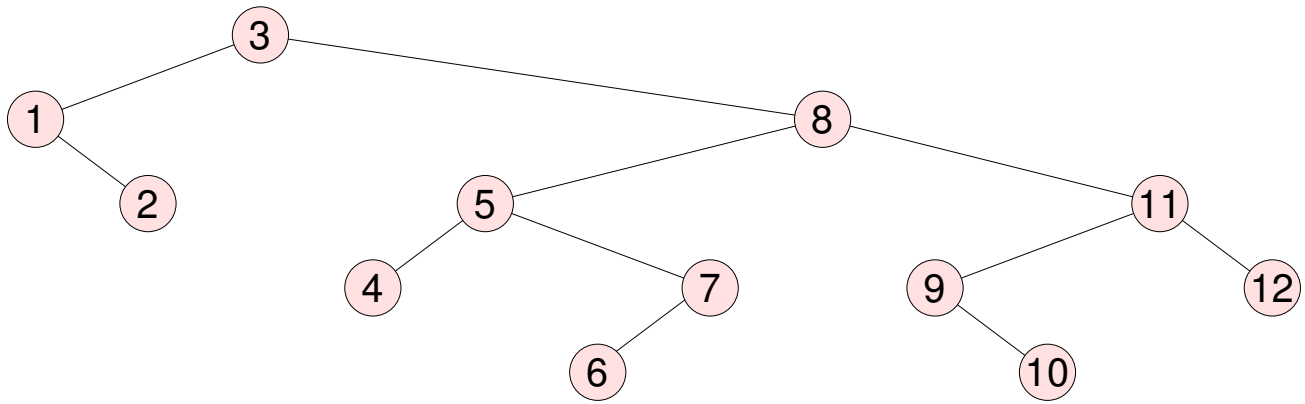
Advantages

- Optimal space.
- Only information in DS is what can be obtained from queries.
  - "Minimal-knowledge" data structures: contain only information strictly necessary to answer queries.

Exact Encodings for RMQ

Given $A[1..n]$, preprocess to answer:

RMQ(l, r) : return $\arg\max_{l \le i \le r} A[i]$.

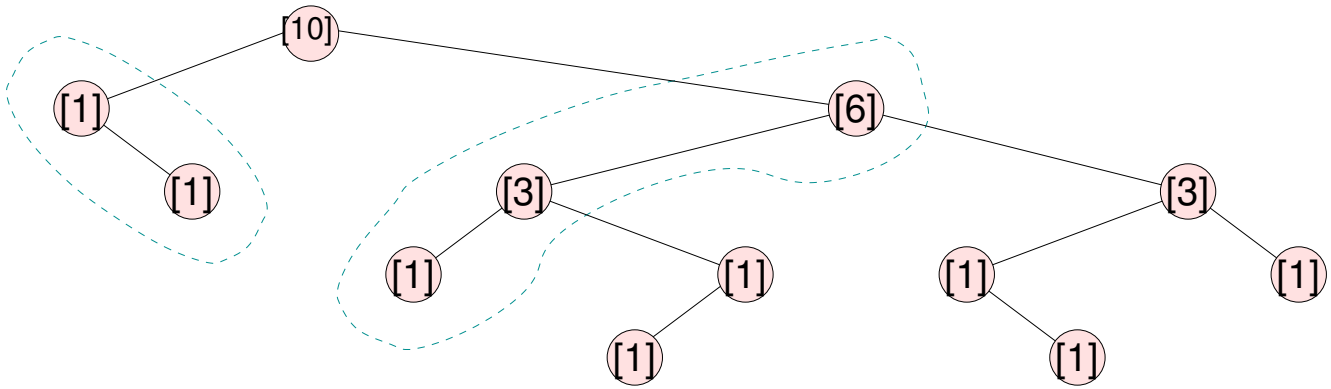


- Shape of Cartesian tree precisely describes all possible RMQs. [Fischer, Heun, SICOMP'11].
  - Pre-process A, output Cartesian tree, delete A.
- There are $\frac{1}{n+1}\binom{2n}{n}$ distinct binary trees on n nodes.
  - Create succinct Cartesian tree DS using $2n + o(n)$ bits.
  - Cartesian trees for uniformly random A in $1.74n$ or $1.92n$ bits. [Golin et al. TCS][Davoodi et al. PTRS'14].

Exact Encodings for R2MQ

Given $A[1..n]$, encode A to answer:

R2MQ(l, r): return $\arg\max_{i \in \{l,\ldots,r\} - RMQ(l,r)} A[i]$.

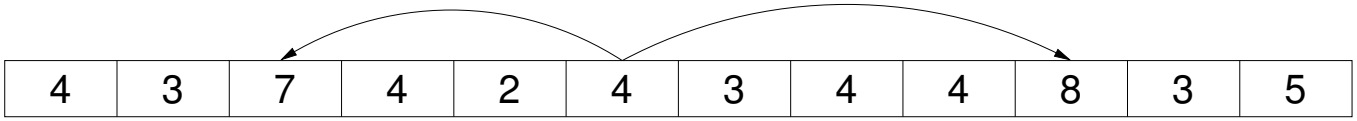


- Need to *merge* inner spines of Cartesian tree.
- Precisely described by "extended Cartesian tree".
- Counting them is hard. Somewhere between $2.65n$ and $(2 + \lg(1 + \sqrt{2}))n < 3.25n$ bits.
- Asymptotically shown to be $\sim 2.76n$ bits [Gawrychowski and Nicholson,'14].

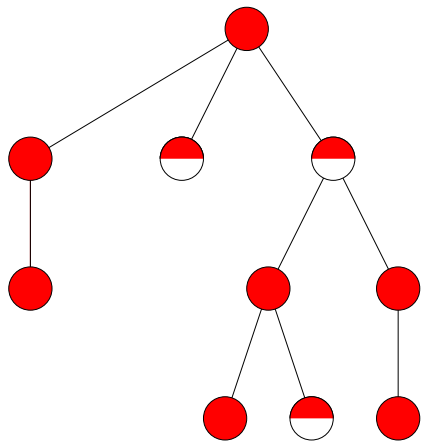
Minimal for Bidirectional NLV Problem

Given $A[1..n]$, encode A to answer:

BNLV(i): return $j > i$ such that $A[j] > A[i]$ and $j - i$ is minimized, and $j' < i$ such that $A[j'] > A[i]$ and $i - j'$ is minimized.



- Described by a subclass of *Schröder trees*.
- Number of n-node Schröder trees is $\leq (3 + 2\sqrt{2})^n$.
- Encoding using $\leq \lg(3 + 2\sqrt{2})n < 2.54n$ bits.



Minimal Encodings for Range Min-Max Queries

Given $A[1..n]$, encode A to answer:

Range-Min-Max(l, r): return both $\arg\max_{i \in \{l,\ldots,r\}} A[i]$ and $\arg\min_{i \in \{l,\ldots,r\}} A[i]$.

New paper by [Nicholson and Gawrychowski,'14]:

- Precisely (?) characterized by *Baxter* permutations.
  - Do not exist $1 \leq l < i < r \leq n$ such that:

$$\pi(i+1) < \pi(l) < \pi(r) < \pi(i)$$

or

$$\pi(i) < \pi(r) < \pi(l) < \pi(i+1)$$

  - If A is a Baxter permutation, it can be recovered using Range-Min-Max queries.
- Number of Baxter permutations on $[n] = 2^{3n}/n^{O(1)}$, gives $3n - O(\log n)$ encoding size.

Conclusions and Open Problems

Conclusions:

- Introduced the notion of encoding DS.
- Minimal encodings are combinatorially interesting and have good privacy properties.

Wide range of open problems:

- Challenging data structuring open problems:
  - Asymptotically optimal 2D RMQ encoding of [Brodal et al. ESA'13] does not support efficient 2D RMQ queries.
  - Minimal top-k encoding of [Gawrychowski and Nicholson] does not support queries.
- Determining minimal encodings for a number of problems.
- Pre-processing time — ideally want $O(n)$ time preprocessing.
- Apply encoding DS to reducing the space usage of "normal" DS. [cf. Chan and Wilkinson, SODA'13]