Compressed Range-Minimum Queries

Average-Case Analysis of Search Trees Meets Space-Efficient Data Structures

Sebastian Wild
wild@uwaterloo.ca

based on joint work with
Ian Munro & Patrick Nicholson

2019-06-24

My goals (for you!)

1. Showcase new **applications** of average-case **analysis**

   You know analysis.

   → You will like this.

2. Give flavor of supporting **queries** on **compressed data**

   - very rough ideas & black boxes
   - bit fiddling
   - I will treat $n \cdot \frac{\log \log n}{\log n}$ as "much smaller" than n ...

→ tougher ride, but cool stuff

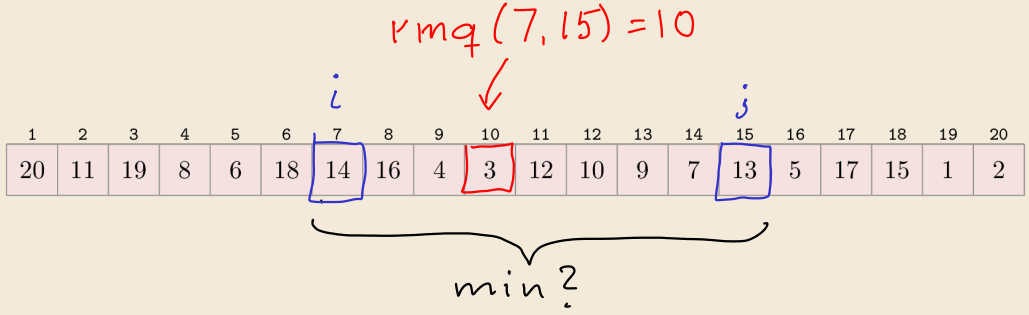
Overview

1. Range minimum queries & Cartesian trees
2. Random RMQ & random BSTs
3. Tree covering
4. Entropy trees

1 Range-Minimum Queries

The RMQ problem:

Given: array $A[1..n]$ of numbers



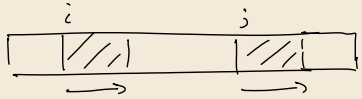
Task: ① Preprocess A

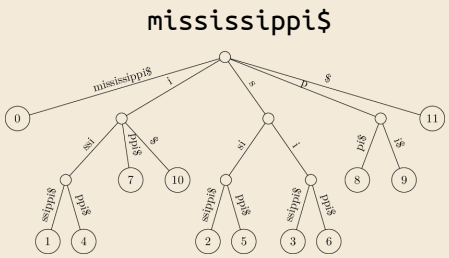
② Answer $rmq(i,j)$ = position of minimum in $A[i..j]$

$$= \arg\min_{i \le k \le j} A[k]$$

Applications of RMQ

longest common extensions in strings





all suffixes
mississippi$
ississippi$
ssissippi$
sissippi$
issippi$
ssippi$
sippi$
ippi$
ppi$
pi$
i$
$

sort

longest common prefix
$
i$
ippi$
issippi$
ississippi$
mississippi$
pi$
ppi$
sippi$
sissippi$
ssissippi$
ssippi$

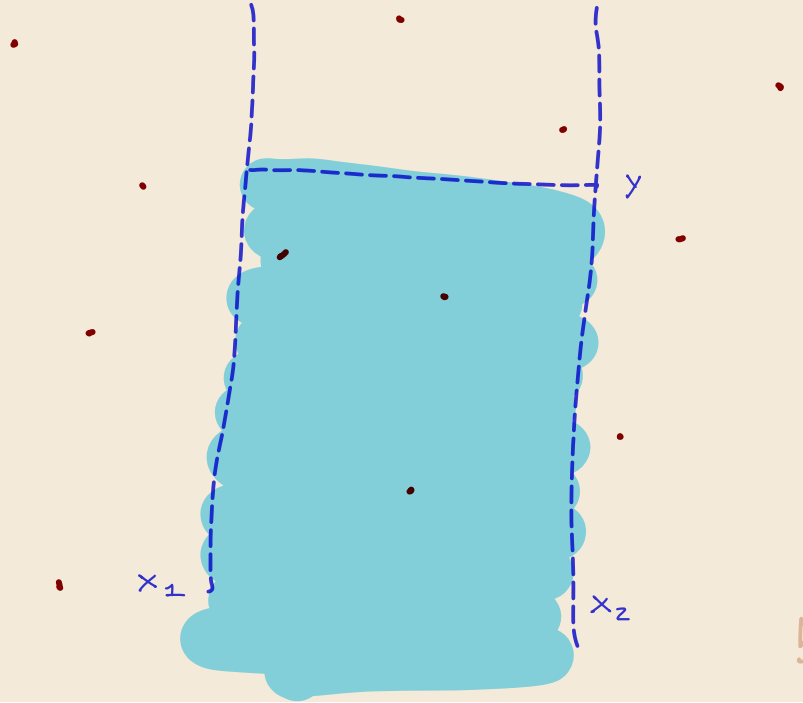
suffix array
12
11
8
5
2
1
10
9
7
4
3
6

longest common extension of two positions

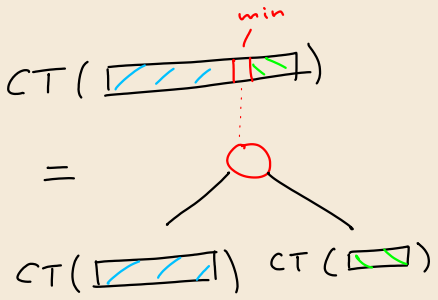
$lce(i,j) = LCP[\,rmq_{LCP}(i+1, j)\,]$

$LCP[i] = lcp$ (ith and (i-1)st suffix)

3-sided orthogonal range queries in 2D

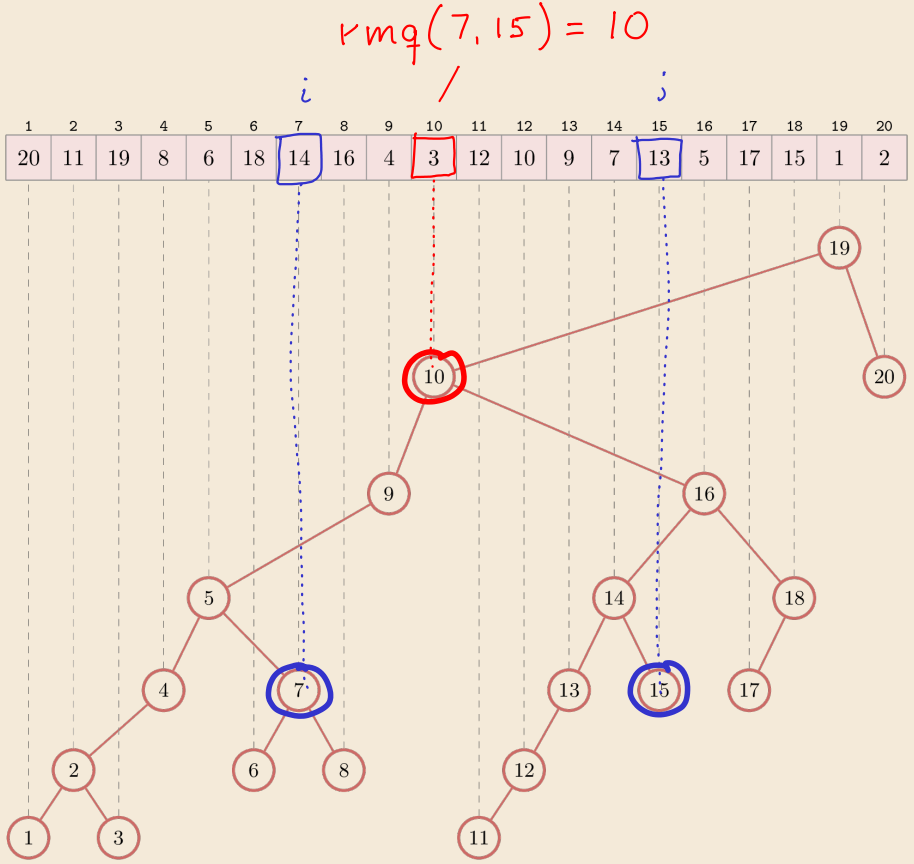


Cartesian Trees



rmq(i, j) =

lowest common ancestor (lca)
of nodes with inorder i and j

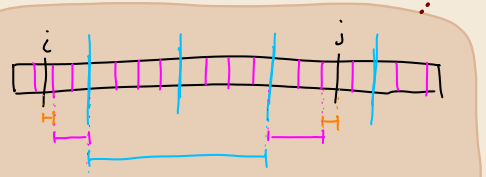


Elementary Solutions

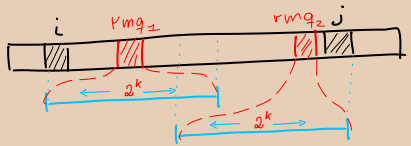
query time

trivial: store nothing
scan A[i..j] O(n)

⚠ still need original array A ⇒ must store n integers



blocking, "4 Russians" & sparse tables
O(1) Θ(n)



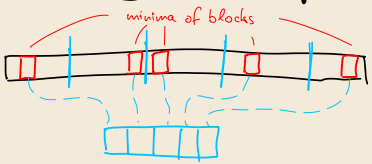
sparse tables
O(1) Θ(n log n)

trivial: precompute all
O(1) Θ(n²)

space usage (integers)

Details on O(n) Space RMQ

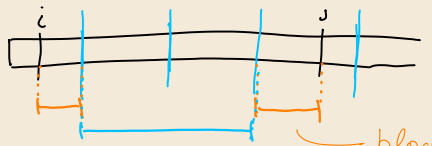
① Blocking / Input Reduction



reduced instance $O(\frac{n}{\log^3 n})$

↳ use $O(n \log^2 n)$ bit solution: (sparse tables)

$O(\frac{n}{\log^3 n} \cdot \log^2(\frac{n}{\log^3 n}) = O(\frac{n}{\log n})$ bits



lookup answer

② 4-Russians base case

block-local queries: $\leq 4^{\frac{1}{4}\lg n} = \sqrt{n}$ different RMQ answer set

Cartesian trees

↳ store
- pre-computed answers for all possible blocks $O(n)$
- the type of each block (using $2 \cdot \frac{1}{4}\lg n$ bits each) $\frac{n}{\frac{1}{4}\lg n} \cdot 2 \cdot \frac{1}{4}\lg n = 2n$

③ Bootstrapping

① needs block size $w(\log^2 n)$

② needs block size $< \frac{1}{2}\lg n$

⇒ 2-level blocking

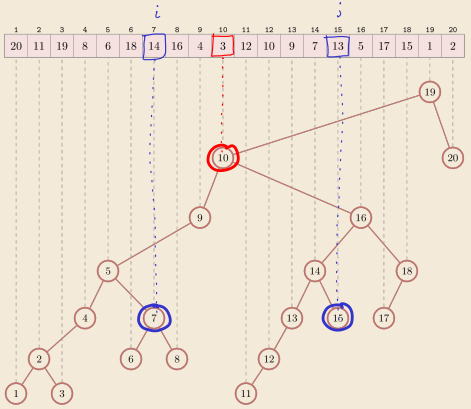
middle level: $\frac{n}{\log^3 n}$ blocks each size $\frac{\log^3 n}{\frac{1}{4}\lg n}$

$O(\frac{n}{\log^3 n} \cdot \log^2 n \log(\log^2 n)) = O(n \frac{\log \log n}{\log n})$

Succinct RMQ

goal: minimal-space (asymptotically) data structure with O(1)-time rmq (on word-RAM)

no access to A at query time (encoding model)



⇝ can use any data structure for **binary trees** with

- access to nodes by **inorder** index
- **lca** of 2 nodes

} O(1)

- $2n + o(n)$ bits of space optimal ($\lg(C_n) = 2n - \Theta(\log n)$)

- level-order unary degree sequence (LOUDS)
- balanced parentheses (BP)
- depth-first unary degree sequence (DFUDS)
- tree covering (TC)

from ordinal to binary ⇝ Davoodi, Raman, Rao 2017

Excursion: Machine Models

logarithmic-cost model

think: Turing machine

numbers $\gg 2^{64}$

machine fixed, unbounded input

vs

?

vs

vs

unit-cost model

#arithmetic operations

64 bit numbers

fixed-size input

word-RAM model

transdichotomous model

constant-time ops on w bit numbers

$w = \Theta(1)$

w grows with n

the machine grows with the input!

$w \geq$ size of numbers in input

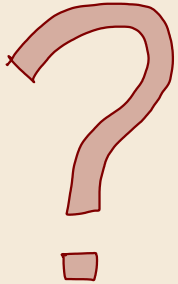
$w = \Theta(\log n)$

standard model in algorithm theory

less powerfull

more powerfull

2 Random RMQ



Can we save space for "typical" RMQ instances?

without compromising query time

↳ on average

natural model: all orderings equally likely

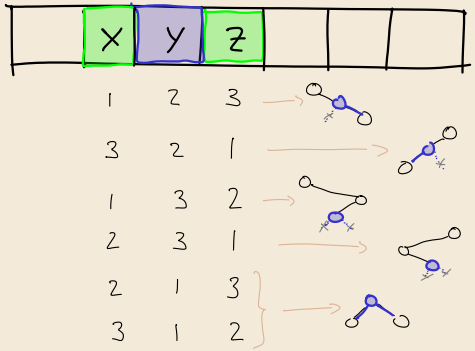
random-permutations model

↳ equivalent: A[i] iid real random variables

Previous Work

- local order / node degrees

  Davoodi, Navarro, Raman, Rao 2014



$H(\frac{1}{6}, \frac{1}{6}, \frac{1}{3}, \frac{1}{3}) \approx 1.918$

→ solves RMQ with average space $1.918n + o(n)$

Can we do better?

- compressible array (empirical entropy)

  Fischer, Heun 2011

- few runs in array

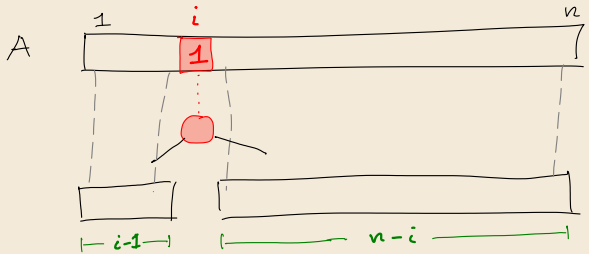
  Barbay, Fischer, Navarro 2012

} won't help on random arrays

RMQ = BST !

CT for random permutation:

position of global min $\overset{\mathcal{D}}{=}$ Uniform [1..n]



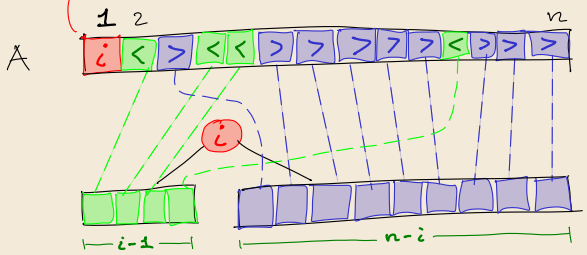
same recursively

$CT(A) = BST(A^{-1})$ — inverse function

same shape distribution for random perm.

BST built by insertions from random permutation

first inserted value $\overset{\mathcal{D}}{=}$ Uniform [1..n]



same recursively

Structural Complexity of Random Binary Trees
J. C. Kieffer, E-H. Yang, and W. Szpankowski

entropy ~ 1.736n bits

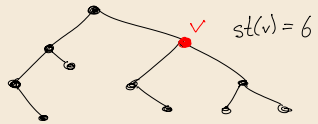
Subtree-Size Entropy

$$\Pr[T=t] = \prod_{v \in t} \frac{1}{st(v)}$$

prob that among all values in v's subtree, v is inserted first

Notation:

- t binary tree (shape)
- v node in t
- $st(v)$ size of v's subtree
- T random BST (shape)



$$\text{entropy of } T\text{'s distribution} = \sum_t \Pr[T=t] \cdot \lg \frac{1}{\Pr[T=t]}$$

$$= \text{---''---} \lg \left(\prod_{v \in t} st(v) \right)$$

$$= \text{---''---} \sum_{v \in t} \lg(st(v)) =: H_{st}(t) \quad \text{"subtree-size entropy"}$$

$$= \mathbb{E}\left[H_{st}(T) \right].$$

If we can encode trees using $H_{st}(t) + l.o.t.$ bits, we use optimal average space ! → matches entropy

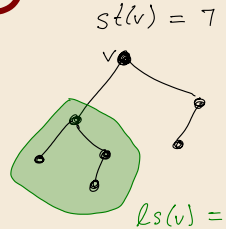
Remark:
subtree sizes are **global** property
previous methods exploit **local** properties

Left-subtree-size encoding

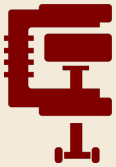
Idea: store the sizes of all left subtrees

for node v : $ls(v) \in [0 .. st(v)-1]$

↖ uniform dist. in random BSTs!
↳ local entropy $\lg(st(v))$



$ls(v) = 4 \in [0 .. st(v)-1]$



encode nodes in **preorder** traversal ⇒ $st(v)$ already encoded / known

store $ls(v)$ with **arithmetic coding**

⇝ compression scheme with $\mathcal{H}_{st}(t) + 2$ bits

How to access specific nodes?
How to compute lca?

Bounding the worst case

⚠ $\mathcal{H}_{st}$ can be $\sim n \lg n$

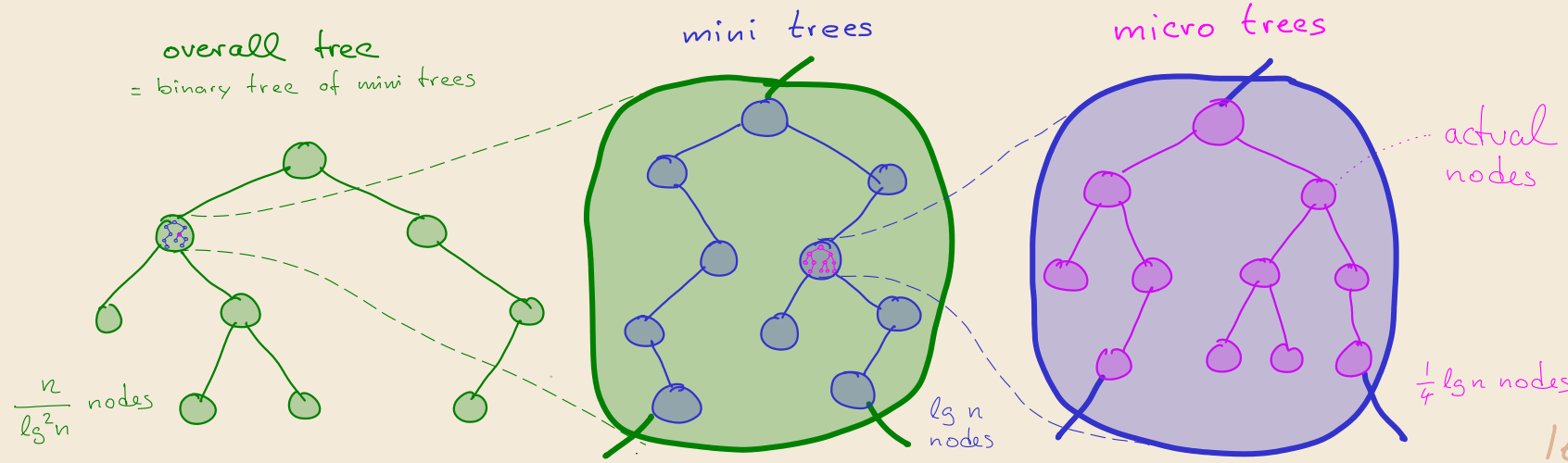
use 1 bit to specify:
0 ⇝ above encoding
1 ⇝ standard $2n$ bit encoding

$\min\{\mathcal{H}_{st} + 3, 2n + 1\}$ bit

3 Tree Covering

technique for succinct tree data structure
using 3-level hierarchical decomposition

Geary, Raman, Raman 2006
Farzan, Munro 2012



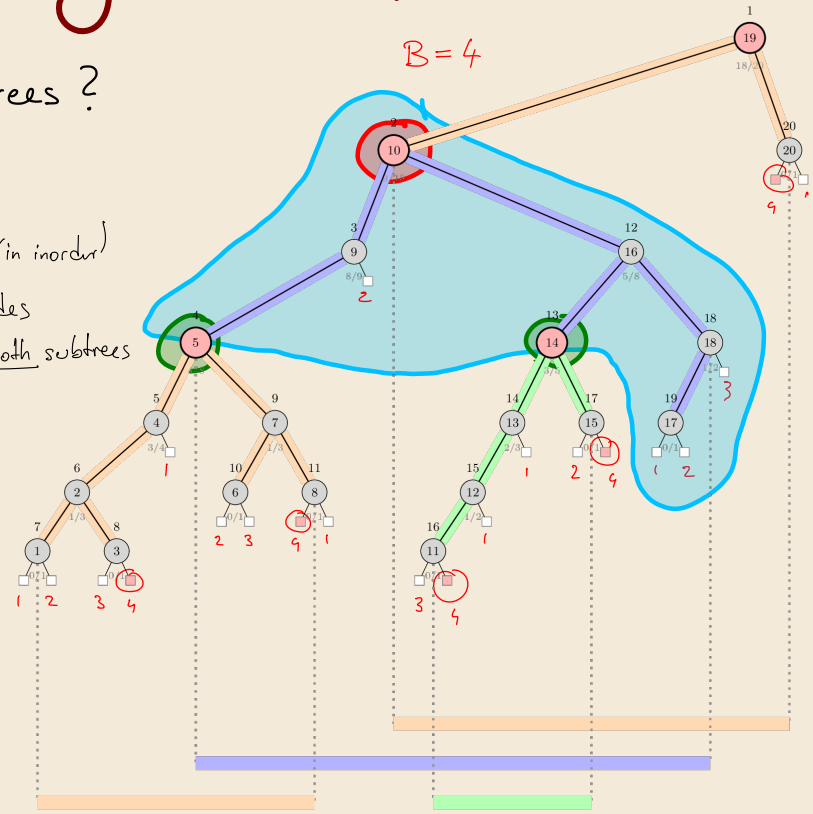
Partitioning Binary Trees

extended binary tree

How to define subtrees?

fix parameter B

1. mark every Bth leaf (in inorder)
2. mark all internal nodes with marked leaf in both subtrees (+ always mark root)
3. marked nodes = roots of blocks



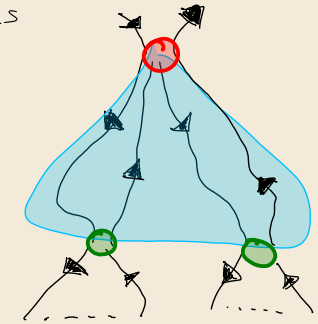
Properties:

(a) $\approx \frac{n}{B}$ subtrees

(b) between 2 and 3B nodes

(c) complete subtree except for two portal nodes

(d) traversals visit each subtree ≤ 3 times



Operations on TC representation

node ids "τ-names"

(τ_1, τ_2, τ_3)

τ_1: mini tree id; τ_2: micro tree id; τ_3: offset in micro tree

Geary, Raman, Raman 2006

node id $\leftrightarrow$ (global) preorder

Geary, Raman, Raman 2006

node id $\leftrightarrow$ (global) inorder

Davoodi, Navar ro, Raman, Rao 2014

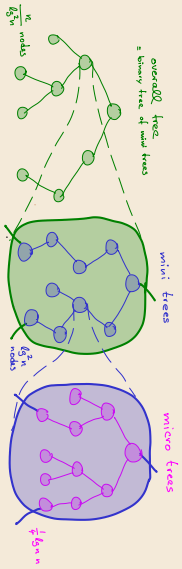
lowest common ancestor

He, Munro, Rao 2012

simplifications

Farzan, Munro 2012

all operations possible in $O(1)$ time w/ $o(n)$ indices



$O\left(\frac{n}{\log^2 n}\right)$ nodes

↳ can afford pointer/word based representation — $O(n \log n)$ bit

$O\left(\frac{n}{\log^2 n}\right)$ trees of $O(\log n)$ nodes each — $o(n)$

↳ as above!

$\leq \frac{1}{4}\lg n$ nodes each $\rightsquigarrow$ encode in $\leq \frac{1}{2}\lg n$ bit each

↳ use lookup table by type for all local operations

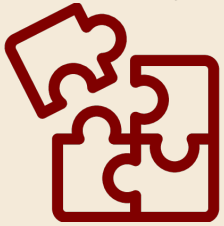
↳ only $2^{\frac{1}{2}\lg n} = \sqrt{n}$ different types

$\oplus$ store type for all microtrees ← dominant space

4 Entropy Trees

use left-subtree-size encoding for micro trees

+ compressed bitvector to find starting position of ith entry



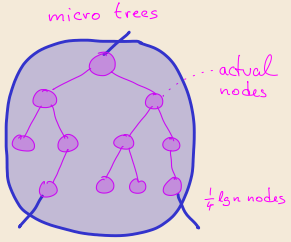
$$\sum_{\text{micro tree } \mu} |\text{type}(\mu)| \leq \sum_{\mu} \left(\sum_{v \in \mu} st_{\mu}(v) + 2 \right) \leq \sum_{\mu,\, v \in \mu} st(v) + o(n)$$

$$\leq H_{st}(t) + o(n).$$

Theorem: We can support all TC operations on static binary trees in constant time using $\min\{H_{st}, 2n\} + o(n)$ bits.

Corollary: There is a RMQ datastructure with constant query time that occupies the **optimal** $1.736n + o(n)$ bits on **average**.

Hypersuccinct Trees



dominant space: type for all microtrees

construction works with **any** (bounded) **code** for micro trees

↝ why not use an **instance-optimal** one?

Huffman code for micro-tree types

↳ never worse, but unclear how good → analysis?

Conclusion

a new AofA playground ?

1. Average-case analysis helps for **average-succinct data structures**.

   Range-min queries is one example. There are more!

   └ optimal avg space ✓

2. Known data-structures techniques work for compressed **trees**.

   How about other **combinatorial structures** ? Other input **distributions** ?

3. Theoretically optimal data structures are often **impractical**.

   Better trade-offs ? **Simpler** solutions ?