# Tight Bounds for the Partial-Sums Problem

**Mihai Pătraşcu**
**(presenting)**

**Erik Demaine**

**MIT CSAIL**

# The Problem

Maintain A[1..n] under:

| | |
|---|---|
| update(k, Δ) | modify A[k] = A[k] + Δ |
| sum(k) | report A[1] + … + A[k] |
| select(x) | return k such that sum(k) ≤ σ < sum(k+1) |

# Motivation

- range query (1D)
- many applications in the literature
  - list indexing, dynamic ranking [Dietz 89]
  - dynamic arrays [Raman et al 2001]
  - arithmetic coding [Fenwick 94]
  - …
- playground for lower bound techniques (many results)

**Restricted models:** group, semigroup

- can only use algebraic operations as black box

**General models:** RAM, cell probe

- integers
- word size: $b \geq \lg n$
- update parameter $\Delta$ limited to $\delta$ bits ($\delta \leq b$)

  natural in applications

  all previous studies considered this

# Results, old and new

| Model | Upper Bounds | Lower Bounds |
|---|---|---|
| semigroup | $O(\lg n)$ | $\Omega(\lg n / \lg\lg n)$ [Yao85]<br>$\Omega(\lg n)$ [HF98] |
| group | $O(\lg n)$ | $\Omega(\lg n / \lg\lg n)$ [FS89]<br>$\Omega(\lg n)$<br>NEW |
| RAM<br>cell probe | $O(\lg n / \lg\lg n)$<br>for $\delta = O(\lg\lg n)$ [Die89]<br>$O(\lg n / \lg (b / \delta))$ NEW | $\Omega(\lg n / \lg b)$<br>[FS89]<br>$\Omega(\lg n / \lg (b / \delta))$<br>NEW |
| other: bit-probe model, dynamic word problems | | |

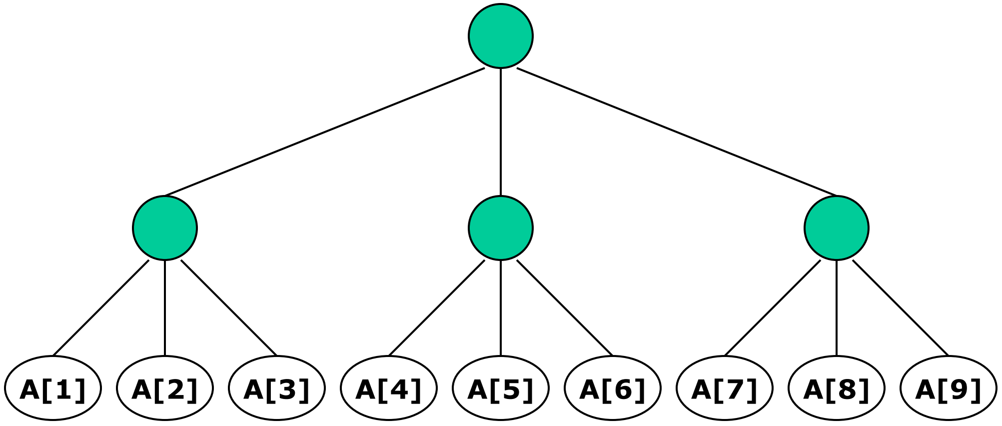
Upper bounds don't use precomputed table

standard operations (multiplication, shifts, bitwise)

New, powerful lower bound technique

# Upper bounds: The Big Picture

Build a tree with branching factor $B \approx b/\delta$
Handle all operations in $O(1)$ for arrays of size $B$
➔ $O(\lg n / \lg B)$ running times



## Upper Bounds: The Small Picture

- even with small δ, partial sums can get large after many updates
- break each sum in two components: S[i] = B[i] + C[i]
  - B[i] holds value of S[i] from some past moment
  - C[i] holds more recent changes
- after a few updates, B[i] is rebuilt (constant time, amortized)
- C[i] must remain small after few updates
  - ➔ hold packed in a word and use multiplication tricks to update

## Select

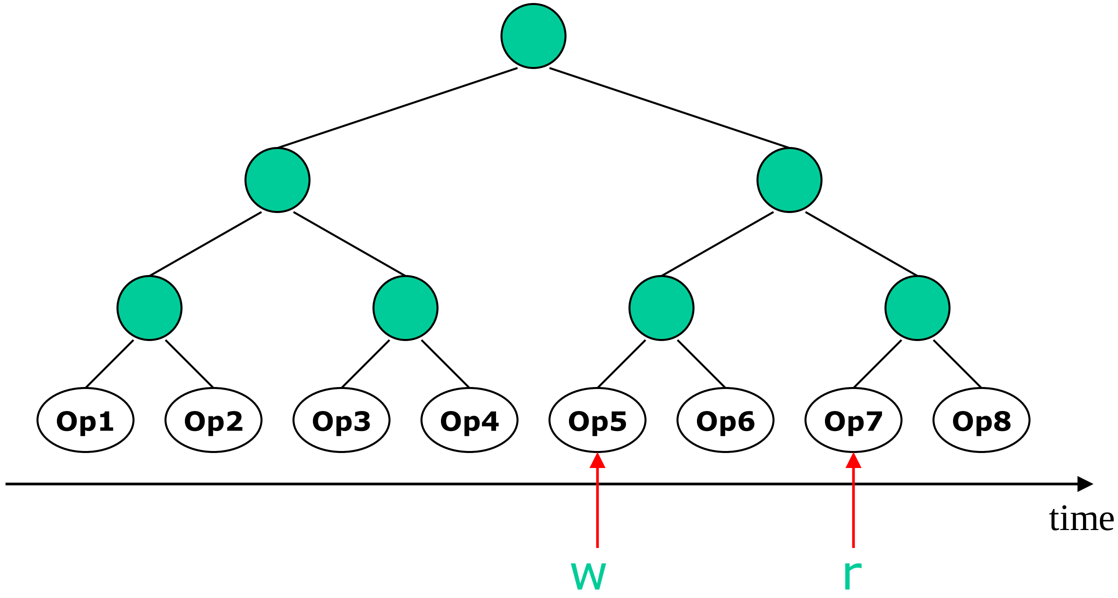
- break into runs of sums, separated by big gaps
- use the fusion structure [FW93] to select among runs
  - big gaps ➔ infrequent updating
- sums inside each run are close ➔ delta-encode relative to head
  - can pack in a word, and use more bit tricks

# Lower Bounds: Trees Again?

- bound only read instructions
- build a tree over the sequence of operations (i.e. update, sum)
- each read instruction is characterized by:

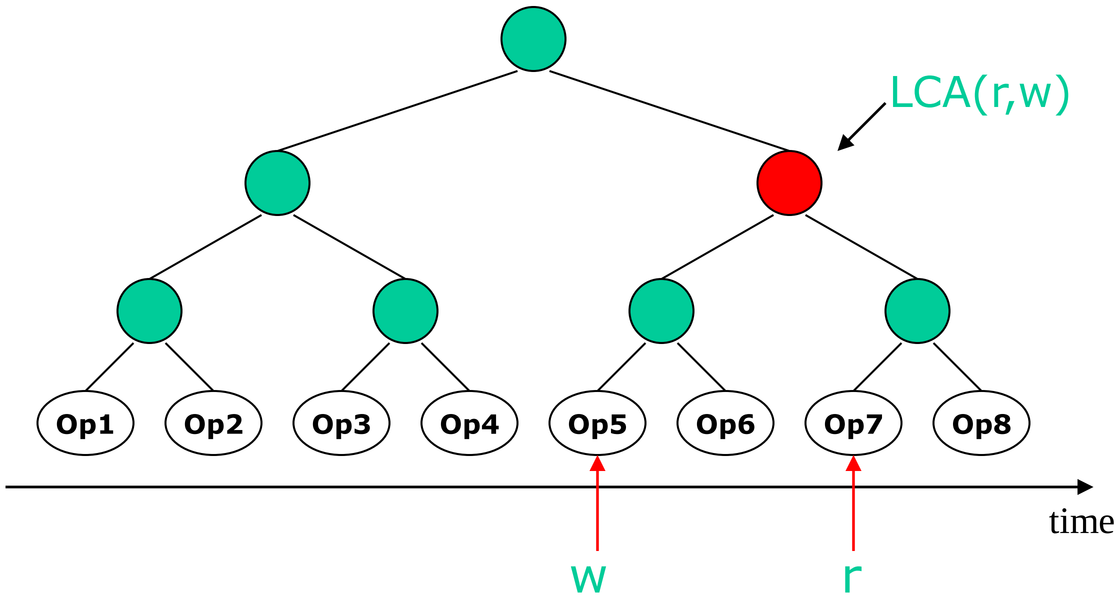
  read time r: the read happens while handling operation r

  write time w: the cell was last written while handling operation w



# Lower Bounds: Node Complexity

- associate each read with LCA(r, w)
- prove average case lower bound for each node
- sum up lower bounds
  - ➔ not double-counting
  - ➔ holds in average case



# Lower Bounds: Encoding

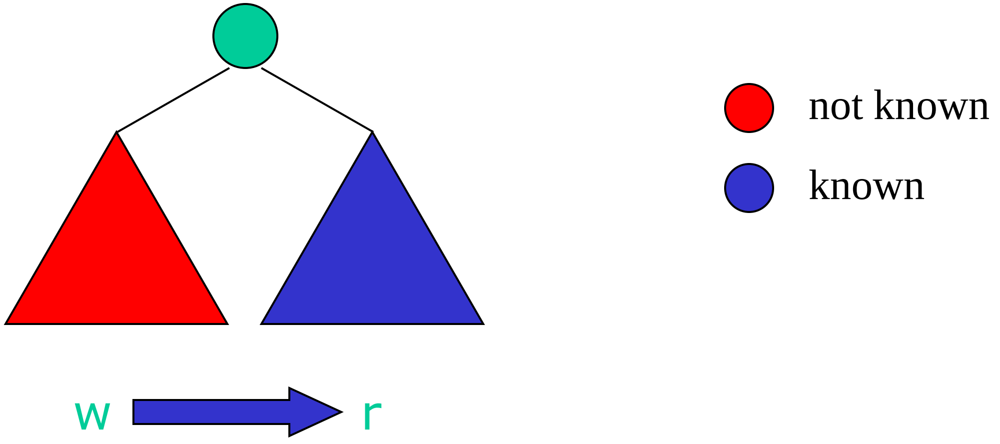
To prove lower bound for one node, consider scenario:

- you don't know anything about the left subtree
- but you know all reads with r in the right subtree, and w in the left subtree

➔ can simulate data structure, get output to queries from right subtree

- output encodes a lot of info about left subtree

➔ many read instructions



# Lower Bounds: Last Slide

- idea works well for $\delta = b$
- problem for smaller $\delta$ :

  one word (or one read instruction) can contain a lot of information
- solution:

  future request are unpredictable

  ➔ unlikely that one read instruction helps future queries

  [ ~ round elimination lemma in communication complexity ]

## Open Problems / Recent progress

SOLVED: lower bound for select

problem: output of query encodes little information

SOLVED: lower bounds on tradeoff between update & query times

SOLVED: technique applied to other problems (dynamic connectivity)

These are harder. Require: smart encoding schemes, more probabilities

OPEN: offline problem (likely very hard)

OPEN: bit-probe model

problem: cell addresses (not cell contents) make encoding large

# The End