Dynamic Lower Bounds

Mihai Pătrașcu
What can binary trees do?

• maintain aggregates over a static structure

  Maintain an array A[n] under:
  
  - `update(i, Δ): A[i] += Δ`
  - `sum(i): return A[0] + ... + A[i]`

• insert and delete nodes

  Maintain a list L under:
  
  - insert/delete nodes
  - `select(k): return kth node in list`

• split/concateenate

  Maintain a collection of lists under:
  
  - split / concatenate
  - `list(v): return the head of v’s list`
What can binary trees do?

- "partial sums" \( t_u = t_q = O(\lg n) \)
  
  Maintain an array \( A[n] \) under:
  
  \[
  \begin{align*}
  \text{update}(i, \Delta) & : A[i] += \Delta \\
  \text{sum}(i) & : \text{return } A[0] + \ldots + A[i]
  \end{align*}
  \]

- "dynamic rank/select" \( t_u = t_q = O(\lg n / \lg \lg n) \)
  
  Maintain a list \( L \) under:
  
  \[
  \begin{align*}
  \text{insert/delete nodes} \\
  \text{select}(k) & : \text{return } k\text{th node in list}
  \end{align*}
  \]

- "dynamic connectivity" (in trees) \( t_u = t_q = O(\lg n) \)
  
  Maintain a collection of lists under:
  
  \[
  \begin{align*}
  \text{split / concatenate} \\
  \text{list}(v) : \text{return the head of } v\text{'s list}
  \end{align*}
  \]
Lower Bounds

[Fredman, Saks’89] all 3 problems
if \( t_u = \lg^{O(1)} n \), then \( t_q = \Omega(\lg n / \lg \lg n) \)

[P., Demaine’04] partial sums, dynamic connectivity
\[ \max\{t_u, t_q\} = \Omega(\lg n) \]
if one operation takes \( O(\log_B n) \), the other is \( \Omega(B \lg n) \)
Lower Bounds

[Fredman, Saks’89] all 3 problems

Fast update: $t_u = O(\log_B n)$
Slow query: $t_q = O(B \log_B n)$

if one operation takes $O(\log_B n)$, the other is $\Omega(B \lg n)$

update
Lower Bounds

Fredman, Saks’89+ all 3 problems

if $t_u = \lg O(n)$, then $t_q = \Omega(\lg n / \lg \lg n)$

P., Demaine’04+ partial sums, dynamic connectivity

max\{$t_u, t_q\} = \Omega(\lg n)$

if one operation takes $O(\log_B n)$, the other is $\Omega(B \lg n)$

Slow update: $t_u = O(B \log_B n)$
Fast query: $t_q = O(\log_B n)$
Lower Bounds

[Fredman, Saks’89] all 3 problems
  If $t_u = \lg^{O(1)} n$, then $t_q = \Omega(\lg n / \lg \lg n)$

[P., Demaine’04] partial sums, dynamic connectivity
  $\max\{t_u, t_q\} = \Omega(\lg n)$
  If one operation is $O(\log_B n)$, the other is $\Omega(B \lg n)$
  \text{\textbf{Tight to partial sums}}
  When $t_u \geq t_q$, tight for dynamic connectivity

[P., Thorup’10] dynamic connectivity
  If $t_u = o(\lg n)$, then $t_q \geq n^{1-o(1)}$
Uses of binary trees

✓ partial sums
\[ \max\{t_u, t_q\} = B \log_B n \quad \min\{t_u, t_q\} = \log_B n \]

✓ dynamic connectivity
\[ t_u = o(\lg n) \quad \Rightarrow \quad t_q \approx \text{linear scan} \]
\[ t_u = B \log_B n \quad \Rightarrow \quad t_q = \log_B n \]

? dynamic ranking
\[ t_u \geq \lg n \quad \Rightarrow \quad t_q = \Theta(\lg n / \lg t_u) \]
[Chan, P.’10] \[ t_q = O(\lg n / \lg \lg n), \quad t_u = O(\lg^{0.5+\epsilon} n) \]
Union-Find

- amortized: \( t_u = t_q = \Theta(\alpha) \)
- worst case + union only on roots: \( t_u, t_q = \Theta(\lg n/\lg t_u) \)
  [Alstrup, Ben-Amram, Rauhe’99]
- worst case + general union: \( t_u, t_q = O(\lg n/\lg t_u) \)

Another application of our technique:

For general union, if \( t_u = o(\lg n/\lg \lg n) \), then \( t_q \geq n^{1-o(1)} \)

“Don’t rush into a union. Take time to find your roots!”
Lower Bounds via Epochs

epoch $i+1$: $B^{i+1}$ updates

epoch $i$: $B^i$ updates

Choose $B >> t_u$
Hard Instance

query

π

time

X
Let $W$ = “width of edges” \((W=n^{1-\varepsilon})\)

Operations:

- **update** (setting one $\pi$) \(\Rightarrow\) \(W\) union calls
- **query:**
  - color root and leaf with \(C\) colors \((C=n^\varepsilon)\) \(\Rightarrow\) \(W\) union calls
  - test consistency of coloring \(\Rightarrow 2C\) find queries
The Lower Bound

\( W = \text{width of edges} = n^{1-\varepsilon}; \quad C = \# \text{ colors} = n^\varepsilon \)

Operations:
- update: \( W \) unions
- query: \( W \) unions + 2\( C \) queries

**Theorem:** Let \( B \gg t_u \). The query needs to read \( \Omega(W) \) cells from each epoch, in expectation.

So \( W t_u + C t_q \geq W \log n / \log t_u \).

If \( t_u = o(\log n / \log \log n) \), then \( t_q \geq W/C = n^{1-2\varepsilon} \).
Hardness for One Epoch

\[ \text{time} \]

\[ O(B^{i-1}Wt_u) \text{ cells} \]
Communication

Alice: \( B^i \) permutations on [W] \( (\pi_1, \pi_2, \ldots) \)

Bob: for each \( \pi_i \) a coloring of input&output with C colors

Goal: test if all colorings are consistent
Communication (cont.)

Alice:  $B^i$ permutations on $[W]$  ($\pi_1, \pi_2, \ldots$)

Bob:    for each $\pi_i$ a coloring of input&output with $C$ colors

Goal:   test if all colorings are consistent

Lower bound: $\Omega(B^iW \lg W) = \text{highest possible}$

Proof: Encoding.

If input/output coloring of $\pi$ is consistent,
there are only $(W/C)^W$ possibilities for $\pi$.

So communication $\geq \lg(W!) - W \lg(W/C) = \Omega(W \lg W)$  $\square$
Communication (cont.)

Alice: $B^i$ permutations on $[W]$ ($\pi_1, \pi_2, \ldots$)

Bob: for each $\pi_i$ a coloring of input&output with C colors

Goal: test if all colorings are consistent

Lower bound: $\Omega(B^iW \lg W) = \text{highest possible}$

Upper bound: Alice & Bob simulate the data structure
The queries run $O(B^iW t_{q})$ cells probes.

We use $O(\lg n)$ bits per each...
Nondeterminism

\[ W = \{ \text{cells written by epoch } i \} \]
\[ R = \{ \text{cells read by the } B^i \text{ queries} \} \]

The prover sends:

- address and contents of \( W \cap R \)
  cost: \( |W \cap R| \cdot O(\lg n) \) bits

- separator between \( W \setminus R \) and \( R \setminus W \)
  cost: \( O(|W| + |R|) \) bits

**Lower bound:** \( \Omega \left( B^i W \lg W \right) \)

Since \( |W|, |R| = B^i W \cdot O(\lg n/\lg \lg n) \), separator is negligible.

So \( |W \cap R| = \Omega \left( B^i W \right) \). QED.
An aside: Dictionaries

Storing $A \subseteq U$ takes $O(A \lg(U/A))$ bits.

**But:** if each $x \in A$ has $k$ bits of data, retrieval-only dictionary takes $O(Ak)$ bits
E.g. via cuckoo hashing.

- separator between $W \setminus R$ and $R \setminus W$
  cost: $O(|W| + |R|)$ bits
Open problems

For static data structures of size $n \lg^{O(1)} n$, have $t_q = \Omega(\lg n / \lg \lg n)$
If $t_u = \lg^{O(1)} n$, should read $\Omega(\lg n / \lg \lg n)$ cells per epoch...
So we hope to get $\max\{t_u, t_q\} = \Omega(\lg^2 n / \lg^2 \lg n)$
Trouble (so far): separator too big...

OPEN: $\Omega(\lg^2 n / \lg^2 \lg n)$ even in the bit-probe model.
Open problems

Proposal for $\max\{t_u, t_q\} = \Omega(n^{\epsilon})$
How to get $\Omega(n^\epsilon)$

Conjecture: if $u \cdot X << k$, must have $X = \Omega(u^\epsilon)$

Sample application: maintain array $A[1..n]$ under updates
Query: what's the most frequent element in $A[i..j]$?

Conjecture $\Rightarrow \max\{t_u, t_q\} = \Omega(n^\epsilon)$

$$[u]\backslash S_1 ; S_1 ; \ldots ; [u]\backslash S_i ; S_i ; \ldots ; [u]\backslash S_k ; S_k ; T$$
3-Party, Number-on-Forehead

\[ S_1, \ldots, S_k \subseteq [u] \]

\[ T \subseteq [u] \]

\[ S_i \cap T? \]

- Time \( O(k \cdot u \cdot X) \)
- Time \( O(u \cdot X) \)
- Time \( O(X) \)
A bit of cheating

Conjecture: if $u \cdot X << k$, must have $X = \Omega(u^\epsilon)$

3SUM-hardness

3SUM: among $n$ numbers, do there exist $a + b + c = 0$?

Popular conjecture: $\Omega(n^2 / \text{polylog } n)$
The End