Dynamic Lower Bounds

Mihai Pătrașcu

at&t

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 **k**th node in list

- split/concatenate

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:
> update(**i**, Δ): A[**i**] += Δ
> sum(**i**): return A[0] + … + A[**i**]

- "dynamic rank/select" $t_u=t_q=O(\lg n/\lg\lg n)$

> Maintain a list **L** under:
> insert/delete nodes
> select(**k**): return **k**th node in list

- "dynamic connectivity" (in trees) $t_u=t_q=O(\lg n)$

> Maintain a collection of lists under:
> split / concatenate
> list(**v**) : return the head of **v**'s list

Lower Bounds

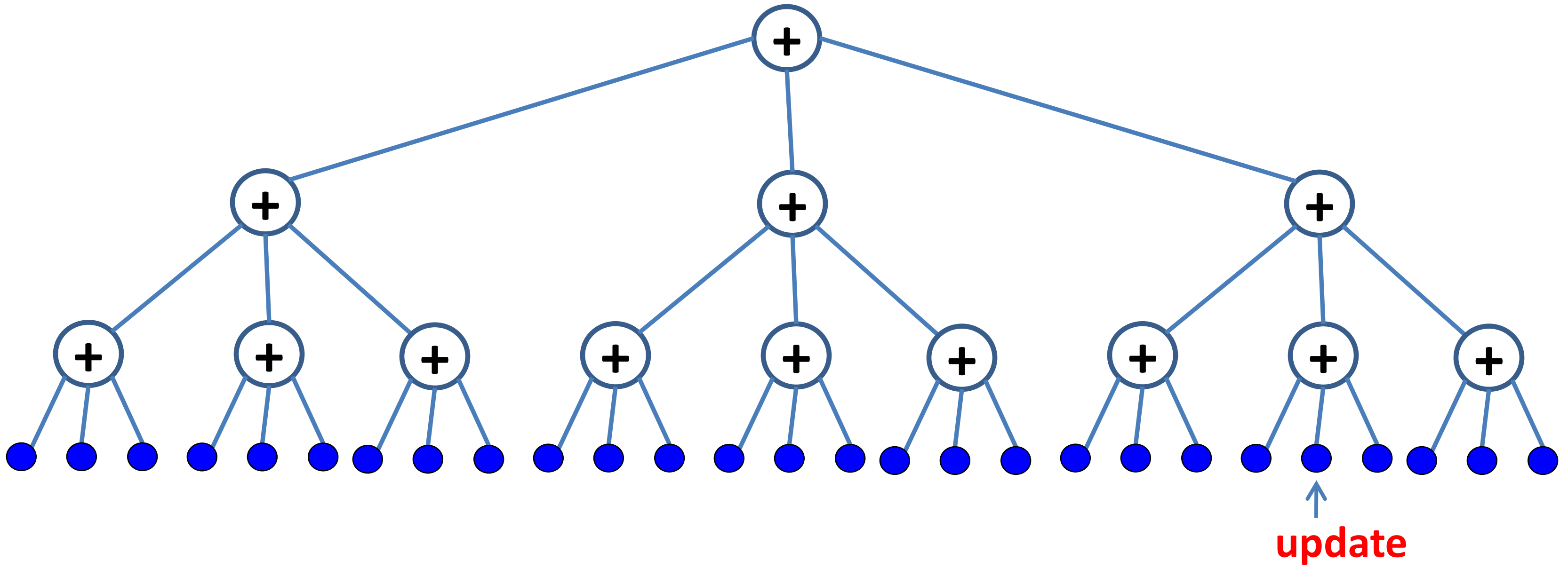
[Fredman, Saks'89] all 3 problems

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

if t

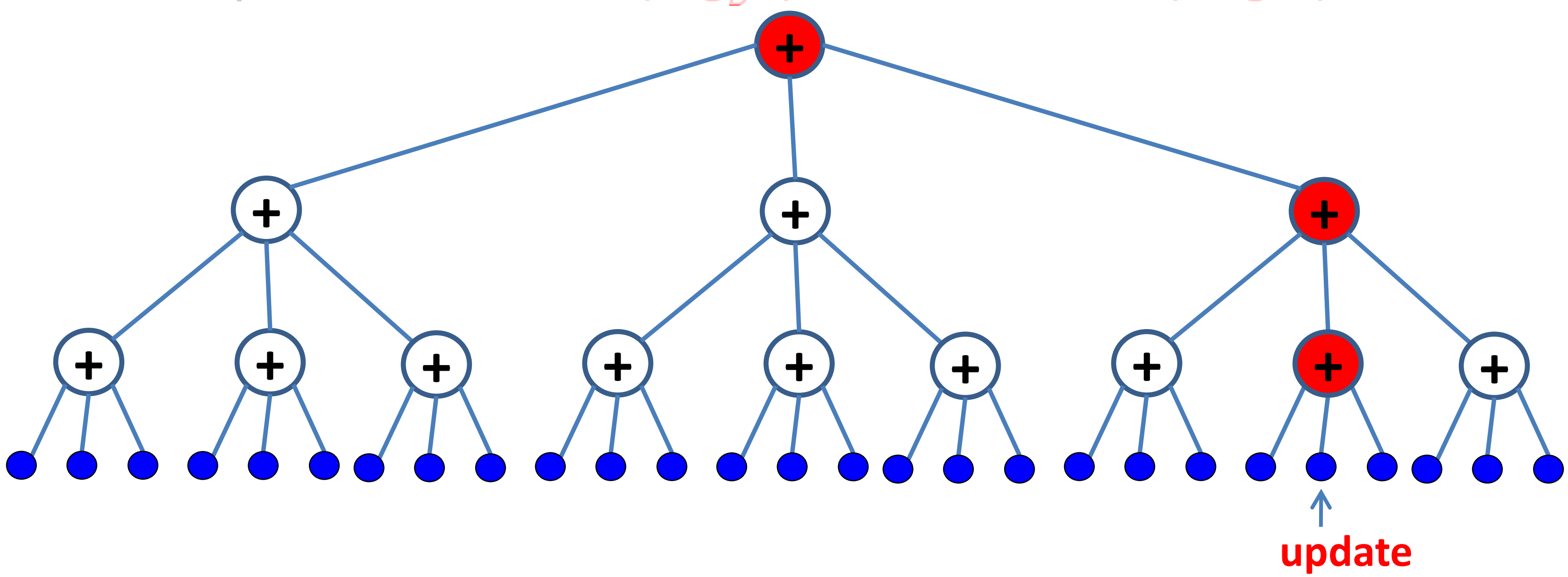
[P., De

ma

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 $\Omega(B \lg n)$



Lower Bounds

[Fredman, Saks'89] all 3 problems

if t

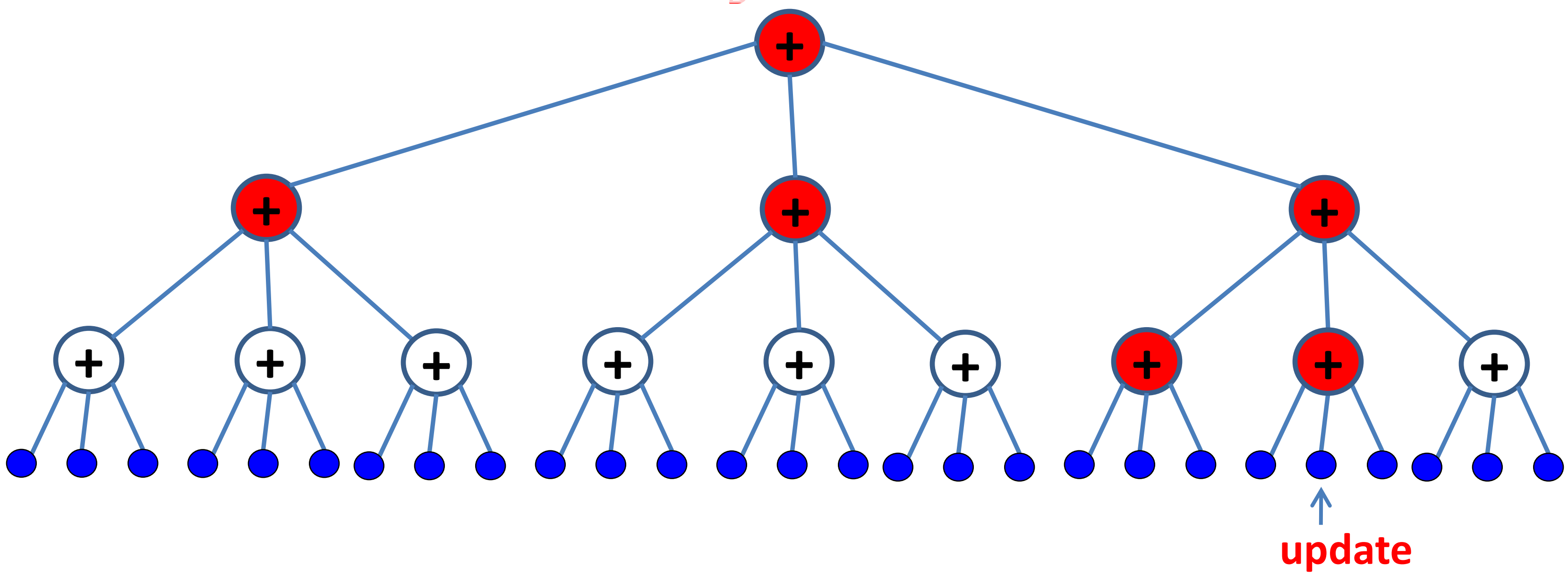
[P., D

ma

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

Fast query: $t_q = O(\log_B n)$

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



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 $\Omega(B \lg n)$

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

$\max\{t_u, t_q\} = B \log_B n$ $\min\{t_u, t_q\} = \log_B n$

✓ dynamic connectivity

$t_u = o(\lg n)$ ➔ $t_q \approx$ linear scan

$t_u = B \log_B n$ ➔ $t_q = \log_B n$

? dynamic ranking

$t_u \geq \lg n$ ➔ $t_q = \Theta(\lg n / \lg t_u)$

[Chan, P.'10] $t_q = O(\lg n / \lg\lg n)$, $t_u = O(\lg^{0.5+\varepsilon} n)$

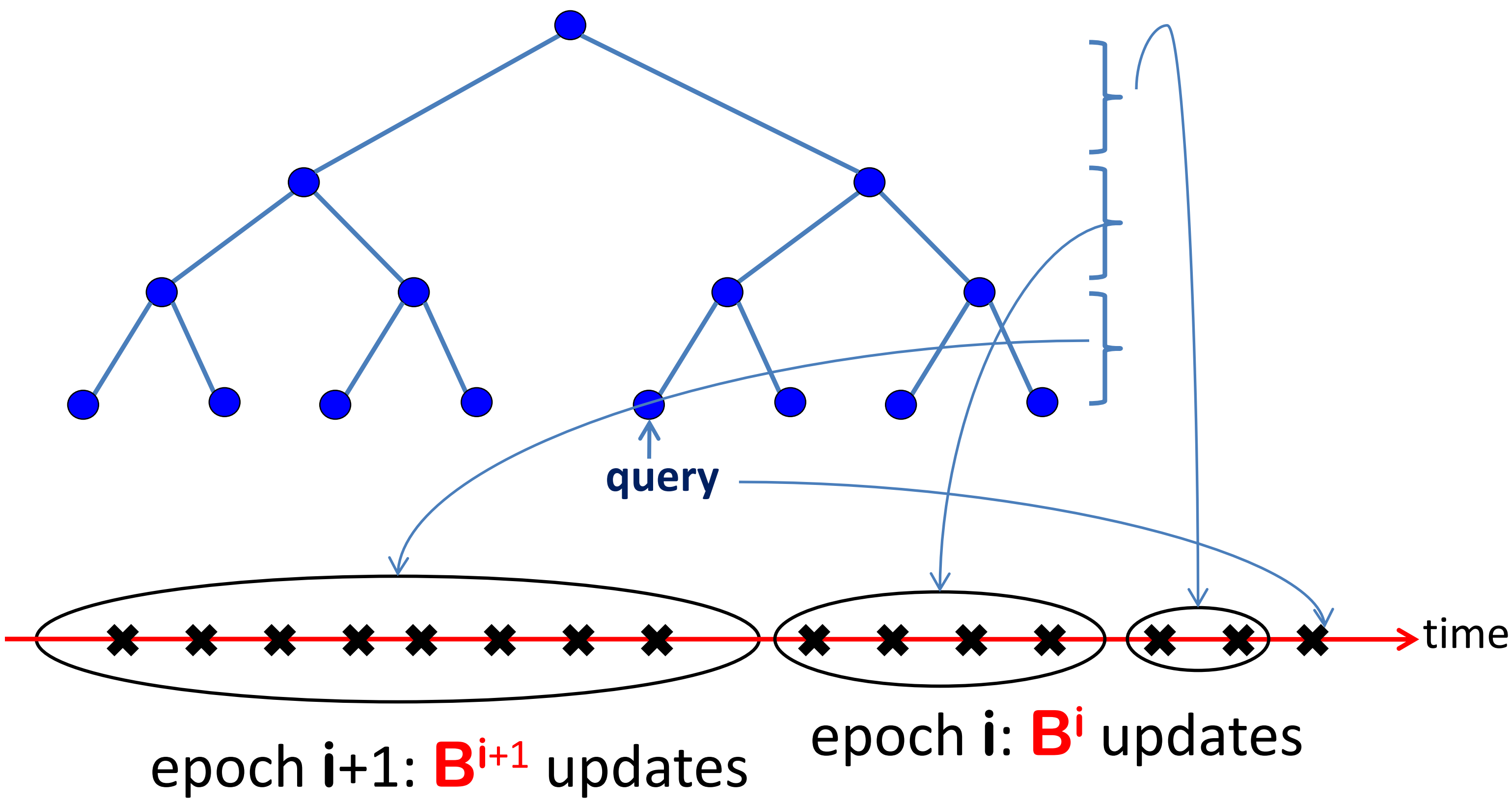
Union-Find

- 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 $t_q \geq n^{1-o(1)}$

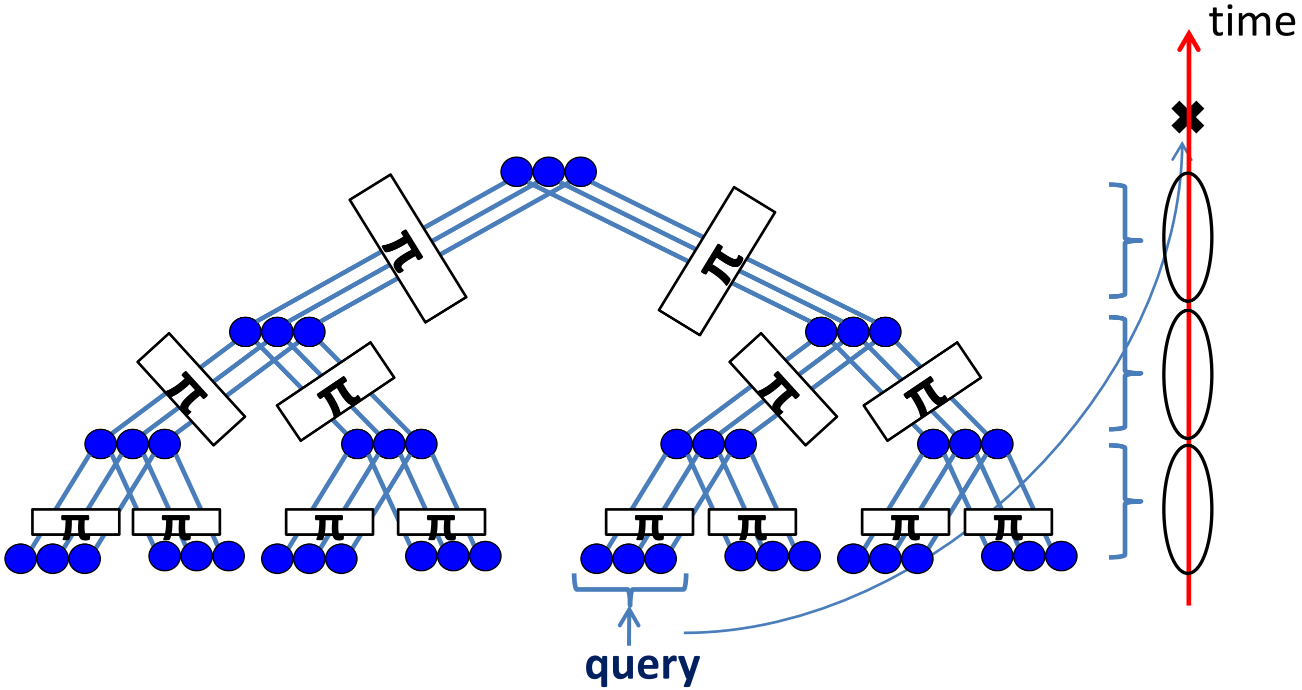
"Don't rush into a union. Take time to find your roots!"

Lower Bounds via Epochs



Choose **B >> t_u**

Hard Instance

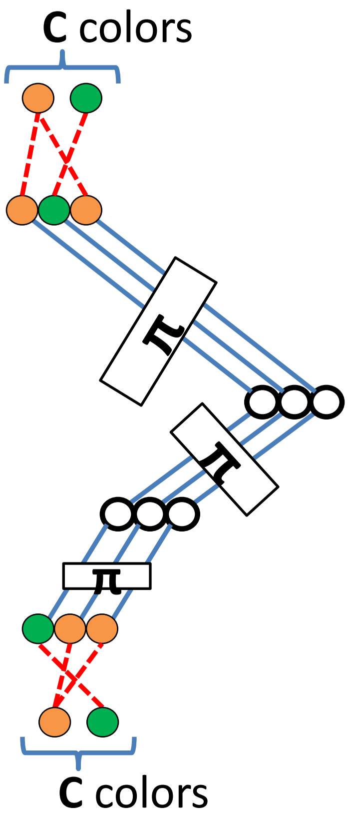


Hard Instance (cont.)

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

Operations:

- update (setting one π) → W union calls
- query:
  - color root and leaf with C colors ($C=n^{\varepsilon}$)
    → W union calls
  - test consistency of coloring
    → 2C find queries



The Lower Bound

W = width of edges = $n^{1-\varepsilon}$;   C = # colors = n^{ε}

Operations:

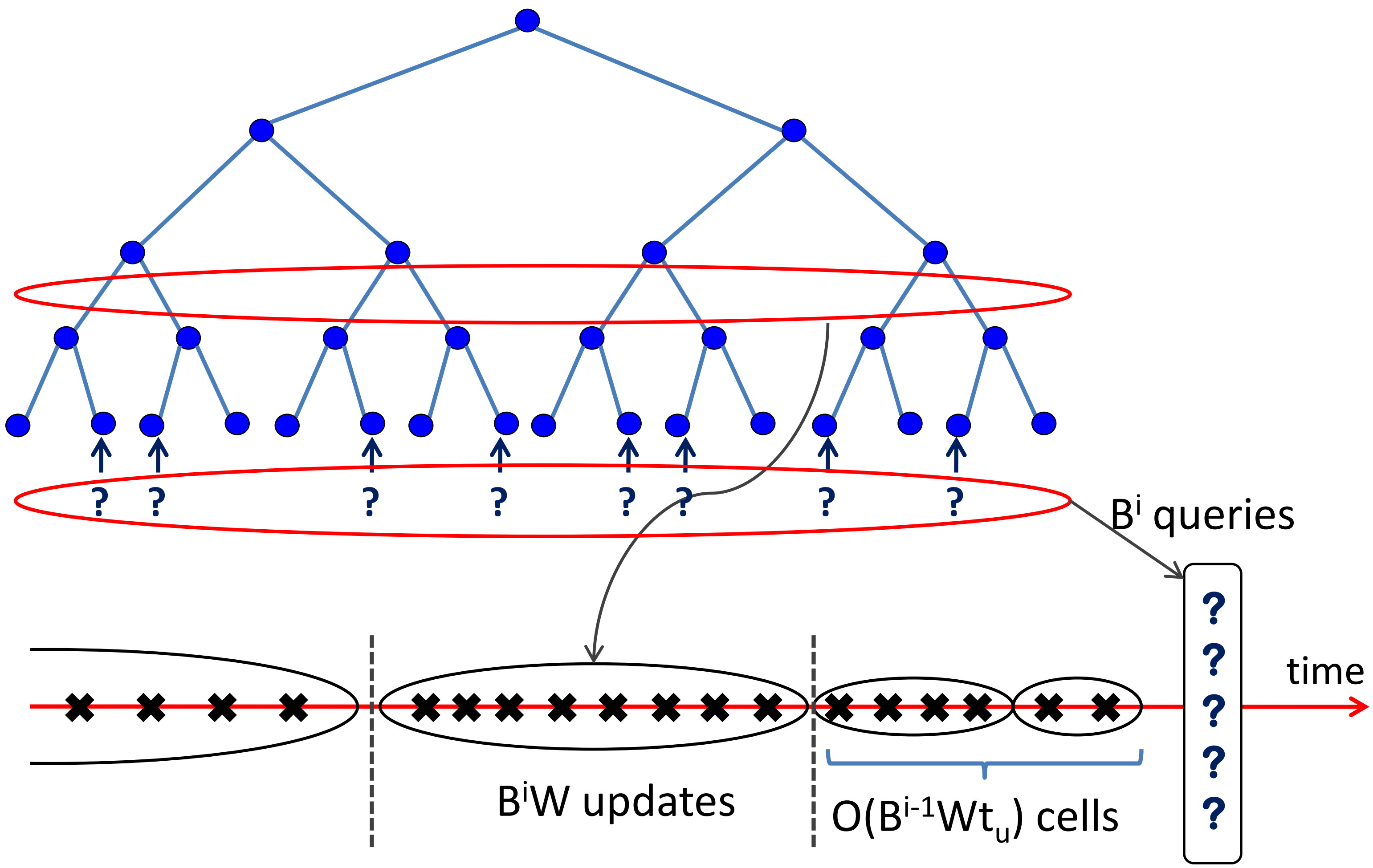
- update: W unions
- query: W unions + 2C queries

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

So $W\ t_u + C\ t_q \geq W \lg n / \lg t_u$.

If $t_u = o(\lg n / \lg\lg n)$, then $t_q \geq W/C = n^{1-2\varepsilon}$.

Hardness for One Epoch

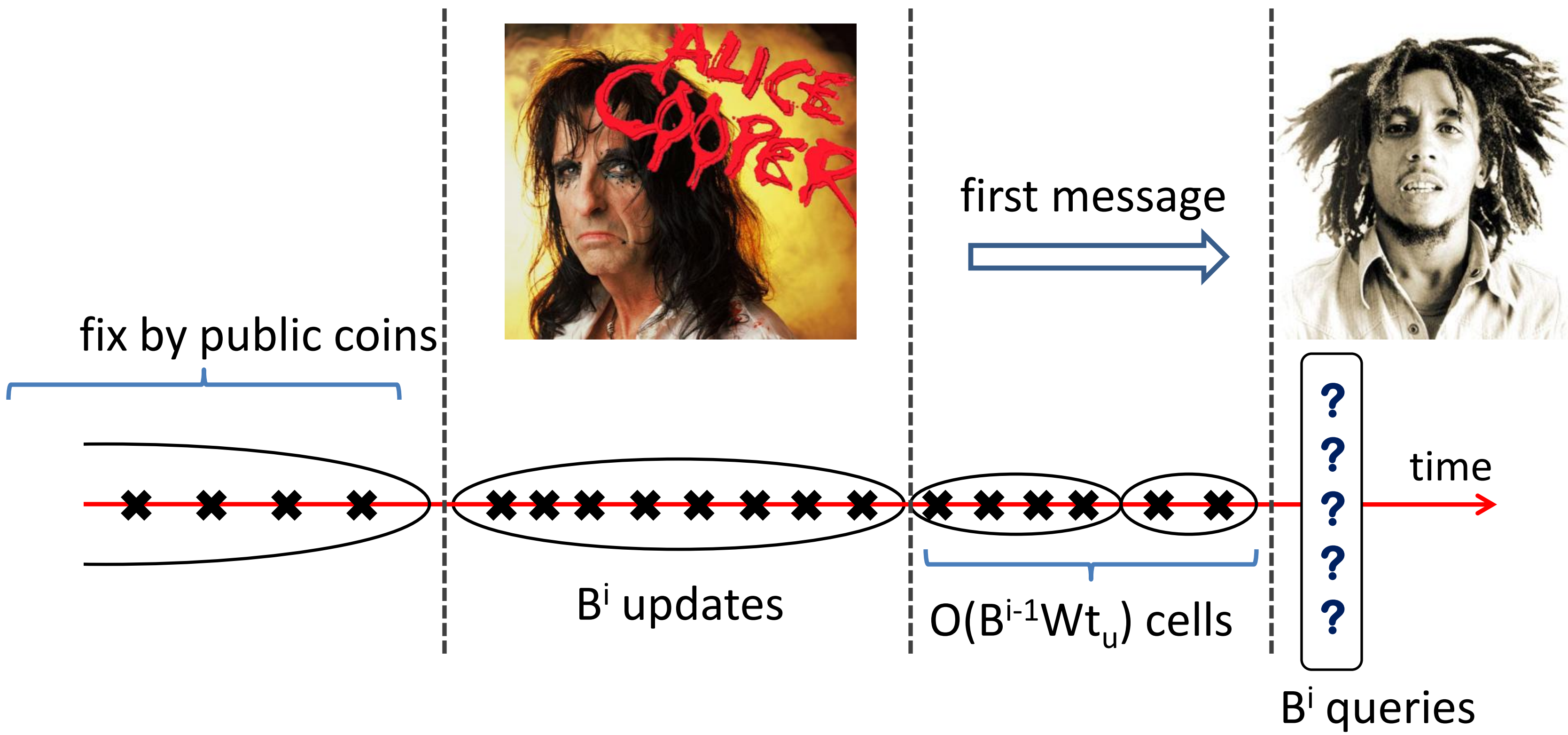


Communication

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

Bob: for each π_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 π_i a coloring of input&output with C colors

Goal: test if all colorings are consistent

Lower bound: $\Omega(B^i W \lg W)$ = highest possible

Proof: Encoding.

If input/output coloring of π is consistent,

there are only $(W/C)^W$ possibilities for π.

So communication ≥ $\lg(W!) - W \lg(W/C) = \Omega(W \lg W)$ □

Communication (cont.)

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

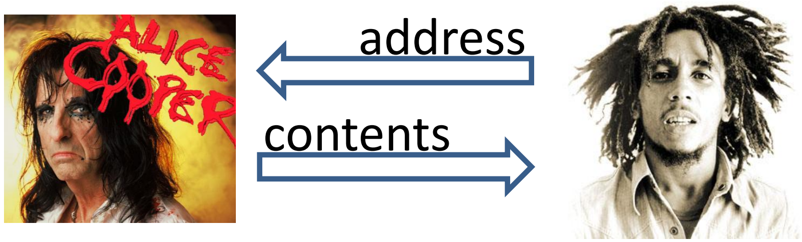
Bob: for each π_i a coloring of input&output with C colors

Goal: test if all colorings are consistent

Lower bound: $\Omega(B^i W \lg W)$ = highest possible

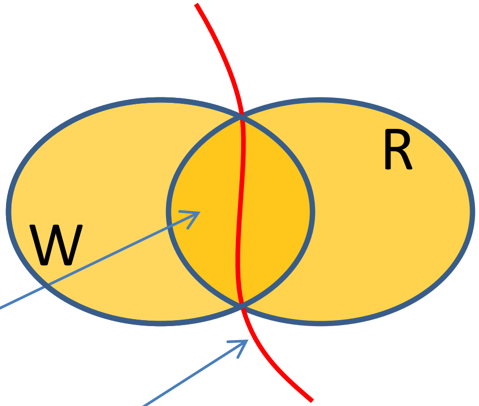
Upper bound: Alice & Bob simulate the data structure

The queries run $O(B^i W\, t_q)$ cells probes.



We use O(lg n) bits per each…

Nondeterminism



W = { cells written by epoch **i**}

R = { cells read by the B^i queries }

The prover sends:

- address and contents of W∩R
  cost: |W∩R|·O(lg n) bits
- separator between W\R and R\W
  cost: O(|W| + |R|) bits

Lower bound: $\Omega(B^i W \lg W)$

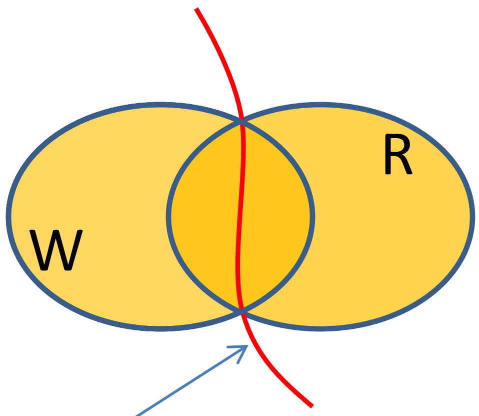
Since $|W|,|R|=B^iW \cdot O(\lg n/\lg\lg n)$, separator is negligible.

So $|W \cap R| = \Omega(B^i W)$. QED.

An aside: Dictionaries

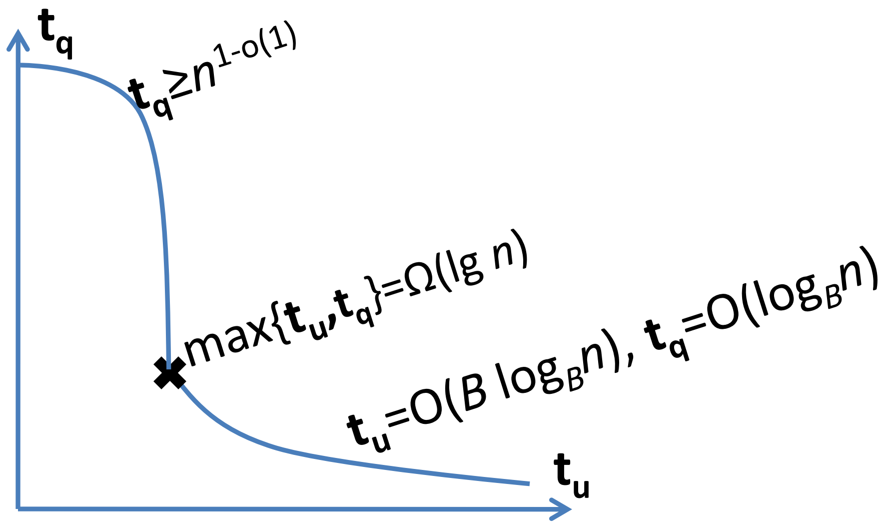
Storing $A \subseteq U$ takes $O(A \lg(U/A))$ bits.

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\R and R\W
  cost: $O(|W| + |R|)$ bits

Open problems



For static data structures of size $n \lg^{O(1)} n$, have $t_q = \Omega(\lg n / \lg\lg n)$

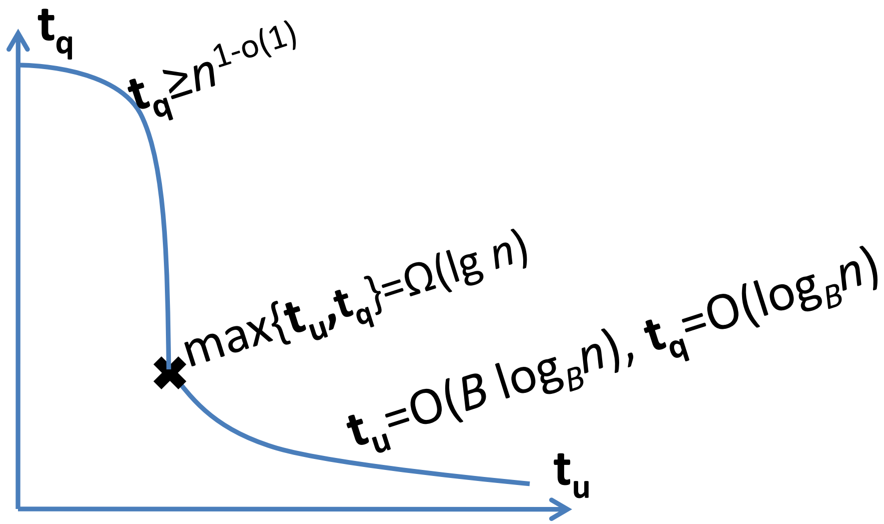
If $t_u = \lg^{O(1)} n$, should read $\Omega(\lg n/\lg\lg n)$ cells per epoch…

So we hope to get $\max\{t_u, t_q\} = \Omega(\lg^2 n / \lg^2 \lg n)$

Trouble (so far): separator too big…

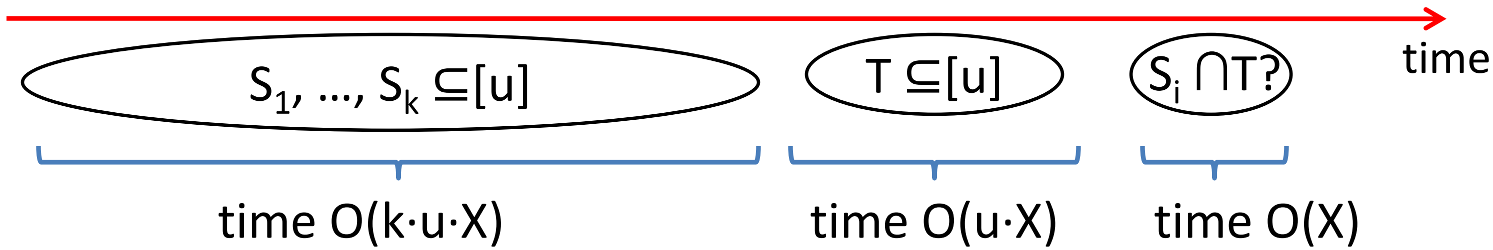
OPEN: $\Omega(\lg^2 n/\lg^2 \lg n)$ even in the bit-probe model.

Open problems



[P. STOC'10] Proposal for $\max\{t_u, t_q\} = \Omega(n^{\varepsilon})$

How to get $\Omega(n^\varepsilon)$

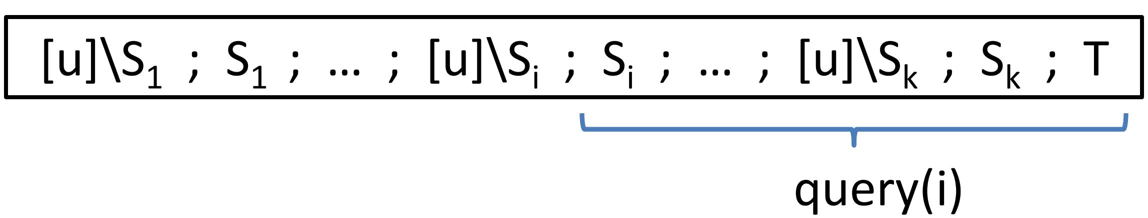


Conjecture: if $u \cdot X \ll k$, must have $X = \Omega(u^\varepsilon)$

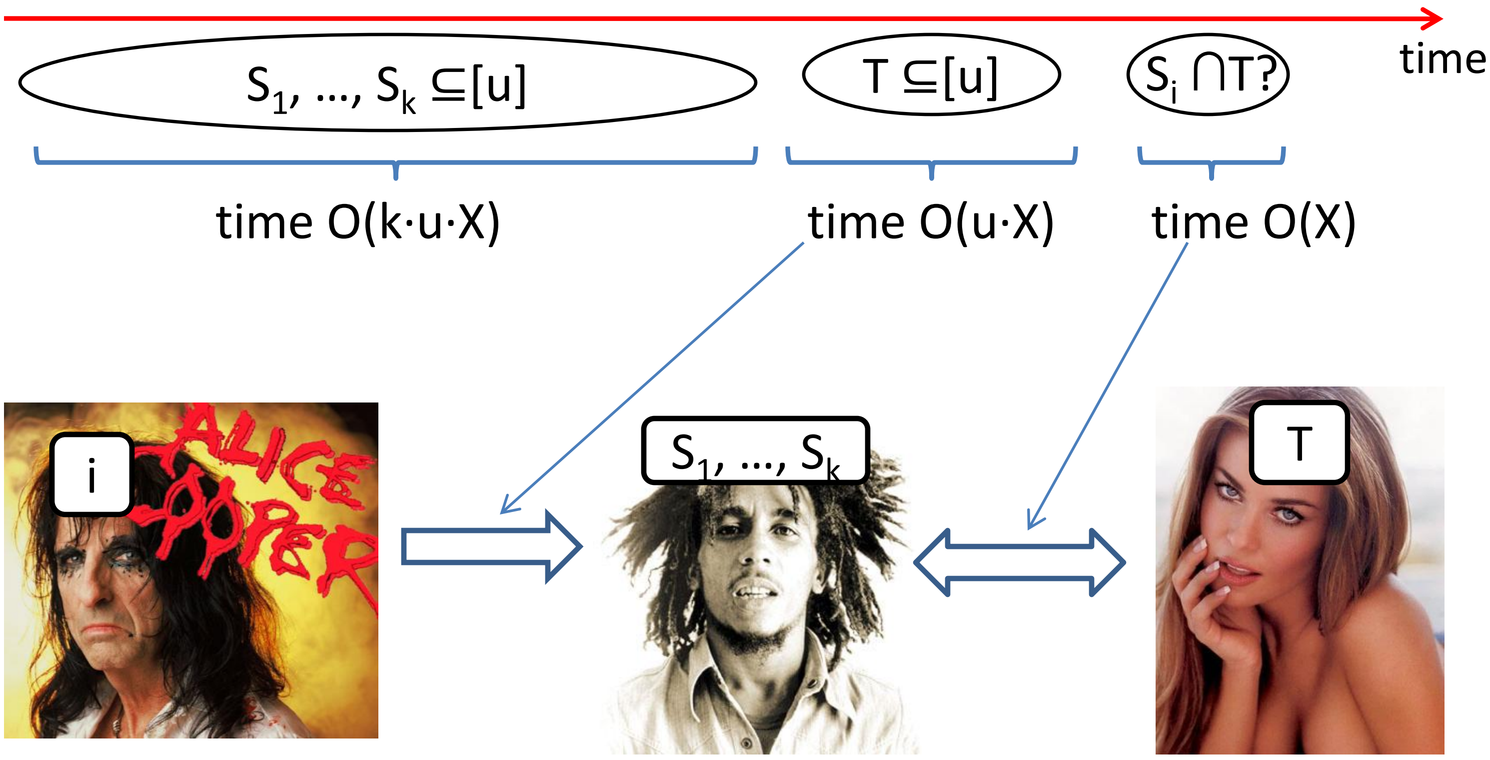
Sample application: maintain array A[1..n] under updates

Query: what's the must frequent element in A[i..j]?

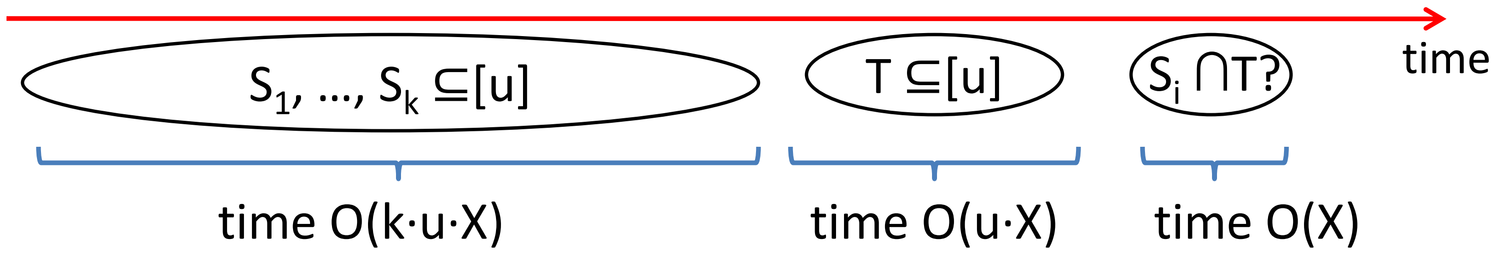
Conjecture ➔ $\max\{t_u, t_q\} = \Omega(n^\varepsilon)$



3-Party, Number-on-Forehead



A bit of cheating



Conjecture: if $u \cdot X \ll k$, must have $X = \Omega(u^{\varepsilon})$

⇧

3SUM-hardness

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

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

The End