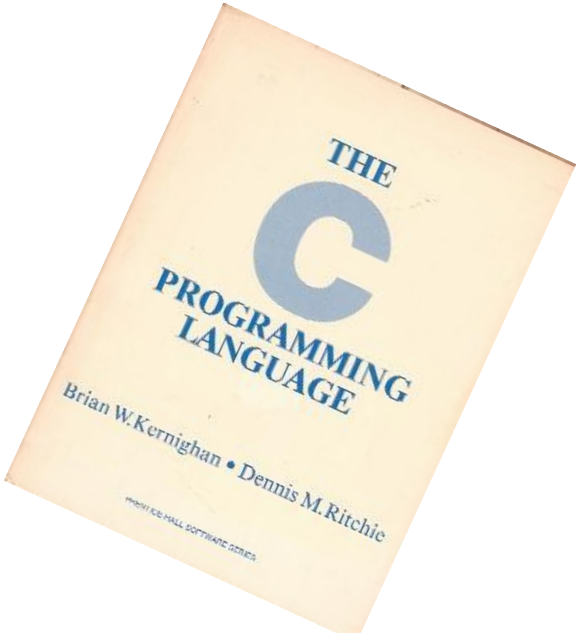
Lower Bounds for Data Structures

Mihai Pătrașcu

at&t

2^{nd} Barriers Workshop, Aug. 29 '10

Model of Computation



Word = w-bit integer

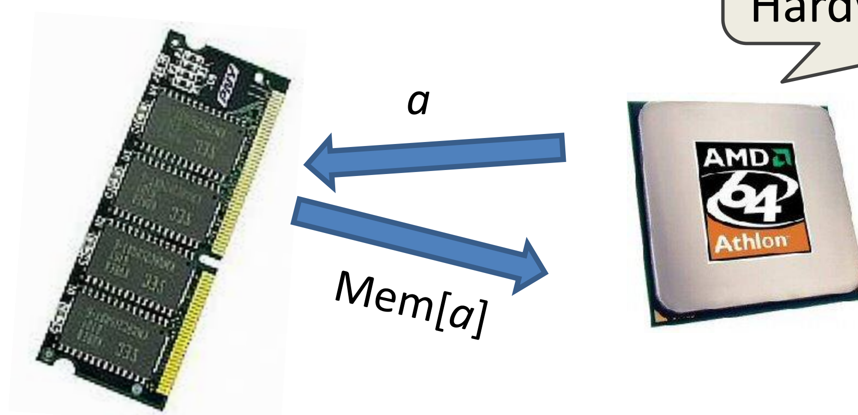
Memory = array of S words

Unit-time operations <u>on words</u>:

- random access to memory
- +, -, *, /, %, <, >, ==, <<, >>, ^, &, |, ~

Word size: $w = \Omega(\lg S)$

Internal state: $O(w)$ bits
Hardware: NC^1



Cell-Probe Model

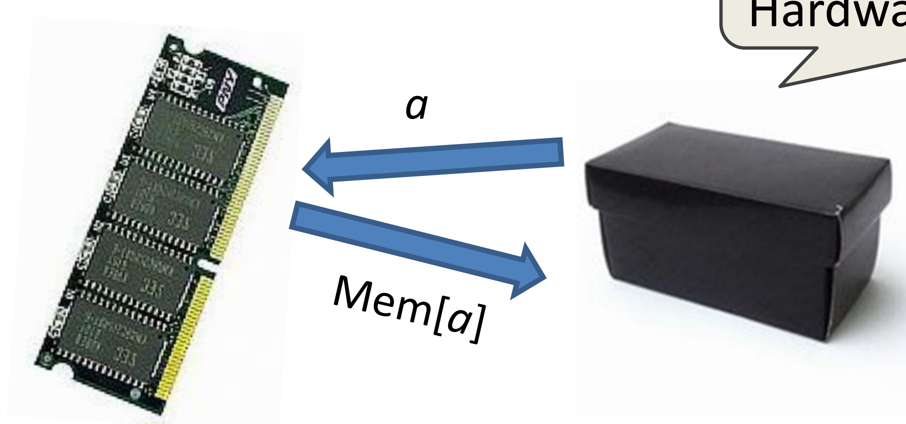
Cell = w bits

Memory = array of S cells

CPU:

- state: O(w) bits
- apply *any* function on state [***non-uniform!***]
- read/write memory cell in O(1) time

Internal state: O(w) bits
Hardware: anything



Classic Results

[Yao FOCS'78]

- (kind of) defines the model
- membership with low space

[Ajtai '88]

- static lower bound: predecessor search

[Fredman, Saks STOC'89]

- dynamic lower bounds: partial sums, union-find

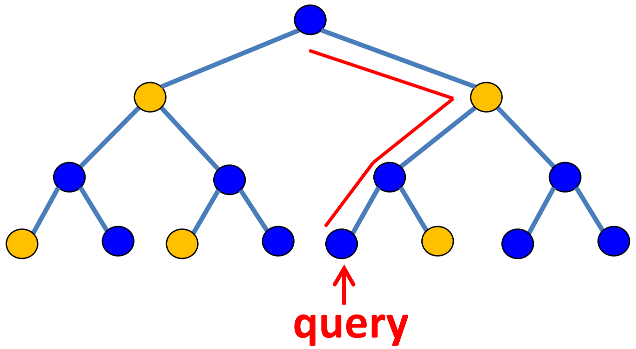
Have we broken barriers?

Dynamic Lower Bounds

Toy problem:

update: set node v to {0,1}

query: xor of root–leaf path



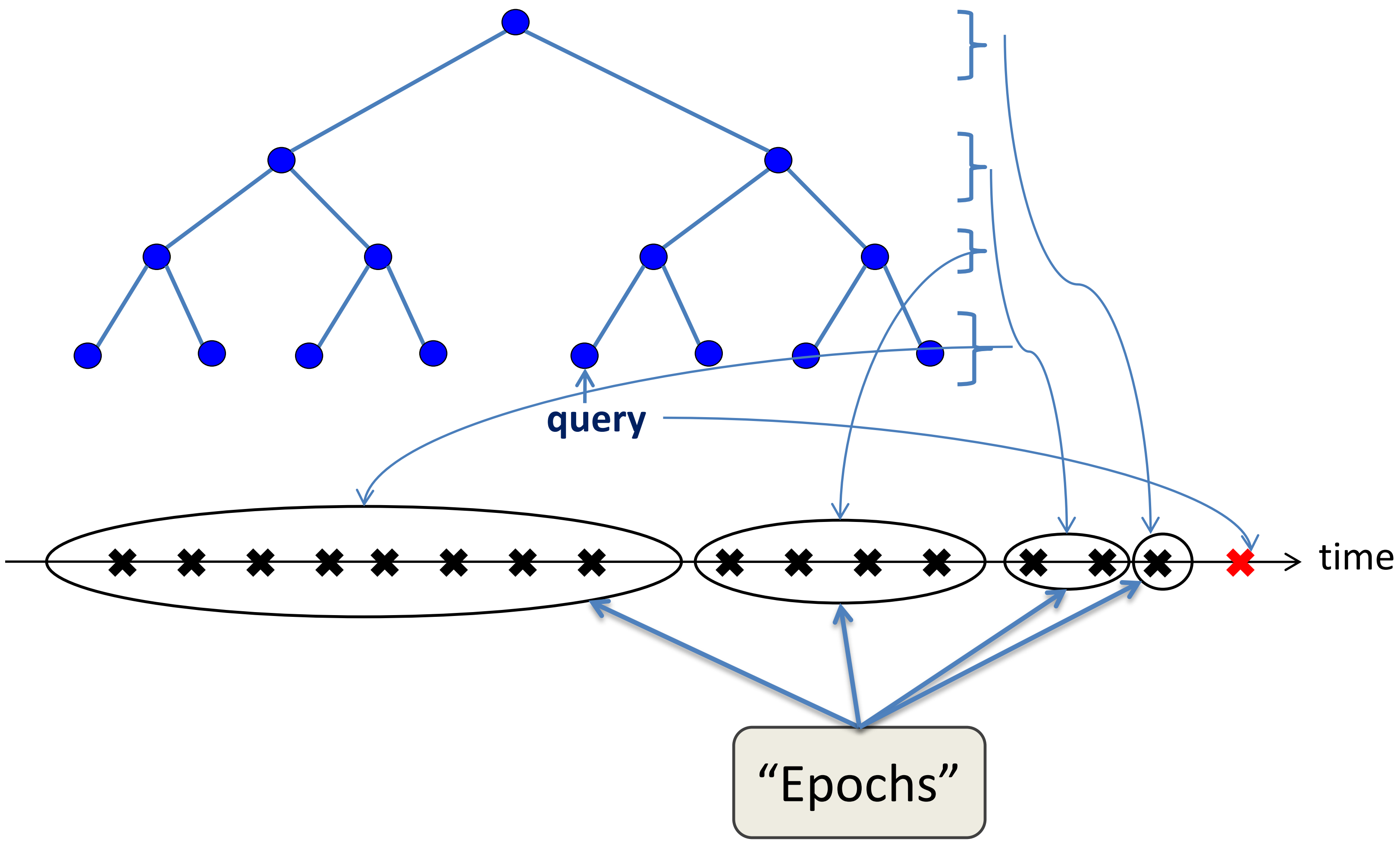
- n = # nodes
- $w = O(\lg n)$
- B = branching factor

[Fredman, Saks STOC'89]

Any data structure with update time $t_u = \lg^{O(1)} n$

requires query time $t_q = \Omega(\lg n / \lg\lg n)$

Hard Instance

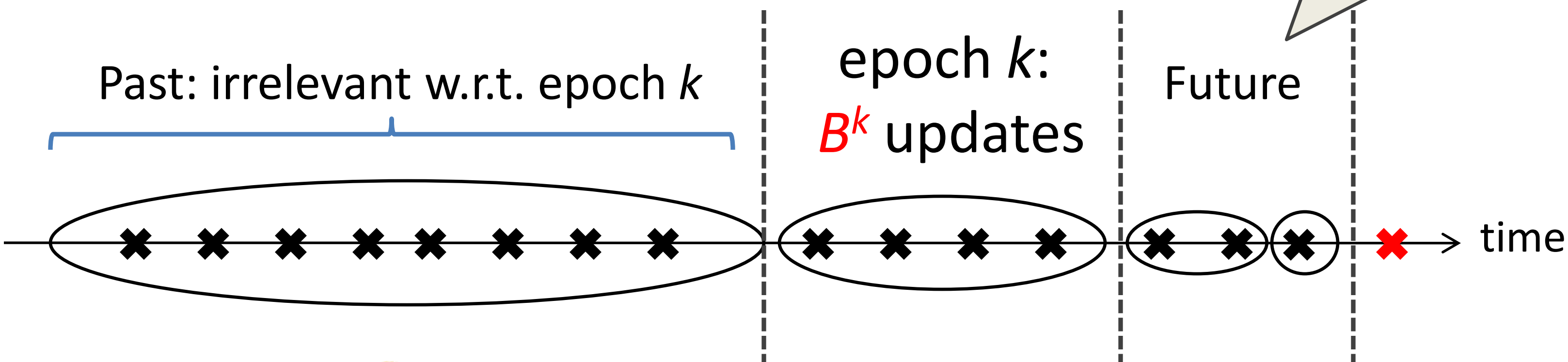


Proof Overview

Claim. (∀) *k*, Pr[query reads something from epoch *k*] ≥ 0.1

$\Rightarrow$ **$E[t_q] = \Omega(\log_B n)$**

Only $O(B^{i-1} t_u \lg n)$ bits are written



Let $B \gg t_u \lg n \Rightarrow B^k \gg B^{k-1} t_u \lg n$

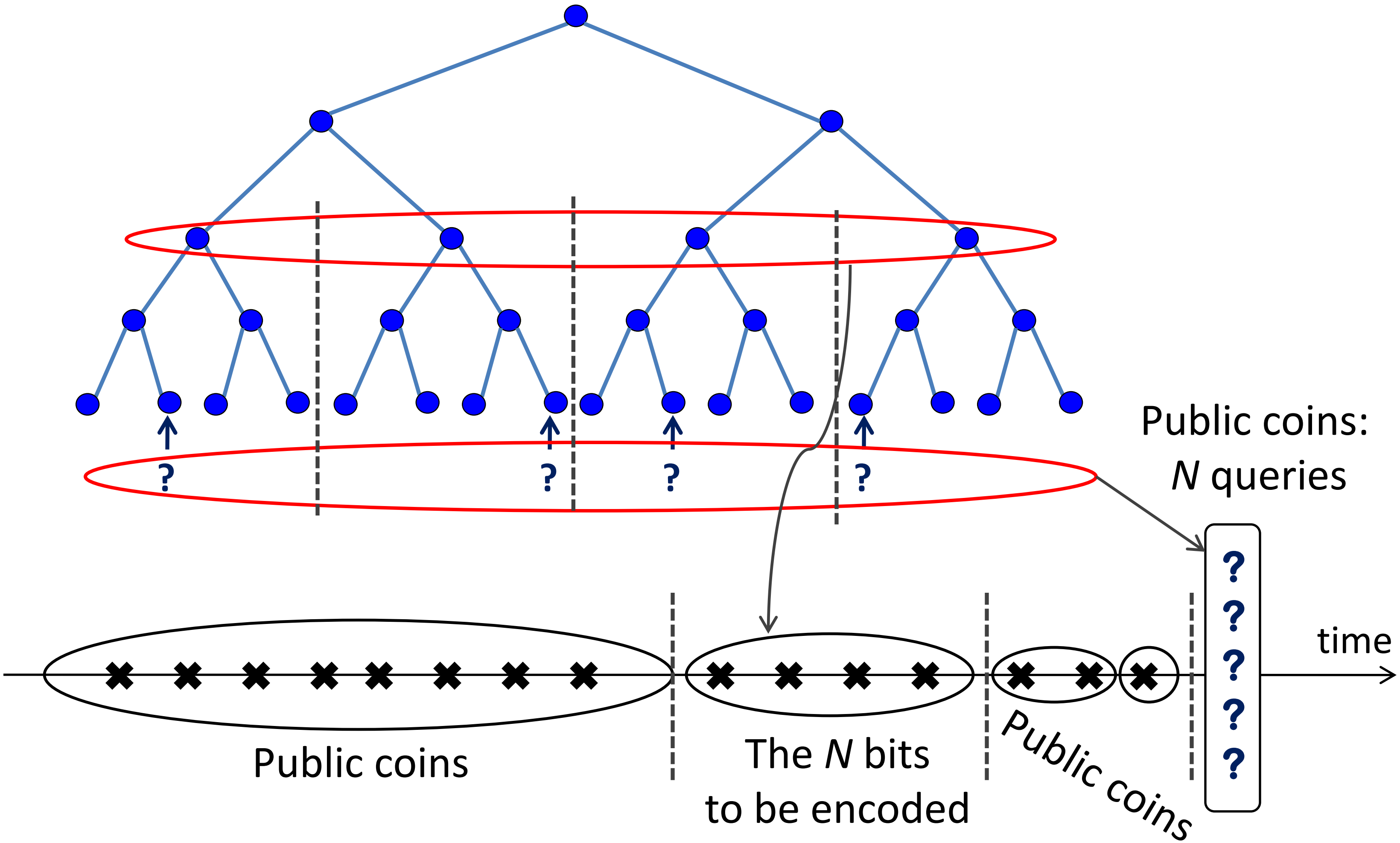
Formal Proof

Claim. (∀) k, Pr[query reads something from epoch k] ≥ 0.1

Proof: Assume not.

Encode $N=B^k$ random bits with $<N$ bits on average

Formal Proof



Formal Proof

Claim. (∀) k, Pr[query reads something from epoch k] ≥ 0.1

Proof: Assume not.

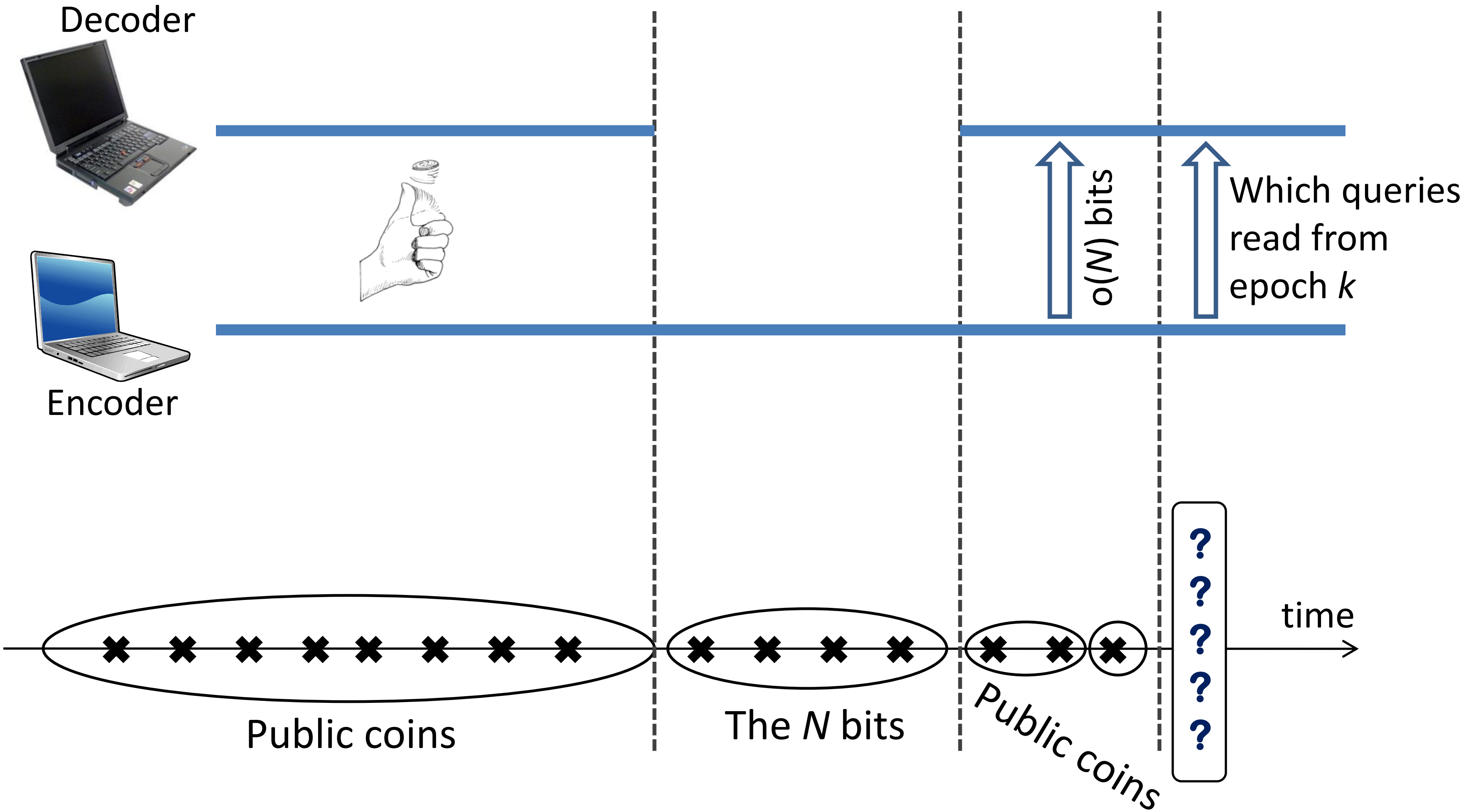
Encode N=B^k random bits with <N bits on average

Public coins: past updates, future updates, N queries

Equivalent task: encode query answers

Assumption ⇒ 90% of queries can be run ignoring epoch k

Formal Proof



Formal Proof

Claim. (∀) k, Pr[query reads something from epoch k] ≥ 0.1

Proof: Assume not.

Encode $N=B^k$ random bits with $<N$ bits on average

Public coins: past updates, future updates, N queries

Assumption ⇒ 90% of queries can be run ignoring epoch k

Encoding:

- what future epochs wrote — $o(N)$ bits
- which queries read from epoch k — $\lg$ (N choose $N/10$) $\ll$ N bits

□

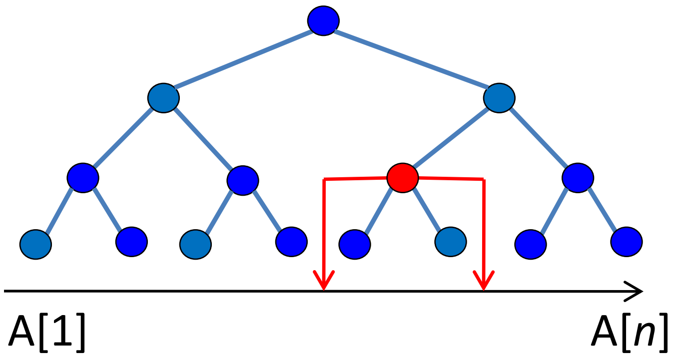
Applications

Partial sums:

Maintain an array A[1..n] under:

update(i, Δ): A[i] = Δ

sum(i): return A[1] + … + A[i]

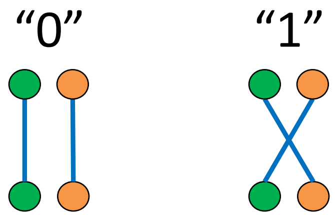


Incremental connectivity (union-find):

Maintain a graph under:

link(*u*,*v*): add edge

query(*u*,*v*): are *u* and *v* connected?

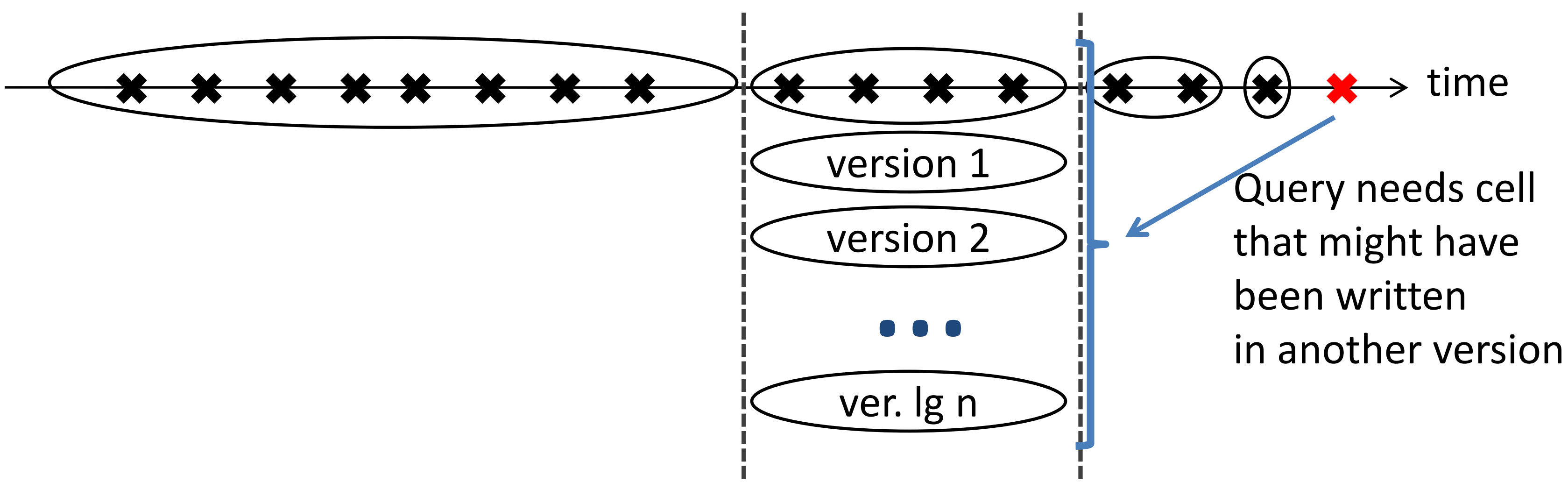


Fancy Application: Marked Ancestor

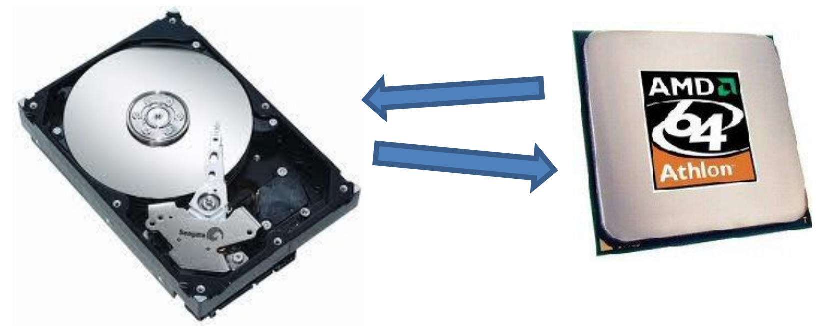
[Alstrup, Husfeldt, Rauhe FOCS'98]

- mark(v) / unmark(v)
- query(v): any marked ancestor ?

Only mark a node with probability ≈ 1/lg *n*



Fancy Application: Buffer Trees



External memory: $w = B \lg n$

Dictionary problem:

- $t_u = t_q = O(1)$
- $t_u = O(\lambda/B) \ll 1, \quad t_q = O(\log_\lambda n)$

[Verbin, Zhang STOC'10] [Iacono, Pătraşcu '11]

If $t_u = O(\lambda/B) \leq 0.99$, then $t_q = \Omega(\log_\lambda n)$

Fancy Application: Buffer Trees

[Verbin, Zhang STOC'10] [Iacono, Pătraşcu '11]

If $t_u = O(\lambda/B) \leq 0.99$, then $t_q = \Omega(\log_\lambda n)$

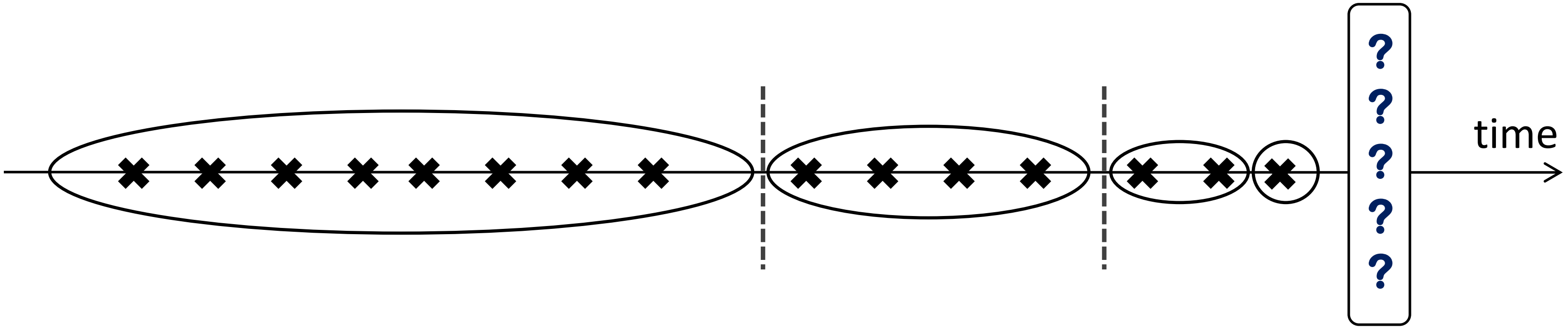
Queries = { N elements from epoch k } ∪ { N random elements }

Which is which? 2N bits to tell…

If true queries read from epoch k & false queries don't

⇒ can distinguish

So: Random false query reads from the epoch.

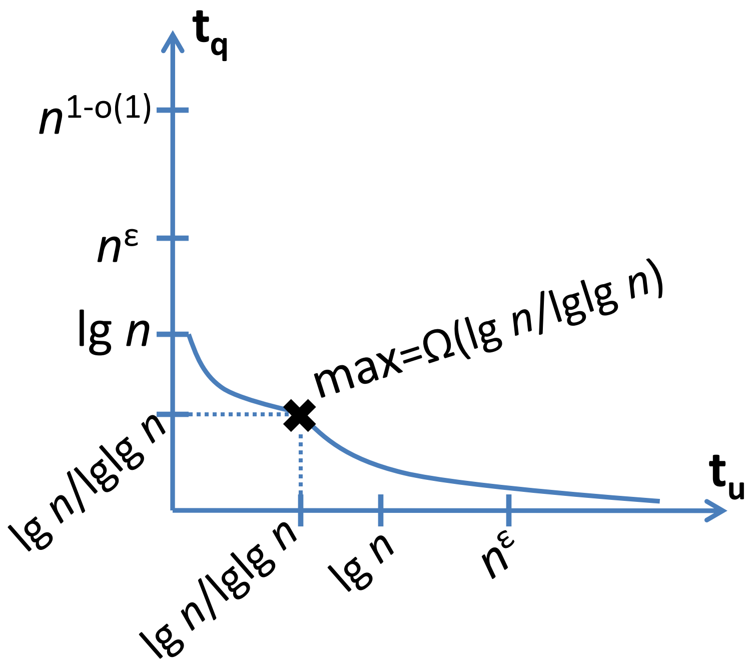


Higher Bounds

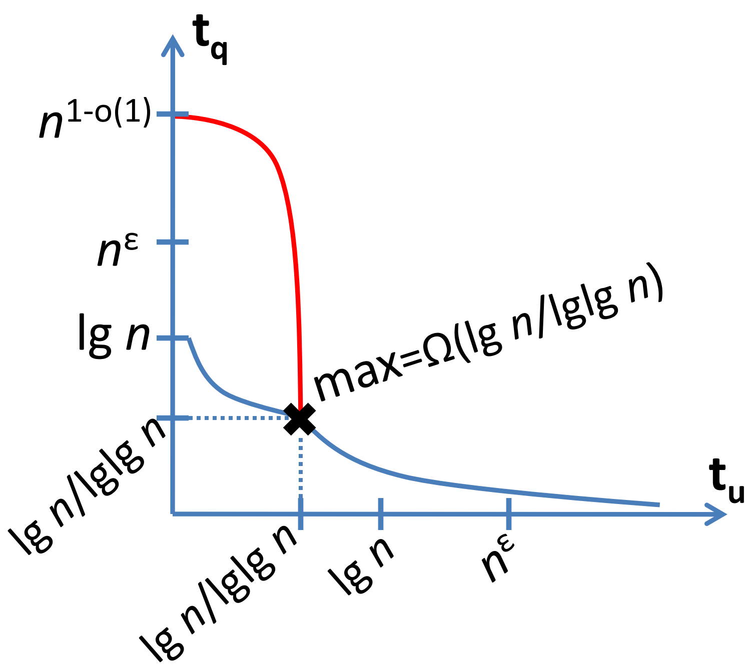
[Fredman, Saks STOC'89]

[Alstrup, Husfeldt, Rauhe FOCS'98]

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



Higher Bounds



[Fredman, Saks STOC'89]

[Alstrup, Husfeldt, Rauhe FOCS'98]

$t_q = \Omega(\lg n / \lg t_u)$

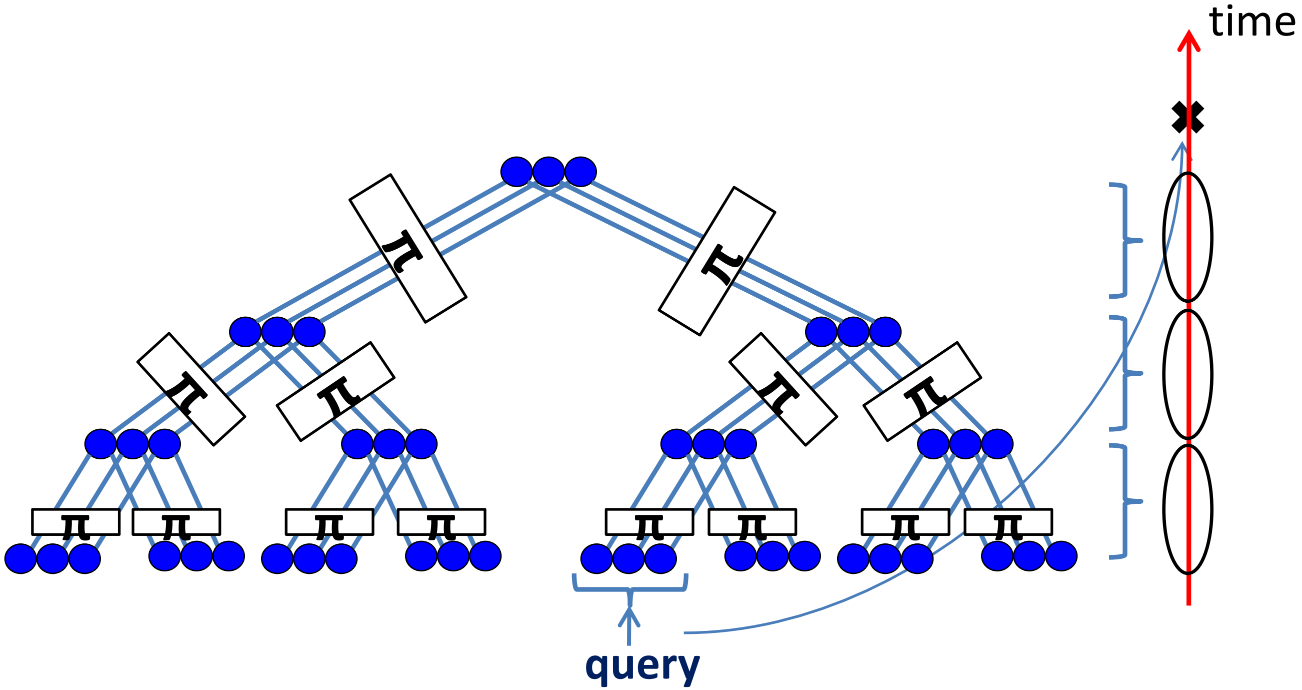
[Pătraşcu, Thorup '11]

$t_u = o(\lg n / \lg\lg n) \Rightarrow t_q \geq n^{1-o(1)}$

...for incremental connectivity

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

Hard Instance

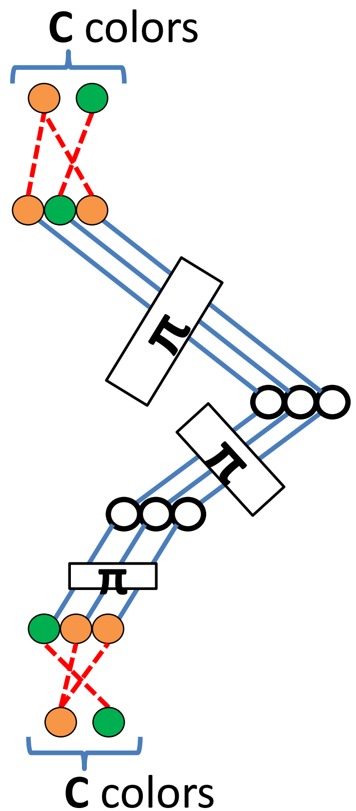


Hard Instance (cont.)

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

Operations:

- macro-update (setting one π)
  $\mapsto$ M edge inserts
- macro-query:
  - color root and leaf with C colors ($C=n^{\varepsilon}$)
    $\mapsto$ M edge inserts
  - test consistency of coloring
    $\mapsto$ C^2 connectivity queries



The Lower Bound

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

Operations:

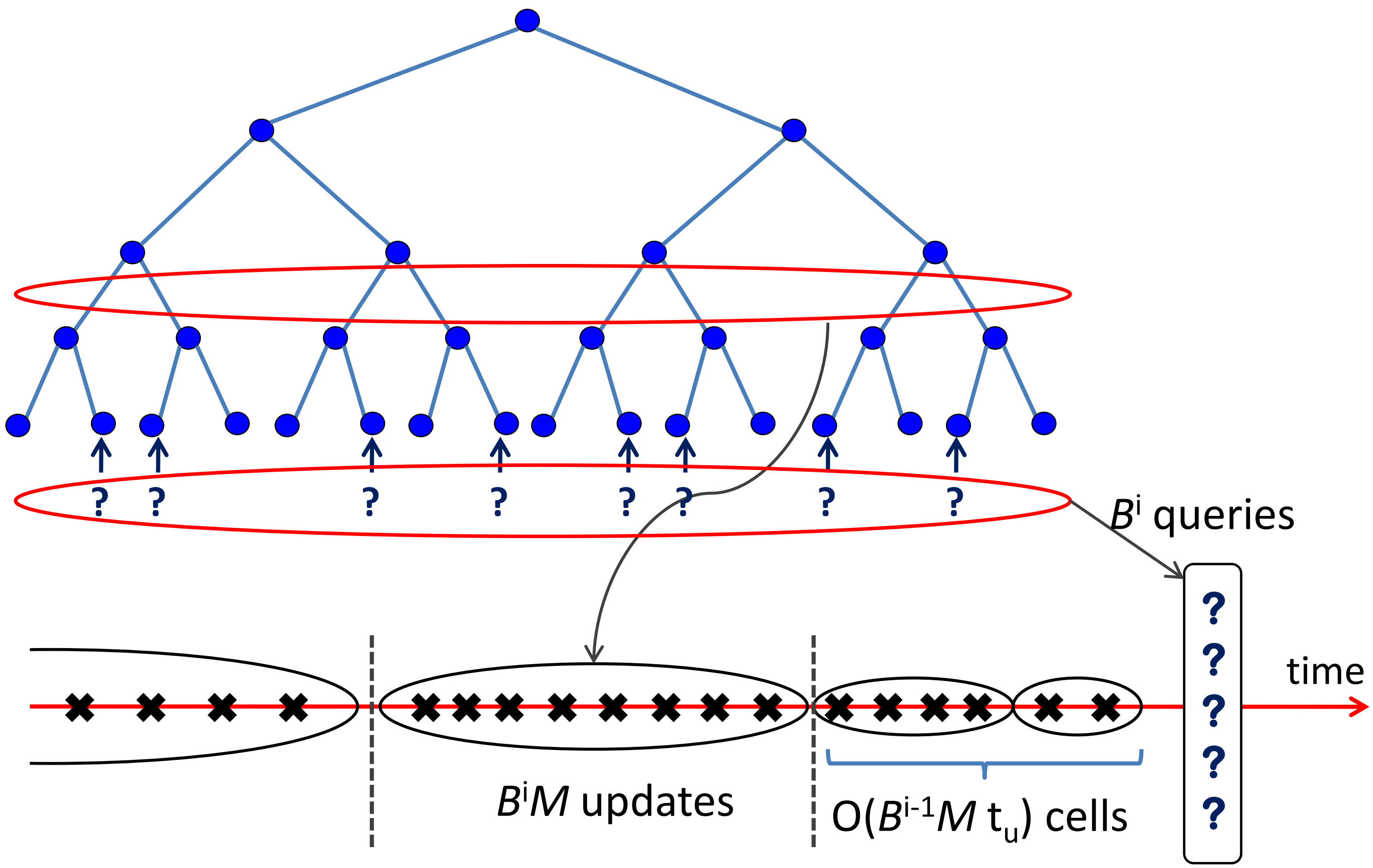
- macro-update: M insertions
- macro-query: M insertions + C^2 queries

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

So $M\ t_u + C^2\ t_q \geq M \cdot \lg n / \lg t_u$.

If $t_u = o(\lg n / \lg\lg n)$, then $t_q \geq M/C^2 = n^{1-3\varepsilon}$.

Hardness for One Epoch

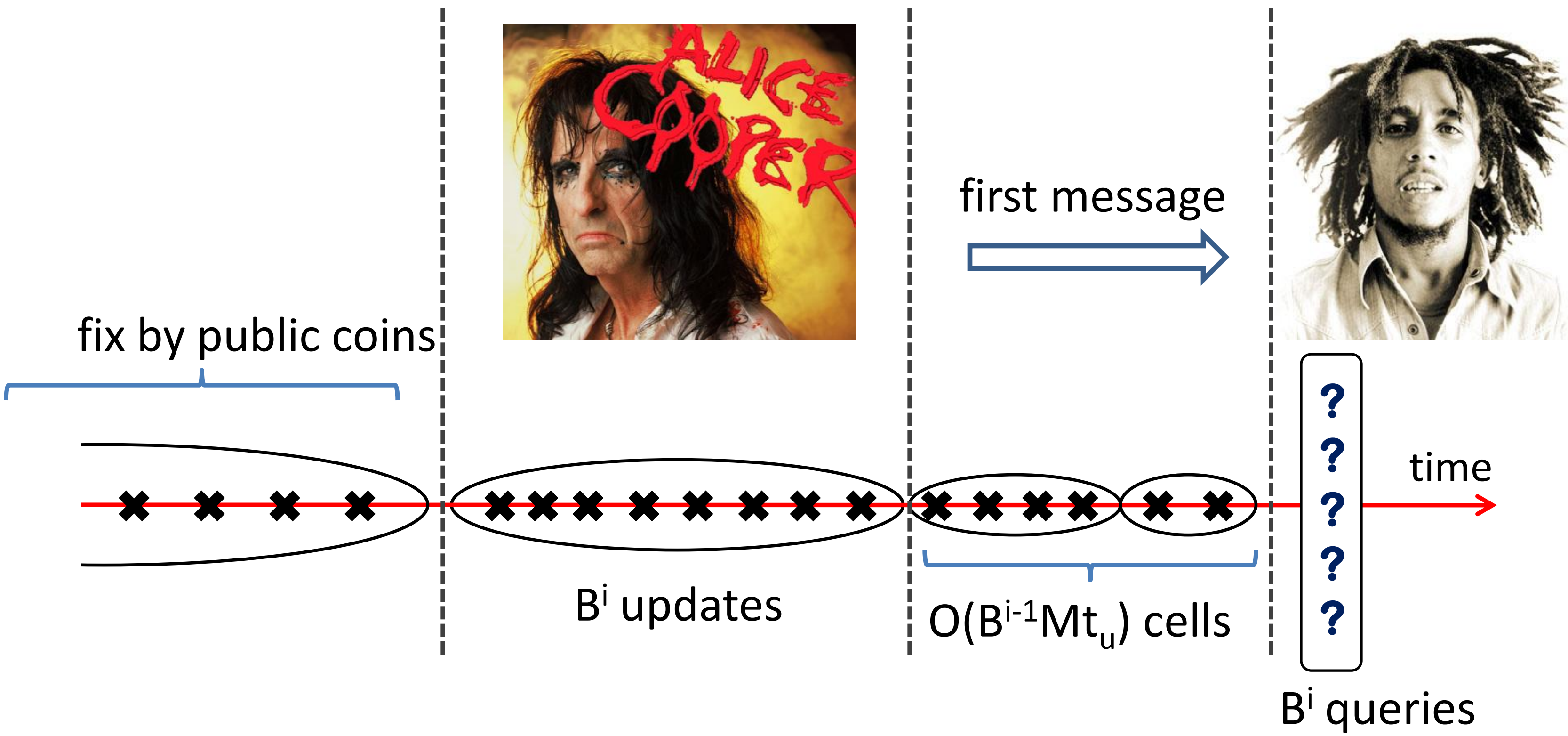


Communication

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

Bob: for each π_i, a coloring of inputs & outputs with C colors

Goal: test if all colorings are consistent



Communication (cont.)

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

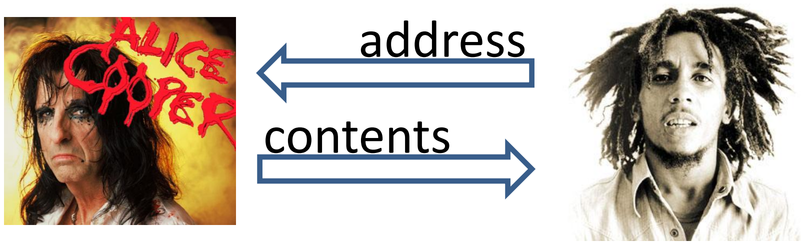
Bob: for each π_i , a coloring of inputs & outputs with C colors

Goal: test if all colorings are consistent

Lower bound: $\Omega(B^i M \lg M)$ [highest possible]

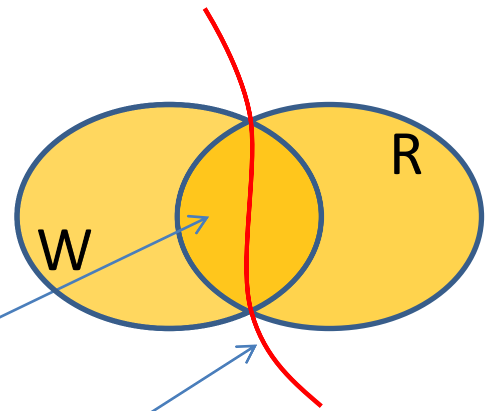
Upper bound: Alice & Bob simulate the data structure

The queries run $O(B^i M\ t_q)$ 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:

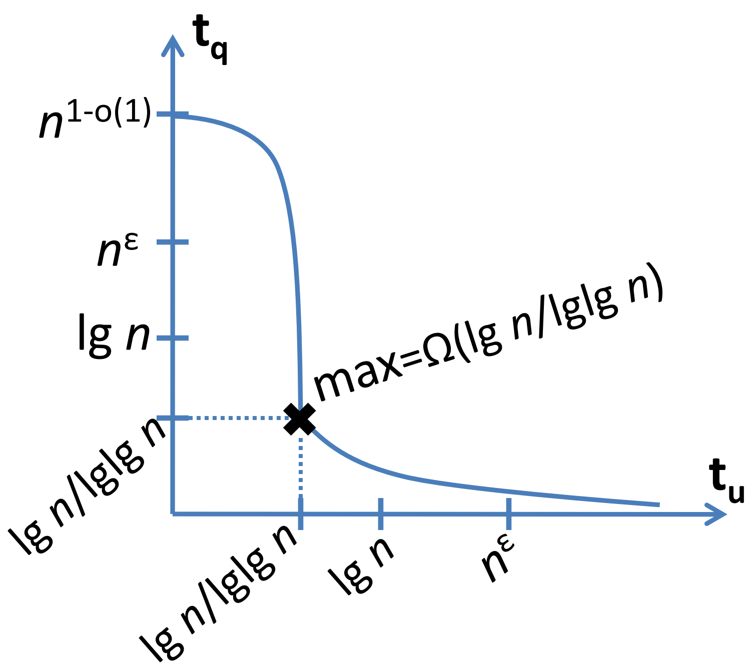
- address and contents of W∩R
  cost: $|W \cap R| \cdot O(\lg n)$ bits
- separator between W\R and R\W
  cost: $O(|W| + |R|)$ bits

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

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

So $|W \cap R| = \Omega(B^i M)$. □

Higher Bounds



[Fredman, Saks STOC'89]

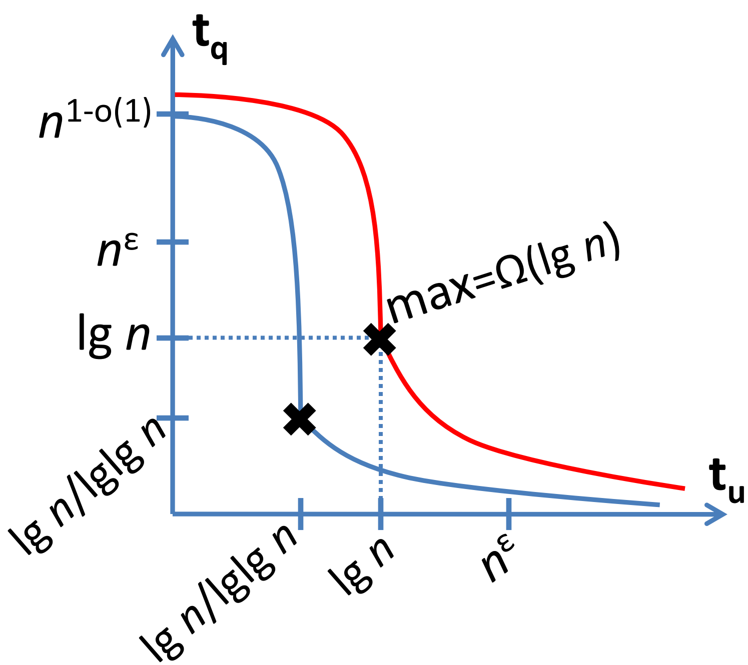
[Alstrup, Husfeldt, Rauhe FOCS'98]

$t_q=\Omega(\lg n / \lg t_u)$

[Pătraşcu, Thorup '11]

$t_u = o(\lg n / \lg\lg n) \Rightarrow t_q \geq n^{1-o(1)}$

Higher Bounds



[Fredman, Saks STOC'89]

[Alstrup, Husfeldt, Rauhe FOCS'98]

$t_q = \Omega(\lg n / \lg t_u)$

[Pătraşcu, Thorup '11]

$t_u = o(\lg n / \lg\lg n) \Rightarrow t_q \geq n^{1-o(1)}$

[Pătraşcu, Demaine STOC'04]

$t_q = \Omega(\lg n / \lg (t_u / \lg n))$

Also: $t_u = o(\lg n) \Rightarrow t_q \geq n^{1-o(1)}$

Maintain an array A[**n**] under:

update(**i**, Δ): A[**i**] = Δ

sum(**i**): return A[0] + … + A[**i**]

The hard instance:

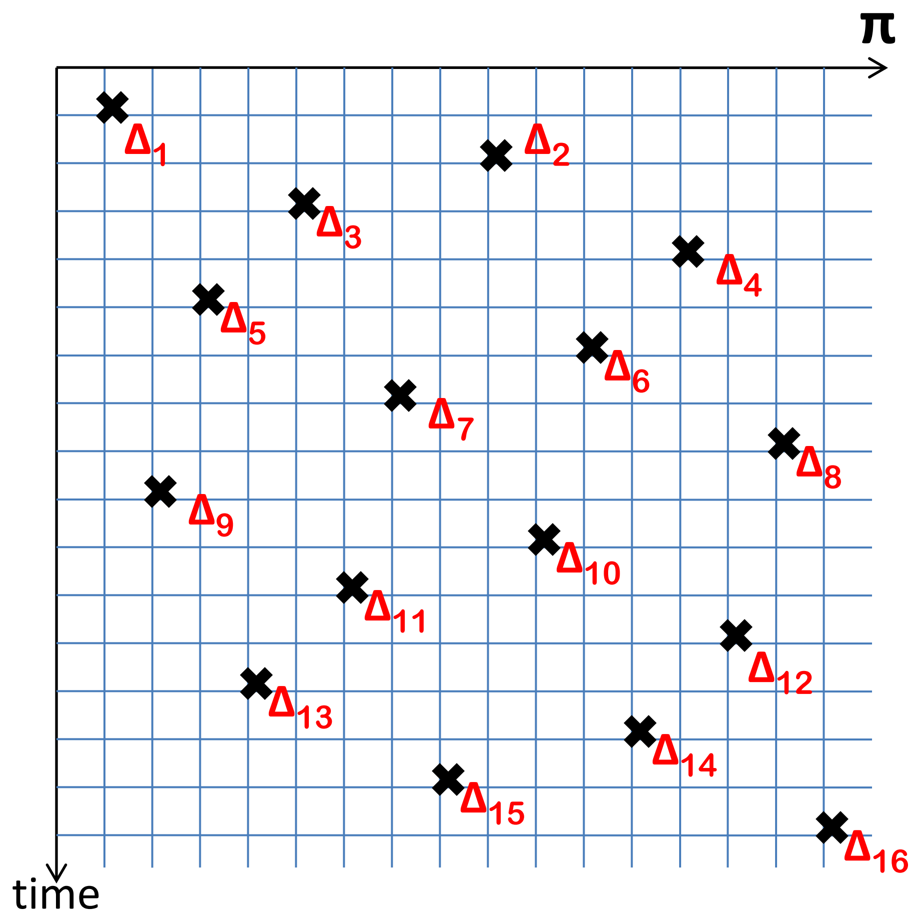
π = random permutation

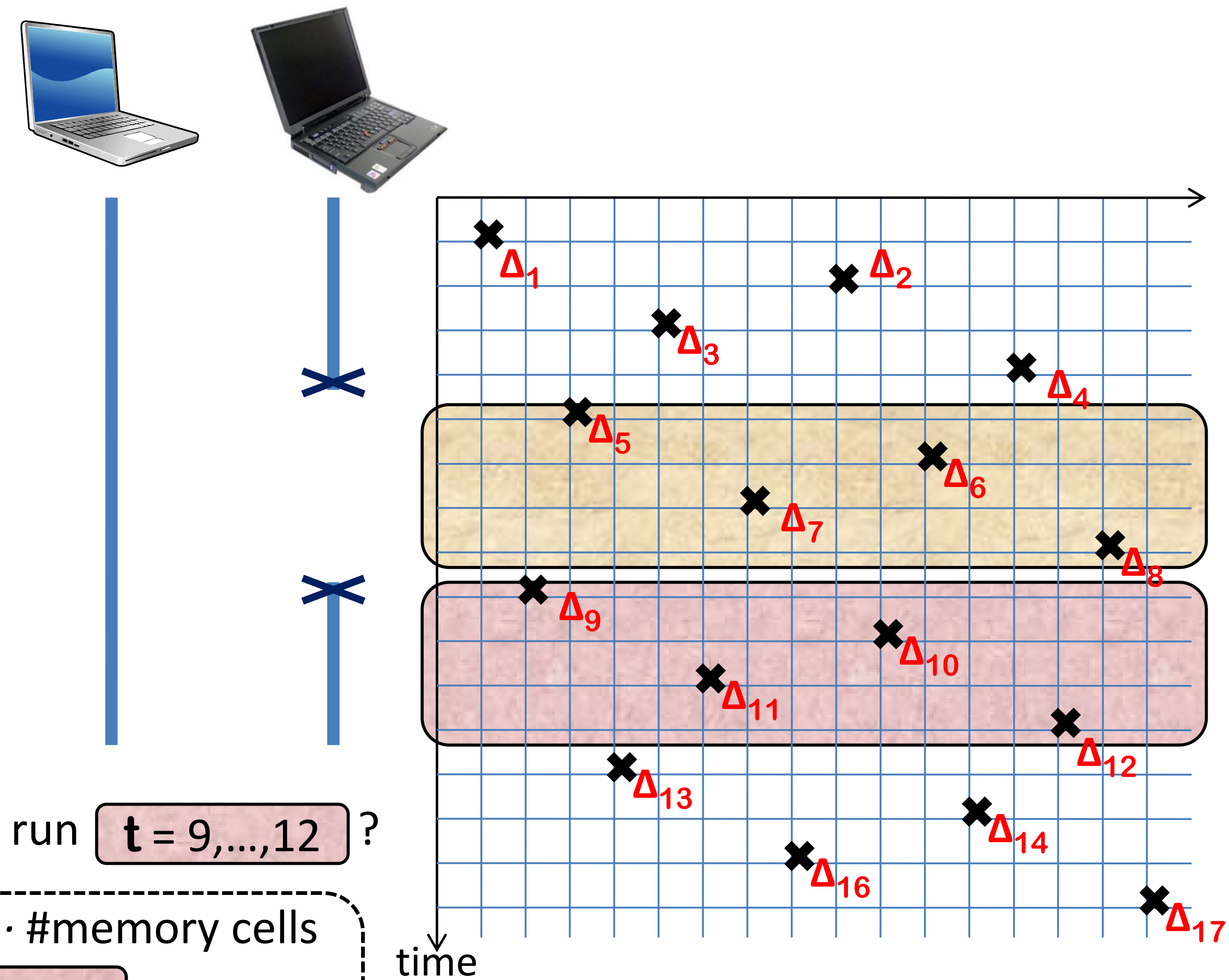
for **t** = **1** to **n**:

query: sum(**π**(**t**))

$\mathbf{\Delta_t}$= rand()

update(**π**(**t**), $\mathbf{\Delta_t}$)

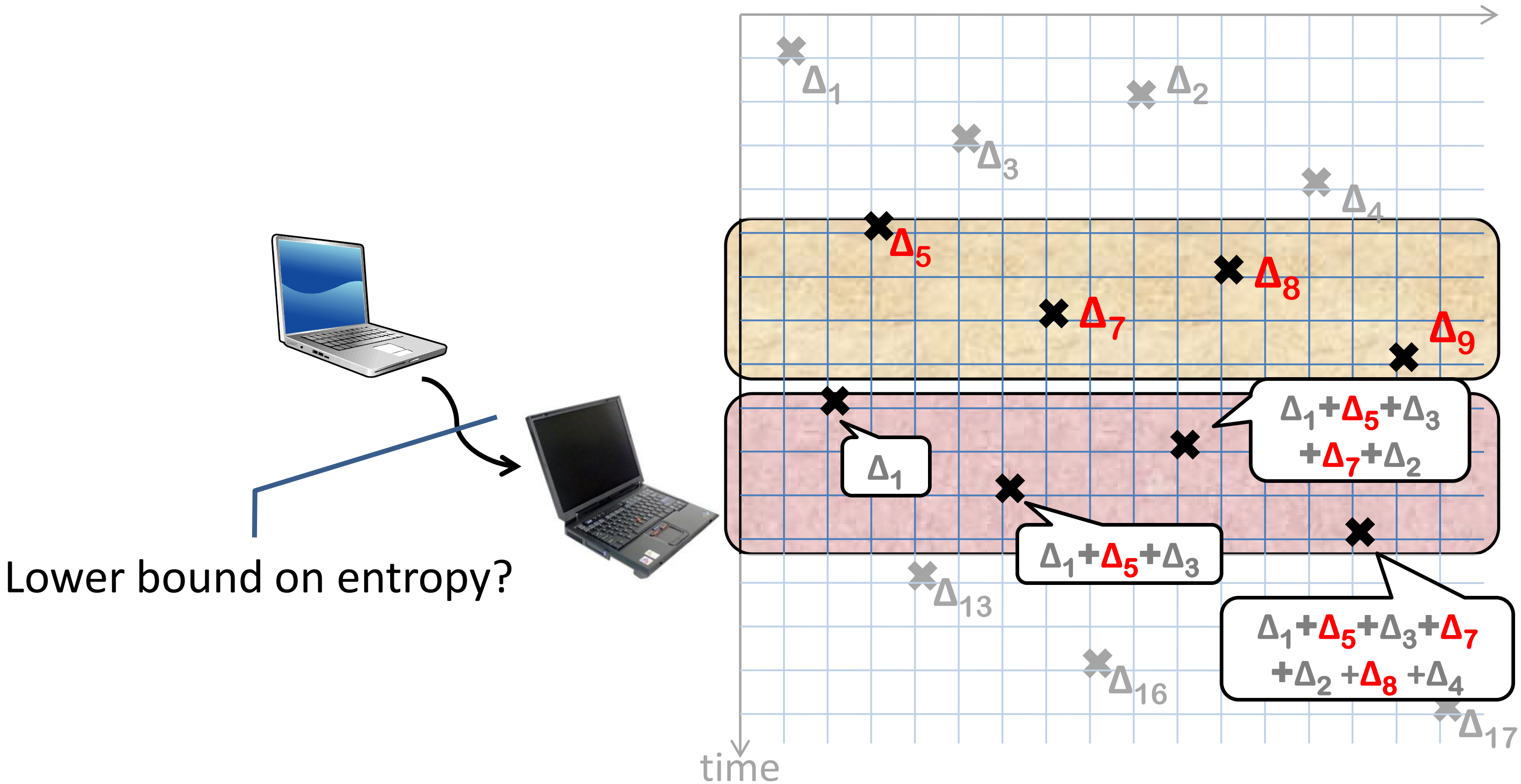




How can Mac help PC run $\mathbf{t} = 9,\ldots,12$?

Communication = $2w \cdot$ #memory cells

* read during $\mathbf{t} = 9,\ldots,12$
* written during $\mathbf{t} = 5, \ldots, 8$

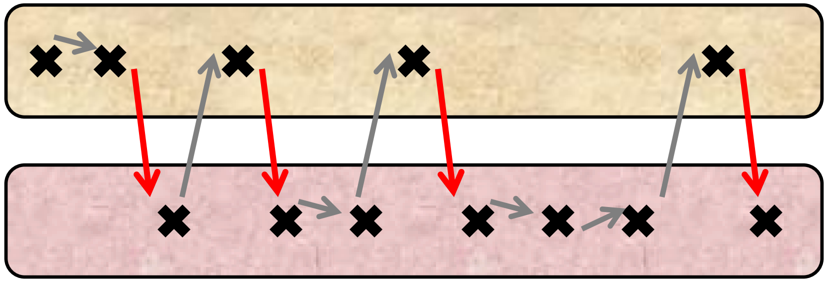


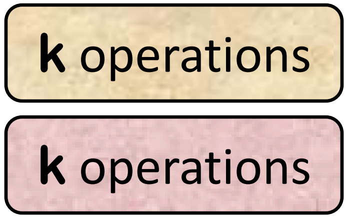
Lower bound on entropy?
Δ1
Δ2
Δ3
Δ4
Δ5
Δ7
Δ8
Δ9
Δ1
Δ1+Δ5+Δ3
Δ1+Δ5+Δ3
+Δ7+Δ2
Δ1+Δ5+Δ3+Δ7
+Δ2 +Δ8 +Δ4
Δ13
Δ16
Δ17
time

The general principle

Lower bound

= # down arrows





E[#down arrows] = $\Omega(\mathbf{k})$

Recap

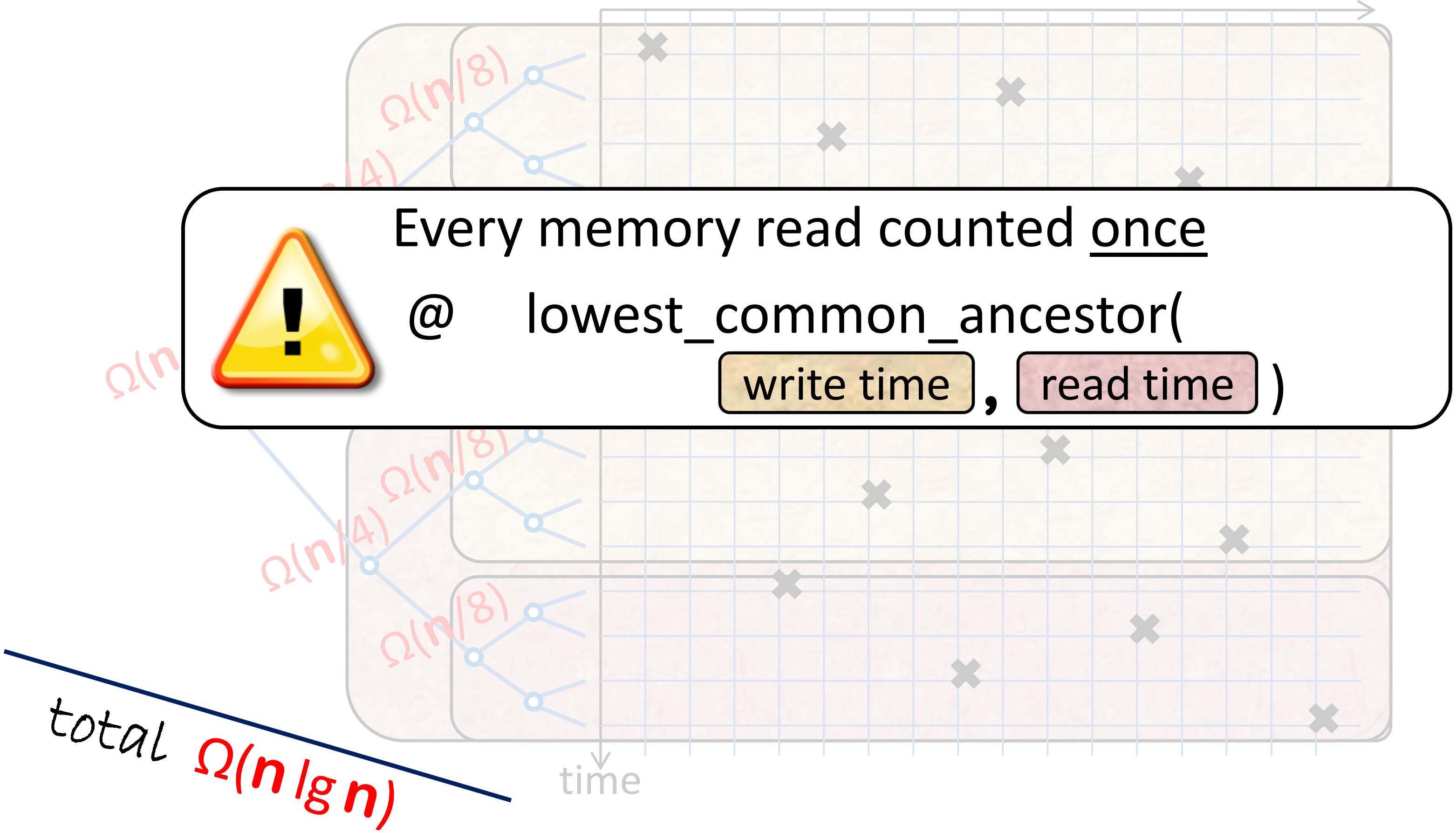
Communication = # memory locations
* read during mauve period
* written during beige period

Communication between periods of **k** items
= $\Omega(\mathbf{k})$

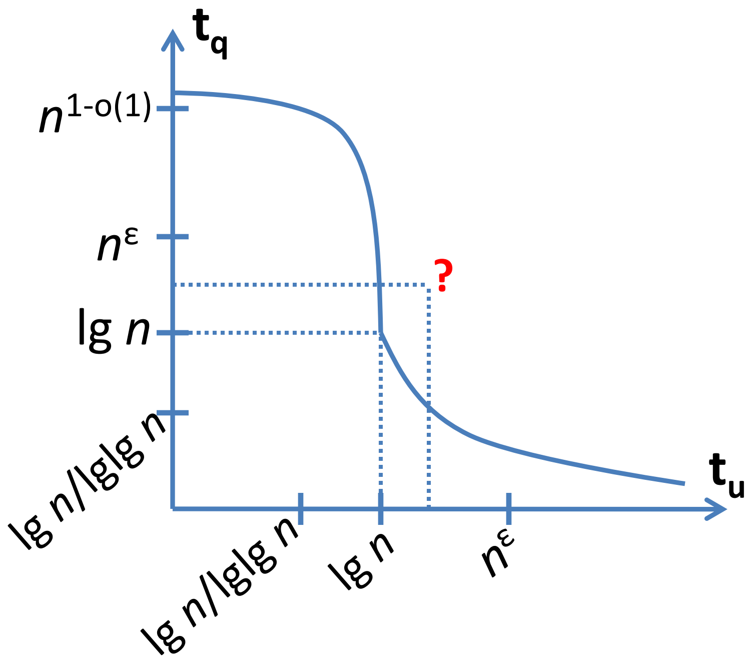
---

memory locations $\left\{ \begin{array}{l} \text{* read during mauve period} \\ \text{* written during beige period} \end{array} \right\}$ = $\Omega(\mathbf{k})$

Putting it all together



Dynamic Lower Bounds



[Fredman, Saks STOC'89]

[Alstrup, Husfeldt, Rauhe FOCS'98]

$t_q = \Omega(\lg n / \lg t_u)$

[Pătraşcu, Demaine STOC'04]

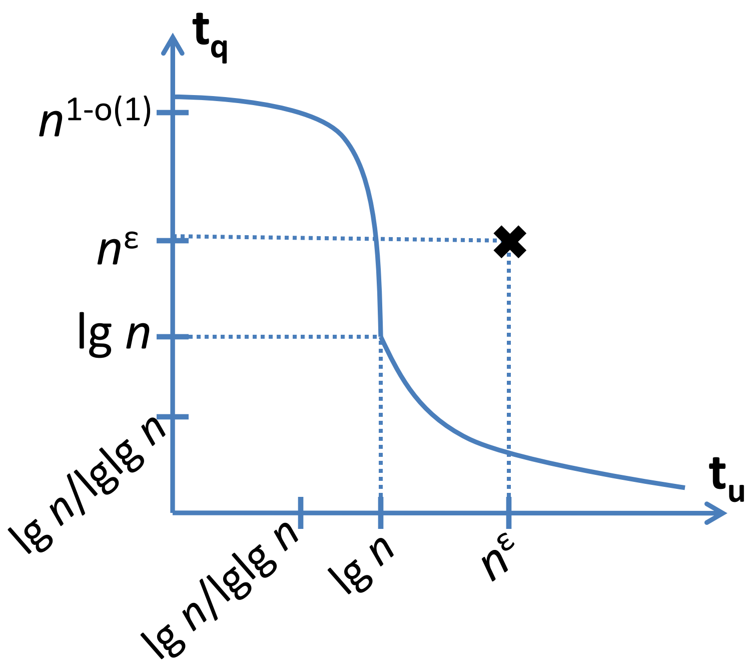
$t_q = \Omega(\lg n / \lg (t_u / \lg n))$

[Pătraşcu, Thorup '11]

$t_u = o(\lg n) \Rightarrow t_q \geq n^{1\text{-}o(1)}$

Some hope: $\max\{t_u, t_q\} = \Omega^*(\lg^2 n)$

Dynamic Lower Bounds



[Fredman, Saks STOC'89]

[Alstrup, Husfeldt, Rauhe FOCS'98]

$t_q = \Omega(\lg n / \lg t_u)$

[Pătraşcu, Demaine STOC'04]

$t_q = \Omega(\lg n / \lg (t_u / \lg n))$

[Pătraşcu, Thorup '11]

$t_u = o(\lg n) \Rightarrow t_q \geq n^{1-o(1)}$

[Pătraşcu STOC'10]

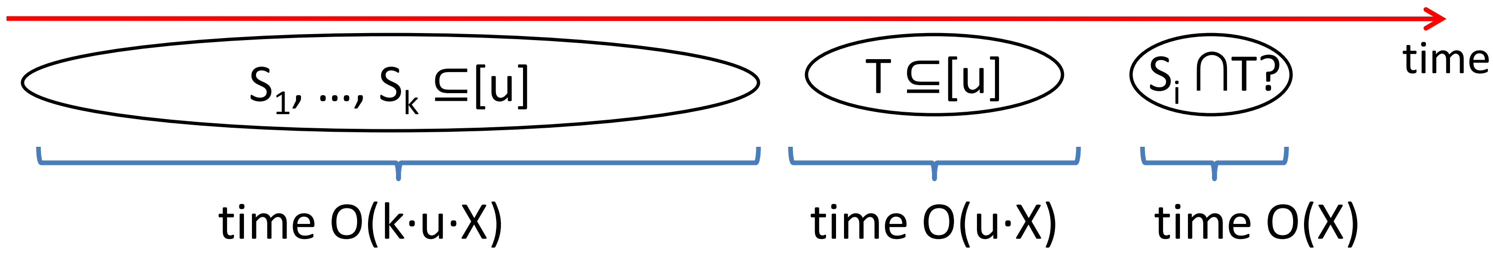
NOF conjecture $\Rightarrow \max\{t_u, t_q\} = \Omega(n^{\varepsilon})$

3SUM conjecture $\Rightarrow$ RAM lower bnd

3SUM: S = {n numbers}, $(\exists) x,y,z \in S$ with $x+y+z=0$?

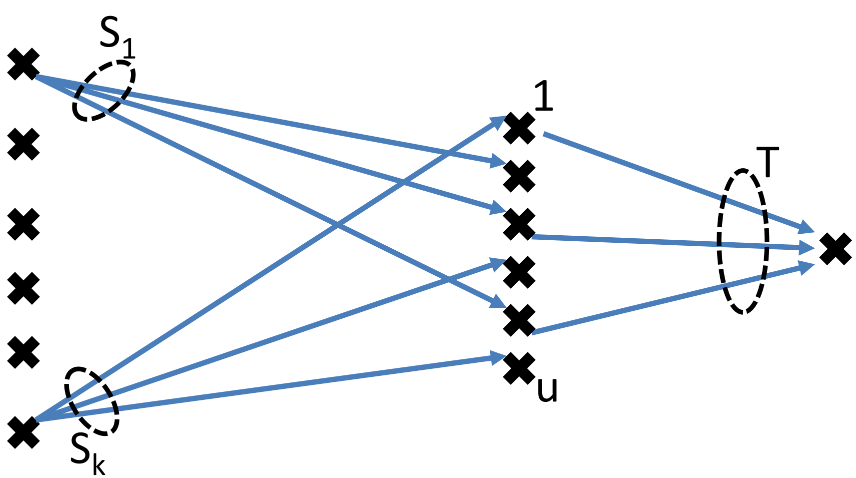
Conjecture: requires $\Omega^*(n^2)$ on RAM

The Multiphase Problem



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

$\Rightarrow$ reachability in dynamic graphs requires $\Omega(n^{\varepsilon})$

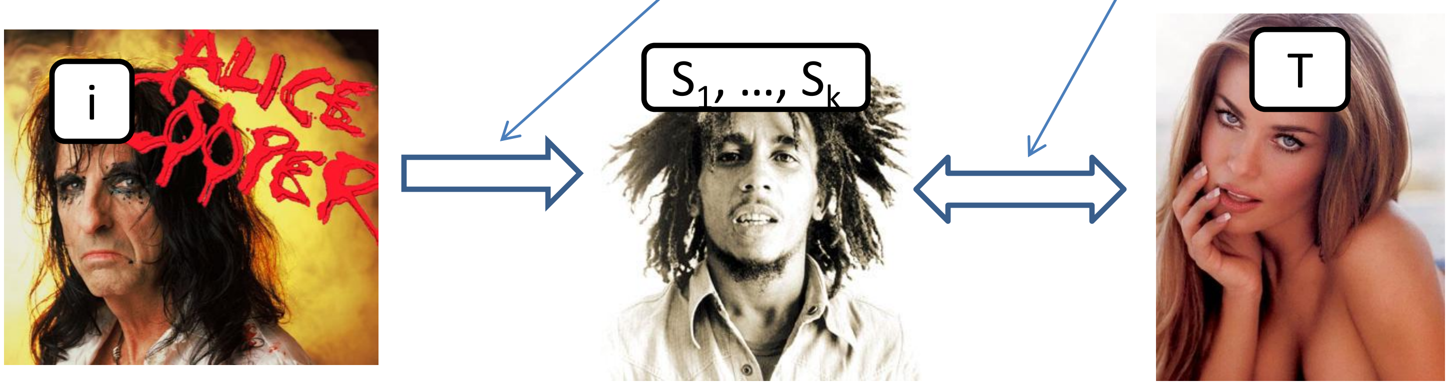


3-Party, Number-on-Forehead

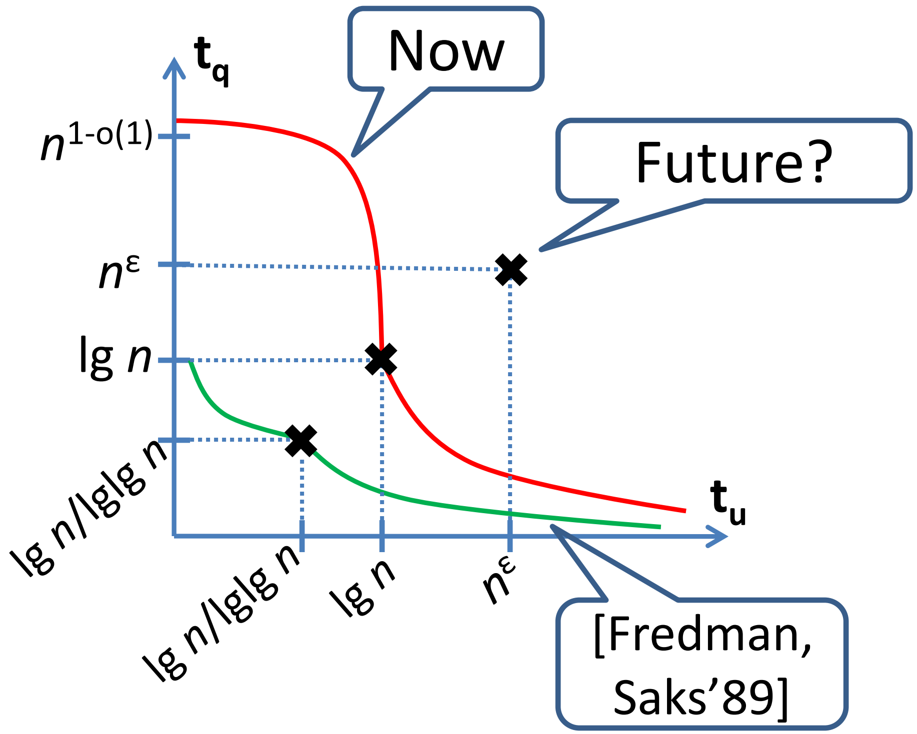
time

$S_1, \ldots, S_k \subseteq [u]$ — $T \subseteq [u]$ — $S_i \cap T$?

time $O(k \cdot u \cdot X)$ — time $O(u \cdot X)$ — time $O(X)$



Dynamic Lower Bounds



Classic Results

[Yao FOCS’78]

- (kind of) defines the model
- membership with low space

[Ajtai ’88]

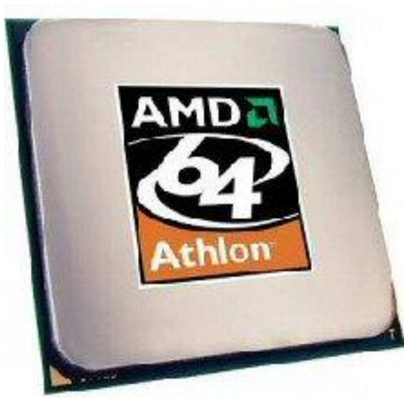
- static lower bound: predecessor search

[Fredman, Saks STOC’89]

- dynamic lower bounds: partial sums, union-find

Have we broken barriers?

Communication Complexity → Data Structures



lg S bits

w bits

lg S bits

w bits

Input: O(w) bits

Input: n bits

Asymmetric communication complexity

Tools in Asymmetric C.C.

[Ajtai'88] [Miltersen, Nisan, Safra, Wigderson STOC'95] [Sen, Venkatesh '03]

- round elimination
  …also message compression [Chakrabarti, Regev FOCS'04]

[Miltersen, Nisan, Safra, Wigderson STOC'95]

- richness

[Pătraşcu FOCS'08]

- lopsided set disjointness (via information complexity)

Round Elimination

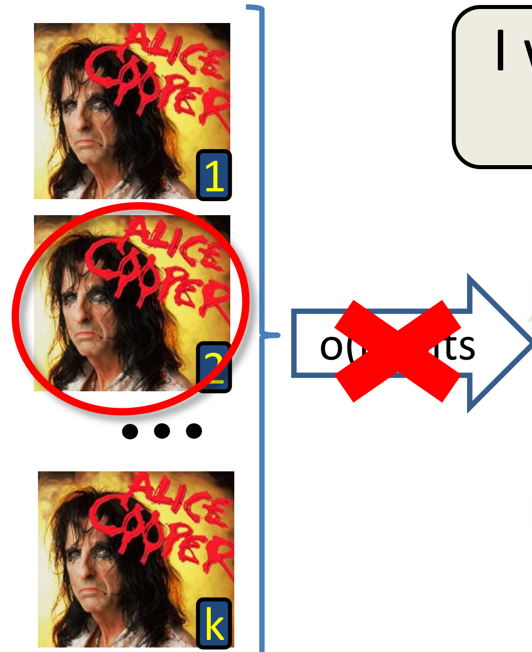
Round Elimination

important technicality

Setup: $f^{(k)}$
- Alice has input vector $(x_1, \dots, x_k)$
- Bob has inputs y, $i \in [k]$ and sees $x_1, \dots, x_{i-1}$
- Output: $f(x_i, y)$

If Alice sends a message of $m \ll k$ bits => **fix i** and **eliminate round**

Now: f
- Alice has input x_i
- Bob has an input y
- Output: $f(x_i, y)$



I want to talk to Alice i

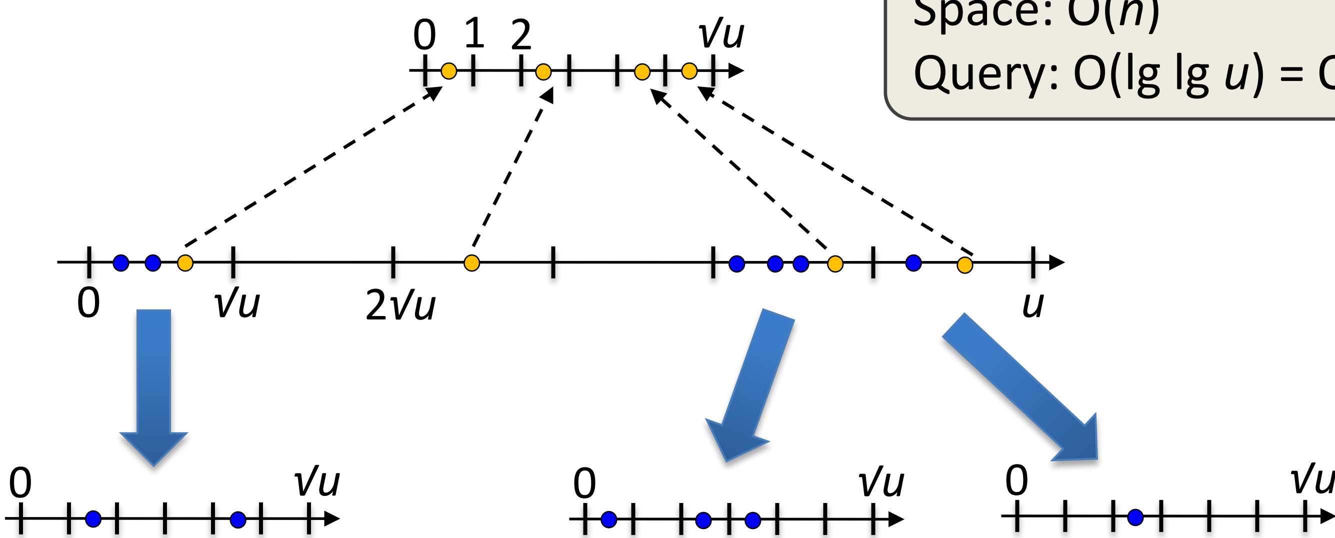
Predecessor Search

pred(q, S) = max { x ∈ S | x ≤ q }

[van Emde Boas FOCS'75]

if $\lfloor q/\sqrt{u} \rfloor$ ∈ hash table, **return** pred(q mod $\sqrt{u}$, bottom structure)

else return pred($\lfloor q/\sqrt{u} \rfloor$, top structure)



Space: O(n)

Query: O(lg lg u) = O(lg w)

Round Elimination $\mapsto$ Predecessor

[Ajtai'88] [Miltersen STOC'94] [Miltersen, Nisan, Safra, Wigderson STOC'95] [Beame, Fich STOC'99] [Sen, Venkatesh '03]

Alice: $q = (q_1, q_2, \ldots, q_k)$

Bob: $i \in [k]$, $(q_1, \ldots, q_{i-1})$, S

Goal: $\mathrm{pred}(q_i, S)$

Reduction to pred(q,T): $T = \{ (q_1,\ldots,q_{i-1}, x, 0,0,\ldots) \mid (\forall) x \in S \}$

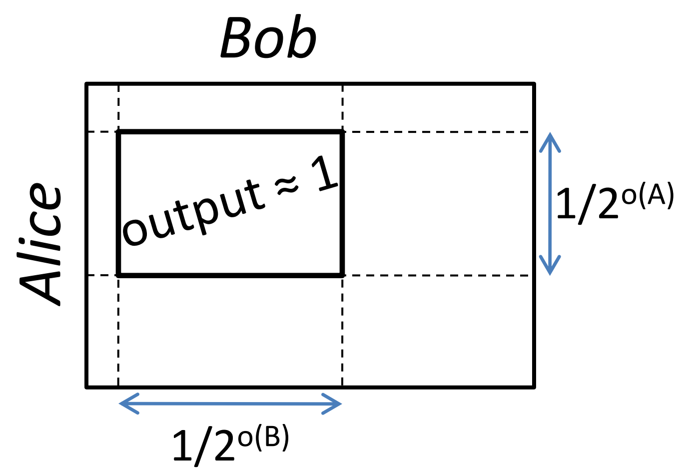
Space = $O(n)$ $\Rightarrow$ set $k = O(\lg n)$ $\Rightarrow$ lower bound: $\Omega(\log_{\lg n} w)$

Richness Lower Bounds

Prove: "either Alice sends *A* bits or Bob sends *B* bits"

Assume Alice sends o(A), Bob sends o(B)
=> big monochromatic rectangle

Show any big rectangle is bichromatic



E.g. Alice has $q \in \{0,1,*\}^d$ Bob has S=n points in $\{0,1\}^d$
Goal: does the query match anything?

[Pătraşcu FOCS'08] $A=\Omega(d)$, $B=\Omega(n^{1-\varepsilon})$
=> $t_q \geq \min \{ d/\lg S, n^{1-\varepsilon}/w \}$

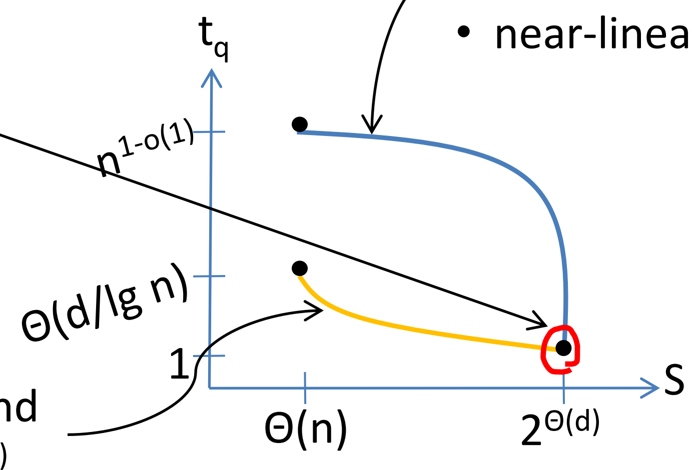
Richness Lower Bounds

What does this really mean?

“optimal space lower bound for constant query time”

upper bound ≈ either:

- exponential space
- near-linear query time



lower bound
$S = 2^{\Omega(d/t_q)}$

E.g. Alice has $q \in \{0,1,*\}^d$ Bob has $S=n$ points in $\{0,1\}^d$

Goal: does the query match anything?

[Pătraşcu FOCS’08] $A=\Omega(d)$, $B=\Omega(n^{1-\varepsilon})$

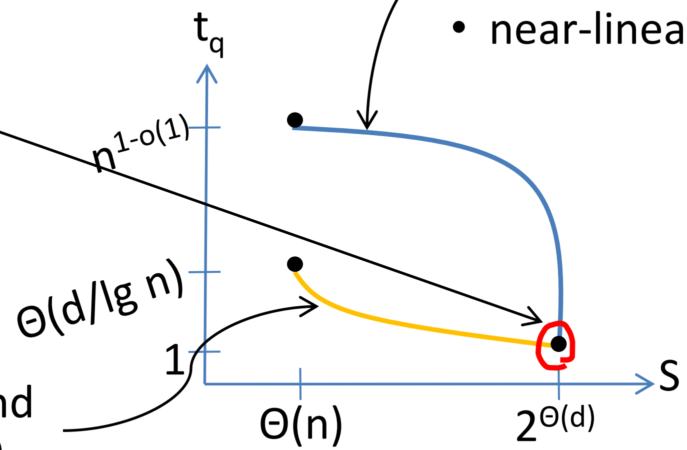
=> $t_q \geq \min\{d/\lg S, n^{1-\varepsilon}/w\}$

Richness Lower Bounds

upper bound ≈ either:
- exponential space
- near-linear query time

What does this really mean?

"optimal space lower bound for constant query time"



lower bound
$S = 2^{\Omega(d/t_q)}$

E.g. $\{0,1\}^d$?

Also: optimal lower bound for decision trees

[Pătraşcu FOCS'08] $A=\Omega(d)$, $B=\Omega(n^{1-\varepsilon})$

=> $t_q \geq \min \{ d/\lg S, n^{1-\varepsilon}/w \}$

Results

Partial match -- database of **n** strings in $\{0,1\}^d$, query ϵ $\{0,1,*\}^d$

[Borodin, Ostrovsky, Rabani STOC'99]

[Jayram,Khot,Kumar,Rabani STOC'03] $A = \Omega(d/\lg n)$

[Pătraşcu FOCS'08] $A = \Omega(d)$

Nearest Neighbor on hypercube (ℓ_1, ℓ_2):

deterministic γ-approximate: [Liu'04] $A = \Omega(d/\gamma^2)$

randomized exact: [Barkol, Rabani STOC'00] $A = \Omega(d)$

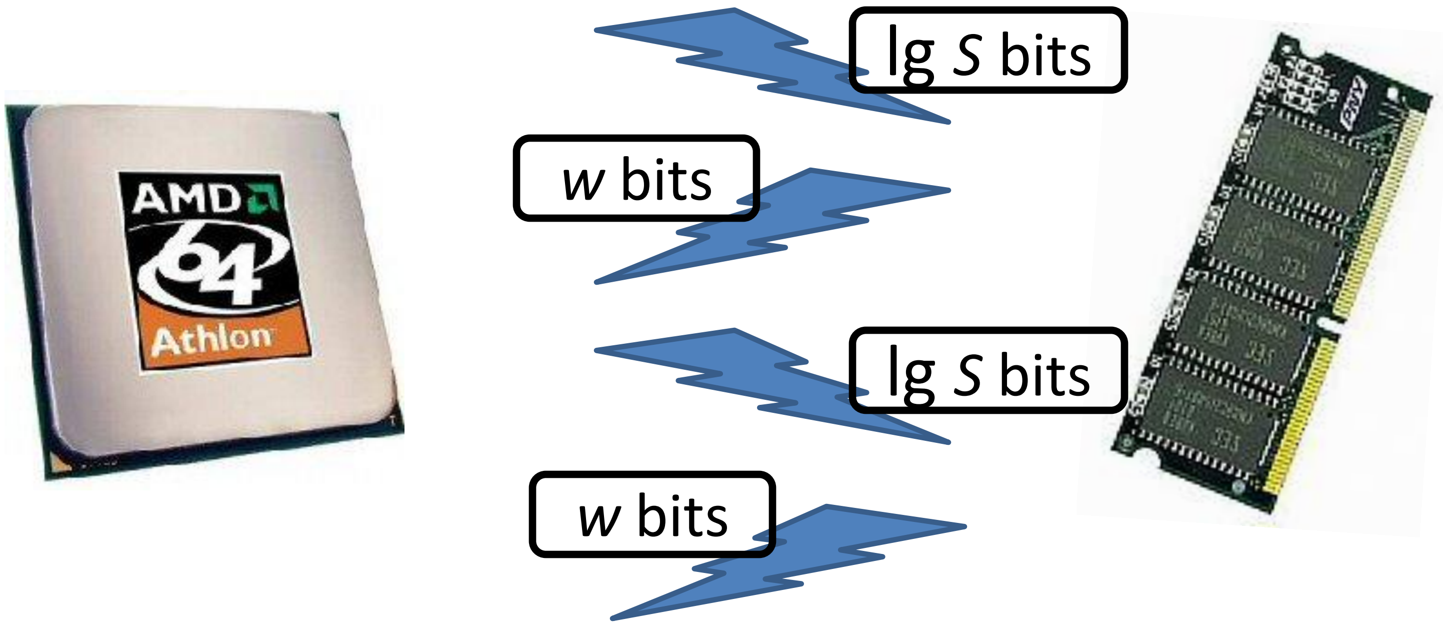
rand. $(1+\varepsilon)$-approx: [Andoni, Indyk, Pătraşcu FOCS'06] $A = \Omega(\varepsilon^{-2}\lg n)$

"Johnson-Lindenstrauss space is optimal!"

Approximate Nearest Neighbor in ℓ_∞:

[Andoni, Croitoru, Pătrașcu FOCS'08] "[Indyk FOCS'98] is optimal!"

The Barrier



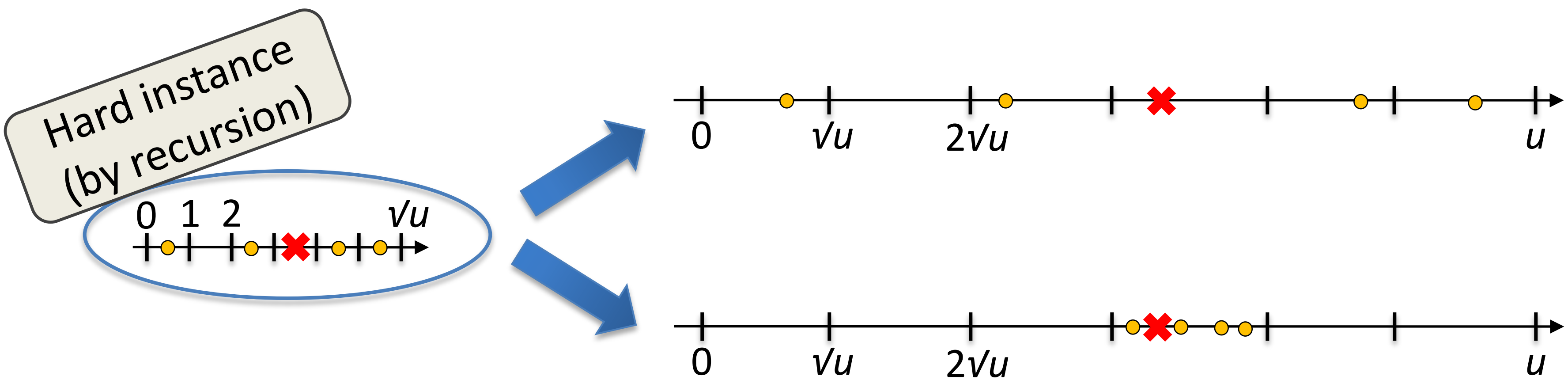
No separation between
$\mathbf{S}=O(\mathbf{n})$ and $\mathbf{S}=\mathbf{n}^{O(1)}$!

Predecessor Search

[Pătrașcu, Thorup STOC'06]

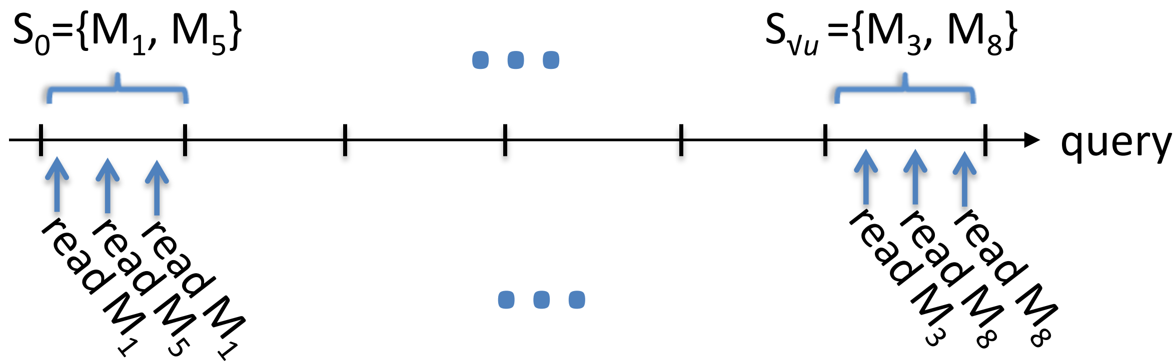
For $w = (1+\varepsilon) \lg n$ and space $O(n)$, predecessor takes $\Omega(\lg\lg n)$

Separation $O(n)$ space vs. $n^{1+\varepsilon}$



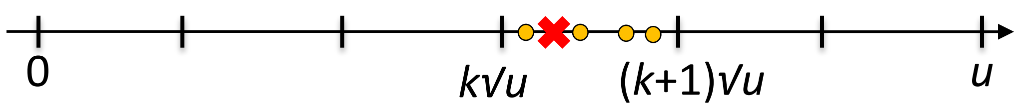
Claim: The 1st cell-probe can be restricted to set of $O(\sqrt{n})$ cells

Restricting 1st Cell Probe

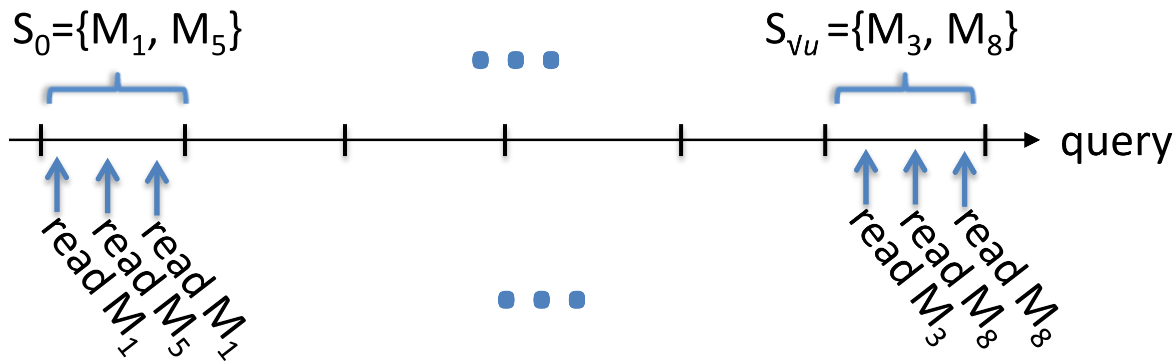


If $(\exists)k \quad |S_k| \leq \sqrt{n}$:

- place query & data set in segment k
- 1st memory access = $f(\mathrm{lo}(q)) \in S_k$

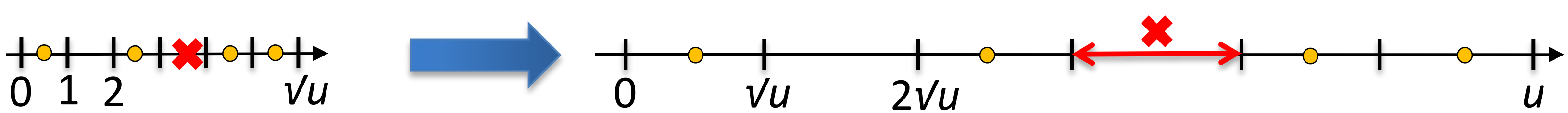


Restricting 1st Cell Probe



Otherwise $(\forall)k \quad |S_k| \geq \sqrt{n}$:

- choose T = { $O(\sqrt{n} \cdot \lg n)$ cells } $\Rightarrow$ each S_k is hit
- 1st memory access = $f(\text{hi}(q), \text{lo}(q)) \in S_{\text{lo}(q)}$
- make lo(q) irrelevant $\Rightarrow$ fix to make $f(\text{hi}(q), *) \in T$



What Did We Prove?

If there exists a solution to Pred(n, u) with:

- space complexity: O(n)
- query complexity: t memory reads

$\Rightarrow$

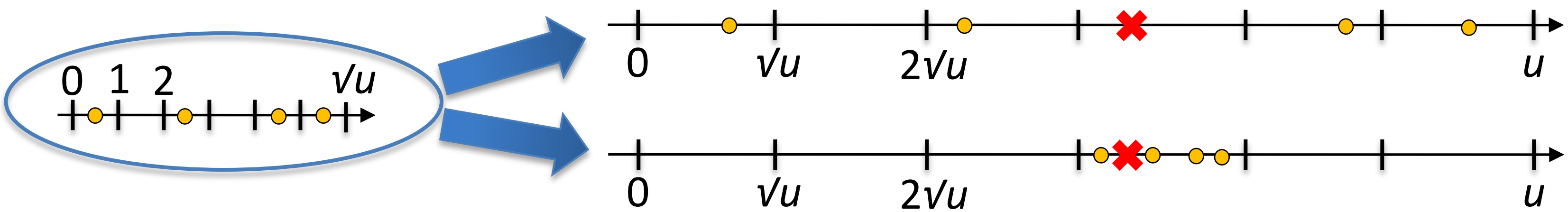
There exists a solution to Pred(n, $\sqrt{u}$) with:

- space complexity: O(n)
- O($\sqrt{n} \cdot \lg n$) *"published cells"*
- query complexity: t-1 memory reads

… can be read free of charge

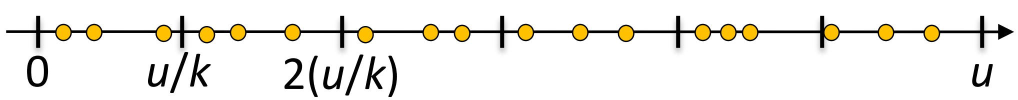
Dealing with Public Bits

Hardness came from one "secret" bit:



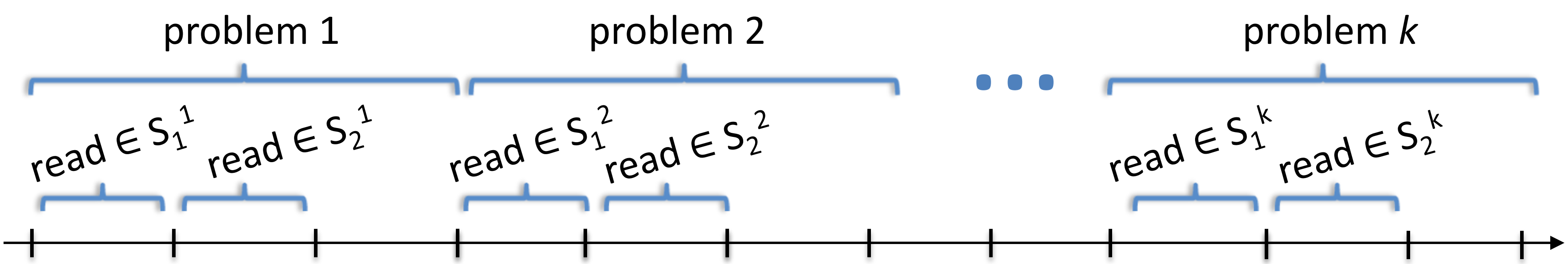
In 2^{nd} round, there are $O(\sqrt{n} \cdot \lg^2 n)$ published bits.

Direct sum: Pred(n, u) = k × Pred(n/k, u/k)



$k \gg \sqrt{n} \cdot \lg^2 n \quad \Rightarrow$ With $O(\sqrt{n} \cdot \lg^2 n)$ public bits, most sub-problems are still hard.

New Induction Plan



Main Lemma: Fix algorithm to read from a set of $(nk)^{½}$ cells

"Proof":

- problem j is nice if $(\exists)\alpha$: $|S_\alpha^j| \leq (n/k)^{½}$

  $\Rightarrow$ fix hi-part in problem j t

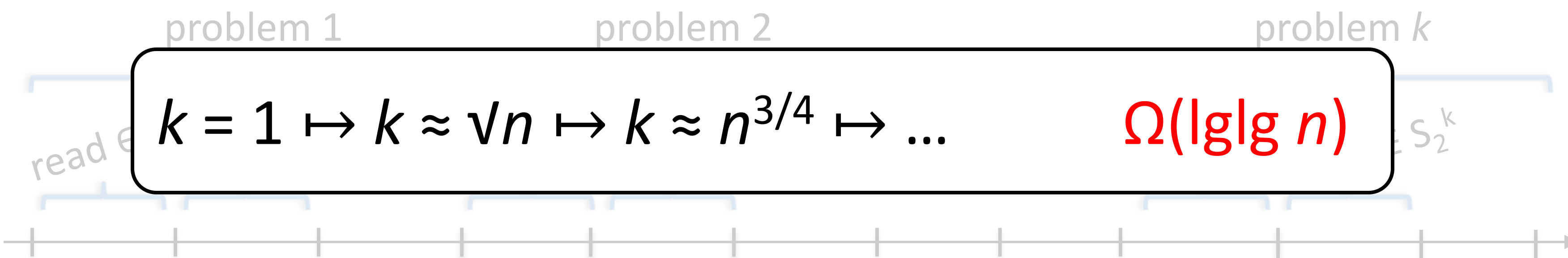
- problem j is not nice if $(\forall)\alpha$: $|S_\alpha^j| > (n/k)^{½}$

  $\Rightarrow$ choose T to hit all suc

$k \cdot (n/k)^{½} = (nk)^{½}$

$|T| \approx n \,/\, (n/k)^{½} = (nk)^{½}$

New Induction Plan



$k = 1 \mapsto k \approx \sqrt{n} \mapsto k \approx n^{3/4} \mapsto \dots$ $\Omega(\lg\lg n)$

Main Lemma**:** Fix algorithm to read from a set of $(nk)^{1/2}$ cells

"Proof":

- problem j is nice if $(\exists)\alpha$: $|S_\alpha^j| \le (n/k)^{1/2}$

  $\Rightarrow$ fix hi-part in problem j to α
- problem j is not nice if $(\forall)\alpha$: $|S_\alpha^j| > (n/k)^{1/2}$

  $\Rightarrow$ choose T to hit all such S_α^j

Main Lemma: "Proof" $\mapsto$ Proof

Main Lemma: Fix algorithm to read from a set of $(nk)^{½}$ cells

- problem j is nice if $(\exists)\alpha$: $|S_\alpha^j| \le (n/k)^{½}$
  $\Rightarrow$ fix hi-part in problem j to α
- problem j is not nice if $(\forall)\alpha$: $|S_\alpha^j| > (n/k)^{½}$
  $\Rightarrow$ choose T to hit all such S_α^j

But: Published bits = f(database)

1st cell read by query = f(published bits)

Main Lemma: “Proof” $\mapsto$ Proof

New claim. We can publish $(nk)^{½}$ cells such that:
Pr[random query reads a published cell] ≥ 1/100

Induction: If initial query time < (lglg n)/100
$\Rightarrow$ at the end E[query time] < 0 $\Rightarrow$ contradiction

But why does is work?

Proof:

- Publish random sample T = { $(nk)^{½}$ cells }
- For each problem j where lo(q) is relevant (fixed hi=α)
  publish S_α^j only if $|S_\alpha^j| \le (n/k)^{½}$

An Encoding Argument

Assume Pr[random query reads a published cell] < 1/100

Use data structure to encode $A[1..k] \in \{0,1\}^k$ with $<k$ bits.

1. Choose one random query/subproblem: $q_1, q_2, \ldots, q_k$

2. Choose random database:

   $A[j]=0 \Rightarrow lo(q_j)$ is relevant in problem j

   $A[j]=1 \Rightarrow hi(q_j)$ is relevant in problem j

3. Encode published bits → **o(k) bits**

4. Decoder classifies queries:

   when $|S_{hi(qj)}^{j}| \le (n/k)^{½}$ query is ☺ iff A[j]=0

   when $|S_{hi(qj)}^{j}| > (n/k)^{½}$ query is ☺ iff A[j]=1

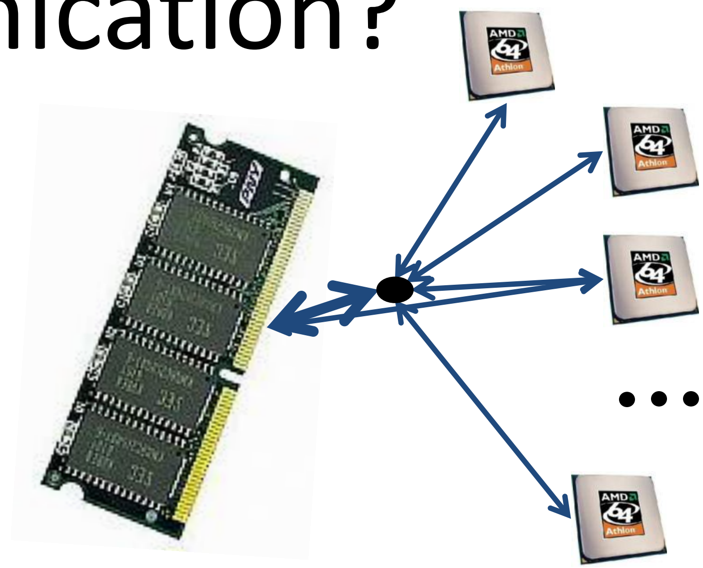
5. By assumption, E[number of ☹ queries] ≥ 99% k

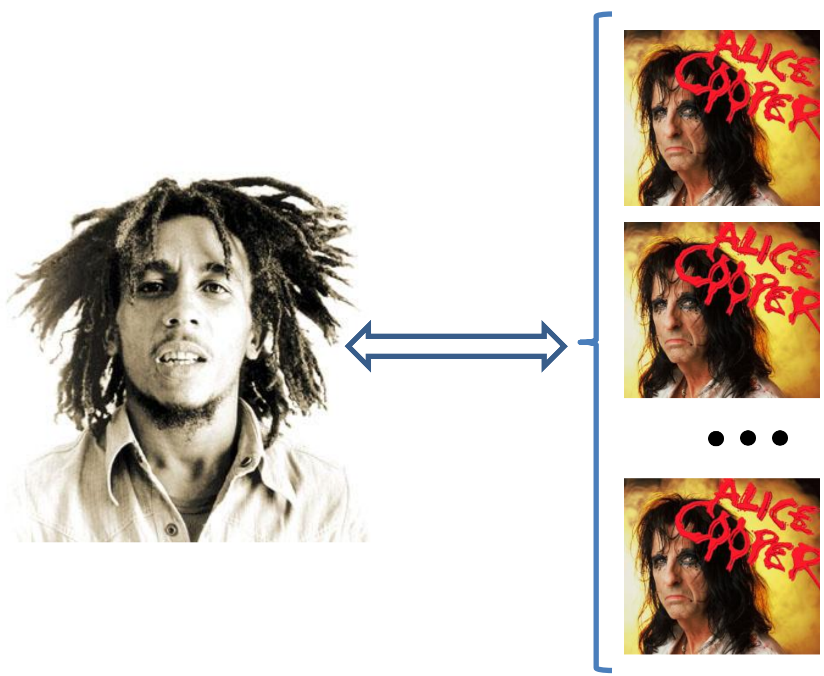
   So decoder can learn 99% of $A[1..k]$ □

Beyond Communication?

CPU → memory communication:

- one query: $\lg \mathbf{S}$
- $\mathbf{k}$ queries: $\lg \binom{\mathbf{S}}{\mathbf{k}} = \Theta\left(\mathbf{k} \lg \frac{\mathbf{S}}{\mathbf{k}}\right)$

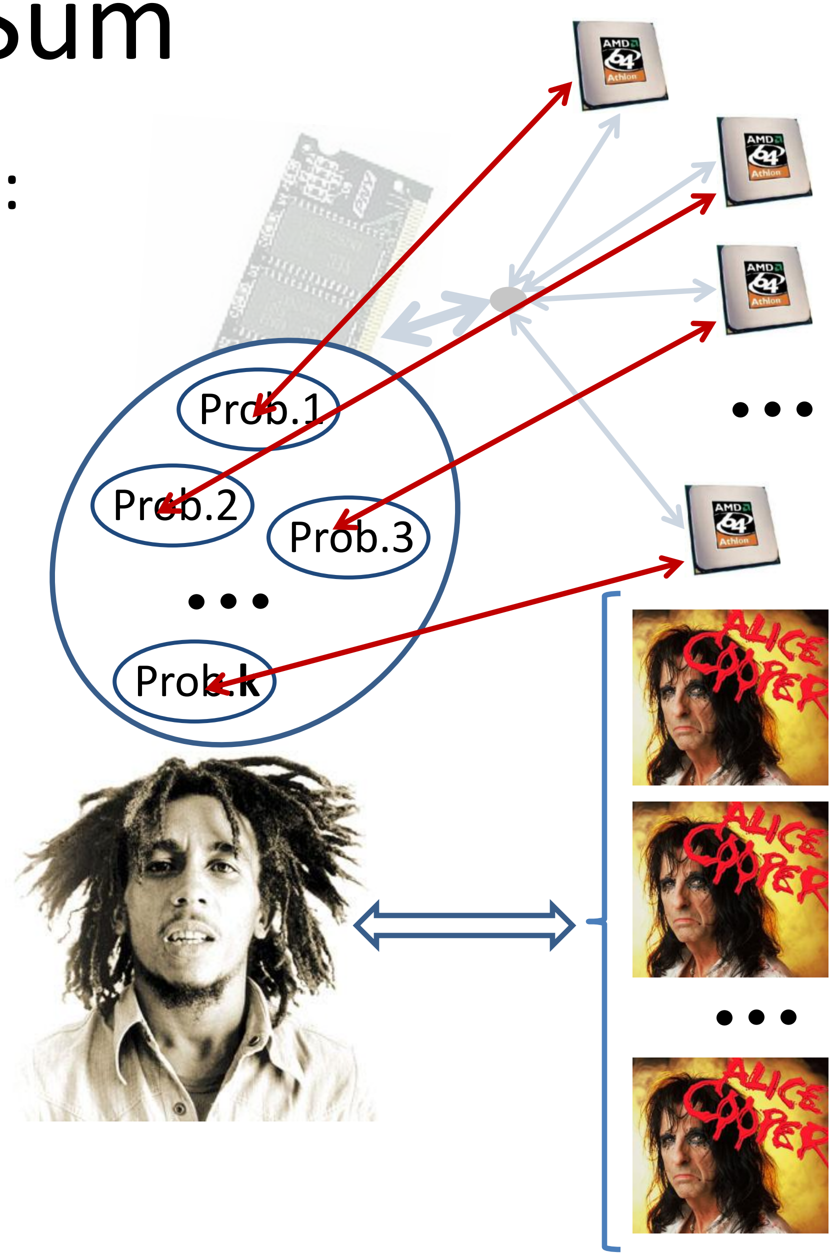




Direct Sum

CPU → memory communication:

- one query: $\lg \mathbf{S}$
- $\mathbf{k}$ queries: $\lg \binom{\mathbf{S}}{\mathbf{k}} = \Theta\left(\mathbf{k} \lg \frac{\mathbf{S}}{\mathbf{k}}\right)$

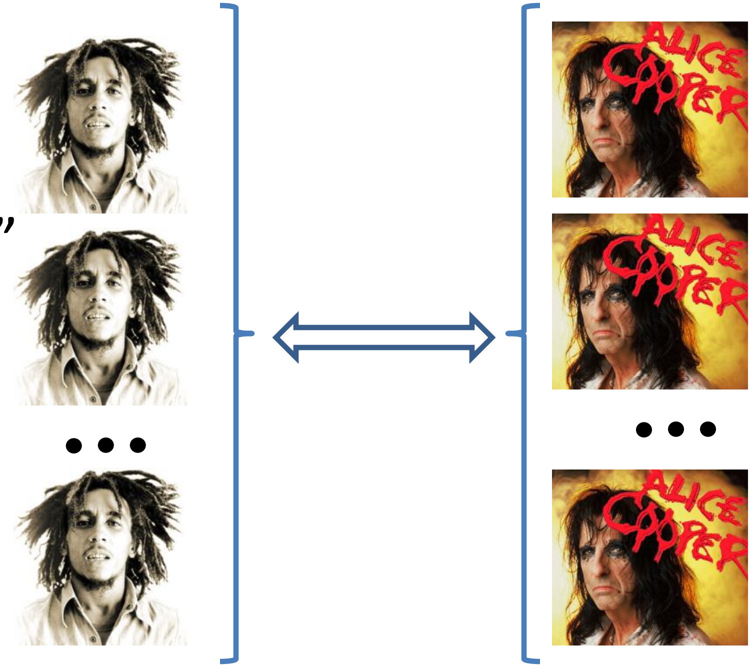


Direct Sum

CPU → memory communication:

- one query: $\lg \mathbf{S}$
- $\mathbf{k}$ queries: $\lg \binom{\mathbf{S}}{\mathbf{k}} = \Theta\left(\mathbf{k} \lg \frac{\mathbf{S}}{\mathbf{k}}\right)$

[Pătrașcu, Thorup FOCS’06]
Any richness lower bound
“Alice must send *A* or Bob must send *B*”
⇒
k×Alice must send k·*A*
or **k×Bob** must send k·*B*



Direct Sum

CPU → memory communication:

- one query: $\lg \mathbf{S}$
- **k** queries: $\lg \binom{\mathbf{S}}{\mathbf{k}} = \Theta\left(\mathbf{k} \lg \frac{\mathbf{S}}{\mathbf{k}}\right)$

Old: $t_q = \Omega(A/\lg S)$
New: $t_q = \Omega(A/\lg(S/k))$

Set $k = n/\lg^{O(1)} n$
$\Rightarrow$ $\Omega(\lg n/\lg\lg n)$ time
for space $n \cdot \lg^{O(1)} n$

[Pătrașcu, Thorup FOCS'06]
Any richness lower bound
"Alice must send A or Bob must send B"
$\Rightarrow$
k×Alice must send $k \cdot A$
or **k×Bob** must send $k \cdot B$

$\Omega(\lg n/\lg\lg n)$ for space $n\lg^{O(1)}n$

[Pătrașcu, Thorup FOCS'06]	nearest neighbor
[Pătrașcu STOC'07]	range counting
[Pătrașcu FOCS'08]	range reporting
[Sommer, Verbin, Yu FOCS'09]	distance oracles
[Greve, Jørgensen, Larsen, Truelsen'10]	range mode
[Jørgensen, Larsen'10]	range median

[Panigrahy, Talwar, Wieder FOCS'08] c-aprox. nearest neighbor

Also $n^{1+\Omega(1/c)}$ space for O(1) time

Classic Results

[Yao FOCS'78]

- (kind of) defines the model
- membership with low space

[Ajtai '88]

- static lower bound: predecessor search

[Fredman, Saks STOC'89]

- dynamic lower bounds: partial sums, union-find

Have we broken barriers?

Succinct Data Structures

Membership: n values $\in [u]$

$\Rightarrow$ optimal space $\quad H = \lg(u \text{ choose } n)$ bits

Space: H + *redundancy*

What is the redundancy / time trade-off?

[Pagh'99] membership $\mapsto$ prefix sums

[Pătrașcu FOCS'08] prefix sums: time $O(t)$,

redundancy $\approx n / \lg^t n$

Succinct Lower Bounds

[Gál, Miltersen '03] polynomial evaluation

$\Rightarrow$ redundancy $\times$ query time $\geq \Omega(n)$

[Golynski SODA'09] store a permutation and query $\pi(\cdot)$, $\pi^{-1}(\cdot)$

If space is $(1 + o(1)) \cdot n \lg n \Rightarrow$ query time is $\omega(1)$

[Pătrașcu, Viola SODA'10] prefix sums

For query time $t \Rightarrow$ redundancy $\geq n / \lg^t n$

The End