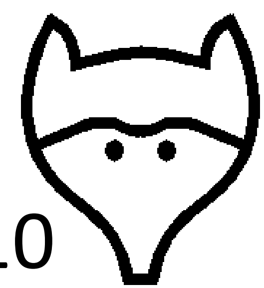
How to Grow Your Lower Bounds

Mihai Pătrașcu

at&t



Tutorial, FOCS'10

A Personal Story

MIT freshman, 2002

What problem could I work on?

P vs. NP

… half year and no solution later

How far did you guys get, anyway?

What lower bounds can we prove?

"Partial Sums" problem:

Maintain an array A[n] under:

update(k, Δ): A[k] = Δ

query(k): return A[1] + … + A[k]

(Augmented) Binary search trees: $t_u = t_q = O(\lg n)$

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

$\Omega(\lg n)$ not known for <u>any</u> dynamic problem

What kind of “lower bound”?

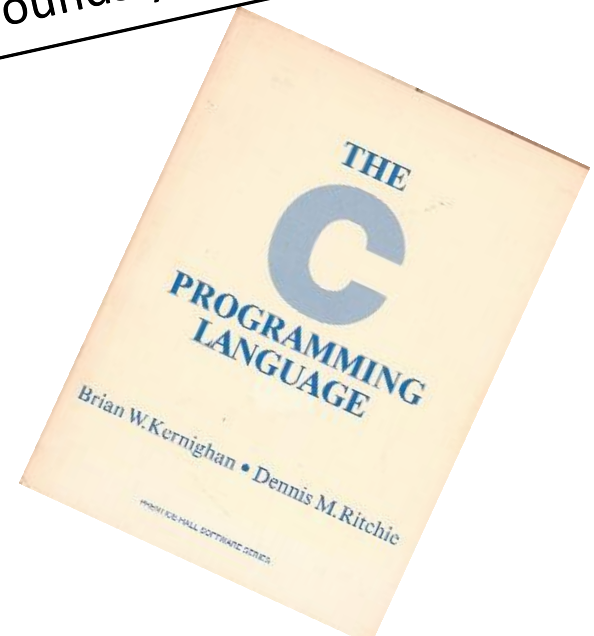
Lower bounds you can trust.™

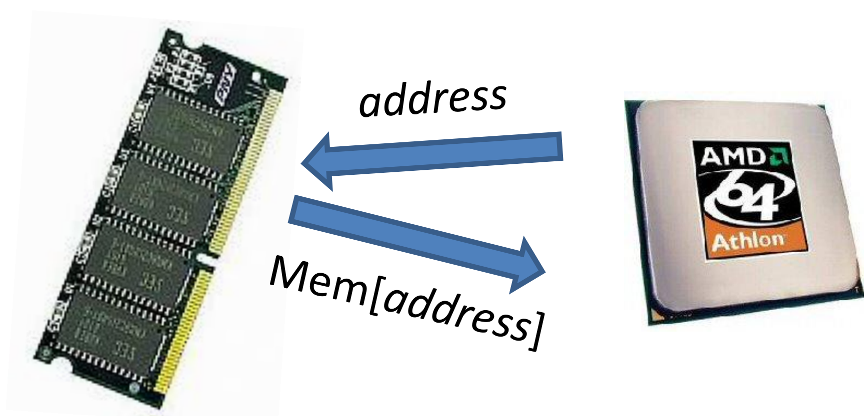
Memory: array of *S* words

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

Unit-time operations:

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





Internal state: $O(w)$ bits
Hardware: TC^0

What kind of "lower bound"?

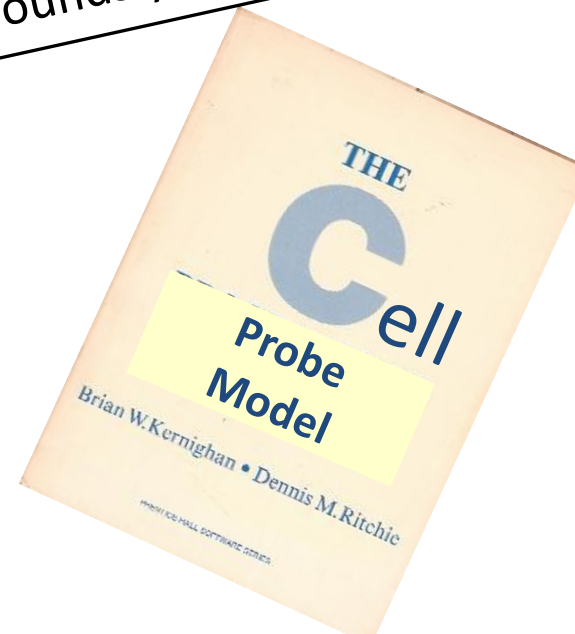
Lower bounds you can trust.™

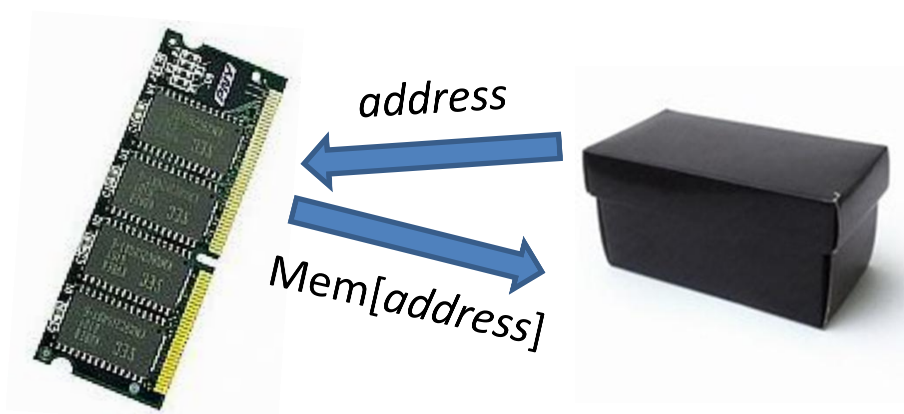
Memory: array of S words

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

Unit-time operations:

- random access to memory
- *any* function of two words (***nonuniform***)





Maintain an array A[*n*] under:

update(k, Δ): A[k] = Δ

query(k): return A[1]+…+A[k]

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

[Pătrașcu, Demaine SODA'04]

I will give the full proof.

Maintain an array A[n] under:

update(k, Δ): A[k] = Δ

query(k): return A[1]+…+A[k]

The hard instance:

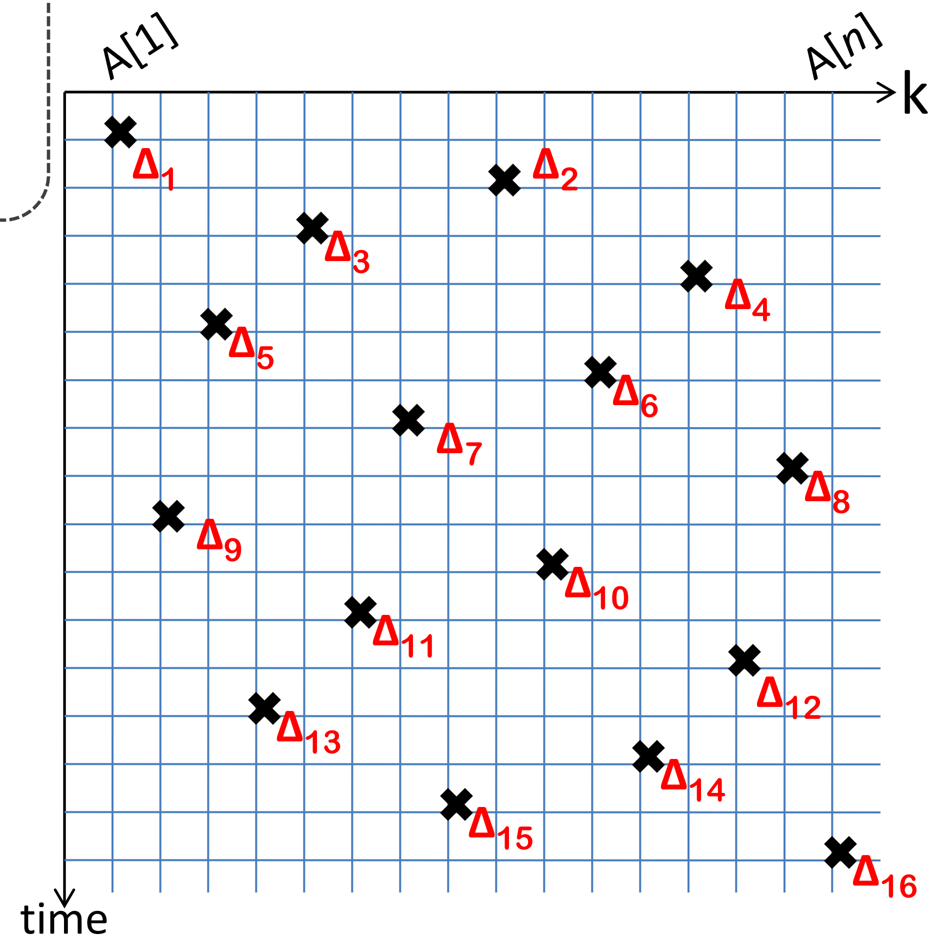
π = random permutation

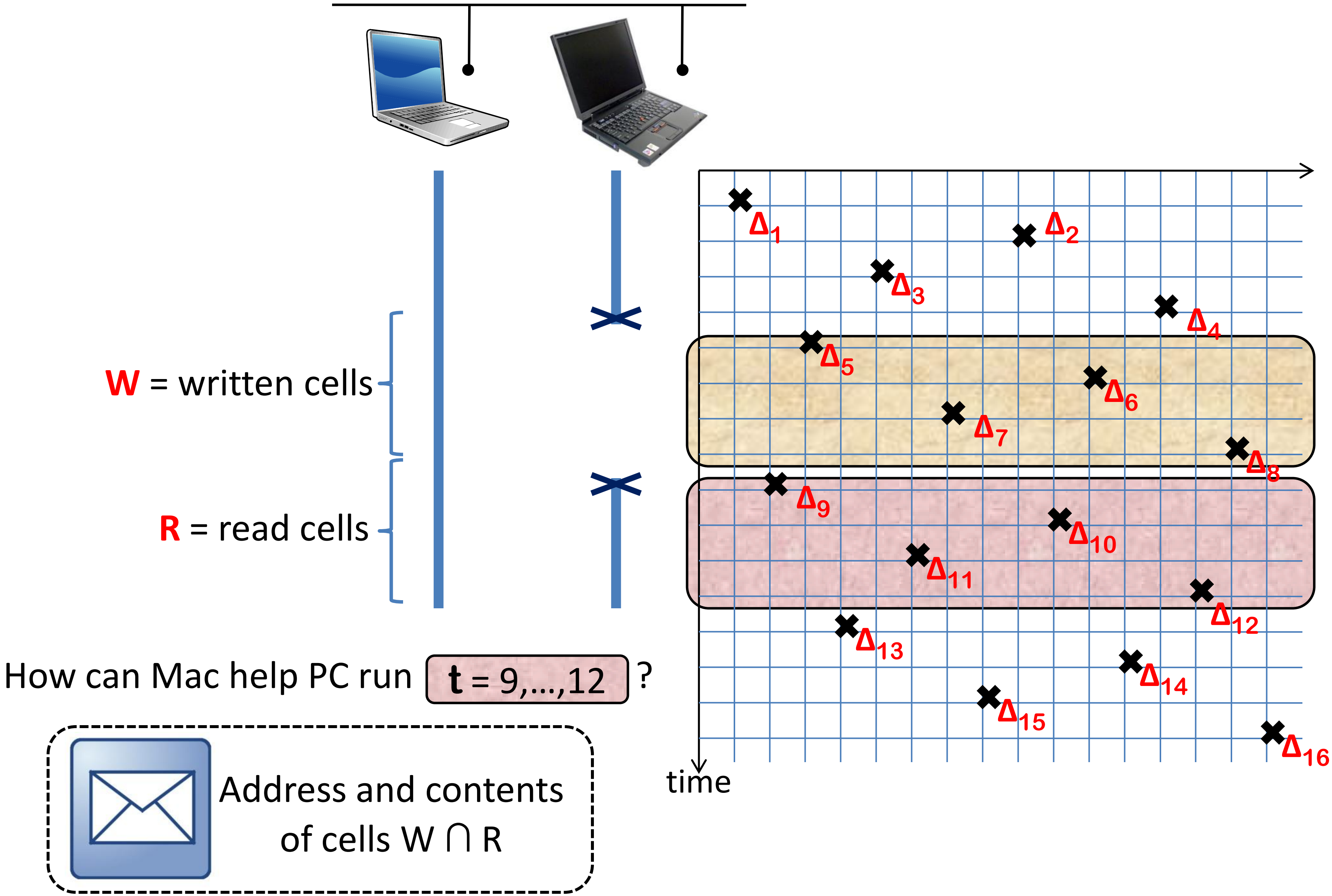
for t = 1 to n:

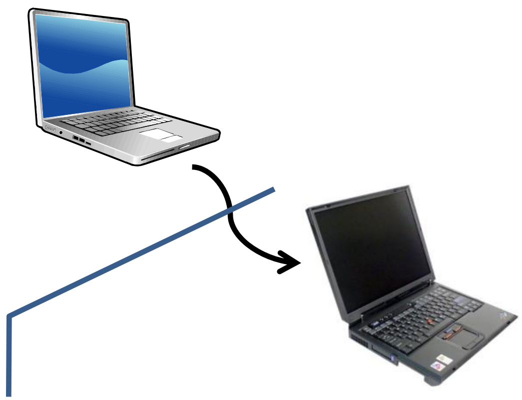
query(π(t))

Δ_t= random() mod 2^w

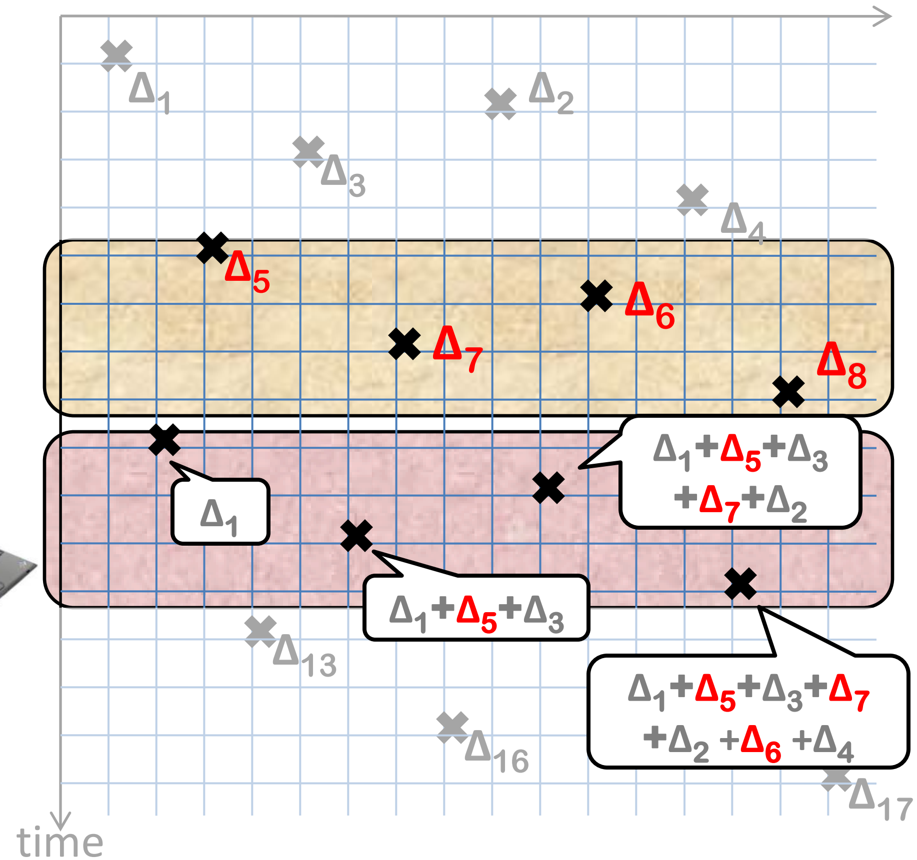
update(π(t), Δ_t)




W = written cells
R = read cells
How can Mac help PC run t = 9,…,12 ?
Address and contents of cells W ∩ R
time
Δ1
Δ2
Δ3
Δ4
Δ5
Δ6
Δ7
Δ8
Δ9
Δ10
Δ11
Δ12
Δ13
Δ14
Δ15
Δ16



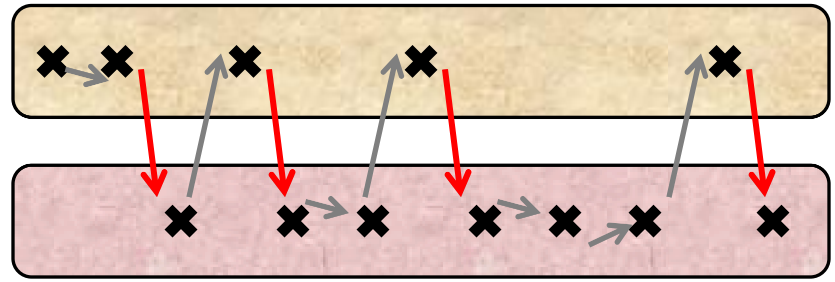
How much information needs to be transferred?



PC learns Δ_5 , $\Delta_5+\Delta_7$, $\Delta_5+\Delta_6+\Delta_7$
$\Rightarrow$ **entropy** $\geq$ 3 words

The general principle

k updates



k queries

Message entropy
$\geq w \cdot \#$ down arrows

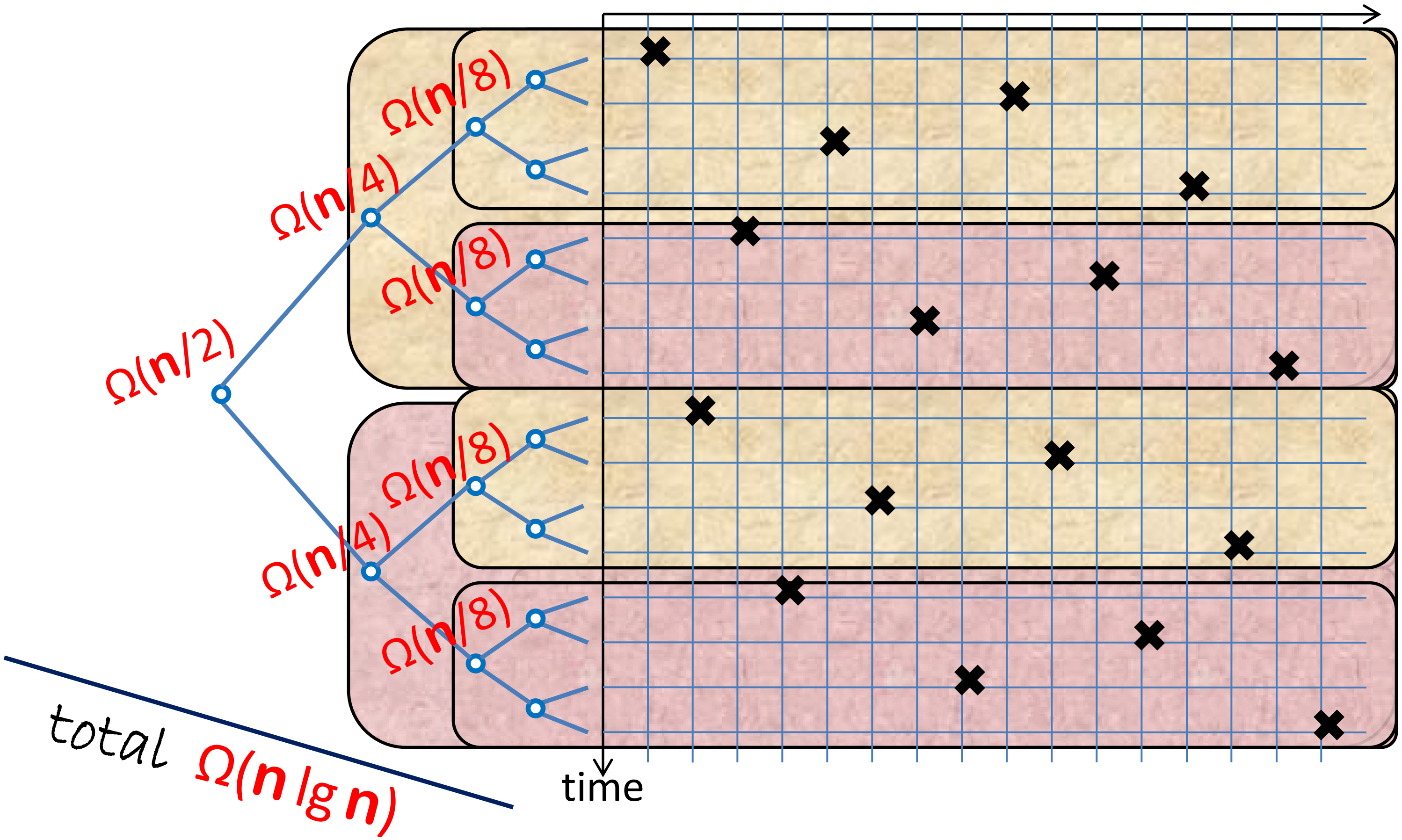
E[down arrows]
$= (2k-1) \cdot \Pr[\ \ ] \cdot \Pr[\ \ ]$
$= (2k-1) \cdot ½ \cdot ½ = \Omega(k)$

---

memory cells { * read during mauve period
* written during beige period } = Ω(k)

Every read instruction counted <u>once</u>

@ lowest_common_ancestor(write time , read time)



Q.E.D.

The optimal solution for maintaining partial sums
= binary search trees

What were people trying before?

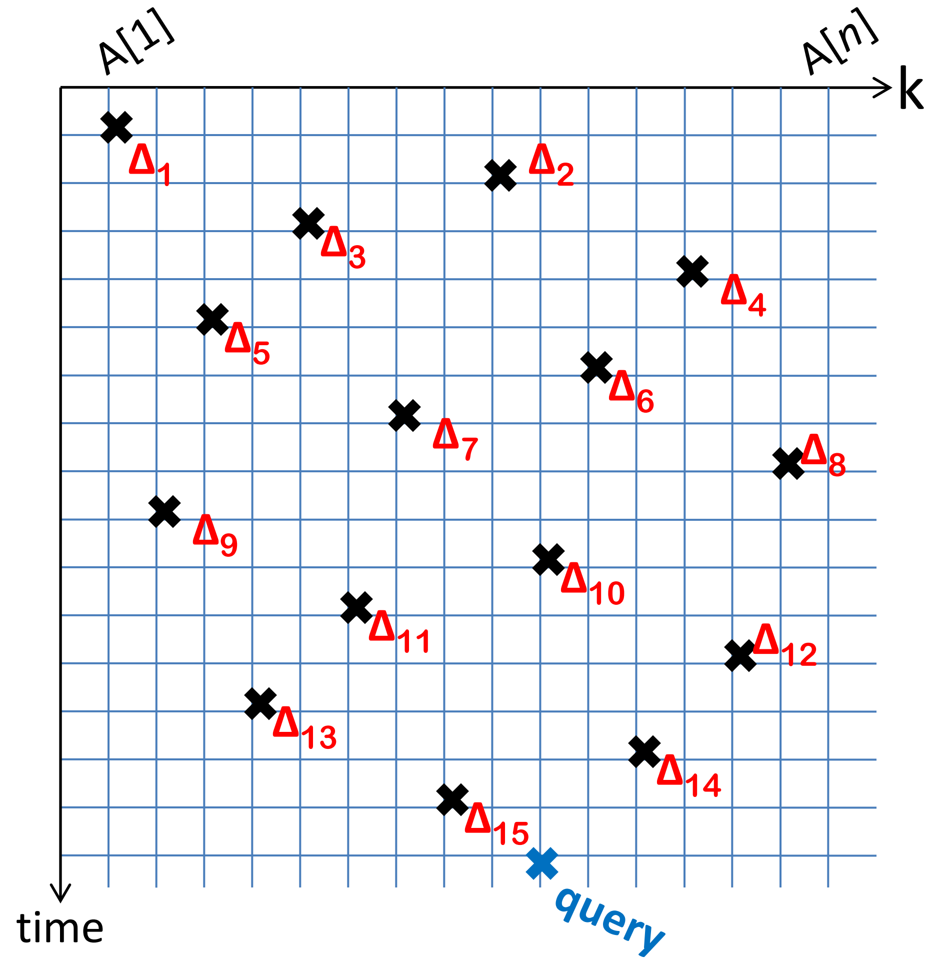
[Fredman, Saks STOC'89]
$\Omega(\lg n / \lg\lg n)$

The hard instance:

π = random permutation

for t = 1 to n:
    Δ_t= random() mod 2^w
    update(π(t), Δ_t)

query(random() mod n)



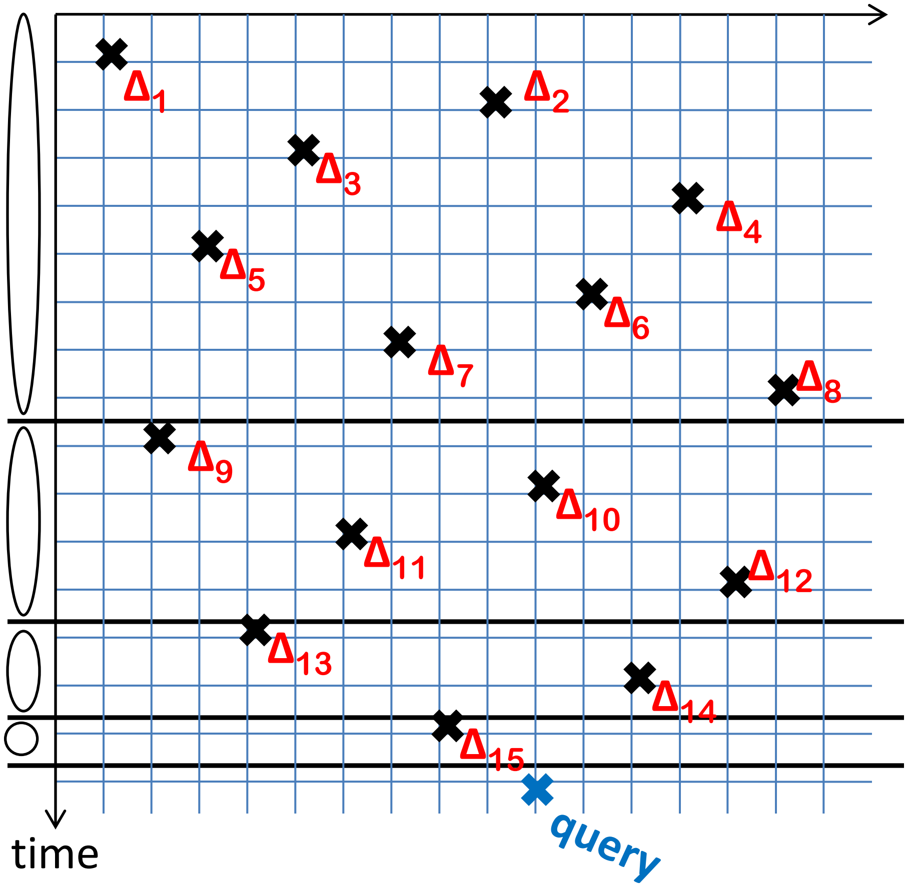
Epochs

Build ***epochs*** of $(\lg n)^i$ updates

W_i = cells last written in epoch i

Claim: E[#cells read by query from W_i] = $\Omega(1)$

$\Rightarrow E[t_q] = \Omega(\lg n / \lg\lg n)$

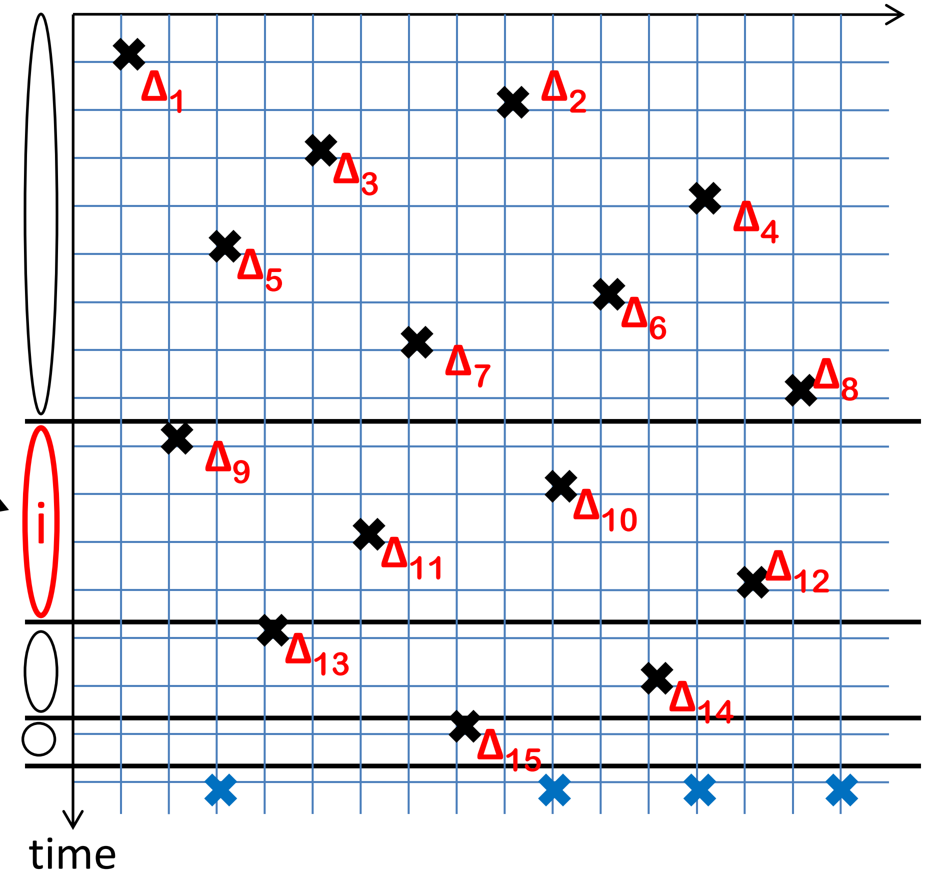


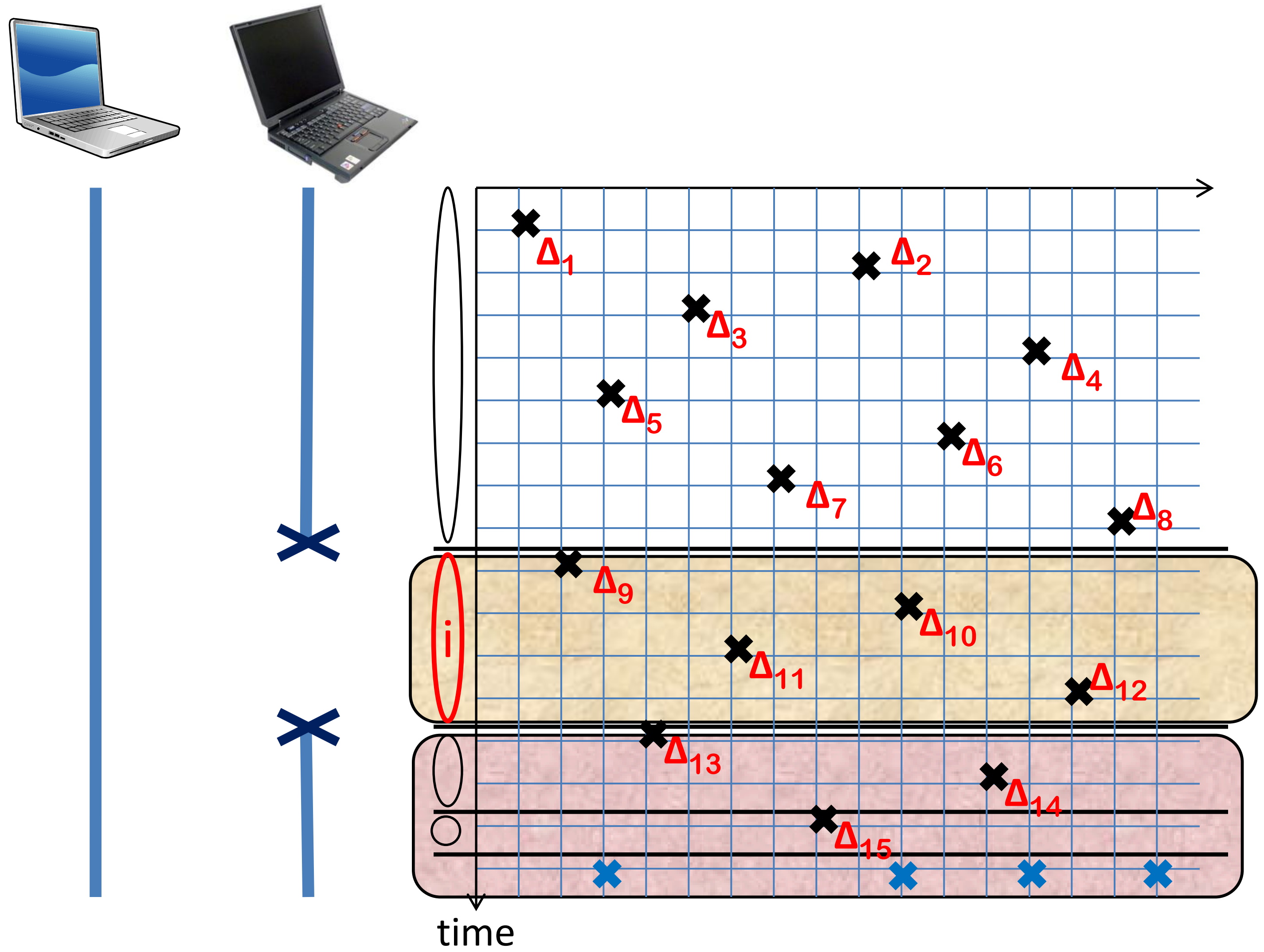
Epochs

Focus on some epoch i

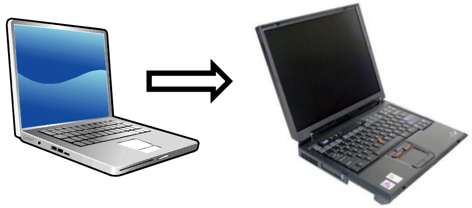
x = E[#cells read by query from W_i]

Generate $(\lg n)^i$ random queries





Δ1
Δ2
Δ3
Δ4
Δ5
Δ6
Δ7
Δ8
Δ9
Δ10
Δ11
Δ12
Δ13
Δ14
Δ15
i
time



Entropy = $\Omega(w \cdot \lg^i n)$ bits

Possible message:

$W_0 \cup W_1 \cup \ldots \cup W_{i-1}$

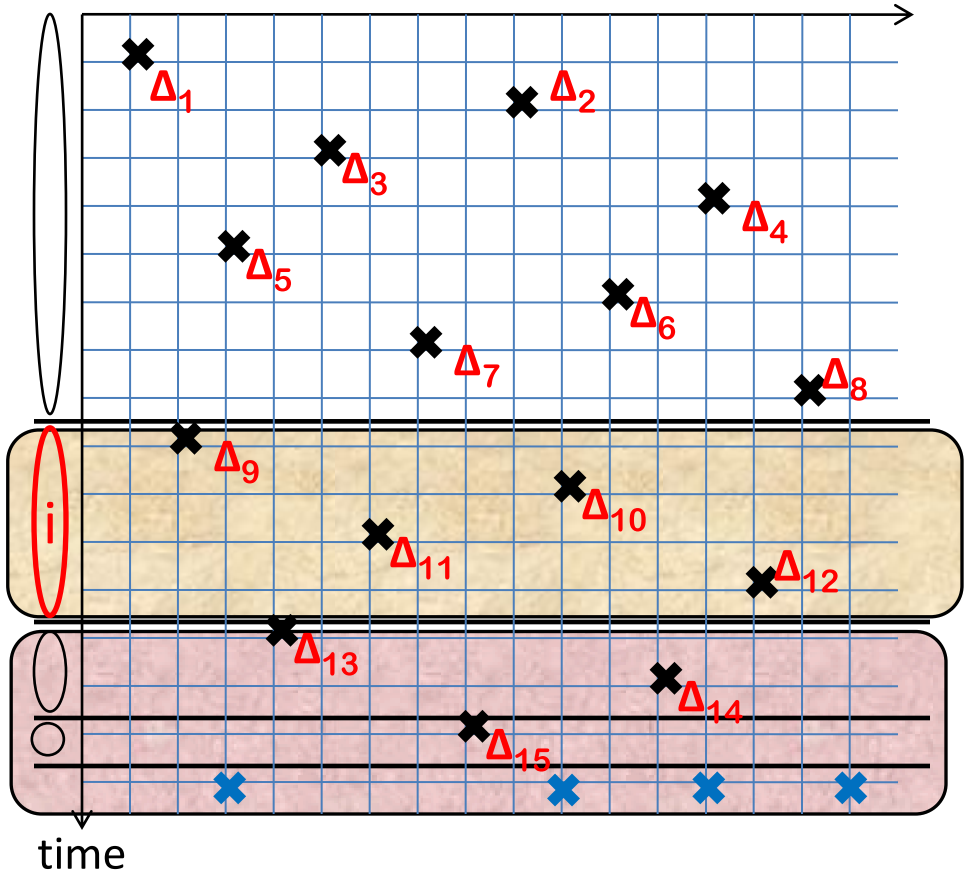
$\Sigma_{j<i} (\lg n)^j t_u = O(\lg^{i-1} n \cdot t_u)$ cells

cells read by queries from W_i

$E[\#\text{cells}] = x \cdot \lg^i n$

$\Rightarrow x = \Omega(1)$

Q.E.D.



Dynamic Lower Bounds

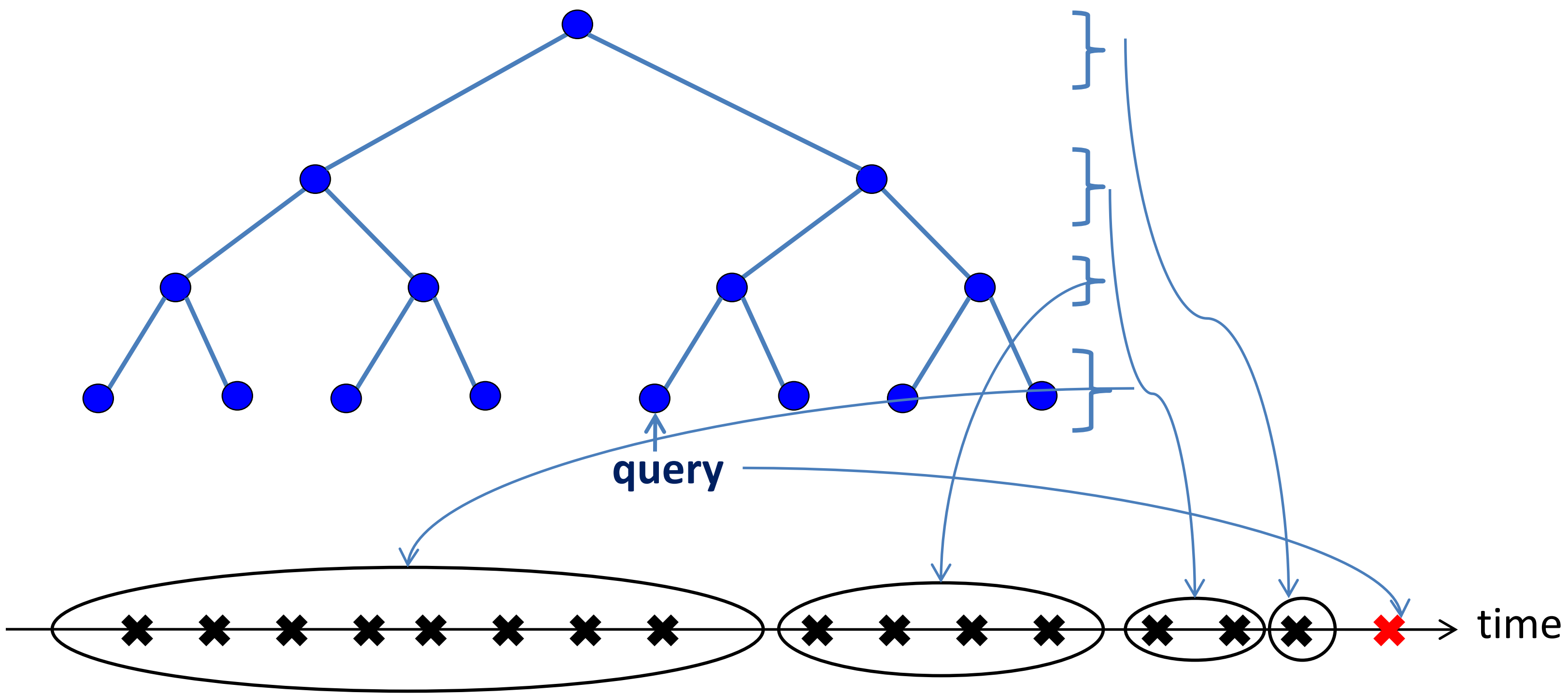
1989	[Fredman, Saks]	partial sums, union-find
1991	[Ben-Amram, Galil]	
1993	[Miltersen, Subramanian, Vitter, Tamassia]	
1996	[Husfeldt, Rauhe, Skyum]	
1998	[Fredman, Henzinger]	dynamic connectivity
	[Husfeldt, Rauhe]	nondeterminism
	[Alstrup, Husfeldt, Rauhe]	marked ancestor
1999	[Alstrup, Ben-Amram, Rauhe]	union-find
2001	[Alstrup, Husfeldt , Rauhe]	dynamic 2D NN
2004	[Pătrașcu, Demaine]	partial sums $\Omega(\lg n)$
	[Pătrașcu, Demaine]	dynamic connectivity
2005	[Pătrașcu, Tarniță]	$\Omega(\lg n)$ by epochs
2010	[Pătrașcu]	proposal for $n^{\Omega(1)}$
	[Verbin, Zhang]	buffer trees
2011	[Iacono, Pătrașcu]	buffer trees
	[Pătrașcu, Thorup]	dynamic connectivity, union-find

Marked Ancestor

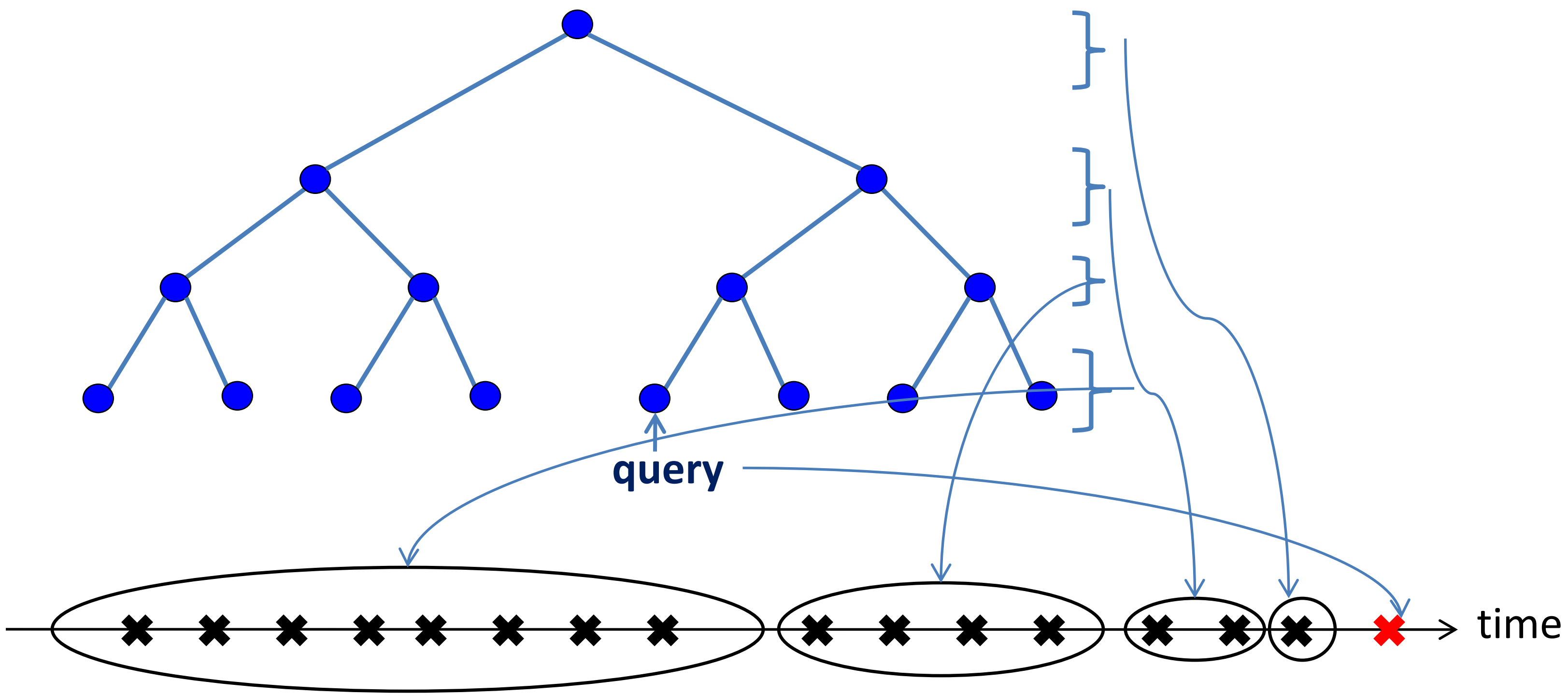
Maintain a perfect B-ary tree under:

mark(v) / unmark(v)

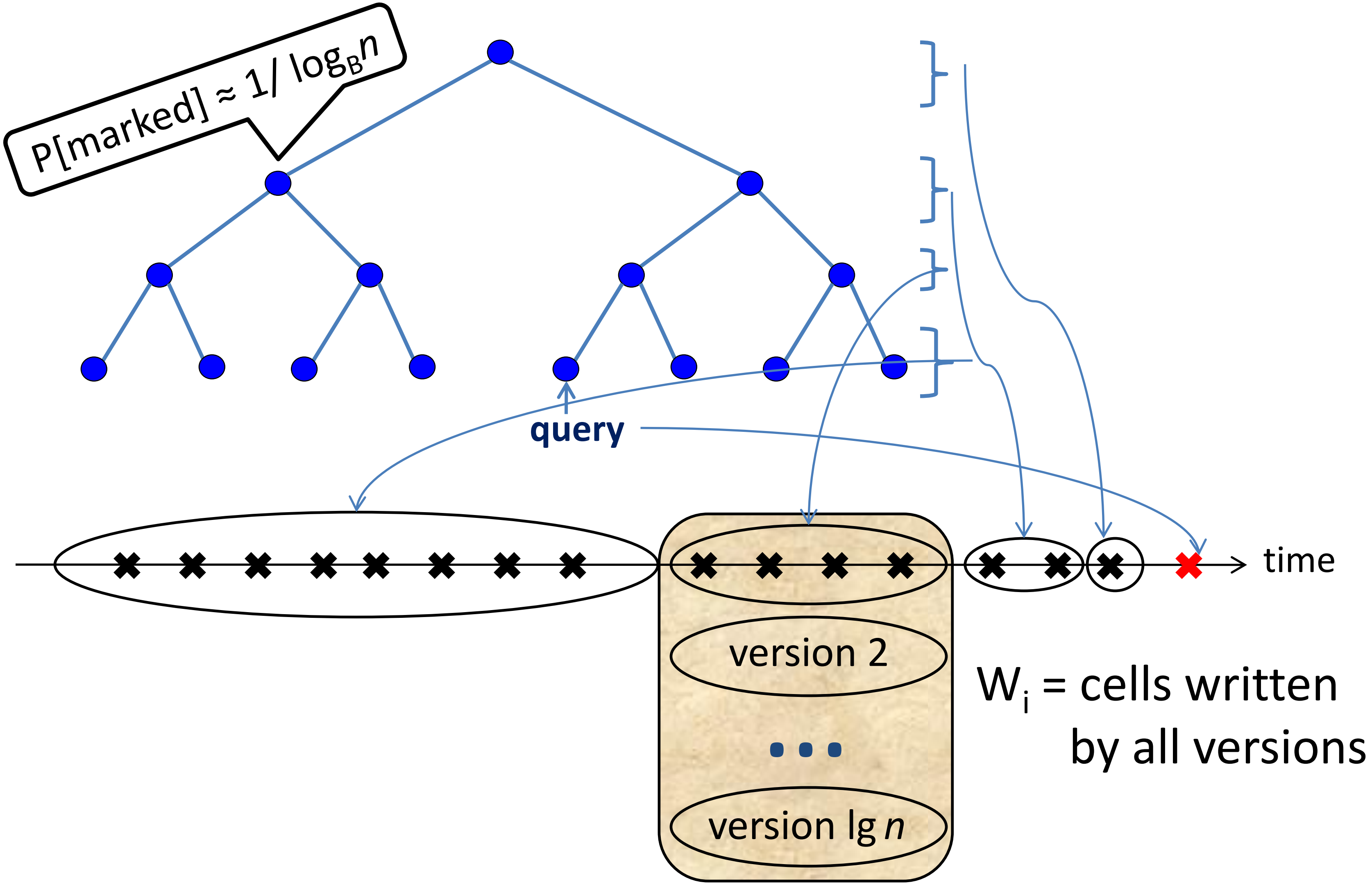
query(v): does *v* have a marked ancestor?



Marked Ancestor



Marked Ancestor

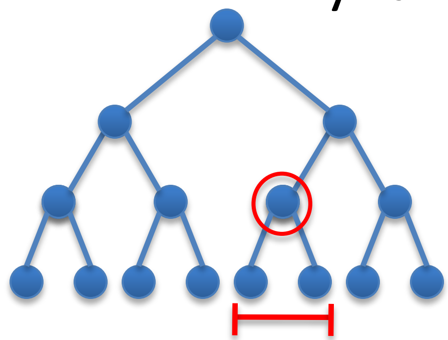


Reductions from Marked Ancestor

Dynamic 1D stabbing:

> Maintain a set of segments $S = \{ [a_1,b_1], [a_2,b_2], \ldots \}$
>
> insert / delete
>
> query(x): is $x \in [a_i, b_i]$ for some $[a_i, b_i] \in S$?

Marked ancestor $\mapsto$ Dynamic 1D stabbing

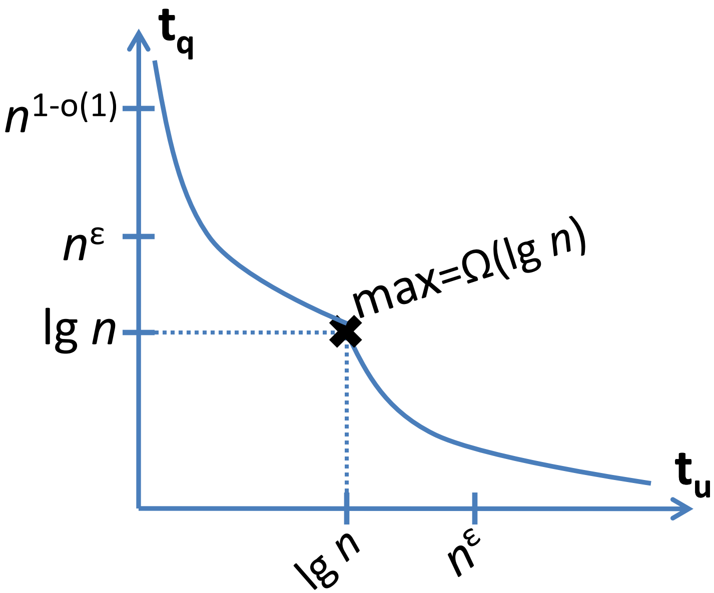


Dynamic 1D stabbing $\mapsto$ Dynamic 2D range reporting

Dynamic Lower Bounds

1989	[Fredman, Saks]	partial sums, union-find
1991	[Ben-Amram, Galil]	
1993	[Miltersen, Subramanian, Vitter, Tamassia]	
1996	[Husfeldt, Rauhe, Skyum]	
1998	[Fredman, Henzinger]	dynamic connectivity
	[Husfeldt, Rauhe]	nondeterminism
	[Alstrup, Husfeldt, Rauhe]	marked ancestor
1999	[Alstrup, Ben-Amram, Rauhe]	union-find
2001	[Alstrup, Husfeldt , Rauhe]	dynamic 2D NN
2004	[Pătrașcu, Demaine]	partial sums $\Omega(\lg n)$
	[Pătrașcu, Demaine]	dynamic connectivity
2005	[Pătrașcu, Tarniță]	$\Omega(\lg n)$ by epochs
2010	[Pătrașcu]	proposal for $n^{\Omega(1)}$
	[Verbin, Zhang]	buffer trees
2011?	[Iacono, Pătrașcu]	buffer trees
	[Pătrașcu, Thorup]	dynamic connectivity, union-find

Dynamic Lower Bounds



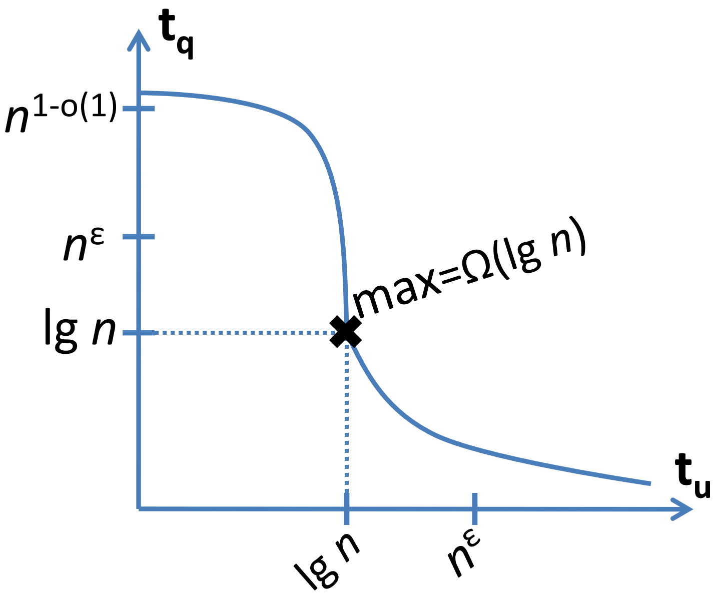
[Pătraşcu, Demaine'04]

Partial sums:

$\max\{ t_u, t_q \} = B \cdot \lg n$

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

Dynamic Lower Bounds



[Pătraşcu, Thorup'10]

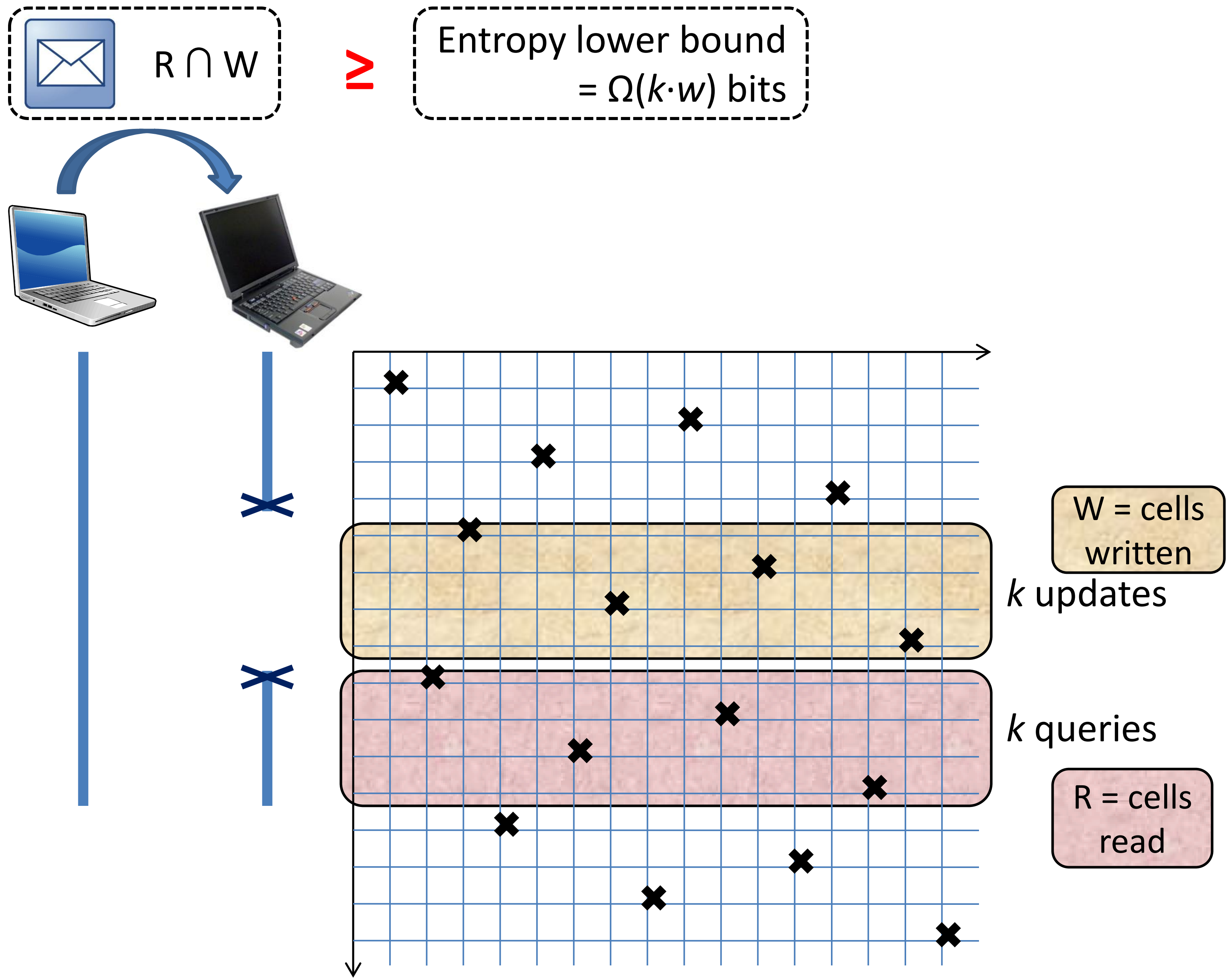
Dynamic connectivity:

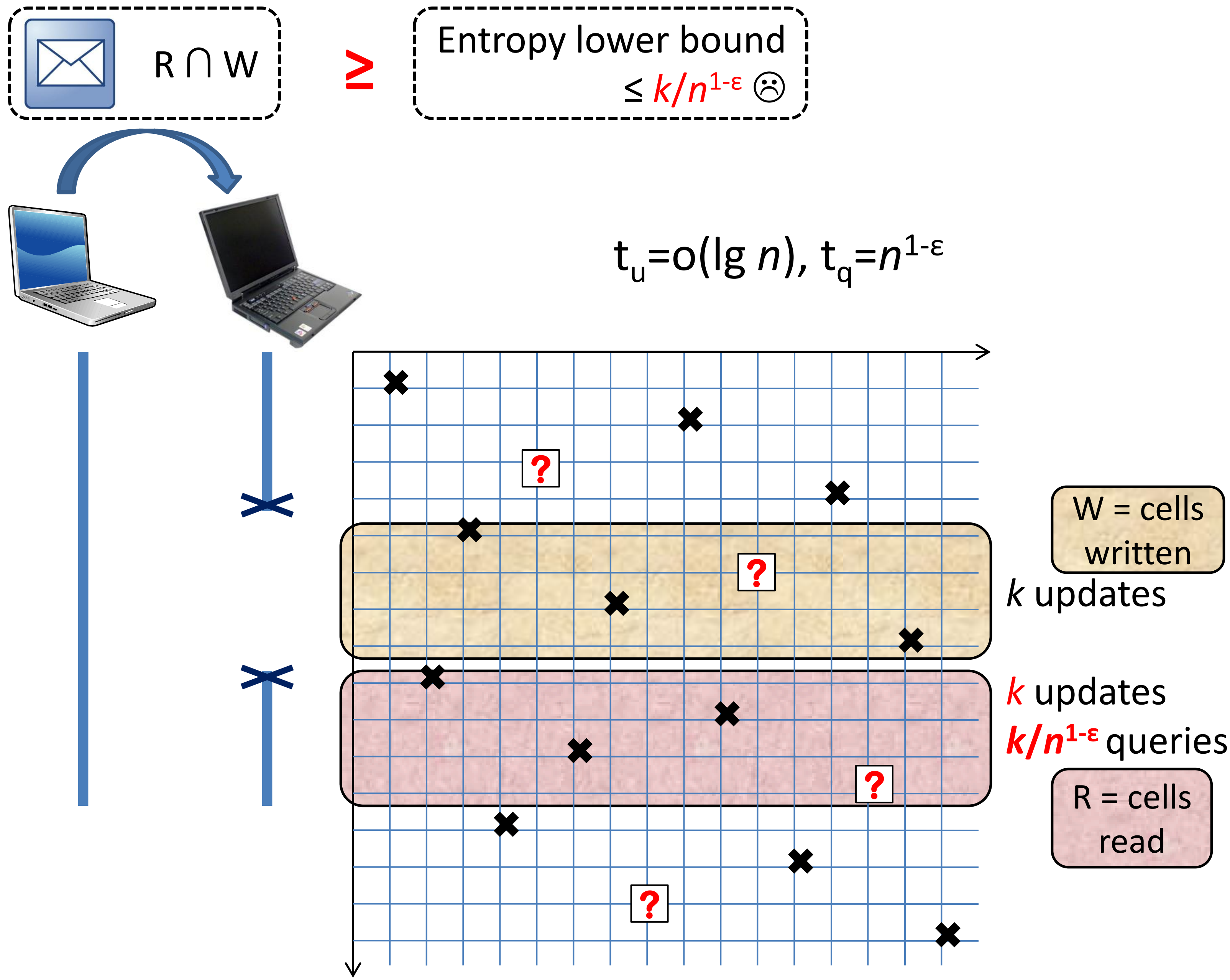
- $t_u = B \cdot \lg n$, $t_q = \log_B n$
- $t_u = o(\lg n) \Rightarrow t_q \geq n^{1-o(1)}$

Maintain an acyclic *undirected* graph under:

insert / delete edges

connected(u,v): is there a path from u to v?


R ∩ W
≥
Entropy lower bound
= Ω(k·w) bits
W = cells written
k updates
k queries
R = cells read


R ∩ W
≥
Entropy lower bound
≤ $k/n^{1-\varepsilon}$ ☹
$t_u = o(\lg n)$, $t_q = n^{1-\varepsilon}$
W = cells written
k updates
k updates
$k/n^{1-\varepsilon}$ queries
R = cells read

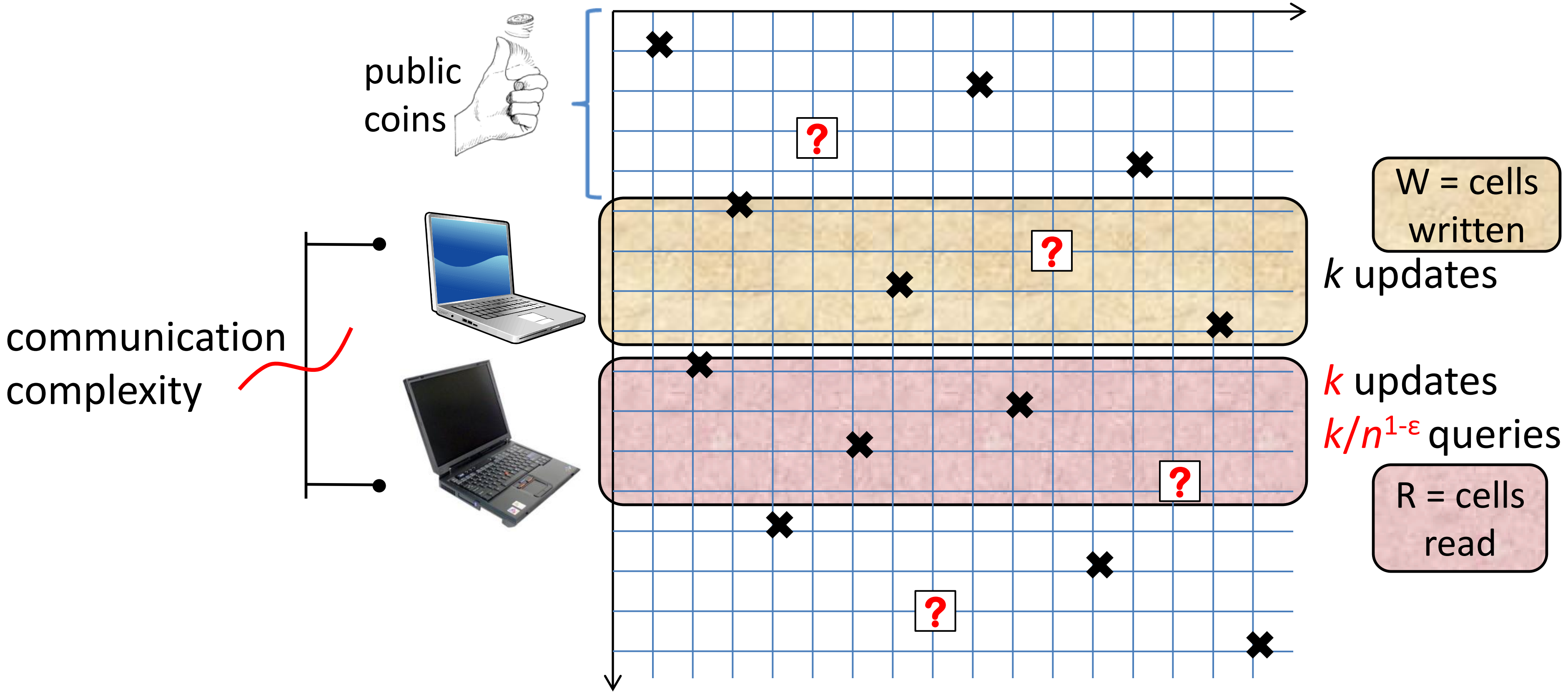
Partial sums: Mac doesn't care about PC's updates

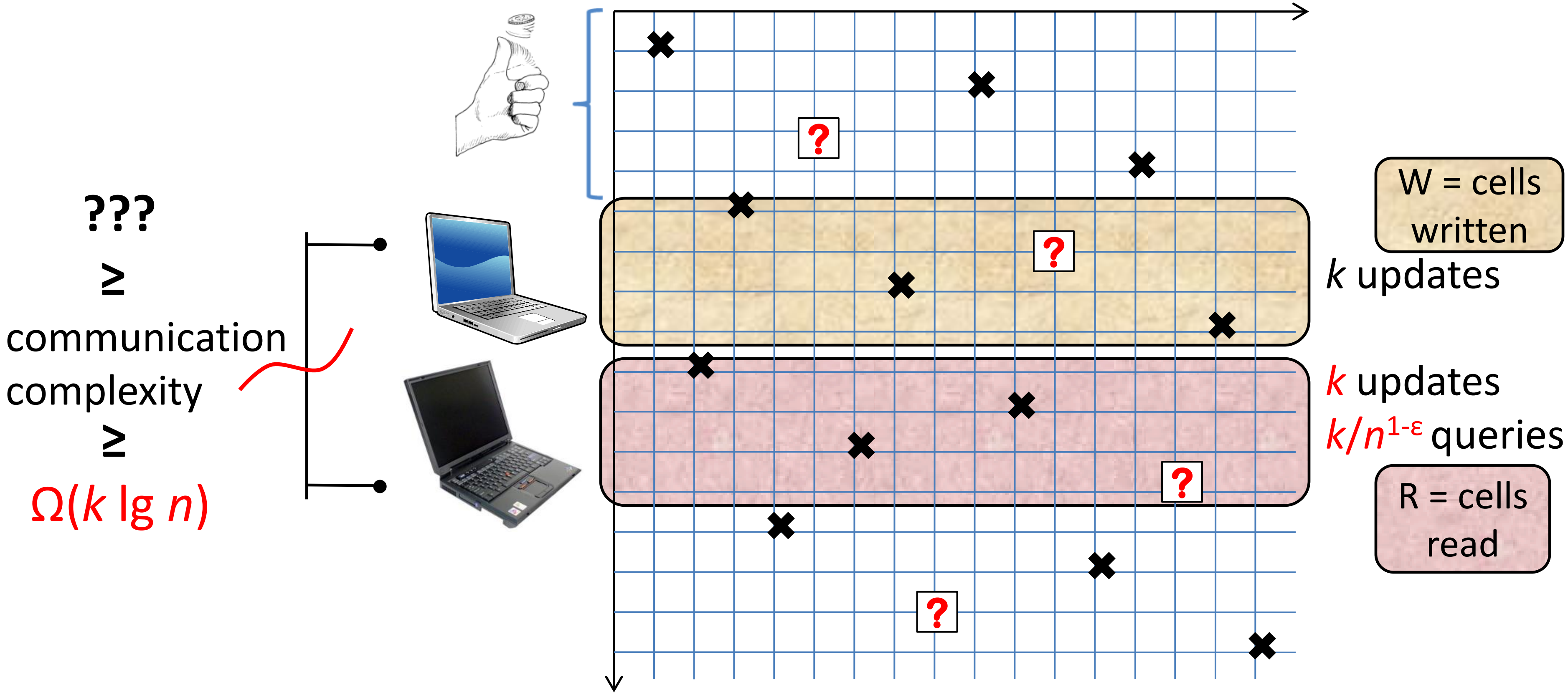
⇒ communication complexity ≈ $k/n^{1-\varepsilon}$

Dynamic connectivity:

nontrivial interaction between Mac's and PC's edges

⇒ communication complexity = $\Omega(k \lg n)$




???
≥
communication
complexity
≥
$\Omega(k \lg n)$
W = cells
written
k updates
k updates
$k/n^{1-\varepsilon}$ queries
R = cells
read

Note: $|R|, |W| = o(k \lg n)$

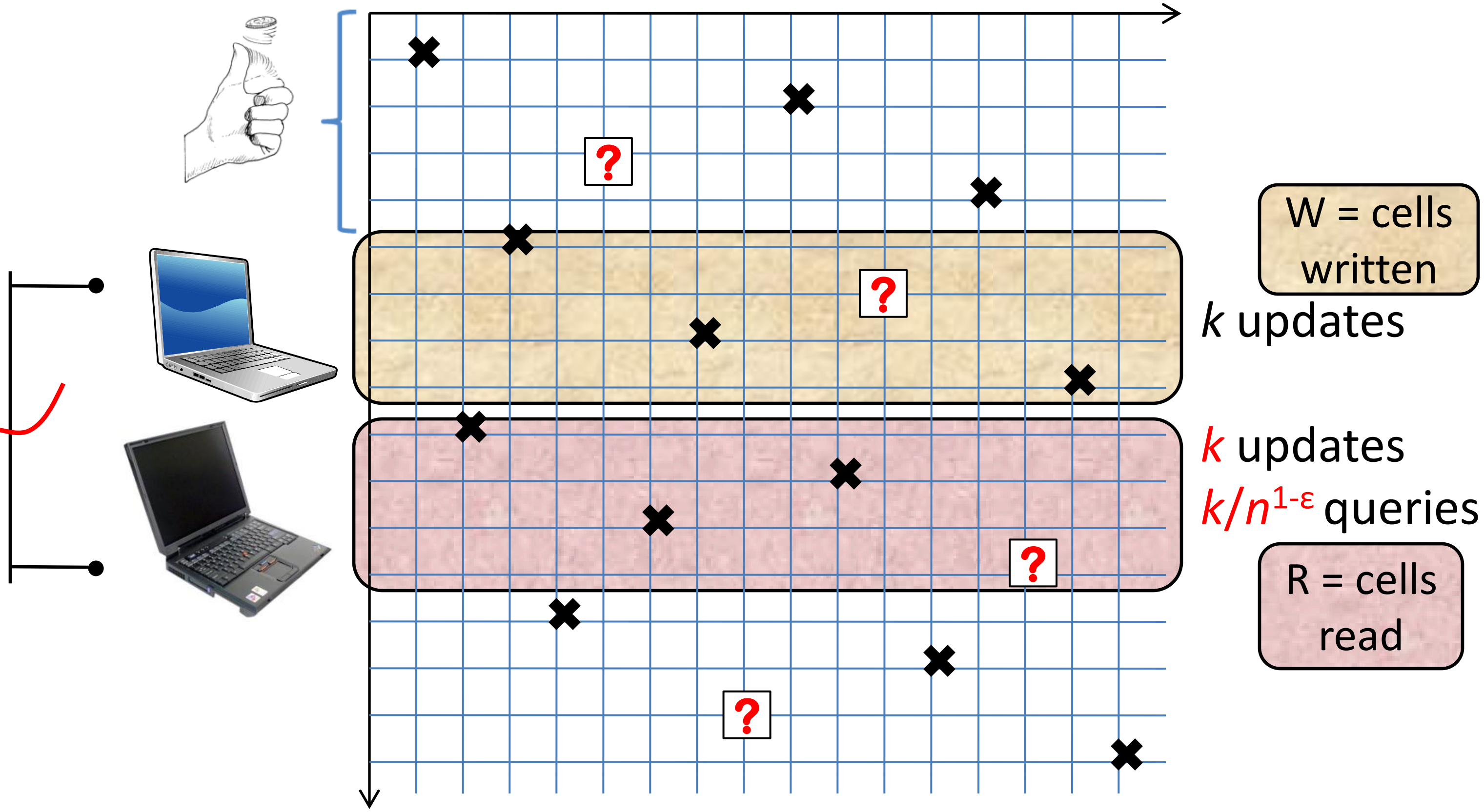
Trivial protocol: $O(|R| \cdot \lg n)$ bits

≥

communication complexity

≥

$\Omega(k \lg n)$



Note: $|R|, |W| = o(k \lg n)$

or

$|R \cap W| = \Omega(k)$

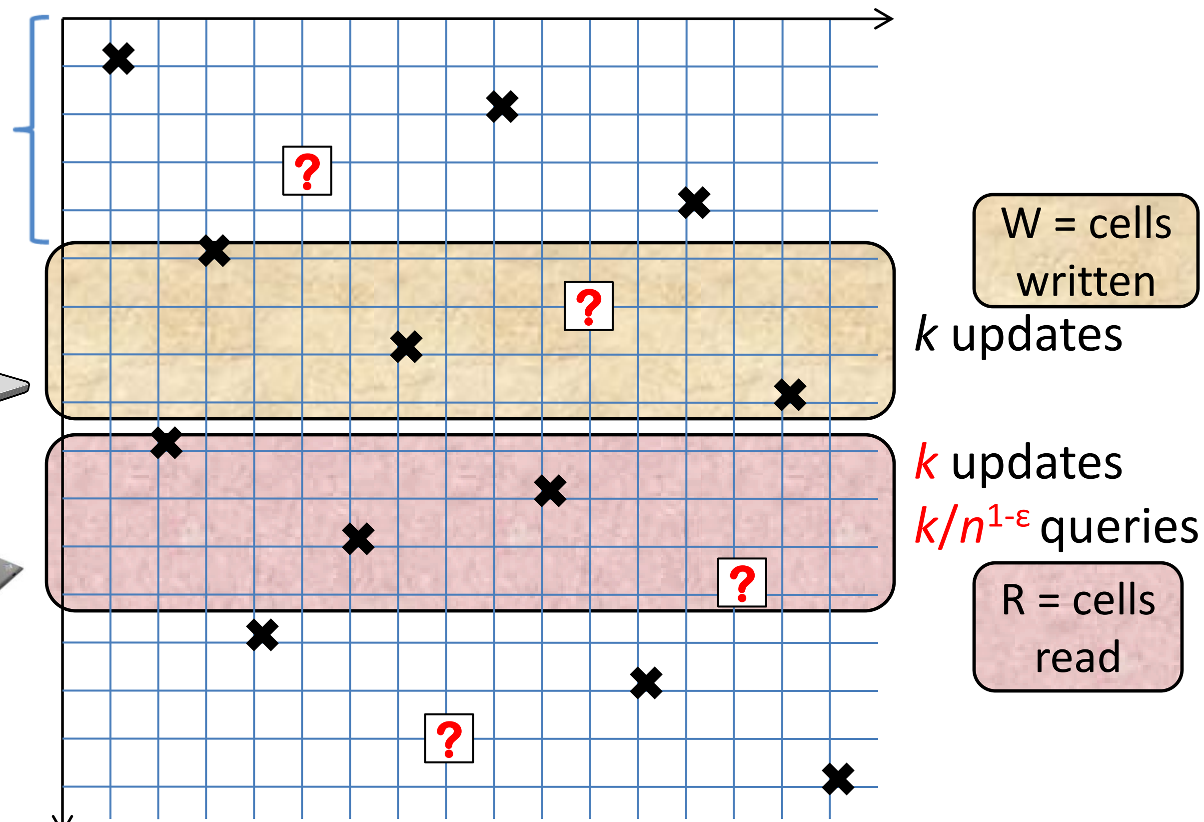
$|R \cap W| \cdot \lg n + |R| + |W|$

$\geq$

nondeterministic communication complexity

$\geq$

$\Omega(k \lg n)$



Dynamic Lower Bounds

1989	[Fredman, Saks]	partial sums, union-find
1991	[Ben-Amram, Galil]	
1993	[Miltersen, Subramanian, Vitter, Tamassia]	
1996	[Husfeldt, Rauhe, Skyum]	
1998	[Fredman, Henzinger]	dynamic connectivity
	[Husfeldt, Rauhe]	nondeterminism
	[Alstrup, Husfeldt, Rauhe]	marked ancestor
1999	[Alstrup, Ben-Amram, Rauhe]	union-find
2001	[Alstrup, Husfeldt , Rauhe]	dynamic 2D NN
2004	[Pătrașcu, Demaine]	partial sums $\Omega(\lg n)$
	[Pătrașcu, Demaine]	dynamic connectivity
2005	[Pătrașcu, Tarniță]	$\Omega(\lg n)$ by epochs
2010	[Pătrașcu]	proposal for $n^{\Omega(1)}$
	[Verbin, Zhang]	buffer trees
2011?	[Iacono, Pătrașcu]	buffer trees
	[Pătrașcu, Thorup]	dynamic connectivity, union-find

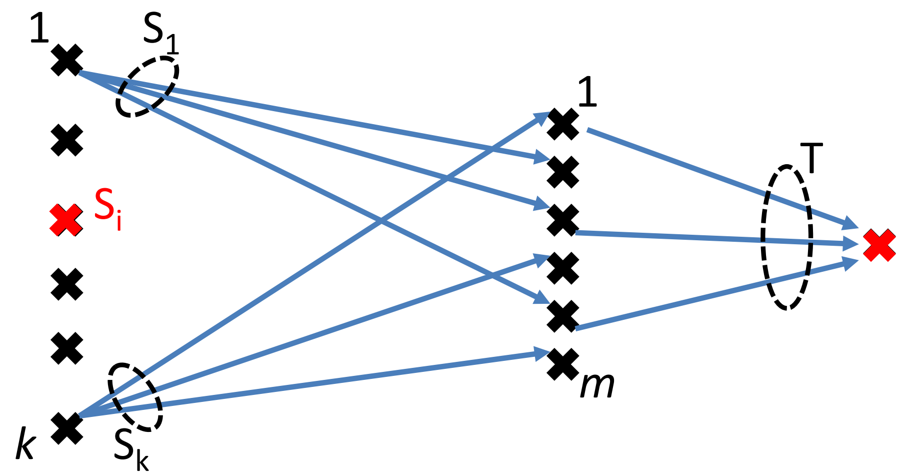
The Multiphase Problem

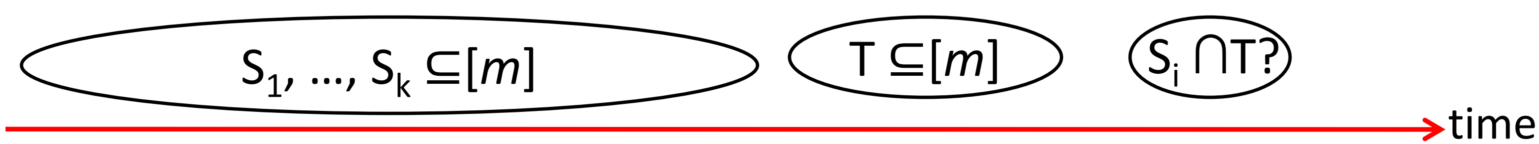
Dynamic reachability:

Maintain a directed graph under:

- insert / delete edges
- connected(u,v): ∃ path from u to v?

Hard-looking instance:

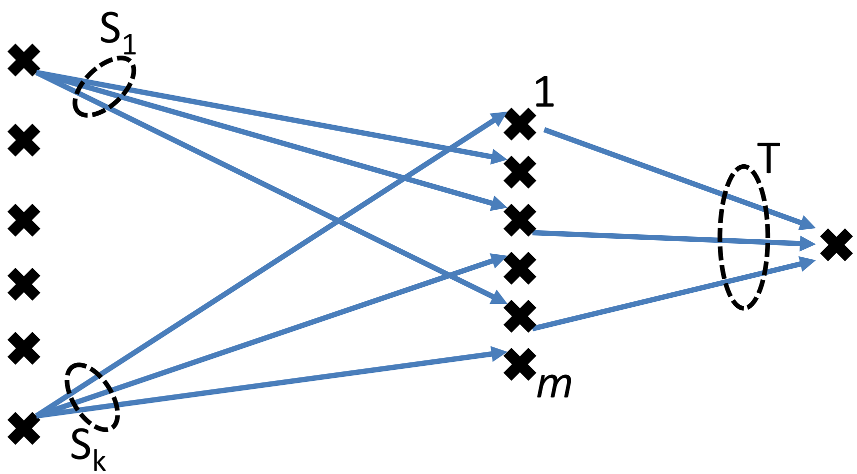


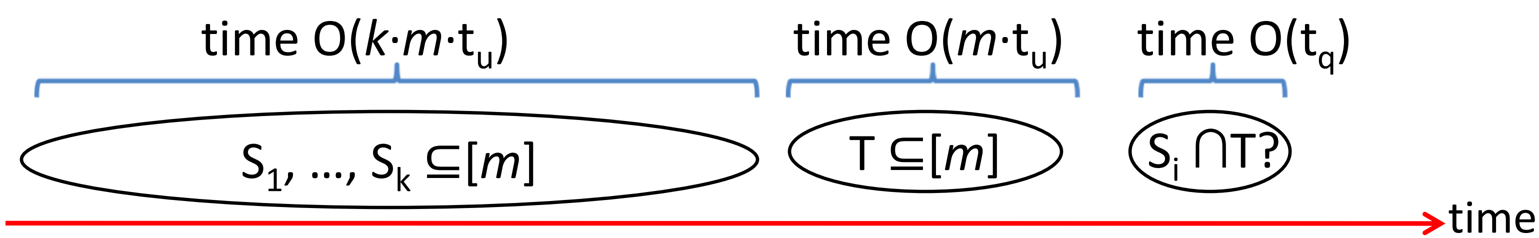


The Multiphase Problem

Conjecture: If $m \cdot t_u \ll k$, then $t_q = \Omega(m^{\varepsilon})$

Hard-looking instance:





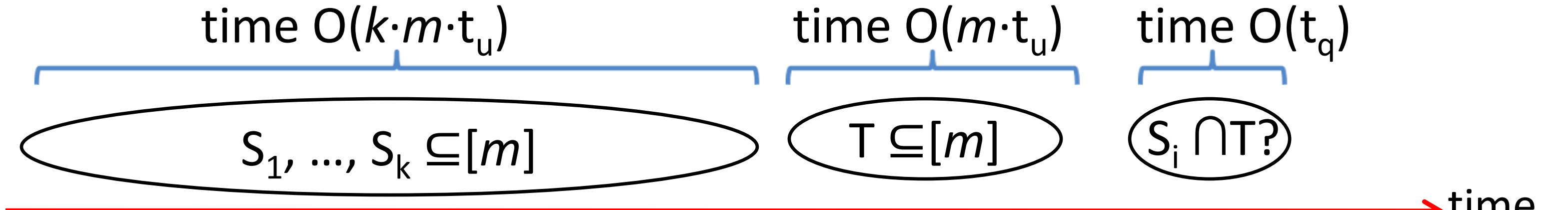
The Multiphase Problem

Conjecture: If $m \cdot t_u \ll k$, then $t_q = \Omega(m^{\varepsilon})$

Follows from the 3SUM Conjecture:

3SUM: Given S = { n numbers }, ∃ a,b,c ∈S: a+b+c = 0?

Conjecture: 3SUM requires $\Omega^*(n^2)$ time

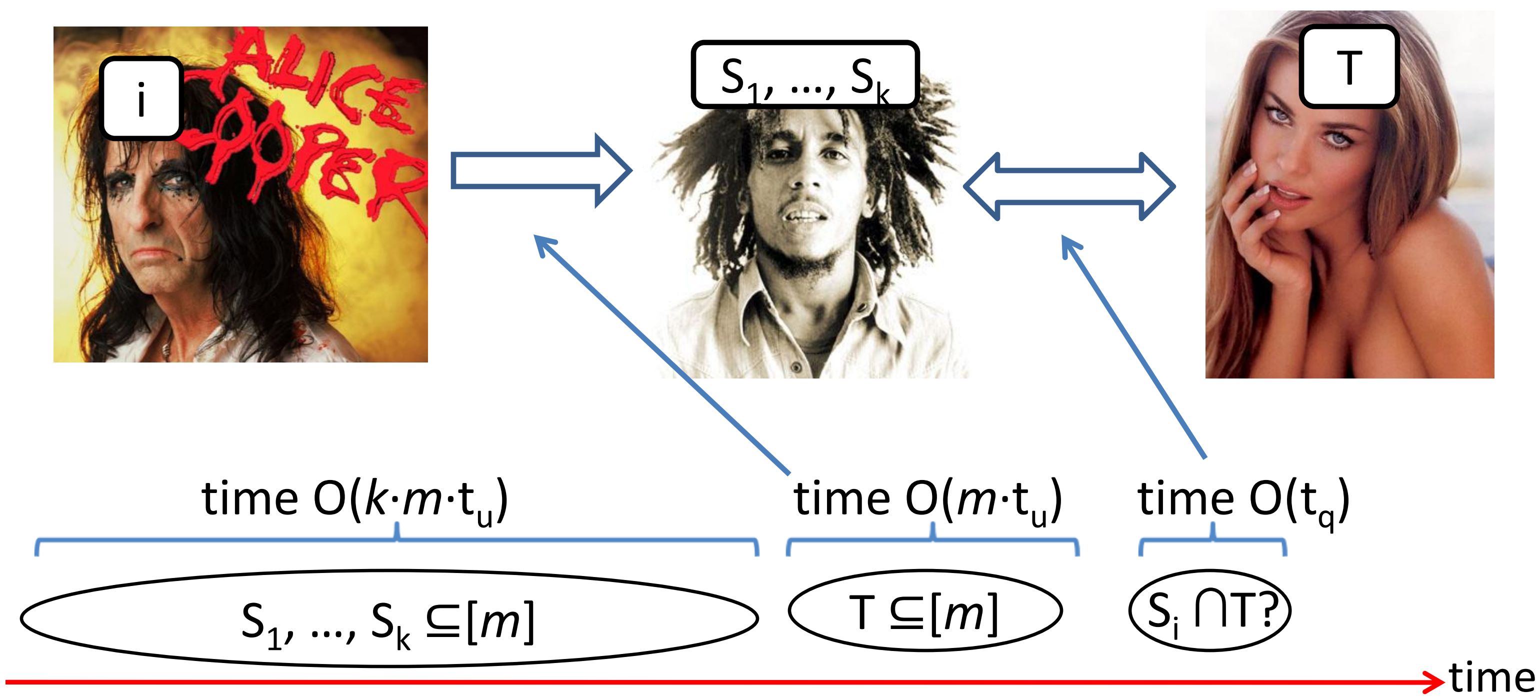


The Multiphase Problem

Conjecture: If $m \cdot t_u \ll k$, then $t_q = \Omega(m^{\varepsilon})$

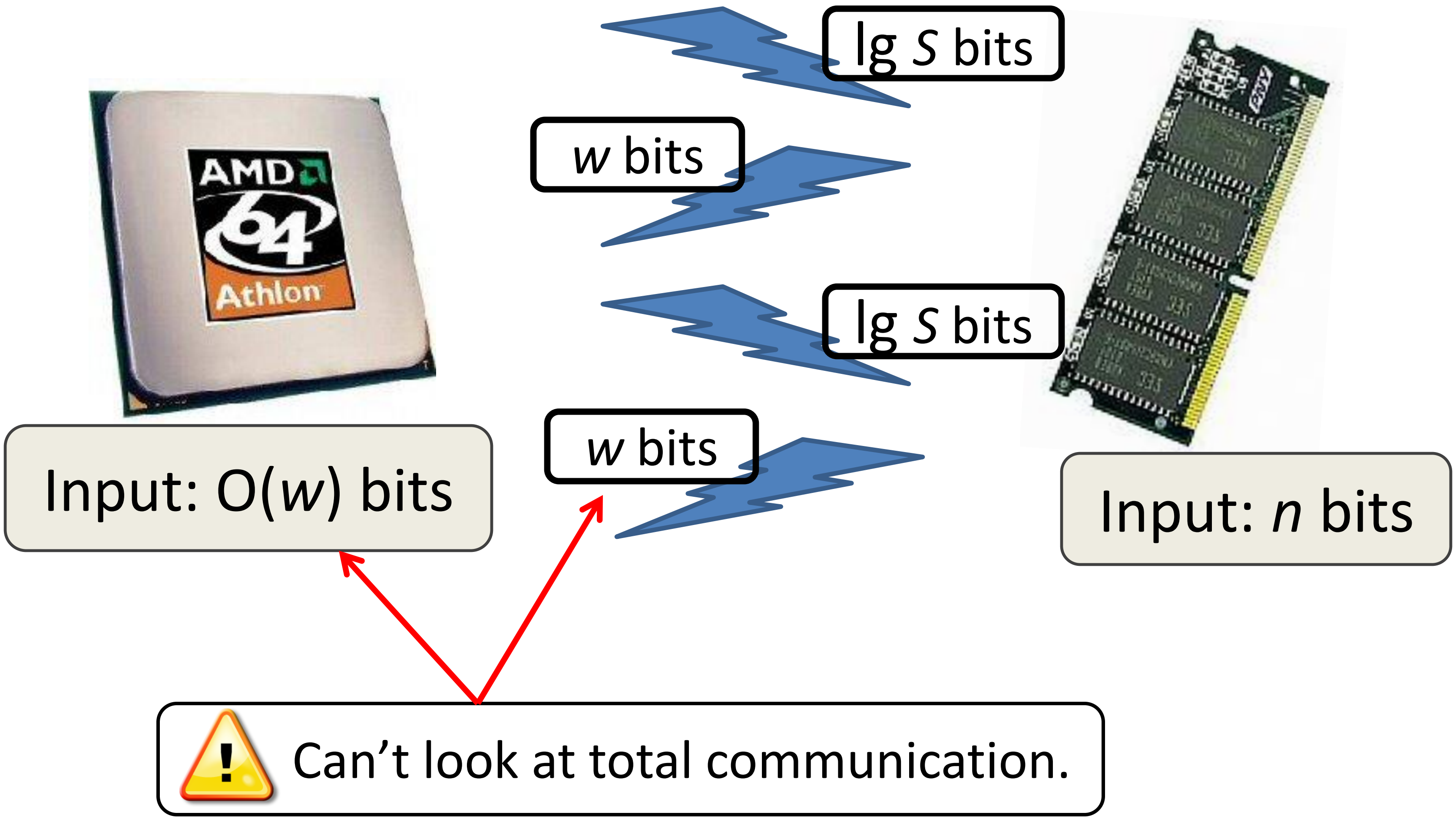
Attack on unconditional proof:

3-party number-on-forehead communication



Static Data Structures

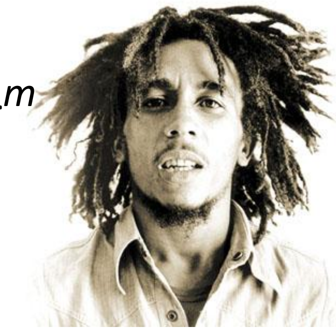
Asymmetric Communication Complexity



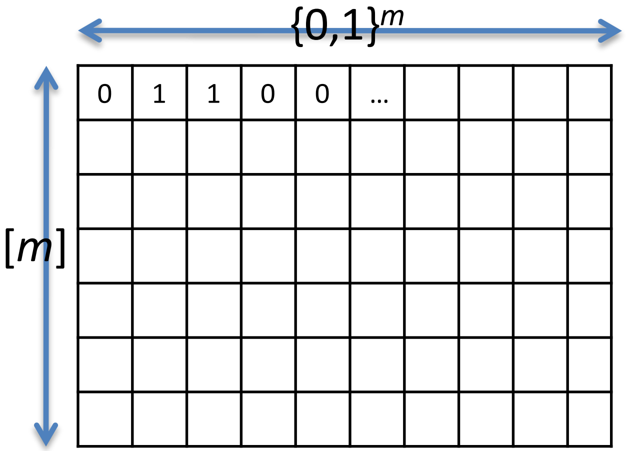
Indexing

$i \in [m]$ $\qquad$ $v \in \{0,1\}^m$

$v[i]$

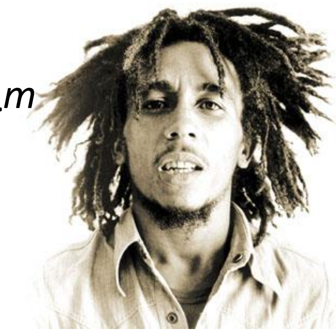


Theorem. Either Alice communicates $a = \Omega(\lg m)$ bits,
or Bob communicates $b \geq m^{1-\varepsilon}$ bits.



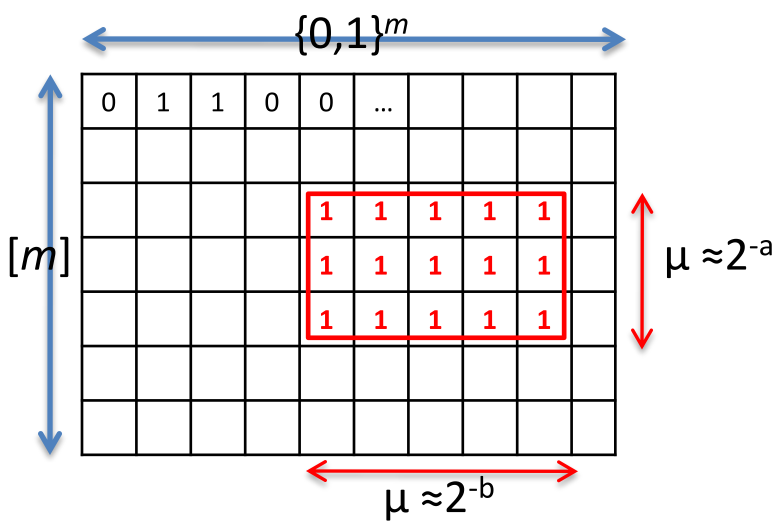
Indexing

$i \in [m]$ $v \in \{0,1\}^m$



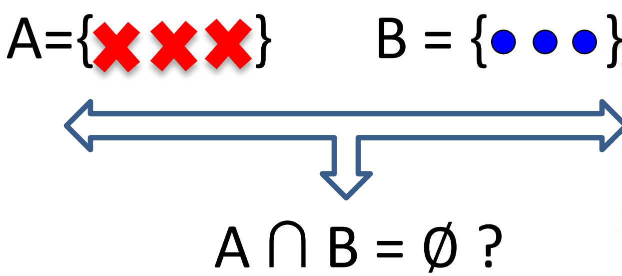
$v[i]$

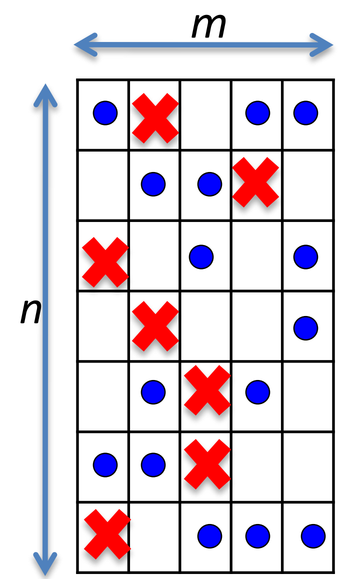
Theorem. Either Alice communicates $a = \Omega(\lg m)$ bits,
or Bob communicates $b \geq m^{1-\varepsilon}$ bits.



$m/2^a$ positions of v fixed to 1
$\Rightarrow b \geq m/2^a$

Lopsided Set Disjointness





Theorem. Either Alice communicates $\Omega(n \lg m)$ bits,
or Bob communicates $\geq n \cdot m^{1-\varepsilon}$ bits

Direct sum on Indexing:

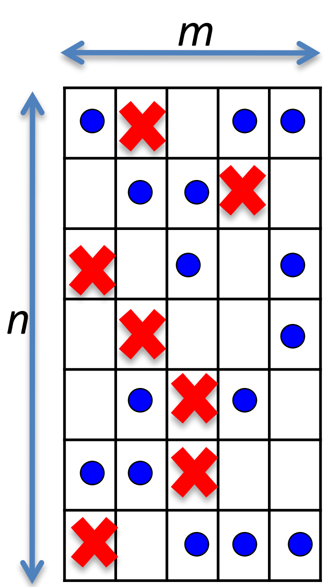
- deterministic: trivial
- randomized: [Pătraşcu'08]

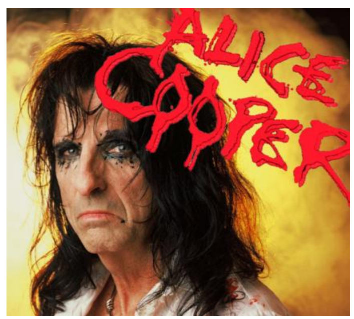
A Data Structure Lower Bound

Partial match: Preprocess a database D = strings in $\{0,1\}^d$

query($x \in \{0,1,*\}^d$) : does x match anything in D?

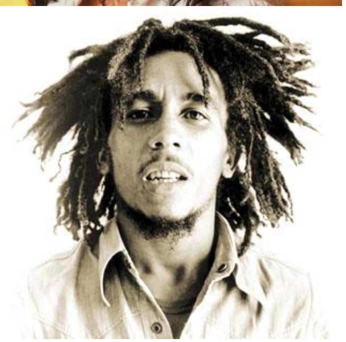
$C : [m] \rightarrow \{0,1\}^{3\lg m}$ constant-weight code





A={ n ✖'s} = { $(1,x_1), \ldots, (n,x_n)$ }

$\mapsto$ query($C(x_1) \circ \ldots \circ C(x_n)$)



B={ ½mn ●'s} $\mapsto$ D = { ½mn strings }

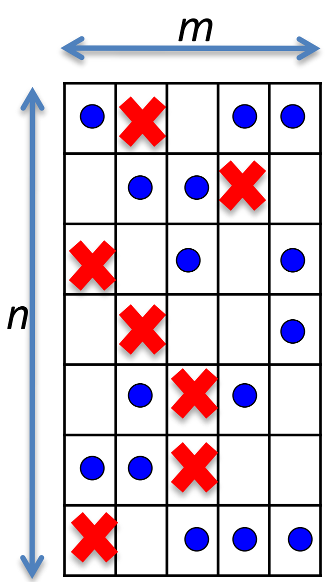
$(i,x_i) \mapsto 0 \circ \ldots \circ 0 \circ C(x_i) \circ 0 \circ \ldots$

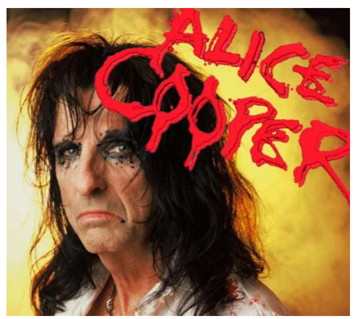
A Data Structure Lower Bound

LSD(n, m) $\mapsto$ Partial Match: $d = \Theta(n \lg m)$

Alice sends $\Omega(n \lg m)$ bits, | CPU sends $\Omega(d)$ bits,

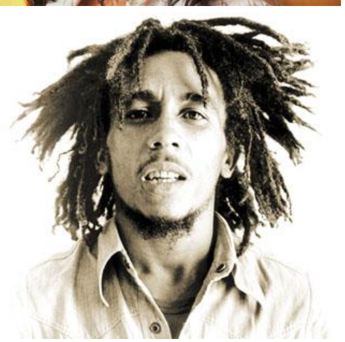
or Bob sends $\geq n \cdot m^{1-\varepsilon}$ bits | or Memory sends $\geq |D|^{1-\varepsilon}$





A={ n ✖'s} = { $(1,x_1), \ldots, (n,x_n)$ }

$\mapsto$ query($C(x_1) \circ \ldots \circ C(x_n)$)



B={ ½mn ●'s} $\mapsto$ D = { ½mn strings }

$(i,x_i) \mapsto 0 \circ \ldots \circ 0 \circ C(x_i) \circ 0 \circ \ldots$

A Data Structure Lower Bound

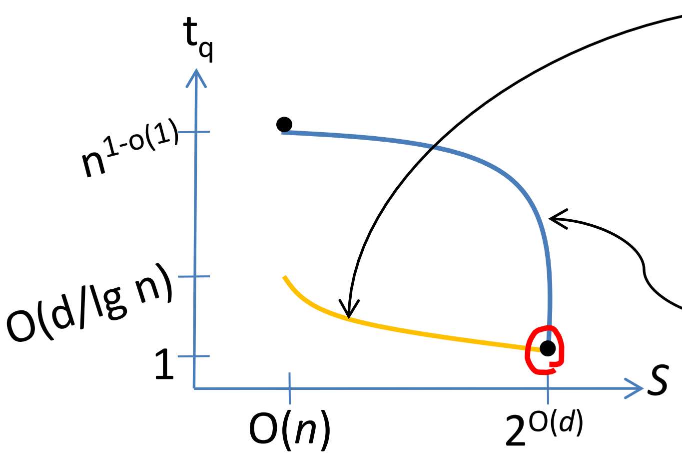
LSD(n, m) ↦ Partial Match: $d = \Theta(n \lg m)$

Alice sends $\Omega(n \lg m)$ bits, | CPU sends $\Omega(d)$ bits,

or Bob sends $\geq n \cdot m^{1-\varepsilon}$ bits | or Memory sends $\geq |D|^{1-\varepsilon}$

$\Rightarrow t \lg S = \Omega(d)$ or $t \cdot w \geq |D|^{1-\varepsilon}$

$\Rightarrow S = 2^{\Omega(d/t)}$



upper bound ≈ either:

- exponential space
- near-linear query time

Space Lower Bounds for t_q=O(1)

1995	[Miltersen, Nisan, Safra, Wigderson]		
1999	[Borodin, Ostrovsky, Rabani]	partial match	
2000	[Barkol, Rabani]	randomized exact NN	$S = 2^{\Omega(d)}$
2003	[Jayram,Khot,Kumar,Rabani]	partial match	
2004	[Liu]	deterministic O(1)-apx NN	$S = 2^{\Omega(d)}$
2006	[Andoni, Indyk, Pătraşcu]	randomized (1+ε)-apx NN	$S = n^{\Omega(1/\varepsilon^2)}$
	[Pătraşcu, Thorup]	direct sum for near-linear space	
2008	[Pătraşcu]	partial match	$S = 2^{\Omega(d)}$
	[Andoni, Croitoru, Pătraşcu]	ℓ_∞: apx=$\Omega(\log_\rho \log d)$ if	$S = n^\rho$
	[Panigrahy, Talwar, Wieder]	c-apx NN	$S \geq n^{1+\Omega(1/c)}$
2009	[Sommer, Verbin, Yu]	c-apx distance oracles	$S \geq n^{1+\Omega(1/c)}$
2010	[Panigrahy, Talwar, Wieder]		

Asymmetric Communication Complexity

Input: O(w) bits

lg S bits

w bits

lg S bits

w bits

Input: n bits

No difference between S=O(n) and S=$n^{O(1)}$

Separation $S = n \lg^{O(1)} n$ vs. $S = n^{O(1)}$

2006	[Pătrașcu, Thorup]	✖	predecessor search
	[Pătrașcu, Thorup]		exact near neighbor
2007	[Pătrașcu]	✖	2D range counting
2008	[Pătrașcu]	✖	2D stabbing, etc.
	[Panigrahy, Talwar, Wieder]		*c*-apx. near neighbor
2009	[Sommer, Verbin, Yu]		c-apx. distance oracles
2010	[Panigrahy, Talwar, Wieder]		*c*-apx. near neighbor
	[Greve,Jørgensen,Larsen,Truelsen]		range mode
2011	[Jørgensen, Larsen]	✖	range median

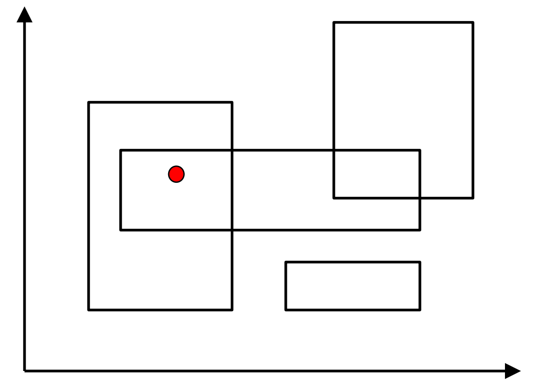
✖ = Tight bounds for space $n \lg^{O(1)} n$

2D Stabbing

Static 2D Stabbing:

Preprocess D = { n axis-aligned rectangles }

query(x,y): is $(x,y) \in R$, for some $R \in D$?



Goal: If $S = n \lg^{O(1)} n$, the query time must be $t = \Omega(\lg n / \lg\lg n)$

Remember: Dynamic 1D Stabbing

Maintain a set of segments $S = \{ [a_1,b_1], [a_2,b_2], \ldots \}$

insert / delete

query(x): is $x \in [a_i, b_i]$ for some $[a_i, b_i] \in S$?

We showed: If $t_u = \lg^{O(1)} n$, then $t_q = \Omega(\lg n / \lg\lg n)$

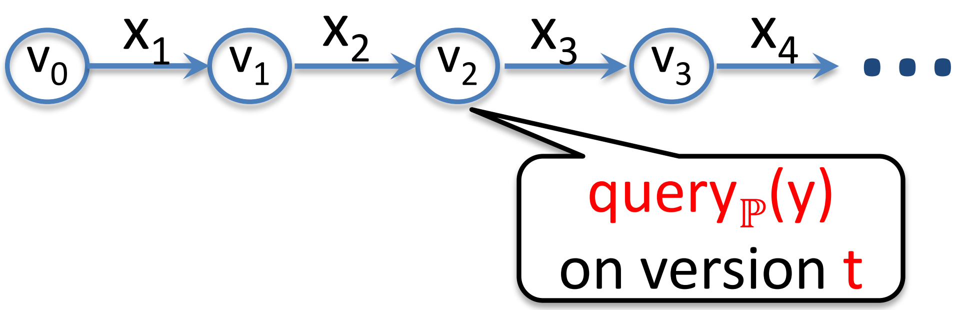
Persistence

Persistent : { dynamic problems } $\mapsto$ { static problems }

Given dynamic problem $\mathbb{P}$ with $\text{update}_{\mathbb{P}}(x)$, $\text{query}_{\mathbb{P}}(y)$

Persistent($\mathbb{P}$) = the static problem

Preprocess $(x_1, x_2, \ldots, x_T)$ to support:
query(y, t) = the answer of $\text{query}_{\mathbb{P}}(y)$ after $\text{update}_{\mathbb{P}}(x_1), \ldots, \text{update}_{\mathbb{P}}(x_t)$



Persistence

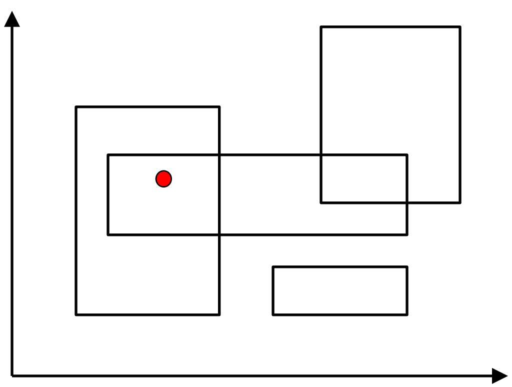
Persistent : { dynamic problems } ↦ { static problems }

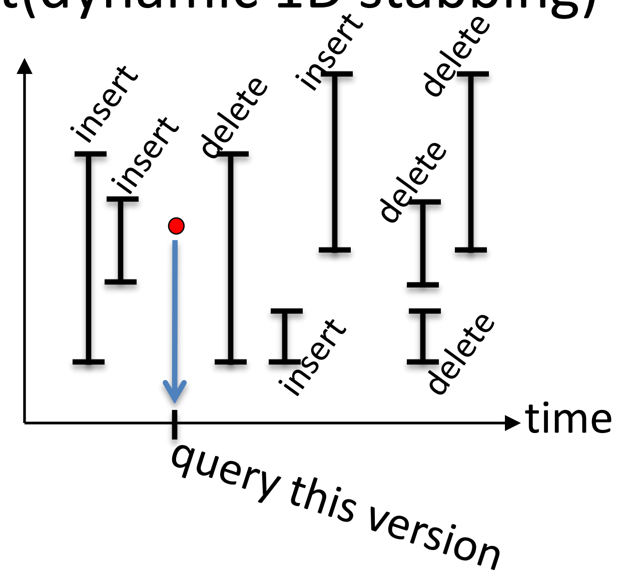
Given dynamic problem $\mathbb{P}$ with $\text{update}_{\mathbb{P}}(x)$, $\text{query}_{\mathbb{P}}(y)$

Persistent($\mathbb{P}$) = the static problem

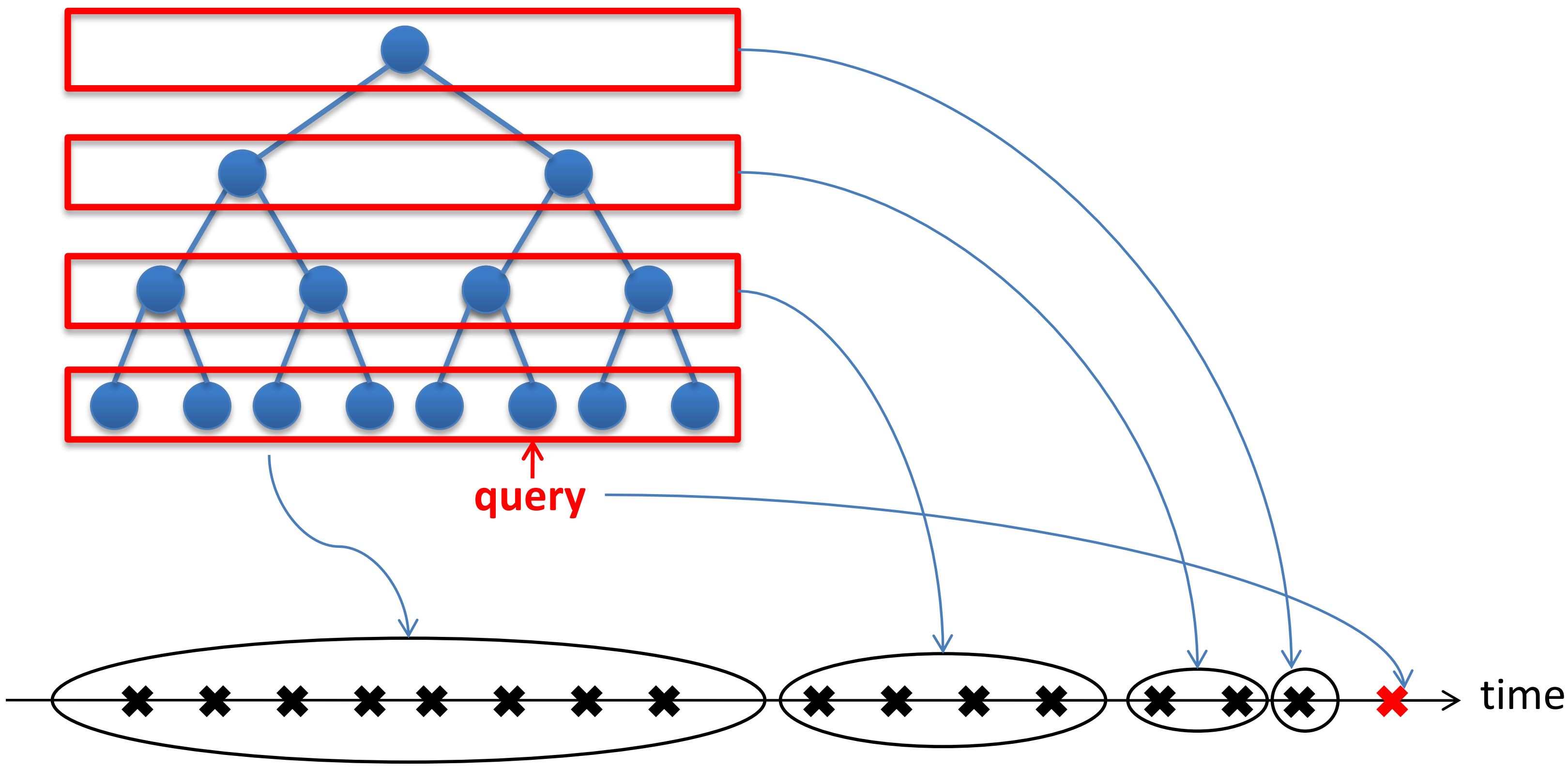
Preprocess $(x_1, x_2, \ldots, x_T)$ to support:
query(y, t) = the answer of $\text{query}_{\mathbb{P}}(y)$ after $\text{update}_{\mathbb{P}}(x_1), \ldots, \text{update}_{\mathbb{P}}(x_t)$

Static 2D stabbing ≤ Persistent(dynamic 1D stabbing)

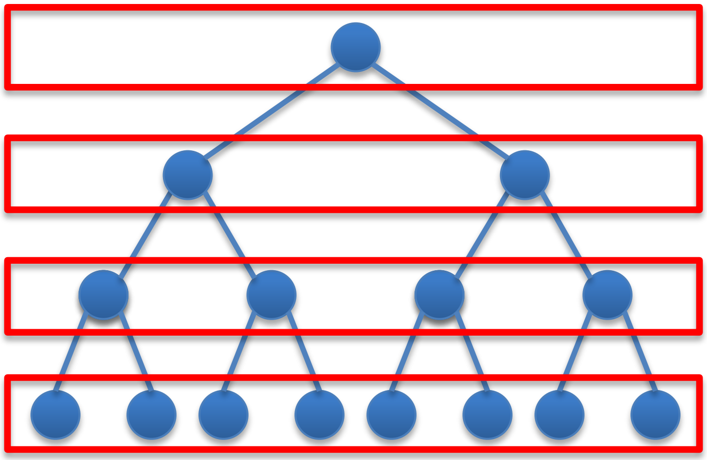


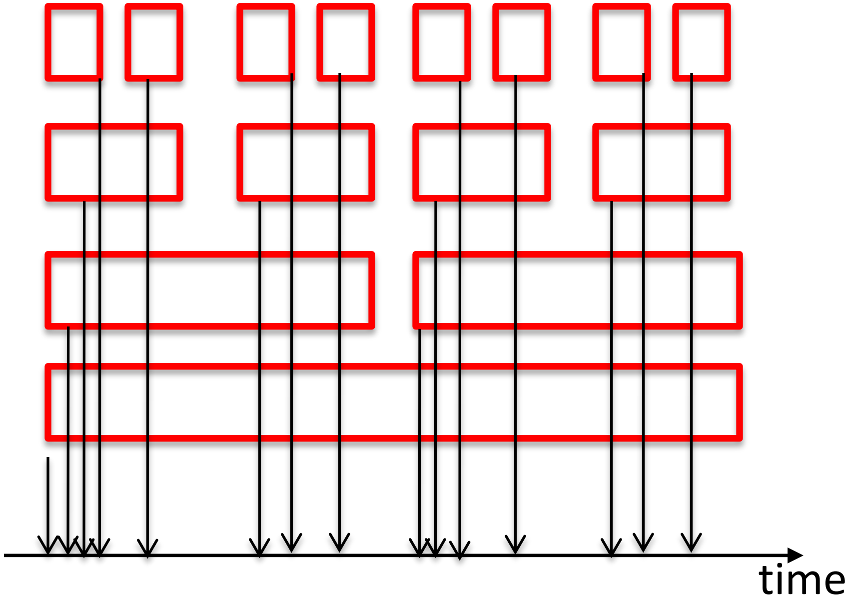


Recap: Marked Ancestor

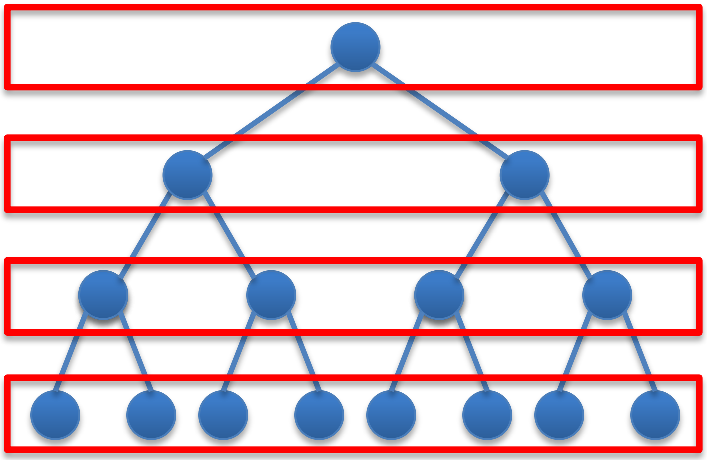


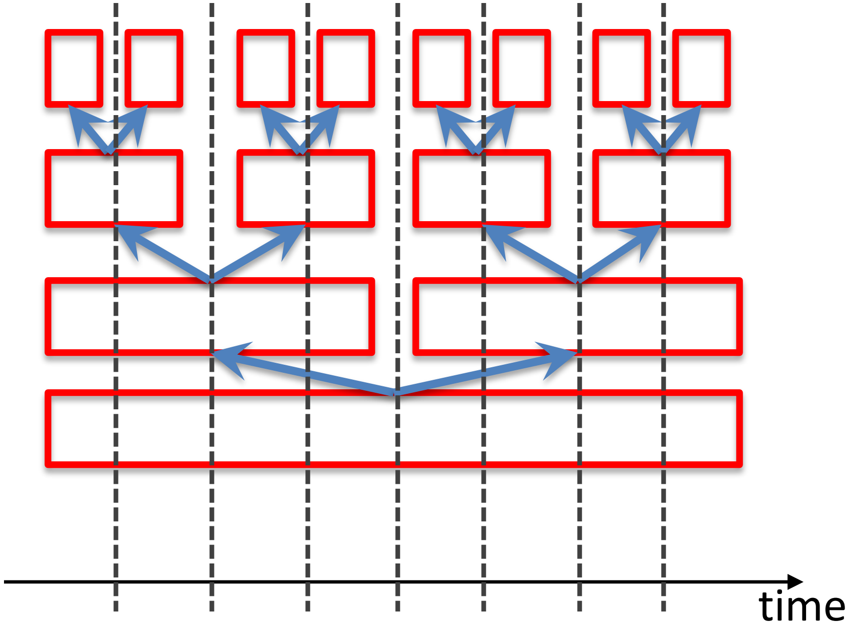
Persistent (Marked Ancestor)



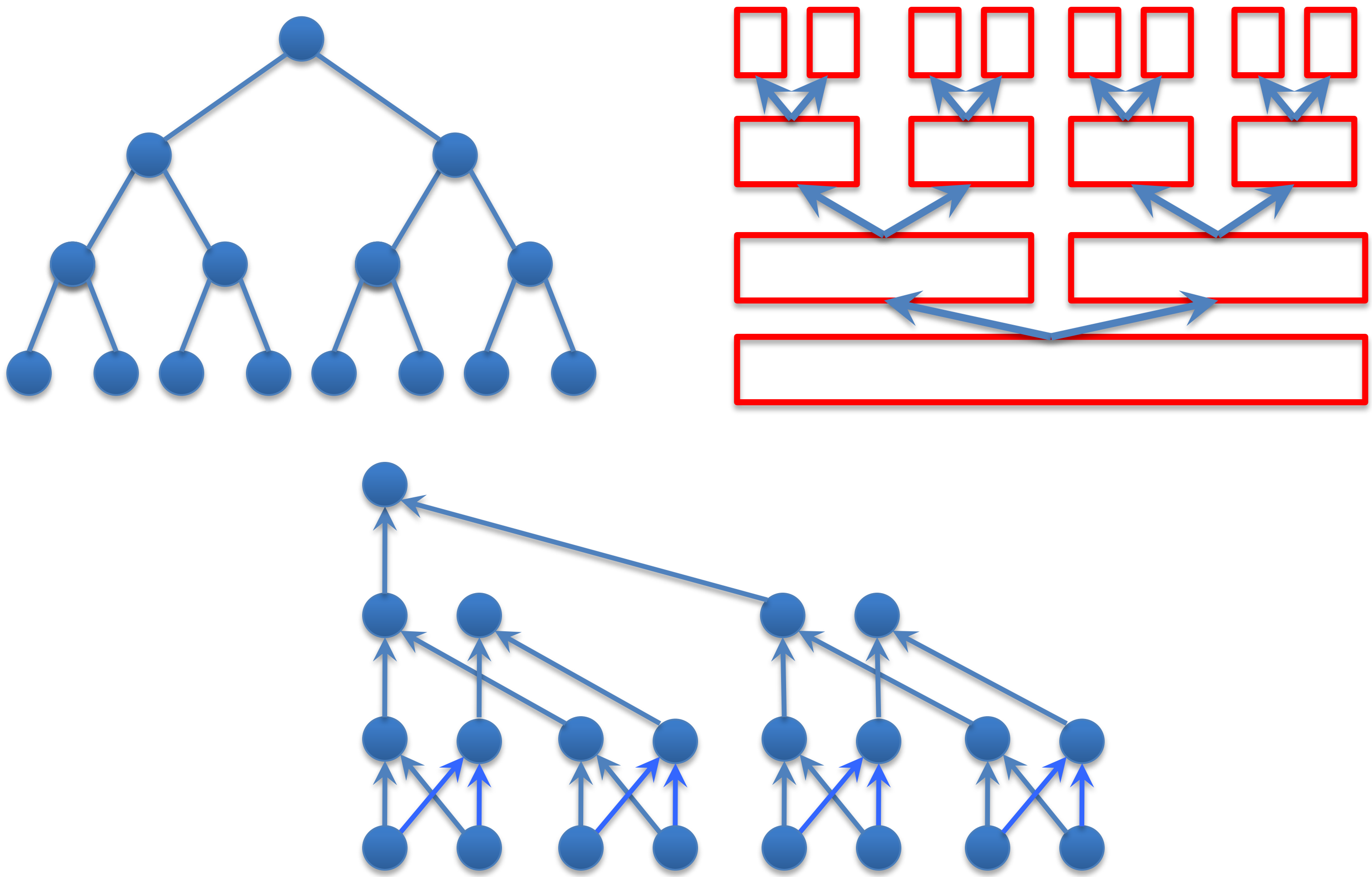


Persistent (Marked Ancestor)

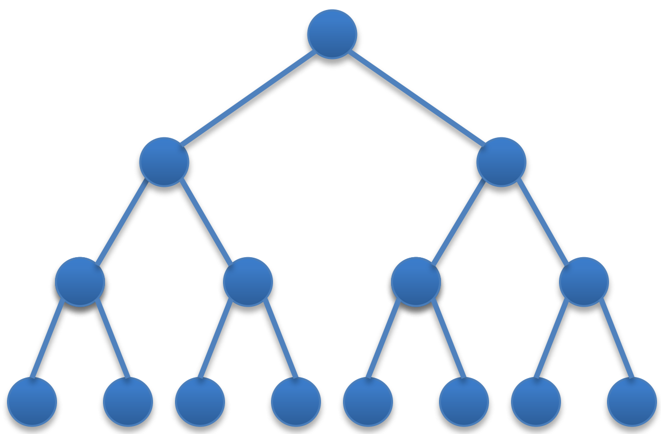


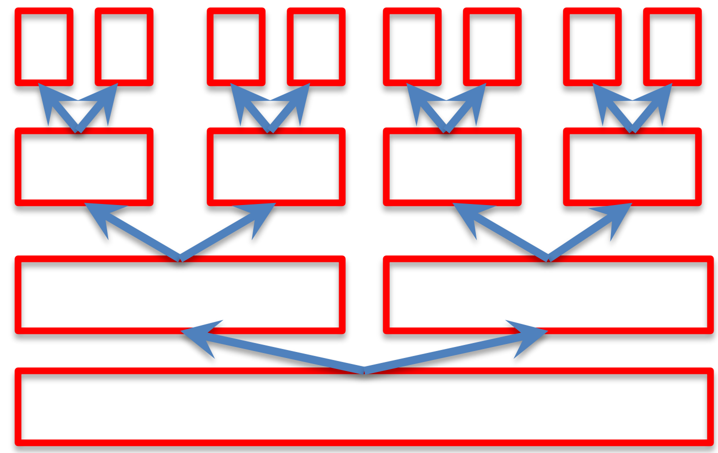


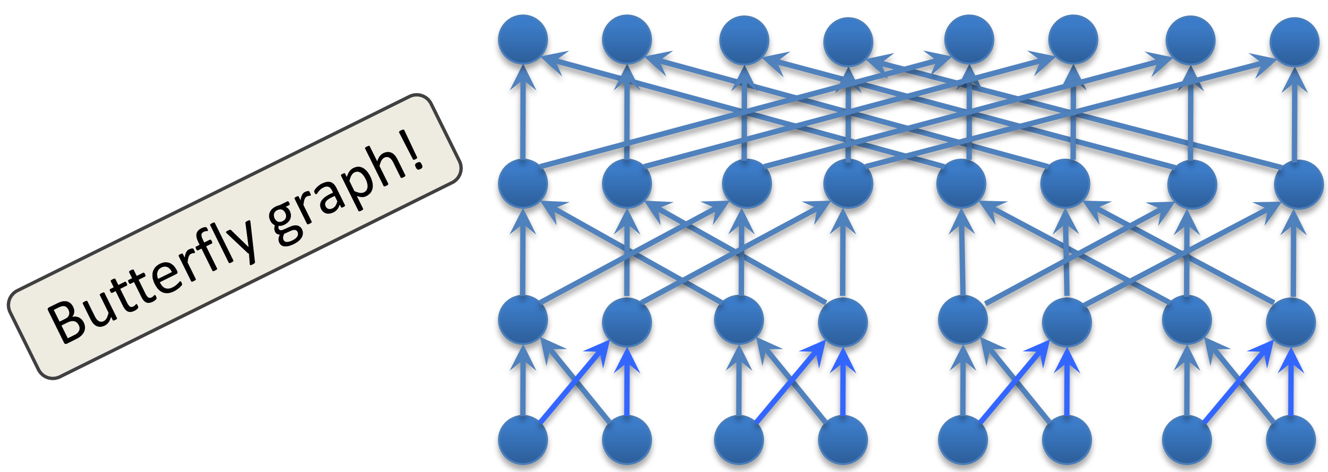
Persistent (Marked Ancestor)



Persistent (Marked Ancestor)



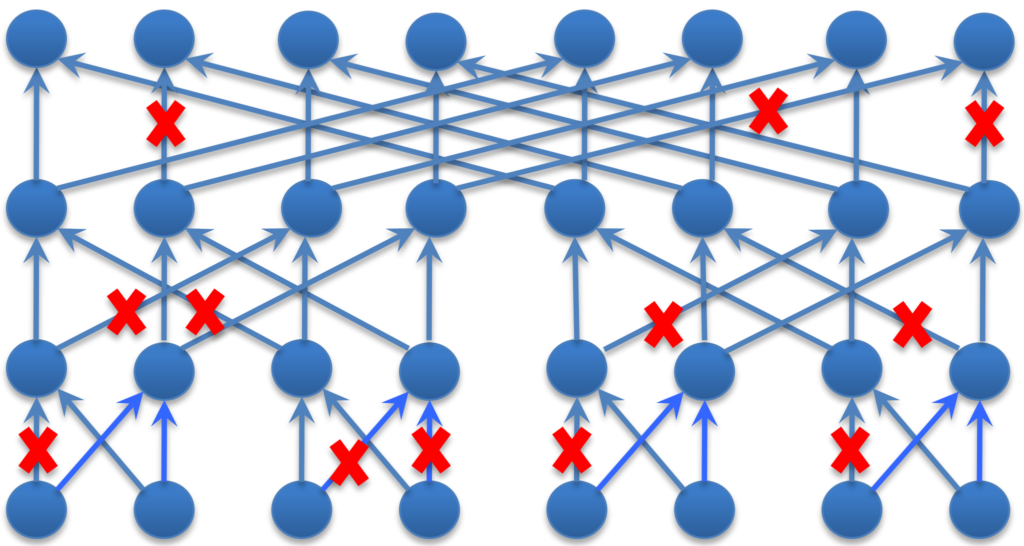




Butterfly Reachability

Let G = butterfly of degree *B* with *n* wires

Preprocess a subgraph of G
query(u, v) = is there a path from source *u* to sink *v*?



\# vertices: $N = n \cdot \log_B n$
\# edges: $N \cdot B$

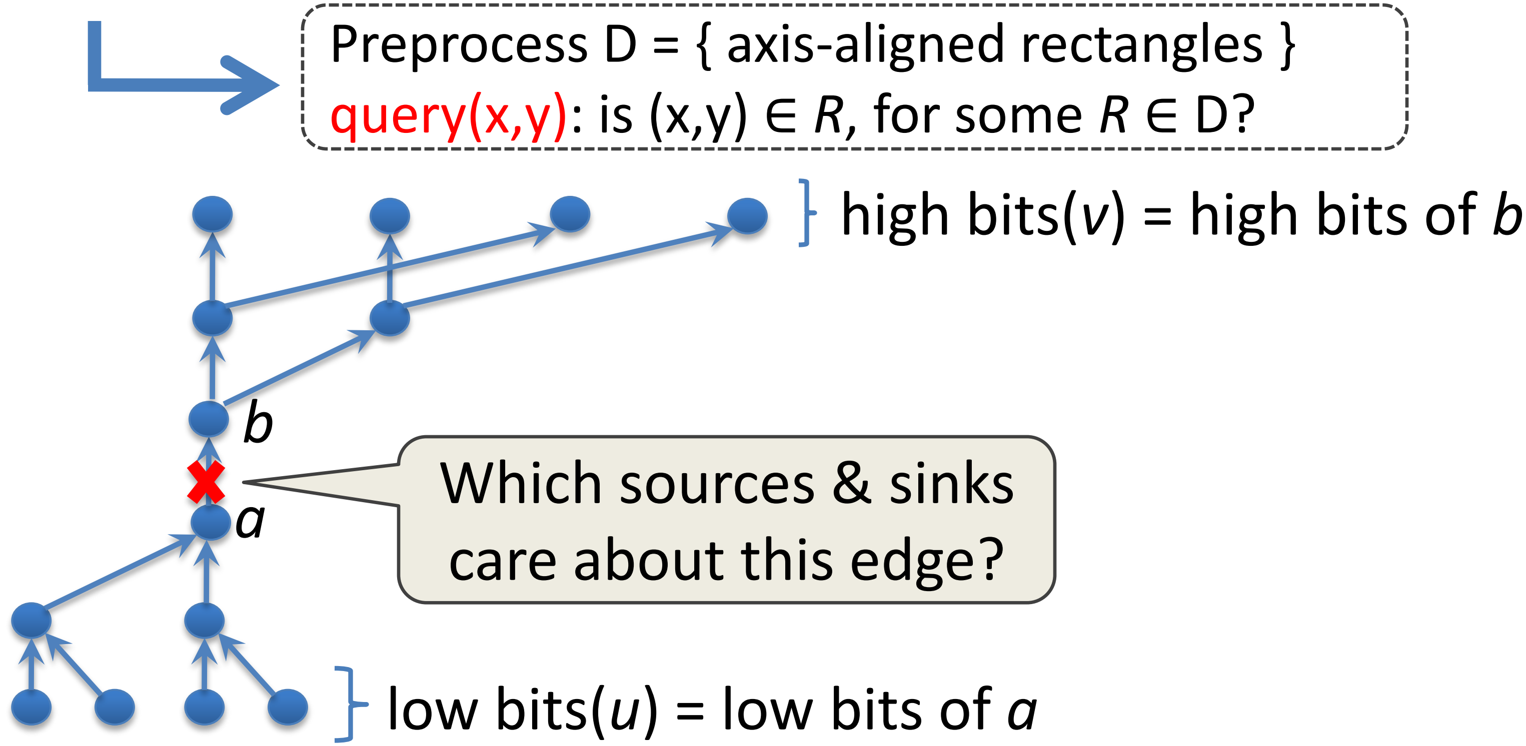
Database = {✘'s}
= vector of $N \cdot B$ bits

Butterfly Reachability ↦ 2D Stabbing

Let G = butterfly of degree *B* with *n* wires

Preprocess a subgraph of G
query(u,v) = is there a path from source *u* to sink *v*?

Preprocess D = { axis-aligned rectangles }
query(x,y): is (x,y) ∈ *R*, for some *R* ∈ D?

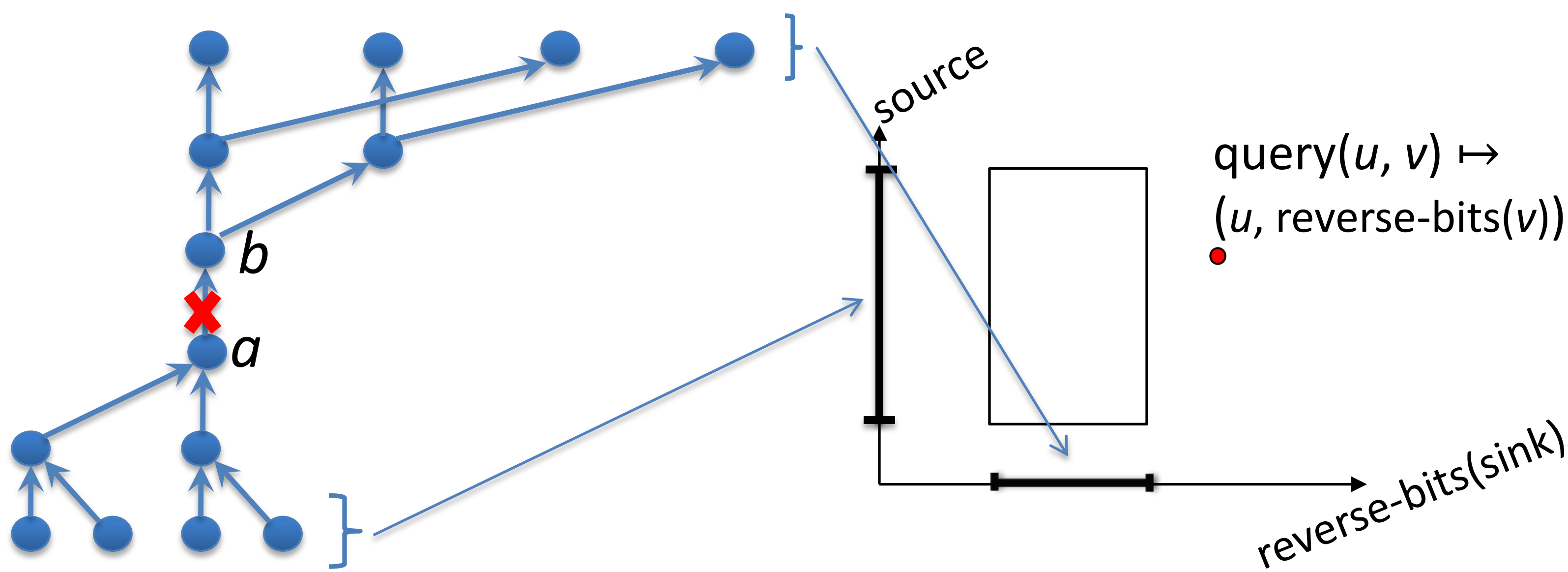


Butterfly Reachability ↦ 2D Stabbing

Let G = butterfly of degree *B* with *n* wires

Preprocess a subgraph of G
query(u,v) = is there a path from source *u* to sink *v*?

Preprocess D = { axis-aligned rectangles }
query(x,y): is (x,y) ∈ *R*, for some *R* ∈ D?



Hardness of Butterfly Reachability

query(u,v)

Input: 2 lg n bits

lg S bits

O(lg n) bits

Input: $N \cdot B$ bits

Either Alice sends Ω(lg n) bits
or Bob sends $B^{1-\varepsilon}$ bits

$$\lg \binom{S}{n} \approx n \lg \frac{S}{n} = O(n \lg\lg n)$$

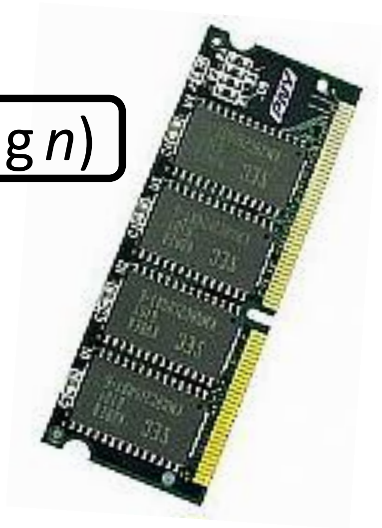
query(u_1,v_1)

...

query(u_n,v_n)

$O(n \lg\lg n)$

$O(n \lg n)$ bits



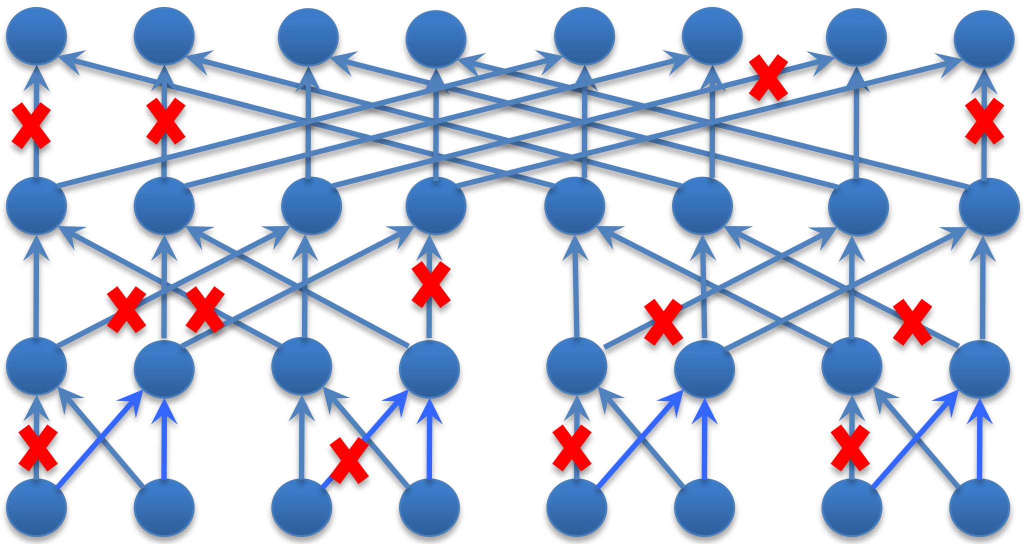
Input: $O(n \lg n)$

Input: $N \cdot B$ bits

Either Alice sends $\boldsymbol{n} \times \Omega(\lg n)$ bits $\Rightarrow t = \Omega(\lg n / \lg\lg n)$

or Bob sends $\boldsymbol{n} \times B^{1-\varepsilon}$ bits

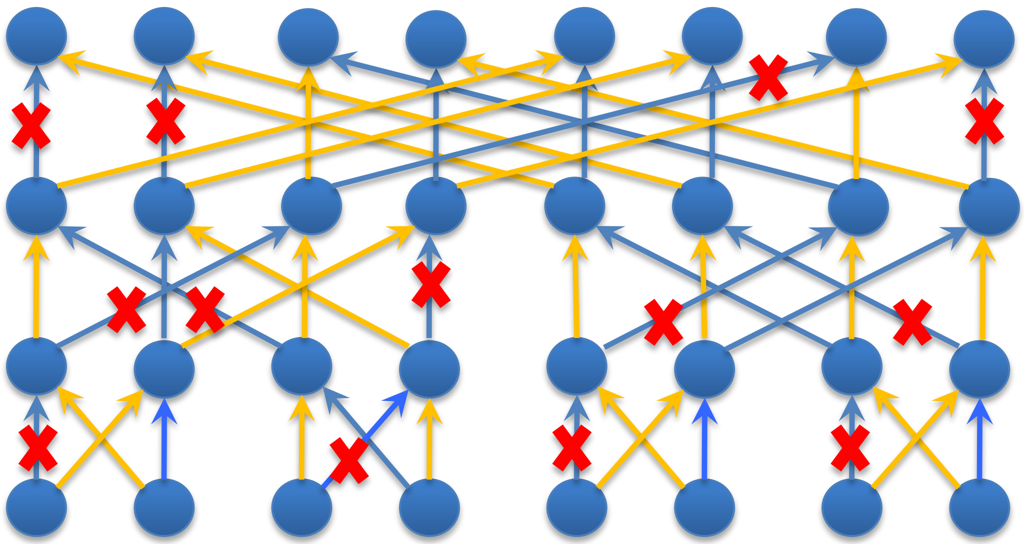
Hardness of Butterfly Reachability



B = { ½NB ✗'s }
↦ database

Either Alice sends $n \times \Omega(\lg n)$ bits
or Bob sends $n \times B^{1-\varepsilon}$ bits

Hardness of Butterfly Reachability



B = { ½*NB* ✗'s }

$\mapsto$ database

A={ $\log_B n$ matchings }

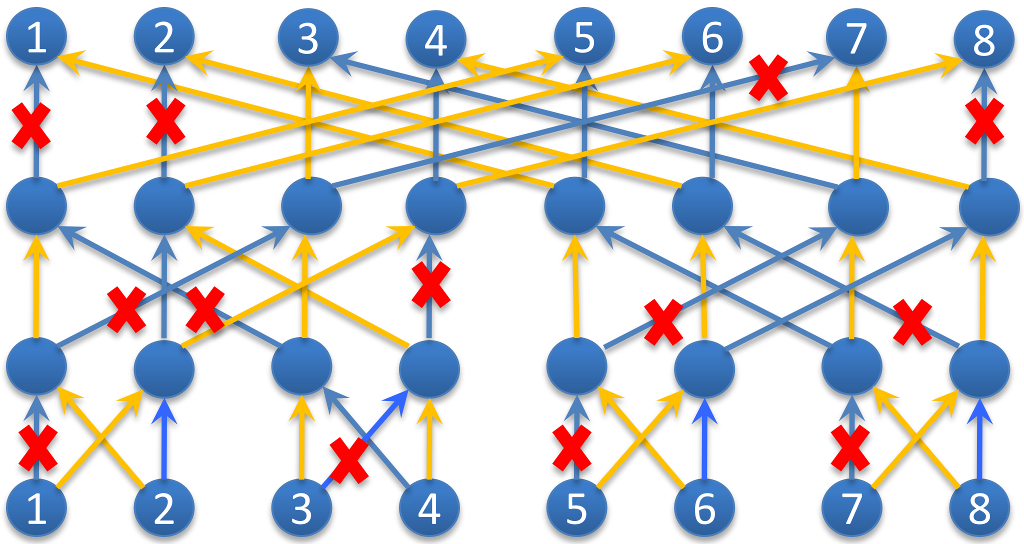
= { *N* ↗'s}

Either Alice sends $n \times \Omega(\lg n)$ bits

or Bob sends $n \times B^{1-\varepsilon}$ bits

$A \cap B = \emptyset \iff$

query(1,8)=true ∧ query(2,5)=true ∧ …



B = { ½*NB* ✖'s }

↦ database

A={ $\log_B n$ matchings }

= { *N* ↗'s}

↦ *n* queries

LSD lower bound:

Either Alice sends $n \times \Omega(\lg n)$ bits
or Bob sends $n \times B^{1-\varepsilon}$ bits

Bibliography: Predecessor Search

Preprocess a set S = { *n* integers }

predecessor(x) : max { y ∈ S | y ≤ x }

1988	[Ajtai]	1st static lower bound
1992	[Xiao]	
1994	[Miltersen]	
1995	[Miltersen, Nisan, Safra, Wigderson]	round elimination lemma
1999	[Beame, Fich]	optimal bound for space $n^{O(1)}$
	[Chakrabarti, Chazelle, Gum, Lvov]**	
2001	[Sen]	randomized
2004	[Chakrabarti, Regev]**	
2006	[Pătraşcu, Thorup]	optimal bound for space $n \lg^{O(1)} n$
		1st separation between polynomial and linear space
2007	[Pătraşcu, Thorup]	randomized

**) Work on approx. nearest neighbor

Bibliography: Succinct Data Structures

On input of n bits, use $n + o(n)$ bits of space.

[Gál, Miltersen '03] polynomial evaluation
$\Rightarrow$ redundancy $\times$ query time $\geq \Omega(n)$

[Golynski '09] store a permutation and query $\pi(\cdot)$, $\pi^{-1}(\cdot)$
If space is $(1+\varepsilon)\cdot n \lg n$ bits $\Rightarrow$ query time is $\Omega(1/\sqrt{\varepsilon})$

Tight!

[Pătrașcu, Viola '10] prefix sums in bit vector
For query time t $\Rightarrow$ redundancy $\geq n / \lg^{O(t)} n$

Tight!

NB: Also many lower bounds under the indexing assumption.

The End