Lower Bounds for Data Structures

Mihai Pătrașcu

2nd Barriers Workshop, Aug. 29 ’10
Model of Computation

Word = $w$-bit integer
Memory = array of $S$ words
Unit-time operations on words:
• random access to memory
• $+, -, *, /, \%, <, >, ==, <<, >>, ^, \&, |, \sim$

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

Internal state: $O(w)$ bits
Hardware: $\text{NC}^1$
Cell-Probe Model

Cell = \( w \) bits
Memory = array of \( S \) cells
CPU:
• state: \( O(w) \) bits
• apply \emph{any} function on state \([\text{non-uniform!}]\)
• 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

[Ajtabi ’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

“Epochs”
**Proof Overview**

**Claim.** $(\forall) \ k$, $\Pr[\text{query reads something from epoch } k] \geq 0.1 \Rightarrow E[t_q] = \Omega(\log_B n)$

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. \( (\forall \, k \), \Pr[\text{query reads something from epoch } k] \geq 0.1 \)

Proof: Assume not.

Encode \( N = B^k \) random bits with \(<N \) bits on average
Formal Proof

Public coins: \(N\) queries

Public coins

The \(N\) bits to be encoded

Public coins
Formal Proof

Claim. $(\forall) \ k$, $\Pr[\text{query reads something from epoch } k] \geq 0.1$

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 $\Rightarrow$ 90% of queries can be run ignoring epoch $k$
Formal Proof

Decoder

Encoder

Public coins

The $N$ bits

Public coins

Which queries read from epoch $k$

$o(N)$ bits

time
Formal Proof

**Claim.** \((\forall) \, k, \text{ Pr}[\text{query reads something from epoch } k] \geq 0.1\)

**Proof:** Assume not.

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

Public coins: past updates, future updates, \(N\) queries

Assumption \(\Rightarrow 90\%\) of queries can be run ignoring epoch \(k\)

**Encoding:**

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

\(\square\)
Applications

Partial sums:

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

- update(i, Δ): A[i] = Δ

Incremental connectivity (union-find):

Maintain a graph under:

- link(u,v): add edge
- query(u,v): are u and v connected?

“0”    “1”
Fancy Application: Marked Ancestor

[Alstrup, Husfeldt, Rauhe FOCS’98]

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

Only mark a node with probability $\approx 1/\lg n$

Query needs cell that might have been written in another version!
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 $k$ \} $\cup$ \{ $N$ random elements \}
Which is which? $2N$ bits to tell...

If true queries read from epoch $k$ & false queries don’t
$\Rightarrow$ 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]

\[ t_q = \Omega \left( \frac{\lg n}{\lg t_u} \right) \]

[Pătraşcu, Thorup ’11]

\[ t_u = o \left( \frac{\lg n}{\lg \lg n} \right) \Rightarrow t_q \geq n^{1-o(1)} \]

...for incremental connectivity

“Don’t rush into a union. Take time to find your roots!”
Hard Instance
Hard Instance (cont.)

Let $M = \text{“width of edges”}$ \hspace{1cm} (M=n^{1-\varepsilon})

Operations:

- macro-update (setting one $\pi$)
  \hspace{1cm} $\mapsto$ \hspace{1cm} $M$ edge inserts

- macro-query:
  
  - color root and leaf with $C$ colors \hspace{0.5cm} (C=n^{\varepsilon})
    \hspace{1cm} $\mapsto$ \hspace{1cm} $M$ edge inserts
  
  - test consistency of coloring
    \hspace{1cm} $\mapsto$ \hspace{1cm} $C^2$ connectivity queries
The Lower Bound

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

Operations:
- \text{macro-update: } M \text{ insertions}
- \text{macro-query: } M \text{ insertions} + C^2 \text{ queries}

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

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

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

\text{time} \quad O(B^{i-1} M t_u) \text{ cells}

\text{B}^i \text{ queries}
Communication

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

Bob: for each \(\pi_i\), a coloring of inputs & outputs with \(C\) colors

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

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

Bob: for each $\pi_i$, 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

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

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: \( \Omega(B^iM \lg M) \)

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

So \( |W \cap R| = \Omega(B^iM) \). \( \square \)
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]

\[ t_q = \Omega(\frac{\log n}{\log t_u}) \]

[Alstrup, Husfeldt, Rauhe FOCS’98]

\[ t_q = \Omega(\frac{\log n}{\log \frac{t_u}{\log n}}) \]

[Pătraşcu, Thorup ’11]

\[ t_u = o(\frac{\log n}{\log \log n}) \Rightarrow t_q \geq n^{1-o(1)} \]

[Pătraşcu, Demaine STOC’04]

\[ t_q = \Omega(\frac{\log n}{\log (t_u/\log n)}) \]

Also:

\[ t_u = o(\log n) \Rightarrow t_q \geq n^{1-o(1)} \]
The hard instance:

\[ \pi = \text{random permutation} \]

for \( t = 1 \) to \( n \):

query: \( \text{sum}(\pi(t)) \)

\( \Delta_t = \text{rand()} \)

update(\( \pi(t), \Delta_t \))

Maintain an array \( A[n] \) under:

update(\( i, \Delta \)): \( A[i] = \Delta \)

sum(\( i \)): return \( A[0] + \ldots + A[i] \)
How can Mac help PC run $t = 9, \ldots, 12$?

Communication = $2w \cdot \#\text{memory cells}$

- read during $t = 9, \ldots, 12$
- written during $t = 5, \ldots, 8$
Lower bound on entropy?
The general principle

Lower bound
= # down arrows

\[ E[\# \text{down arrows}] = \Omega(k) \]
Recap

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

Communication between periods of $k$ items
$= \Omega(k)$

# memory locations
* read during mauve period
* written during beige period
$= \Omega(k)$
Putting it all together

Every memory read counted once

@ lowest_common_ancestor(
  write time, read time
)
Dynamic Lower Bounds

[Fredman, Saks STOC’89]

\[ t_q = \Omega(\lg n / \lg t_u) \]

[Alstrup, Husfeldt, Rauhe FOCS’98]

\[ t_q = \Omega(\lg n / \lg (t_u/\lg n)) \]

[Pătraşcu, Demaine STOC’04]

\[ t_q = \Omega(\lg n / \lg (t_u/\lg n)) \]

[Pătraşcu, Thorup ’11]

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

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

[Fredman, Saks STOC’89]
\[ t_q = \Omega\left(\frac{\lg n}{\lg t_u}\right) \]

[Alstrup, Husfeldt, Rauhe FOCS’98]
\[ t_q = \Omega\left(\frac{\lg n}{\lg (t_u/\lg n)}\right) \]

[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:** \( S = \{n \text{ numbers}\}, \ \ (\exists)x,y,z \in S \text{ with } x+y+z=0? \)
Conjecture: requires \( \Omega^*(n^2) \) on RAM
The Multiphase Problem

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

\[ \Rightarrow \text{reachability in dynamic graphs requires } \Omega(n^\epsilon) \]
3-Party, Number-on-Forehead

\[
S_1, ..., S_k \subseteq [u] \quad \text{time } O(k \cdot u \cdot \tau)
\]

\[
T \subseteq [u] \quad \text{time } O(u \cdot \tau)
\]

\[
S_i \cap T? \quad \text{time } O(X)
\]
Dynamic Lower Bounds

\[ t \sim n^{1-o(1)} \]

Now

Future?

[Fredman, Saks’89]
Classic Results

[Yao FOCS’78]
• (kind of) defines the model
• membership with low space

[Ajtaı́ ’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

Input: $O(w)$ bits

Asymmetric communication complexity

Input: $n$ bits

$w$ bits

$\lg S$ bits
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

[Pătrașcu FOCS’08]

• lopsided set disjointness  (via information complexity)
Round Elimination
Round Elimination

**Setup:**

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

**Output:**

If Alice sends a message of \(m \ll k\) bits \(\Rightarrow\) **fix** \(i\) and **eliminate round**

**Now:**

- Alice has input \(x_i\)
- Bob has an input \(y\)
- Output: \(f(x_i, y)\)
Predecessor Search

\[ \text{pred}(q, S) = \max \{ x \in S \mid x \leq q \} \]

[van Emde Boas FOCS’75]

if \( \lfloor q/\sqrt{u} \rfloor \in \text{hash table} \), return \( \text{pred}(q \mod \sqrt{u}, \text{bottom structure}) \)
else return \( \text{pred}(\lfloor q/\sqrt{u} \rfloor, \text{top structure}) \)

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: $q = (q_1, q_2, ..., q_k)$
Bob: $i \in [k], (q_1, ..., q_{i-1}), S$
Goal: $\text{pred}(q_i, S)$

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

Space = $O(n)$ $\Rightarrow$ set $k = O(\lg n)$ $\Rightarrow$ lower bound: $\Omega(\log_{\lg n} n)$
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] $A=\Omega(d)$, $B=\Omega(n^{1-\varepsilon})$

=> $t_q \geq \min \left\{ \frac{d}{\lg S}, \frac{n^{1-\varepsilon}}{w} \right\}$
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 $q \in \{0,1,\ast\}^d$ Bob has $S=n$ points in $\{0,1\}^d$
Goal: does the query match anything?

[Pătraşcu FOCS’08] $A=\Omega(d)$, $B=\Omega(n^{1-\varepsilon})$
$\Rightarrow t_q \geq \min \left\{ \frac{d}{\lg S}, \frac{n^{1-\varepsilon}}{w} \right\}$
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 $S = 2^\Omega(d/t_q)$

Also: optimal lower bound for decision trees

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

$\Rightarrow t_q \geq \min \{d/\lg S, n^{1-\epsilon}/w\}$
Results

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

[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 ($\ell_1$, $\ell_2$):

deterministic $\gamma$-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 $\ell_\infty$:
[Andoni, Croitoru, Pătraşcu FOCS’08] “[Indyk FOCS’98] is optimal!”
The Barrier

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

[Pătraşcu, 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 1\textsuperscript{st} Cell Probe

If (∃)k \ |S_k| \leq \sqrt{n}:

- place query & data set in segment k
- 1\textsuperscript{st} memory access = f(lo(q)) ∈ S_k
Restricting 1\textsuperscript{st} Cell Probe

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

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

If there exists a solution to \( \text{Pred}(n, u) \) with:
- space complexity: \( O(n) \)
- query complexity: \( t \) memory reads

\[ \Rightarrow \]

There exists a solution to \( \text{Pred}(n, \forall u) \) with:
- space complexity: \( O(n) \)
- \( O(\forall n \cdot \lg n) \) “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\textsuperscript{nd} round, there are $O(\sqrt{n} \cdot \lg^2 n)$ published bits.

**Direct sum:** \[ \text{Pred}(n, u) = k \times \text{Pred}(n/k, u/k) \]

\[ k \gg \sqrt{n} \cdot \lg^2 n \implies \text{With } O(\sqrt{n} \cdot \lg^2 n) \text{ public bits, most sub-problems are still hard.} \]
**Main Lemma:** Fix algorithm to read from a set of \((nk)^{\frac{1}{2}}\) cells

"Proof":

- problem j is nice if \((\exists)\alpha: |S_{\alpha}^j| \leq (n/k)^{\frac{1}{2}}\)
  \[ \Rightarrow \text{ fix hi-part in problem } j \]

- problem j is not nice if \((\forall)\alpha: |S_{\alpha}^j| > (n/k)^{\frac{1}{2}}\)
  \[ \Rightarrow \text{ choose } T \text{ to hit all such } S_{\alpha}^j \]

\[ |T| \approx n / (n/k)^{\frac{1}{2}} = (nk)^{\frac{1}{2}} \]
New Induction Plan

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

“Proof”:

• problem j is nice if \((\exists)\alpha: |S_\alpha^j| \leq (n/k)^{\frac{1}{2}}\)
  \[\Rightarrow\text{ fix hi-part in problem j to } \alpha\]

• problem j is not nice if \((\forall)\alpha: |S_\alpha^j| > (n/k)^{\frac{1}{2}}\)
  \[\Rightarrow\text{ choose } T \text{ to hit all such } S_\alpha^j\]

\[\Omega(lglg n)\]
Main Lemma: Fix algorithm to read from a set of \((nk)^{\frac{1}{2}}\) cells

- problem j is nice if \(\exists \alpha: |S_\alpha^j| \leq (n/k)^{\frac{1}{2}}\)
  \[\Rightarrow \text{fix hi-part in problem j to } \alpha\]
- problem j is not nice if \(\forall \alpha: |S_\alpha^j| > (n/k)^{\frac{1}{2}}\)
  \[\Rightarrow \text{choose T to hit all such } S_\alpha^j\]

But: Published bits = \(f\) (database)

\[1^{\text{st}} \text{ cell read by query} = f(\text{published bits})\]
Main Lemma: “Proof” ⇒ Proof

New claim. We can publish \((nk)^{\frac{1}{2}}\) cells such that:
Pr[random query reads a published cell] ≥ 1/100

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

Proof:
• Publish random sample \(T = \{(nk)^{\frac{1}{2}}\text{ cells}\}\)
• For each problem \(j\) where \(\text{lo}(q)\) is relevant (fixed \(hi=\alpha\))
  publish \(S_\alpha^j\) only if \(|S_\alpha^j| \leq (n/k)^{\frac{1}{2}}\)
An Encoding Argument

Assume \( \Pr[\text{random query reads a published cell}] < 1/100 \)

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:
   
   \[ A[j]=0 \Rightarrow \text{lo}(q_j) \text{ is relevant in problem j} \]
   
   \[ A[j]=1 \Rightarrow \text{hi}(q_j) \text{ is relevant in problem j} \]

3. Encode published bits \( \rightarrow o(k) \) bits

4. Decoder classifies queries:
   
   \( S_{hi(q_j)} \)
   
   when \( |S_{hi(q_j)}| \leq (n/k)^{1/2} \) query is \( \circ \) iff \( A[j]=0 \)
   
   when \( |S_{hi(q_j)}| > (n/k)^{1/2} \) query is \( \smile \) iff \( A[j]=1 \)

5. By assumption, \( E[\text{number of } \frown \text{ queries}] \geq 99\% \) \( k \)
   
   So decoder can learn 99\% of \( A[1..k] \)
Beyond Communication?

CPU → memory communication:

- one query: \( \lg S \)
- \( k \) queries: \( \lg \left( \frac{S}{k} \right) = \Theta \left( k \lg \frac{S}{k} \right) \)
Direct Sum

CPU → memory communication:
• one query: \( \lg S \)
• \( k \) queries: \( \lg \binom{S}{k} = \Theta(k \lg \frac{S}{k}) \)
Direct Sum

CPU → memory communication:
• one query: $\lg S$
• $k$ queries: $\lg \binom{S}{k} = \Theta(k \lg \frac{S}{k})$

[Pătrașcu, Thorup FOCS’06]
Any richness lower bound
“Alice must send $A$ or Bob must send $B$”
⇒
$k \times \text{Alice}$ must send $k \cdot A$
or $k \times \text{Bob}$ must send $k \cdot B$
Direct Sum

CPU \rightarrow \text{memory communication:}

- one query: \( \log S \)
- \( k \) queries: \( \log \binom{S}{k} = \Theta(k \log \frac{S}{k}) \)

[Pastra\c{c}u, Thorup FOCS’06]

Any richness lower bound

“Alice must send \( A \) or Bob must send \( B \)”

\( \Rightarrow \)

- \( k \times \text{Alice} \) must send \( k \cdot A \)
- or \( k \times \text{Bob} \) must send \( k \cdot B \)

\text{Old: } t_q = \Omega(A/\log S)

\text{New: } t_q = \Omega(A/\log(S/k))

\text{Set } k = n/\lg^{O(1)} n

\Rightarrow \Omega(\log n/\lg \lg n) \text{ time for space } n \cdot \lg^{O(1)} n
$\Omega(\lg n / \lg \lg n)$ 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 \( \in [u] \)
\[ \Rightarrow \text{optimal space} \quad H = \lg(u \text{ choose } n) \text{ bits} \]

Space: \( H + \text{redundancy} \)
What is the redundancy / time trade-off?

\[ [\text{Pagh'99}] \quad \text{membership} \mapsto \text{prefix sums} \]
\[ [\text{Pătrașcu FOCS'08}] \quad \text{prefix sums: time } O(t), \]
\[ \quad \text{redundancy} \approx \frac{n}{\lg^t n} \]
Succinct Lower Bounds

[Gál, Miltersen ’03] polynomial evaluation
\[ \Rightarrow \text{redundancy} \times \text{query time} \geq \Omega(n) \]

[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 \( t \Rightarrow \text{redundancy} \geq n / \lg^t n \)
The End