An Adaptive Packed-Memory Array

Michael A. Bender
Stony Brook and Tokutek\textsuperscript{R}, Inc
Physicists Beget Computer Scientists (Generational Divide)

- I’m a theoretical computer scientist
- Linguistic divide: Computer scientists speak discretely and physicists speak in quanta (and they speak continuously).

- But I want my talk in the spirit of QMCD ’09.
My father has written papers ...

...but I won’t talk about these.

with me:

and my brother:

WHAT IS THE OPTIMAL SHAPE OF A BLOB?

Oblong geyser
(How I View)
My Father’s Style

• Ask simple-to-state questions about elementary physics.
• Extract all the richness and complexity out of these problems
  - e.g., anharmonic oscillator, relativistic brachistochrone, QMCD
(How I View) 
My Father’s Style

• Ask simple-to-state questions about elementary physics.
• Extract all the richness and complexity out of these problems
  - e.g., anharmonic oscillator, relativistic brachistochrone, QMCD
I've done my best to imitate this style

- Style that I strive for.
- Here’s my attempt to do the same sort of thing.
Fall 1990. I'm an undergraduate.

Insertion-Sort Lecture.

insertion: $O(N)$
insertion sort: $O(N^2)$
Fall 1990. I'm an undergraduate.

Insertion-Sort Lecture.

What a boneheaded way to implement insertion!
Fall 1990. I'm an undergraduate.

Insertion-Sort Lecture.

Leave empty spaces or gaps to accommodate future insertions.

\[
\begin{array}{cccccc}
2 & 5 & 7 & 10 & 12 & 15 \\
3 & 13 & 8 & & & \\
\end{array}
\]
Anybody who has spent time in a library knows that insertions are cheaper than linear time.

Insertion sort is $O(N \log N)$.  

LibrarySort [Bender, Farach-Colton, Mosteiro 04] :  
$O(N \log N)$ sorting for average-case insertions.
How is LibrarySort like a library?

• Leave gaps on shelves so shelving is fast
• Putting books randomly on shelves with gaps: At most $O(\log N)$ books need to be moved with high probability* to make room for a new book.

* Probability $>1-1/poly(N)$, $N$=#books.
But what if Library buys many copies of....

- Bender and Orszag
But what if Library buys many copies of...?

- Bender and Orszag
- Bender and Orszag, 2\textsuperscript{nd} Ed.
- Bender, and Orszag, 3\textsuperscript{rd} Ed.
- Bender and Orszag, 500\textsuperscript{th} Ed.
But what if Library buys many copies of....

- Bender and Orszag
- Bender and Orszag, 2nd Ed.
- Bender, and Orszag, 3rd Ed.
- Bender and Orszag, 500th Ed.
- Books by other Benders
But what if Library buys many copies of....

- Bender and Orszag
- Bender and Orszag, 2nd Ed.
- Bender, and Orszag, 3rd Ed.
- Bender and Orszag, 500th Ed.
- Books by other Benders
- Bender & Orszag, Volume II (?)
But what if Library buys many copies of....

- Bender and Orszag
- Bender and Orszag, 2nd Ed.
- Bender, and Orszag, 3rd Ed.
- Bender and Orszag, 500th Ed.
- Books by other Benders
- Bender & Orszag, Volume II (?)

Now there’s a bolus of books on in one place. Can one still maintain small shelving costs?
Insertions into Array with Gaps

- Dynamically maintain $N$ elements sorted in memory/on disk in a $\Theta(N)$-sized array

- Idea: rearrange elements & gaps to accommodate future insertions

- **Objective**: Minimize amortized (technical form of ave) # of elts moved per update.
Actually Two Objectives

• Minimize \# elements moved per insert.

• Minimize \# block transfers per insert.

• Disk Access Model (DAM) of Computer
  - Two levels of memory
  - Two parameters:
    block size $B$, memory size $M$.
Packed-Memory Array (PMA)
[Bender, Demaine, Farach-Colton 00, 05]

- **(Worst-case) Inserts/Deletes:**
  - $O(\log^2 N)$ amortized element moves
  - $O(1+(\log^2 N)/B)$ amortized memory transfers

- **Scans of $k$ elements after given element:**
  - $O(1+k/B)$ memory transfers

```
28
14 18 23 34 50 59 66
```

Rebalance carefully chosen neighborhood.
Problem: a worst case for PMA is sequential inserts, but this is a common case for databases. Industrial data structures (Oracle, TokuDB) are optimized for sequential inserts.
An Adaptive PMA
[Bender, Hu 2007]

• **Same guarantees as PMA:**
  \[ O(\log^2 N) \] element moves per insert/delete
  \[ O(1+ (\log^2 N)/B) \] memory transfers

• **Optimized for common insertion patterns:**
  insert-at-head (sequential inserts)
  random inserts
  bulk inserts (repeatedly insert \( O(N^b) \) elements in random position, \( 0 \leq b \leq 1 \))

**Guarantees:**
\[ O(\log N) \] element moves
\[ O(1+ (\log N)/B) \] mem transfers
Sequential Inserts

Inserts “hammer” on one part of the array.

Amortized moves over $\log N$ running time
Random Inserts

Insertions are after random elements.

Amortized moves over $\log N$  

running time
Sample Applications

Maintain data physically in order on disk
• Traditional and “cache-oblivious” B-trees
  - Core of all databases and file systems
• My startup Tokutek
• Even an online dating website
Try to insert in \( \log N \)-sized interval.

If interval already full, rebalance smallest enclosing interval within thresholds.

Imaginary Intervals in PMA

\[2^{\log N + O(1)} = O(N)\]

Density Thresholds

Upper

- \( T_3 = 0.7 \)
- \( T_4 = 0.8 \)
- \( T_5 = 0.9 \)
- \( T_6 = 1.0 \)

Lower

- \( P_3 = 0.3 \)
- \( P_4 = 0.25 \)
- \( P_5 = 0.2 \)
- \( P_6 = 0.15 \)

\[T_i - T_{i+1} = \Theta(\rho_{i+1} - \rho_i) = \Theta(\frac{1}{\log N})\]
Imaginary Intervals in PMA

\[ 2^{\log N + O(1)} = O(N) \]

Density Thresholds

<table>
<thead>
<tr>
<th>Upper</th>
<th>Lower</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_3 = 0.7 )</td>
<td>( p_3 = 0.3 )</td>
</tr>
<tr>
<td>( \tau_4 = 0.8 )</td>
<td>( p_2 = 0.25 )</td>
</tr>
<tr>
<td>( \tau_5 = 0.9 )</td>
<td>( p_1 = 0.2 )</td>
</tr>
<tr>
<td>( \tau_6 = 1.0 )</td>
<td>( p_0 = 0.15 )</td>
</tr>
</tbody>
</table>

\[ \tau_i - \tau_{i+1} = \Theta(p_{i+1} - p_i) = \Theta(\frac{1}{\log N}). \]

To insert:

- Try to insert in leaf interval.
- If interval full, **rebalance** smallest enclosing interval within thresholds.
Analysis Idea: $O(\log^2 N)$ amortized element moves per insert

- $O(\log N)$ amort. moves to insert into interval
  - Amortized analysis: Charge rebalance of interval $u$ to inserts into child interval $v$
- Insert in $O(\log N)$ intervals for insert in PMA
Analysis of $O(\log^2 N)$ Moves/Insert

Before rebalance: $\text{density}(v) > T_e$ or $\text{density}(v) < P_e$

After rebalance: $P_{e+1} < \text{density}(v) < T_{e+1}$

density thresholds $T_{e+1}, P_{e+1}$

density thresholds $T_e, P_e$
Analysis of $O(\log^2 N)$ Moves/Insert

Before rebalance: $\text{density}(v) > \tau_e$ or $\text{density}(v) < \rho_e$

After rebalance: $\rho_{e+1} < \text{density}(v) < \tau_{e+1}$

Amortized cost of rebalancing $u$ = \frac{\text{cost of rebalance}}{\text{inserts before rebalance}} = \frac{\text{Size}(u)}{\text{Size}(v)} \max \left\{ \frac{1}{\tau_e - \tau_{e+1}}, \frac{1}{\rho_{e+1} - \rho_e} \right\}

= O(\log N)$
• Charge rebalance cost of $u$ to inserts into $v$
  - After rebalance $v$ within threshold of parent $u$
• Amortized cost of $O(\log N)$ to insert into $u$
Analysis Summary

- Charge rebalance cost of $u$ to inserts into $v$
  - After rebalance $v$ within threshold of parent $u$
- Amortized cost of $O(\log N)$ to insert into $u$
- But each insert is into $O(\log N)$ intervals

- Total: $O(\log^2 N)$ amortized moves
Idea of Adaptive PMA

• Adaptively remember elements that have many recent inserts nearby.
• Rebalance *unevenly*. Add extra space near these volatile elements.

\[28\]

\[
\begin{array}{cccccccc}
14 & 18 & 23 & 34 & 50 & 59 & 66 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
14 & 18 & 23 & 28 & 34 & 50 & 59 & 66 \\
\end{array}
\]

• This strategy overcomes a \(\Omega(\log^2 N)\) lower bound [Dietz, Sieferas, Zhang 94] for “smooth” rebalances
Why $O(\log^2 N)$ Can Be Improved in the Common Case.

To guarantee $O(\log^2 N)$, we only need....

**Rebalance Property:** After a rebalance involving $v$, $v$ is within parent $u$’s density threshold.

**Summary:** As long as $v$ is within $u$’s threshold, it can be sparser or denser than $t$’s density thresholds.
Why $O(\log^2 N)$ Can Be Improved in the Common Case.

To guarantee $O(\log^2 N)$, we only need....

**Rebalance Property**: After a rebalance involving $v$, $v$ is within parent $u$'s density threshold.

**Summary**: Remarkable that this is good enough. Only large rebalances have slop and most are small.
How to Remember Hot Elements Adaptively

- Maintain an $O(\log N)$-sized predictor, which keeps track of PMA regions with recent inserts:
  - $O(\log N)$ counters, each up to $O(\log N)$.
  - Remembers up to $O(\log N)$ hotspot elements.
  - Tolerates “random noise” in inputs.

- (Generalization of how to find majority element in an array with a single counter.)

- Rebalance to even out weight of counters, while maintaining rebalance property.
Summary

• Insertion sort with gaps
  - LibrarySort [Bender, Farach, Colton, Mosteiro ’04] (+ Wikipedia entry)

• Worst-possible inserts
  - PMA [Bender, Demaine, Farach-Colton ’00, ’05]
  - Cache-oblavious B-trees and other data structures

• Adapt to common distributions
  - APMA [Bender, Demaine, Farach-Colton ’00, ’05]

• Implementation of cache-oblavious data structures
  - Tokutek
• Is it practical to keep data physically in order in memory/on disk?

Speaking for B-trees... I believe yes.
Overview of Talk