Predecessor Queries in Dynamic Integer Sets

Gerth Stølting Brodal

Max-Planck-Institut für Informatik
Saarbrücken, Germany
The Problem

Maintain a set $S$ of size $n$ under the operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert $(e)$</td>
<td>Insert $e$ into $S$</td>
</tr>
<tr>
<td>Delete $(e)$</td>
<td>Delete $e$ from $S$</td>
</tr>
<tr>
<td>Pred $(e)$</td>
<td>Return the largest element $\leq e$ in $S$</td>
</tr>
<tr>
<td>FindMin / Max</td>
<td>Return the minimum / maximum in $S$</td>
</tr>
</tbody>
</table>
The Problem

Maintain a set $S$ of size $n$ under the operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Priority Queue</th>
<th>Dictionary</th>
<th>Trade off</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert ($e$)</td>
<td>$O(\log n)$</td>
<td>$O(\log n)$</td>
<td>$O(t)$</td>
</tr>
<tr>
<td>Delete ($e$)</td>
<td>$O(\log n)$</td>
<td>$O(\log n)$</td>
<td>$\downarrow \frac{n}{2^{\Omega(t)}}$</td>
</tr>
<tr>
<td>Pred ($e$)</td>
<td>$-$</td>
<td>$O(\log n)$</td>
<td></td>
</tr>
<tr>
<td>FindMin / Max</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td></td>
</tr>
</tbody>
</table>

Brodal/Chauhuri/Radhakrishnan 96
The Problem on a Practical RAM

- Unit cost RAM
- $+$, Shifting, bit-wise boolean operations, direct and indirect addressing, jumps, conditional statements
- Wordsize is $w$
- Elements are integers in the range $0, 2^w - 1$
Packed search trees containing \( \frac{W}{2^{f(n)}} \) bit integers (i.e., \( 2^{f(n)} \) integers can be packed into a word) supporting:

- Insert, Delete: \( \mathcal{O}(f(n)) \)
- Pred: \( \mathcal{O}\left(\frac{\log n}{f(n)}\right) \)
- FindMin, FindMax: \( \mathcal{O}(1) \)
The Problem

Maintain a set $S$ of size $n$ under the operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>van Emde Boas 77</th>
<th>Thorup 96</th>
<th>Andersson 95</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert ($e$)</td>
<td>$O(\log w)$</td>
<td>$O(\log \log n)$</td>
<td>$O(\sqrt{\log n})$</td>
</tr>
<tr>
<td>Delete ($e$)</td>
<td>$O(\log w)$</td>
<td>$O(\log \log n)^*$</td>
<td>$O(\sqrt{\log n})$</td>
</tr>
<tr>
<td>Pred ($e$)</td>
<td>$O(\log w)$</td>
<td>$-$</td>
<td>$O(\sqrt{\log n})$</td>
</tr>
<tr>
<td>FindMin/Max</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>

*Delete requires amortized $O(\log \log n)$ time.*

DeleteMin is supported in worst case $O(\log \log n)$ time.
The Problem

Maintain a set $S$ of size $n$ under the operations

Results (for Practical RAM)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Main*</th>
<th>Corollary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert($e$)</td>
<td>$O(f(n))$</td>
<td>$O(\log \log n)$</td>
</tr>
<tr>
<td>Delete($e$)</td>
<td>$O(\frac{\log n}{f(n)})$</td>
<td>$O(\frac{\log n}{\log \log n})$</td>
</tr>
<tr>
<td>Pred($e$)</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>FindMin/Max</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>

* $f(n)$ is a smooth function satisfying

$$\log \log n \leq f(n) \leq \sqrt{\log n}$$
Packed Search Trees - Preliminaries

- Assumption: Lists containing at most $k$ integers can be stored in $O(1)$ words, $(k = 2^{f(n)})$

- Lemma (Albers/Hagerup 92)
  Two sorted lists can be merged in $O(\log k)$ time

- Lemma
  Given two sorted lists $A$ and $B$, the list $A \setminus B$ can be computed in $O(\log k)$ time
The tree is an \((1, \frac{k}{\log^4 n})\)-tree of height \(\frac{\log n}{\log k}\).

Leaf buffers have always size \(\Theta(k)\).

Internal buffers have size \(\leq \frac{k}{\log^3 n}\).
Packed Search Trees

- The tree is an \((1, \frac{k}{\log^3 n})\)-tree of height \(\frac{\log n}{\log k}\).
- Leaf buffers have size \(\Theta(4)\).
- Internal buffers have size \(\leq \frac{k}{\log^3 n}\).

Removing \(O(\log n)\) elements from the root buffer takes \(O(\log \frac{n}{\log k} \cdot \log 4) = O(\log n)\) time.

\[\text{Insert}(17)\] and \(\text{Insert}(18)\)
Packed Search Trees

- The tree is an \((1, \frac{k}{\log^6 n})\)-tree of height \(\frac{\log n}{\log k}\).
- Leaf buffers have always size \(\Theta(k)\).
- Internal buffers have size \(\leq \frac{k}{\log^3 n}\).
- \(L\) and \(R\) are nonempty and have size \(O(k)\).
Open Problems

- Can Insert/Delete be supported in $O(\log \log n)$ time, while supporting Pred in $O(\sqrt{\log n})$ time (or better)?
- Find a tradeoff between the update time and Pred.
- FindMin/FindMax.