Finger Search Trees with Constant Insertion Time

Gerth Stølting Brodal

Max–Planck–Institut für Informatik
Saarbrücken
Germany

January 1998
The problem

Maintain a sorted list of elements under the operations

- **INSERT**(\(f, x\)), insert element \(x\) to the right of finger \(f\)

  ![Insertion Example](image)

- **SEARCH**(\(f, x\)), search for element \(x\) in the list starting at finger \(f\)

  ![Search Example](image)

\(\delta\) = the distance (rank difference) between \(f\) and \(x\)
A simple finger search tree

Brown, Tarjan ’80

Represent the sorted list by a level linked (2,3)–tree

**INSERT**: worst-case $O(\log n)$ time, amortized $O(1)$ time

**SEARCH**: worst-case $O(\log \delta)$ time

Question: Can the insertion time be made worst-case $O(1)$?
Finger Search Trees with Constant Insertion Time

## Known results

<table>
<thead>
<tr>
<th></th>
<th><strong>INSERT</strong></th>
<th><strong>SEARCH</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Search trees</td>
<td>$\log n$</td>
<td>$\log n$</td>
</tr>
<tr>
<td>Levcopoulus, Overmars ’88</td>
<td>1</td>
<td>$\log n$</td>
</tr>
<tr>
<td>Brown, Tarjan ’80</td>
<td>1</td>
<td>$\log \delta$</td>
</tr>
<tr>
<td>Dietz, Raman ’94</td>
<td>1</td>
<td>$\log \delta$</td>
</tr>
<tr>
<td>Harel, Lucker ’79</td>
<td>$\log^* n$</td>
<td>$\log \delta$</td>
</tr>
<tr>
<td>Guibas <em>et al.</em> ’77</td>
<td>1</td>
<td>$\log \delta$</td>
</tr>
<tr>
<td>Brodal ’98</td>
<td>1</td>
<td>$\log \delta$</td>
</tr>
</tbody>
</table>
A new algorithm for splitting nodes in search trees

To each leaf $l$ is associated a binary counter $c_l$

All internal nodes of height $d$ have degree $\geq \Delta_d$ (except for the root)

---

**INSERT($\ell, \ell'$)**

1. $c_\ell := c_\ell + 1$
2. $c_{\ell'} := c_\ell$
3. Find $d : c_\ell \mod 2^d = 2^{d-1}$
4. Split the $d$th ancestor of $\ell$ (if degree $\geq 2\Delta_d$)
Main lemma

Lemma If $\Delta_d \geq 2^{2d} - 1$, then all nodes of height $d$ have degree $\leq 2^{2d} \Delta_d$

Proof Let $v$ be an internal node of height $d$, and $\ell$ any leaf

Define potentials

$$\Phi^d_\ell = 2^{2d} - 1 - ((c_\ell + 2^d - 1) \mod 2^d)$$

$$\Phi^d_v = \sum_{\ell \in T^d_v} \Phi^d_\ell$$

Invariant

$$\Phi^d_v \leq 2^{2d} \prod_{i=1}^d \Delta_i$$

The lemma follows from $|T^d_v| \geq \prod_{i=1}^d \Delta_i$
**Splitting nodes of large degree**

Split nodes incrementally in advance by introducing intermediate nodes of degree $\Delta_d / 2 \cdot 2 \Delta_d - 1$

\[ \text{SPLIT}(v) \equiv \text{move one intermediate node.} \]

**Finger searches**

1. Level link the tree
2. Represent the children of each internal/intermediate node by the search trees of Levcopoulus, Overmars ’88
   \[ \Rightarrow \text{finger searches take } O(\log \delta) \text{ time} \]
Finding level $d$ ancestors

1. Represent each counter $c_\ell$ by a stack $S_\ell$ of intervals of 1’s
2. With each interval $(i, j)$ in $S_\ell$ store a pointer to the $j + 1$st ancestor of $\ell$

⇒ in worst-case $O(1)$ time $S_\ell$ can be updated and the $d$th ancestor found
**Copying $S_\ell$ stacks in $O(1)$ time**

Let the $S_\ell$ stacks be functionally implemented

⇒ $S_{\ell'} := S_\ell$ takes worst-case $O(1)$ time

**Pointers to wrong subtrees**

Splitting a node $v$ can make $S_\ell$ stacks contain pointers to wrong subtrees!

*But* the algorithm still works — nodes of height $d$ have degree $\leq 2^{3 \cdot 2^d} \Delta_d$
Conclusion and open problems

Theorem

A pointer based implementation of finger search trees exists with worst-case time bounds

\[
\begin{align*}
\text{Insert} & : O(1) \\
\text{Search} & : O(\log \delta) \\
\text{Delete} & : O(\log^* n)
\end{align*}
\]

The data structure requires linear space

---

Open problems

- **DELETE** in worst-case \( O(1) \) time too
- Make other splitting based data structures worst-case
  
  *e.g. the full persistence technique of Driscoll et al. ’89.*