The Randomized Complexity of Maintaining the Minimum

The FindMin problem

Ordered universe

Data structure

Input: Sequence of operations

- Ins, insert an element
- Del, delete an element (given a pointer)
- FindMin, return the current minimum

- Ins\(\rightarrow\) \text{FindMin}
- Del\(\rightarrow\) \text{FindMin}

Example

- \text{Ins}(a1)
- \text{Ins}(a5)
- \text{Ins}(a8)
- \text{Del}(a5)

- on-line, comparison model
- worst-case cost

- Brodal, Chaudhuri, Radhakrishnan
Sorting ≥ one of the operations must cost $\Omega(n \log n)$.

By symmetry, if

<table>
<thead>
<tr>
<th></th>
<th>Ins</th>
<th>Del</th>
<th>Find Min</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority queues</td>
<td>$n \log n$</td>
<td>$n \log n$</td>
<td>$n \log n$</td>
</tr>
<tr>
<td>Search trees</td>
<td>$n \log n$</td>
<td>$n \log n$</td>
<td>$n \log n$</td>
</tr>
<tr>
<td>Heap</td>
<td>$n \log n$</td>
<td>$n \log n$</td>
<td>$n \log n$</td>
</tr>
<tr>
<td>Double linked list</td>
<td>$n \log n$</td>
<td>$n \log n$</td>
<td>$n \log n$</td>
</tr>
</tbody>
</table>
The Randomized Complexity of Maintaining the Minimum

A simple data structure

The Randomized Complexity of Maintaining the Minimum
The Randomized Complexity of Maintaining the Minimum

The FindAny problem /\{Ins, Del, \{FindAny, return an arbitrary element and its current rank.\}/, Not harder than the FindMin problem. /\(\log n\)/.

- \textsc{FindAny}, return an arbitrary element and its current rank.
- \textsc{Ins}, \textsc{Del},

Data structure:

- The \textsc{FindAny} problem

The Randomized Complexity of Maintaining the Minimum
The Randomized Complexity of Maintaining the Minimum Lowerbounds

Theorem. For any deterministic data structure with cost for \( \text{Ins} \) and \( \text{Del} \) at most \( t \), the cost of \( \text{FindAny} \) is

\[
\frac{\sqrt{22}}{u} - 1.
\]

Theorem. For any randomized data structure with expected cost for \( \text{Ins} \) and \( \text{Del} \) at most \( t \), the expected cost of \( \text{FindMin} \) is

\[
\frac{\sqrt{2}}{n^{e/2}} t - \frac{1}{2}.
\]

Lower bounds

The Randomized Complexity of Maintaining the Minimum
The Randomized Complexity of Maintaining the Minimum

An adversary strategy for FINDMIN

An infinite binary tree. Initially, all elements are in the root.

The Randomized Complexity of Maintaining the Minimum
The Randomized Complexity of Maintaining the Minimum

If \( v \) and \( w \) are on the same path from the root, then \( v \) and \( w \) are incomparable in the algorithms poset.

A comparison pushes 2 elements one level down.

If \( n \) elements are deleted at some node and \( b_i \) is the first among them, then replacing \( \text{Del}(b_i) \) by \( \text{FindAny} \) takes at least \( \frac{n}{2} \) comparisons.

There exists a node where \( \frac{n + 3}{n} \) elements were deleted.

\[ u + \frac{n}{2} \]

\[ \sum \text{of the depths of } q_1, \ldots, q_n \]

If \( n \) elements are deleted at some node and \( b_i \) is the element that would have been returned by \( \text{FindAny} \), assign \( b_i \) to the node where it is deleted.

Consider \( \text{Ins}(a_i) \) \( \text{Ins}(a_i) \) \( \text{Del}(b_i) \) \( \text{Ins}(a_i) \) \( \text{Ins}(a_i) \) \( \text{Del}(b_i) \).

A comparison pushes 2 elements one level down.

If \( v \) and \( w \) are on the same path from the root, then \( v \) and \( w \) are incomparable in the algorithms poset.

Adversary answers consistently.
The Randomized Complexity of Maintaining the Minimum An explicit sorting adversary

When the elements are sorted, each element must lie in a distinct leaf.

\[
\sum_{\text{depths of nodes}} \leq \frac{\log n \cdot 2}{\log n}
\]

\[
\sum_{\text{number of comparisons}} \leq \frac{n \log n}{2}
\]