Soft Sequence Heaps

Gerth Stølting Brodal
Aarhus University

SIAM Symposium on Simplicity in Algorithms (SOSA), January 11, 2021
Heap

value

EXTRACTMIN
= lowest intersected line

5
4
3
2
1

time

1 2 3 4 5

Insertions

↑ INSERT(x)

↓ EXTRACTMIN()

Soft Heap

value

Car-pooling
equal values

5
4
3
2
1

time

1 2 3 4 5

Extraction

2 4

New corruptions
(created by EXTRACTMIN)

Soft heap properties

- EXTRACTMIN can increase values (corruptions)
  Returns new corruptions
- \( \leq \varepsilon N \) corrupted elements in soft heap, \( 0 \leq \varepsilon \leq \frac{1}{2} \), \( N = \# \) insertions

(Other operations not discussed in this talk: MAKESHAP, MELD, FINDMIN, DELETE)
# Soft heap results

<table>
<thead>
<tr>
<th>Soft heaps</th>
<th>INSERT / EXTRACTMIN</th>
<th>Reference</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduced car-pooling</td>
<td>O(log (\frac{1}{\varepsilon})) / O(1)</td>
<td>Chazelle ESA98*/JACM00 *2018 ESA Test-of-Time award</td>
<td>Selection MST O(m\cdot\alpha(m, n))</td>
</tr>
<tr>
<td>Binomial trees</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“A simpler ... soft heaps”</td>
<td></td>
<td>Pettie, Ramachandran JACM02</td>
<td>MST optimal Unknown complexity</td>
</tr>
<tr>
<td>Balanced binary trees</td>
<td>O(log (\frac{1}{\varepsilon})) / O(1)</td>
<td>Kaplan, Zwick SODA09</td>
<td></td>
</tr>
<tr>
<td>“Soft heaps simplified”</td>
<td>O(1) / O(log (\frac{1}{\varepsilon}))</td>
<td>Kaplan, Tarjan, Zwick SICOMP13</td>
<td></td>
</tr>
<tr>
<td>Balanced binary trees</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Report corruptions</td>
<td>O(1) / O(log (\frac{1}{\varepsilon}))</td>
<td>Kaplan, Kozma, Zamir, Zwick SOSA19</td>
<td>Heap selection (and related) Simplifying Frederickson JCSS93</td>
</tr>
<tr>
<td>Tag corrupted reported items</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corruptions only EXTRACTMIN</td>
<td>O(1) / O(log (\frac{1}{\varepsilon}))</td>
<td>Kaplan, Kozma, Zamir, Zwick SOSA19</td>
<td></td>
</tr>
<tr>
<td>Soft sequence heaps</td>
<td>O(log (\frac{1}{\varepsilon})) / O(1)</td>
<td>Brodal SOSA21</td>
<td></td>
</tr>
</tbody>
</table>

Time bounds are all amortized
Application of Soft Heaps – \(O(n)\) Selection

**function** select\((A, k)\)

if \(k = 1\) then
    return min\((A)\)

\(Q = \text{softheap}(1/3)\)

for \(a \in A\) do
    \(Q.\text{INSERT}(a)\)

for \(i = 1\) to \(|A|/3\) do
    pivot = \(Q.\text{EXTRACTMIN}()\)
    small, large = partition\((A, \text{pivot})\)

if \(k \leq |\text{small}|\) then
    return select\((\text{small}, k)\)

return select\((\text{large}, k - |\text{small}|)\)

\(T(n) \leq T(2n/3) + O(n)\)

use soft heap to find pivot

Chazelle JACM00
Application of Soft Heaps – O(k) Heap Selection

\[
\text{function } \text{select}(\text{root}, k) \\
\text{S} = \{\text{root}\} \\
\text{Q} = \text{softheap}(1/4) \\
\text{Q}.\text{INSERT}(\text{root}) \\
\text{for } i = 1 \text{ to } k - 1 \text{ do} \\
    (e, C) = \text{Q}.\text{EXTRACTMIN}() \\
    \text{if } e \text{ not corrupted then} \\
    \quad C = C \cup \{e\} \\
\text{for } e \in C \text{ do} \\
    \quad \text{Q}.\text{INSERT}(e.\text{left}) \\
    \quad \text{Q}.\text{INSERT}(e.\text{right}) \\
    \quad \text{S} = \text{S} \cup \{e.\text{left}, e.\text{right}\} \\
\text{return } \text{select}(\text{S}, k)
\]
Sequence Heaps

Sequence heap properties
- Sorted lists, each list a rank
- Two lists rank $r \implies \text{merge}$, rank $r+1$
- Rank $r$ list $\leq 2^r$ values
- $N \text{ INSERT} \implies \text{rank} \leq \log N$
- INSERT and EXTRACTMIN amortized $O(\log N)$

**INSERT**($x$)
- Create rank 0 list containing $x$
- while two list have equal rank do
- merge the two lists

**EXTRACTMIN**()
- Find list with smallest head element $e$
- Remove and return $e$
Soft Sequence Heap properties
- Sorted lists, each list a rank
- Prune every 2\textsuperscript{nd} element of a new list of even rank > \log \frac{1}{\varepsilon}
- x pruned ⇒ \{ x \} \cup C(x) added to C(y) where y successor of x
- Rank r list ⇒ size ≤ \sqrt{2^r/\varepsilon}

\textsc{Insert}(x)
Create rank 0 list containing x
while two list have equal rank r do
  merge the two lists
if r even and r > \log 1/\varepsilon then
  prune list

\textsc{ExtractMin()}
Find list with smallest head element e
if |C(x)| = 0 then
  Remove and return e
else
  Remove and return an element from C(e)

How can this work?
- Only O(\sqrt{n/\varepsilon}) elements are not pruned
Solution
- Not all pruned elements need to be considered corrupted
Suffix-min pointers and witness sets

- Each non-pruned element has a corruption set $C(e)$ and witness-set $W(e)$
- $x \in C(e) \implies x \leq e$
- $x \in W(e) \implies e \leq x$
- $x$ corrupted $\iff x \in C(e')$ for some $e'$ and $x \not\in W(e'')$ for any $e''$
- When EXTRACTMIN removes $e$, $W(e)$ is reported as corrupted
Analysis – corruptions $\leq \varepsilon n$

- $C(e)$ doubles when pruning $\Rightarrow |C(e)| \leq 2^{(r-\log\frac{1}{\varepsilon})/2} \leq \sqrt{2^r/\varepsilon}$
- "width" doubles when merging and increases by $\sqrt{2^r/\varepsilon}$ when pruning $\Rightarrow$ "width" $\leq \varepsilon n$
Summary - Soft Sequence Heaps

- At most $\epsilon n$ corruptions in heap
- `INSERT` and `EXTRACTMIN` amortized time $O(\log \frac{1}{\epsilon})$
- **Witness-sets** used in analysis and for reporting corruptions
  - can be removed from construction if reporting not needed
  - only $\sqrt{n/\epsilon}$ elements are not in corruption sets (previous constructions $\Theta(n)$)

Further results in paper
- Discuss how to remove buffering insertions from previous constructions

Open problems
- I/O & cache oblivious soft heaps with $O(B)$ operations taking $O(1)$ I/Os?
- Other applications of soft heaps?