Strict Fibonacci Heaps

Gerth Stølting Brodal
Aarhus University

George Lagogiannis
Agricultural University of Athens

Robert Endre Tarjan
Princeton University & HP

The Problem — Priority Queues

- INSERT(value, key)
- FINDMIN
- DELETEMIN / DELETE(&value)
- MELD(Q₁, Q₂)
- DECREASEKEY(&value, Δ)

Applications

- **Shortest Path**
  Dijkstra (1956)

- **Minimum Spanning Tree**
  Borůvka (1926)
  Jarník (1930)

\[
\begin{align*}
\text{INSERT/DELETEMIN} & \quad (m+n) \cdot \log n \\
\text{+ DECREASEKEY} & \quad m+n \cdot \log n \\
\text{MST only} & \quad m \cdot \beta(m,n)
\end{align*}
\]

Fredman, Tarjan 1984
## History

<table>
<thead>
<tr>
<th>Operation</th>
<th>Binary heaps</th>
<th>Binomial queues</th>
<th>Fibonacci heaps</th>
<th>Run-relaxed heaps</th>
<th>Strict Fibonacci heaps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert</td>
<td>log $n$</td>
<td>log $n$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FindMin</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Delete</td>
<td>log $n$</td>
<td>log $n$</td>
<td>log $n$</td>
<td>log $n$</td>
<td>log $n$</td>
</tr>
<tr>
<td>Meld</td>
<td>-</td>
<td>log $n$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DecreaseKey</td>
<td>log $n$</td>
<td>log $n$</td>
<td>log $n$</td>
<td>n</td>
<td>1</td>
</tr>
</tbody>
</table>

Amortized complexity (Tarjan 1983)

Arrays Complicated

Pointer Based
Technical History

- **Binary heaps**: 1964
  - Heap-order: Rigid structure
  - Forest Linking

- **Binomial queues**: 1978

- **Fibonacci heaps**: 1984
  - Subtrees cut
  - Cascades $\Rightarrow$ Amortized DECREASEKEY

- **Run-relaxed heaps**: 1988
  - Global control
  - Redundant counters

- **Brodal 1995**: Local control
  - Redundant counters

- **Brodal 1996**: Local redundant counters
  - Heap order violations

- **Strict Fibonacci heaps**: 2012
  - Global partial control
  - Pigeonhole principle

---

**Diagram**

- **Binary heaps diagram**
  - Nodes: 6, 27, 54, 36, 13, 86

- **Fibonacci heaps diagram**
  - Nodes: 8, 86, 17, 11, 16, 7, 24, 42
  - Minimum node: 4

---

**Table**

<table>
<thead>
<tr>
<th>Structure</th>
<th>Year</th>
<th>Key Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary heaps</td>
<td>1964</td>
<td>Heap-order: Rigid structure, Forest Linking</td>
</tr>
<tr>
<td>Binomial queues</td>
<td>1978</td>
<td></td>
</tr>
<tr>
<td>Fibonacci heaps</td>
<td>1984</td>
<td>Subtrees cut: Cascades $\Rightarrow$ Amortized DECREASEKEY</td>
</tr>
<tr>
<td>Run-relaxed heaps</td>
<td>1988</td>
<td>Global control: Redundant counters</td>
</tr>
<tr>
<td>Brodal 1995</td>
<td>1995</td>
<td>Local control: Redundant counters</td>
</tr>
<tr>
<td>Brodal 1996</td>
<td>1996</td>
<td>Local redundant counters: Heap order violations</td>
</tr>
<tr>
<td>Strict Fibonacci heaps</td>
<td>2012</td>
<td>Global partial control: Pigeonhole principle</td>
</tr>
</tbody>
</table>

---

**Notes**

- Single tree implementation

---
Ideas

Invariants
1. **white** nodes share one color record
2. **white** children left of **red**
3. root is **red**

Intuition
1. Only maintain structure for **white** nodes
2. **Red** non-root nodes never increase degree
3. **DELETE**: O(1) **red** nodes → **white**

Definitions
1. **white** node **rank** = # **white** children
2. **DECREASEKEY**: cut + **mark** parent (if white)

Invariants
1. i’th rightmost **white** child of a **white** node:
   \[ \text{rank} + \text{#marks} \geq i - 1 \]
2. \( \text{#marks} + \text{#white roots} = O(\log n) \)

Theorem
\( \text{max rank} = O(\log n) \)
Priority Queue Operations

- **FINDMIN** = return root
- **INSERT** = make single red node + MELD
- **DELETE** = DECREASEKEY to $-\infty$ + DELETEMIN
- **MELD** = color smaller tree red + link + $O(1)$ transformations
  - root degree +1
- **DECREASEKEY** = cut + link with root + $O(1)$ transformations
  - root degree +1; white root (+1); marks (+1)
- **DELETEMIN** = cut root + find new root + $O(\log n)$ link
  - root degree +$O(\log n)$; white root +$O(\log n)$
  - + $O(\log n)$ transformations

**Invariant**

\[ R = \alpha \cdot \log (3/2 \cdot \#\text{white nodes} + \#\text{red nodes}) + \beta \]

- degree unmarked white nodes $\leq R$
- degree red and marked white nodes $\leq R - 1$
Transformations

Root degree reduction
Converts two red nodes to white; reduces the root degree by two; creates one new white root

White root reduction
Two white roots of rank r is replaced by one rank r+1; increases root degree by one

One node mark reduction
White node with ≥ 2 marks becomes a white root (unmarked); increases the root degree by one

Two node mark reduction
Two nodes of equal rank r with 1 mark become unmarked; one parent one more mark

r / m = rang / #marks
Representation

- **heap record**: size, root, color-record, non-linkable-child, Q-head, rank-list, singles, fix-list
- **color record**: flag, ref-count
- **Q-head**
- **rank-list**
- **rank**
- **inc**, **dec**
- **max rank**
- **white roots**
- **marked**
- **rank**
- **left**, **right**
- **node**
- **left-child**, **Q-prev**, **Q-next**
- **fix-list**
- **transformable**
- **singles**
- **nodes with marks**
- **transformable**

**Notes**:
- Points to unmarked white nodes with rank 1 and with white parent
- Nodes with marks
- Fix-list
## Binary heaps
- **Insert**: $\log n$
- **FindMin**: 1
- **Delete**: $\log n$
- **Meld**: -
- **DecreaseKey**: $\log n$

## Binomial queues
- **Insert**: $\log n$
- **FindMin**: 1
- **Delete**: $\log n$
- **Meld**: $\log n$
- **DecreaseKey**: $\log n$

## Fibonacci heaps
- **Insert**: $\log n$
- **FindMin**: 1
- **Delete**: $\log n$
- **Meld**: 1
- **DecreaseKey**: 1

## Run-relaxed heaps
- **Insert**: 1
- **FindMin**: 1
- **Delete**: $\log n$
- **Meld**: 1
- **DecreaseKey**: $\log n$

## Strict Fibonacci heaps
- **Insert**: 1
- **FindMin**: 1
- **Delete**: $\log n$
- **Meld**: 1
- **DecreaseKey**: 1

### Amortized Complexity
- Arrays
- Pointer Based

---

**Thank You**