15 - Fibonacci Heaps

Binomial Trees

The binomial tree of height $h$ is a tree obtained from two binomial trees of height $h-1$ joined at the roots.

<table>
<thead>
<tr>
<th>height</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>...</th>
<th>$h$</th>
</tr>
</thead>
<tbody>
<tr>
<td>degree of root</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>$h = d$</td>
<td></td>
</tr>
<tr>
<td>num. of nodes</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>$2^h = n$</td>
<td>$h = \lg n$</td>
</tr>
</tbody>
</table>
Binomial Heaps

We want to come up with a heap-like structure that will hold $n$ items. Can we use a binomial tree?

A binomial heap is a collection of binomial trees that each obey the min-heap property. For any integer $k > 0$, there is at most one $B_k$ in the collection.
\( k \leq \log n \)

- \( n \) nodes → Which \( B_i \)'s are in our binomial heap?

\[
\begin{align*}
    n \mod 2 &= 1 & \text{if } B_0 \\
    n \mod 2 &= 0 & \text{if } B_0
\end{align*}
\]

least significant bit tells us if \( B_0 \) is in our \( b \).heap

\[
\begin{align*}
    n &= 11110 = 30_{10} \\
    n &= 100000 = 32_{10}
\end{align*}
\]
Merging two binomial heaps

\[ n_1 = 1110 \]

\[ n_2 = 01111 \]

\[ n = n_1 + n_2 = 10101 \]

\[ O(n) \text{ time on regular heap} \]

\[ O(\log n) \text{ on B. heap} \]
Fibonacci Heaps

Collection of heap-ordered trees.
(close to binomial trees)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time (Amortized)</th>
</tr>
</thead>
<tbody>
<tr>
<td>H = makeHeap();</td>
<td>O(1)</td>
</tr>
<tr>
<td>insert(H, x);</td>
<td>O(1)</td>
</tr>
<tr>
<td>x = minimum(H);</td>
<td>O(1)</td>
</tr>
<tr>
<td>H = merge (H₁, H₂);</td>
<td>O(1)</td>
</tr>
<tr>
<td>decrease-key(H, x, Δ);</td>
<td>O(1)</td>
</tr>
<tr>
<td>deleteMin(H)</td>
<td>O(lg n)</td>
</tr>
<tr>
<td>delete(H, x)</td>
<td>O(lg n)</td>
</tr>
</tbody>
</table>
Fibonacci Heap implementation

- child pointers
- sibling pointers
- parent pointers