Fibonacci Heaps

It supports the following operations efficiently in amortized sense.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Amortized time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make-Heap</td>
<td>O(1)</td>
</tr>
<tr>
<td>Insert ((H, x))</td>
<td>O(1)</td>
</tr>
<tr>
<td>Min ((H))</td>
<td>O(1)</td>
</tr>
<tr>
<td>Meld ((H_1, H_2))</td>
<td>O(1)</td>
</tr>
<tr>
<td>Delete-min((H))</td>
<td>O((\log n))</td>
</tr>
<tr>
<td>Delete ((H, x))</td>
<td>O((\log n))</td>
</tr>
<tr>
<td>Decrease-key ((H, x, \Delta))</td>
<td>O(1)</td>
</tr>
</tbody>
</table>
2. Structure.

It is a collection of heap ordered trees.

Ex. (heap ordered tree)

parent has a lower key than its children

representation

- Siblings are doubly-linked in a cycle
- Parent pointer
- Child pointer (one child)
The roots of the heap-ordered trees are linked in a doubly linked cycle plus a pointer to the minimum key node.

A single node $x$ has:
4 pointers: (parent, child, l-sibling, r-sibling)
$p(x)$, child$(x)$, left$(x)$, right$(x)$

1 real (key)
$k(x)$

1 integer (item)
i$(x)$

1 bit (marking)
mark$(x)$

1 integer (degree)
degree$(x)$

Simple Manipulations

Linking: Given two heap-ordered trees, make root with bigger key the child of the other root.

$O(1)$ time
Simple Manipulations.

Unlinking:

\[ t_1 \rightarrow t_2 \rightarrow \cdots \rightarrow t_k \]

\[ \Downarrow \]

\[ t_1 \rightarrow \cdots \rightarrow t_k \]

O(1) Time.

Merge Cycles: Cut open both cycles and connect at their ends.

Potential function. Let \( z_1, z_2, \ldots, z_n \) be the sequence of operations.

\[ t_i(H) = \text{number of roots in the root list of } H \text{ after operation } z_i. \]

\[ m_i(H) = \text{number of marked nodes in } H \text{ after operation } z_i. \]
\[ P_i = \text{potential} \]
\[ = t_i(H) + 2m_i(H) \]

Initial potential \( P_0 = 0 \) (Empty F-heap)

\[ T(z_i) = \text{actual time for operation } z_i \]
\[ A(z_i) = \text{amortized time} \]
\[ = T(z_i) + (P_i - P_{i-1}) \]

\[ \sum_{i=1}^{n} T(z_i) = \sum_{i=1}^{n} (A(z_i) + (P_{i-1} - P_i)) \]
\[ = P_0 - P_n + \sum_{i=1}^{n} A(z_i) \]
\[ = (t_0 - t_n) + 2(m_0 - m_n) + \sum_{i=1}^{n} A(z_i) \]
\[ = -(t_n + 2m_n) + \sum_{i=1}^{n} A(z_i) \]

\[ \sum_{i=1}^{n} T(z_i) \leq \sum_{i=1}^{n} A(z_i) \]

**Operations**

**Make-heap**

1. Create null ptrs.
   \( P_0 = 0 \)
   Amortized cost = actual cost = \( O(1) \)

**Min (H)**

1. Return the minimum key in H
   \( P_i - P_{i-1} = 0 \)
   Amortized cost = actual cost = \( O(1) \)
**Meld** \((H_1, H_2)\):  
1. Merge root cycles of \(H_1\) and \(H_2\)  
2. Adjust \(\min(H)\) to be \(\min(\min(H_1), \min(H_2))\)  
   
   potential change  
   \[ P_i - P_{i-1} = (t_i + 2m_i) - (t_{i-1}(H) + 2m_{i-1}(H)) - (t_i(H) + 2m_{i-1}(H_i)) = 0 \]  
   
   amortized cost = Actual cost = \(O(1)\)

**Insert** \((H, x) \Rightarrow H'\)  
1. Create F-heap \(H_1\), which has only one node \(x\)  
2. Meld \((H, H_1)\)  
   
   Let \(H'\) be the new F-heap  
   \[ P_i(H') = t_i(H) + 1 + 2m_{i-1}(H) \]  
   \[ P_i - P_{i-1} = 1 \]  
   
   amortized cost = \(1\) + actual cost = \(O(1)\)

**Delete-min** \((H)\)  
1. remove the node with min. key from the root cycle  
2. Merge the root cycle with the cycle of children of this node
3. While two roots $r_1, r_2$ have same degree
   link $r_1, r_2$
endwhile

4. Adjust the minimum key

---

Step 3. To find the roots with equal degrees, we use an array $R$; we scan the root list and let $R[i]$ point to the root $r$ with degree $i$. If $R[i]$ is already occupied with root $r'$, link $r'$ with $r$ and try to place it in $R[i+1]$. 

Cost analysis. Let $D(n)$ be the max. degree of any node in an $n$-node F-heap.

**Actual cost**

Step 1. $O(1)$
Step 2. $O(1)$
Step 3. $O(t_{i-1}(H) + D(n))$ \[ O(t_{i-1}(H) + D(n)) \]
Step 4. $O(t_{i-1}(H) + D(n))$

**Potential change**

Potential afterward is at most $D(n) + 1 + 2m_{i-1}(H)$

Potential before $t_{i-1}(H) + 2m_{i-1}(H)$

Potential change $D(n) + 1 - t_{i-1}(H)$

Amortized cost: $O(t_{i-1}(H) + D(n)) + (D(n) + 1) - t_{i-1}(H)$

= $O(D(n) + O(t_{i-1}(H)) - t_{i-1}(H)$

= $O(D(n))$

Since we can scale up the units of potential to dominate the constant hidden in $O(t_{i-1}(H))$. 
\[ D(n) = O(\log n) \text{ for heaps used in F-heap}. \]

Decrease key and Delete: These two ops.
destroy the heap-ordered properties of trees in
F-heap. However, the change is not too much.

Decrease-key \((H, x, \Delta)\).

1. Unlink tree rooted at \(x\)
2. Decrease the key by \(\Delta\)
3. Add \(x\) to the root-cycle
4. Do cascading cuts. (not described)
   (Marking is used here).

Delete \((H, x)\)

1. Unlink tree rooted at \(x\)
2. Add the cycle of children of \(x\) to the
   root cycle
3. Dispose off \(x\).

We skip the proof that these two opps. take
\[ O(\log n) \text{ amortized} \]

Ex. 

\[
\begin{array}{c}
\text{\(\leq\)}
\end{array}
\]

\[
\begin{array}{c}
\text{\(\Rightarrow\)}
\end{array}
\]