## Binomial Heaps

<table>
<thead>
<tr>
<th></th>
<th>Leftist trees</th>
<th>Binomial heaps</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Actual</td>
</tr>
<tr>
<td>Insert</td>
<td>$O(\log n)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Remove min (or max)</td>
<td>$O(\log n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Meld</td>
<td>$O(\log n)$</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>
Min Binomial Heap

- Collection of min trees.
Node Structure

- **Degree**
  - Number of children.

- **Child**
  - Pointer to one of the node’s children.
  - Null iff node has no child.

- **Sibling**
  - Used for circular linked list of siblings.

- **Data**
Binomial Heap Representation

- Circular linked list of min trees.
Insert 10

- Add a new single-node min tree to the collection.
- Update min-element pointer if necessary.
• Combine the 2 top-level circular lists.
• Set min-element pointer.
Remove Min

- Empty binomial heap => fail.
Nonempty Binomial Heap

- Remove a min tree.
- Reinsert subtrees of removed min tree.
- Update binomial heap pointer.

A
Remove Min Tree

• Same as remove a node from a circular list.

• No next node => empty after remove.

• Otherwise, copy next-node data and remove next node.
Reinsert Subtrees

- Combine the 2 top-level circular lists.
  - Same as in meld operation.
Update Binomial Heap Pointer

- Must examine roots of all min trees to determine the min value.
Complexity Of Remove Min

• Remove a min tree.
  ▪ O(1).

• Reinsert subtrees.
  ▪ O(1).

• Update binomial heap pointer.
  ▪ O(s), where s is the number of min trees in final top-level circular list.
  ▪ s = O(n).

• Overall complexity of remove min is O(n).
Enhanced Remove Min

- During reinsert of subtrees, pairwise combine min trees whose roots have equal degree.
Examine the $s = 7$ trees in some order.
Determined by the 2 top-level circular lists.
Use a table to keep track of trees by degree.
Pairwise Combine
Combine 2 min trees of degree 0.
Make the one with larger root a subtree of other.
Update tree table.
Combine 2 min trees of degree 1.
Make the one with larger root a subtree of other.
Pairwise Combine

Update tree table.
Pairwise Combine

tree table
Pairwise Combine

Combine 2 min trees of degree 2.
Make the one with larger root a subtree of other.
Pairwise Combine

Combine 2 min trees of degree 3.
Make the one with larger root a subtree of other.
Pairwise Combine

Update tree table.
Pairwise Combine

tree table
Pairwise Combine

Create circular list of remaining trees.
Complexity Of Remove Min

- Create and initialize tree table.
  - $O(\text{MaxDegree})$.
  - Done once only.
- Examine $s$ min trees and pairwise combine.
  - $O(s)$.
- Collect remaining trees from tree table, reset table entries to $\text{null}$, and set binomial heap pointer.
  - $O(\text{MaxDegree})$.
- Overall complexity of remove min.
  - $O(\text{MaxDegree} + s)$.
Binomial Trees

• $B_k$ is degree $k$ binomial tree.

• $B_k$, $k > 0$, is:

```
      B_0
     /|
    / |\
   /  |  \
  B_1  B_2  ...
  /|
 / |
B_0
```

$B_k$,
Examples

$B_0$  $B_1$  $B_2$  $B_3$
Number Of Nodes In $B_k$

- $N_k = \text{number of nodes in } B_k$.

- $B_0, k > 0$, is:

- $N_k = N_0 + N_1 + N_2 + \ldots + N_{k-1} + 1 = 2^k$. 
Equivalent Definition

- \( B_k, \ k > 0, \) is two \( B_{k-1} \)'s.
- One of these is a subtree of the other.
\( N_k \) And MaxDegree

- \( N_0 = 1 \)
- \( N_k = 2N_{k-1} = 2^k \).
- If we start with zero elements and perform operations as described, then all trees in all binomial heaps are binomial trees.
- So, \( \text{MaxDegree} = O(\log n) \).