Fibonacci Heaps
# Fibonacci Heaps

<table>
<thead>
<tr>
<th></th>
<th>Fibonacci</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>insert</td>
<td>O(1)</td>
<td>O(log(n))</td>
</tr>
<tr>
<td>find</td>
<td>O(1)</td>
<td>N/A</td>
</tr>
<tr>
<td>union</td>
<td>O(1)</td>
<td>N/A</td>
</tr>
<tr>
<td>minimum</td>
<td>O(1)</td>
<td>O(1)</td>
</tr>
<tr>
<td>decrease key</td>
<td>O(1)</td>
<td>O(log(n))</td>
</tr>
<tr>
<td>delete</td>
<td>O(log(n))</td>
<td>O(log(n))</td>
</tr>
<tr>
<td>delete minimum</td>
<td>O(log(n))</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Fibonacci Heaps

Binomial Tree:
A binomial tree of order 0 is a single node.
A binomial tree of order k has a root of degree k and its children are roots of binomial trees of orders k-1, k-2, ..., 2, 1, 0 (in order).
A binomial tree of order k has $2^k$ nodes.
Fibonacci Heaps

Data Structures:
Circular doubly linked list of siblings (− − −)
All nodes have pointers to their parents
One pointer to a child
Fibonacci Heaps

Forest of binomial trees - key of node is less than keys of children
Fibonacci Heaps

Forest of binomial trees - key of node is less than keys of children
Node with minimum key is a root of one of the trees
Fibonacci Heaps

Forest of binomial trees - key of node is less than keys of children

Node with minimum key is a root of one of the trees

A node may have degree greater 2 but no larger than $O(\log(n))$
Fibonacci Heaps

Forest of binomial trees - key of node is less than keys of children
Node with minimum key is a root of one of the trees
A node may have degree greater 2 but no larger than $O(\log(n))$
Size of a subtree rooted in node of degree $k$ is $F_{k+2}$ where $F_k$ is the $k$th Fibonacci number
Fibonacci Heaps

Forest of binomial trees - key of node is less than keys of children

Node with minimum key is a root of one of the trees

A node may have degree greater 2 but no larger than $O(\log(n))$

Size of a subtree rooted in node of degree $k$ is $F_{k+2}$ where $F_k$ is the $k$th Fibonacci number
Fibonacci Heaps

**Operation**: Find Minimum

Simple lookup using Min Node pointer - $O(1)$
Fibonacci Heaps

Operation: Union of two heaps
   Attach higher numbered node to smaller
   Remove former root from linked list
   Add former root to children linked list - O(1)
Fibonacci Heaps

**Operation**: Union of two heaps
- Attach higher numbered node to smaller
- Remove former root from linked list
- Add former root to children linked list - $O(1)$
**Fibonacci Heaps**

**Operation:** Insert
- Add new node as a heap
- Attach to the root linked list - $O(1)$
Fibonacci Heaps

Operation: Delete Minimum
**Fibonacci Heaps**

**Operation**: Delete Minimum

Remove minimum node and make children roots
**Operation**: Delete Minimum

Remove minimum node and make children roots
Union roots of same degree until all roots have different degree
Fibonacci Heaps

Operation: Delete Minimum
  Remove minimum node and make children roots
  Union roots of same degree until all roots have different degree
Fibonacci Heaps

**Operation**: Delete Minimum

- Remove minimum node and make children roots
- Union roots of same degree until all roots have different degree
Fibonacci Heaps

Operation: Delete Minimum

- Remove minimum node and make children roots
- Union roots of same degree until all roots have different degree
- Reset Minimum Node pointer
Fibonacci Heaps

Operation: Decrease Key

Minimum Node
Fibonacci Heaps

Operation: Decrease Key

Marked nodes are those having had exactly one child promoted to a root previously.
Fibonacci Heaps

**Operation**: Decrease Key

Decrease the key
- if violation, cut from the tree,
- promote it to a root
- mark its parent if it is unmarked
- if the parent had been marked cut it from its tree, and promote it to a root, and unmark it.
Fibonacci Heaps

Operation: Decrease Key

Decrease the key
- if violation, cut from the tree,
- promote it to a root
- mark its parent if it is unmarked
- if the parent had been marked cut it from its tree, and promote it to a root, and unmark it.

Minimum Node

Diagram of Fibonacci Heap with nodes 0, 1, 2, 3, 4, 5, 6, 7, 8.
Operation: Decrease Key

Decrease the key
- if violation, cut from the tree,
- promote it to a root
- mark its parent if it is unmarked
- if the parent had been marked cut it from its tree, and promote it to a root, and unmark it. - O(1) !!!
Fibonacci Heaps

Operation: Decrease Key
  Decrease the key
  - Change the Minimum Node pointer
**Fibonacci Heaps**

**Operation:** Delete

Key of node to be deleted changed to minus infinity (decrease key operation)
Followed by simple delete minimum
**Fibonacci Heaps**

**Operation**: Delete

Key of node to be deleted changed to minus infinity (decrease key operation)

Followed by simple delete minimum
Operation: Delete

Key of node to be deleted changed to minus infinity
(decrease key operation)
Followed by simple delete minimum
**Operation**: Delete

Key of node to be deleted changed to minus infinity (decrease key operation)
Followed by simple delete minimum
Fibonacci Heaps

Operation: Delete

Key of node to be deleted changed to minus infinity (decrease key operation)
Followed by simple delete minimum - $O(\log(n))$
Fibonacci Heaps

Result:

Complexity of algorithms using priority queues is reduced!
Example: Shortest Path – $O((m+n)\log(n))$ with binary heap
$O(m + n\log(n))$ with Fibonacci heap