

# Fibonacci Heaps

Heap (Priority Queue) data structure designed using amortized analysis. Supports the following operations:

MAKE-HEAP :  $\Theta(1)$

INSERT :  $\Theta(1)$

MINIMUM :  $\Theta(1)$

EXTRACT-MIN :  $O(\log n)$

UNION :  $\Theta(1)$

DECREASE-KEY :  $\Theta(1)$

DELETE:  $O(\log n)$

Fibonacci (Fib) heaps perform better when the number of Decrease-Key operations is much less than the number of Extract-Min or Delete operations.

Traditional heaps can take as much as  $\Theta(\log n)$  time for these operations.

# Fibonacci Heaps

A collection of (min-)heap-ordered trees each of which is (exactly or approximately) a **binomial** tree.

A (unordered) Binomial tree is recursively defined as:

1.  $B_0$  consists of a single node.
2.  $B_k$  consists of two binomial trees  $B_{k-1}$  linked together such that one of them is a child of the other.

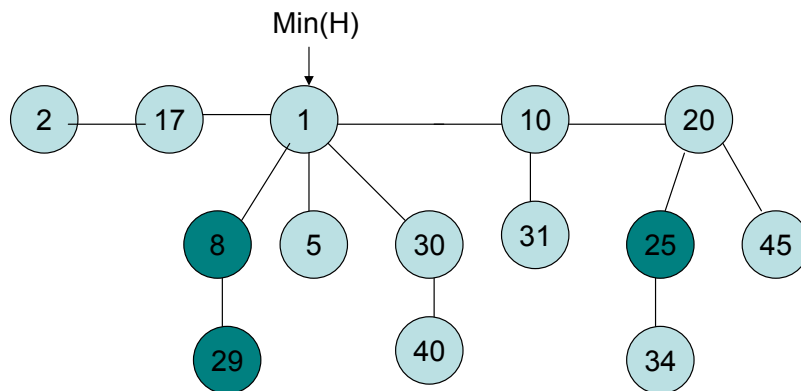
## Properties of Binomial tree $B_k$ :

1. Has  $2^k$  nodes.
2. height of tree is  $k$ .
3. there are exactly  $\binom{k}{i}$  nodes at depth  $i$ , for  $i = 0, 1, \dots, k$ .
4. the root has degree  $k$  (greater than any other node). The subtrees hanging from the root are  $B_0, B_1, \dots, B_{k-1}$  in some order.

If a  $n$ -node Fibonacci heap is a collection of unordered binomial trees then the maximum degree of any node  $D(n) = \lg n$ .

We will show that max degree  $D(n)$  is always bounded by  $O(\lg n)$  after every operation.

# Example



The root nodes of the trees are linked together in a circular, doubly linked list called the root list.

Each node has a pointer to its parent and pointers to one of its children.

The children are linked together in a circular, doubly linked list.

## Definitions

$\min(H)$  points to the minimum element in the Fib heap  $H$ .

$n(H)$  gives the number of nodes in  $H$ .

$t(H)$  gives the number of trees in  $H$ .

**Marking:** Some nodes in a Fib heap are **marked** (indicated by a boolean variable). A node  $x$  is marked if it has lost a child since the last time  $x$  was made the child of another node.

Newly inserted nodes are unmarked. Also a node  $x$  is unmarked whenever it is made the child of another node. Marking is done when doing Decrease-Key.

$m(H)$  gives the number of marked nodes in  $H$ .

**Potential Function:**  $\Phi(H) = t(H) + 2m(H)$

## Insert and Union

Simply add the new node to the root list.

Amortized cost of Insert is  $O(1)$  since the actual cost is  $O(1)$  and the increase in potential is  $O(1)$ :

$$(t(H) + 1) + 2m(H) - (t(H) + 2m(H)) = 1.$$

Union of two Fib heaps is done by concatenating the root lists of the two heaps.

Amortized cost of Union is  $O(1)$ .

Finding the minimum is  $O(1)$ .

## Extract-Min

1. Remove the minimum node from the heap and add its children (if any) to the root list.
2. Repeatedly find two nodes having the **same degree** in the root list and make the node with the larger key a child of the other. When a node  $x$  is made the child of another,  $x$  becomes unmarked.
3. Update pointer to the minimum node.

Actual cost is  $O(D(n) + t(H))$ .

Amortized cost is

$$\begin{aligned} &\leq O(D(n) + t(H)) + ((D(n) + 1) + 2m(H)) - (t(H) + 2m(H)) \\ &= O(D(n)) + O(t(H)) - t(H) \\ &= O(D(n)). \end{aligned}$$

since we can scale up the potential to dominate the constant hidden in  $O(t(H))$ .

## Decrease-Key

Given a Fib heap  $H$  and a pointer to a key  $x$ , we want to decrease its value to  $k$ .

This operation does not preserve the property that all trees in the Fib Heap are binomial trees. But we will show that the max degree  $D(n)$  is still  $O(\log n)$ .

Decrease-Key ( $H, x, k$ ):

Decrease the value of node  $x$  to  $k$ . If this decrease violates the heap property we do the following:

1. Cut the subtree rooted at  $x$  and add it to the root list. Unmark  $x$ .
2. If the parent of  $x$  is unmarked then mark the parent of  $x$ .

Otherwise, repeatedly cut the marked nodes (and its subtrees) till we reach an unmarked parent or the root.

# Delete

Deleting a node  $x$  can be implemented by doing  $\text{Decrease-Key}(H, x, -\infty)$  and then doing an  $\text{Extract-Min}$ .

Amortized cost is the sum of the Amortized costs of  $\text{Decrease-Key}$  and  $\text{Extract-Min}$ .

## Amortized Cost of Decrease-Key

The actual cost of Decrease-Key is  $O(c)$  where  $c$  is the number of cuts made.

Each cut (except for the last one) cuts a marked node and makes it unmarked.

The change in potential is at most

$$(t(H) + c) + 2(m(H) - (c - 1) + 1) - (t(H) + 2m(H)) = 4 - c$$

Thus the amortized cost is at most  $O(c) + 4 - c = O(1)$ , since we can scale up the potential to dominate the constant hidden in  $O(c)$ .

# Maximum Degree of Fibonacci Heaps

**Lemma 1.** *Let  $x$  be any node in a Fibonacci heap, and let  $\text{degree}[x] = k$ . Let  $y_1, y_2, \dots, y_k$  denote the children of  $x$  in the order in which they were linked to  $x$ , from the earliest to the latest. Then  $\text{degree}[y_1] \geq 0$  and  $\text{degree}[y_i] \geq i - 2$  for  $i = 2, 3, \dots, k$ .*

**Proof.**  $\text{degree}[y_1] \geq 0$ .

For  $i \geq 2$ , when  $y_i$  is linked to  $x$ ,  $\text{degree}[x] = i - 1$ .

At the time of linking,  $\text{degree}[y_i] = i - 1$  and it could have lost at most one child.  $\square$

**Lemma 2.** *Let  $F_k$  be the  $k(\geq 0)$ th Fibonacci number. Then*

$$F_{k+2} = 1 + \sum_{i=0}^k F_i$$

$$F_{k+2} \geq \left(\frac{1+\sqrt{5}}{2}\right)^k$$

**Proof:** By induction on  $k$ .

**Lemma 3.** *Let  $x$  be any node in a Fibonacci heap, and let  $k = \text{degree}[x]$ .*

*Then,  $\text{size}(x) \geq F_{k+2}$ , where  $\text{size}(x)$  denotes the size of the subtree rooted at  $x$ .*

**Proof.** Let  $s_k$  denote the minimum possible value of  $\text{size}(z)$  over all nodes  $z$  such that  $\text{degree}[z] = k$ .

$$s_0 = 1, s_1 = 2 \text{ and } s_2 = 3.$$

Let  $y_1, y_2, \dots, y_k$  denote the children of  $x$  in the order in which they were linked to  $x$ . Then

$$\begin{aligned} \text{size}(x) &\geq s_k = 2 + \sum_{i=2}^k s_{\text{degree}[y_i]} \\ &\geq 2 + \sum_{i=2}^k s_{i-2} \end{aligned}$$

since  $\text{degree}[y_i] \geq i - 2$  and  $s_k$  is monotonically increasing with  $k$ .

By induction on  $k$ , we can show that  $s_k \geq F_{k+2}$ .

The base cases  $k = 0, 1$  are trivial.

$$\begin{aligned}
s_k &\geq 2 + \sum_{i=2}^k s_{i-2} \geq 2 + \sum_{i=2}^k F_i \\
&= 1 + \sum_{i=0}^k F_i = F_{k+2} \quad \square
\end{aligned}$$

**Theorem 1.** *Let  $D(n)$  be the maximum degree in any  $n$  node Fibonacci heap. Then  $D(n) = O(\log n)$ .*

**Proof.** Let  $x$  be any node in an  $n$ -node Fib heap and let  $k = \text{degree}[x]$ .

$$\text{Then } n \geq \text{size}(x) \geq F_{k+2} \geq \left(\frac{1+\sqrt{5}}{2}\right)^k.$$

Thus  $k = O(\log n)$ .  $\square$