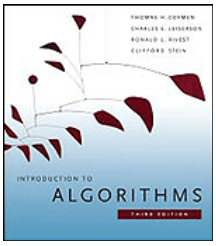

Chapter 19 - Fibonacci Heaps
Introduction to Algorithms, Third Edition
by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein
The MIT Press © 2009 *Citation*

Previous Next

19.3 Decreasing a Key and Deleting a Node

In this section, we show how to decrease the key of a node in a Fibonacci heap in $O(1)$ amortized time and how to delete any node from an n-node Fibonacci heap in $O(D(n))$ amortized time. In Section 19.4, we will show that the maximum degree $D(n)$ is $O(\lg n)$, which will imply that FIB-HEAP-EXTRACT-MIN and FIB-HEAP-DELETE run in $O(\lg n)$ amortized time.

Decreasing a Key

In the following pseudocode for the operation FIB-HEAP-DECREASE-KEY, we assume as before that removing a node from a linked list does not change any of the structural attributes in the removed node.

```
FIB-HEAP-DECREASE-KEY(H, x, k)
1 if k > x.key
2     error "new key is greater than current key"
3 x.key = k
4 y = x.p
5 if y ≠ NIL and x.key < y.key
6     CUT(H, x, y)
7     CASCADING-CUT(H, y)
8 if x.key < H.min.key
9     H.min = x
```

```
CUT(H, x, y)
1.remove x from the child list of y, decrementing y.degree
2. add x to the root list of H
3. x.p = NIL
4. x.mark = FALSE
```

```
CASCADING-CUT(H, y)
1. z = y.p
2. if z ≠ NIL
3.     if y.mark== FALSE
4.         y.mark= TRUE
5.     else CUT(H, y, z)
6.         CASCADING-CUT(H, z)
```

The FIB-HEAP-DECREASE-KEY procedure works as follows. Lines 1–3 ensure that the new key is no greater than the current key of x and then assign the new key to x. If x is a root or if $x.key \geq y.key$, where y is x's parent, then no structural changes need occur, since min-heap order has not been violated. Lines 4–5 test for this condition.

If min-heap order has been violated, many changes may occur. We start by ***cutting*** x in line 6. The CUT procedure "cuts" the link between x and its parent y, making x a root.