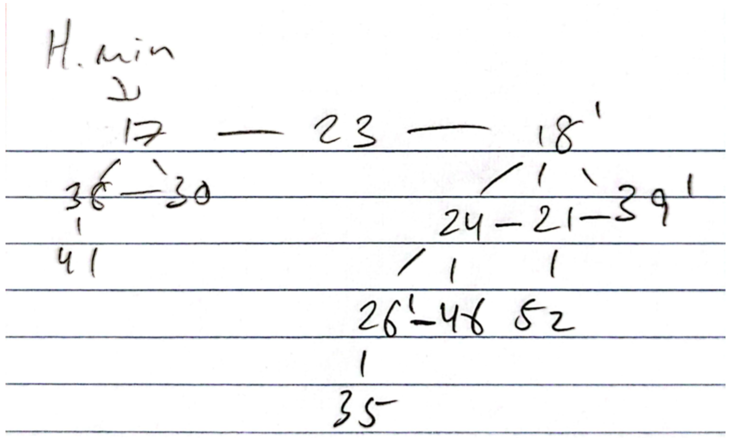
David Zhang – dzhang [at] cs [dot] toronto [dot] edu

19.2-1 Show the Fibonacci heap that results from calling Fib-Heap-Extract-Min on the Fibonacci heap shown in Figure 19.4(m).



19.3-1 Suppose that a root x in a Fibonacci heap is marked. Explain how x came to be a marked root. Argue that it doesn't matter to the analysis that x is marked, even though it is not a root that was first linked to another node and then lost one child.

A node is marked when exactly one of its children has been cut. A marked *root* x was previously cut from $H.min$ following an extract min operation. If x becomes a child of some other root, then its marked bit can be used to pay for a subsequent cut. Otherwise, x remains a marked root or is eventually removed via an extract min. In both cases, the extra potential that x's marked bit introduces is either unused or not significant enough to meaningfully pay for any operation.

19.3-2 Justify the $O(1)$ amortized time of Fib-Heap-Decrease-Key as an average cost per operation by using aggregate analysis.

If a decrease key causes c nodes to be cut, then it must be that one node caused the cascading, and the remaining $c-1$ nodes had each lost a child to at least one (possibly cascading) decrease key. Note that each decrease key operation pays for both cutting the requested node and marking the first unmarked ancestor following the single or cascading cut. Such an ancestor can be cut by at most one other cascading cut before being sent to the root list and having its marked bit reset. Hence, c calls to decrease key are required to cut $\leq 2c$ nodes, so charging each operation $O(1)$ is sufficient to cover the total cost of cascades.

19.4-1 Professor Pinocchio claims that the height of an n-node Fibonacci heap is $O(\log n)$ Show that the professor is mistaken by exhibiting, for any positive integer n, a sequence of Fibonacci-heap operations that creates a Fibonacci heap consisting of just one tree that is a linear chain of n nodes.

Suppose that all inserts are on monotonically decreasing keys. We can trivially construct a single-node chain with a single insert. Let $k \geq 1$ and suppose that we can construct any k node chain via the standard Fibonacci heap operations. To extend this chain by one node,

- Insert 3 nodes $x < y < z$ and then extract min;
- Delete z.

y will have degree 1 after z is linked to it during the first extract min. The k node chain also has degree 1, so it is linked to y as well. The subsequent delete completes the $k+1$ node chain.

19.4-2 Suppose we generalize the cascading-cut rule to cut a node x from its parent as soon as it loses its kth child, for some integer constant k. (The rule in Section 19.3 uses $k = 2$). For what values of k is $D(n) = O(\log n)$?

Let k be arbitrary and suppose that Fibonacci heaps were modified so that a node was only cut after losing k children. Let x be any node in such a heap with $x.degree = j$. Let $y_1, \ldots y_j$ denote x's children in the order that they were linked to x, from least recent to most recent. When y_i was linked to x, $y_1, \ldots, y_{i-1}$ must have been linked as well so it must have had degree $i - 1$. y_i can lose at most $k - 1$ children without being cut. Hence, $y_1, \cdots y_{k-1}$ each have degrees ≥ 0 and y_i must have degree $\geq i - k$, for $i \geq k$.

Define a sequence

$$F_i = \begin{cases} 0 & i = 0 \\ 1 & 1 \leq i \leq k - 1 \\ F_{i-1} + F_{i-k} & i \geq k \end{cases}$$

Note that this specifies the Fibonacci sequence when $k = 2$. By induction, $F_{i+k} = 1 + \sum_{r=0}^{i} F_r$.

Let $size(x)$ denote the number of nodes that are in x's subtree, and let s_i denote the minimum size of any tree whose root has degree i. Notice that $n \geq size(x) \geq s_j$.

Next, we prove that for any $i \geq 0$, $s_i \geq F_{i+k}$. When $i \leq k - 1$, $s_i = i + 1 = F_{i+k}$ since each child can have degree 0. If $i \geq k$, then suppose that for all $r < i$, $s_r \geq F_{r+k}$. Then, it follows that $s_i \geq k + \sum_{l=k}^{i} s_{y_l.degree} \geq k + \sum_{l=k}^{i} s_{l-k} \geq k + \sum_{l=k}^{i} F_l = 1 + \sum_{l=0}^{i} F_l = F_{l+k}$ where the third inequality is by induction.

Let ℓ be a positive root of $x^k = x^{k-1} + 1$, which exists and satisfies $1 < \ell \leq k^{1/k-1}$. We again induct on i, this time to show that $F_{i+k} \geq \ell^i$. If $i \leq k - 1$, then $F_{i+k} \geq \ell^i$ is true whenever $F_{2k-1} \geq \ell^{k-1}$ holds. By definition, $\ell^{k-1} \leq k = F_{2k-1}$. Hence, we can proceed to consider values of $i \geq k$. First, suppose that for all $r < i$, $F_{r+k} \geq \ell^r$. Then, it follows by induction that $F_{i+k} = F_{i+k-1} + F_i \geq \ell^{i-1} + \ell^{i-k} = \ell^{i-k}(\ell^{k-1} + 1) = \ell^{i-k} \cdot \ell^k = \ell^i$.

We have shown that $n \geq size(x) \geq s_j \geq F_{j+k} \geq \ell^j$ so $n \geq \ell^j$. Hence, $j \leq log_\ell n$ so $D(n) = O(\log n)$.