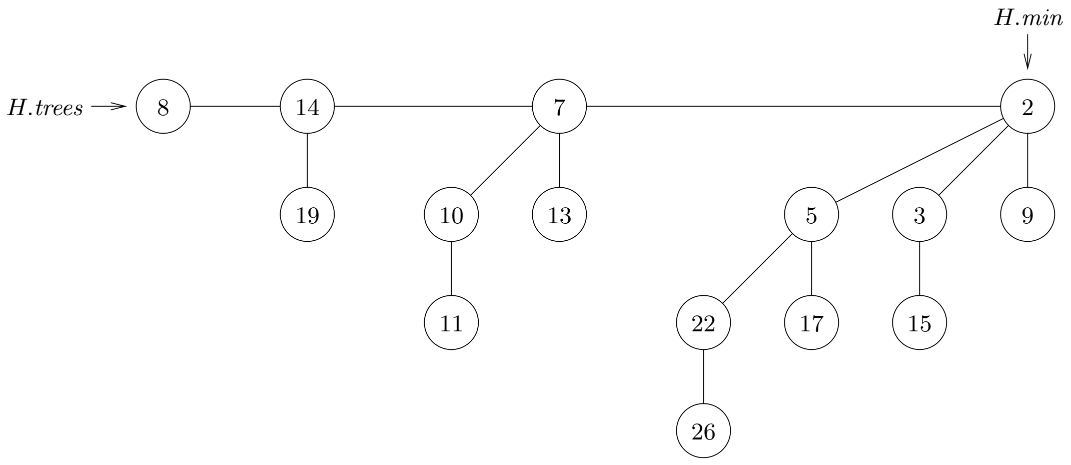
CSCE 750, Homework 6

This assignment covers material from the lectures on binomial heaps and Fibonacci heaps, in preparation for Quiz 6. The third edition of the book has a chapter on Fibonacci heaps (Chapter 19) and covers binomial heaps in some detail in Problem 19-2 on pages 527–529. The fourth edition only mentions Fibonacci heaps briefly on page 478 (and a few other places); it does not mention binomial heaps at all. I will assume that only my online notes for both types of heaps are available to you.

NIT1: Show the binomial heap that results from calling DELETEMIN(H) on the binomial heap H shown below:

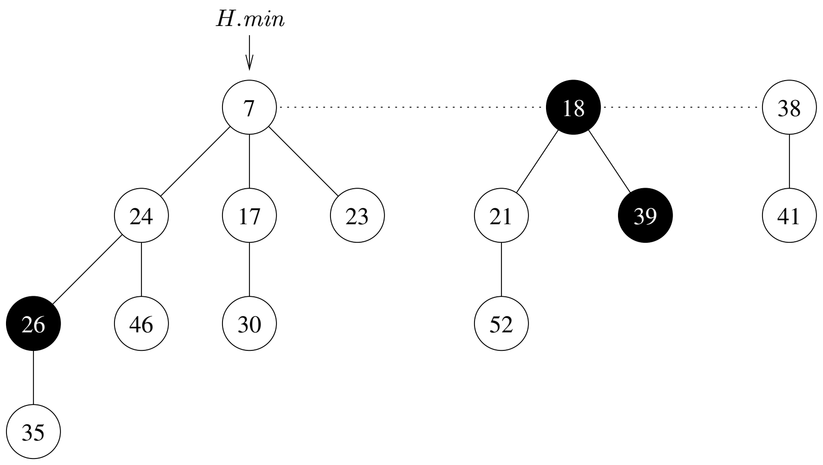


NIT2 (OPTIONAL): Profs. Misner, Thorne, and Wheeler need to implement binomial heaps on a device with limited memory, and so they are looking for ways to save space.

1. Prof. Misner suggests omitting the *parent* pointer from each node, and instead making the *right_sibling* field of each rightmost child point to the parent (instead of being NULL as with the standard implementation). She shows that one can still implement all the mergeable heap operations with the same asymptotic run time, except for DECREASEKEY (and thus DELETE as well), which now take $\Theta((\lg n)^2)$ worst-case time. (This may be an acceptable trade-off, depending on the application.) How is this done, and why could DECREASEKEY take that long?
2. Prof. Thorne suggests instead omitting the *degree* field from each node, computing the degree of a node on the fly as needed, by counting its children. All the operations are still implementable, but now MERGE (and hence DELETEMIN and INSERT) now take $\Theta((\lg n)^2)$ worst-case time. How is this done, and why does MERGE take so long?

3. Prof. Wheeler suggests a minor tweak of Prof. Thorne's idea: Omit the *degree* field from each node but include the single heap-wide attribute $H.size$, which contains the number of elements in the heap H. Show that this change allows all the operations to be implemented with the same worst-case asymptotic running time as with the standard binomial heap implementation. (*Hint:* The shape of a binomial heap is uniquely determined by its size.)
4. What about eliminating both the *parent* pointer *and* the *degree* field from each node. Is this workable? What operations break, if any?

NIT3 [3rd ed. Page 518, Ex 19.2-1]: Show the Fibonacci heap that results from calling FIB-HEAP-EXTRACT-MIN on the Fibonacci heap shown below [3rd ed. Figure 19.4(m)]:



NIT4: Show the result of inserting keys 0, 1, 2, 3, 4, 5, 6, 7 into an initially empty Fibonacci heap, then extracting the minimum.

NIT5 [3rd ed. Page 526, Ex 19.4-1]: Professor Pinocchio claims that the height of an n-node Fibonacci heap is $O(\lg n)$. 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.

NIT6 [paraphrased from 3rd ed. Page 529, Problem 19-3a]: You wish to augment a Fibonacci heap H to support the new heap operation

$$\text{FIB-HEAP-CHANGE-KEY}(H, x, k)$$

without changing the amortized running time of any other Fibonacci-heap operations. This operation changes the key of node x to the value k. Give an efficient implementation of FIB-HEAP-CHANGE-KEY, and analyze the amortized running time of your implementation for the cases in which k is greater than, less than, or equal to $x.key$.