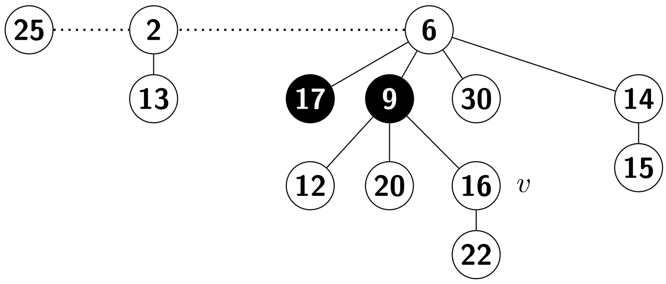
University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
Marc Fuchs

Algorithm Theory
Exercise Sheet 7

Due: Friday, 5th of December, 2025, 10:00 am

Exercise 1: Short questions *(6 Points)*

(a) Consider the following Fibonacci heap (black nodes are marked, white nodes are unmarked). How does the given Fibonacci heap look after a `decrease-key(v, 1)` operation and how does it look after a subsequent `delete-min` operation? *(4 Points)*



(b) Create a new method on the Fibonacci heap data structure called `Delete-node`(v), which deletes node v from the Fibonacci heap in $O(\log n)$ amortized time. Explain the runtime. *(2 Points)*
Hint: You may want to reuse the methods of Fibonacci heaps you already know.

Exercise 2: Worst Case Decrease *(7 Points)*

We've seen in the lecture that Fibonacci heaps are only efficient in an *amortized* sense. However, the time to execute a single, individual operation can be large. Show that in the worst case, the `decrease-key` operation can require time $\Omega(n)$ (for any heap size n).

Hint: Describe an execution in which there is a decrease-key operation that requires linear time.

Exercise 3: Fibonacci Heap simplification *(7 Points)*

Suppose we "simplify" Fibonacci heaps such that we do *not* mark any nodes that have lost a child and consequentially also do *not* cut marked parents of a node that needs to be cut out due to a `decrease-key`-operation.
Is the *amortized* running time

(a) ... of the `decrease-key`-operation still $\mathcal{O}(1)$? *(2 Points)*

(b) ... of the `delete-min`-operation still $\mathcal{O}(\log n)$? *(5 Points)*

Explain your answers.
Remark: You should NOT re-do the amortized analysis or search for a new potential function. Instead, think of what the implications are, i.e., do the statements from the lecture still work after this change? Especially, can we still bound the size/rank of a tree/node as we did in the lecture?