Data Structures

Fall 2020, Homework # 4

Date: Jan. 4, 2021

1. (20 pts) Suppose that we start from an empty Fibonacci heap.

   (a) We perform a sequence of operations that includes only insertions of elements and find-and-remove-min operations, but does not include any decrease-priority operations. Is it possible for the resulting Fibonacci heap to include a tree with exactly five nodes in it? Explain why (by finding a sequence of operations of this type that produces a five-node tree) or why not.

   (b) For the question in (a) above, what is your answer (and why) if decrease-priority operations are allowed?

2. (20 pts) Suppose that we are using a union-find structure for a family of disjoint sets of elements that have a total ordering (that is, we can compare any two elements and determine which one is smaller, in constant time). We wish to modify the $find(x)$ operation so that, instead of returning an arbitrary member of the set containing x, it always returns the smallest member. Describe a way of performing this modification with the same time per operation (in terms of its O-notation) as the original union-find data structure.

   (a) Explain why it does not work to change which node gets linked to which in a union in order to make the minimum element be the root of the tree for its set.

   (b) Propose a method that works. (Hint: you may add extra information to the tree nodes, but do not change the connectivity of the tree.)

3. (20 pts)

   (a) For any positive integer n, give a sequence of Fibonacci-heap operations that creates a Fibonacci heap consisting of just one tree that is a linear chain of n nodes.

   (b) For any positive integer n, give a sequence of skew-heap operations that creates a skew heap consisting of just one tree that is a linear chain of n nodes.

4. (30 pts) Consider an undirected graph (given its adjacency list) with n vertices and m edges.

   (a) Give an algorithm that counts the connected components using a union-find data structure.

   (b) How much time does the above algorithm take? Why?

   (c) Can you count the connected components in $O(n + m)$ time using perhaps a different data structure/algorithm? Show how and analyze your algorithm.

5. (10 pts) Draw the decision tree for bubble-sort for an array A[0..2] of $n = 3$ elements. Recall that in the first pass, the largest element is going to be moved to the very right of the array.