10 Algorithms 2 (jkf21)

(*a*) True or false? Using the standard operations of a mergable priority queue, it is possible to construct a Fibonacci heap with the structure of a linked list, i.e. one node in the root list and it has one child, which has one child, ... to height n, where n is the number of nodes in the Fibonacci heap.

If true, outline the sequence of operations to achieve it; if false, prove why it cannot be achieved. [6 marks]

(*b*) Two programmers are worried about the difficulty of implementing a Fibonacci heap because the root list and every parent node's child list is cyclic.

(*i*) The first programmer says that they could use *acyclic* doubly linked lists instead of the usual cyclic lists, provided they make three adjustments:

- always move the MIN node to the head of the root list when the minimum changes;
- the pointer to the min node becomes a 3-tuple containing the key count and pointers to the first and last elements of the root list; and
- parents contain pointers to their first and last children instead of one pointer to an arbitrary child.

The programmer believes these changes will not alter the big-O costs for any of insert, extract-min, decrease-key, and destructive-union. Are they correct? Justify your answer by considering the work done for each operation in turn. [4 marks]

(*ii*) The second programmer says that, since they only need a Fibonacci heap implementation to speed up Dijkstra's algorithm on very large graphs, they could make the lists acyclic but not make the other changes. Are they correct? Justify your answer carefully. [4 marks]

(*c*) It is suggested that we can work out whether a directed graph G is acyclic by running Dijkstra's algorithm with a Fibonacci heap and checking whether any decrease-key operation causes a node to be cut out of the parent's child list. Either prove the correctness of this algorithm or provide a counterexample. [3 marks]

(*d*) You overhear suggestions to replace the Fibonacci heap used by Dijkstra's algorithm as a priority queue with a sorted linked list, a red-black tree, or a hash table. Deduce the big-O running times of Dijkstra's algorithm with these three alternative priority queues. [3 marks]