Data Structures - Assignment no. 6

Remarks:

- **Write both your name and your ID number very clearly on the top of the exercise. Write your exercises in pen, or in clearly visible pencil. Please write *very* clearly.**
- **Give correctness and complexity proofs for every algorithm you write.**
- **For every question where you are required to write pseudo-code, also explain your solution in words.**

1. On an initially empty Fibonacci heap, carry out the following sequence of operations: insert(27), insert(17), insert(19), insert(20), insert(24), insert(12), insert(11), insert(10), insert(14), insert(18), deletemin, decreasekey(19, 7), delete (17), decrease-key(24,5), deletemin.

   After each operation, draw the resulting structure of the Fibonacci heap. (Whenever elements enter the root list, they are inserted to the right of the current minimum. Successive linking of the root list also starts with the element to the right of the removed minimum and continues cyclicly.)

2. Let H be a Fibonacci heap consisting of a single tree. The tree consists of three nodes with keys $key[x] = 7$, $key[y] = 25$, and $key[z] = 31$, such that $y = parent[z]$, and $x = parent[y]$. The node that contains y is marked. Find a sequence of legal operations (insert, deletemin, decreasekey, delete) by which H could have emerged from an empty Fibonacci heap.

3. 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.

4. Suppose we generalize the "cut rule" (in the implementation of decrease-key operation for a Fibonacci heap) to cut a node x from its parent as soon as it loses its kth child, for some integer constant k. (The rule that we studied uses k = 2.) For what values of k can we upper bound the maximum degree of a node of an n-node Fibonacci heap with O(log n)?

5. Professor Cohen has devised a new data structure based on Fibonacci heaps. A *Cohen heap* has the same structure as a Fibonacci heap and implements exactly the same abstract data type. The implementations of the operations are the same as for Fibonacci heaps, except that insert and meld perform successive linking as their last step. What are the worst-case and amortized running times of all operations on Cohen heaps?

6. We wish to augment a Fibonacci heap H to support two new operations without changing the amortized running time of any other Fibonacci-heap operations.

   - Give an efficient implementation of the operation FIB-CHANGE-KEY(H, x, k), which changes the key of node x to the value k. Analyze the amortized running time of your implementation for the cases in which k is greater than, less than, or equal to key[x].
   - Give an efficient implementation of FIB-PRUNE(H, r), which deletes $min(r, n[H])$ nodes from H. Which nodes are deleted should be arbitrary. Analyze the amortized running time of your implementation. (Hint: You may need to modify the data structure and potential function.)