Data Structures - Assignment no. 3

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.**
- **Recall that 80% of the theoretical exercises must be submitted. The exercises can and must be worked on and submitted alone.**
- **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. Insert the keys 5, then 9 and then 2 to the heap depicted in Figure 1. Then perform delete-min four times. Now draw the resulting heap.

2. Describe an algorithm that melds two binary heaps, represented by arrays, into one binary heap. Denote by n the sum of the sizes of the heaps. (Assume that no key appears more than once in the input). Try to make the algorithm as asymptotically efficient as possible. (Hint: the solution is very easy, and can be described in one or two lines).

3. Describe an algorithm that prints the k smallest elements in a binary heap. You can assume that the heap is represented as an array or as a tree, whichever is more comfortable for you. You may also assume that no key appears more than once in the heap. The algorithm should take $O(k \log k)$ time. The algorithm should not modify the heap. Give: (i) pseudo-code; (ii) an explanation of the algorithm; (iii) an explanation why it is correct; and (iv) an explanation why the running time is indeed $O(k \log k)$.
   Note: Observe that getting an algorithm that runs in time $O(k \log n)$, where n is the size of the heap, is easy – just perform k delete-mins. (In order to avoid modifying the heap, you need to undo your actions, which takes another $O(k \log n)$ time).

4. A d-ary heap is like a binary heap, but instead of 2 children, nodes have d children.
   (a) How would you represent a d-ary heap in an array?
   (b) What is the height of a d-ary heap of n elements in terms of n and d?
   (c) Give an efficient implementation of *find-min*. Analyze its running time in terms of d and n.
   (d) Give an efficient implementation of *insert*. Analyze its running time in terms of d and n.
   (e) Give an efficient implementation of *decrease-key*(A, i, δ), which sets $A[i]$ to $\min(A[i], \delta)$ and updates the heap structure appropriately. Analyze its running time in terms of d and n.

5. 1) Prove the following properties of binomial trees:

   (a) A binomial tree of rank k has 2^k vertices.

   (b) A binomial tree of rank k has depth k.

   (c) A binomial tree of rank k has $\binom{k}{r}$ vertices of depth r.

6. (a) Draw the corresponding binomial heap at the end of each line in the following sequence.
   - Insert the keys 10, 20, 3, 8, 30, 2, 25, 22, 35, 1, 32 to an empty binomial heap H_1.
   - Insert the keys $60, 12, 30, 82$ to an empty binomial heap H_2
   - H_3=Meld(H_1, H_2)
   - Insert the keys $15, 18, 8, 10, 17, 8$ to an empty heap H_4.
   - H_5=Meld(H_3, H_4)
   - Delete-Min(H_5)
   - Decrease the key of the node in H_5 that contain '17' – to '10'
   - Decrease the key of the node in H_5 that contain '20' – to '0'

   (b) Say you perform the same sequence as in (a) on a binomial heap with "lazy meld". Draw the heap at the end of the sequence. Assume that:
   - when you meld two heaps H_1 and H_2 you put the trees of H_1 before the trees of H_2;
   - After successive linking you put the trees in the list sorted by increasing ranks;

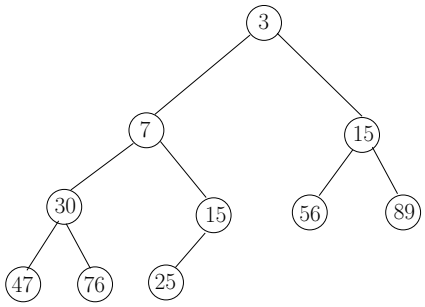

Figure 1: A Heap.