Data Structures - Assignment no. 7, June 13, 2007

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. (a) Solve the recurrence relation $T(n) = 2T(n/2) + n^2$ (you can use the recursion-tree method).

   (b) Prove the correctness of the solution you found in (a). (You can use induction).

2. Describe a data structure that implements a dictionary ADT. (The dictionary ADT maintains a set of keys, S, and supports the operations $insert(x)$, $delete(x)$ and $find(x)$). Let n be the number of operations performed on the data structure since it was created. The data structure should implement insert and delete in time $O(1)$ worst-case, and find in time $O(n \log n)$ worst case. Also, the amortized complexity of all operations should be $O(\log n)$. (In other words, the worst-case time of performing n operations should be $O(n \log n)$). Describe the data structure (no need to give pseudocode), and prove your claims about the running time.

3. A *well-ordered heap* is a regular heap, with the additional property that in each level of the heap the elements, viewed from left to right, are in increasing order.

   (a) Give an algorithm that gets as input n numbers, and outputs a well-ordered heap that contains these numbers.

   (b) Prove that the algorithm that you showed is optimal. That is, give a matching lower bound for the problem. (Hint: you'll probably want to prove the lower bound by giving a reduction, like we showed in class and in exercise 6, question $3(b)$).

4. (a) Suppose you are given an array of size n in which no element appears more than once, and two integers k_1 and k_2 such that $1 \leq k_1 \leq k_2 \leq n$. Describe an $O(n)$ time algorithm that returns all of the elements whose ranks in the array are $k_1, k_1 + 1, k_1 + 2, \ldots, k_2$. Note that you do not have to return these elements in sorted order. (Note: The *rank* of an element in an array is the number of elements smaller or equal to it).

   (b) What running time could you get if we did require that the elements be returned in sorted order?

5. Suppose you are given 3 sorted arrays of size n. You may assume that no element appears in the input more than once. Describe an algorithm that computes the median of the union of these arrays in time $O(\log n)$. Prove the correctness of your algorithm and prove the running-time bound.