# Data Structures - Assignment no. 6

Date due: 3/9/02

**Remarks:**

- For every algorithmic problem, try to find the most efficient solution. Analyze the complexity of every algorithm you give, and in case it is not trivial, prove its correctness.
- Unless you are explicitly asked otherwise, please describe algorithms only to the extent which makes them clear to a reasonable reader. You may use plain language, or pseudo-code, whichever is more convenient.

1. (a) Show that in an AVL tree of height $n$, the depth of every leaf is at least $\lceil \frac{n}{2} \rceil$ (hint: separate the odd and the even cases).
   (b) Show that for every non-negative $n$, there exists an AVL tree of height $n$, that has a leaf of depth $\lceil \frac{n}{2} \rceil$.
2. Show an efficient implementation of a data-structure that supports the following operations:
   - **insert(x)** : inserts an element $x$ into the structure
   - **find(x)** : returns whether $x$ is in the structure
   - **delete(x)** : deletes $x$ from the structure
   - **print_first(k)** : prints (visits) the $k$ smallest elements in the structure.
3. Show that the best-case complexity of the quick-sort algorithm, when restricted to inputs with distinct elements, is in $\Omega(n \log n)$.
4. Consider the following version of the comparison model. You are given a data-structure which, instead of one sequence, contains two sequences $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$, where one of the sequences is a permutation of the other. The black-box structure only supports the operation $\mathsf{compare}(i, j)$, which compares $a_i$ and $b_j$, and returns 1 if $a_i > b_j$, $-1$ if $a_i < b_j$, and zero if they are equal. Show a randomized algorithm in this model that sorts the sequence $a_1, \ldots, a_n$, whose expected-case complexity is $\Theta(n \log n)$.
5. Let $T_1$ and $T_2$ be two binary trees, each with $n$ nodes (and no keys in these nodes). Show that $T_2$ can be obtained from $T_1$ by applying at most $2n - 2$ rotations.
6. Show an algorithm that receives as input a sequence $a_1, \ldots, a_n$ of real numbers, satisfying $0 \leq a_i \leq 1$ for every $i$. The output of the algorithm should be a permutation $b_1, \ldots, b_n$ of the input sequence, such that
$$\sum_{i=1}^{n-1} |b_{i+1} - b_i| \leq 1 + \frac{1}{n}$$
The running time of the algorithm should be linear in $n$.