Homework Assignment #6

Note

This assignment is due 2:10PM Tuesday, April 16, 2013. Please write or type your answers on A4 (or similar size) paper. Drop your homework by the due time in Yih-Kuen Tsay's mail box on the first floor of Management College Building II. Late submission will be penalized by 20% for each working day overdue. You may discuss the problems with others, but copying answers is strictly forbidden.

Problems

There are five problems in this assignment, each accounting for 20 points. (Note: problems marked with "(X.XX)" are taken from [Manber 1989] with probable adaptation.)

1. Perform insertions of the numbers 7, 6, 3, 1, 4, 5, 2 (in this order) into an empty AVL tree. Show each AVL tree after a number has been inserted. If re-balancing operations are performed, please also show the tree before re-balancing and indicate what type of rotation is used in the re-balancing.

2. The *Partition* procedure for the `Quicksort` algorithm discussed in class is as follows, where *Middle* is a global variable.

   Partition $(X, Left, Right)$;
   begin
   $pivot := X[Left]$;
   $L := Left$; $R := Right$;
   while $L < R$ **do**
   while $X[L] \leq pivot$ and $L \leq Right$ **do** $L := L + 1$;
   while $X[R] > pivot$ and $R \geq Left$ **do** $R := R - 1$;
   if $L < R$ **then** $swap(X[L], X[R])$;
   $Middle := R$;
   $swap(X[Left], X[Middle])$
   end

   (a) Apply the *Partition* procedure to the following array (assuming that the first element is chosen as the pivot).

6	14	9	10	12	13	2	11	1	7	15	5	3	8	16	4

   Show the result after each exchange (swap) operation.

   (b) Apply the `Quicksort` algorithm to the above array. Show the result after each partition operation.

3. (6.10) Find an adequate loop invariant for the main while loop in the *Partition* procedure of the `Quicksort` algorithm, which is sufficient to show that after the execution of the last two assignment statements the array is properly partitioned by $X[Middle]$. Please express the loop invariant as precisely as possible, using mathematical notation.

4. (6.21) The input is a set S with n real numbers. Design an $O(n)$ time algorithm to find a number that is *not* in the set. Prove that $\Omega(n)$ is a lower bound on the number of steps required to solve this problem.

5. (6.32) Prove that the sum of the heights of all nodes in a complete binary tree with n nodes is at most $n - 1$. (A complete binary tree with n nodes is one that can be compactly represented by an array A of size n, where the root is stored in $A[1]$ and the left and the right children of $A[i]$, $1 \leq i \leq \lfloor \frac{n}{2} \rfloor$, are stored respectively in $A[2i]$ and $A[2i + 1]$. Notice that, in Manber's book a complete binary tree is referred to as a balanced binary tree and a full binary tree as a complete binary tree. Manber's definitions seem to be less frequently used. Do not let the different names confuse you. "Balanced binary tree" in the problem description is the same as "complete binary tree")