ICS621 Homework 2: Water Jugs & AVL Trees

Problem 8-4 from CLRS. Suppose that you are given n red and n blue water jugs, all of different shapes and sizes. All red jugs hold different amounts of water, as do the blue ones. Moreover, for every red jug, there is a blue jug that holds the same amount of water, and vice versa. Your task is to find a grouping of the jugs into pairs of red and blue jugs that hold the same amount of water. To do so, you may perform the following operation: pick a pair of jugs in which one is red and one is blue, fill the red jug with water, and then pour the warer into the blue jug. This operation will tell you whether the red or the blue jug can hold more water, or they have the same volume. Assume that such a comparison takes one time unit. Your goal is to find an algorithm that makes a minimum number of comparisons to determine the grouping. Remember that you may not directly compare two red jugs or two blue jugs.

a) Describe a deterministic algorithm that uses $\Theta(n^2)$ comparisons to group the jugs into pairs.

b) Prove a lower bound of $\Omega(n \lg n)$ for the number of comparisons that an algorithm solving this problem must make.

c) Give a randomized algorithm whose expected number of comparisons is $O(n \lg n)$, and prove that this bound is correct. What is the worst-case number of comparisons for your algorithm?

Problem 13-3 from CLRS. An **AVL tree** is a binary search tree that is height balanced: for each node x, the heights of the left and right subtrees of x differ by at most 1. To implement an AVL tree, we maintain an extra attribute in each node: $x.h$ is the height of node x. As for any other binary search tree T, we assume that $T.root$ points to the root node.

a) Prove that an AVL tree with n nodes has height $O(\lg n)$. (*Hint:* Prove that an AVL tree of height h has at least F_h nodes, where F_h is the hth Fibonacci number.)

b) To insert into an AVL tree, we first place a node into the appropriate place in binary search tree order. Afterward, the tree might no longer be height balanced. Specifically, the heights of the left and right children of some node might differ by 2. Describe a procedure BALANCE(x), which takes a subtree rooted at x whose left and right children are height balanced and have heights that differ by at most 2, i.e., $|x.right.h - x.left.h| \leq 2$, and alters the subtree rooted at x to be height balanced. (*Hint:* Use rotations.)

c) Using part (b), describe a recursive procedure AVL-INSERT(x, z) that takes a node x within an AVL tree and a newly created node z (whose key has already been filled in), and adds z to the subtree rooted at x, maintaining the property that x is the root of an AVL tree. As in TREE-INSERT from Section 12.3, assume that $z.key$ has already been filled in and that $z.left =$ NIL and $z.right =$ NIL; also assume that $z.h = 0$. Thus, to insert the node z into the AVL tree T, we call AVL-INSERT($T.root, z$).

d) Show that AVL-INSERT, run on an n-node AVL tree, takes $O(\lg n)$ time and performs $O(1)$ rotations.