Homework 3

Fundamental Algorithms, Fall 2002, Professor Yap

Due: Thu Oct 17, in class

INSTRUCTIONS:

- Please pick up the handout on AVL Trees in this directory. Recall that we will not cover Red-Black Trees from Chapter 13, but use AVL Trees instead.
- TIP OF THE DAY: Some students wanted extra practice questions. I suggest downloading from the OldHW directory found under the current hw directory.

---

1. (10 Points if correct answer, -8 if wrong answer, -2 points if unanswered) *Joe Dice is faced with a multiple choice question in which he gets 10 points if correct and -8 if wrong, and -2 points if left unanswered. He must choose one out of three possible answers. Now Joe happen to have no clue about this question.* Which of the following advice would you give Joe?
(a) Randomly choose one of the three answers.
(b) Leave the question unanswered.
(c) Does not matter what he does.

2. (10 Points) Prove by real induction that if $T(n) = T(n/2) + T(n/4) + 1$ then $T(n) = O(n^\alpha)$ where $\alpha = 0.694...$ is the solution to the equation
$$\frac{1}{2^\alpha} + \frac{1}{4^\alpha} = 1.$$
HINT: you will need to strengthen the hypothesis from $T(n) \le Cn^\alpha$ to $T(n) \le Cn^\alpha \pm g(n)$ for some $g(n) = O(n^\alpha)$.

3. (10+10+10 Points) Linear Time Median
(i) The linear time median algorithm in the Text has a time complexity which satisfies the recurrence
$$T(n) = T(n/5) + T(7n/10) + cn$$
where $c > 0$ is any constant. Determine the constant c from the description in the Text. Unlike the Text, do not worry about the difference between "$n - 6$" and "n", nor the difference between "$\lceil n/5 \rceil$" and "$n/5$".
(ii) Give the best constant $C > 0$ such that $T(n) \le Cn$ when n is sufficiently large. Note that C will depend on the c you found in part (i). As usual, you can choose your own boundary conditions.
(iii) Joe Quick suggests that we can simplify the linear time median algorithm by forming groups of 3 elements rather than groups of 5. How can you convince Joe that this is impossible?

---

4. (5+5+5+15 Points) There are three systematic ways to visit all the nodes in a binary search tree (BST). These are called **in-order**, **post-order** and **pre-order** traversals of the tree [CLRS, p. 254].
(i) Consider the BST in Figure 12.2 [CLRS, p. 257]. List the keys of this BST in an in-order traversal.
(ii) Repeat (i) using post-order traversal.
(iii) Repeat (i) using pre-order traversal.
(iv) Prove or disprove: If I give you two lists of keys, and tell you that one is a pre-order traversal of a tree T_0, and the other is the post-order traversal of the same tree, then you can re-construct the tree T_0. Assume the keys in T_0 are distinct.

5. (5+5 Points) Recall the $\mathtt{rotate}(u)$ operation where u is a node of a binary tree. A node u has three pointers $u.\mathtt{Left}, u.\mathtt{Right}, u.\mathtt{Parent}$ (possibly equal to nil).
(i) In the worst case, how many such pointers do $\mathtt{rotate}(u)$ update?
(ii) Describe an implementation of $\mathtt{rotate}$.

6. (10+15 Points) Consider a variant of the usual LookUp operation. Let T_u denote the subtree rooted at an arbitrary node u of a BST. The procedure LookUp(u, k) which searches T_u for the node v whose key is k. If such a node v exists, then v becomes the root of this subtree. If v does not exist, we return a nil and we do not care how the tree T_u has been transformed. We want to implement the LookUp procedure by repeated rotation. Here is one attempt:

```
LookUp(u, k)
0.   If u = nil return(nil);
1.   If u.Key = k, return u;
2.   If u.Key > k, return LookUp(rotate(u.Left), k);
3.   Return LookUp(rotate(u.Right), k)
```

(i) Why is this implementation is incorrect?
(ii) Give a correct version of this procedure. Remember we must reduce all pointer manipulations to rotations only.

7. (10+10+5 Points) Consider the BST in Figure 13.1(c) [CLRS, p. 275]. This happens to be an AVL tree. In showing the results of AVL operations, you must always draw the intermediate results after each rotation (you may schematically represent the parts that do not change).
(a) Insert the key "1" into this tree.
(b) Delete the key "15" from the original tree (not the tree after part (a)).
(c) Suppose you have an AVL tree with 20 nodes. What is the range for its height?