COT5405 ANALYSIS OF ALGORITHMS

Mid Term Exam I

Date: Feb 16, 2006, Tuesday
Time: 10:40am – 12:40pm
Professor: Alper Üngör (Office CSE 430)

This is a closed book exam. No collaborations are allowed. Your solutions should be concise, but complete, and handwritten clearly. Use only the space provided in this booklet, including the even numbered pages. You should answer all the questions to get full credit.

GOOD LUCK!

Your name:

	Credit	Max
Problem 1		20
Problem 2		20
Problem 3		20
Problem 4		20
Problem 5		20
Total		100

1. [20 = 10 + 10 points] BINARY SEARCH TREES

   Recall that inserting a new element into a binary search tree involves a topdown search and an append operation at the leaf level. Suppose we construct a binary search tree by successively inserting n distinct items into an initially empty tree, without ever rebalancing the tree.

   (a) How many different trees can you get for $n = 3$ items? Draw the trees.

   (b) Is it true that if you pick a random sequence of the n items then each of the possible trees is equally likely? Justify your answer.

2. [20=5+5+10 points] TREE-DEPTH AND RECURRENCE

A binary tree is full if each node has exactly zero or two children. Consider a full binary tree T with n leaves. Define the right-depth of a node v as the number of right edges from root to v. Let R_T denote the sum of right depths over all leaves of T.

(a) Which tree with n leaves minimizes R_T? What is R_T for this tree?

(b) Which tree with n leaves maximizes R_T? What is R_T for this tree?

(c) Prove that if every internal node has at least as many leaves in the left as in the right subtree then $R_T \leq n \log_2 n$.

3. [20 = 10+10 points] DYNAMIC PROGRAMMING

Given a sequence S of n integers (not necessarily positive), MAXIMUM SUM CONSECUTIVE SUBSEQUENCE PROBLEM asks to find a consecutive subsequence of S whose summation is maximized. for example for $S =< -6, 12, -7, 0, 14, -7, -3 >$, the maximum sum of 19 is achieved for the subsequence $< 12, -7, 0, 14 >$.

Note that it is straight-forward to design a $O(n^3)$-time algorithm for this problem. However, your goal is to design a dynamic programming algorithm with better running time, i.e., $o(n^3)$.

(a) Write and describe a recurrence to structure a dynamic programming algorithm for finding solving the MAXIMUM SUM CONSECUTIVE SUBSEQUENCE PROBLEM.

(b) Desribe and analyze your algorithm based on the recurrence you constructed in (a).

[Note/Hint: While there is a solution with $\Theta(n)$ running time, $\Theta(n^2)$ time solution is also worth full credit.]

4. [20 = 10+10 points] GREEDY ALGORITHMS

(a) You are asked to tile an $m \times n$ room, using any square tiles, minimizing the number of tiles used ($m \geq n$ are integers). Consider a greedy strategy which uses the largest fitting tile first. Does this algorithm give an optimal solution? Justify your answer.

(b) Consider the following version of the 0/1 Knapsack Problem. Given a set of n objects with weights $w_1 \geq w_2 \geq \ldots \geq w_n$, and profits $p_1 \leq p_2 \leq \ldots \leq p_n$, find a subset of objects such that the total weight is bounded by a given capacity W and the total profit is maximized. Describe the best greedy algorithm you can for solving this problem. Does your algorithm result in an optimal solution? Justify your answer.

5. [20 points] AMORTIZED ANALYSIS

   Recall that a QUEUE is a first-in-first-out data structure with two basic operations *enqueue* and *dequeue*, and a STACK is a first-in-last-out data structure with operations *push* and *pop*. Describe how to implement a queue with two stacks so that the amortized cost of each *enqueue* and *dequeue* operations is constant.