ECE608 Midterm 2 Spring 2011

Please fill in the following information.

THEN PUT YOUR EMAIL ADDRESS ON EVERY PAGE OF THE EXAM !

NAME: **SOLUTION**

Email (please put complete address):

Neatness counts. We will not grade what we cannot read.

Exam is worth 100 points. Please phrase your answer succinctly. Write your answer on the same page, on which the question is given. There are four questions and a total of six pages. Make sure you turn in all these pages.

Do not attempt to look at other students' work. Keep your answers to yourself.
Any sort of cheating will result in a zero grade.

Read and sign the statement below. Wait for instructions to start the examination before continuing to the next page.

"I signify that the work shown in this examination booklet is my own and that I have not received any assistance from other students nor given any assistance to other students."

_____(Signature)

Q1. Binary Search Tree (20 points total)

List all possible permutations of the following sequence which can result in generating the worst case binary search tree. Assume a sequence is read from left to right.

3, 2, 1, 4

Solution 1:
1234, 1243, 1432,1423
4321,4312,4123,4132

Q 2 (25 points) . Let b_n denote the number of all possible different binary trees with n nodes. Assume, $b_0 = 1$

(a) (15 points) For $n \geq 1$, write a recursive expression to compute b_n

(b) (10 points) What other problem we have studied that shows the same recurrence? Based on that problem find the solution for b_n

Solution 2:

(a) $$b_n = \sum_{k=0}^{n-1} b_k b_{n-1-k}$$

(b) It is similar to the optimal parenthesization problem for matrix-chain multiplication. Accordingly, the solution $b_n = \frac{1}{n+1}\binom{2n}{n}$, (which is the Catalan number, *C(n)*).

Q 3 (25 points). Suppose that we are given a key k to search for in a hash table with positions $0,1,...,m-1$, and suppose that we have a hash function h mapping the key space into the set $\{0,1,...,m-1\}$. The search scheme is as follows.

1. Compute the value $i \leftarrow h(k)$, and set $j \leftarrow 0$.

2. Probe in position i for the desired key k. If you find it, or if this position is empty, terminate the search.

3. Set $j \leftarrow j+1$. If $j = m$, the table is full, so terminate the search. Otherwise, set $i \leftarrow (i + j) \bmod m$, and return to step 2.

Assume that m is a power of 2.

Show that this scheme is an instance of the general "quadratic probing" scheme by exhibiting the appropriate constants c_1 and c_2 for the following general equation used for quadratic probing:

$$h(k,i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$$

Solution 3:
From how the probe-sequence computation is specified, it is easy to see that the probe sequence is $\langle h(k), h(k)+1, h(k)+1+2, h(k)+1+2+3, ..., h(k)+1+2+3+\cdots+i, ...\rangle$, where all the arithmetic is modulo m. Starting the probe numbers from 0, the ith probe is offset (modulo m) from $h(k)$ by

$$\sum_{j=0}^{i} j = \frac{i(i+1)}{2} = \frac{1}{2}i^2 + \frac{1}{2}i.$$

Thus, we can write the probe sequence as

$$h'(k,i) = \left(h(k) + \frac{1}{2}i + \frac{1}{2}i^2 \right) \bmod m,$$

which demonstrates that this scheme is a special case of quadratic probing.

Q4 (30 points). Dynamic Programming and Greedy Algorithms

Suppose it's nearing the end of the semester and you're taking n courses, each with a final project that still has to be done. Each project will be graded on the following scale: It will be assigned an integer number on a scale of 1 to $g > 1$, higher numbers being better grades. Your goal, of course, is to maximize your average grade on the n projects.

You have a total of $H > n$ hours in which to work on the n projects cumulatively, and you need to decide how to divide up this time. For simplicity, assume H is a positive integer, and you'll spend an integer number of hours on each project. To figure out how best to divide up your time, you've come up with a set of functions $\{f_i : i = 1, 2, ..., n\}$ (rough estimates, of course) for each of your n courses; if you spend $h \le H$ hours on the project for course i, you'll get a grade of $f_i(h)$. (You may assume that the functions f_i are *nondecreasing*: i.e. if $h < h'$, then $f_i(h) \le f_i(h')$.)

Using these functions $\{f_i\}$, you need to provide a dynamic programming/greedy approach to decide how many hours to spend on each project (in integer values only) so that your average grade, as computed according to the f_i, is as large as possible. To solve this problem, answer the following questions:

(a) **(15 points)** Define a suitable cost function and set up a recurrence equation to obtain an optimal solution. In order to be efficient, the running time of your algorithm should be polynomial in *n*, *g*, and *H*; none of these quantities should appear as an exponent in your running time.

(b) **(15 points)** Provide the complexity of your algorithm.

Solution 4:

(a) First note that it is enough to maximize one's *total* grade over the *n* courses, since this differs from the average grade by the fixed factor of *n*. Let the *(i,h)-subproblem* be the problem in which one wants to maximize one's grade on the first *i* courses, using at most *h* hours.

Let *A[i,h]* be the maximum total grade that can be achieved for this subproblem. Then *A[0,h] = 0* for all *h*, and $A[i,0] = \sum_{j=1}^{i} f_j(0)$. Now, in the optimal solution to the *(i,h)*-subproblem, one spends *k* hours on course *i* for some value of $k \in [0,h]$; thus

$$A[i,h] = \max_{0 \le k \le h} \{f_i(k) + A[i-1, h-k]\}$$

We also record the value of *k* that produces this maximum.

(b) From output *A[n,H],* and can trace-back through the entries using the recorded values to produce the optimal distribution of time. The total time to fill in each entry *A[i,h]* is *O(H)*, and there are *nH* entries, for a total time of $O(nH^2)$.