CPS 130 Exam 2

Summer 2002

3:30-4:45, Friday June 14th
Closed book exam

NAME:____

Problem	Max	Obtained
1	15	
2	20	
3 (a)	20	
3 (b)	10	
3 (b)	10	
4 (a)	5	
4 (b)	10	
4 (c)	10	
Total	100	

Comments:

- You can use any of the algorithms/procedures covered in class without describing them.
- When asked to describe an algorithm it is okay to do so with words (and a few accompanying pictures if it helps the description). Remember to include an argument for both correctness and running time.

HONOR CODE

I have obeyed the honor code.

SIGNATURE:____

[*15 points*] **Problem 1:**

1. Is it true that in the worst case, a red-black tree insertion requires $O(1)$ rotations?

2. Is it true that in the worst case a red-black tree deletion requires $O(1)$ node recolorings?

3. Is it true that walking a red-black tree with n nodes in pre-order takes $\Theta(n \log n)$ time?

4. Draw a left rotation and a right rotation.

5. What type of tree-walk on a red-black tree outputs the elements in sorted order?

6. Given a red-black tree with n elements, how fast can you sort them using the tree?

7. How fast can we build a red-black tree with n elements?

8. If a data structure supports an operation FOO such that a sequence of n FOO's takes $O(n \log n)$ time in the worst case, then the amortized time of a FOO operation is $\Theta(\qquad)$ while the actual time of a single FOO operation could be as low as $\Theta(\qquad)$ and as high as $\Theta(\qquad)$.

9. In order for dynamic programming to be applicable to optimization problems the structure of the optimal solution must satisfy a certain condition. What is this condition and what does it mean?

[*20 points*] **Problem 2:**

In this problem we consider a data structure $\mathcal{D}$ for maintaining a set of integers under the normal INIT, INSERT, DELETE, and FIND operations, as well as a COUNT operation, defined as follows:

- INIT($\mathcal{D}$): Create an empty structure $\mathcal{D}$.
- INSERT($\mathcal{D}$,x): Insert x in $\mathcal{D}$.
- DELETE($\mathcal{D}$,x): Delete x from $\mathcal{D}$.
- FIND($\mathcal{D}$,x): Return pointer to x in $\mathcal{D}$.
- COUNT($\mathcal{D}$,x): Return number of elements larger than x in $\mathcal{D}$.

Describe how to modify a standard red-black tree in order to implement $\mathcal{D}$ such that INIT is supported in $O(1)$ time and INSERT, DELETE, FIND, and COUNT are supported in $O(\log n)$ time.

[*40 points*] **Problem 3:**

A pharmacist has W pills and n empty bottles. Let $\{p_1, p_2, ..., p_n\}$ denote the number of pills that each bottle can hold.

a) Describe a greedy algorithm, which, given W and $\{p_1, p_2, ..., p_n\}$, determines the fewest number of bottles needed to store the pills. Prove that your algorithm is correct (that is, prove that the *first* bottle chosen by your algorithm will be in some optimal solution).

b) How would you modify your algorithm if each bottle also has an associated cost c_i, and you want to minimize the total cost of the bottles used to store all the pills?

Give a recursive formulation of this problem (formula is enough). You do not need to prove correctness.

(*Hint: Let $MinPill[i, j]$ be the minimum cost obtainable when storing j pills using bottles among* 1 *through i. Thinking of the* 0-1 KNAPSACK PROBLEM *formulation may help.*)

c) Describe briefly how you would design an algorithm for it using dynamic programming and analyse its running time.

[*25 points*] **Problem 4:**

In this problem we look at the amortized cost of insertion in a dynamic table. Initially the size of the table is 1. The cost of insertion is 1 if the table is not full. When an item is inserted into a full table, it first expands the table and then inserts the item in the new table. The expansion is done by allocating a table of size 3 times larger than the old one and copying all the elements of the old table into the new table.

a) What is the cost of the i-th insertion?

b) Using the accounting method, prove that the amortized cost of an insert in a sequence of n inserts starting with an empty table is $O(1)$.

b) Prove the same amortized cost by defining an appropriate potential function. You can use the standard notation $num(T)$ for the number of elements in the table T and $size(T)$ for the total number of slots (maximum size) of the table.