CSci 231 Exam 2

Fall 2004

2:30 - 3:55, Monday November 29th
Closed book exam

NAME:_____

Problem	Max	Obtained
1	10	
3 (a)	15	
3 (b)	5	
3 (c)	15	
3 (a)	15	
3 (b)	15	
3 (c)	15	
4	20	
Total	120	

[*10 points*] **Problem 1:**

1. If a data structure supports an operation FOO in $\Theta(\lg n)$ amortized time and $O(n)$ worst case time, is it true that sequence of n FOO's takes $O(n \log n)$ time in the worst case?

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

3. Is it true that any dynamic programming problems can be solved (faster) using a greedy algorithm?

4. 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 $\Theta($ $)$.

5. 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, the actual time of a single FOO operation could be as low as $\Theta($ $)$ and as high as $\Theta($ $)$.

[*35 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 $\mathcal{D}$.

a) Describe **briefly** 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. (In particular, for COUNT give pseudocode and draw a picture.)

b) Given an array $S[1..n]$ of integers, an *inversion* is a pair of elements $S[i]$ and $S[j]$, $i < j$, such that $S[i] > S[j]$. How many inversions does the array $[5, 2, 7, 1, 9, 4, 6]$ have?

c) Using the data structure $\mathcal{D}$ designed in problem a), describe an $O(n \log n)$ algorithm for computing the number of inversions in an array $S[1..n]$.

[*45 points*] **Problem 3:**

Consider (if you haven't already!) a quiz with n questions. For each $i = 1, 2, ..., n$, question i has integral point value $v_i > 0$ and requires $m_i > 0$ minutes to solve. Suppose further that no partial credit is awarded (unlike this quiz).

Your goal is to come up with an algorithm which, given $v_1, v_2, ..., v_n, m_1, m_2, ..., m_n$ and V, computes the minimum number of minutes required to earn at least V points on the quiz. For example, you might use this algorithm to determine how quickly you can get an A on the quiz.

a) Let $M(i, v)$ denote the minimum number of minutes needed to earn v points when you are restricted to selecting from questions 1 through i. Give a recurrence expression for $M(i, v)$.

We shall do the base case for you:

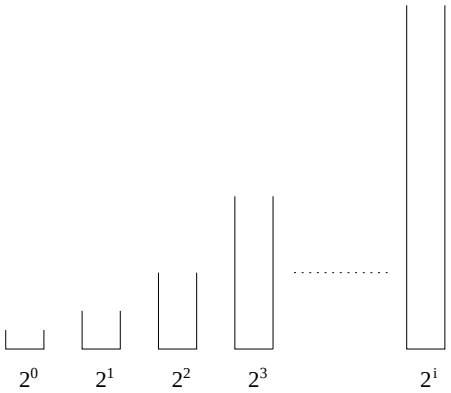
- $M(i, v) = 0$ for all i, and $v \leq 0$.
- $M(0, v) = \infty$ for $v > 0$

b)Write the recurrence relation for the running time of an algorithm implementing the recursive formulation above. What is the running time?

c) Give pseudocode for a dynamic programming algorithm to compute the minimum number of minutes required to earn V points on the quiz and analyze its running time.

[*20 points*] **Problem 4: EXTRA CREDIT**

Consider a *meta-stack* consistin of an infinite series of stacks $S_0, S_1, S_2, ...$ where the i-th stack S_i can hold at most 2^i elements. An element x is PUSH-ed onto a meta-stack by PUSH-ing x onto S_0. If S_0 is full, its elements are POP-ed from S_0 and PUSH-ed onto stack S_1. In general, if S_i runs full all its elements are PUSH-ed onto stack S_{i+1}.



What is the worst-case running time of a meta-stack PUSH operation, in a sequence of n such operations? What is its amortized cost?