due Friday, 12 October 2012, at 5:00 PM

A total of 38 points.

1. *Purpose: reinforce your understanding of chain matrix multiplication and greedy algorithms (when these fail).*

For each of the following greedy strategies for matrix chain multiplication, prove, by coming up with a counterexample, that it does not work.

(a) First multiply A_i by A_{i+1} where common dimension d_i is smallest and repeat on the reduced instance in which $A_i \cdot A_{i+1}$ is treated as a single matrix, until there is only one matrix.

(b) First multiply A_i by A_{i+1} where common dimension d_i is largest and repeat on the reduced instance in which $A_i \cdot A_{i+1}$ is treated as a single matrix, until there is only one matrix.

(c) First multiply A_i by A_{i+1} where the product $d_{i-1}d_id_{i+1}$ is minimized and repeat on the reduced instance in which $A_i \cdot A_{i+1}$ is treated as a single matrix, until there is only one matrix.

(d) First multiply A_i by A_{i+1} where the product $d_{i-1}d_id_{i+1}$ is maximized and repeat on the reduced instance in which $A_i \cdot A_{i+1}$ is treated as a single matrix, until there is only one matrix.

2 points each part

2. *Understanding recursive formulation and optimal substructure.*

Do Exercise 15.3-6 on page 390. You must write a recursive formulation before solving the given problem. This is not in the second edition, but the third edition is on two-hour reserve in the Textiles Library.

Hint: Consider choosing the very first exchange as a way of defining a smaller instance.

4 points for a recursive formulation; 3 points for proof that it exhibits optimal substructure when $c_k = 0$ *for all k; 3 points for showing giving a counterexample for the more general case.*

For all of the dynamic programming problems below, a large portion of the credit is devoted to the recursive formulation.

3. *Understanding recursive formulation and dynamic programming.*

Do Problem 15-2 on page 405. Also not in the second edition.

6 points

4. *Understanding recursive formulation and dynamic programming.*

Do Problem 15-9 on page 410. Also not in the second edition, but there's a description below.

***Breaking a string*:** A certain string processing language allows a programmer to break a string into two pieces. Because this operation copies the string, it costs n time units to break a string of n characters into two pieces. Suppose a programmer wants to break a string into many pieces. The order in which the breaks occur can affect the total amount of time used. For example, suppose that the programmer wants to break a 20-character string after characters 2, 8, and 10 (numbering the characters from left to right starting at 1). If she programs the breaks to occur in left-to-right order, then the first break costs 20 time units, the second break costs 18 time units and the third costs 12 time units for a total of 50. If the breaks are programmed in right-to-left order, however, then the first break costs 20 time units, the second 10, and the third 8, for a total of 38.

Design an algorithm, that, given the positions of the characters at which to break, determines the least-cost sequence of breaks. More formally, given a string S with n characters and an array $L[1 \ldots m]$ containing the break points, compute the lowest cost for a sequence of breaks, along with a sequence of breaks that achieves this cost.

8 points

5. *Understanding activity selection and when greedy algorithms fail.*

Do Exercise 16.1-3 on page 422 (16.1-4 on page 379 in 2/e).

3 points for each part