CS 317, Assignment 6

Due on 21 November in class

The assignment is all about dynamic programming, and so your algorithms must be dynamic programming algorithms unless otherwise stated.

1. This question refers to the Matrix Chain Multiplication as discussed in class.

   (a) Refine the algorithms MatrixChainMult from the lecture notes such that in addition to the table m_{ij} (which stores the cost of the optimal bracketing) you also fill in a data structure s that stores the actual optimal bracketing. You decide what s is, within the constraints that (a) it has to be usable for the process of performing the actual multiplication, and (b) your refined MatrixChainMult must maintain the original $O(n^3)$ running time.

   Explain how the structure s can be used afterward. Provide an argument for the correctness of your algorithm and explain how the original running time is maintained.

   (b) Give a divide and conquer algorithm MatrixChainMultiply(M, s, i, j) that actually performs the optimal matrix-chain multiplication $M_i \times \cdots \times M_j$. The inputs are the chain $M = \langle M_1, M_2, \ldots, M_n \rangle$ of matrices to be multiplied, the structure s returned by the algorithm that you developed for Question 1a, and the two indices i and j, $1 \le i \le j \le n$. The call MatrixChainMultiply$(M, s, 1, n)$ will thus optimally multiply the whole chain of matrices. Assume that MatrixMultiply(A, B) returns the product of the two matrices A and B.

2. You are given a directed acyclic graph $G = (V, E)$ with real-valued edge weights and two distinguished vertices s and t. The weight of a path is the sum of the weights of the edges in the path. Describe a dynamic-programming approach for finding the path from s to t with the largest weight. Prove the correctness of your algorithm and analyze its running time.

   Hint: Pay attention to the order of evaluation in the memoization structure, as this not as straightforward as in other dynamic programming algorithms.

3. A palindrome is a nonempty string over some alphabet that reads the same forward and backward. Examples of palindromes include all strings of length 1, `civic`, `racecar`, and `aibohphobia` (which is the officially unofficial name for the fear of palindromes). Design an efficient algorithm for finding the longest palindrome that is a substring of a given input string. For this problem a substring of the string s is obtained by deleting an arbitrary number of characters from s; the deleted characters need not be adjacent to each other. For example, given the input `character` your algorithm should return `carac`. Prove the correctness of your algorithm and analyze its running time.

Make sure you review the submission guidelines posted on the course's Web site before submitting.