The One-Page, 3-Problem, Pre-Spring-Break Problem Set

Question 1. Warm-up
Answer Exercise 1 (page 312) in Chapter 6 of your text. For part (c) include a justification of correctness and a time/space complexity analysis.

Question 2. Longest Paths In DAGs
Answer Exercise 3 (page 314) in Chapter 6 of your text. For part (b) include a justification of correctness and a time/space complexity analysis.

Question 3. Optimal Matrix Multiplication
If you are familiar with matrix multiplication, you can skip down to The Problem; otherwise read the next few paragraphs.

Given two vectors \( u = (u_1, \ldots, u_n) \) and \( v = (v_1, \ldots, v_n) \) the dot product (a.k.a. inner product) of \( u \) and \( v \) is given by \( u \cdot v = u_1v_1 + \ldots + u_nv_n \). Note that it takes \( n \) scalar multiplications to compute the dot product of two vectors of length \( n \).

Now suppose that \( A \) and \( B \) are \( n \times q \) and \( q \times m \) matrices (arrays), respectively. The product of \( A \) and \( B \) is the \( n \times m \) matrix \( P \) for which \( P[i, j] \) is the dot product of row \( i \) of \( A \) with column \( j \) of \( B \). Note that this is well-defined since the rows of \( A \) and the columns of \( B \) are all vectors of length \( q \). This operation is referred to as matrix multiplication.

If you aren’t familiar with matrix multiplication, don’t worry. Here are the only things you need to know to answer this question.

- Computing the product \( AB \) by directly using the definition above requires \( nqm \) scalar multiplications: There are \( nm \) entries in the product, and each requires \( O(q) \) multiplications to compute.
- In general, \( AB \neq BA \): Matrix multiplication is not commutative, so don’t even try!
- Matrix multiplication is, however, associative: \((A B) C = (A (B C))\). That is, a product \( A_1 \ldots A_k \) can be parenthesized in any order without changing the value of the result.

The Problem
While the order of evaluation of a chain \( A_1 \ldots A_k \) of matrices does not affect the value of the result, it can greatly affect the time required to compute the result! For example. If \( A, B, C \) have sizes \( 3 \times 20, 20 \times 4 \) and \( 4 \times 10 \) respectively, then

- Computing \( ((AB)C) \) takes \( 3 \cdot 20 \cdot 4 \) scalar multiplications for \( AB \), which is a \( 3 \times 4 \) matrix, plus \( 3 \cdot 4 \cdot 10 \) scalar multiplications for multiplying \( AB \) by \( C \), giving a total of \( 240 + 120 = 360 \) scalar multiplications.
- Computing \( (A(BC)) \) takes \( 20 \cdot 4 \cdot 10 \) scalar multiplications for \( BC \), which is a \( 20 \times 10 \) matrix, plus \( 3 \times 20 \times 10 \) scalar multiplications for multiplying \( A \) by \( (BC) \), giving a total of \( 800 + 600 = 1400 \) scalar multiplications.

So, consider a product \( A_1 \ldots A_k \) of matrices where each \( A_i \) has size \( r_i \times c_i \). Assume that \( c_i = r_{i+1} \) for \( i = 1 \ldots n - 1 \), so that the product of any two consecutive matrices is well-defined. We call an order of evaluation (a particular parenthesizing) of the product \( A_1 \ldots A_k \) optimal if it uses the minimum number of scalar multiplications.

[a] Design a dynamic programming algorithm to compute the number of scalar multiplications in an optimal order of evaluation for \( A_1 \ldots A_k \). Justify its correctness. (Hint: Think about the final matrix multiplication that happens in the ordering.)

[b] Determine the time and space complexity of your algorithm.

[c] Describe how you would modify your algorithm to report the (or an) optimal order. What is the complexity of this algorithm.

Due: noon, 15 March