CS180 Spring 2013

Homework 5

The following homework is due Thursday, May 9 at the beginning of lecture.

When submitting your homework, please include your name at the top of each page. If you submit multiple pages, please staple them together. We also ask you to indicate which name is your last name on the first page, such as underlining it.

Please provide the proof why your proposed algorithm works and time complexity analysis for all solutions

1. You have to place n files on a tape. A file F_i has a length l_i and a frequency of access p_i ($p_i > 0$, $\sum_i p_i = 1$). The time to access a file given some order of placement on a tape is proportional to the sum of the lengths of all the files preceding it in the order. Find the order that will minimize average access time.

2. We are given two strings: string S of length n, and string T of length m. Describe an efficient algorithm to produce their *longest common subsequence*: the longest sequence of characters that appear left-to-right (but not necessarily in a contiguous block) in both strings.
   Example:
   $S = abcdgh$
   $T = aedfh$
   The longest common subsequence of S and T is adh.

3. Suppose you are a consultant for the networking company CluNet, and they have the following problem. The network that theyre currently working on is modeled by a connected graph $G = (V, E)$ with n nodes. Each edge e is a fiber-optic cable that is owned by one of two companies— creatively named X and Y—and leased to CluNet.

   Their plan is to choose a spanning tree T of G and upgrade the links corresponding to the edges of T. Their business relations people have already concluded an agreement with companies X and Y stipulating a number k so that in the tree T that is chosen, k of the edges will be owned by X and $n - k - 1$ of the edges will be owned by Y.

   CluNet management now faces the following problem. It is not at all clear to them whether there even exists a spanning tree T meeting these conditions, or how to find one if it exists. So this is the problem they put to you: Give a polynomial-time algorithm that takes G, with each edge labeled X or Y, and either (i) returns a spanning tree with exactly k edges labeled X, or (ii) reports correctly that no such tree exists.

4. We are given a sequence $(A_1, A_2, \dots A_n)$ of n matrices to be multiplied, and we wish to compute the product

   $$A_1 A_2 \dots A_n.$$

   We can evaluate the expression using standard algorithm for multiplying pairs of matrices as a subroutine once we have parenthesized it to resolve all ambiguities in how the matrices are multiplied together. A product of matrices is *fully parenthesized* if it is either a single matrix or the product of two fully parenthesized matrix products, surrounded by parentheses. Matrix multiplication is associative and so all parenthesizations yield the same product. For example, sequence (A_1, A_2, A_3, A_4) can be parenthesized in the following ways-

$$(A_1(A_2(A_3A_4)))$$

$$(A_1((A_2A_3)A_4))$$

$$((A_1A_2)(A_3A_4))$$

$$((A_1(A_2A_3))A_4)$$

$$(((A_1A_2)A_3)A_4)$$

The way we parenthesize a chain can have a dramatic impact on the cost of evaluating the product.

Problem Given a sequence $(A_1, A_2, \ldots A_n)$ of n matrices, where $\forall i \in \{1, 2 \ldots n\}$, matrix A_i has dimensions $p_{i-1} \times p_i$, fully parenthesize the product $(A_1A_2 \ldots A_n)$ in a way that minimizes the number of scalar multiplication when evaluating the matrix product.