Fall 18: CSci 5421—Advanced Algorithms and Data Structures

Out 9/19  Homework 2  Due 10/3

Please do all problems; we will grade a subset of the assigned problems (same subset for everyone). Please follow all of the instructions given in the handout for Homework 1.

1. (8+7 points)
   (a) Use the bottom-up (i.e., iterative) algorithm Matrix-Chain-Order(p) seen in class to determine the minimum number of multiplications needed to compute the product of a sequence of six matrices, whose dimensions are \( p = (p_0, p_1, \ldots, p_6) = (30, 1, 40, 10, 25, 50, 5) \). You must show your work, i.e., the filled-in lookup table, the optimal parenthesization, and its cost.
   (b) Ex. 15.2-5, p. 378.

2. (12 points) Give a top-down, memoized version of the algorithm LCS-Length(X, Y) to compute, in \( \Theta(mn) \) time, the length of a longest common subsequence of strings \( X \) and \( Y \), where \( m = |X| \) and \( n = |Y| \). (You do not have to retrieve the LCS itself; just compute its length.) Give a careful analysis of the running time.

3. (10 points) Ex. 25.2-4, p. 699. Justify your answer carefully.

4. (12 points) This problem explores an improvement in the \( \Theta(n^3) \) running time of the algorithm Optimal-BST (Sec. 15.5). It can be shown that the optimal root, \( \text{root}_{ij} \), satisfies \( \text{root}_{ij} \leq \text{root}_{i,j-1} \leq \text{root}_{i+1,j} \), \( 1 \leq i < j \leq n \). (You may assume and use this result without proof.) Using this result rewrite the algorithm Optimal-BST and prove carefully that it runs in time \( \Theta(n^2) \).

5. (15 points)
   Consider the following multiplication table defined on an alphabet \( \Sigma = \{a, b\} \).

   |   | a | b |
   --|--|----|
   a | b | a |
   b | b | b |

   The rows correspond to the left operand and the columns to the right operand; thus, \( aa = b \), \( ab = a \) etc.

   Design a bottom-up dynamic programming algorithm which takes a string \( X = x_1x_2 \cdots x_n \), where each \( x_i \in \Sigma \), and outputs “true” if there is a parenthesization of \( X \) for which the expression evaluates to \( a \) (under the above multiplication table), and “false” otherwise. (You do not have to compute the parenthesization itself if the output is “true”.) The target time bound is \( \Theta(n^3) \).

   For instance, if \( X = abab \), then your algorithm should return “true” since \( (a(b(ab))) = a \). If \( X = baab \), then it should return “false”, since no parenthesization of \( X \) evaluates to \( a \). (You should verify this.)
Your answer should include (i) a brief description of the main ideas, including the dynamic programming recurrence and justification for it, (ii) pseudocode, and (iii) an analysis of the running time.

Note: This is a decision problem (where the answer is “true” or “false”), not an optimization problem. Such problems can also be solved sometimes via dynamic programming.

Hint: Let \( a_{ij} \) (resp. \( b_{ij} \)) be “true” if there is a parenthesization of \( x_i x_{i+1} \ldots x_j \) which evaluates to \( a \) (resp. \( b \)) and “false” otherwise.

6. (15 points) Let \( a_1, \ldots, a_n \) be a sequence of positive integers. A labeled tree for this sequence is a binary tree \( T \) of \( n \) leaves named \( v_1, \ldots, v_n \), from left to right. We label \( v_i \) by \( a_i \), for all \( i, 1 \leq i \leq n \). Let \( D_i \) be the length of the path from \( v_i \) to the root of \( T \). (The length of the path is the number of edges on it.) The cost of \( T \) is given by

\[
\text{cost}(T) = \sum_{i=1}^{n} a_i D_i.
\]

Give an efficient, bottom-up dynamic programming algorithm to compute a labeled tree of minimum cost for \( a_1, \ldots, a_n \) in \( \Theta(n^3) \) time. You must compute the cost of the optimal tree and also show how to compute the optimal tree itself.

Your answer should include (i) a brief discussion of the main ideas, including the dynamic programming and justification for it, (ii) pseudocode, and (iii) an analysis of the running time.

Hint: Let \( c(i, j) \) denote the cost of an optimal tree on \( a_i, \ldots, a_j \).