CSPP 55001 Algorithms — Autumn 2009

Homework 3 (assigned October 14, due October 21)

Reading: CLRS chapter 15, sections 15.1–15.4. Reading for next week’s lecture: chapters 9, 11, and 12.

Written assignment: Solve the following "Do" exercises and assigned problems. **Only solutions to the assigned problems should be turned in.**

Note: You are responsible for the material covered in both "Do" exercises and assigned problems.

Note: If you work with others, indicate their names at the top of your homework paper. Everyone must submit their own independently written solutions.

"Do" Exercises (not to be handed in):

1. Exercise 15.4-2 on page 396.
2. Problem 15-5 (**edit-distance**), parts a and b, on pages 406–408.
   2nd Edition: A palindrome is a nonempty string over some alphabet that reads the same forward and backward. Examples of palindromes are all strings of length one, civic, racecar, and aibohphobia (fear of palindromes).
   Give an efficient algorithm to find the longest palindrome that is a subsequence of a given input string. For example, given the input character, your algorithm should return carac. Describe your algorithm in pseudocode. What is the running time of your algorithm?
4. Give an $O(nk)$ dynamic programming algorithm for the following task:
   **Input:** A list of $n$ positive integers $a_1, a_2, \ldots, a_n$; a positive integer $k$.
   **Question:** Does there exist a subset of the $a_i$’s that adds up to $k$?
   Describe your algorithm in pseudocode.
5. Duru and Denis are considering opening a series of restaurants along 55th Street. The $n$ possible locations are along a straight line, and the distances of these locations from the start of 55th Street are, in miles and in increasing order: $m_1, m_2, \ldots, m_n$. The constraints are as follows:
   - At each location, the TAs may open at most one restaurant. The expected profit from opening a restaurant at location $i$ is $p_i$, where $p_i > 0$ and $i = 1, 2, \ldots, n$.
   - Any two restaurants should be at least $k$ miles apart, where $k$ is a positive integer.
   Give a dynamic programming algorithm that determines the locations to open restaurants, subject to the above constraints, which maximizes the total expected profit. Describe your algorithm in pseudocode. Make your algorithm as efficient as possible.

Problems (to be handed in):

1. You are given $n$ types of coin denominations of values $d[1] < d[2] < \ldots < d[n]$ (all integers). You have an unlimited supply of coins of each denomination. Assume that $d[1] = 1$, so you can always make change for any amount of money $A$. Give a dynamic programming algorithm which makes change for an amount of money $A$ with as few coins as possible. Describe your algorithm in pseudocode. Analyze its running time. Your algorithm should be as efficient as possible. Maximum
points will be given for the most efficient correct algorithms. (15 points)

2. A contiguous subsequence of a sequence $S$ is a subsequence made up of consecutive elements of $S$. For example, if $S$ is $5, 15, -30, 10, -5, 40, 10$, then $15, -30, 10$ is a contiguous subsequence but $5, 15, 40$ is not.

Write a dynamic programming algorithm for the following task:

   Input: A list of $n$ real numbers $a_1, a_2, \ldots, a_n$.
   Output: A contiguous subsequence $a_i, \ldots, a_j$ for which the sum of the elements in the subsequence is maximized. (A subsequence of length 0 has sum zero.)

In the above example, a contiguous subsequence of maximum sum is $10, -5, 40, 10$, with a sum of 55. Describe your algorithm in simple pseudocode. Analyze its running time. Maximum points will be given for correct $O(n)$-time algorithms. Partial credit will be given for correct but less efficient algorithms. (20 points)

3. Given a sequence of $n$ real numbers $a_1, a_2, \ldots, a_n$, give a dynamic programming algorithm that finds a subsequence (not necessarily contiguous) of maximum length in which the values in the subsequence form a strictly increasing sequence. Describe your algorithm in pseudocode and analyze its running time. Maximum points will be given for correct $O(n \log n)$-time algorithms. Partial credit will be given for correct but less efficient algorithms. (20 points)

4. We are given a checkerboard with 4 rows and $n$ columns, and a set of $2n$ pebbles. Each pebble can be placed on exactly one square of the checkerboard. We define a placement to be a positioning of some or all of the pebbles on the board, such that no two pebbles are placed on horizontally or vertically adjacent squares. Diagonal adjacencies are permitted. On each square of the checkerboard is written an integer. The value of a placement is the sum of the integers in the squares that are covered by the pebbles of that placement.

   1. Determine the number of legal patterns that can occur in any column and describe these patterns. (5 points)
   
   We say that two patterns are compatible if they can be placed on adjacent columns to form a legal placement. Let us consider subproblems consisting of the first $k$ columns $1 \leq k \leq n$. Each subproblem can be assigned a type, which is the pattern occurring in the last column.

   2. Using the notions of compatibility and type, give an $O(n)$-time dynamic programming algorithm for computing a placement of maximum value. (15 points)

\begin{flushright}
Gerry Brady
Thursday October 15 13:21:10 CDT 2009
\end{flushright}