Analysis of Algorithms: Assignment 9

Due date: December 2 (Thursday)

If you submit the assignment by 12:30pm on November 30, then (1) you will earn 2 bonus points toward your grade for this assignment (not toward the final grade for the course) and (2) it will be graded by December 2.

Problem 1 (3 points)
Determine a longest common subsequence of $\langle 0,1,1,0,1,1 \rangle$ and $\langle 1,0,1,0,0,1,0,1 \rangle$. Using Figure 16.3 in the book as a model, draw the table constructed by the LCS-Length algorithm for these two sequences (you do *not* need to show arrows in your table).

Problem 2 (4 points)
A sequence of numbers is called *increasing* if its elements are in sorted order. For example, $\langle 1,2,2,3 \rangle$ is an increasing sequence.

Write an algorithm Increasing-Length(A, n) that determines the *length* of a longest increasing subsequence of an array $A[n]$; your algorithm does *not* have to find the longest subsequence itself. For example, if the input array is $\langle 1,2,1,2,3 \rangle$, then its longest increasing subsequence has 4 element: $\langle 1,2,2,3 \rangle$; thus, the algorithm must return 4.

Note that an efficient version of Increasing-Length is based on dynamic programming. If you write an algorithm with exponential running time, you will get only a partial credit.

Problem 3 (3 points)
Suppose that you drive along some road, and you need to reach its end. Initially, you have a full tank, which holds enough gas to cover a certain distance d.

The road has n gas stations, where you can refill your tank. The distances between gas stations are represented by an array $A[1..n]$, where $A[1]$ is the distance from the start to the first gas station, $A[2]$ is the distance from the first to the second station, $A[3]$ is that from the second to the third station, and so on. The last gas station is located exactly at the end of the road. You wish to make as few stops as possible along the way.

Give an algorithm Choose-Stops(d, A, n) that identifies all places where you have to refuel, and returns the set of selected gas stations. You may assume that, for each i, $A[i] \leq d$.

Problem 4 (bonus)
This problem is optional; it allows you to get 2 bonus points toward your final grade.

Suppose that the weights of all items in the 0-1 Knapsack Problem are integers, and the weight limit W is also an integer. Design an algorithm that finds a *globally optimal* solution, and give its time complexity in terms of the number of items n and weight limit W.