# Analysis of Algorithms: Assignment 8

**Due date: April 8 (Thursday)**

**Problem 1** (3 points)
Using Figure 16.2 in the textbook as a model, draw the recursion tree for the MERGE-SORT procedure on a sixteen-element array. Explain why dynamic programming is ineffective for speeding up MERGE-SORT.

**Problem 2** (3 points)
Determine a longest common subsequence of $\langle 1, 0, 0, 1, 0, 1 \rangle$ and $\langle 0, 1, 0, 1, 1, 0, 1, 1 \rangle$. 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 3** (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 a recursive algorithm with exponential running time, then you will get only a partial credit.