Algorithms
Computer Science 140 & Mathematics 168
Instructor: B. Thom
Spring 2004
Homework 5a
Due on 2/19/04 (beginning of class)

1. **[5 Points] Reinforcing LCS** Show what values would be stored in the Dynamic Programming Table for the LCS problem if the inputs were

$$X_m = < 1, 0, 0, 1, 0, 1, 0, 1 >$$
$$Y_n = < 0, 1, 0, 1, 1, 0, 1, 1, 0 >$$

Also show a corresponding optimal subsequence.

2. **[20 Points] More Fun at the Firm of Weil, Proffet, and Howe!**

You have been hired as Vice President for Algorithm Design at the Wall Street firm of Weil, Proffet, and Howe. Consider an array A of values of a share for a particular company over the last n days. Assume that A is indexed from 1 to n and $A[1]$ corresponds to the value of a share on day 1, $A[2]$ corresponds to the value of a share on the next day, and so forth.

A *monotonically increasing subsequence* of the array is a sequence of array elements $A[i_1], A[i_2], \ldots, A[i_k]$ such that $A[i_1] < A[i_2] < \ldots < A[i_k]$ and such that $i_1 < i_2 < \ldots < i_k$. For example, a monotonically increasing subsequence of the array $5, 12, 7, 9, 6$ is $(5, 7, 9)$.

Assume that we are given an array A of length n. Your goal is to find the length of the longest monotonically increasing subsequence *which includes the element* $A[n]$. That is, $A[n]$ must be the last element in the subsequence. For example, for the array $5, 2, 7, 3, 6$, we have $n = 5$ and the length of the longest monotonically increasing subsequence which includes $A[n]$ is 3. (The subsequence $(2, 3, 6)$.)

(a) The previous Vice President for Algorithm Design, Dr. Juss Dooit, proposed a brute-force algorithm for finding the length of the longest monotonically increasing subsequence which includes $A[n]$. This algorithm simply enumerated all possible subsequences, one-by-one, tested to see if each enumerated subsequence was monotonically increasing, and if so, compared the length of that subsequence to the longest monotonically increasing subsequence seen so far. Consider the **Crayfish 2000** computer that can enumerate and test 10^{12} subsequences per second. How long would it take to execute Dr. Dooit's algorithm on an array of length 100 on the Crayfish? Why was Dr. Dooit fired? (FYI: a back of the envelop calculation suffices here.)

(b) Professor Rhea Cursive wants to find a recursive algorithm for solving this problem. The algorithm `mono`(i) will return the *length* of the longest monotonically increasing subsequence in $A[1], \ldots, A[i]$ which includes $A[i]$. Clearly, if she can write such a recursive algorithm, we can run it with $i = n$ to solve our problem. (Notice that `mono` doesn't take the array A as input. We're assuming that A is implicit. That is, it's a global variable that `mono` can access.) Professor Cursive has a vague notion that `mono`(i) can be computed as follows: Look at each j from 1 to $i-1$ and check to see if $A[j] < A[i]$. If so, make a recursive call to compute `mono`(j). Now, do something with these `mono`(j) values to compute the desired `mono`(i). Write pseudo-code for `mono`(i). (It should be comprehensible, but need not be in any particular programming language.)

(c) Explain why the worst-case running time of the algorithm is at least exponential. Use a lower bound argument similar to the ones we saw in class.

(d) Now convert this algorithm into a dynamic programming algorithm as follows: Let M be an array indexed from 1 to n. $M[i]$ will store the value of `mono`(i). However, we will compute $M[1]$ first, then $M[2]$, and so forth all the way up to $M[n]$. Carefully describe the procedure for computing $M[i]$. Then, derive the running time of the algorithm to fill in the entire array M.

(e) Now, assume that the constant in front of your algorithm's asymptotic run-time is bad, like 50. Assume also that your dynamic programming algorithm is implemented and executed on a Granny Smith Personal Computer which is only able to execute 10^4 instructions per second. How long does it take to solve the client's problem (again, for an input of length 100) using this algorithm on the Granny Smith?

(f) Show the table constructed by your dynamic programming algorithm for the array $4, 5, 2, 7, 3, 8$.

(g) Explain how your algorithm can be modified slightly to allow you to find the actual longest monotonically increasing subsequence (it is no longer required that the last element in this answer includes $A[n]$). To clarify: by "actual" we mean that the algorithm actually reports the corresponding longest monotonic subsequence. What does this modification do to the running time of your algorithm?