comp363 Handout #14
Design and Analysis of Computer Algorithms
Michael Goldwasser
Loyola University Chicago Thursday, 27 April 2003

Homework #5: Dynamic Programming
Due Date: Tuesday, 8 April 2003

Guidelines

Please make sure you adhere to the policies on collaboration and academic honesty as outlined in Handout #1.

Reading

Review Ch. 15 of CLRS.

Problems

Problem A (40 points) "You may discuss ideas with other students."

Given a sequence of numbers $X = \langle x_1, x_2, \ldots, x_n \rangle$, a *monotonically increasing subsequence* is a sequence $X' = \langle x_{a_1}, x_{a_2}, \ldots, x_{a_k} \rangle$ such that $1 \leq a_1 < a_2 < \ldots < a_k \leq n$ (i.e., the sequence is truly a subsequence), and that $x_{a_1} \leq x_{a_2} \leq \ldots \leq x_{a_k}$ (i.e., the sequence is monotonically increasing). Our goal is to find the *longest* monotonically increasing sequence. Actually, to make things easier, let's assume that we are only interested in knowing the *length* of the longest such sequence (although all of these techniques can be extended to build the actual sequence).

For example if $X = \langle 5, 1, 4, 2, 3, 8, 6, 7 \rangle$, then the longest monotonically increasing subsequence is $\langle 1, 2, 3, 6, 7 \rangle$, with length 5.

To design a dynamic programming algorithm we will consider the following subproblem for each $k \in \{1, \ldots, n\}$, namely, what is the longest monotonically increasing subsequence which ends with x_k. If we let $L(k)$ equal the *length* of the longest such sequence which ends with x_k, we claim the following recursive formula exists,

$$L(k) = 1 + \max_{\substack{j \in \{1,\ldots,k-1\} \\ \text{with } x_j \leq x_k}} L(j) \tag{1}$$

a. Prove Equation (1).
(Make sure you prove both the "$\geq$" and the "$\leq$" implied by equality.)

b. Show that the *length* of the longest monotonically increasing sequence can be computed using dynamic programming based directly on Equation (1). What is the running time of the algorithm?

Problem B (60 points) "You may discuss ideas with other students."

Coin Changing

Consider the problem of making change for n cents using the fewest number of coins. Assume that each coin's value is an integer.

a. Describe a greedy algorithm to make change consisting of quarters, dimes, nickels, and pennies. Prove that your algorithm yields an optimal solution.

b. Suppose that the available coints are in the denominations that are powers of c, e.e., the denominations are $c^0, c^1, \ldots, c^k$ for some integers $c > 1$ and $k \geq 1$. Show that the greey algorithm always yields an optimal solution.

c. Give a set of coin denominations and a value for n, such that the greedy algorithm does not yield an optimal solution. Your set should include a penny so that there is a solution for every value of n.

d. Give an $O(nk)$-time dynamic programming algorithm that makes change for any set of k different coin denominations, assuming that one of the coins is a penny.