Algorithms
Computer Science 140 & Mathematics 168
Instructor: B. Thom
Fall 2003
Homework 3b
Due on 9/23/03 (Tuesday in class)

1. **[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. Our 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?

   (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.

Let $T(n)$ denote the worst-case running time of the algorithm as a function of n. Then $T(n) = T(n-1) + T(n-2) + \cdots + T(1)$. Thus, $T(n) \geq T(n-1) + T(n-2)$. Moreover, $T(n-1) \geq T(n-2)$ since $T(n-1) = T(n-2) + T(n-3) + \cdots + T(1)$. Thus, $T(n) \geq 2T(n-2) \geq 2^2 T(n-4) \geq \cdots \geq 2^{n/2}$, which is exponential in n.

(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. The running time should be polynomial in n. That is $\Theta(n^c)$ for some fixed positive integer c (like 2 or 3).

(e) Now, assume that the constant in front of your polynomial is something 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 find the length of the longest monotonically increasing subsequence using this algorithm on the Granny Smith?

(f) Show the table constructed by your dynamic programming algorithm for the array $5, 2, 7, 3, 6$.

(g) Explain how your algorithm can be modified slightly to allow you to find the actual longest monotonically increasing subsequence.What does this modification do to the running time of your algorithm?

2. **[30 Points] Ski Optimization!**

Your job at Snapple is pleasant but in the winter you've decided to become a ski bum. You've hooked up with the Triple Black Diamond Ski Resort. They'll let you ski all winter for free in exchange for helping their ski rental shop with an algorithm to assign skis to skiers.

Ideally, each skier should obtain a pair of skis whose height matches his or her own height exactly. Unfortunately, this is generally not possible. We define the *disparity* between a skier and his or her skis to be the absolute value of the difference between the height of the skier and the pair of skis. Our objective is to find an assignment of skis to skiers that minimizes the sum of the disparities. We'll solve some interesting subproblems along the way to finding an efficient algorithm for the ski assignment problem.

(a) First, let's assume that there are n skiers and n skis. The previous computer scientist at the ski resort, Professor R. U. Kitting, proposed the following algorithm for this case. Consider all possible assignments of the n skis to the n skiers. For each one, compute the sum of the disparities. Finally, select the assignment that minimizes the sum of the disparities. Explain why Professor Kitting was fired.

How much time would this algorithm take on a computer thatp erforms 1 billion operations per second if there were 50 skiers and 50 pairs of skis?

(b) The next computer scientist at the resort, Dr. G. I. M. Fast, spent most of her time on the slopes. She did however make an interesting discovery one day on the chair lift. She observed that if we have a short person and a tall person, it would never be better to give the shorter person a taller pair of skis than were given to the tall person. Show that this is always true. (If you like, you may split this into cases and consider each case separately.)

(c) After Dr. Fast's brief stint at the ski resort, you were hired. Your first job is to find a greedy algorithm that minimizes the sum of the disparities assuming that there are n people and n pairs of skis. (*Hint:* First sort the people by increasing height and sort the skis by increasing height.) Argue carefully that your algorithm does indeed find an optimal solution. In other words, show that no better solution is possible. What is the time complexity of your algorithm?

(d) Now, your task is to design an efficient dynamic programming algorithm for the more general case that there are m skiers and n pairs of skis and $m \leq n$. Again, start by sorting the skiers and skis by increasing height. Let h_i denote the height of the i^{th} skier in sorted order and let s_j denote the height of the j^{th} pair of skis in sorted order. Let $A[i, j]$ be the optimal cost (sum of absolute differences of heights) for matching the first i skiers with skis from the set $\{1, 2, \ldots, j\}$. The solution we seek is then simply $A[m, n]$. Fill in the blanks below to define A recursively:

$$A[i,j] = \begin{cases} \underline{\hspace{3cm}} & \text{if } i = 0 \\ \underline{\hspace{3cm}} & \text{if } i = j \\ \underline{\hspace{3cm}} & \text{if } i < j \end{cases}$$

Now give pseudocode (with explanatory comments) or a concise and complete English description of a dynamic programming algorithm that solves the optimal assignment problem of skis to skiers using a table for $A[i, j]$.

(e) What is the running time of your program? Explain.

(f) Describe how your algorithm can be modified to allow you to find an actual optimal assignment (rather than just the cost) of skis to skiers. How does this affect the running time of your algorithm?

x

(g) Illustrate your algorithm by explicitly filling out the $A[i, j]$ table for the following sample data:

- Ski heights: $1, 2, 5, 7, 13, 21$.
- Skier heights: $3, 4, 7, 11, 18$.