

**PROBLEM SET 6**  
**Due: Friday, November 8**

*Important: On Friday, November 8, you will receive a take-home exam. In order to give you maximal time to work on your exam, I require Problem Set 6 to be turned in no later than 6pm on Friday, November 8. At that time, solutions to Problem Set 6 will be made available.*

**Reading:** CLR Chapter 16 (Dynamic Programming); CLR Chapter 17, Sections 17.1--17.3 (Greedy Algorithms)

**Suggested Problems:** 16.1-1; 16.3-1; 16.3-6; 16-2; 16-3; 16-4; 17.2-2, 17.2-3, 17.2-4, 17.2-5, 17.2-6; 17.3-1, 17.3-8; 17-1

**Problem 1 [25]** Consider the following recursive definition of the Fibonacci function:

```
Fib(n)
  if n < 2 then
    return n
  else
    return Fib(n-1) + Fib(n-2)
```

**a [5]** Draw a function call tree for  $\text{Fib}(6)$  and determine the number of times that  $\text{Fib}(2)$  is called. (In this case, a function call tree is a binary tree in which each node is labelled by  $\text{Fib}(i)$  and has left and right subtrees that are function call trees for  $\text{Fib}(i-1)$  and  $\text{Fib}(i-2)$ , respectively.)

**b [10]** The above recursive definition of  $\text{Fib}$  has a running time that is exponential in  $n$ . Using memoization (with an auxiliary array), develop a  $O(n)$  recursive definition  $\text{Fib-Memo}$  of the Fibonacci function. Express your solution in pseudocode.

**c [10]** Transform the top-down recursive strategy from part b into a bottom-up iterative strategy expressed in terms of a loop rather than a recursion. Express your solution in pseudocode.

**Problem 2 [10]** CLR 16.3-1 (p. 319)

**Problem 3 [25]** Solve CLR 16.3-5 (p. 319) using the two strategies described below. Assume that the input is stored in an array  $A[1..n]$ . For each strategy:

- describe your algorithm (in English is fine)
- briefly argue why it is correct
- briefly argue why it takes  $O(n^2)$  time.

**a[10]** Develop a solution that uses the  $(mn)$  LCS algorithm as a black box as part of the solution. Note: you can express a solution of this form in just a few lines!

**b[15]** Develop a solution that does *not* use the LCS algorithm as a black box, but instead uses an auxiliary array  $M[1..n]$ , where each  $M[i]$  stores the longest monotonically increasing sequence in  $A[i..n]$  that begins with  $A[i]$ .

**Problem 4[25]** CLR 16-2 (p. 325)

**Problem 5[15]** CLR 17.2-4 (p. 337)

**Extra Credit Problem [20]** CLR 16.3-6 (p. 319)

*Problem Set Header Page*  
*Please make this the first page of your hardcopy submission.*

**CS231 Problem Set 6**  
**Due Friday, November 8, 1996**

Name:

Date & Time Submitted (*only if late*):

Collaborators (*anyone you collaborated with in the process of doing the problem set*):

*In the **Time** column, please estimate the time you spent on the parts of this problem set. Please try to be as accurate as possible; this information will help me to design future problem sets. I will fill out the **Score** column when grading your problem set.*

<b>Part</b>	<b>Time</b>	<b>Score</b>
General Reading		
Problem 1 [25]		
Problem 2 [10]		
Problem 3 [25]		
Problem 4 [15]		
Problem 5 [25]		
Extra Credit [20]		
<b>Total</b>		