Prof. Sergio A. Alvarez
21 Campanella Way, room 569
Computer Science Department
Boston College
Chestnut Hill, MA 02467 USA

http://www.cs.bc.edu/∼alvarez/
alvarez@cs.bc.edu
voice: (617) 552-4333
fax: (617) 552-6790

CS383, Algorithms
Spring 2009
HW8

1. Consider the dynamic programming algorithm for the computation of the Fibonacci populations $F(n)$ that we discussed in class, shown in Algorithm 1.

   Algorithm 1: Fibonacci numbers via dynamic programming.
   Input: Integer $n \geq 0$.
   Output: The Fibonacci rabbit population $F(n)$ in the n-th generation.
   DPFIB(n)
   (1) allocate table(0...n)
   (2) table(0) = 0, table(1) = 1
   (3) **foreach** $i = 2...n$
   (4) table(i) = table($i-1$) + table($i-2$)
   (5) **return** table(n)

   (a) Determine the simplest possible big Θ expression for the running time of Algorithm 1 in terms of n, assuming that addition of two d-digit operands takes time $\Theta(d)$. Explain in detail. You may assume that the Fibonacci numbers $F(n)$ grow at the asymptotic rate $\Theta(b^n)$ for some $b > 1$.

   (b) Based on your analysis in 1a, is the dynamic programming solution in Algorithm 1 actually faster asymptotically than straightforward recursive implementation of the Fibonacci recurrence relation, or does the increasing time needed for the additions of increasingly large operands swamp the runtime to the point that the two algorithms are asymptotically equivalent? Explain.

2. Consider driving from one street corner to another in a city covered by a grid of perpendicular streets. Any *monotonic* path from the source location to the destination is allowed; that is, each corner along the path should be closer to the destination than the preceding one. Paths should proceed along existing streets, without cutting across a block diagonally. You will design a dynamic programming algorithm that counts how many different such paths exist.

   (a) Let $P(n, m)$ denote the total number of monotonic paths between two street corners that are separated by n North-South "avenues" and m East-West "streets". For example, $P(1, 1)$ is the number of monotonic paths between two street corners located at diagonally opposite points of a city block; therefore, $P(1, 1)$ equals 2. Derive a recurrence relation for $P(n, m)$ by considering all street corners that can occur on the path one block away from the destination. Explain your thinking.

(b) Based on the recurrence relation for $P(n, m)$ that you derived above, sketch the domain of dependence of a given point $(n0, m0)$ in the (n, m) plane for that recurrence relation. Explain.

(c) Describe suitable boundary cases for the recurrence relation for $P(n, m)$. Clearly state what the value of $P(n, m)$ is for each boundary point (n, m). Explain, and indicate the locations of the boundary points in your sketch above.

(d) Using your work in the preceding parts of this task, write detailed pseudocode for a dynamic programming algorithm that computes the number of different monotonic paths between two street corners in a city as described above.

(e) Determine the big Θ running time of your dynamic programming algorithm for computing $P(n, m)$. Explain your analysis in detail.

3. In this task you will design an efficient dynamic programming algorithm for the maximum consecutive subsequence sum task (MSS) from HW6. Recall that the input is an array of n real numbers, each of which can be positive, zero, or negative. The output is the maximum sum that can be obtained by adding some set of consecutive elements of the input array. Consecutive here means that the elements occupy some segment $i, i+1, ...i+k$ of consecutive positions of the array, not that the elements' values are related in any way. *In the present task, only the maximum sum needs to be returned, not the index range of the original array over which that sum is attained.*

   (a) Define a hierarchy of subproblems $P(1)...P(n)$ that will allow the solution of the MSS task for the original input $a[1...n]$ to be computed. *It is of crucial importance to define the subproblems in the right way in order to allow the solution of the target problem to be obtained readily in terms of the solutions of smaller subproblems in the hierarchy.* Provide a concise definition of $P(i)$ in terms of i.

   (b) Derive a recurrence relation that expresses the solution of the MSS task for $a[1...n]$ in terms of the solutions of smaller subproblems that belong to the subproblem hierarchy from 3a. Specify suitable boundary conditions for the recurrence. Explain.

   (c) Write detailed pseudocode for an $O(n)$ dynamic programming solution to the MSS task, using your work from 3a and 3b.

   (d) What is the running time of your algorithm from 3c in terms of n? Provide the simplest possible big Θ expression. Explain your analysis.