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
HW5

1. Manually perform depth-first search (DFS) on the graph shown in Fig. 3.2 of the textbook. Explore vertices in alphabetical order whenever there is a choice between two or more vertices. Draw the resulting DFS tree. Label each vertex with its pre and post numbers.

2. Consider the specific form of the breadth first search algorithm shown in Algorithm 1.

   Algorithm 1: Breadth First Search
   Input: Graph G, start vertex s of G.
   Output: For each vertex v of G that is reachable from s, the number of edges along the shortest path from s to v in G.

   ```
   BFS(G)
   (1)    foreach vertex v of G
   (2)       dist(v) = ∞
   (3)    dist(s) = 0
   (4)    queue Q = {s}
   (5)    while Q not empty
   (6)       v = removeFront(Q)
   (7)       foreach vertex w to which there is an edge from v
   (8)          if dist(w) = ∞
   (9)             insertRear(w, Q)
   (10)            dist(w) = dist(v) + 1
   ```

   (a) Provide a detailed running time analysis of Algorithm 1, assuming an incidence matrix representation for the graph G. Recall that the incidence matrix for a graph G with n vertices is an $n \times n$ matrix M (two–dimensional table with n rows and n columns) such that $M_{i,j} = 1$ whenever there is an edge in G from vertex i to vertex j, and $M_{i,j} = 0$ otherwise. You may assume that the queue insertion and removal operations run in time $O(1)$. Express the result of your running time analysis in asymptotic notation, as a function of the number of vertices of the graph and the number of edges.

   (b) Repeat the running time analysis of Algorithm 1, now assuming an adjacency list representation for the graph G. Recall that the adjacency list of a graph G with n vertices is an array L of length n, where the i-th entry $L[i]$ is a linked list whose elements are the vertices j such that there is an edge in G from vertex i to vertex j. Express the result of your running time analysis in terms of the number of vertices of the graph and the number of edges. Give the tightest possible bound in asymptotic notation.

3. (a) Design an algorithm that takes a directed graph as the input and determines whether the graph is acyclic or not. Follow the description in section 3.3.2 of the book. Use pre and post numbers. Give detailed pseudocode.

   (b) Carefully analyze the running time of your acyclicity detecting algorithm from the preceding subtask. Does the asymptotic running time depend on what graph representation is used (incidence matrix or adjacency list)? Explain.