Homework 3
Fundamental Algorithms, Fall 2005, Professor Yap

Due: Thu Nov 10, in class.

INSTRUCTIONS:

- Please read questions carefully. When in doubt, please ask.

Question 1

(30 Points)
(a) Prove that if a bigraph has no odd cycles, then it is bipartite.
(b) Prove that if a connected bigraph has an odd cycle, then BFS search from any source vertex will detect a level edge.
(c) Write the pseudo code for bipartite test algorithm outlined in the text. This algorithm is to return YES or NO only. You only need to program the shell routines. NOTE: Please be concise! All pseudo code must be reduced to $O(1)$ time operations or operations on standard data structures such as queues.
(d) Modify the algorithm in (c) so that in case of YES, it returns a Boolean array $B[1..n]$ such that $V_0 = \{i \in V : B[i] = \mathsf{false}\}$ and $V_1 = \{i \in V : B[i] = \mathsf{true}\}$ is a witness to the bipartiteness of G. In the case of NO, it returns an odd cycle.

Question 2

(15 Points)
Explore the relationship between the traversals of binary trees and DFS.
(a) Why are there not two versions of DFS, corresponding to pre- and postorder tree traversal? What about inorder traversal?
(b) Give the analogue of DFS for binary trees. As usual, you must provide place holders for shell routines. Further assume that the DFS returns some values which is processed at the appropriate place. Be brief.

Question 3

(25 Points)
A vertex u is called a **bottleneck** if for every other vertex $v \in V$, either there is a path from v to u, or there is a path from u to v.
(a) Give an algorithm to determine if a DAG has a bottleneck.
(b) Prove the correctness of your algorithm.
(c) Analyze the complexity of your algorithm (it should be no more than $O(n^4)$ but hopefully faster).

Question 4

(10 Points)
Please read the revised notes for computing strong components of a digraph. Give a counter example to our original strong component algorithm (find a small an example as possible – we deduct points for unnecessarily large graphs).