

### Assignment 8

Due: Monday, April 14, 2008 (before class)

1) (10 points) Given is a weighted, directed graph  $G = (V, E)$  and a source vertex  $s \in V$ . Show how to modify the Bellman-Ford single-source shortest-paths algorithm to first determine whether there exists a negative weight cycle reachable from source vertex  $s$  and, if there exists one, output such a negative weight cycle. Note that it is not enough to say that there exists a negative weight cycle, but you should output all the vertices of the cycle as well. The running time of your algorithm should be as Bellman-Ford's.

2) (15 points) Let  $T = (V, E)$  be an  $n$ -vertex, arbitrary tree. The tree is represented as a rooted tree with root  $r$  and edges  $(i, p(i))$ , where  $p(i)$  is the parent of vertex  $i$ ,  $1 \leq i \leq n$ . Note that the orientation of the edges is only for representation purposes only.

The *diameter* of a tree is defined as the longest distance between any two vertices in the tree. Hence,  $\text{diameter} = \max_{u,v \in V} \delta(u, v)$ , where  $\delta(u, v)$  is the unique distance between vertices  $u$  and  $v$ .

- (i) Show that the distance determining the diameter will always be between two leaves.
- (ii) Describe and analyze an  $O(n)$  time algorithm to compute the diameter of tree  $T$ .

3) (10 points) Consider the following algorithm for determining the strongly connected components in a directed graph  $G$  that uses only the original graph  $G$  (and not the reverse graph).

1. Call  $\text{DFS}(G)$  to compute finish times  $f[u]$  for each vertex  $u$ .
2. Call  $\text{DFS}(G)$ , but in the main loop of DFS, consider the vertices in *increasing* order of finish times (as computed in line 1).
3. Output the vertices of each tree in the DFS forest formed in line 2 as a separate strongly connected component.

Is the above algorithm correct? Justify your answer.

4) (15 points) An undirected graph  $G = (V, E)$  is a *bipartite* graph if it has no odd cycle (i.e., an odd cycle is a cycle in  $G$  with an odd number of edges). Note that a graph with no cycles is also bipartite. Give an  $O(|V| + |E|)$  time algorithm using depth-first search to check whether a given graph is bipartite or not. You can assume that the graph is connected.

**Hint:** You may find it useful to maintain a *parity* array on the vertices defined as follows. Assume that DFS is started from a vertex  $r$ , i.e., the depth-first spanning tree is rooted at  $r$ .  $\text{parity}[v]$  is either 0 or 1 depending on whether the number of tree edges in the (unique) path

from the root  $r$  to the vertex  $v$  in the depth-first spanning tree is even or odd. Initially, only the parity of the root is defined, i.e.,  $parity[r] = 0$  and is undefined for all other vertices in the graph. When a node is visited for the first time, its parity will be set to either 0 or 1 accordingly. Use the parity values to test for the existence of an odd cycle during DFS search.