Purdue University
CS 381

S. Hambrusch - C. Guerra
Fall 1999

Assignment 7

Due: Wednesday, November 10, 1999, in class

1) (10 pts.) Consider the strongly connected component algorithm presented in class. The DFS traversal on G^T considered the vertices by decreasing finish times. (Recall that the finish time refers to the order of completion of the recursive calls of the DFS on G.) Give an example of a directed graph consisting of 12 vertices and three strongly connected components for which the processing of vertices by decreasing finish times is crucial. In particular, show by example that when the DFS on G^T does not consider vertices by decreasing finish times, the algorithm fails to detect the three strongly connected components.

2) (10 pts.) Consider the biconnected component algorithm presented in class. Would the algorithm work properly if the test for a bicomponent were changed to $back \geq discoverTime[v]$? If so, explain why; if not, give an example in which it does not work.

3) (15 pts.) The 3-coloring problem is defined as follows: Given are 3 colors (say, R, G, and B) and an undirected graph $G = (V, E)$ in which the degree of every vertex is at most 3. You want to determine whether it is possible to assign to every vertex one of the three colors so that every edge of G is incident to vertices having different color.
(i) Give an example of a graph in which every vertex has degree at most 3 which is not 3-colorable. Explain why the graph is not 3-colorable.
(ii) Develop an $O(2^n n)$ time algorithm to determines whether a graph is 3-colorable. Hint: Once a vertex is colored, there are only two possible colors left for its adjacent vertices. Use an approach based on DFS or BFS to take advantage of this when exploring the graph.

4) (15 pts.) Let T be a Minimum-cost Spanning Tree of a weighted, undirected graph $G = (V, E)$. Let T' be a Minimum Spanning Tree of the new graph G obtained by one of the following changes in the edge weights:

- all edge weights are increased by a constant number c

- all edge weights are decreased by a constant number c
- the weight of a single edge known **not** to be in T is increased by c
- the weight of a single edge known **not** to be in T is decreased by c

For each case describe and analyze an efficient algorithm to generate T' from T. Show that your algorithm is correct.