Purdue University
CS 381

S. Hambrusch - C. Guerra
Fall 1999

Solution Sketches to Assignment 6

1) (Graded by Cenyu Zhang) (i) Consider the undirected graph shown in Figure 7.10 on page 326. Assume DFS starts at vertex 3. Show the graph partitioned into tree and back edges and give for every vertex its depth-first number.

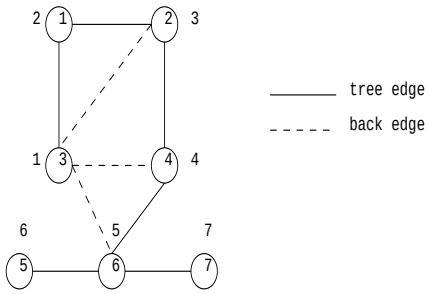

Figure 1: *Resulting tree*

(ii) Consider the directed graph shown in Figure 7.29 on page 377. Assume the vertices are listed in alphabetical order in the adjacency lists and DFS starts at vertex A. If necessary, it restarts on the alphabetically smallest vertex remaining. Show the graph partitioning into tree, back, cross, and forward edges. Give the depth-first number for each vertex.

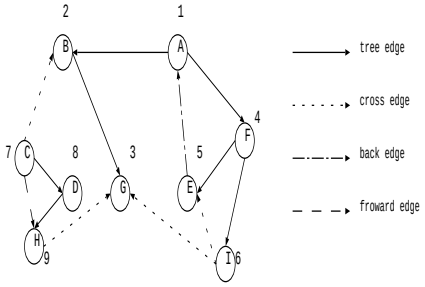

Figure 2: *Resulting tree*

(iii) Give an example of a graph in which DFS backs up from a vertex before all the vertices that can be reached from it via one or more edges are discovered. Explain how this can happen.

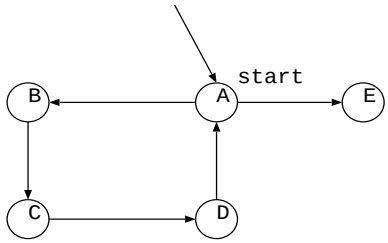

Figure 3: *Example*

In the above given example, we start at node A, then go to B, and then go to C, and then go to D. At D, although we can reach E, we backtrack since A is grey. This happens when there is cycle in the simple graph which introduces a back edge. So when running DFS, we find a grey node and back up, although there are white vertices that could be reached.

2) (Graded by Cenyu Zhang) Let $G = (V, E)$ be a directed, acyclic graph (dag) represented by adjacency lists.
(i) Can a dag have $\Theta(n^2)$ edges?
Yes, a dag can have $\Theta(n^2)$ edges. Consider, for example, a graph with n vertices, $v_1, v_2, \ldots, n_n$. Assume for v_1 has $n-1$ edges to $v_2, \ldots, v_n$, v_2 has $n-2$ edges to $v_3, \ldots, v_n$, and so on. The generated graph is a dag and it has $(n-1) + (n-2) + \ldots + 2 + 1$ edges, which is $\Theta(n^2)$.

(ii) A dag G is a lattice if G contains two vertices s and t such that every vertex in G can be reached from s and every vertex in G can reach vertex t. Describe and analyze an efficient algorithm to determine whether a given dag is a lattice. If it is, s and t should be output.

The following describes an $O(n+m)$ time algorithm to determine whether G is a lattice. We first state a property of acyclic graphs: an acyclic graph contains at least one vertex of in-degree 0 and at least one vertex of out-degree 0. (If this were not the case, there would be a cycle.)

Now, for G to be a lattice, there must exists exactly one vertex of in-degree 0 and exactly one vertex of out-degree 0. The algorithm checks whether this holds. Then, knowing the potential source s and potential sink t, it checks whether s can reach every vertex and every vertex can reach t. Below are the details of the algorithm.
(1) First we go through G's adjacency list and determine all nodes whose in-degrees are zero, and all nodes whose out-degrees are zero. If there is more than one node whose in-degree is zero or whose out-degree is zero, then G is not a lattice. We call the node with in-degree zero s, and the node with out-degree zero t.
(2) Do a DFS on G starting from s. If this traversal can reach all other nodes in G, proceed with the third step. Otherwise G is not a lattice.
(3) Create the transpose of G, that is G^T. Then, do a DFS on G^T starting from t. If this traversal can reach all other nodes in G^T, then we know G is a lattice and s and t are the vertices we need, else G is not a lattice.

Analysis of algorithm: Since each of the steps needs at most $O(n+m)$ time, the total time complexity of this algorithm is $O(n+m)$.

3) (Graded by Cenyu Zhang) Let G be a connected graph, and let s be a vertex in G. Let T_D be the depth-first search tree formed by doing a depth-first search of G starting at s. Let T_B be the breadth-first spanning tree formed by doing a breadth-first search of G starting at s. Is it always true that $height(T_D) \geq height(T_B)$?

Yes, this is always true. When doing the BFS, let vertex s be at level 0. Then, if a vertex v is on level i, there exists a path of length $i-1$ from s to v. Also, there does not exist a shorter path (otherwise v would be on a smaller numbered level). For DFS, a vertex v is placed on a level equal to the shortest path from s to v or on a higher level. Vertex v cannot be placed on a level smaller than the shortest path. Hence, $height(T_D) \geq height(T_B)$. It doesn't matter whether the graph is directed or undirected. (The same argument applies.)