Michael Kaiser
Assignment 6
Comp 510
Fall 2012

22.2-1
Show the d and pi values that result from running breadth-first search on the directed graph of Figure 22.2(a), using vertex 3 as the source.

After running a breadth first search on the graph, the following are the results:

#	pi	d
1	nil	inf
2	4	3
3	nil	0
4	5	2
5	3	1
6	3	1

22.2-2
Show the d and pi values that result from running breadth-first search on the directed graph of Figure 22.3, using vertex u as the source.

After running a breadth first search on the graph, the following are the results:

#	pi	d
u	nil	0
t	u	1
y	u	1
x	u	1
w	t/x	2
s	w	3
r	s	4
v	r	5

22.2-6
There are two types of professional wrestlers: "good guys" and "bad guys." Between any pair of professional wrestlers, there may or may not be a rivalry. Suppose we have n professional wrestlers and we have a list of r pairs of wrestlers for which there are rivalries. Give an O(n + r)-time algorithm that determines whether it is possible to designate some of the wrestlers as good guys and the remainder as bad guys such that each rivalry is between a good guy and a bad guy. If is it possible to perform such a designation, your algorithm should produce it.

We can set up a graph as follows: each vertex represents a wrestler and an edge represents a rivalry. We can perform a BFS (O(n+r) operation) such that if the number of edges between two vertices are odd, they are rivals, and if the number of edges is even between two verticies, then the wrestlers are on the same side. If this holds true (the result is a bipartite graph), then we have found a labeling for each wrestler.

22.3-11

Show that a depth-first search of an undirected graph G can be used to identify the connected components of G, and that the depth-first forest contains as many trees as G has connected components. More precisely, show how to modify depth-first search so that each vertex v is assigned an integer label cc[v] between 1 and k, where k is the number of connected components of G, such that cc[u] = cc[v] if and only if u and v are in the same connected component.

Modified DFS algorithm:

```
DFS(G)
        for each vertex u in V[G]
                u.color = white;
                u.pi = NULL;
                cc[u] = 0;
        time = 0;
        componentNum = 0;
        for each vertex u in V[G]
                if(u.color == WHITE)
                        DFS-VISIT(G, u, componentNum);
                        componentNum++;

DFS-VISIT(G, u, componentNum)
        u.color = GRAY;
        cc[u] = componentNum;
        time++;
        u.d = time;
        for each v in u.adjacency-list
                if(v.color == WHITE)
                        v.pi = u;
                        DFS-VISIT(G, v, componentNum);
        u.color = BLACK
        time++;
        u.f = time;
```

22.4-1
Show the ordering of vertices produced by TOPOLOGICAL-SORT when it is run on the dag of Figure 22.8, under the assumption of Exercise 22.3-2.

x	d	f
m	1	20
n	21	26
o	22	25
p	27	28
q	2	5
r	6	19
s	23	24
t	3	4
u	7	8
v	10	17
w	11	14
x	15	16
y	9	18
z	12	13

The resulting order is:
p-n-o-s-m-r-y-v-x-w-z-u-q-t

22.5-1
How can the number of strongly connected components of a graph change if a new edge is added?

The number of strongly connected components of a graph could be decreased, if separate strongly connected components were connected by the added edge, merging them into one "component".

The number of strongly connected components of a graph could stay the same if the edge is redundant, if for instance the edge B->A were added to 22.9(a).

The number of strongly connected components of a graph could increase by one if the edge satisfies the previously unsatisfied requirement of there being an edge between verticies v and u such that u -> v and v -> u.

22.5-2
Show how the procedure STRONGLY-CONNECTED-COMPONENTS works on the graph of Figure 22.6. Specifically, show the finishing times computed in line 1 and the forest produced in line 3. Assume that the loop of lines 5-7 of DFS considers vertices in alphabetical order and that the adjacency lists are in alphabetical order.

The finishing times as computed by line 1 are as follows:

x	f
q	16
r	20
s	7
t	15
u	19
v	6
w	5
x	12
y	14
z	11

The sorted (by decreasing finishing time) order of these vertices is:
r-u-q-t-y-x-z-s-v-w

The forest produced by line 3 is:
[r]->[u]->[q, y, t] -> [x, z] -> [s, w, v]

22.5-3
Professor Deaver claims that the algorithm for strongly connected components can be simplified by using the original (instead of the transpose) graph in the second depth-first search and scanning the vertices in order of increasing finishing times. Is the professor correct?

Deaver is incorrect:

Using a graph where V = (X, Y, Z) and E = (X -> Y, Y->X, X->Zz), the first call to DFS results in the following finishing times:

x	f
X	6
Y	3
Z	5

Sorting by increasing finishing time gives us: B-C-A.
The DFS using this order gives us one component: {B, A, C}, which isn't correct, as {A,B} is a component and {C} is a component, giving us two.

References
Third edition of the textbook
http://jt-web-site.tripod.com/MSCS/COMP510/Assignment12/Assignment12.htm
http://student.csuci.edu/~douglas.holmes253/Assignment6.html (for second edition copy of questions)
http://en.wikipedia.org/wiki/Strongly_connected_component
http://en.wikipedia.org/wiki/Bipartite_graph