Spencer Shimko
CMSC 441
HW 9

1.Exercise 22.5-7.

A directed graph *G* = (*V*, *E*) is said to be semi-connected if, for all pairs of vertices *u*, *v* ε *V*, we have *u* -> *v* or *v* -> *u*. Give an efficient algorithm to determine whether or not *G* is semi-connected. Prove that your algorithm is correct, and analyze its running time.

Note: I approach this problem assuming "semi-connected" is equivalent to weakly connected for this particular algorithm since directionality is alluded to (using the directional arrows) but not defined in the question.

Start by finding all of the strongly connected components of the graph *G*. According to CLRS page 1082, strongly connected components are the equivalence classes of vertices under the "are mutually reachable" relation. Obviously, from this definition, we can traverse from any vertex to any other vertex listed in the same component. Each of these components can strongly connected component can shrink to a single vertex. This will create a new directed acyclic graph *G'*. From this we can see that any semi-connected (weakly connected) graph *G* is semi-connected iff *G'* is semi-connected. Now we need to place the vertices in *G'* order topologically (DFS/Topological sort). Number the vertices in this list from *c*0..*cj*. Since this graph is acyclic we know that for any node *ci* < *cj* there exists a path from *ci*->*cj* but not from *cj*->*ci*. The graph *G'* is semi-connected if for all *i* there exists an edge between *ci* and *ci+1* (*ci*->*ci+1*). The runtime

performance of this algorithm is $O(|E|)$.

An additional note: a much more simple algorithm is to use DFS from each node v to create a list of all nodes visited from that starting point (precisely as we did in the last homework). Do that for all nodes. Then take every possible pair of vertices from V and ensure that each is listed in the others DFS list. The drawback to this algorithm is it runs in $O(|V||E|)$.

2. **Exercise 23.1-6.**

Show that a graph has a unique minimum spanning tree if, for every cut of the graph, there is a unique light edge crossing cut. Show that the converse is not true by giving a counterexample.

Converse: There is a unique light edge crossing the cut if the graph has a unique minimum spanning tree, MST.

Suppose there is a unique light edge at each cut and there are two MSTs for the graph. W can show that on of these trees can be changed into the other to form a “lighter” spanning tree. This means that it is not minimum and therefore there is a unique MST.

Suppose, without loss of generality, that the two MSTs are exactly the same except for a pair of edges such that one edge is used by one tree and the other edged is used by the other tree. Since these are spanning trees, we can come up with a cut that crosses the two edges, only crossing one edge in each tree. One of these two edges must have a lower weight (assuming unique weights). We can replace the tree that uses the higher weighted edge

with the other tree, and now it is a better tree. Therefore, the tree could not have been a minimum.

3. **Exercise 23.1-8.**

   Let T be a minimum spanning tree of a graph G, and let L be a sorted list of the edge weights of T. Show that for any other minimum spanning tree T' of G, the list L is also the sorted list of edge weights of T'.

4. **Exercise 23.2-4.**

   Suppose that all edges weights in a graph are integers in the range from 1 to $|V|$. How fast can you make Kruskal's algorithm run? What if the edge weights are integers from 1 to W for some constant W?