CS5800 Algorithms Spring 2015 Assignment #3

In the union algorithm, when combining trees which cannot otherwise be distinguished (such as being of the same rank), the new root should be from the tree specified by the first parameter to union.

In the DFS algorithm, when there is a choice for the next vertex, use the one that is smaller numerically.

When an algorithm is to be executed on an example, show the calls, operations and depths of calls and operations as usual.

1. (25 points) Software development projects are divided into relatively small units of work called **tasks**. Tasks can **depend on** other tasks. A dependency between tasks is **direct** if it is explicitly specified or **indirect** if it is a consequence of explicitly specified direct dependencies. For example, if X directly depends on Y and Y directly depends on Z but X does not directly depend on Z, then X indirectly depends on Z. In a software development project all of the tasks require the same amount of development effort D. It is possible that two tasks can depend on each other (directly or indirectly). Each time this happens, an additional coordination effort C is required. Develop an algorithm for computing the effort required for a software development project. The input is the set of tasks and their direct dependencies. Your algorithm must have a linear time complexity in the size of the input. Prove that your algorithm has linear time complexity.

2. (15 points) Run the union-find algorithm using path compression on the following set of edges in the order given:

(5,7) (4,2) (4,6) (6,2) (8,1) (7,1) (2,8) (7,6) (9,3) (5,2) (3,8)

3. (22 points) Simultaneously run the DFS algorithm and union-find using union with ranks on the following directed graph. Union is performed on each edge just before calling explore recursively in the loop over the adjacent edges. The union operation is performed only when explore is actually called on the other vertex of the edge.

Edge	Edge	Edge	Edge
(1,3)	(1,6)	(2,3)	(2,9)
(3,9)	(4,1)	(4,2)	(4,3)
(4,6)	(4,7)	(4,10)	(5,2)
(5,3)	(5,9)	(6,2)	(6,3)
(6,10)	(7,5)	(7,6)	(7,9)
(7,10)	(8,2)	(8,5)	(8,7)
(8,9)	(8,10)	(10,2)	(10,3)

4. (18 points) Perform BFS on the following directed graph, starting with vertex 9:

Edge	Edge	Edge	Edge
(1,7)	(2,5)	(3,1)	(3,2)
(3,5)	(5,1)	(5,4)	(5,11)

(6,5)	(8,1)	(8,3)	(8,6)
(8,7)	(8,10)	(8.11)	(9,1)
(9,2)	(9,4)	(9,6)	(9,10)
(10,5)	(10,6)		

5. (20 points) Show what happens when Tarjan's strongly-connected component algorithm is applied to the following directed graph:

Edge	**Edge**	**Edge**	**Edge**
(1,5)	(1,6)	(3,4)	(3,7)
(3,10)	(4,2)	(5,4)	(5,7)
(5,8)	(5,10)	(7,5)	(6,8)
(8,2)	(8,7)	(8,10)	(9,1)
(9,3)	(9,4)	(9,8)	(9,10)
(10,2)	(10,5)		

Use the technique explained in class. Show how the strongly-connected components "grow" as the DFS algorithm proceeds. In other words, as each edge is processed by the DFS algorithm, show what the strongly-connected components are up to that point in the algorithm. When there is a change in the set of SCCs, show what happens and explain why it happens. Note that one shows an SCC by simply listing the nodes in the SCC. It is not necessary to draw a picture.