## CS 575: Design and Analysis of Algorithms, Spring 2020

**Solution to Theory Assignment 3.2**

1. [20 points] In this problem, you will apply different traversal methods on a given directed graph. The start node is denoted by *S*. When arbitrary decisions on order must be made assume that child nodes are visited from left to right.
2. [10 points] Perform a breadth-first search on the directed graph below (on the left). (i) *Number the nodes* according to the order in which they are visited (become gray). For example, the first node visited is *S* so *S* is numbered 1. Then, the left child of *S* is visited so it is numbered 2. Show the order numbers inside the circles. (ii) Show the distance of each node to *S* beside each circle. (iii) Show the breadth-first tree.



**Answer**: The results are shown in the figure on the right above. The numbers in the circles are sequence numbers based on the order of visit. The numbers outside the circles are the distances to *S*.

1. [10 points] Perform a depth-first search on the directed graph below (on the left). (i) Number the nodes according to the order in which they are visited. Show the order numbers inside the circles. (ii) Show the depth-first tree(s).



**Answer**: The results are shown in the figure on the right above. The numbers in the circles are sequence numbers based on the order of visit. Unlabeled (red) edges are the tree edges which form the depth-first tree. The edges marked by F and C are forward edges and cross edges, respectively (I did not ask you to mark this, you can ignore these).

1. [15 points] Please modify the depth first search algorithm (slide 36 and 37) to find all connected components in an undirected graph. Comment on where you made the modification. You need to print out each component ID (starting from 1) and the corresponding vertices. For example, your output for the DFS example (slide 38-41 of the graphs basics lecture notes) will look like the following: Component 1: u, v, y, x. Component 2: w, z.

Answer:

**DFS** (G:graph; **var** color:carray; parent:parray);

**for** **each** vertex u **do**

color[u]=white; parent[u]=nil;

**end** **for**

groupID = 1 //Initialize the group ID

**for** **each** vertex u **do**

**if** color[u] == white **then**

Print groupID **//Print the groupID**

*DFS-Visit*(u);

*groupID=groupID+1;* **//The next potential groupID**

**end** **if**

**end** **for**

**end** *DFS*

DFS-Visit(u)

{

color[u]=gray;

print “u” //print out the vertex

**for** **each** v in adj[u] **do**

**if** color[v] = white {

parent[v] = u;

DFS-Visit(v);

}

color[u] = red;

}