Cpt S 515 Homework #2

No late homework!

1. In Lesson 3, we talked about the Tarjan algorithm (SCC algorithm). Now, you are required to find an efficient algorithm to solve the following problem. Let G be a directed graph where every node is labeled with a color. Many nodes can share the same color. Let v_1, v_2, v_3 be three distinct nodes of the graph (while the graph may have many other nodes besides the three). I want to know whether the following items are all true: there is a walk α from v_1 to v_2 and a walk β from v_1 to v_3 such that

- α is longer than β;
- α contains only red nodes (excluding the two end nodes);
- β contains only green nodes (excluding the two end nodes).

2. In Lesson 4, we learned network flow. In the problem, capacities on a graph are given constants (which are the algorithm's input, along with the graph itself). Now, suppose that we are interested in two edges e_1 and e_2 whose capacities c_1 and c_2 are not given but we only know these two variables are nonnegative and satisfying $c_1+c_2 < K$ where K is a given positive number (so the K is part of the algorithm's input). Under this setting, can you think of an effcicient algorithm to solve network flow problem? This is a difficult problem.

3. There are a lot of interesting problems concerning graph traversal — noticing that a program in an abstract form can be understool as a directed graph. Let G be a SCC, where v_0 is a designated initial node. In particular, each node in G is labeled with a color. I have the following property that I would like to know whether the graph satisfies:

> For each inifintely long path α starting from v_0, α passes a red node from which, there is an infinitely long path that passes a green node and after this green node, does not pass a yellow node.

Please design an algorithm to check whether G satisfies the property.

4. Path counting forms a class of graph problems. Let G be a DAG where v and v' be two designated nodes. Again, each node is labeled with a color.

(1). Design an algorithm to obtain the number of paths from v to v' in G.

(2). A good path is one where the number of green nodes is greater than the number of yellow nodes. Design an algorithm to obtain the number of good paths from v to v' in G.

Assignment 2

Sheryl Mathew

October 2, 2018

1. **Question 1**

   Step 1: Run a depth first search along the edges of G starting from V_1.

   Step 2: As we traverse along the edges, record the finishing time for each node within the node.

   Step 3: Flip the graph in such a way that all the edges from one node to another are reversed by keeping the finishing time of the node.

   Step 4: Consider 3 values IsRed, IsGreen, NeitherRedGreen. If the node is red in colour then then we set IsRed to the node. If the node is green in colour then then we set IsGreen to the node. If the node is neither red or green in colour then then we set NeitherRedGreen to the node.

   Step 5: Perform depth first search fromV_2 to V_1 and store one of the values IsRed, IsGreen, NeitherRedGreen for each node along the path in an array A_1. In A_1 don't include V_2 and V_1.

   Step 6: Perform depth first search fromV_3 to V_1 and store one of the values IsRed, IsGreen, NeitherRedGreen for each node along the path in an array A_2. In A2 don't include V_3 and V_1.

   Step 7: If length of A_1 > length of A_2, then

   $$lenght\ of\ \alpha > \ length\ of\ \beta$$

   Step 8: Check if A_1 contains only IsRed values, then

   $$\alpha\ contains\ only\ Red\ nodes$$

   Step 9: Check if A_2 contains only IsGreen values, then

   $$\beta\ contains\ only\ Green\ nodes$$

2. **Question 2**

Step 1: Initialize $C_1 = 0$ and $C_2 = K - 1.C_1$ Let G be the network flow

Step 2: Construct a residual graph G_r

Step 3: For an augmenting path A of G_r send maximum flow f through it.

Step 4: After sending maximum flow calculate the value of C_1, C_2

Step 5: If $C_1 + C_2 < K$:

Then update the residual graph and keep repeating till the condition remains true until no augmenting path remains.

Step 6: If $C_1 + C_2 >= K$:

Then increment C_1 by 1 and decrement C_2 by 1 and repeat Step 3 onwards till the condition becomes true

3. **Question 3**

Step 1: Perform SCC of G by running a depth first search over G starting from V_o after dropping all the yellow nodes.

Step 2: As we traverse along the edges, record the finishing time for each node within the node.

Step 3: Flip the graph in such a way that all the edges from one node to another are reversed by keeping the finishing time of the node.

Step 4: Find the different sets of SCC

Step 5: In the different SCC check if any one SCC involves V_o, red node and green node.

Step 6: Then check if the SCC containing V_o , red and green node has a loop or not ie either the size of the SCC should be greater than 1 or either red or green node has a self loop or both have a self loop

4. **Question 4**

Algorithm 1:

Step 1: Consider a variable n = 0 which stores the number of paths from v to v'

Step 2: Start Depth First Search from v and move on to the next node from v ie v_{next}

Step 3: If v_{next} != v', then move on to the next node which in turn becomes the new v_{next} node.

Step 4: If v_{next} = v', then update n by 1 and start depth first search from v once again taking another path from the previous path.

Step 5: If v_{next} = v' condition is never satisfied then there is no path from v to v'. Therefore n=0

Algorithm 2:

Step 1: Consider a variable n = 0 which stores the number of paths from v to v', g = 0 which stores number of green nodes, y = 0 which stores number of yellow nodes

Step 2: Start Depth First Search from v and move on to the next node from v ie v_{next}.

Step 3: If v_{next} != v', then check if v_{next} is a green node, then update g by 1 and if v_{next} is a yellow node then update y by 1. Then move on to the next node which in turn becomes the new v_{next} node.

Step 4: If v_{next} = v' and g>y, then update n by 1 and start depth first search from v once again taking another path from the previous path.

Step 5: If v_{next} = v' and g>y condition is never satisfied then there is no path from v to v' in such that a way that number of green nodes is greater than yellow nodes. Therefore n=0