Cpt S 450 Homework #7 Solutions

1. G is a DAG iff G does not have a loop. Hence, you only need to run SCC algorithm to make sure that there is no SCC containing more than one node. Complexity: linear – $O(V + E)$.

2. For this problem, the easiest way is to use automata theory approach (instead of DFS). From the decorated graph G, we construct a machine M as follows. M is eqwuipped with an integer variable x. The nodes in G corresponds to the states in M. Hence a run of M corresponds a walk over G. Initially, M starts with state v and with $x = 0$. M walks on the graph while changing its state (for instance, if M's current state is v_1 and, then, when it walks on the edge (v_1, v_2), the next state of M will be v_2). In particular, M increments (resp. decrements) x by one when a red (resp. a green) edge is walked. M halts when the node v' is the current state and $x \leq 0$. Notice that M halts iff the answer to the question asked in the problem is false. Notice also that M can be modified into a PDA (use a stack to simulate the variable x) and from 317, we know that we have an algorithm to check whether a PDA halts.

3. Notice that the answer to the question asked in the problem is false iff we have a path p from v to v' on which the number of red edges or the number of green edges is < 4. So, to solve the problem, you need only solve the following problems:

(1). Do we have a path from v to v' on which the number of red edges is 0, 1, 2, or 3?

(2). Do we have a path from v to v' on which the number of green edges is 0, 1, 2, or 3?

Both problems can be solved by a simply change to the DFS algorithm over G.

4. First notice that an infinite path must contain loops (since G has only finitely many states). The liveness property is not true iff there is an infinite path p from s_0 such that every state on p is marked with not-good. The existence of such a p is equivalent to the following statement:

there is a finite path q on G such that q starts with s_0 and q contains a loop and every state on q is marked with not-good.

Notice that the above statement can be justified the following algorithm:

Drop all states that are marked with good from graph G
Run SCC on G
For each state r in G do the following:
use DFS to check whether s_0 can reach r
check whether there is an edge from r to r in G or r is contained in a SCC that has at least two nodes
if both checkings are yes, then return yes;
return no.