cpts350 hw6

1. Let G be a DAG (a graph without loops) and u, v be two designated nodes (there are many other nodes in G). Design an efficient algorithm to count the number of paths from u to v.

2. Let G be a DAG (a graph without loops) and u, v be two designated nodes (there are many other nodes in G). In particular, each node in G is labeled with a color and multiple nodes can share the same color. A good path is one where the number of green nodes is bigger than the number of yellow nodes. Design an efficient algorithm to count the number of good paths from u to v.

3. Let G be a graph (so it may have loops) and u, v be two designated nodes (there are many other nodes in G). In particular, each node in G is labeled with a color and multiple nodes can share the same color. Suppose that γ is a regular expression on colors (e.g., $(green + yellowyellow)^* yellowblue$). An ugly path is one that the color sequence on the path satisfies the regular expression γ. Design an efficient algorithm to count the number of ugly paths from u to v (when the count is infinite, return ∞). (Hint: you have two cases to consider, afer using a Cartesian product construction: a. the count is infinite – where an SCC algorithm can be used to decide. b. the count is finite, where prob 1 can be used.)

4. Let G be a graph that may contain loops and hence, the number of paths from a designated start node to a designated end node may be infinite. Unfortunately, you usually can't say that one infinite number is larger than another. Here is the problem: sketch a way to compare the number of paths in two graphs (that both may contain loops). (Hint: google perron, graph, path count).

5. Let G be a control flow diagram of a C-progrm (which can be automatically generated). For each node u in the diagram, one can obtain the total number $C(u)$ of paths from the root of the diagram to u. Then, from what you got from 4 above, you may sort all the $C(u)$'s for all u (even though some of $C(u)$'s are infinite) and then pick a u^* that has the maximal $C(u)$. Write a mini-paper on how this will address the problem of testing a C-program. (if you want, you can actully publish a paper on this get a Master degree!)

6. (a job interview question for a top tech company) Design an efficient algorithm to compute the number of binary strings with length n that satisfy

the regular expression $((0 + 11 + 101)^*(1101))^*$. (Hint: use Prob 3 above. I will talk about the solutions in class.)