CS3110: Assignment 3
May 30 2017

Q1. G is a graph with N nodes, and every node has a unique label.
Provide an algorithm *GetNode(source, d)* that given a *source* node and an integer *d*, returns the node with minimum unique number at minimum distance d from *source*. (35)

Q2. Given a graph G, design an algorithm that can detect a cycle in O(|V|) (30)

Q3. A directed graph G =(V,E) is semiconnected if, for all pairs of vertices u, v, there is either a path from u to v, **or** from v to u.

1. If G does not have cycle, give an efficient algorithm (state the complexity) to test for semiconnectedness. Proof the correctness of the algorithm (35)
   Tip: use the topological sorting of the graph

2. **Bonus**: How about the case where G can have cycles? (10)
   Tipe: use the algorithm discussed in the class to extract the strongly connected component, and build on that)

Q4 (Implementation). Bonus (50): Implement an algorithm that prints out a solution for this version of the classic "*river crossing*" (your solution does not have to be optimal, in fact I like to see **bad solutions (**by bad I mean unnecessary moves)!).

Tip: Encode each state as an array of size 9, the first 8 elements represent the position of the passengers, and the last bit the position of the boat. The root is 000000000 and the goal is 111111111. In the BFS (or DFS) search make sure you visit nodes that are not visited before (which is obvious) and are legal. A move is legal if:

1. It flips one or two elements of the array, consistent with the position of the move
2. The move does not violate any of the rules of the game.
   You have an extra week for this question.