Written Problems. due Feb 24 (Wed), 6.30 pm.

1. **(Counting Topological Orderings)** Textbook Chapter 3, Exercise 1

2. **(Finding a Cycle)** Textbook Chapter 3, Exercise 2

Programming Problems. due Feb 26 (Fri), 11.59 pm.

1. **(Counting Shortest Paths)** Recall that the breadth-first search (BFS) algorithm allows us to compute the shortest paths between any two nodes in a graph.

   (a) You are provided with a buggy implementation of the BFS algorithm (that given a graph G, outputs a BFS tree for G) in `bfs.py`. Fix the bugs; this should require only changing a few lines of code. Insert comments in your code pointing out where the bugs are.

   (NOTE. You may assume that the input graph G does not contain repeated edges.)

   (b) Next, you are asked to implement `bfs_path` in `bfs.py`, which given a graph and two nodes s, t, outputs the shortest path from s to t.

   (c) Textbook Chapter 3, Exercise 10. Implement an algorithm that given a graph G and two nodes s, t in G, counts the number of shortest s-t paths in G. Put your implementation in `count_shortest_paths.py`. Comment your code with brief explanations as to how your algorithm works and why it runs in $O(m + n)$ time.

   (HINT. As in BFS, it is easier to solve a more general problem. For BFS, instead of computing the shortest path from a specific node s to a specific node t, we *incrementally* compute the shortest paths from a specific node s to *all* nodes t. Here, we will compute the number of shortest paths from the node s to *all* nodes t. You should do this computation incrementally for each layer.)

   Run `test_bfs.py` to help determine if your solutions work. Submit `bfs.py` and `count_shortest_paths.py` by copying the files to your `submit` directory. The test suite in `test_csp.py` must complete running in under 0.5 secs on `owl.cs.qc.edu`.