1. Go to csilm.usu.edu, CS3, Graph Storage. Create a graph. Note, the buttons at the bottom to create a random graph (New Graph) and Show Storage. Look at the various methods of storing the graph [adjacency list, adjacency matrix, edge list]. From the Show Storage window, experiment with the operations (Successors, Predecessors, Are Adjacent) to see the number of operations required. For each of the three methods of storage, give a situation where the method of graph storage is a superior way to represent the graph.

2. For the graph below, trace through the algorithm of Figure 9.7 to compute a topological ordering. At each iteration of the while loop, indicate v, the topNum (Topological order number) for v, and the values in q.

3. A friend says, "When a topological ordering has been performed, each vertex has a topNum such that if v is labeled with topNum of k, all vertices that can be reached from v have topNums greater than k." Is this true? How can finding a topological ordering be useful in other algorithms?

4. Go to csilm.usu.edu, CS3, Dijkstra's Shortest Path. Experiment with the algorithm. When you click on a demo, you have to click on the applet area several times. Each time you click, it takes the next step of the algorithm. Once you are familiar with how Dijkstra's algorithm works, on paper, create an example for which negative edge weights cause a problem in finding the shortest path from node A to all other nodes.

Notes

Turn in your written homework through Eagle in a .doc, .odt, .pdf format or you can bring a paper copy of the homework with you to class. It will be graded by randomly selecting a subset of problems to evaluate. Not every problem will be graded. Bring a copy of the answers to class so that we can discuss them.

Written homework provides an excellent framework for achieving the goals of obtaining a working knowledge of data structures, perfecting programming skills, and developing critical thinking strategies to aid the design and evaluation of algorithms. Since programming has a high overhead in terms of program entry and debugging, all important topics in this course cannot be covered via programming projects. Written homework exercises allow students to learn important material without a high time investment. Although the point value is low, the benefits are great. You can perfect your programming skills without spending hours at the computer and can get feedback on your thinking skills from your study partners.
Students that consistently do quality homework, have far superior test scores. Because assignments are done as a group and any questions are discussed in class or during office hours, written solutions to the homework will not be provided.

Note, these exercises may be done in groups of one, two, or three. If more than one person is involved, list all the names on one set of answers. Groups may change throughout the semester. Answers should not be compared with others not in your group. You will learn much more by working in a group than you will learn working by yourself.