Review Problems:

Here are a few additional practice questions on graphs to help you prepare for the final. This set is not comprehensive, although if you work through these questions in addition to the homework questions and in-class problems you should be well-prepared for the graph questions on the final.

1. (a) Find a topological sort of the following directed acyclic graph. You need only list the vertices in order, although showing your work will help earn partial credit if you make a mistake.

![Directed Acyclic Graph](attachment:image.png)

**Solution:** Here are the six different possibilities:

6, 5, 4, 2, 7, 3, 1
6, 5, 4, 7, 2, 3, 1
6, 5, 4, 7, 3, 2, 1
6, 5, 7, 3, 4, 2, 1
6, 5, 7, 4, 3, 2, 1
6, 5, 7, 4, 2, 3, 1

(b) This graph has more than one topological sort. Briefly describe the general structure of directed acyclic graphs that have exactly one topological sort.
Solution: Any directed acyclic graph where there is a single path joining all the vertices. There may be more edges, as long as they don’t form a cycle. Thus, it is not enough to say that the graph is a chain of edges.

2. Find an example to show that Dijkstra’s algorithm gives the wrong answer when there is a graph with negative edges but no negative cost cycle.

Solution: Here’s a simple case. In the graph shown, Dijkstra’s algorithm computes a distance of 2 to C because it is marked known first. The correct shortest distance is 1 because of the negative weight.

3. Does Prim’s algorithm work when there are negative weights in the graph? Why or why not?

Solution: Yes. Looking at Prim’s algorithm, the only issue that truly matters is a rank ordering of the edge weights (distances). This determines the order in which vertices are added to the MST. Hence, negative edges do not affect the algorithm.

4. 9.13b on page 380

Solution: This question is asking how to map the problem specification into the bipartite matching problem, not how to solve the bipartite matching problem. Question 9.13c asks how to solve the bipartite matching problem using techniques you are not responsible for knowing. Hence we only need to consider the mapping here.

The solution is to make each instructor a vertex, each course a vertex, and form an edge between an instructor and a vertex if the instructor is capable of teaching the course.