Decrease-by-one algorithms: Topological Sort

- Read 4.2, pages 138-141
  - How does DFS for a directed graph differ from one for an undirected graph?
    - Are *cross edges* in this section the same as the cross edges for BFS?
  - True or False: The presence of a forward edge indicates the presence of a directed cycle.
  - What graphs to do we call **directed acyclic graphs** (dag)?
  - Explain how we can represent the set of courses a student has to take for a major as dag.
  - What do we refer to as a **topological sort** of a dag?
    - Why does this only make sense for dags and not for all directed graphs?
  - Give some real life examples where a topological sort might be needed.
  - Construct at least two different topological sorts of the dag in Figure 4.6.
    - How many different topological sorts can you construct?
  - Explain how DFS can help us construct a topological sort. Make sure to also explain why the resulting order is for sure a topological sort.
  - Describe the decrease-by-one-and-conquer algorithm to performing a topological sort.
    - What vertex do we need to find in each step of the algorithm?
    - What do we do with that vertex after we find it?
    - How do we obtain the resulting topological sort order?
    - What's the name of this algorithm?
  - Some questions about this decrease-and-conquer algorithm:
    - Are we certain that every dag has a source?
    - How would we find such a vertex in a graph represented as an adjacency list? What is the time efficiency of that step?
  - How can we implement the decrease-and-conquer algorithm in time $O(|V| + |E|)$?
  - Work out problem 4.2.9 about **strongly connected components**.