Depth-first search

- Read 3.5, pages 122-125
  - Describe the basic idea of the depth-first search approach to traversing the vertices of a graph.
  - When does a vertex become **visited**? When does it become a **dead-end**?
  - Explain why one might want to use a *stack* to keep track of the vertices as they are being visited.
  - The presence of the stack suggests two kinds of orders to the vertices, visually shown in part b of Figure 3.10. Explain what those two orders are.
    - Explain which of these orders the count variable in the algorithm tracks.
  - Looking at the graph of Exercise 3.5.1, carry out the DFS process following the alphabetical ordering of the edges.
  - What graph do we refer to as the **depth-first search forest** of a DFS search?
  - What edges of the original graph do we refer to as **tree edges** and which do we refer to as **back edges**?
  - Carry out the construction of the depth-first search forest for the graph from exercise 3.5.1.
  - Repeat the process for the same graph but now using reverse alphabetical ordering (so g is the vertex that is visited first).
  - Is the algorithm in the book about DFS recursive or non-recursive?
  - True or False: When starting from a root vertex, all vertices within the same connected component as that vertex will be visited within that iteration and no other vertices will be visited (unless we restart from a new vertex).
  - True or False: The original graph has a cycle if and only if we end up having "back edges".
  - Explain the run-time efficiency of $\Theta(|V| + |E|)$ for the DFS algorithm (Hint: How many times are each vertex and each edge visited?).