Basic Graph notes
Cole Frederick, Spring 2017

Graph structures
In computer science, graphs (networks) represent relationships between entities. We implement them in two general ways: adjacency lists and adjacency matrix.

Breadth First Search
(see website: https://visualgo.net/en/dfsbfs)
- Starts at a single start vertex (first item added to the queue)
- Iteratively takes a node from the queue, adds all neighbors (in no particular order) to the queue
- When queue empties, does not restart, the search is finished
  - Some vertices may never enter the queue (still white at the end)
- Finds shortest paths (in terms of edges) from a start vertex to all reachable vertices
- Runtime
  - Processes up to every node in the graph, inspects up to every edge in the graph
  - Queue operations enqueue and dequeue can be constant time
  - Setup and bookkeeping all linear in number of nodes, edges
  - $O(n + m)$ overall, linear in size of graph

Depth First Search
(see website: https://visualgo.net/en/dfsbfs)
- Starts at any white vertex
- Marks it grey (note discovery time), recursively searches all white neighbors (in no particular order), then marks it black (note finish time)
- Restarts at white nodes until none are left
- Runtime
  - Processes every node and every edge
  - Stack operations push pop are constant time
  - Setup and bookkeeping are linear in size of graph
  - $O(n + m)$ overall

Topological Sort
- Orders the nodes of a directed acyclic graph (DAG) such that all edges are directed from left to right
- Equivalently, orders nodes by descending finish time
• Run DFS, as nodes finish add them to the front of a linked list. This list will contain the nodes in descending finish time.

• Runtime
  ○ DFS O(n + m)
  ○ List building O(n)
  ○ O(n + m) overall

Strongly Connected Components

• Runs on directed graphs
• A strongly connected component is a group of nodes where there is a path from every node to every other node in the group (they form one or more cycles)
• The problem is to find all strongly connected components. The SCC graph is a graph of the strongly connected components. It’s like the original graph with cycles collapsed into single nodes. The SCC graph is always a DAG.
• Run DFS, note finish times. Take the transpose of the graph (edge directions reversed). Run DFS on transpose, starting on nodes in descending finish time order.
  ○ Each start will recursively blacken all reachable white nodes. This group of nodes is a strongly connected component

• Runtime
  ○ Two DFS - O(n + m)
  ○ Transpose - O(n + m) for adjacency list representation
  ○ O(n + m) overall