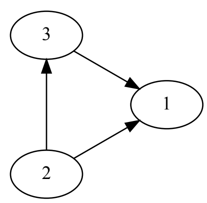
CS41 lab 4

In typical labs this semester, you'll be working on a number of problems in groups of 3-4 students. You will not be handing in solutions; the primary purpose of these labs is to have a low-pressure space to discuss algorithm design. However, it will be common to have some overlap between lab exercises and homework sets.

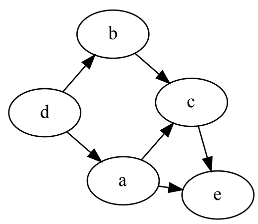
The learning goals of this lab session are to gain more experience with directed and undirected graphs, and to practice algorithm design for graph problems. There are more problems on this writeup than you have time to complete. Consider the lab a success if you can make good progress on two problems.

1. **Topological Sorting.** For each of the following directed graphs, determine whether or not the graph is *acyclic*. If the graph is cyclic, identify a cycle. If the graph is acyclic, give a topological ordering of vertices.

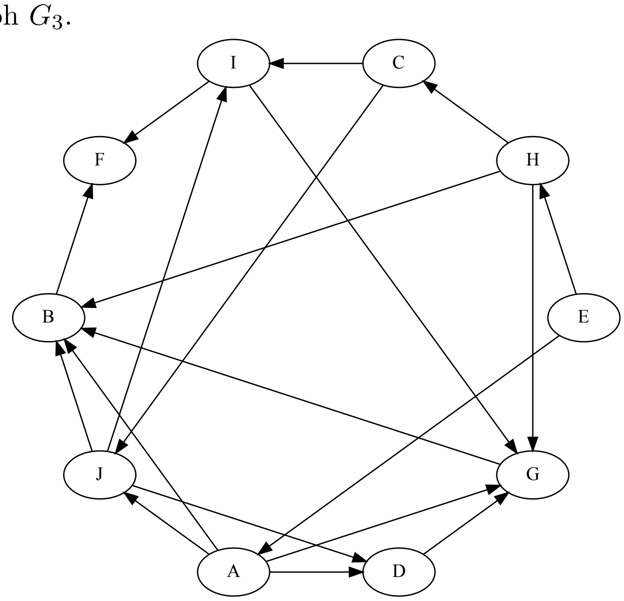
   (a) Directed graph G_1.



   (b) Directed graph G_2.



   (c) Directed graph G_3.



2. **(K&T 3.9)** There's a natural intuition that two nodes that are far apart in a communication network—separated by many hops—have a more tenuous connection than two nodes that are closer together. There are a number of algorithmic results that are based to some extent on different ways of making this notion precise. Here's one that involves the susceptibility of paths to the deletion of nodes.

   Suppose that an n-node undirected graph $G = (V, E)$ contains two nodes s and t such that the distance between s and t is strictly greater than $n/2$. (The **distance** between two nodes is the number of edges along the shortest path between them.)

   (a) Show that there must exist some node v, not equal to either s or t, such that deleting v from G destroys *all* paths $s \rightsquigarrow t$.

   (b) Give an algorithm with running time $O(m + n)$ to find such a node v.

3. **Testing Tripartiteness.** Call a graph $G = (V, E)$ *tripartite* if V can be partitioned into disjoint sets A, B, C such that for any edge $(u, v) \in E$, the vertices u, v lie in different sets. In other words, a tripartite graph is three-colorable.

   (a) Design and analyze an algorithm which takes as input an undirected graph $G = (V, E)$ and returns YES if G is three-colorable, and NO otherwise.

   (b) Design and analyze an algorithm which takes as input a *three-colorable* graph $G = (V, E)$ and colors the vertices of the graph using at most $\Delta + 1$ colors, where Δ is the largest degree (number of neighbors) of a vertex. (Note: while the input graph *is* three-colorable, it does not mean that we know what that coloring is!)

   (c) Design and analyze an efficient algorithm which takes as input a *three-colorable* graph $G = (V, E)$ and colors the vertices of the graph using $O(\sqrt{n})$ colors. (Note: while the input graph *is* three-colorable, it does not mean that we know what that coloring is!)