CMPS 256 **Problem Set 6**

Style guide and expectations: Please see the "Homework" part of the syllabus for guidance on what we are looking for in homework solutions. We will grade according to these standards. You should cite all sources you used outside of the course material.

What we expect: Make sure to look at the "**We are expecting**" blocks below each problem to see what we will be grading for in each problem!

Exercises

We suggest you do these on your own. As with any homework problem, though, you may ask the course staff for help.

---

1. **[BFS and DFS Recap.]**
   Give one example of a *connected undirected* graph on four vertices, A, B, C, and D, so that both DFS and BFS search *discover* the vertices in the **same** order when starting at vertex A. Then give one example of a *connected undirected* graph on four vertices, A, B, C, and D where BFS and DFS discover the vertices in a **different** order when started at A. Assume that both DFS and BFS iterate over neighbors in *alphabetical order*.

   Above, *discover* means the time that the algorithm first reaches the vertex.

   Note on drawing graphs: You might try `http://madebyevan.com/fsm/` which allows you to draw graphs with your mouse and convert it into LaTeX code. (By default it makes directed graphs; you can add an arrow in both directions to get something that approximates an undirected graph).

   Sticky Graph (`http://jzacharyg.github.io/StickyGraph/`) is another useful tool for drawing graphs. You can use the +v and +e buttons on the bottom right corner to add nodes and edges, and the `TikZ` button on the bottom left corner to generate the corresponding LaTeX code.

   [**We are expecting:** A drawing of your two graphs **and** an ordered list of vertices discovered by BFS and DFS for each of them.]

2. **[DFS Order.]** Consider the result of a run of the depth first search algorithm (presented in class) on a *directed* graph. Explain how it is possible that some vertex u that has both incoming and outgoing edges in the graph ends up in a tree containing only itself in the DFS forest, i.e., we explore no node following u's edges and that node u is not explored by following some edge. Construct an example and explain why, in your example, the DFS algorithm can end up having such a node u. For this question, assume the graph has no self-loops.

   [**We are expecting:** An example directed graph G and a brief explanation.]

Problems

- Try the problems on your own *before* asking questions.
- Write up your answers yourself, in your own words. You should never share your typed-up solutions.

---

3. **[Exact Traversal.]** Let G be a directed **acyclic** graph. Let $v_1, v_2, \ldots, v_n$ be a **topological order** on G. An **exact traversal** is a path that touches all nodes of G exactly once.

   (a) Prove that G has an exact traversal *if and only if* $v_i \to v_{i+1}$ is an edge for all $i = 1, 2, \ldots, n-1$.

   [**We are expecting:** An informal justification for both directions of *if and only if*.]

   (b) Let $m = |E|$ and $n = |V|$. Describe an $O(m+n)$ time algorithm that determines whether an exact traversal exists. No need to justify the correctness of the algorithm.

   [**We are expecting:** Pseudocode for the algorithm and a brief justification for the running time.]

4. [Viewpoints.] Given a directed graph $G = (V, E)$, let us call a node $v \in V$ a *viewpoint* if all other nodes are reachable from v (in other words, v has a path to all other nodes).

(a) Recall that any directed graph can be thought of as a meta-graph of its strongly connected components (SCCs), and that this meta-graph is acyclic. Briefly argue that if G has a viewpoint node, G must have only one *source* SCC, which is an SCC such that there are no edges from another SCC to it. (You may assume that at least one *source* SCC exists.)

[**We are expecting:** A brief justification on why multiple *source* SCC cannot exist if a viewpoint node exists.]

(b) Let $m = |E|$ and $n = |V|$, give an $O(n + m)$-time algorithm to find a viewpoint node in a directed graph, or report that none exists. You may use any algorithm covered in class directly.

[**We are expecting:** A description of the algorithm, a brief explanation of why the algorithm works and an informal justification of the algorithm's running time.]

5. [Safe Return Party Planning.] You are a party planner. You carefully chose $n \geq 3$ guests to invite to a party so that for every two guests x and y, even if x doesn't know y, at least x knows a guest who knows a guest who knows a guest who ... (this might go on for any number of guests) ... who knows y (we call this the *fun party* property). Note that "knowing" is bidirectional (i.e. a graph with edge representing "knowing" should be undirected). But now, moments before sending out the invitation, you were told that you can only invite $n - 1$ guests.

Explain why there exists a guest that you can un-invite without violating the *fun party* property, and give an efficient algorithm for finding such a guest. Give an algorithm that finds such a guest in $O(m + n)$ time, where m is the number of pairs of guests who know each other.

[**We are expecting:** A description of the algorithm, a brief explanation of why the algorithm works and an informal justification of the algorithm's running time.]

6. **[Optional] Robust Fuel Distribution.** In September 2019, an explosion in Shelby County, Alabama severed the Colonial Pipeline which transports more than 100 million gallons of refined petroleum products from Houston to the southeast every day. The incident caused gasoline prices to jump 15% and might cause jet fuel and gasoline shortages in the region. This event has made you concerned about fuel supply/price stability in the US.

   As a student of CMPS256, you would like to analyze this system. Here we will consider the robustness of the fuel distribution system under a few simplifications. Let our system be a graph such that:

   - Each edge in the graph is a pipe.
   - Each edge is directed (fuel flow is not bidirectional).
   - Each node in the graph is a combination refinery/distribution terminal (nodes both produce and consume fuel products, so they can have both incoming and outgoing edges).
   - Every refinery produces the same fuel that every distribution terminal accepts. In other words, we will not worry about dedicated pipelines for different products.

   Assume the system has n nodes and m edges. You would like the system to have stronger robustness by designing algorithms that verify if the system has certain properties:

   (a) Design an algorithm that verifies if the fuel distribution system has the following property:

   There is a way to transport fuel from any node in the graph to any other node in the graph. Your algorithm should run in $O(n + m)$.

   [**We are expecting:** A description of the algorithm, a brief explanation of why the algorithm works and an informal justification of the algorithm's running time.]

   (b) Design an algorithm that verifies if the fuel distribution system has the following property:

   No single pipe failure can isolate any node from either distributing fuel to the rest of the system or consuming fuel from the rest of the system. That is, removing a single edge cannot cause any node to be unreachable from any other node. You algorithm should run in $O(mn+m^2))$. You may assume an algorithm that satisfies the property in part (a) exists and use it directly.

   [**We are expecting:** A description of the algorithm, a brief explanation of why the algorithm works and an informal justification of the algorithm's running time.]