MATH 416, PROBLEM SET 5

Comments about homework.

- Solutions to homework should be written clearly, with justification, in complete sentences. Your solution should resemble something you'd write to teach another student in the class how to solve the problem.
- You are encouraged to work with other 416 students on the homework, but solutions must be written independently. Include a list of your collaborators at the top of your homework.
- You should submit your homework on Gradescope, indicating to Gradescope where the various pieces of your solutions are. The easiest (and recommended) way to do this is to start a new page for each problem.
- Attempting and struggling with problems is **critical** to learning mathematics. Do not search for published solutions to problems. I don't have to tell you that doing so constitutes academic dishonesty; it's also a terrible way to get better at math.

  If you get stuck, ask someone else for a hint. Better yet, go for a walk.

Problem 1.

(a) Prove the Handshake Lemma: in a graph, the sum of the degrees is twice the number of edges.

(b) Deduce that every graph has an even number of odd-degree vertices.

(c) Prove that if a graph G has exactly 2 vertices of odd degree, then there must be a path from one to the other.

Problem 2.

(a) Give an example of a connected graph with 8 vertices in which every vertex has degree 3. What is the length of the shortest cycle in your graph? the longest?

(b) Give an example of a connected graph with 10 vertices in which every vertex has degree 3. What is the length of the shortest cycle in your graph? the longest?

Problem 3. Consider the problem of taking a graph G and two vertices $s, t \in G$ and determining the number of shortest paths from s to t in G. Give a linear-time algorithm that solves this problem.

(*Hint:* Adapt BFS. Your algorithm should run in linear time if the graph is represented by an adjacency list.)

Problem 4. Prove that the following are equivalent for a graph $G = (V, E)$:

(a) G is a *tree*, i.e., a connected acyclic graph;

(b) G is connected and $|E| = |V| - 1$;

(c) G is acyclic and $|E| = |V| - 1$;

(d) for any vertices $x, y \in V$ there is a unique path from x to y.

(e) G has no cycles but for any $x, y \in V$ with $e = \{x, y\} \notin E$ the graph $G + e := (V, E \cup \{e\})$ has a cycle.

Problem 5. Show that every tree with at least 2 vertices must have a *leaf*, i.e., a vertex of degree 1.

Problem 6. Let G be a connected graph.

(a) Prove that if any edge e in G is part of a cycle, then removing e does not disconnect G.
(b) Using part (a), briefly describing an algorithm for finding a spanning tree of G that starts with the set of edges E and deletes edges until what remains is the edges of a spanning tree.

Problem 7. Let G be a strongly connected digraph and let T be the DFS tree obtained from a particular ordering of the vertices of G. Prove that, if all the forward edges (in the sense of the DFS tree T) are removed from G, then the resulting graph is still strongly connected.

Problem 8. You just discovered your best friend from elementary school on Twitbook. You both want to meet as soon as possible, but you live in two different cites that are far apart. To minimize travel time, you agree to meet at an intermediate city, and then you simultaneously hop in your cars and start driving toward each other. But where exactly should you meet?

You are given a weighted graph $G = (V, E)$, where the vertices V represent cities and the edges E represent roads that directly connect cities. Each edge e has a weight $w(e)$ equal to the time required to travel between the two cities. You are also given a vertex p, representing your starting location, and a vertex q, representing your friend's starting location. Describe and analyze an algorithm to find the target vertex t that allows you and your friend to meet as quickly as possible.

Problem 9.

(a) Give an example of a weighted graph with negative weights and a source vertex s on which Dijkstra's algorithm does not produce the minimum-weight path from s to some vertex v. In your example, there should be no negative-weight cycles (so that there *is* a min-weight path from s to v).
(b) Given a weighted graph, possibly with negative weights, you might try to add a sufficiently large number to all the weights so that they all become nonnegative and then run Dijkstra's algorithm on the modified graph. Does this successfully find min-weight paths (in the original graph)? Either prove that it works or give a counterexample.