Ve281 Data Structures and Algorithms
Written Assignment Seven

This assignment is announced on Nov. 29th, 2018. It is due by 5:40 pm on Dec. 7th, 2018. The assignment consists of 5 problems. For those problems that ask you to design algorithms, you can either describe your algorithms in plain English or write pseudo-code. If you choose to write pseudo-code, you should write in a way that can be easily understood. Otherwise, you will get a zero for the problem.

1. (10%) Figure 1 shows a weighted undirected graph. Show the adjacency matrix representation of the graph.

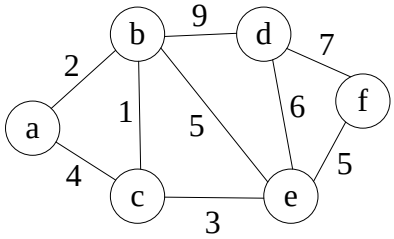

Figure 1: A weighted undirected graph.

2. (25%) Apply Prim's algorithm to the graph shown in Figure 1 to obtain its minimum spanning tree. Suppose the first node your pick is node a. Show the intermediate steps of applying the algorithm. Draw the final minimum spanning tree.

3. (20%) Suppose that you are given a directed acyclic graph $G = (V, E)$ with real-valued edge weights and two distinct nodes s and d. Describe an algorithm for finding a longest weighted simple path from s to d. For example, for the graph shown in Figure 2, the longest path from node A to node C should be $A \to B \to F \to C$. If there is no path exists between the two nodes, your algorithm just tells so. What is the efficiency of your algorithm? (Hint: consider topological sorting on the DAG.)

4. (20%) You are given a directed graph $G = (V, E)$ on which each edge $(u, v) \in E$ has an associated value $r(u, v)$, which is a real number in the range $0 \leq r(u, v) \leq 1$ that represents the reliability of a communication channel from vertex u to vertex v. We interpret $r(u, v)$ as the probability that the channel from u to v will not fail, and we

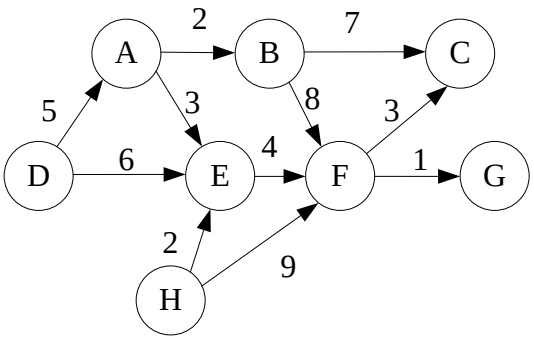

Figure 2: A weighted directed graph.

assume that these probabilities are independent. Give an efficient algorithm to find the most reliable path between two given vertices. (Hint: consider the shortest path algorithm.)

5. (25%) Let $G = (V, E)$ be a connected, undirected graph. Give an $O(|V| + |E|)$-time algorithm to compute a path in G that traverses each edge in E **exactly once in each direction**. For example, for the graph shown in Figure 3, one path satisfying the requirement is

$$A \to B \to C \to D \to C \to A \to C \to B \to A$$

Note that in the above path, each edge is visited exactly once in each direction. (Hint: consider the graph search algorithm.)

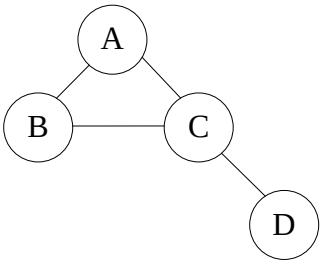

Figure 3: An undirected graph.