Ve281 Data Structures and Algorithms
Written Assignment Six

This assignment is announced on Nov. 30th, 2017. It is due by 5:40 pm on Dec. 8th, 2017. The assignment consists of 7 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. Apply Kruskal's algorithm to the graph shown in Figure 1 to obtain its minimum spanning tree. Show the intermediate steps of applying the algorithm. Draw the final minimum spanning tree.

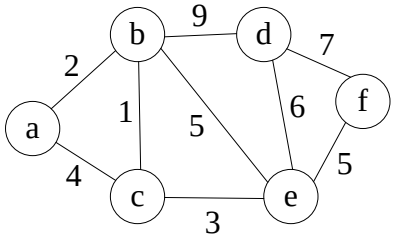

Figure 1: A weighted undirected graph.

2. 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.)

3. 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 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.)

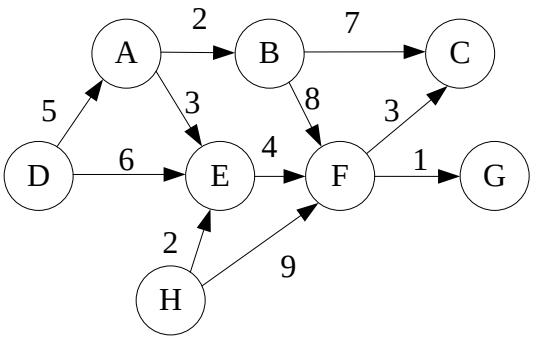

Figure 2: A weighted directed graph.

4. 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.)

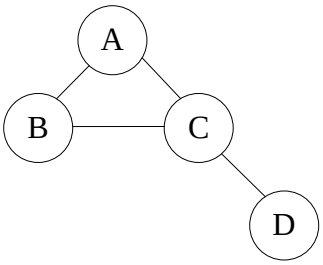

Figure 3: An undirected graph.

5. Apply the Bellman-Ford algorithm to the graph shown in Figure 4 to obtain the shortest path from node a to every other node in the graph. Show the intermediate result. You should run the full version of the Bellman-Ford algorithm (i.e., including the last iteration that checks the existence of a negative cycle). If there is a negative cycle, report so. When running the algorithm, besides keeping track of the shortest path length, you should also keep track of the predecessor on the shortest path. If there is no negative cycle, please write the final shortest path and its length for each destination node.

6. Prove the following early stopping criterion for the Bellman-Ford algorithm. For a given graph $G = (V, E)$ and a source vertex s, if you apply the Bellman-Ford algorithm and find that that for a $j < |V| - 1$, $L(v, j) = L(v, j - 1)$ for all vertices $v \in V$, then you can stop and claim that there is no negative cycle and the length of the shortest s-v path is $L(v, j)$ for all $v \in V$. Here $L(v, i)$ is the length of the shortest s-v path with at most i edges.

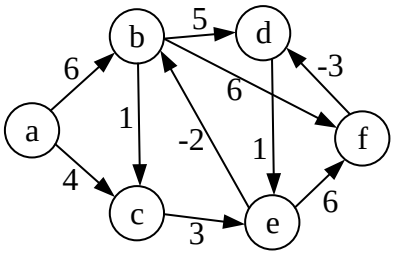

Figure 4: A weighted directed graph.

7. Consider the problem of neatly printing a paragraph with a monospaced font (all characters having the same width) on a printer. The input text is a sequence of n words of lengths $l_1, l_2, \ldots, l_n$, measured in characters. We want to print this paragraph neatly on a number of lines that hold a maximum of M characters each. Our criterion of "neatness" is as follows. If a given line contains words i through j, where $i \leq j$, and we leave exactly one space between words, the number of extra space characters at the end of the line is $M - j + i - \sum_{k=i}^{j} l_k$, which must be nonnegative so that the words fit on the line. We wish to minimize the sum, over all lines except the last, of the cubes of the numbers of extra space characters at the ends of lines. Give a **dynamic-programming** algorithm to print a paragraph of n words neatly on a printer. What is the running time of your algorithm?