Fall 2018, CMPSC 465: Homework 5/Exam 5 practice solutions.

1. We are given a directed graph $G = (V, E)$ on which each edge $\langle u, v \rangle$ has an associated value $r(u, v)$, which is a real number in the range $(0, 1)$ 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.
   (**10 points**)
   Solution:
   The reliability of a single edge is a probability value. So, the closer it is to 1, the better. Now consider a path with multiple edges e_1, e_2, etc. Since the probabilities are assumed to be independent, the probability that the path is reliable and will not fail is the product of probabilities of the individual edges. We want to maximize this product, since the closer it is to 1, the better. We use the same trick as Exercise 4.21 (discussed in recitation), and use the negative log of the probability values as the edge weights. Notice that these values will all be positive, because the probabilities are given to be in $(0, 1)$. So we can apply Dijkstra's algorithm in this case.

   Grading scheme: 2 points for attempt + 2 points for observation that the probabilities are non-negative + 2 points for observation that we want to maximum product + 2 points for observation on negative log of product + 2 points for saying we can apply Dijkstra's algorithm.

2. You are given a directed graph $G(V, E)$ with (possibly negative) weighted edges, along with a specific node $s \in V$ and a tree $T = (V, E'), E' \subseteq E$. Give an algorithm that checks whether T is a shortest-path tree for G with starting point s. Your algorithm should run in linear time. (**7 points**)

   Solution:

   See solutions to exercise 4.7 (refer to Lecture 27 notes).

   Grading scheme: 2 points for attempt + 2 points for introducing appropriate notation + 1/2/3 points for clear solution.

3. Draw a simple directed weighted graph G with 8 vertices and 16 edges, such that G contains a minimum-weight cycle with at least 4 edges. Show that the Bellman-Ford algorithm will find this cycle.
(**8 points**)
Solution:
Detecting a minimum weight cycle suggests we create a graph with a negative cycle. Assuming the length of the longest shortest path in the graph is k edges, if we perform one extra iteration of Bellman-Ford (i.e., $k + 1$ iterations in all) and we see a reduction in distance labels, that means there is a negative cycle in the graph.

Consider a negative cycle of 5 edges (and 5 vertices). We need to introduce 11 more edges and 3 more vertices. There are many ways of doing this without introducing another negative weight cycle. Let us perform 8 iterations of Bellman-Ford. The distance labels will change and this indicates the presence of a minimum-weight cycle.

Grading scheme: 2 points for attempt + 3 points for explaining how Bellman-Ford is applicable + 3 points for valid graph as example.

4. Design an efficient algorithm for finding the longest shortest path connecting any two vertices in an acyclic weighted directed graph. Also, analyze the time complexity of your algorithm. (**8 points**)

   Solution:

   Perform a topological sort of the DAG. Suppose there are k source vertices, i.e., vertices with no incoming edges. All source vertices can be identified in $O(n)$ time. Use the linear-time shortest paths algorithm to compute the distance labels for each vertex, for each source vertex. In this linear-time algorithm, we inspect vertices in topological sort order, and for all adjacencies of the vertex, we perform a relax operation. Once we are done processing a vertex in topological sort order, the vertex is guaranteed to have the correct label. Simultaneously keep track of the longest finite-length shortest path length after processing each vertex. Since we assume k source vertices, this algorithm will take $\mathrm{O}(k(m+n))$ time. Because k can be $\mathrm{O}(n)$, the running time is $\mathrm{O}(mn)$.

   Grading scheme: 1 point for attempt + 1 point for mentioning topological sort + 4 points for mentioning shortest paths from each vertex + 2 points for running time analysis.

5. Explain why there are no non-tree forward edges in a BFS tree constructed for a directed graph. (**7 points**)

   Solution:

   Let us assume that there is a non-tree forward edge $\langle u, v \rangle$. Since this is a forward edge, $d(v) > d(u)$. The only possibility is for $d(v)$ to be $d(u) + 1$. That would not be possible because $\langle u, v \rangle$ is not a tree edge. So we cannot have non-tree forward edges when performing BFS on a directed graph.

   Grading scheme: 2 points for attempt + 3 points for approach in the right direction + 2 points for short and clear proof.