1. (10 pts total) First, review the material on depth-first spanning forests in Chapter 22.3 of the textbook. Then, consider the directed graph G defined by the edge list

   $G = \{(1,2),(1,4),(1,8),(2,3),(3,1),(4,8),(5,2),(5,6),(6,2),(6,3),(6,5),(7,4),(8,7)\}$

   (a) (5 pts) Draw the depth-first spanning forest, including and identifying all tree, back, forward, and cross edges. (If you prefer, you can identify the forward, back, and cross edges in separate lists instead of trying to draw and label them.)

   (b) (5 pts) List all the strong components in G.

2. (30 pts total) On an overnight camping trip at the Weathertop National Park you and your hobbit friend Samwise are woken from a restless sleep by a scream. Crawling out of your tent to investigate, you see a terrified park ranger running out of the woods, covered in blood and clutching a crumpled piece of paper to his chest. Reaching your tent, he gasps "Get out... while... you...", thrusts the paper into your hands and falls to the ground, dead.

   Looking at the crumpled paper, you recognize a map of the park, drawn as an undirected graph, where vertices represent landmarks in the park, and edges represent trails between those landmarks. (Trails start and end at landmarks and do not cross.) Coincidentally, you recognize one of the vertices as your current location; several vertices on the boundary of the map are labeled EXIT.

   On closer examination, you notice that someone (perhaps the dead ranger) has written a real number between 0 and 1 next to each vertex and each edge. A scrawled note on the back of the map indicates that a number next to a vertex or edge is the probability of encountering a deadly ringwraith along the corresponding trail or landmark. The note warns you that stepping off the marked trails will surely result in death.

   You glance down at the corpse at your feet. His death certainly looked painful. On closer examination, you realize that the ranger is not dead at all, or rather, is turning into a wraith who will surely devour you. After burning the undead ranger's body, you wisely decide to leave the park immediately.

   (a) (5 pts) Give a (small!) example G such that the path from your current location to the EXIT node that minimizes the *expected number* of encountered ringwraiths is different from the path that minimizes the *probability of encountering any* ringwraiths at all. Explain why, in general, these two criteria lead to different answers.

   (b) (15 pts) Describe and analyze an efficient algorithm to find a path from your current location to an arbitrary EXIT node, such that the *total expected number*

of ringwraiths encountered along the path is as small as possible. Be sure to account for both the vertex probabilities and the edge probabilities.

Gandalf's hint: This is clearly an SSSP problem, but you must identify how to reduce the input G to a form that can be solved by SSSP. Remember to include the cost of this transformation in your running-time analysis.

(c) (10 pts) Describe and analyze an efficient algorithm to find a path from your current location to an arbitrary EXIT node, such that the *probability of encountering any* ringwraiths at all is minimized.

3. (30 pts total) In a late-night algorithms study session, you and Golum argue about the conditions under which a minimum spanning tree is unique. You agree that if all edges in G have unique weights the MST is also unique, but you disagree about how to relax this assumption. Let $w(e)$ be a function that returns the weight of some $e \in E$.

   (a) (5 pts) Give an example of a (small!) weighted graph that has both a unique MST and some e and e' such that $w(e) = w(e') = x$.

   (b) (5 pts) Golum claims that the following is true. Prove via (small!) counter examples that it is false.

   Golum's Claim: G has a unique MST if and only if (i) for any partition of the vertices of G into two subsets, the minimum-weight edge with one endpoint in each subset is unique, and (ii) the maximum-weight edge in any cycle of G is unique.

   (c) (10 pts) Golum now demands that you produce the correct relaxed condition, which you claim is the following. Prove that you are correct.

   Your Claim: an edge-weighted graph G has a unique *MST T_{mst} if and only if the following conditions hold:*
   (i) for any bipartition of the vertices induced by removing some edge $e \in T_{\text{mst}}$, the minimum-weight edge with one endpoint in each subset is unique, and
   (ii) the maximum-weight edge of any cycle constructed by adding one edge f to T_{mst}, where $f \notin T_{\text{mst}}$, is unique.

   Gandalf's hint: Note that for any spanning tree T on G, removing some edge $e \in T$ induces a bipartition of the vertices. Consider the edges that span this cut.

   (d) (10 pts) Describe and analyze an algorithm that will determine whether an input graph G has a unique MST in (effectively) $O(E \lg V)$ time.

   Gandalf's hint: Think about Kruskal's algorithm.

4. (30 pts total) Implement Dijkstra's algorithm for the SSSP problem using a binary min-heap, and show (in a nicely structured figure) that its running time is $O(E \lg V)$ on the following type of graphs.

   Recall from problem set 7 that $G(n,p)$ denotes a simple (undirected, unweighted, no self-loops) random graph with n vertices in which each unordered pair (u,v), where $u \neq v$, is connected with probability $p = c/(n-1)$, for a constant expected degree c. Let $G_d(n,p)$ denote a directed generalization of this model in which we relax the undirected requirement by letting each *ordered* pair (u,v), where $u \neq v$, be connected with probability p. That is, instead of flipping $\binom{n}{2}$ coins for the upper triangle of the adjacency matrix, we flip $n^2 - n$ coins (because we still prohibit self loops) for the upper and lower triangles. Let $G_d(n,p)$ with $c = 5$ define the input graph for Dijkstra's algorithm.

   The deliverables here are (i) a nice figure plotting the running time $T(n)$ versus input graph size $\log_{10} n$, including trend lines showing the expected running time, and (ii) your code. No credit if you do not submit your code as part of your solutions.

   Gandalf's hints: Because $G_d(n,p)$ is a random variable, you will want to average the running time $T(n)$ for a particular choice of n over several draws from the model. Also, $G_d(n,p)$ for this choice of p will contain one very large component and a few very small components; make sure your implementation of Dijkstra chooses a source vertex in the large component. And, as always, you will want a sufficiently broad range of n to properly show the trend, e.g., $n \in [10^1, 10^3]$.

5. (10 pts extra credit) Currency arbitrage is a form of financial trading that uses discrepancies in foreign currency exchange rates to transform one unit of some currency into more than one unit of the same currency. For instance, suppose 1 U.S. dollar bought 0.82 Euro, 1 Euro bought 129.7 Japanese Yen, 1 Japanese Yen bought 12 Turkish Lira and one Turkish Lira bought 0.0008 U.S. dollars. Then, by converting currencies, a trader could start with 1 U.S. dollar and buy $0.82 \times 129.7 \times 12 \times 0.0008 \approx 1.02$ U.S. dollars, thus turning a 2% profit. Of course, this is not how real currency markets work because each transaction must pay a commission to a middle-man for making the deal.

   Suppose that we are given n currencies $c_1, c_2, \ldots, c_n$ and an $n \times n$ table R of exchange rates, such that one unit of currency c_i buys $R[i,j]$ units of currency c_j. A traditional arbitrage opportunity is thus a cycle in the induced graph such that the product of the edge weights is greater than unity. That is, a sequence of currencies $\langle c_{i_1}, c_{i_2}, \ldots, c_{i_k} \rangle$

such that $R[i_1, i_2] \times R[i_2, i_3] \times \cdots \times R[i_{k-1}, i_k] \times R[i_k, i_1] > 1$. Each transaction, however, must pay a commission, which is typically some α fraction of the transaction value, e.g., $\alpha = 0.01$ for a 1% rate.

(a) (5 pts extra credit) Give an efficient algorithm to determine whether or not there exists such an arbitrage opportunity, given a commission rate α. Analyze the running time of your algorithm.

Gandalf's hint: It is possible to solve this problem in $O(n^3)$. Recall that Bellman-Ford can be used to detect negative-weight cycles in a graph.

(b) (5 pts extra credit) Explain what effect varying α has on the structure of the set of possible arbitrage opportunities your algorithm might identify.