1. (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.

   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.

2. (10 pts total) For 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.)

   Hint: remember that the ordering of the vertices matters.

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

3. (25 pts total) The hobbits of The Shire have devised an ingenious new way to get around the valley, via "busses." Their system uses horse-drawn wagons that carry people from one place to another. Each wagon (a "bus") stops only at a predefined set of locations and at predefined times; a set of busses that all follow the same such schedule is called a "line." Being so popular, there are now many bus lines crisscrossing The Shire. One evening, after a great party, you decide to take a bus home. Thinking ahead, your friend Samwise wisely wrote down a list of all the times and locations of every stop of every bus in The Shire. Unfortunately, there is no bus that visits both the party and your home, so you'll need to transfer between bus lines at least once.

   (a) (15 pts) Describe and analyze an algorithm to determine the sequence of bus rides that will get you and Sam home as early as possible, assuming there are b different bus lines, and each bus line has n stops per day. Your goal is to minimize your *arrival time*, not the time you actually spend traveling. Assume that buses run exactly on schedule, that you have an accurate watch, and that you are too tired to walk between bus stops. Express the running time in terms of b and n.

   Gandalf's hint: Convert the bus schedule into a graph in which each bus stop is represented by a single vertex. To express your solution in terms of b and n, think about the largest number of vertices or edges that could be produced by a single bus line with n stops per day.

   (b) (10 pts) Briefly discuss how your algorithm could be adapted to mimic the behavior of `hopstop.com`, a popular website for transportation directions, including bus, subway, train and walking directions, in certain major metro areas. Comment on the graph representation of the metro area, how your algorithm might work on such a data structure, and what modifications you might need to make.

4. (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.

5. (15 pts) Implement an efficient algorithm that computes the *diameter* of an unweighted, undirected input graph $G = (V, E)$, when G is given in adjacency list format, and which takes $O(V + E)$ additional space. Write a paragraph explaining the algorithm, analyze its running time, and prove that it satisfies the space requirement.

No credit if you don't submit your code at *the end* of your solutions.

6. (15 pts total) Use your implementation from 5 to conduct the following numerical experiment.

An Erdös-Rényi random graph, denoted $G(n,p)$, is a graph with n vertices and where each pairwise connection (u,v) for $u \neq v$ exists independently with probability p (edges are unweighted, undirected and self-loops prohibited). If each vertex has expected degree $E[k] = c$, then $p = c/(n-1)$.

(a) (7 pts) For $c = 5$, show numerically that the expected diameter increases like $O(\log n)$. Because $G(n,p)$ is a random variable, you should average the diameter at a given value of n over several independent realizations (e.g., > 10). (One figure.)

(b) (8 pts) Show that the running time and space requirements also follow your analytic prediction. (Two figures.)

Gandalf's hints:
(i) If two vertices are not reachable, their distance is ∞; exclude such distances from your diameter calculation. (Because $G(n,p)$ is connected only probabilistically, at $c = 5$ there will sometimes be a few small, disconnected components. You may simplify your calculation by only analyzing the largest component of the input graph.)
(ii) As before, you'll need to implement some measurement code within your `diameter()` function that counts the number of atomic operations it performs. This time, you'll also need to implement code to measure the amount of space it uses.
(iii) You'll also need to write a function `Random-Graph(n,p)` that returns an instance of $G(n,p)$ as an adjacency list. Note that this takes $\Theta(V^2)$ time since you need to flip a coin for each of the n-choose-2 possible connections. Do not count this time toward the diameter calculation running time.

7. (**25 pts extra credit**) Let A be the adjacency matrix for a directed multigraph with n vertices, i.e., each A_{ij} is a non-negative integer counting the number of connections $i \to j$. Let $h : V \to \{1,\ldots,k\}$ denote a *block assignment* vector such that $h(i)$ returns the group or block label of the ith vertex, where there are k possibly blocks. Given some choice h and the input A, we may write down a *block matrix* B where $B_{ab} = \sum_{i\in a, j\in b} A_{ij}$. That is, the element B_{ab} counts the number of edges with one endpoint in group a and one endpoint in group b. For simplicity, assume that the size of each group is fixed at c members (which implies $k = n/c$).

The *score* f of a block assignment h is computed as the sum of the upper triangle of B plus the sum of the diagonal of B minus the sum of the lower triangle of B;

mathematically:

$$f(h) = \sum_{b \geq a}^{k} B_{ab} - \sum_{b < a}^{k} B_{ab} \ .$$

(a) (**7 pts extra credit**) Golum claims the following procedure will maximize f. First, compute the in- and out-degrees of each vertex; second, sort them perhaps in (i) descending order of out-degree k^{out}, or (ii) descending order of $k^{\text{out}} - k^{\text{in}}$; third, label the first c nodes as belonging to the first block in h, the second c nodes as the second block, etc.

Give a counter example for each version of Golum's claim.

(b) (**18 pts extra credit**) Describe and analyze an algorithm to efficiently find the h that maximizes f.

A polynomial-time algorithm will receive full credit; an exponential-time algorithm will receive almost-full credit; no credit will be given for the brute force algorithm that simply explores all possible assignments of vertices to blocks (which takes $O(n!)$ time). A proof that the brute-force approach is the only solution will also receive full credit.

8. (**20 pts extra credit**) The dice game Yahtzee is played with five 6-sided dice; the highest scoring roll is five-of-a-kind and is called a "yahtzee". Each round, each player is allowed to roll each of the five dice at most three times. The probability of getting a yahtzee all at once is $6 \cdot 1/6^5 = 1/1296$, but because players are allowed to re-roll dice, the probability of getting a yahtzee is much larger.

(a) (**10 pts extra credit**) What is the probability of getting a yahtzee? Show your work.

Gandalf's hint: Assume that the player is trying to maximize the probability of getting a yahtzee, and note that different intermediate states lead to different decisions about which dice to re-roll.

(b) (**10 pts extra credit**) Suppose we generalize the game to n 6-sided dice, with each die being rolled at most $n-2$ times, and a "yahtzee" being n-of-a-kind. Now what is the probability of a yahtzee?

Gandalf's hint: Determine this numerically, via direct simulation of the dice throwing process, as a function of n, and make a nice figure. (As far as I know, no analytic solution exists.)