1. (10 pts) Describe and analyze an algorithm that produces a uniformly random permutation of n arbitrarily-valued integers in $O(n)$ time and $O(1)$ additional space. Assume the existence of a function `random()` that returns, in $O(1)$ time, a real value distributed uniformly at random on the unit interval.

   Note: A full credit solution must include (i) a verbal description of the algorithm (possibly including pseudocode) and (ii) a mathematical argument (a "proof") that it returns the correct output on every input and meets the time and space requirements. This is what is meant by "describe and analyze an algorithm."

2. (30 pts total) In the MST lecture, we assumed that all edges in the input graph G had distinct weights, which implied uniqueness of the minimum spanning tree. However, only a weaker condition on the input weights is necessary to imply MST uniqueness. Let $w(e)$ be a function that returns the weight of the edge $e \in E$.

   (a) (5 pts) Give an example of a (small!) weighted graph that has both a unique MST and two edges with equal weights x.

   (b) (5 pts) Prove, via (small!) counter examples, that the following claim is false. G has a unique MST if and only if the following conditions hold:

   - for any partition of the vertices of G into two subsets, the minimum-weight edge with one endpoint in each subset is unique, and
   - the maximum-weight edge in any cycle of G is unique.

   (c) (10 pts) Prove that an edge-weighted graph G has a *unique* MST T_{mst} if and only if the following conditions hold:

   - 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.
   - The maximum-weight edge of any cycle constructed by adding one edge f to T_{mst}, where $f \notin T_{\text{mst}}$, is unique.

   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.

   Hint: Think about Kruskal's algorithm.

3. (10 pts total) For the following graph, defined by this (directed) 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.

4. (25 pts total) After a grueling algorithms class, you decide to take the bus home. Since you planned ahead, you have a schedule that lists the times and locations of every stop of every bus in the Boulder/Denver metro-area. Unfortunately, there is no bus that visits both your class building and your home; you must 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 home as early as possible, assuming there are b different bus lines, and each bus stops n times per day. Your goal is to minimize your *arrival time*, not the time you actually spend traveling. Assume that the 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.

   Hint: Convert the bus schedule into a graph.

   (b) (10 pts) Comment briefly on how this algorithm could be adapted to mimic the behavior of `hopstop.com`, a popular website for transportation directions in certain major metro areas that include bus, subway, train and walking directions. Be sure to mention any modifications and their impact on performance.

5. (20 pts total) On an overnight camping trip in Mystic Falls State Park, you 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 park 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 vampire 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. Wait, was that a twitch? Are his teeth getting longer? After driving a wooden tent stake through the undead ranger's heart, you wisely decide to leave the park immediately.

(a) (5 pts) Give a (small!) example of this problem where the path from your current location to the EXIT node that minimizes the expected number of encountered vampires is different from the path that minimizes the probability of encountering any vampires 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 vampires encountered along the path is as small as possible. Be sure to account for both the vertex probabilities and the edge probabilities.

Note: Assume "efficient" means your algorithm must run as fast as possible.

6. (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.

7. (15 pts total) Use your implementation from 6 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.)

Hint 1: 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 could simplify your calculation by only analyzing the largest component of the input graph.)

Hint 2: 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.

Hint 3: 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.

8. (25 pts *extra credit*) You are given an $n \times n$ adjacency matrix A that represents a multigraph, i.e., A_{ij} is a non-negative integer counting the number of connections $i \to j$. Let h denote a *block assignment* vector such that $h(i)$ returns the group or block label of the ith vertex. If k denotes the number of such groups, h implies a $k \times k$ sized *block matrix* B where $B_{ab} = \sum_{i \in a, j \in b} A_{ij}$. That is, 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 (thus, $k = n/c$).

Describe and analyze an algorithm to efficiently find the h that maximizes the following function

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

In words, f is 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.

9. (20 pts total *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) What is the probability of getting a yahtzee? Show your work.

   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) 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?

   Hint: Determine this numerically, as a function of n, and make a nice figure. (As far as I know, no analytic solution exists.)