Problem 1. Let $G = (V, E)$ be a directed graph.

(a) Assuming that $G$ is represented by an adjacency matrix $A[1..n, 1..n]$, give a $\Theta(n^2)$-time algorithm to compute the adjacency list representation of $G$. (Represent the addition of an element $v$ to a list $l$ using pseudocode by $l \leftarrow l \cup \{v\}$.)

(b) Assuming that $G$ is represented by an adjacency list $\text{Adj}[1..n]$, give a $\Theta(n^2)$-time algorithm to compute the adjacency matrix of $G$.

Problem 2. Let $G = (V, E, p)$ be a directed graph representing a network of roads between cities. The weight $p(e)$ is the probability that road $e$ will be open, so that $0 \leq p(e) \leq 1$. The probabilities are assumed to be independent. You want to take a trip from city $a$ to city $b$. Give an algorithm to find the route that has the most chance of being open.

Problem 3. Assume that in a group of people everyone is friends with at least half of the people. (So, for example, in a group of three or four people everyone is friends with at least two people. Friendship symmetric but not reflexive.) It turns out that in this situation the group can be seated around a table so that everyone is seated next to friends on both sides.

(a) Reword this as a statement about graphs.

(b) Prove your statement.

(c) Give an efficient algorithm for finding such a seating. Analyze its running time.

Problem 4. A simple cycle is a cycle in a graph such that no vertex (on the cycle) is visited more than once. The optimization version of the Weighted Longest Simple Cycle Problem is, given an undirected, weighted graph $G = (V, E)$, find a simple cycle whose sum of the weights on the edges is as large as possible. You can assume that the weights are integers between 0 and $|V|$.

(a) What is the decision version of the Longest Simple Cycle Problem?

(b) Show that the decision version of the Longest Simple Cycle Problem is in $\text{NP}$. What is the certificate?

(c) Show that if you can solve the optimization problem in polynomial time, then you can solve the decision problem in polynomial time.

(d) Show that if you can solve the decision problem in polynomial time, then you can solve the optimization problem in polynomial time. HINT: First find the weight of the cycle.