1. (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 $\mathrm{Adj}[1..n]$, give a $\Theta(n^2)$-time algorithm to compute the adjacency matrix of G.

2. Let $= (V, E)$ be a directed, weighted graph. Show that Dijkstra's algorithm does not work if there are negative weight edges but no negative weight cycles. Make your counterexample as simple as possible.

3. (a) Show how to modify Dijkstra's algorithm to solve the single source shortest path problem if there is exactly one negative weight edge but no negative weight cycles.

   (b) Justify the correctness of your algorithm.

   (c) How efficient is your algorithm? You should be able to answer this briefly without much detail.

4. The *decision version* of the Traveling Salesman Problem (TSP) is given an weighted, undirected graph $G = (V, E)$ and a target T, does G have a tour (i.e., a cycle visiting every edge vertex exactly once) of total weight at most T?

   (a) Show that the decision version of TSP is in **NP**. Make sure to state what the certificate is, and to show that the verification is in polynomial time.

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

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