Problem 1. [10 points - No help question]
In your own words, explain the worst-case running time complexity of Dijkstra’s algorithm. Briefly explain what data structures would you use to implement the algorithm, and how they affect the running time complexity of the algorithm. Note: You don’t need to rewrite the algorithm. Just mention the data structures and how they’d be used within the algorithm.

Problem 2. [15 points]
Suppose you are given a connected graph G, with edge costs that are all distinct. Prove that G has a unique minimum spanning tree.

Problem 3. [10 points]
For the following statement, decide whether it is true or false. If it is true, give a short explanation. If it is false, give a counterexample.

Suppose we are given an instance of the Shortest s-t Path Problem on a directed graph G. We assume that all edge costs are positive and distinct. Let P be a minimum-cost s-t path for this instance. Now suppose we replace each edge cost $c_e$ by its square, $c_e^2$, thereby creating a new instance of the problem with the same graph but different costs.

True or false? P must still be a minimum-cost s-t path for this new instance.

Problem 4. [20 points - Proof Problem]
A small business—say, a photocopying service with a single large machine—faces the following scheduling problem. Each morning they get a set of jobs from customers. They want to do the jobs on their single machine in an order that keeps their customers happiest. Customer $i$’s job will take $t_i$ time to complete. Given a schedule (i.e., an ordering of the jobs), let $C_i$ denote the finishing time of job $i$. For example, if job $j$ is the first to be done, we would have $C_j = t_j$; and if job $j$ is done right after job $i$, we would have $C_j = C_i + t_j$. Each customer $i$ also has a given weight $w_i$ that represents his or her importance to the business. The happiness of customer $i$ is expected to be dependent on the finishing time of $i$’s job. So the company decides that they want to order the jobs to minimize the weighted sum of the completion times, $\sum_{i=1}^{n} w_i C_i$.

(a) Design an efficient algorithm to solve this problem. That is, you are given a set of $n$ jobs with a processing time $t_i$ and a weight $w_i$ for each job. You want to order the jobs so as to minimize the weighted sum of the completion times, $\sum_{i=1}^{n} w_i C_i$.

(b) [Proof module - Please submit separately] Prove why it solves the problem efficiently.

Example. Suppose there are two jobs: the first takes time $t_1 = 1$ and has weight $w_1 = 10$, while the second job takes time $t_2 = 3$ and has weight $w_2 = 2$. Then doing job 1 first would yield a weighted completion time of $10 \cdot 1 + 2 \cdot 4 = 18$, while doing the second job first would yield the larger weighted completion time of $10 \cdot 4 + 2 \cdot 3 = 46$.

Problem 5. [25 points]
Your friends are planning an expedition to a small town deep in the Canadian north next winter break. They’ve researched all the travel options and have drawn up a directed graph whose nodes represent intermediate destinations and edges represent the roads between them.

In the course of this, they’ve also learned that extreme weather causes roads in this part of the world to become quite slow in the winter and may cause large travel delays. They’ve found an excellent travel Web site that can accurately predict how fast they’ll be able to travel along the roads; however, the speed of travel depends on the time of year. More precisely, the Web site answers queries of the following form: given an
edge $e = (v, w)$ connecting two sites $v$ and $w$, and given a proposed starting time $t$ from location $v$, the site will return a value $f_e(t)$, the predicted arrival time at $w$. The Web site guarantees that $f_e(t) \geq t$ for all edges $e$ and all times $t$ (you can’t travel backward in time), and that $f_e(t)$ is a monotone increasing function of $t$ (that is, you do not arrive earlier by starting later). Other than that, the functions $f_e(t)$ may be arbitrary.

For example, in areas where the travel time does not vary with the season, we would have $f_e(t) = t + l_e$, where $l_e$ is the time needed to travel from the beginning to the end of edge $e$.

Your friends want to use the Web site to determine the fastest way to travel through the directed graph from their starting point to their intended destination. (You should assume that they start at time 0, and that all predictions made by the Web site are completely correct.) Give a polynomial-time algorithm to do this, where we treat a single query to the Web site (based on a specific edge $e$ and a time $t$) as taking a single computational step. Show (no need for proof) why it’s correct.

Problem 6. [20 points]
Remember your algorithm from last assignment to find a cycle in a graph with complexity of $O(n+m)$. In this problem, give an algorithm that determines whether or not a given undirected graph $G = (V, E)$ contains a cycle. Your algorithm should run in $O(n)$ time only, independent of $m$.

(a) Provide pseudocode for your algorithm.
(b) Justify your algorithm’s running time.