**Problem 1.** You are working at a restaurant and have $n$ customers waiting to be served. Unfortunately, you only have one table, so you can only serve one customer at a time. Each customer $i$ takes $t_i$ minutes to finish eating, at which point you can seat another customer. All of the customers arrived at exactly the same time, so you plan to seat them to minimize the total time spent in the restaurant. For example, if the customers are numbered $1, \ldots, n$ and you seat them in this order, then customer $i$ spends $\sum_{j=1}^{i-1} t_j$ time in the restaurant, because she first waits for everyone before her to finish eating and then has to finish her own meal. We wish to minimize the total time spent in the restaurant, which is,

$$T = \sum_{i=1}^{n} (\text{time spent by customer } i)$$

(a) Design an efficient algorithm that produces an ordering that minimizes $T$.

(b) Prove your algorithm is correct.

**Problem 2.** Given a list of $n$ natural numbers $d_1, d_2, \ldots, d_n$, show how to decide in polynomial time whether there exists an undirected graph $G = (V, E)$ whose node degrees are precisely the numbers $d_1, d_2, \ldots, d_n$. (That is, if $V = \{v_1, v_2, \ldots, v_n\}$, then the degree of $v_i$ should be exactly $d_i$.) $G$ should not contain multiple edges between the same pair of nodes, or “loop” edges with both endpoints equal to the same node.

(a) Provide a pseudocode.

(b) Explain what is the idea behind your algorithm (no need to write a formal proof).

(c) Explain why your algorithm is polynomial.

**Problem 3.** Let $G = (V, E)$ be a directed graph with edge lengths $\ell_e$, for each $e \in E$, and assume that all edge costs are positive and distinct. Let $P$ be a minimum-cost $s-t$ path for this instance. For each of the following two statements, decide whether it is true or false. If it is true, give a short explanation. If it is false, give a counterexample.

(a) Suppose we add 1 to the cost of each edge $c_e$. In this new instance of the problem, $P$ must still be a minimum-cost $s-t$ path.

(b) Suppose we replace $\ell_e$ by its square, $\ell_e^2$. In this new instance of the problem, $P$ must still be a minimum-cost $s-t$ path.
Problem 4. Consider the following undirected graph $G$:

(a) Draw the minimum spanning tree that is induced by performing Kruskal’s algorithm starting at node $d$. Indicate clearly the order in which edges are chosen.

(b) Draw the minimum spanning tree that is induced by performing Prim’s algorithm starting at node $d$. Indicate clearly the order in which edges are chosen.

Proof Problem

\LaTeX source for template: https://www.overleaf.com/read/jpxxhlnhtgnkn

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 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 customer $i$’s job. So the company decides that they want to order the jobs to minimize the weighted sum of the completion times, $\sum_{i=1}^{n} w_iC_i$.

For example, suppose there are two jobs to be done: 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 $w_2 = 2$. Doing job 1 first would yield a weighted completion time of $10 \cdot 1 + 2 \cdot 4 = 18$, while doing job 2 first would yield the larger weighted completion time of $10 \cdot 4 + 2 \cdot 3 = 46$. Therefore, it is better to schedule job 1 first.

(a) Design an 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 $\sum_{i=1}^{n} w_iC_i$.

(b) Explain why your algorithm is efficient and prove it is correct.