Instructions:
Include pseudo-code for your algorithms, a run-time analysis and a proof of correctness. In your pseudo-code you can use subroutines like sorting, depth-first-search, etc. without providing details.
Also, read these instructions carefully: [http://courses.cs.washington.edu/courses/cse417/16wi/grading.html](http://courses.cs.washington.edu/courses/cse417/16wi/grading.html)

Turn your homework in on canvas: Problems 1 and 3 in one file, and problem 2 in the other.

1. You are given a sequence $S$ of $n$ purchases at a stock exchange, possibly containing some events multiple times. For example, $S$ might be

   Buy Amazon, Buy Google, Buy eBay, Buy Google, Buy Google, Buy Oracle.

   You are also given another sequence $S'$ of $m$ purchases $m < n$. Show how to determine if $S'$ is a subsequence of $S$ in linear time.
   For example, if $S'$ is

   Buy Amazon, Buy eBay, Buy Google

   then $S'$ is a subsequence of $S$, whereas if $S'$ is

   Buy Amazon, Buy Google, Buy Google, Buy eBay

   then $S'$ is not a subsequence of $S$.

   Prove that your algorithm is optimal via a "Greedy Stays Ahead" argument as we did in class for the Interval Scheduling problem.

2. You have $n$ jobs $J_1, J_2, \ldots, J_n$ that need to be executed. Each job has two stages: a preprocessing stage that is done on a supercomputer, and a finishing stage that is performed on a PC.

   There are more than $n$ PCs available, so the second stage can be done in parallel (any number of jobs can be in their second stage simultaneously). But the preprocessing stages (on the supercomputer) have to be done sequentially, that is, one job at at time.

   Suppose that job $J_i$ needs $p_i$ seconds of time on the supercomputer followed by $f_i$ seconds of time on a PC. Design an algorithm that finds a schedule (order in which to process the jobs on the supercomputer) that minimizes the completion time of the last job. (Assume that a job starts its finishing time on a PC immediately after its supercomputer preprocessing time is completed.) Prove that your algorithm is correct via an exchange argument.
Example: Suppose there are three jobs with \((p_1, p_2, p_3) = (5, 4, 3)\) and \((f_1, f_2, f_3) = (7, 6, 1)\). If we schedule in the order 1, 2, 3, then \(J_1\) finishes at time 12, \(J_2\) finishes at time 15 and \(J_3\) finishes at time 13. Thus, the completion time of this schedule is 15.

3. Let \(G = (V, E)\) be a graph whose nodes represent sites that want to communicate with each other. Every edge \(e\) is a communication link and has a bandwidth \(b_e\).

We wish to construct a path between every pair \(u, v\) of nodes in \(V\) on which they will communicate. The bottleneck rate of a path \(P\), which we denote by \(b(P)\), is defined as

\[
\text{bottleneck rate } b(P) = \min_{e \in P} b_e.
\]

Ideally, we’d like to choose paths with high bottleneck rates.

For efficiency, it would be convenient if there was a simple description of these \(\binom{n}{2}\) paths, e.g., if there was a tree \(T\) such that for every \(u, v \in V\), the tree path between them achieved the bottleneck rate. Show how to efficiently find such a tree \(T\), by reducing the problem to a minimum spanning tree computation. (No need for pseudocode for the MST part.) Prove that your algorithm is correct.