EC504 Homework Assignment 3 (due October 26 in class)

1. First solve CLRS Exercise 21-1. Then implement your algorithm (coding up the disjoint-set structure yourself) and run it on the file `hw3test.txt`, which contains $n = 100$ inserts and $m = 50$ extract-mins. Report the *extracted* array that is produced as a result of running your code.

2. The wildly popular Spanish-language search engine El Goog needs to do a serious amount of computation every time it re-compiles its index. Fortunately, the company has at its disposal a single large super-computer together with an essentially unlimited supply of high-end PC's.

   They've broken the overall computation into n distinct jobs, labeled $J_1, J_2, \ldots, J_n$, which can performed completely independently of each other. Each job consists of two stages: first it needs to be *pre-processed* on the super-computer, and then it needs to be *finished* on one of the PC's. Let's say that job J_i needs p_i seconds of time on the super-computer followed by f_i seconds of time on a PC.

   Since there are at least n PC's available on the premises, the finishing of the jobs can be performed fully in parallel — all the jobs can be processed at the same time. However, the super-computer can only work on a single job at a time, so the system managers need to work out an order in which to feed the jobs to the super-computer. As soon as the first job in order is done on the super-computer, it can be handed off to a PC for finishing; at that point in time a second job can be fed to the super-computer; when the second job is done on the super-computer, it can proceed to a PC regardless of whether or not the first job is done or not (since the PC's work in parallel); and so on.

   Let's say that a *schedule* is an ordering of the jobs for the super-computer, and the *completion time* of the schedule is the earliest time at which all jobs will have finished processing on the PC's. This is an important quantity to minimize, since it determines how rapidly El Goog can generate a new index.

   Give a polynomial-time algorithm that finds a schedule with as small a completion time as possible.

3. 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 website that can accuracy 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 website 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 website guarantees that $f_e(t) \geq t$ for all edges e and all times t (you can't travel backwards 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 does not not vary with the season, we would have $f_e(t) = t + \ell_e$, where ℓ_e is the time needed to travel from the beginning to the end of edge e.

Your friends want to use the website 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 website are completely correct.) Give a polynomial-time algorithm to do this, where we treat a single query to the website (based on a specific edge e and a time t) as taking a single computational step.