Consider the following variation on the Activity Selection Problem. You have a processor that can operate 24 hours a day, every day. People submit requests to run *daily jobs* on the processor. Each such job comes with a *start time* and an *end time*; if the job is accepted to run on the processor, it must run continuously, every day, for the period between its start and end times. (Note that certain jobs can begin before midnight and end after midnight; this makes for a type of situation different from what we saw in the Activity Selection Problem.)

Given a list of n such jobs, your goal is to accept as many jobs as possible (regardless of their length), subject to the constraint that the processor can run at most one job at any given point in time. Provide an algorithm to do this with a running time that is polynomial in n. You may assume for simplicity that no two jobs have the same start or end times.

Example. Consider the four jobs, specified by *(start-time, endtime)* pairs. *(6 P.M., 6 A.M.), (9 P.M., 4 A.M.), (3 A.M., 2 P.M.), (1 P.M., 7 P.M.).*

The optimal solution would be to pick the two jobs (9 P.M., 4 A.M.) and (1 P.M., 7 P.M.), which can be scheduled without overlapping.

16.1-2

Suppose that instead of always selecting the first activity to finish, we instead select the last activity to start that is compatible with all previously selected activities. Describe how this approach is a greedy algorithm, and prove that it yields an optimal solution.

16.2-6 ★

Show how to solve the fractional knapsack problem in $O(n)$ time.