[Optional] HW 5 Part C: Due Wednesday, November 15, 2017

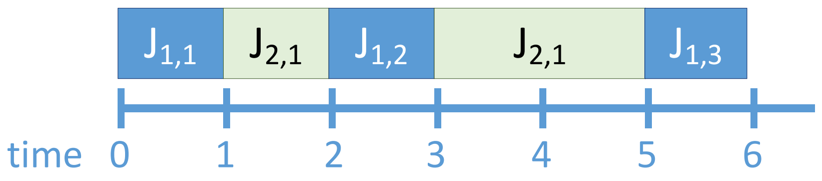
Greedy Task Scheduling, continued

For this problem, we have a setup similar to Problem 3, except you are now scheduling *periodic* tasks, which are defined below. We now only care about tasks completing by their deadlines, so *it is no longer important that they finish quickly, just that they finish on time.* You will write two different greedy scheduling algorithms, one of which gives an optimal solution, and one of which does not.

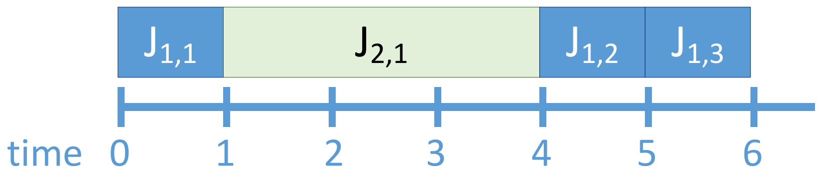
Like before, we have a set $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$ of n tasks. Each task τ_i requires c_i units of computation time to complete once it has started. There is still only one computer on which to run tasks, and the computer can run only one at a time. We also assume that we allow *preemption*, so that the running task can be suspended and restarted at a later time. (Note that some scheduling algorithms, by their nature, don't need to account for preemptions, which can make them simpler to implement.)

Unlike in Problem 3, tasks are now repeated periodically. A task τ_i releases a series of *jobs* each p_i time units. Each of these jobs needs to complete by the time the next one is released. We specify this by saying that a job $J_{i,j}$ of task τ_i has a release time $r_{i,j} = j * p_i$ and deadline of $r_{i,j} + p_i$.

We say that a task set is *schedulable* by some algorithm A if all jobs can be assigned to run on the computer and meet their deadlines. For example, if there are two tasks τ_1 and τ_2 with periods $p_1 = 2$ and $p_2 = 6$ and execution times $c_1 = 1$ and $c_2 = 3$, then jobs of these tasks could be scheduled as follows:



However, the following schedule causes $J_{1,2}$ to miss its deadline:



a. The first scheduling algorithm that you'll implement is called *First-In First-Out (FIFO)*. A FIFO scheduler behaves exactly as it sounds: jobs are executed in order of their releases, chosen greedily. You should fill in the implementation of `scheduleTaskSet` in `FifoScheduler`.

Do you need to handle preemptions? Why or why not? If so, make sure to implement this.

b. Run your FIFO scheduler on the three provided task sets. Include the resulting outputted schedules, drawn in the style of the the schedules above, in your writeup. Which of the task sets make their deadlines when scheduled using FIFO? (Make sure to check for all jobs released in the interval, not just any that complete in the interval.)

c. The second scheduling algorithm is called *Earliest Deadline First (EDF)*. Under EDF, jobs are chosen to run in order of their deadlines; the algorithm always makes the greedy choice of the job with the earliest deadline out of all available jobs. For this, fill in the implementation of `scheduleTaskSet` in `EdfScheduler`.

Do you need to handle preemptions? Why or why not? If so, make sure to implement this.

d. Run your EDF scheduler on the three provided task sets. Include the resulting outputted schedules, drawn in the style of the the schedules above, in your writeup. Which of the task sets make their deadlines when scheduled using EDF? (Make sure to check for all jobs released in the interval, not just any that complete in the interval.)

e. One of the task sets is not schedulable on a single machine under any scheduling algorithm. Describe, in your own words, why this task set is not schedulable.