# CSC 369 Worksheet 5 Solution

August 18, 2020

1. `-M 0` turns off I/O in `mlfq.py`

2. `-n 2` sets number of queues to 2

3. `-j 2` sets number of jobs to 2

Notes

-
- **Multi-level Feeback Queue (MLFQ):**
    - Is one of the most well-known approaches to scheduling
    - Does two things:
        - a) Optimizes turnaround time
        - b) Minimizes response time
    - Uses **priority level** and **Queues** to achieve it's goal
- **MLFQ Basic Rules:**
    - Jobs on same queue $\rightarrow$ Same priority
    - **Rule 1:** If Priority(A) > Priority(B), A runs (B doesn't)
   - **Rule 2:** If Priority(A) = Priority(B), A & B run in RR
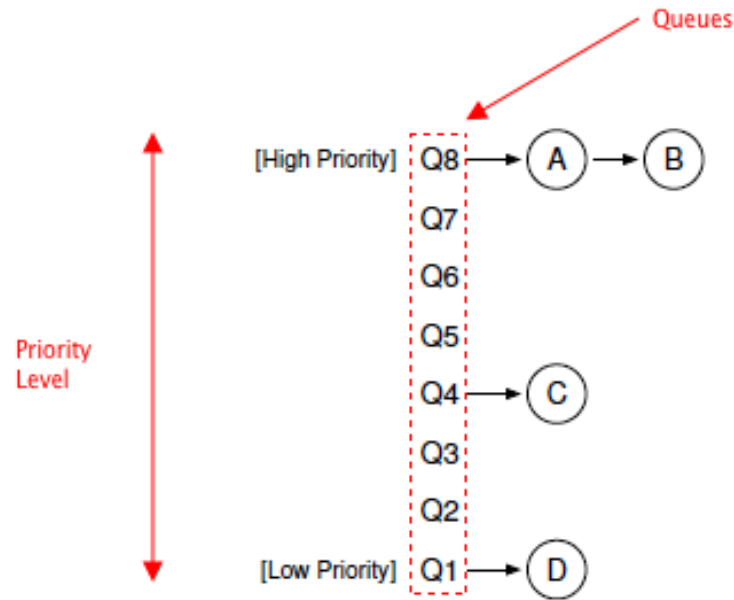
- **Attemp #1: How to Change Priority**
  - **Rule 3:** When a job enters the system, it is placed at the <u>highest</u> priority (the topmost queue)
  - **Rule 4a:** If a job uses up an entire time slice while running , its' priority is <u>reduced</u> (i.e. it moves down on queue).
  - **Rule 4b:** If a job gives up the CPU before the time slice is up, it stays at the <u>same</u> priority level (e.g I/O Operation)
    * Means that the shifting down of priority level only depends on CPU time

  Example (Along Came a Short Job):

  1) A job A enters system
  2) Job is placed on highest Queue Q_2
  3) After time-slice (e.g. 10 ms) in Q_2, A is placed on lower queue Q_1
  4) After time-slice in Q_1, A is placed in lowest priority queue Q_0
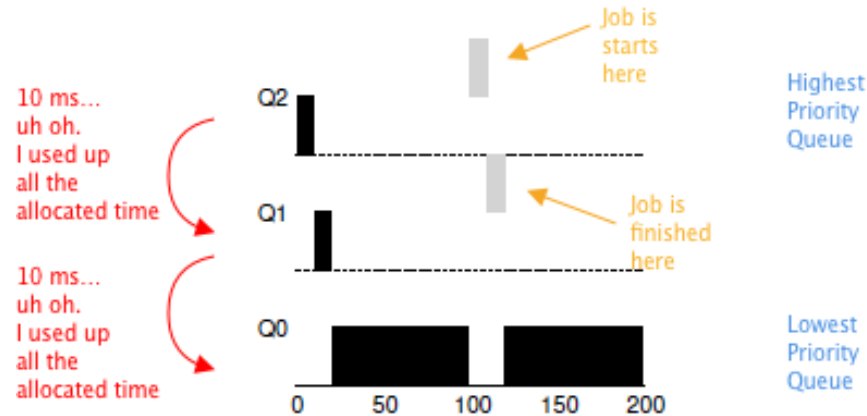
- **Attemp #2: The Priority Boost**

    - **Rule 5:** After some time period S, move all the jobs in the system to the topmost queue.

        * This is to prevent starvation (i.e. a job never being run)

- **Attempt #3: Better Accounting (Fix of Attempt # 1)**

    - Is to prevent programmers from gaming (i.e tricking) the CPU so all programs get a fair share of allotment time
    - **Rule 4:** Once a job uses up its time allotment at a given level (regardless of how many times it has given up the CPU), its priority is reduced (it moves down one queue).