**Another scheduling exercise**

1. The shortest job first (SJF) scheduling algorithm is optimal with respect to process waiting time. Explain what makes it difficult to use in reality, and is there any way to partially overcome that issue?
2. In our exercises, we have imagined a simplification of the SJF pre-emptive scheduling algorithm, with no execution quantum. In reality, pre-emptive scheduling algorithms have an execution quantum associated with them. Why might it be needed?
3. Assume the following scheduling algorithm:

* It has 3 levels with priorities 3, 2, 1 (with 3 being the highest priority)
* Each level uses a round-robin scheduler with quantum

**q = 2(maxlevel-level)**, with maxlevel = 3

* Each time a process is executed for time **q**, it is moved to a level with priority **level-1** below
* Each new process starts at level with highest priority, i.e. 3
* If a process remains on level 1 for 10 time units, then it is moved back up to the level 3. The moving of processes between levels always happens AFTER they are finished being executed
* Only one process is executed at a time, and there is no pre-emption
* At each and every step a process must be executed, or a process is executed and another (or the same) process moves to another level

Your work is to show the scheduling of the following processes (and show your calculations):

|  |  |  |
| --- | --- | --- |
| **PID** | **Arrival Time** | **CPU Burst** |
| P1 | 0 | 10 |
| P2 | 4 | 8 |
| P3 | 4 | 7 |