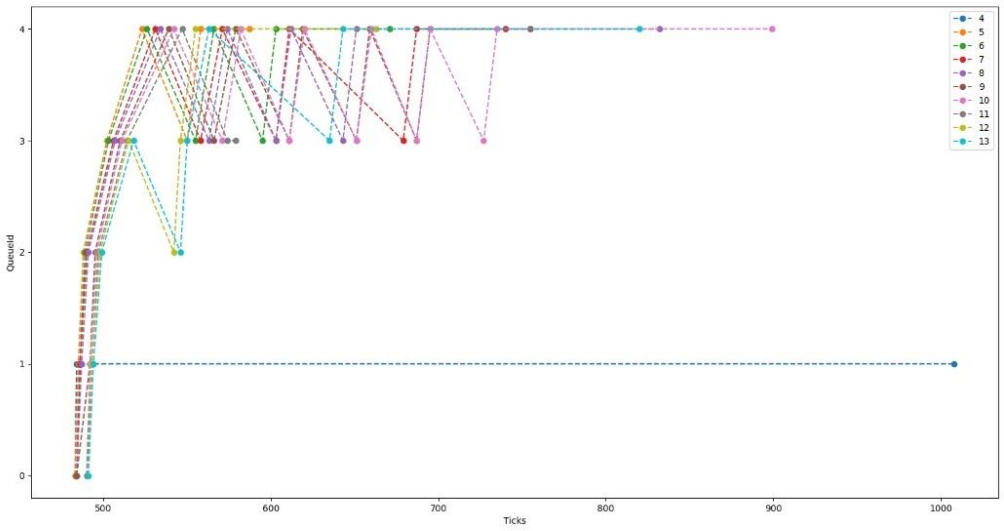
# BONUS

## PLOTTING GRAPHS

I have used the "benchmark2" program to get the values required into the file graphfile.txt. I printed the ticks, PID, and cur_q values into the file "graphfile.txt" and make changes to take out the extra sentences printed. I have added the benchmark2 program as a user program. The file graph.py contains the python code for the plot.

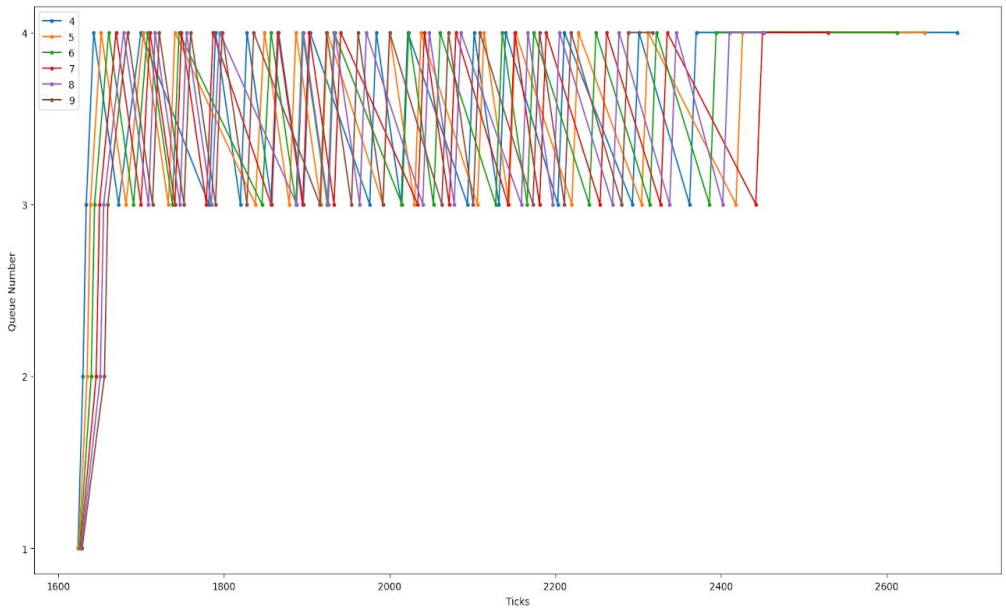
# Graph

The below graph is obtained for the given benchmark.




# MLFQ PLOTS

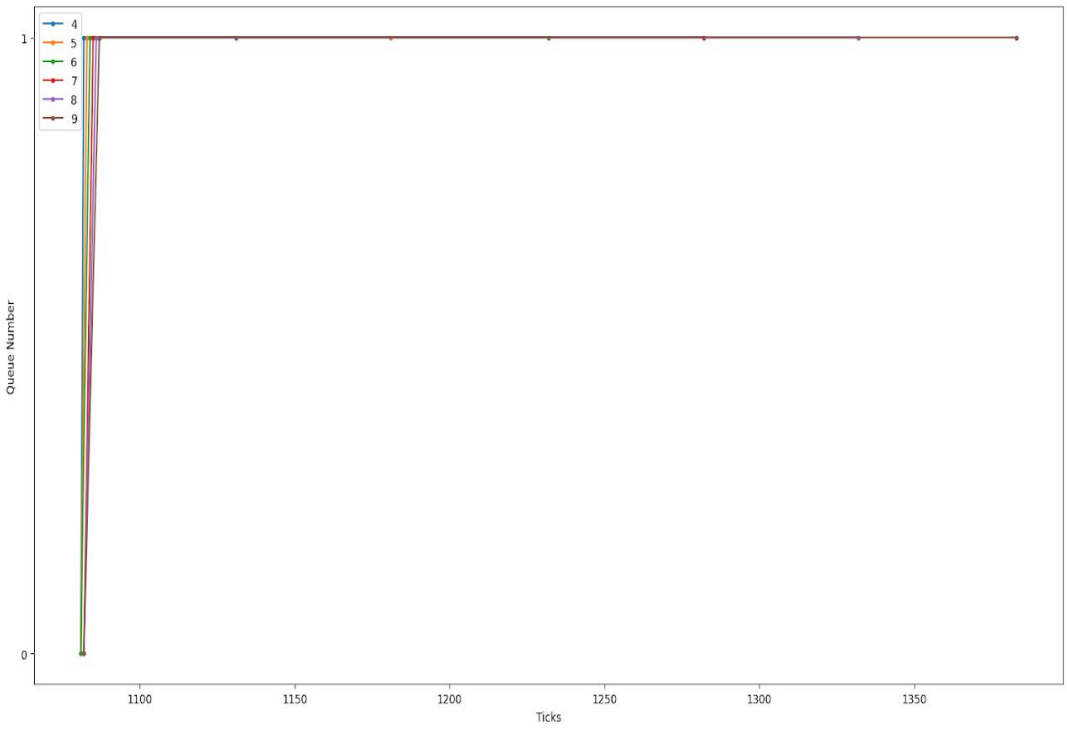
## For CPU-bound Processes:




The CPU bounded graph shows that the majority of the goes to the majority of the process stays in queues 3 and 4 because of the long CPU bursts they are denoted to lower priority queues.

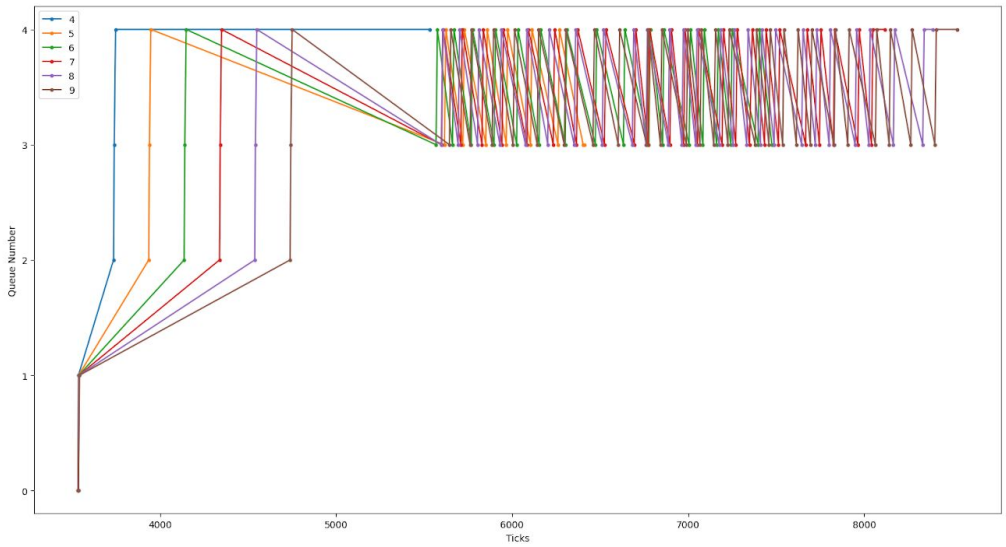
## #I/O bound processes

I/O bound processes




IO Bound Processes have shorter CPU Bursts and stays in Higher Priority Queues like 0, 1, 2 etc..

# MIXED PROCESSES

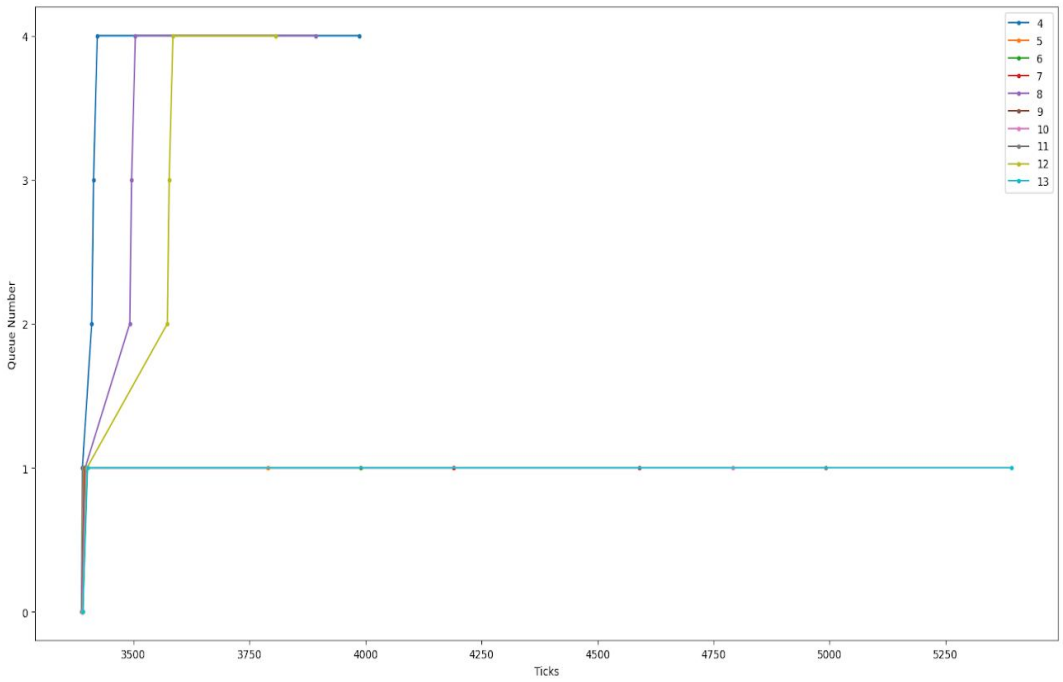



From the above graphs, we can see that I/o bound process with short bursts don’t utilize their full-time slices in the queue and remain at higher priority queues while CPU Bound processes move to higher queues because they won’t voluntarily give up the CPU. The process with very high CPU bursts stays in queue 3 and 4. The aging process brings them back to q4. The aging factor can be changed I have taken a small one to get the show aging.

# ABT process for MLFQ:

To exploit the MLFQ the load may contain a sufficient number of I/O bound processes so that they would be using their timeslice correctly and get executed faster and hence remain in the high priority queue. It supports I/O bound process much better. If the distribution between the I/O process and the CPU bound process is somewhat half then That set of process exploits the MLFQ.

Graph:-




#PERFORMANCE COMPARISONS

I got the following order of processes:

1.PBS

2. RR

3. MLFQ

4.FCFS

ANALYSIS:

TICKS FOR 25 PROCESS:

| | |
|---|---|
| PBS | 5839 |
| RR | 6029 |
| MLFQ | 6894 |
| FCFS | 7600 |

CONCLUSION:

We have tested against a good mix of I/O bound and CPU bound process.

We have got PBS as the best performer since it is preemptive and gives

Priority to the required process. If we assign the priorities correctly then we can exploit the full benefits of PBS.CFS showed the worst performance as all CPU Bound Processes came first and the IO-bound Processes had to wait for them to get over, and there were no processes to run while the IO Bound functioned and were doing IO. RR and MLFQ give similar results RR seems to be good for the CPU bound process and MLFQ seems to be good for the IO-bound process.

PBS:

In PBS the I/O process got executed early since they are given high priorities and they use their time slice and performs I/O. By this, we are able to give fewer priorities to CPU bound processes.

RR:

RR we are pre-empting at each clock tick and hence each type of process get equal process. The only overhead is that the I/o bound process has to wait for a long time if they are at end of the process list even though they need only some time.

MLFQ:

From the above graphs, we can see that the I/O process stays in queue 0,1 mostly and CPU bound process stays mostly in Queue3 and Q4. So the short CPU burst I/o are executed much faster than the CPU bounded ones.CPU bound used their slices and process (IO and CPU Bound) to get equal time slices. Equal opportunity makes it faster. The problem arises when the I/O process is present with a long CPU bound process it waits unnecessarily for getting a small CPU time.

Hence were pushed to lower priority queues and were chosen IO Bound were Sleeping hence the performance gain.

FCFS:

For the FCFS the I/o bound process would be bottleneck since there is no preemption all the process has to wait for the previous process to be completed.

The graph for 40 Process (Both I/O and CPU) looks like this: