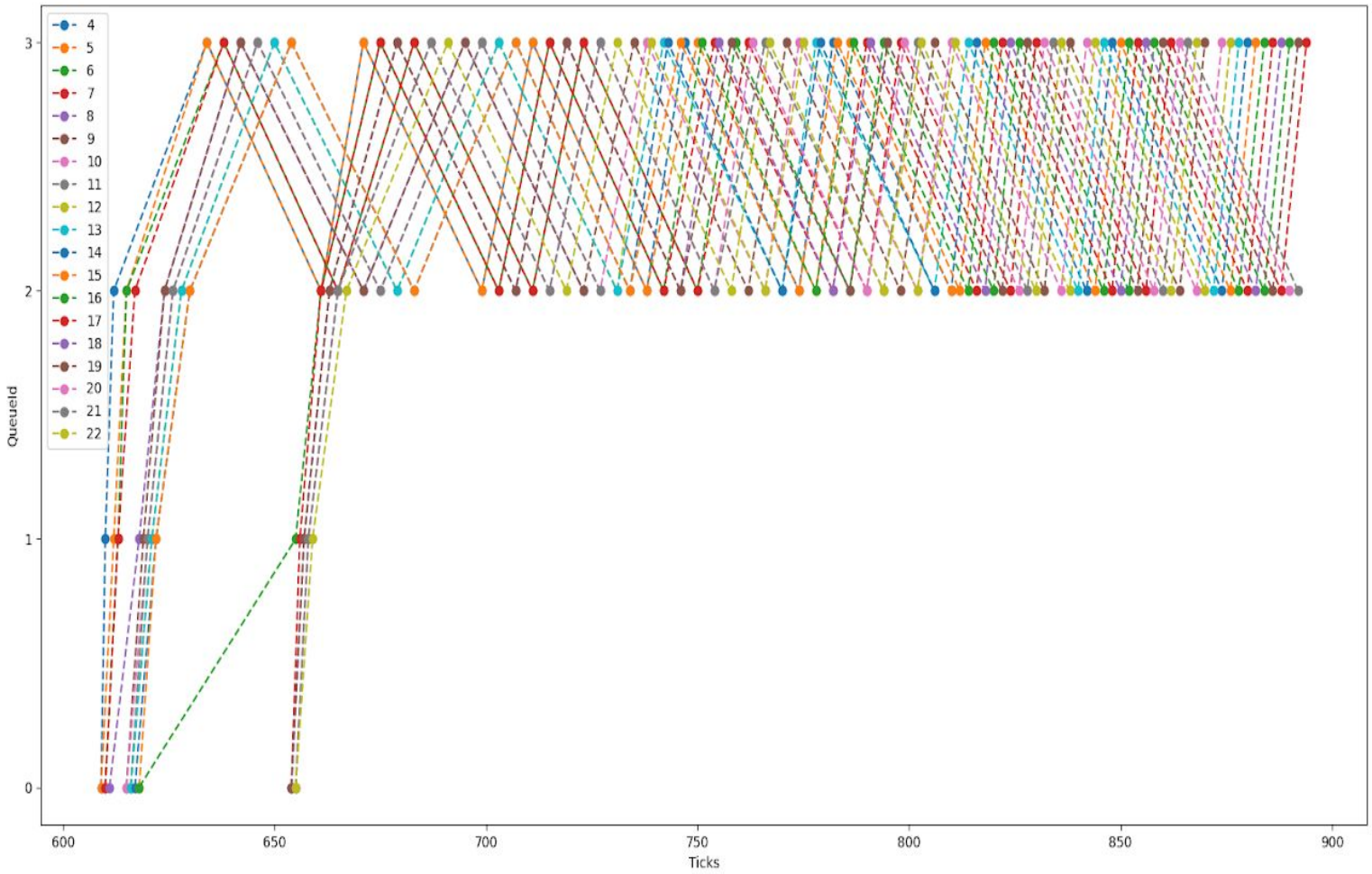
ASSIGNMENT 5 XV6: REPORT

Plotting Graphs

For Bonus, I have made a plotpython.py python program that taking the data from the .txt files (in which the data is printed when the PLOT flag is YESPLOT) and making a graph according to that. And data is collected when the test program is running. In that test program, I have 6 types of processes but I did a graph for 5 types the last thing that is helpful for doing the FCFS thing so I did not do the graph for that. For further instructions on how to run is mentioned in Readme.

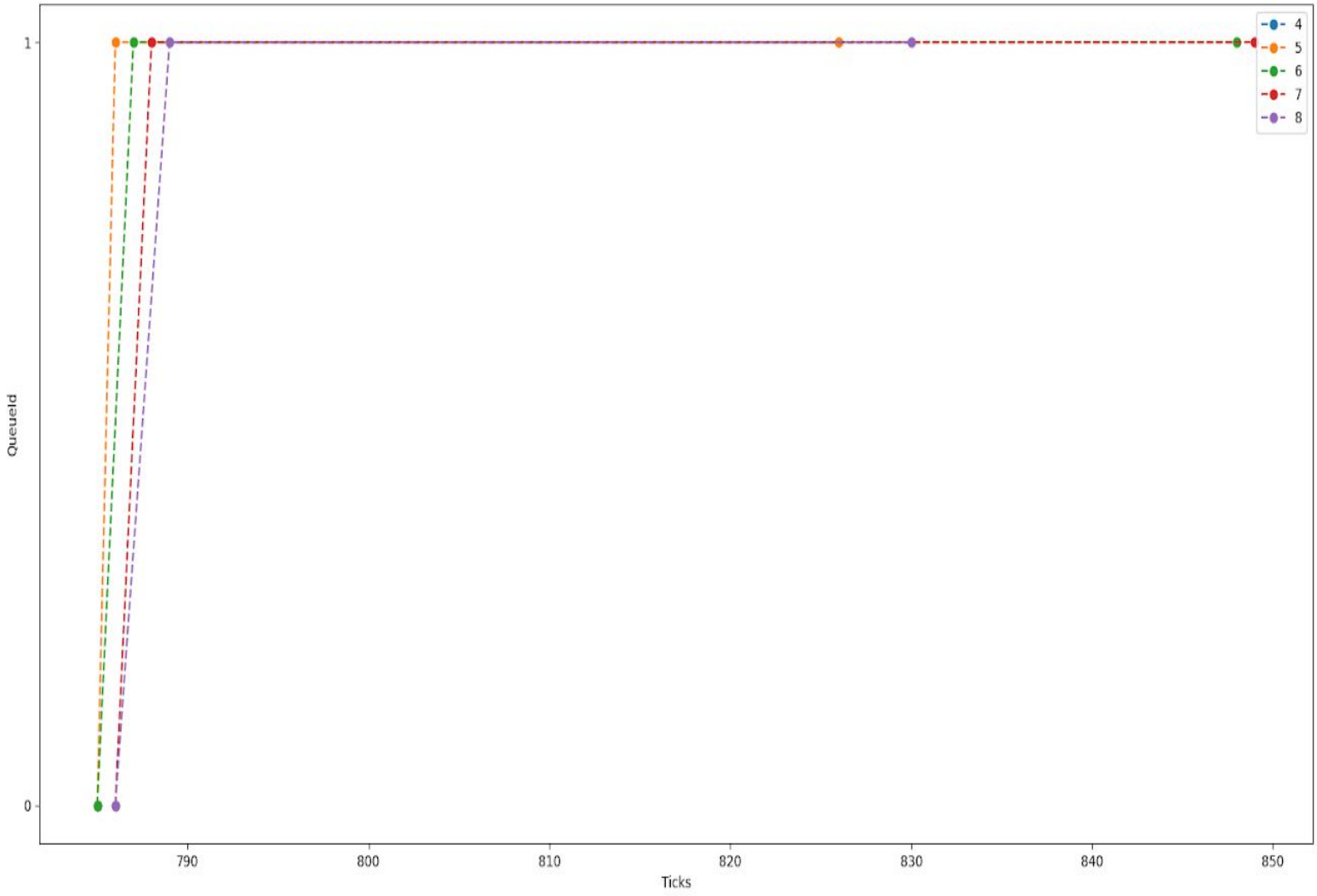
CPU Bound process



As CPU bound processes are CPU heavy and they don’t give up their CPU before their time slices, That why they move to lower priority queues for 3,4 queues and after some processes may wait for a long time so they will be aged that in the graph there are moves between 3 and 4 queues. The above graph is for CPU bound processes.

IO Bound process

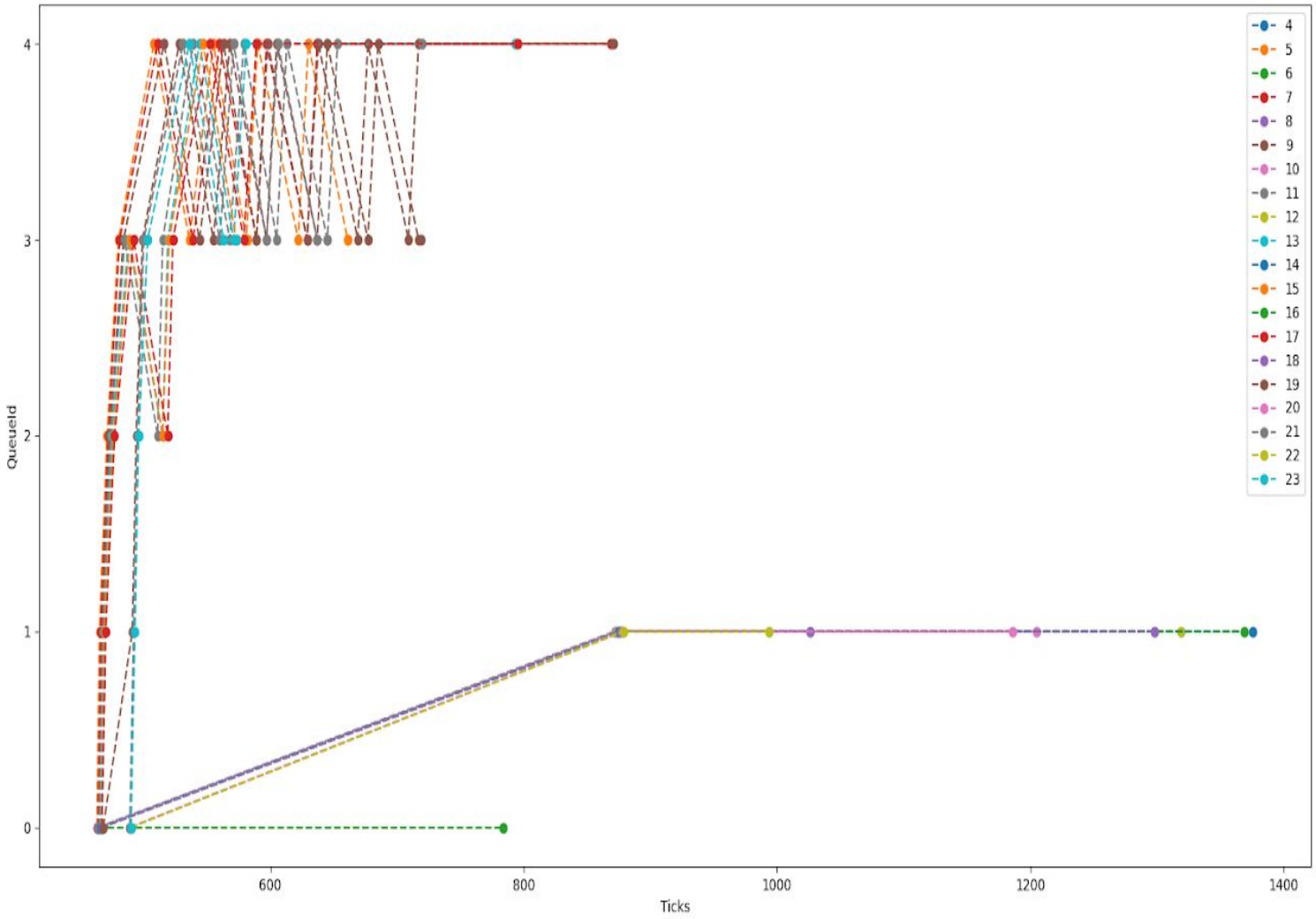
As IO Bound processes don’t take much CPU time (CPU bursts), As they spend much time in doing IO so they will stay at higher priority queues as 0, 1 queue mostly. Here aging may not happen because they are not waiting for CPU they are waiting for IO.



The above graph is for IO-bound processes.

Mixture of IO and CPU Bound processes

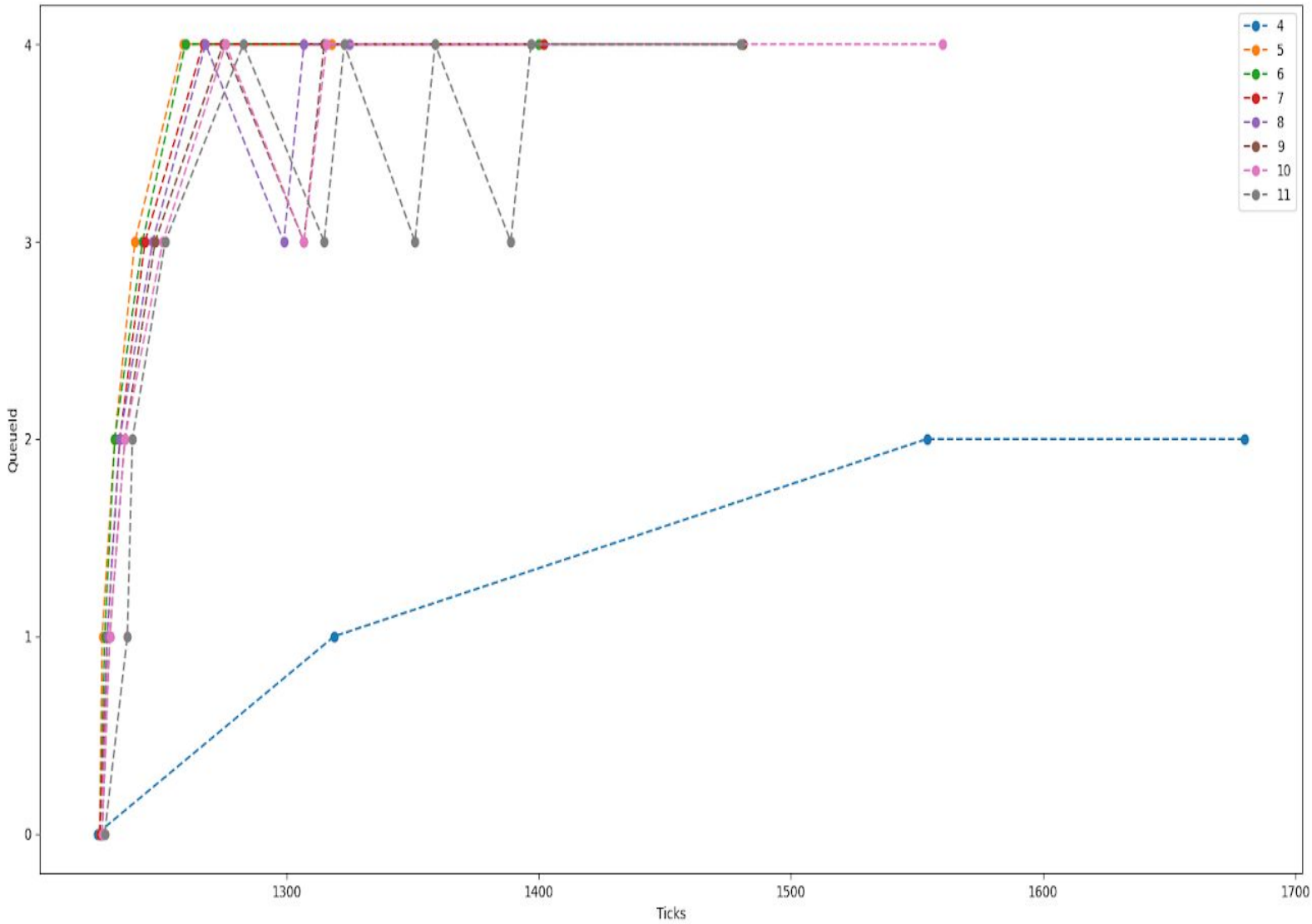
In this case, the IO-bound processes take short CPU times and they may not utilize their whole time slice so they will not be demoted and stays at high priority queues as the CPU bound processes have high CPU times which takes them to lower priority queues and in below graph, the aging also happens for CPU bound process as some CPU bound processes are waiting for the CPU to run on them as other CPU bound



processes taking CPUs. Here I am giving even pids as IO-bound and odd pids as CPU bound.

RandomCPUand IO

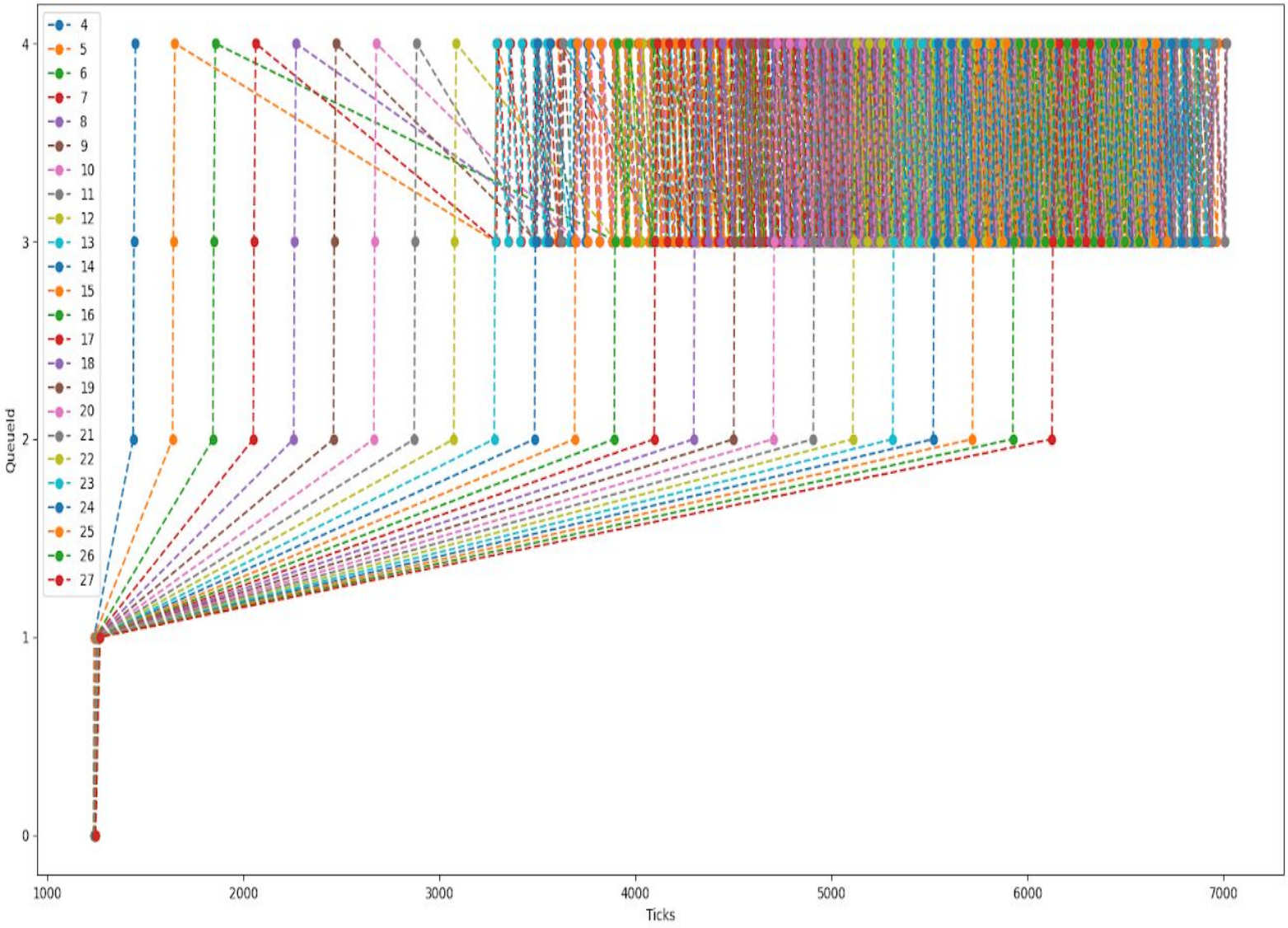
This test case exploits the MLFQ algorithm by giving up CPU before the time slice is completed, so it will remain in the higher priority queue mostly. As in these test cases, I am using 8 processes. In this case, the process with PID 4 is good, In the graph, it stays



at higher priority queues at most it is going to 2 others are going to lower priority queues.

Benchmark program

This is the graph for the given benchmark program. for 25 processes.



PERFORMANCE COMPARISON

Analysis for the given benchmark

Algorithm	Total ticks
RR	4044
FCFS	5727
PBS	4009
MLFQ	4068

As for this benchmark program, the order of performance is

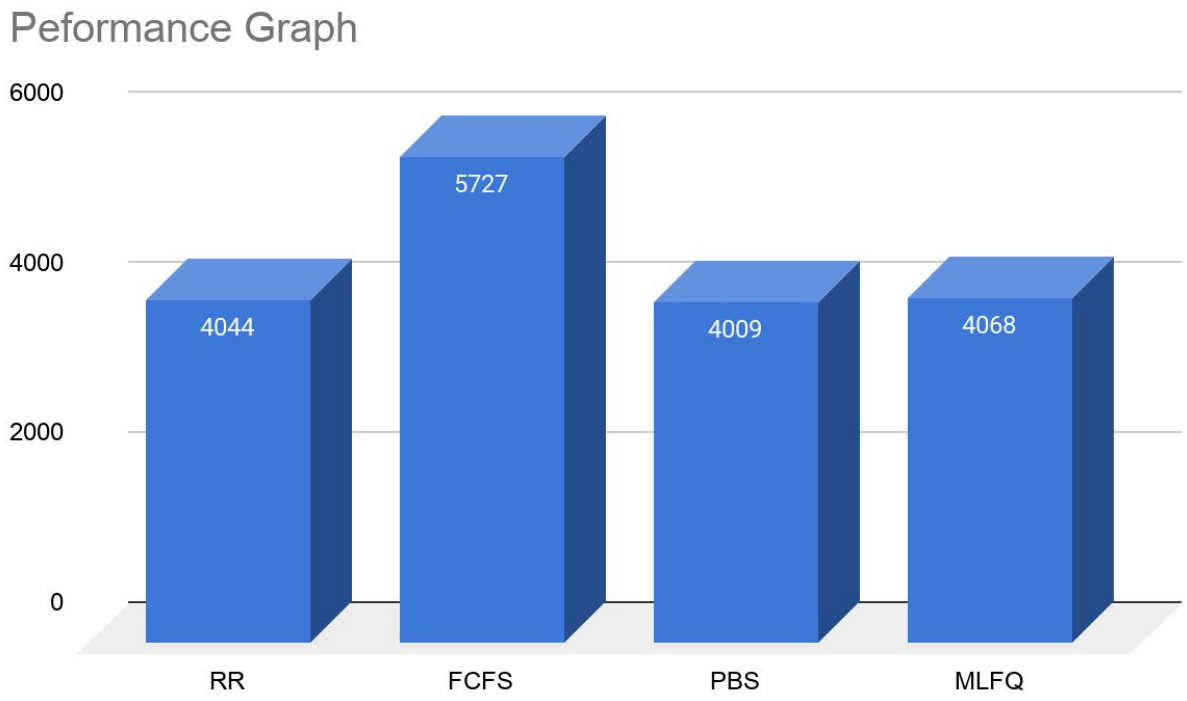
1.PBS

2. RR

3. MLFQ

4. FCFS

And the corresponding bar graph for performance comparing is



RR:

In RR we are preempting(yielding) the process at each tick, So, io and CPU bound processes will execute in the same order i.e will get equal chances of running on the CPU. So here the io bound process may not be waiting for the CPU Bound process As each gives only one tick whichever they come first. So waiting will be lesser so it is somewhat faster.

FCFS:

In FCFS, There is no preempting between the process(yielding), That means if a CPU bound process came first the next process may be io process should have to wait until the CPU bound process completes its execution. After this when the io bound process it gets into sleeping. So as the waiting for that CPU bound makes more waiting for a CPU for other processes Thus this takes a long time.

PBS:

In PBS, The waiting time depends on which it gave more priority. If we give more priority to a CPU Bound process than an IO Bound process it takes more waiting time like in FCFS. If we give more priority to IO Bound process then the waiting time decreases, As in our benchmark it is giving more priority to IO Bound processes so it runs faster.

MLFQ:

As in MLFQ, the IO-bound processes will stay at a higher priority queue because they spend much time in sleeping than running on CPU, and CPU bound processes go to lower priority queues. So the IO processes can run whenever possible so this thing gains performance over FCFS.

All graphs are in the Graphs folder.