

# **Process Scheduling algorithms simulation**

Team Name: Thengakola

Members: Iresh Agarwal, Deepak Singh Shekhawat, Abhiram G P, Vyshak P

## **CPU Sheduling Algorithms Implemented**

- **First Come First Serve**
- **Shortest Job First Preemptive**
- **Shortest Job First Non-Preemptive**
- **Priority Preemptive**
- **Priority Non-Preemptive**
- **Round Robin**
- **Multi-level queue**
- **Multi-level feedback queue**

### **1. First Come First Serve**

In First Come First Serve algorithm we are using a timer variable that tracks the time. As soon as a new process is in ready queue its executed on the basis of its arrival time. Initially a vector of processes is given as argument to the fcfs function which sorts it on the basis of arrival time and uses a timer to track which all processes will be present in the ready queue at each point of time and the processes are executed on the basis of arrival time. This is a non-preemptive scheduling algorithm. Here the processes that go into io are in the waiting queue which are pushed back to the ready queue after their respective io-burst times.

### **2. Shortest Job First Preemptive**

Here we execute the processes based on their remaining burst time. Here we give priority to the process which has minimum burstTime1+burstTime2. ReadyQueue and WaitingQueue are used to track the next process to be executed and the processes going for io. This is a preemptive algorithm and here when there is any other process in the readyqueue with minimum burst time the currently executing process will be preempted and the new process is executed.

### **3. Shortest Job First Non-Preemptive**

Here we execute the processes based on their remaining burst time. Here we give priority to the process which has minimum burstTime1+burstTime2. ReadyQueue and WaitingQueue are used to track the next process to be executed and the processes going for io. This is a non-preemptive algorithm and this wont preempt any process. The process with least burst time is taken for execution at each point of time and the entire burst time is executed.

### **4. Priority Preemptive**

Here we execute the processes based on the priority value. In the implimntation we have considered the lower priority value means higher priority for the process. This is a preemtive scheduling algorithm so whenever a process of higher priority comes (new process or process

from waiting queue) the current process under execution is put on back to ready queue and the process of higher priority is scheduled for execution.

## **5. Priority Non-Preemptive**

Here we execute the processes based on the priority value and arrival time. In the implementation we have considered the lower priority value as a process having higher priority. Since this is a non-preemptive scheduling technique we continue with the execution of current process until IO occurs or it finished even if a higher priority process gets added to the ready queue.

## **6. Round Robin**

Here the function takes in the vector of Processes and a time quantum. Each process according to their arrival time is sent to CPU to run for the specified time quantum and after running for the time it is preempted and sent to the back of the ready queue. Once burst time 1 is over it goes for IO and is pushed back into ready queue to complete burst time 2 after the IO. Here also we use minheaps as ready and waiting queue. Ready queue is FCFS and the waiting queue is a minHeap which is based on the re-entry time for the processes after the IO burst time.

## **7. Multi-level Queue**

Our multilevel priority queue we are dividing the incoming processes into different levels based on the total burst time and IO time. All the process having total burst time less than or equal to 10 units is added to the first queue, and out of the remaining the processes with IO execution time less than or equal to 5 is taken to second queue. Remaining processes are added to the last level. Now we are following the execution in the priority order of level 1 > level2 > level3. The scheduling technique used for level 1 is FCFS and for level 2 we are using round robin with time quantum of 4, and finally for level 3 we are using round robin with time quantum of 8.

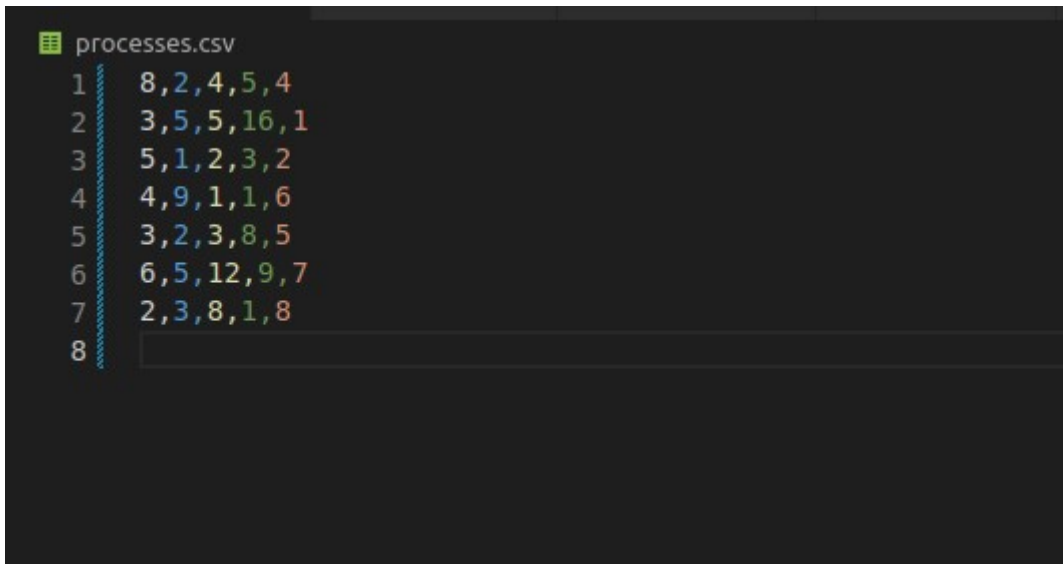
## **8. Multi-level feedback queue**

Our multilevel feedback queue has the first level as round robin queue with time quantum 2, second level as round robin queue with time quantum 5 and the final level is a FCFS queue. Every process that goes for IO be it from any level is pushed back to the first level queue. The priority is high for the top level is least for the bottom level FCFS queue. So here we use 3 readyQueues for each level and one waiting queue for the IO operation. Now each process that comes will be processed through all these layers and the processes with less burst time will be finished earlier and larger processes will reach the FCFS queue where it will be finished when all the high priority processes with less burst time is finished.

## Input Format

Processes.csv will contain data in this order

**[Arrival Time, Burst Time 1, IO Time, Burst Time 2, Priority]**

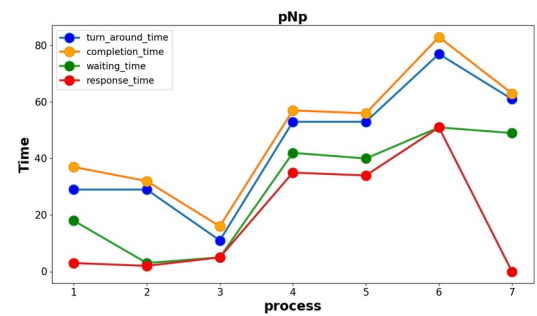
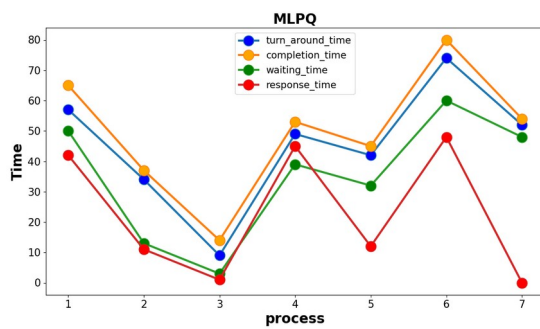
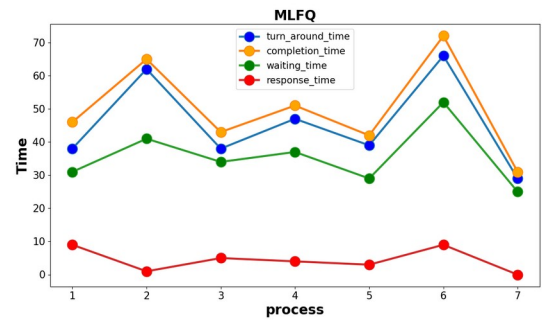
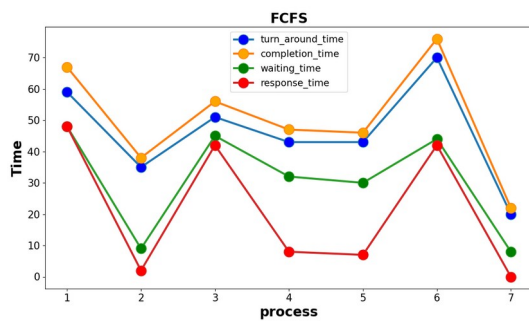
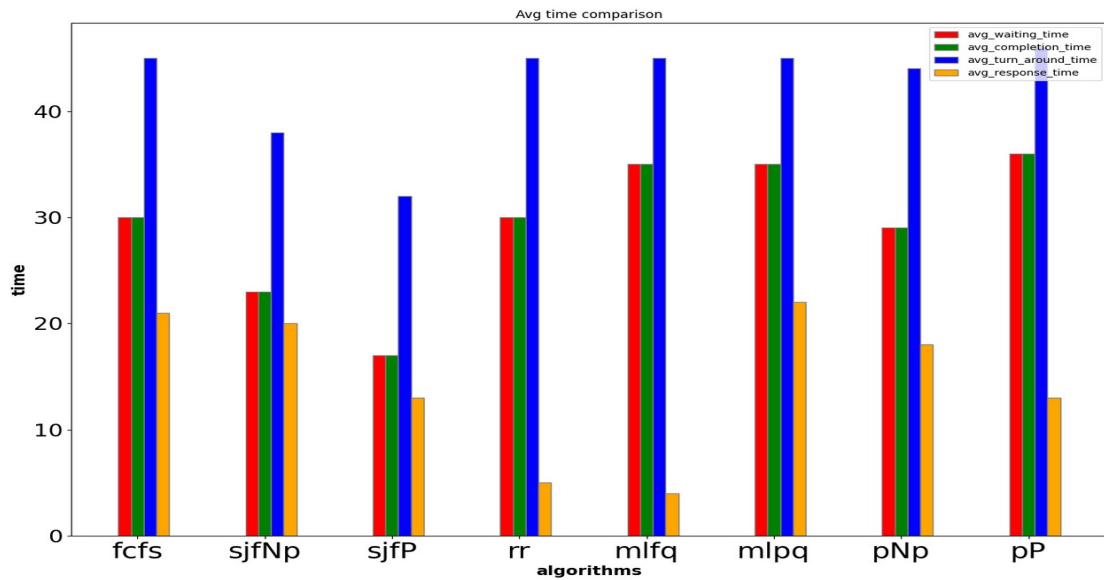
A screenshot of a text editor with a dark background. The file is named 'processes.csv'. It contains 8 lines of data, each with 5 numbers separated by commas. The numbers are color-coded: green for the first number, blue for the second, red for the third, green for the fourth, and red for the fifth. The data is as follows:

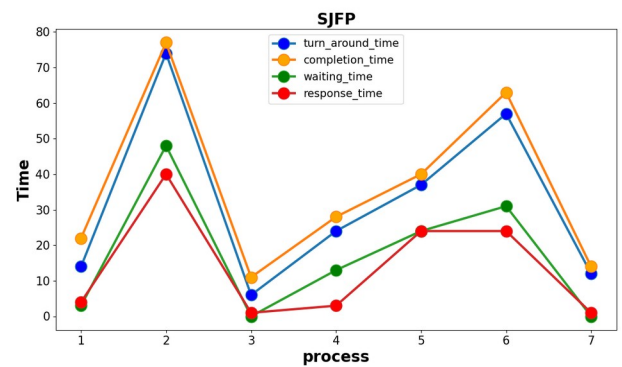
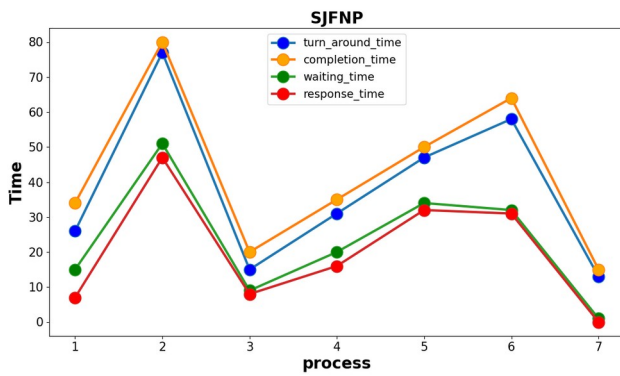
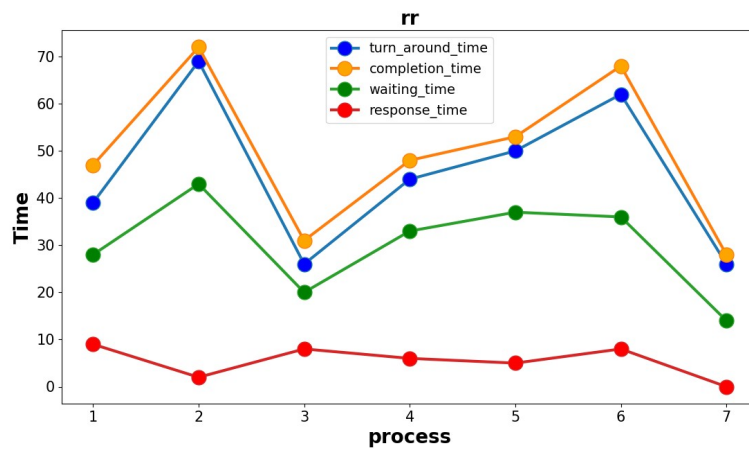
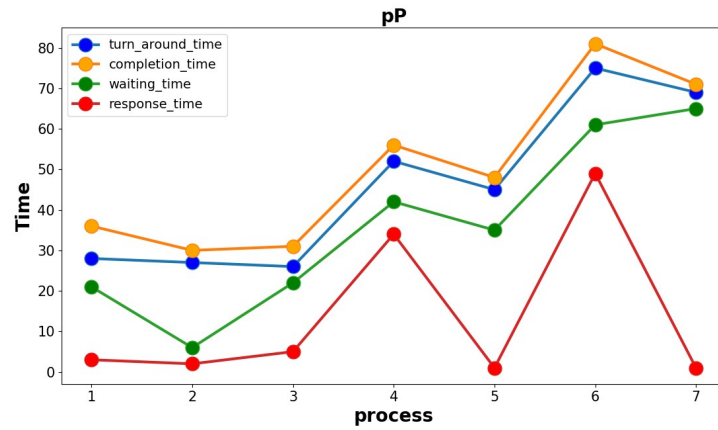
Process ID	Arrival Time	Burst Time 1	IO Time	Burst Time 2	Priority
1	8	2	4	5	4
2	3	5	5	16	1
3	5	1	2	3	2
4	4	9	1	1	6
5	3	2	3	8	5
6	6	5	12	9	7
7	2	3	8	1	8
8					

## How to run ?

1. Open "Process Scheduling algorithms simulation" folder
2. Enter the input in processes.csv file
3. Execute each of these files in the given order
  - a. fcfs
  - b. sjfNp
  - c. sjfP
  - d. rr
  - e. mlfq
  - f. mlpq
  - g. pNp
  - h. pP
4. Now run comparison.py and particular.py. Graphs will be created and saved in the graph folder.

**For the above given input following are the outputs:**





## **Conclusion**

As we can infer from the above graphs Shortest Job First Preemptive has the lowest average waiting time and Round Robin has the lowest average response time. The behaviour of each process based on their waiting time, completion time, turn around time and response time for each algorithm are also elaborated in the graphs. On interpreting these graphs we could get an idea of how each processes behave inside each scheduling algorithms and how priority for each process varies across these algorithms.

Thus this project helps us gain some inside understanding as to what happens to each process throughout a scheduling algorithm and how each process can be treated differently for different scheduling algorithms.

