CPU Scheduling Report

Team Wood:

Mycole Brown
Jordan Fleming
Carter Garcia
Edale Miguel

Introduction:

CPU scheduling is a fundamental aspect of operating system design that determines which processes in the ready queue are allocated CPU time. Efficient CPU scheduling enhances system performance and responsiveness, particularly in multitasking and time-sharing environments where multiple processes compete for the CPU. Various scheduling algorithms have been developed to manage the CPU's task allocation effectively, each with its own strategy and set of trade-offs. These algorithms are crucial for balancing efficiency, fairness, and system throughput. Our team has explored several prominent CPU scheduling algorithms, including First-Come, First-Served (FCFS) Preemptive, Shortest Job First (SJF) Non-Preemptive, Shortest Remaining Time First (SRTF), Priority Scheduling, Round Robin (RR), Multilevel Queue (MLQ), and Multilevel Feedback Queue (MLFQ). Each of these algorithms offers unique approaches to CPU scheduling, tailored to different types of workloads and system requirements.

FCFS – Preemp

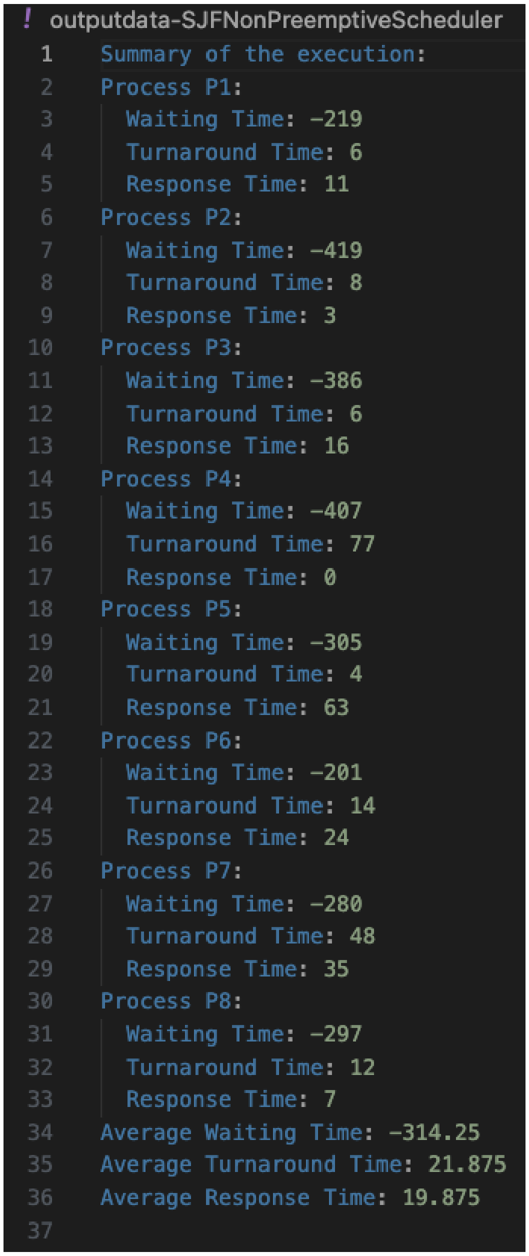
The First Come, First Serve CPU Scheduling algorithm operates by executing processes strictly in the order of their arrival, running each to completion before starting the next. As a non-preemptive method, FCFS is straightforward to implement but can lead to inefficiencies, particularly when a long process delays the execution of shorter, subsequent processes.

Preemption, on the other hand, introduces the capability for the CPU to interrupt a currently running process in order to start or resume another process, thereby enhancing system responsiveness and efficiency. When attempting to create a preemptive version of FCFS, the algorithm inherently transforms into Round Robin scheduling. Round Robin retains the order of process arrival but incorporates a fixed time quantum for each process. If a process does not complete within its allocated time quantum, it is preempted and moved to the back of the queue, ensuring equitable CPU time distribution and improved overall system performance.

The consensus that FCFS with preemption evolves into Round Robin is well-founded. The essence of FCFS lies in its non-preemptive nature, and introducing preemption fundamentally alters this principle. Without adopting a structured approach like time quanta, the scheduling method would deviate from its original identity as FCFS. Consequently, maintaining FCFS as preemptive without transforming it into Round Robin is impractical, as the core attributes of FCFS are inherently non-preemptive.

SJF – NonPreemp

Shortest Job First - Non-Preemptive scheduling selects the process with the shortest CPU burst time for execution next. Once a process starts, it runs to completion without being interrupted. This approach minimizes the average waiting time by prioritizing shorter jobs, ensuring they finish quickly and reducing the time longer processes spend waiting. However, SJF requires accurate predictions of CPU burst lengths, which can be challenging. While effective at reducing waiting times in batch processing systems, SJF can lead to the starvation of longer processes if shorter processes continually arrive.



outputdata-SJFNonPreemptiveScheduler

```
Summary of the execution:
Process P1:
  Waiting Time: -219
  Turnaround Time: 6
  Response Time: 11
Process P2:
  Waiting Time: -419
  Turnaround Time: 8
  Response Time: 3
Process P3:
  Waiting Time: -386
  Turnaround Time: 6
  Response Time: 16
Process P4:
  Waiting Time: -407
  Turnaround Time: 77
  Response Time: 0
Process P5:
  Waiting Time: -305
  Turnaround Time: 4
  Response Time: 63
Process P6:
  Waiting Time: -201
  Turnaround Time: 14
  Response Time: 24
Process P7:
  Waiting Time: -280
  Turnaround Time: 48
  Response Time: 35
Process P8:
  Waiting Time: -297
  Turnaround Time: 12
  Response Time: 7
Average Waiting Time: -314.25
Average Turnaround Time: 21.875
Average Response Time: 19.875
```

SRTF

Shortest Remaining Time First (SRTF) is a preemptive CPU scheduling algorithm that aims to minimize the average waiting time for processes. It prioritizes the process with the shortest remaining burst time (the amount of CPU time still needed to complete the process).

Process	Wait Time	Response Time	Turnaround Time
P1	16	11	241
P2	0	3	312

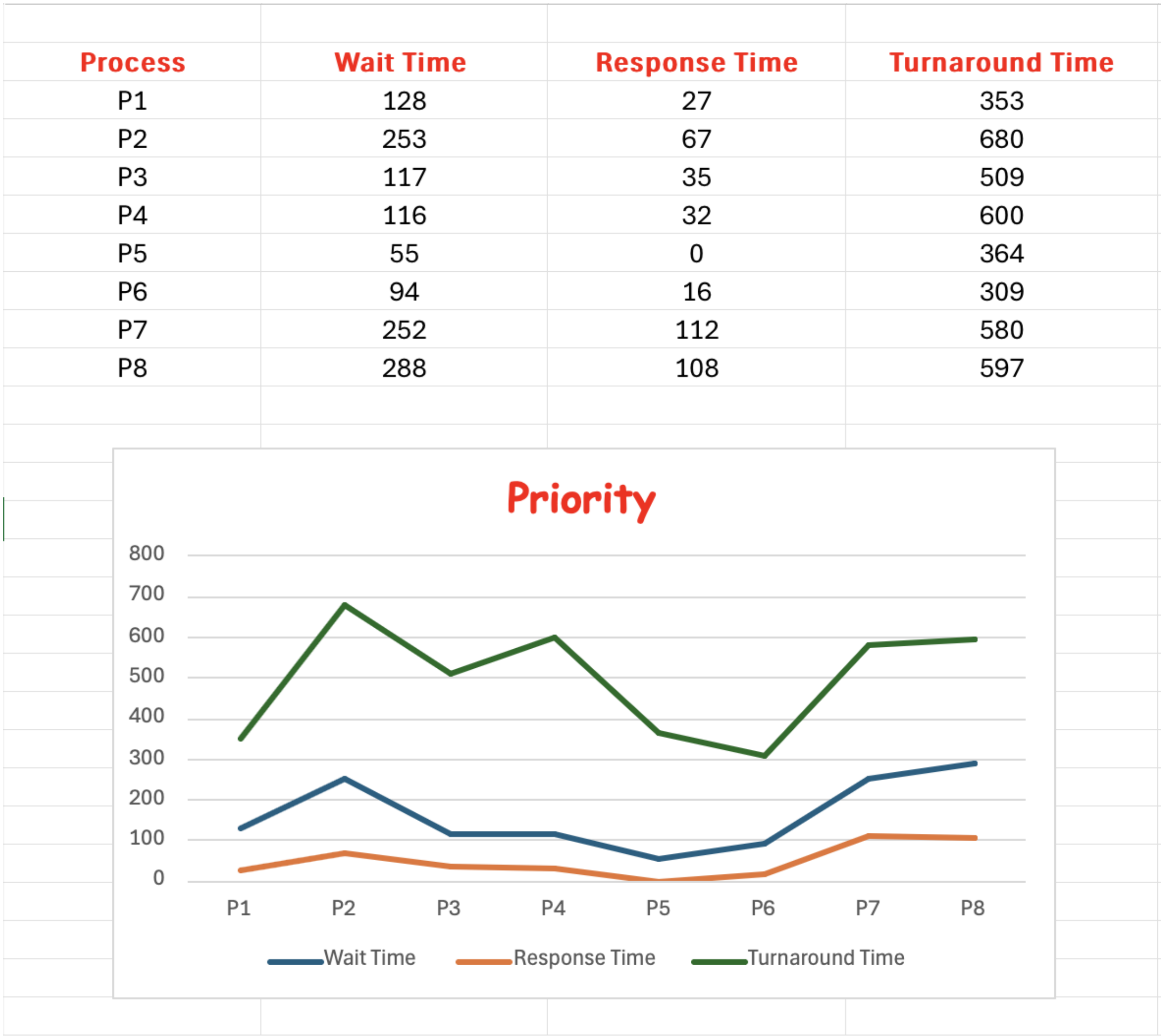
P3	204	16	596
P4	0	0	259
P5	300	177	609
P6	144	24	359
P7	261	52	589
P8	0	7	287

Still unsure as to why but was getting occasional negative wait times. To help alleviate this the wait times were set to be the max of 0 or the calculated wait time (Turnaround time – sum of bursts and IOs). The CPU usage was a very high 90.6%.

Priority Queue

The Priority CPU scheduling algorithm selects processes for execution based on their priority, with higher priority processes being executed before lower priority ones. If two processes have the same priority, they are scheduled according to their arrival times. This algorithm can be preemptive, where a higher-priority process can interrupt a running lower-priority process, or non-preemptive, where a running process completes its burst time even if a higher-priority process arrives.

Report on Scheduling Results:



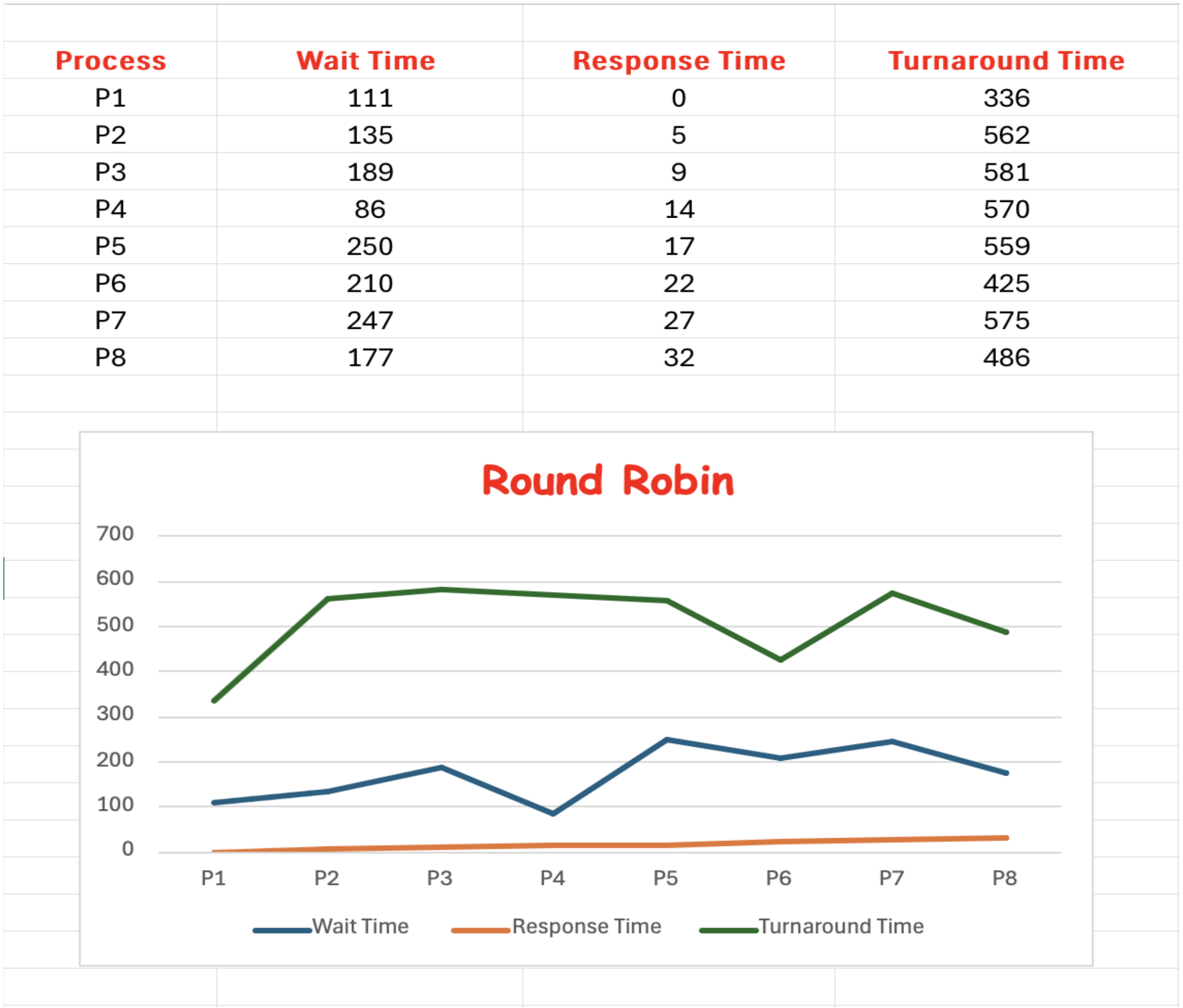
Process	Wait Time	Response Time	Turnaround Time
P1	128	27	353
P2	253	67	680
P3	117	35	509
P4	116	32	600
P5	55	0	364
P6	94	16	309
P7	252	112	580
P8	288	108	597

The Priority CPU scheduler's performance for the given processes shows a wide range of wait times, response times, and turnaround times. The total CPU usage was 81.32%, reflecting moderate resource utilization. The average wait time was 162.88 units, the average response time was 49.63 units, and the average turnaround time was 499.00 units. Process P1 experienced a wait time of 128 units, a response time of 27 units, and a turnaround time of 353 units. Process P2 had a wait time of 253 units, a response time of 67 units, and a turnaround time of 680 units. Process P3's wait time was 117 units, response time was 35 units, and turnaround time was 509 units. Process P4 had a wait time of 116 units, a response time of 32 units, and a turnaround time of 600 units. Process P5 had the shortest wait time of 55 units, a response time of 0 units, and a turnaround time of 364 units. Process P6's wait time was 94 units, response time was 16 units, and turnaround time was 309 units. Process P7 experienced a wait time of 252 units, a response time of 112 units, and a turnaround time of 580 units. Lastly, Process P8 had the longest wait time of 288 units, a response time of 108 units, and a turnaround time of 597 units. Overall, the scheduler prioritized processes effectively based on their priority, resulting in a balanced but varied distribution of CPU time.

Round Robin

The Round Robin (RR) CPU scheduling algorithm allocates a fixed time slice or quantum to each process in the queue, cycling through them in a circular order. If a process does not finish execution within its allocated quantum, it is moved to the end of the queue, and the CPU scheduler assigns the next process in line. This ensures that all processes get an equal share of the CPU time and is particularly effective in time-sharing systems.

Report on Scheduling Results:



Process	Wait Time	Response Time	Turnaround Time
P1	111	0	336
P2	135	5	562
P3	189	9	581
P4	86	14	570
P5	250	17	559
P6	210	22	425
P7	247	27	575
P8	177	32	486

The Round Robin CPU scheduler's performance for the given processes resulted in a fair and efficient distribution of CPU time. The total CPU usage was high at 95.18%, demonstrating effective resource utilization. The processes had varying wait times, with an average wait time of 175.63 units. Response times were prompt, averaging 15.75 units. The average turnaround time for all processes was 511.75 units. Process P1 had a wait time of 111 units and a turnaround time of 336 units, while Process P2 had a wait time of 135 units and a turnaround time of 562 units. Process P3 experienced a wait time of 189 units and a turnaround time of 581 units. Process P4 had a shorter wait time of 86 units and a turnaround time of 570 units. Process P5 had the longest wait time of 250 units and a turnaround time of 559 units. Process P6 had a wait time of 210 units and a turnaround time of 425 units. Process P7 waited for 247 units with a turnaround time of 575 units. Finally, Process P8 had a wait time of 177 units and a turnaround time of 486 units. Overall, the scheduler managed the processes efficiently, ensuring a balanced CPU time distribution and high utilization.

MLQ

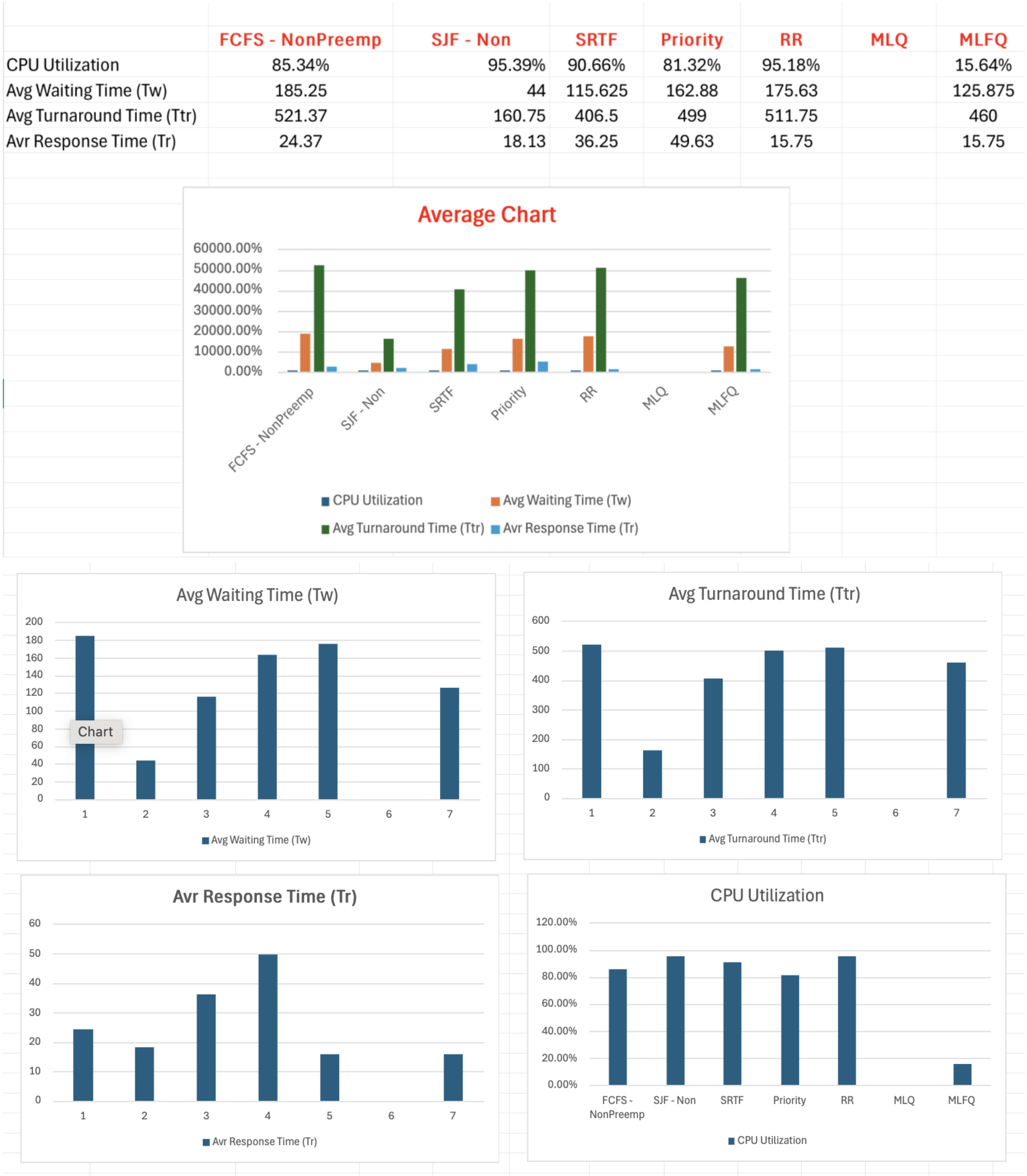
MLFQ

Multilevel Feedback Queue (MLFQ) is a CPU scheduling algorithm designed to improve system performance and fairness for processes with different CPU usage patterns. It works by maintaining multiple queues with different priorities, dynamically adjusting process priorities based on their behavior.

Process	Wait Time	Response Time	Turnaround Time
P1	183	0	408
P2	31	5	458
P3	99	9	491
P4	0	14	468
P5	272	17	581
P6	294	22	509
P7	123	27	451
P8	5	32	314

Still unsure as to why but was getting occasional negative wait times. To help alleviate this the wait times were set to be the max of 0 or the calculated wait time (Turnaround time – sum of bursts and IOs). For some reason the CPU usage is also recorded at an extremely low 15%. I ran out of time to debug this but it has to be skipping increments somehow. The implementation of this algorithm uses a counter that increments when any process on the CPU has a moment of burst.

AVERAGE REPORTS:



	FCFS - NonPreemp	SJF - Non	SRTF	Priority	RR	MLQ	MLFQ
CPU Utilization	85.34%	95.39%	90.66%	81.32%	95.18%		15.64%
Avg Waiting Time (Tw)	185.25	44	115.625	162.88	175.63		125.875
Avg Turnaround Time (Ttr)	521.37	160.75	406.5	499	511.75		460
Avr Response Time (Tr)	24.37	18.13	36.25	49.63	15.75		15.75

The SJF - NonPreemptive and Round Robin exhibit the highest CPU utilization rates at 95.39% and 95.18%, respectively, while MLFQ operates at a notably lower utilization of 15.64%. SJF - NonPreemptive boasts the shortest average waiting time of 44 ms, whereas FCFS - NonPreemptive has the longest waiting time at 185.25 ms. Similarly, SJF - NonPreemptive achieves the best turnaround time of 160.75 ms, contrasting with FCFS - NonPreemptive's highest turnaround time of 521.37 ms. Remarkably, both Round Robin and MLFQ showcase the lowest response time at 15.75 ms. An anomaly appears in the reported response time for SRTF, indicated as 90.66%, which likely necessitates clarification.

The CPU scheduling algorithms exhibit diverse performance characteristics based on the provided metrics. SJF - NonPreemptive and Round Robin demonstrate high CPU utilization rates and efficient task completion, with SJF - NonPreemptive particularly excelling in minimizing waiting and turnaround times. Meanwhile, FCFS - NonPreemptive, while simple, results in longer waiting and turnaround times due to its sequential task execution. The Priority Queue shows competitive CPU utilization and average waiting time, indicating effective prioritization of tasks and SRTF, known for its focus on minimizing remaining execution time, shows promising performance. In summary, each algorithm showcases strengths and weaknesses, and their suitability depends on the specific system requirements and workload characteristics.

Conclusion:

Understanding and implementing these CPU scheduling algorithms is crucial for managing task allocation effectively in an operating system. Each algorithm offers distinct advantages and challenges, making them suitable for different types of systems and workloads. By balancing efficiency, fairness, and responsiveness, these scheduling methods play a vital role in optimizing system performance and ensuring that all processes receive appropriate CPU time.