Project Report
CPU Scheduling Algorithms

Group Member and Contribution

Joy Omaji, jomaji@ttu.edu, R#11828104
Created code for Priority Scheduling Algorithm, drafted problem statement, problem solution, and simulation result
Nivedita Prabhu, nprabhu@ttu.edu, R#11738979
Created code for Round Robin Algorithm, drafted CPU Scheduling Overview
Shravani Kardekar, skardeka@ttu.edu, R#11804620
Created code for Shortest-Job First Algorithm
Yash Nikumb, ynikumb@ttu.edu, R#11722897
Created code for First-Come, First Serve

Problem Statement

In modern computing systems, efficient resource allocation and process scheduling are critical for optimizing performance metrics such as throughput, CPU utilization, and response time. The objective of this project is to evaluate and compare scheduling algorithms such as, First Come First Serve (FCFS), Shortest Job First (SJF), Priority Scheduling, and Round Robin (RR), based on their effectiveness in enhancing system efficiency.

The project draws insights from " A Comprehensive Review and Comparative Analysis with Performance Insights and Simulation-Based Evaluations of Innovative CPU Scheduling Algorithms" written by Izzah Alam. Alam's paper focuses on a comparative analysis that highlights Round Robin as optimal for reducing starvation and improving CPU performance/utilization. The algorithms will be evaluated through simulations to assess their impact on key performance indicators: throughput, CPU utilization, turnaround time, waiting time, and response time.

By addressing these scheduling challenges, the project aims to minimize overhead, achieve stability between response and utilization, and enhance overall system efficiency. The findings will contribute to a deeper understanding of scheduling strategies in computing environments, offering insights into practical applications for enhancing system performance and resource management.

CPU Scheduling Algorithms

First Come First Serve (FCFS) is the simplest type of CPU scheduling algorithm. In this method, the process that arrives first in the ready queue is executed first. It works just like a queue at a service counter — first in, first out.

Round Robin is a preemptive version of FCFS (First-Come, First Serve) where each process stays in the ready state for a fixed time (time quantum). Every process gets cyclically, meaning that their remaining burst time is sent back to the ready state until the next turn and termination. If the burst time is less than or equal to time quantum, then it goes back to Ready State. If the burst time is greater than or equal to the time quantum, then it terminates.

The Shortest Job First (SJF) scheduling technique prioritizes the pending process with the shortest execution time. This method has received considerable recognition as being the most efficient for decreasing the average duration needed to accomplish a task. SJF aims to enhance throughput by reducing process waiting times by selecting the shortest available assignment for execution. The algorithm operates by prioritizing the execution of the process with the shortest duration, regardless of the order in which the processes were received.

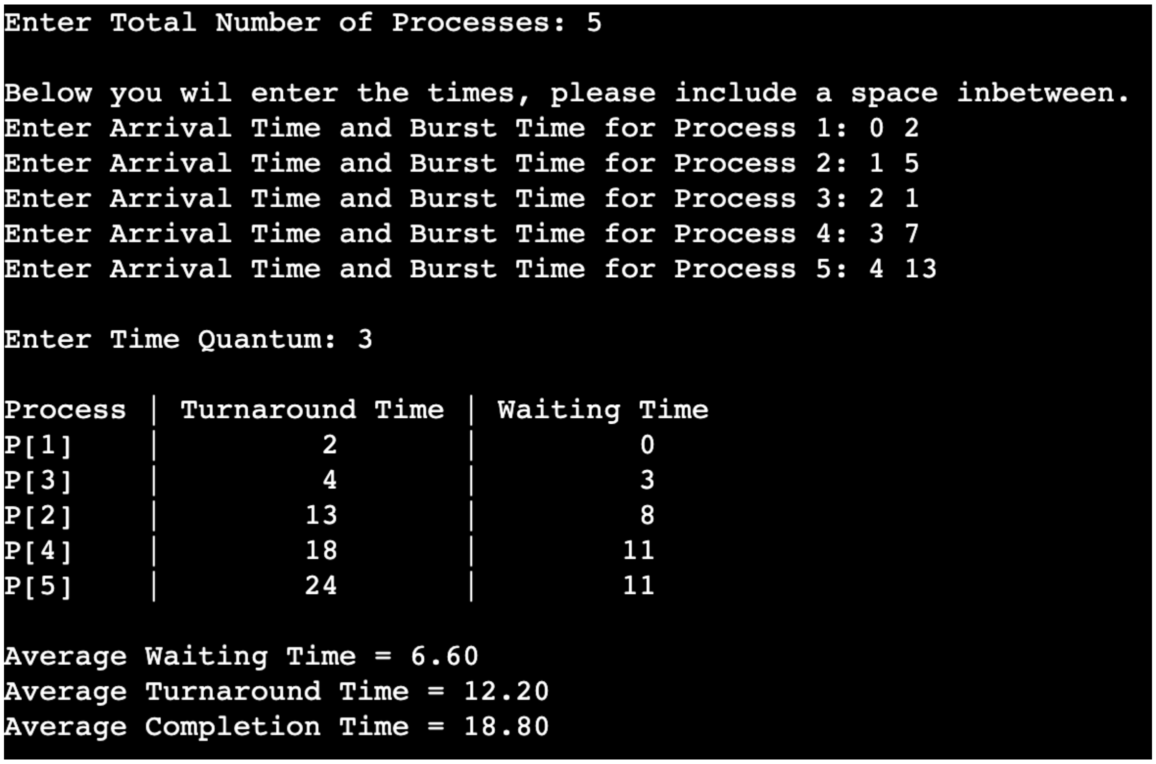
In Priority Scheduling processes are assigned to a priority, and the CPU is allocated to the process with the highest priority. This process also reduces starvation for lower-priority processes.

Problem Solution

To address the problem of efficient CPU scheduling, we implemented and executed code simulations using sample burst times for the four primary algorithms: First Come First Serve (FCFS), Shortest Job First (SJF), Priority Scheduling, and Round Robin (RR). For each algorithm, we assessed key scheduling metrics including arrival time, burst time, completion time, turnaround time, and waiting time. These metrics were used to analyze and compare the overall performance of each scheduling method. Our findings showed that Round Robin, with its time-sharing approach, provided a more balanced allocation of CPU time, reducing the chances of process starvation and optimizing CPU utilization. This resulted in improved system efficiency, especially in scenarios involving multiple processes with varying execution times.

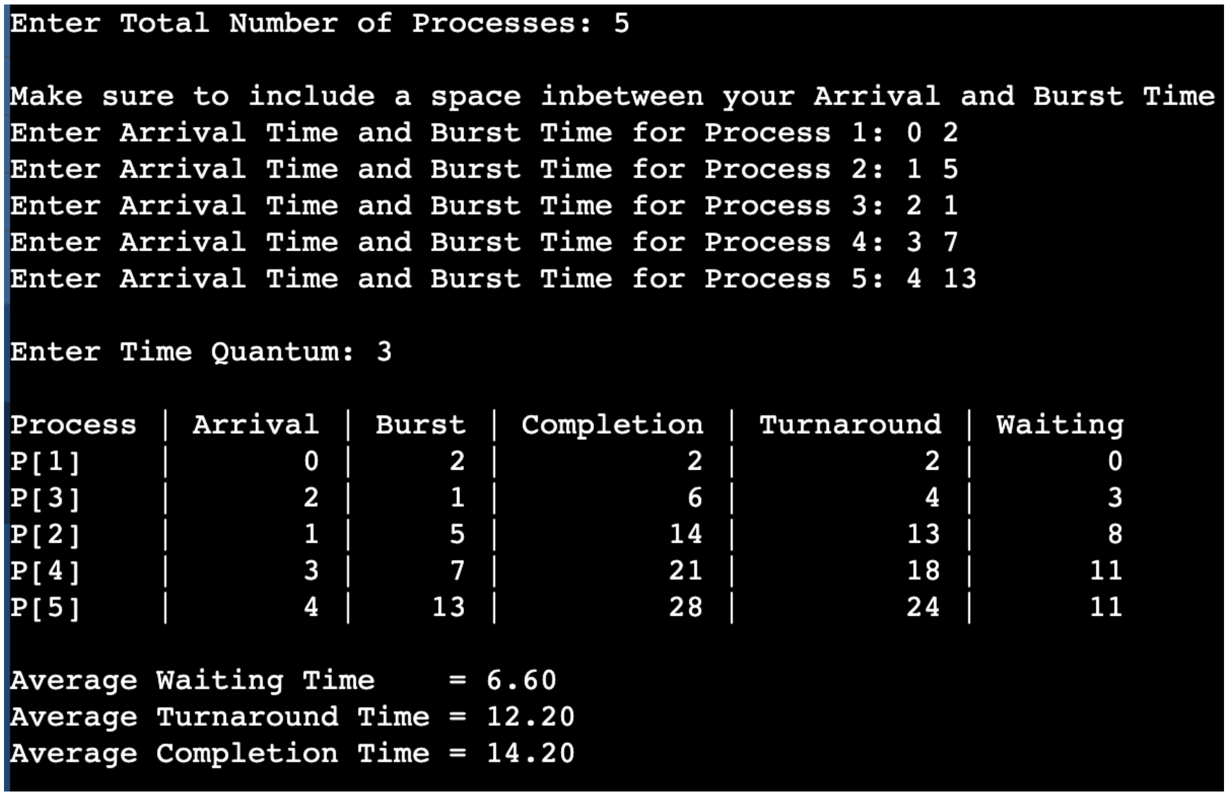
Simulation Result

The results of our respective programs are shown below:



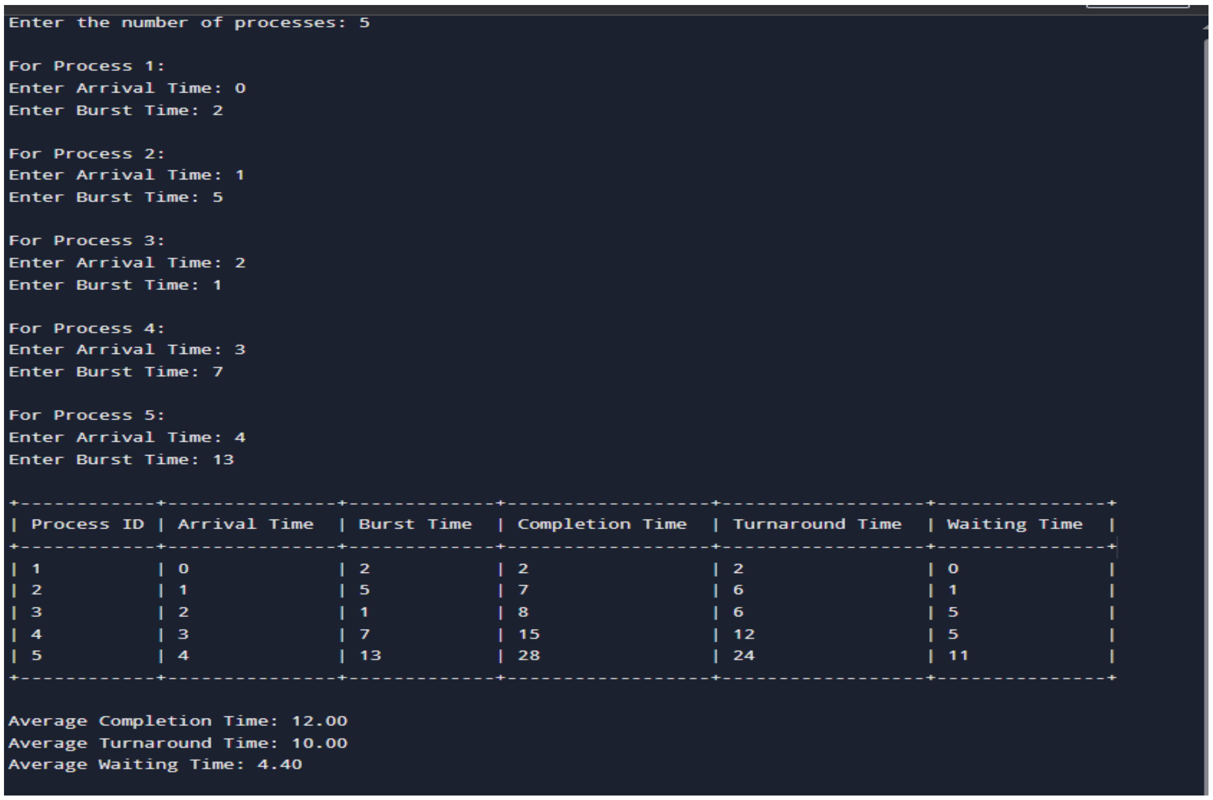
```
Enter Total Number of Processes: 5

Below you wil enter the times, please include a space inbetween.
Enter Arrival Time and Burst Time for Process 1: 0 2
Enter Arrival Time and Burst Time for Process 2: 1 5
Enter Arrival Time and Burst Time for Process 3: 2 1
Enter Arrival Time and Burst Time for Process 4: 3 7
Enter Arrival Time and Burst Time for Process 5: 4 13

Enter Time Quantum: 3

Process | Turnaround Time | Waiting Time
P[1]    |        2        |        0
P[3]    |        4        |        3
P[2]    |       13        |        8
P[4]    |       18        |       11
P[5]    |       24        |       11

Average Waiting Time = 6.60
Average Turnaround Time = 12.20
Average Completion Time = 18.80
```

Fig. 1 Round Robin



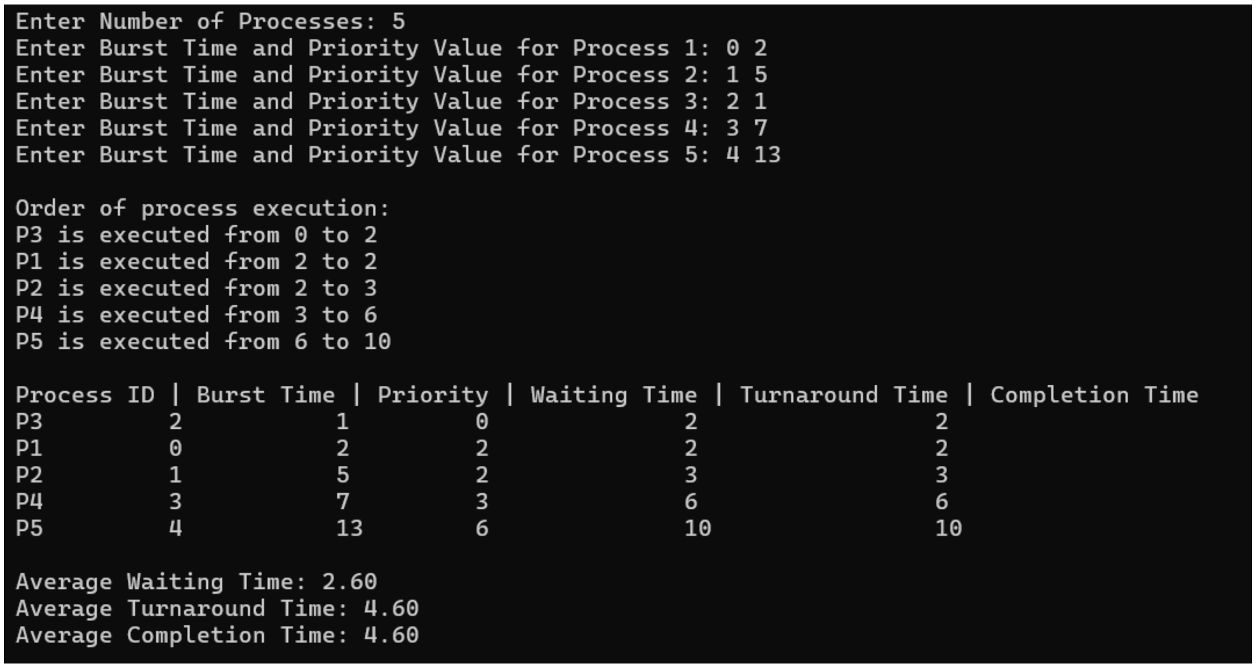
```
Enter Total Number of Processes: 5

Make sure to include a space inbetween your Arrival and Burst Time
Enter Arrival Time and Burst Time for Process 1: 0 2
Enter Arrival Time and Burst Time for Process 2: 1 5
Enter Arrival Time and Burst Time for Process 3: 2 1
Enter Arrival Time and Burst Time for Process 4: 3 7
Enter Arrival Time and Burst Time for Process 5: 4 13

Enter Time Quantum: 3

Process | Arrival | Burst | Completion | Turnaround | Waiting
P[1]    |       0 |     2 |          2 |          2 |       0
P[3]    |       2 |     1 |          6 |          4 |       3
P[2]    |       1 |     5 |         14 |         13 |       8
P[4]    |       3 |     7 |         21 |         18 |      11
P[5]    |       4 |    13 |         28 |         24 |      11

Average Waiting Time    = 6.60
Average Turnaround Time = 12.20
Average Completion Time = 14.20
```

Fig. 2 Round Robin optimized



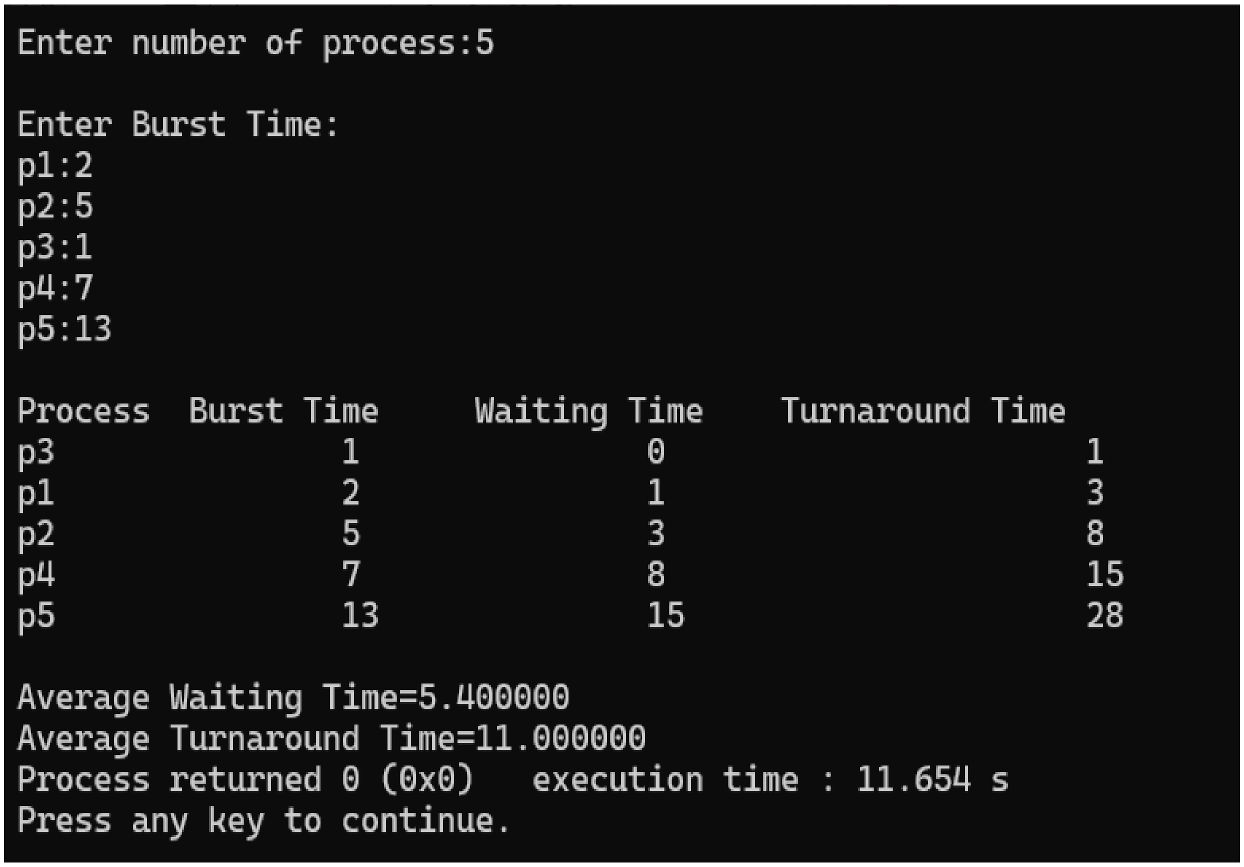
```
Enter the number of processes: 5

For Process 1:
Enter Arrival Time: 0
Enter Burst Time: 2

For Process 2:
Enter Arrival Time: 1
Enter Burst Time: 5

For Process 3:
Enter Arrival Time: 2
Enter Burst Time: 1

For Process 4:
Enter Arrival Time: 3
Enter Burst Time: 7

For Process 5:
Enter Arrival Time: 4
Enter Burst Time: 13

+------------+---------------+-------------+------------------+------------------+--------------+
| Process ID | Arrival Time  | Burst Time  | Completion Time  | Turnaround Time  | Waiting Time  |
+------------+---------------+-------------+------------------+------------------+--------------+
| 1          | 0             | 2           | 2                | 2                | 0            |
| 2          | 1             | 5           | 7                | 6                | 1            |
| 3          | 2             | 1           | 8                | 6                | 5            |
| 4          | 3             | 7           | 15               | 12               | 5            |
| 5          | 4             | 13          | 28               | 24               | 11           |
+------------+---------------+-------------+------------------+------------------+--------------+

Average Completion Time: 12.00
Average Turnaround Time: 10.00
Average Waiting Time: 4.40
```

Fig. 3 First Come, First Serve



```
Enter Number of Processes: 5
Enter Burst Time and Priority Value for Process 1: 0 2
Enter Burst Time and Priority Value for Process 2: 1 5
Enter Burst Time and Priority Value for Process 3: 2 1
Enter Burst Time and Priority Value for Process 4: 3 7
Enter Burst Time and Priority Value for Process 5: 4 13

Order of process execution:
P3 is executed from 0 to 2
P1 is executed from 2 to 2
P2 is executed from 2 to 3
P4 is executed from 3 to 6
P5 is executed from 6 to 10

Process ID | Burst Time | Priority | Waiting Time | Turnaround Time | Completion Time
P3         2            1          0              2                  2
P1         0            2          2              2                  2
P2         1            5          2              3                  3
P4         3            7          3              6                  6
P5         4            13         6              10                 10

Average Waiting Time: 2.60
Average Turnaround Time: 4.60
Average Completion Time: 4.60
```

Fig 4. Priority



```
Enter number of process:5

Enter Burst Time:
p1:2
p2:5
p3:1
p4:7
p5:13

Process  Burst Time     Waiting Time    Turnaround Time
p3              1               0                      1
p1              2               1                      3
p2              5               3                      8
p4              7               8                      15
p5              13              15                     28

Average Waiting Time=5.400000
Average Turnaround Time=11.000000
Process returned 0 (0x0)   execution time : 11.654 s
Press any key to continue.
```

Fig. 5 Shortest Job First

Conclusion

After running the simulations shown above, we were able to assess that while each scheduling algorithm has its own strengths and trade-offs depending on the goal of the system, Priority Scheduling showed better results for overall system efficiency.

Priority scheduling had the lowest waiting time, turnaround time, and completion time out of the 4 scheduling algorithms we tested.

References:

[1] github link: https://github.com/niveprabhu/CPU-Scheduling-and-Optimization-Team-L-

[2] I. Alam, "A Comprehensive Review and Comparative Analysis with Performance Insights and Simulation-Based Evaluations of Innovative CPU Scheduling Algorithms," Academia.edu, https://www.academia.edu/126626328/A_Comprehensive_Review_and_Comparative_Analysis_with_Performance_Insights_and_Simulation_Based_Evaluations_of_Innovative_CPU_Scheduling_Algorithms (accessed 2025).