
# Operating Systems (CS-311)

**Name:** Afnan Bin Abbas

**Reg No.:** 2022048

**Faculty**: Cyber Security

**Submission Date:** 23/10/2024

**Submitted to: Dr. Sarah Iqbal**

# Acknowledgements

I extend my deepest gratitude to Dr. Sara Iqbal, our esteemed mentor and professor, for her invaluable guidance throughout the academic journey. Under her mentorship, I not only solidified our understanding of the subject matter but also learned the importance of striving for excellence in all endeavors.

# Table of Contents

## 0.1 Executive Summary

This report presents a comprehensive analysis of various CPU scheduling algorithms like Shortest Job First (SJF), First Come first SErve (FCFS), Round Robin (RR). The objective of this study is to evaluate the performance and efficiency of First-Come-First-Served (FCFS), Shortest Job First (SJF), and Round Robin (RR) scheduling algorithms in managing CPU tasks. By implementing and comparing these algorithms, we aim to gain insights into their impact on waiting time, turnaround time, and response time.

## 0.2 Explanations of Algorithms

### 1. First Come first Serve (FCFS)

**Usage**: FCFS (First come first serve) is the simplest scheduling algorithm which processes the processes in the order of their arrival. Once a process is running, it will run to completion.
**Advantages**: Easy to implement. All due to the fact that it is not preemptive.
**Disadvantages**: That can cause convoy effect, or short processes wait after longer ones.

### 2. Shortest Job First (SJF)

**Usage**: The process having smallest burst time for execution is selected in SJF, next. It can either be preemptive (Shortest Remaining Time First) or non preemption.
**Advantages**: It reduces waiting time drastically. Average Turnaround time is minimized.
**Disadvantages**: Burst times are difficult to predict in advance. It can cause starve longer processes.

### 3. Round Robin (RR)

**Usage**: Time quantum is the fixed time slice used by RR to let the processes run for a fixed duration and then be preempted and added back to the queue. A process removed from the queue by the arrival of a new process that finishes during the allocated quantum.
**Advantages**: Unbiased for every proccess. Short tasks don't have to wait long.
**Disadvantages**: Switching context occurs too often, and may introduce delays in the longer processes. The smaller the size of the quantum time, the more performance highly depends.

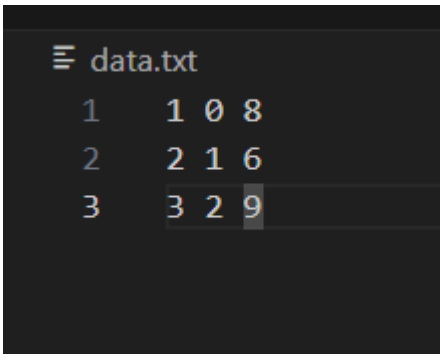
## 0.3 Implementation

**Steps of implementingt the code:**

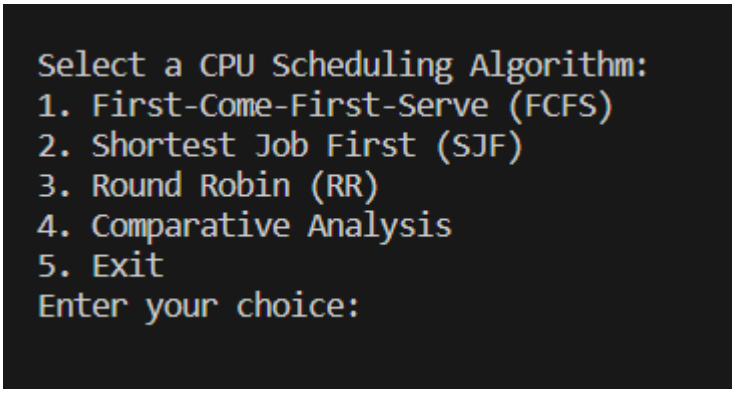
- Create a text file in th esame directory where the project is opened and name it as data.txt.
- Give the process ids, their arival time and bursts in the format (give 1 space between the values):
  1 0 8
  2 1 6
  3 2 9
- Run the Python Code.

### 1. Input and Output Data:

**Scenario#1:**




**Data given**




**Main menu**

```
Select a CPU Scheduling Algorithm:
1. First-Come-First-Serve (FCFS)
2. Shortest Job First (SJF)
3. Round Robin (RR)
4. Comparative Analysis
5. Exit
Enter your choice: 1

                ------------------------------------
                 FIRST COME FIRST SERVED SCHEDULING:
                ------------------------------------
Process 1->  Completion Time = 8, Waiting Time = 0, Turnaround Time = 8
Process 2->  Completion Time = 14, Waiting Time = 7, Turnaround Time = 13
Process 3->  Completion Time = 23, Waiting Time = 12, Turnaround Time = 21
-----------------------------------------------

CPU Utilization: 100.00%
Average Waiting Time: 6.33
Average Turnaround Time: 14.00
```

**FCFS**

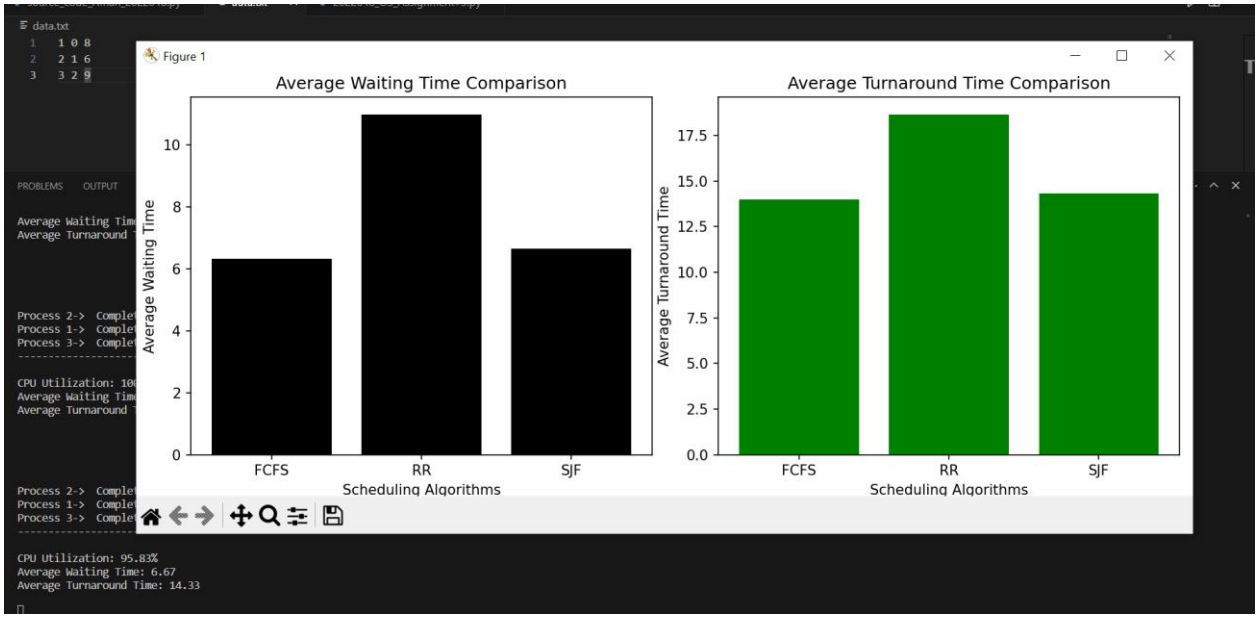
```
Select a CPU Scheduling Algorithm:
1. First-Come-First-Serve (FCFS)
2. Shortest Job First (SJF)
3. Round Robin (RR)
4. Comparative Analysis
5. Exit
Enter your choice: 2

                      ------------------------------
                      SHORTEST JOB FIRST SCHEDULING:
                      ------------------------------
Process 2->  Completion Time = 7, Waiting Time = 0, Turnaround Time = 6
Process 1->  Completion Time = 15, Waiting Time = 7, Turnaround Time = 15
Process 3->  Completion Time = 24, Waiting Time = 13, Turnaround Time = 22
-----------------------------------------------

CPU Utilization: 95.83%
Average Waiting Time: 6.67
Average Turnaround Time: 14.33
```

**SJFS**

```
Select a CPU Scheduling Algorithm:
1. First-Come-First-Serve (FCFS)
2. Shortest Job First (SJF)
3. Round Robin (RR)
4. Comparative Analysis
5. Exit
Enter your choice: 3
Enter the Time Quantum for Round Robin Scheduling: 2

                    ---------------------------
                      ROUND ROBIN SCHEDULING:
                    ---------------------------
Process 2->  Completion Time = 16, Waiting Time = 9, Turnaround Time = 15
Process 1->  Completion Time = 20, Waiting Time = 12, Turnaround Time = 20
Process 3->  Completion Time = 23, Waiting Time = 12, Turnaround Time = 21
-----------------------------------------------

CPU Utilization: 100.00%
Average Waiting Time: 11.00
Average Turnaround Time: 18.67
```

**RR, qt=2**

## 2. Comparative Analysis:

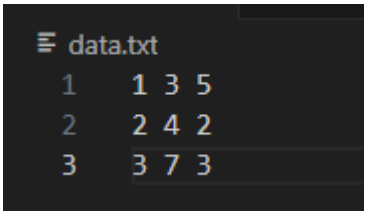



**Black = Average waiting time grpah, Green=turnaround time graph**

```
Select a CPU Scheduling Algorithm:
1. First-Come-First-Serve (FCFS)
2. Shortest Job First (SJF)
3. Round Robin (RR)
4. Comparative Analysis
5. Exit
Enter your choice: 4

                 ------------------------------------
                  FIRST COME FIRST SERVED SCHEDULING:
                 ------------------------------------
Process 1->  Completion Time = 8, Waiting Time = 0, Turnaround Time = 8
Process 2->  Completion Time = 14, Waiting Time = 7, Turnaround Time = 13
Process 3->  Completion Time = 23, Waiting Time = 12, Turnaround Time = 21
-----------------------------------------------

CPU Utilization: 100.00%
Average Waiting Time: 6.33
Average Turnaround Time: 14.00


                       ---------------------------
                         ROUND ROBIN SCHEDULING:
                       ---------------------------
Process 2->  Completion Time = 16, Waiting Time = 9, Turnaround Time = 15
Process 1->  Completion Time = 20, Waiting Time = 12, Turnaround Time = 20
Process 3->  Completion Time = 23, Waiting Time = 12, Turnaround Time = 21
-----------------------------------------------

CPU Utilization: 100.00%
Average Waiting Time: 11.00
Average Turnaround Time: 18.67


                       -------------------------------
                       SHORTEST JOB FIRST SCHEDULING:
                       -------------------------------
Process 2->  Completion Time = 7, Waiting Time = 0, Turnaround Time = 6
Process 1->  Completion Time = 15, Waiting Time = 7, Turnaround Time = 15
Process 3->  Completion Time = 24, Waiting Time = 13, Turnaround Time = 22
-----------------------------------------------

CPU Utilization: 95.83%
Average Waiting Time: 6.67
Average Turnaround Time: 14.33
```

**Comparative analysis Logs created**

**Scenario#2:**




**Data**

```
Select a CPU Scheduling Algorithm:
1. First-Come-First-Serve (FCFS)
2. Shortest Job First (SJF)
3. Round Robin (RR)
4. Comparative Analysis
5. Exit
Enter your choice: 1

                ------------------------------------
                 FIRST COME FIRST SERVED SCHEDULING:
                ------------------------------------
Process 1->  Completion Time = 8, Waiting Time = 0, Turnaround Time = 5
Process 2->  Completion Time = 10, Waiting Time = 4, Turnaround Time = 6
Process 3->  Completion Time = 13, Waiting Time = 3, Turnaround Time = 6
-----------------------------------------------

CPU Utilization: 76.92%
Average Waiting Time: 2.33
Average Turnaround Time: 5.67
```

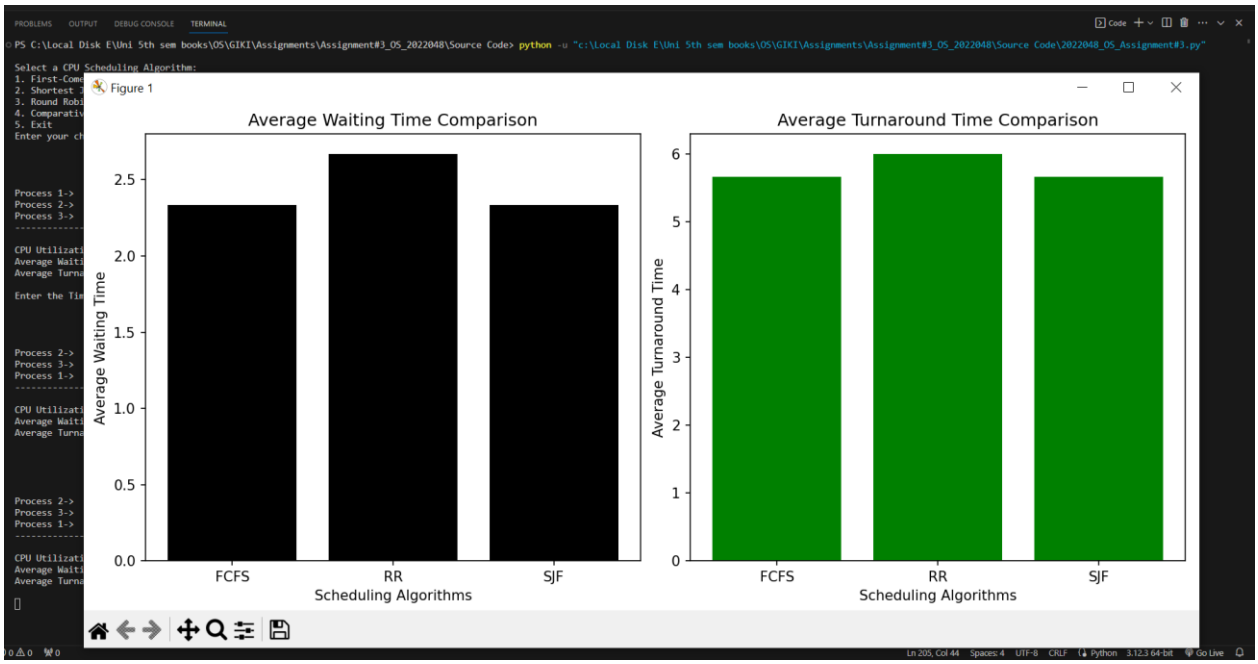
**FCFS**

```
Select a CPU Scheduling Algorithm:
1. First-Come-First-Serve (FCFS)
2. Shortest Job First (SJF)
3. Round Robin (RR)
4. Comparative Analysis
5. Exit
Enter your choice: 2

                    ------------------------------
                    SHORTEST JOB FIRST SCHEDULING:
                    ------------------------------
Process 2->  Completion Time = 6, Waiting Time = 0, Turnaround Time = 2
Process 3->  Completion Time = 10, Waiting Time = 0, Turnaround Time = 3
Process 1->  Completion Time = 15, Waiting Time = 7, Turnaround Time = 12
-----------------------------------------------

CPU Utilization: 66.67%
Average Waiting Time: 2.33
Average Turnaround Time: 5.67
```

**SJFS**

```
Select a CPU Scheduling Algorithm:
1. First-Come-First-Serve (FCFS)
2. Shortest Job First (SJF)
3. Round Robin (RR)
4. Comparative Analysis
5. Exit
Enter your choice: 3
Enter the Time Quantum for Round Robin Scheduling: 3

                    ---------------------------
                      ROUND ROBIN SCHEDULING:
                    ---------------------------
Process 2->  Completion Time = 8, Waiting Time = 2, Turnaround Time = 4
Process 3->  Completion Time = 11, Waiting Time = 1, Turnaround Time = 4
Process 1->  Completion Time = 13, Waiting Time = 5, Turnaround Time = 10
-----------------------------------------------

CPU Utilization: 76.92%
Average Waiting Time: 2.67
Average Turnaround Time: 6.00
```

**RR, qt=3**



```
Select a CPU Scheduling Algorithm:
1. First-Come-First-Serve (FCFS)
2. Shortest Job First (SJF)
3. Round Robin (RR)
4. Comparative Analysis
5. Exit
Enter your choice: 4

                ------------------------------------
                 FIRST COME FIRST SERVED SCHEDULING:
                ------------------------------------
Process 1->  Completion Time = 8, Waiting Time = 0, Turnaround Time = 5
Process 2->  Completion Time = 10, Waiting Time = 4, Turnaround Time = 6
Process 3->  Completion Time = 13, Waiting Time = 3, Turnaround Time = 6
-----------------------------------------------

CPU Utilization: 76.92%
Average Waiting Time: 2.33
Average Turnaround Time: 5.67

Enter the Time Quantum for Round Robin Scheduling: 2

                      ---------------------------
                        ROUND ROBIN SCHEDULING:
                      ---------------------------
Process 2->  Completion Time = 7, Waiting Time = 1, Turnaround Time = 3
Process 3->  Completion Time = 12, Waiting Time = 2, Turnaround Time = 5
Process 1->  Completion Time = 13, Waiting Time = 5, Turnaround Time = 10
-----------------------------------------------

CPU Utilization: 76.92%
Average Waiting Time: 2.67
Average Turnaround Time: 6.00


                      ------------------------------
                      SHORTEST JOB FIRST SCHEDULING:
                      ------------------------------
Process 2->  Completion Time = 6, Waiting Time = 0, Turnaround Time = 2
Process 3->  Completion Time = 10, Waiting Time = 0, Turnaround Time = 3
Process 1->  Completion Time = 15, Waiting Time = 7, Turnaround Time = 12
-----------------------------------------------

CPU Utilization: 66.67%
Average Waiting Time: 2.33
Average Turnaround Time: 5.67
```

**Comparative analysis Logs created**

## 3. Source Code:

2022048_OS_Assignment#3.py > main

```python
# Importing the necessary library for generating comparative charts
import matplotlib.pyplot as plt

# Defining the Process class with attributes such as process_id, arrival_time, and burst_time
class Process:
    def __init__(self, process_id, arrival_time, burst_time):
        self.process_id = process_id  # ID of the process
        self.arrival_time = arrival_time  # Time when the process arrives in the queue
        self.burst_time = burst_time  # The duration for which the process needs the CPU
        self.completion_time = 0  # Time when the process finishes execution (initialized to 0)
        self.waiting_time = 0  # Time spent waiting before execution starts (initialized to 0)
        self.turnaround_time = 0  # Total time taken to complete the process (initialized to 0)
        self.remaining_time = burst_time  # Tracks the remaining execution time for Round Robin scheduling

# Function to read processes' details from a text file and return a list of Process objects
def read_processes_from_file(file_path):
    processes = []
    with open(file_path, 'r') as f:
        for line in f:
            pid, at, bt = map(int, line.strip().split())  # Extracting process ID, arrival time, and burst time from each line
            processes.append(Process(pid, at, bt))  # Creating a Process object and adding it to the list
    return processes  # Returning the list of processes

# FCFS (First-Come, First-Served) Scheduling Algorithm
def fcfs_sched(processes):
    print("\n\t\t -----------------------------------")
    print("\t\t  FIRST COME FIRST SERVED SCHEDULING:")
    print("\t\t -----------------------------------")
    processes.sort(key=lambda x: x.arrival_time)  # Sorting processes based on their arrival times
    current_time = 0  # Tracks the ongoing time in the system

    for process in processes:
        current_time = max(current_time, process.arrival_time)  # Ensures the process starts when it arrives or after the previous one ends
        current_time += process.burst_time  # Update the current time by adding the process's burst time
        process.completion_time = current_time  # Assign the completion time to the process
        process.turnaround_time = process.completion_time - process.arrival_time  # Calculate the turnaround time
        process.waiting_time = process.turnaround_time - process.burst_time  # Calculate the waiting time

        # Displaying the metrics for each process
        print(f"Process {process.process_id}->  Completion Time = {process.completion_time}, "
              f"Waiting Time = {process.waiting_time}, Turnaround Time = {process.turnaround_time}")

    # Compute metrics such as CPU Utilization, Average Waiting Time, and Average Turnaround Time
    cpu_util, avg_waiting, avg_turnaround = calculate_metrics(processes)

    # Output the computed metrics
    print("------------------------------------------------")
    print(f"\nCPU Utilization: {cpu_util:.2f}%")
    print(f"Average Waiting Time: {avg_waiting:.2f}")
    print(f"Average Turnaround Time: {avg_turnaround:.2f}\n")

    return avg_waiting, avg_turnaround  # Return average waiting and turnaround times

# Round Robin Scheduling Algorithm
def round_robin_sched(processes, time_quantum):
    print("\n\t\t\t--------------------------")
    print("\t\t\t  ROUND ROBIN SCHEDULING:")
    print("\t\t\t--------------------------")
    queue = processes.copy()  # Copy processes into a queue for round-robin processing
    current_time = 0  # Tracks the current time

    while queue:
        current_process = queue.pop(0)  # Fetch the first process in the queue
        current_time = max(current_time, current_process.arrival_time)  # Ensure the current time accounts for process arrival

        if current_process.remaining_time <= time_quantum:  # If remaining time is within the time quantum
            current_time += current_process.remaining_time  # Add remaining time to the current time
            current_process.remaining_time = 0  # Mark the process as finished
        else:
            current_time += time_quantum  # Increment current time by the time quantum
            current_process.remaining_time -= time_quantum  # Deduct time quantum from the remaining time
            queue.append(current_process)  # Add the process back to the queue if it's not finished

        if current_process.remaining_time == 0:  # Process has finished execution
            current_process.completion_time = current_time  # Set its completion time
            current_process.turnaround_time = current_process.completion_time - current_process.arrival_time  # Calculate turnaround time
            current_process.waiting_time = current_process.turnaround_time - current_process.burst_time  # Calculate waiting time

            # Output process details
```

2022048_OS_Assignment#3.py > main

```python
            print(f"Process {current_process.process_id}->  Completion Time = {current_process.completion_time}, "
                  f"Waiting Time = {current_process.waiting_time}, Turnaround Time = {current_process.turnaround_time}")

    # Compute CPU Utilization, Average Waiting Time, and Turnaround Time
    cpu_util, avg_waiting, avg_turnaround = calculate_metrics(processes)

    # Display the results
    print("------------------------------------------------")
    print(f"\nCPU Utilization: {cpu_util:.2f}%")
    print(f"Average Waiting Time: {avg_waiting:.2f}")
    print(f"Average Turnaround Time: {avg_turnaround:.2f}\n")

    return avg_waiting, avg_turnaround  # Return average waiting and turnaround times

# Shortest Job First (SJF) Scheduling Algorithm
def shortest_job_first_sched(processes):
    print("\n\t\t\t------------------------------")
    print("\t\t\tSHORTEST JOB FIRST SCHEDULING:")
    print("\t\t\t------------------------------")
    queue = processes.copy()  # Clone the list of processes
    current_time = 0  # Start tracking the current time

    while queue:
        queue.sort(key=lambda x: (x.burst_time, x.arrival_time))  # Sort processes by burst time (and arrival time for tie-breaking)
        current_process = queue.pop(0)  # Pick the process with the shortest burst time
        current_time = max(current_time, current_process.arrival_time)  # Ensure the process starts at the right time
        current_time += current_process.burst_time  # Add the burst time to the current time
        current_process.completion_time = current_time  # Record the completion time
        current_process.turnaround_time = current_process.completion_time - current_process.arrival_time  # Calculate turnaround time
        current_process.waiting_time = current_process.turnaround_time - current_process.burst_time  # Calculate waiting time

        # Output process-specific results
        print(f"Process {current_process.process_id}->  Completion Time = {current_process.completion_time}, "
              f"Waiting Time = {current_process.waiting_time}, Turnaround Time = {current_process.turnaround_time}")

    # Compute CPU Utilization and average time metrics
    cpu_util, avg_waiting, avg_turnaround = calculate_metrics(processes)

    # Display the metrics
    print("------------------------------------------------")
    print(f"\nCPU Utilization: {cpu_util:.2f}%")
```

2022048_OS_Assignment#3.py > main

```python
    print(f"\nCPU Utilization: {cpu_util:.2f}%")
    print(f"Average Waiting Time: {avg_waiting:.2f}")
    print(f"Average Turnaround Time: {avg_turnaround:.2f}\n")

    return avg_waiting, avg_turnaround  # Return the average waiting and turnaround times

# Function to calculate CPU Utilization, Average Waiting Time, and Average Turnaround Time
def calculate_metrics(processes):
    total_waiting_time = 0  # Total waiting time across all processes
    total_turnaround_time = 0  # Total turnaround time across all processes
    total_cpu_burst_time = 0  # Total CPU burst time for all processes
    num_processes = len(processes)  # Count of processes

    for process in processes:
        total_waiting_time += process.waiting_time  # Accumulate the waiting time for each process
        total_turnaround_time += process.turnaround_time  # Accumulate the turnaround time for each process
        total_cpu_burst_time += process.burst_time  # Sum the burst times for CPU utilization calculation

    avg_waiting_time = total_waiting_time / float(num_processes)  # Average waiting time
    avg_turnaround_time = total_turnaround_time / float(num_processes)  # Average turnaround time

    total_time = max([p.completion_time for p in processes])  # The total time for all processes to complete
    cpu_utilization = (total_cpu_burst_time / total_time) * 100  # CPU Utilization percentage

    return cpu_utilization, avg_waiting_time, avg_turnaround_time  # Return calculated metrics

# Function to generate comparative graphs for average waiting time and turnaround time
def plot_comparative_graphs(algorithms, waiting_times, turnaround_times):
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))  # Create two side-by-side plots

    # Bar chart comparing Average Waiting Times
    ax1.bar(algorithms, waiting_times, color='black')
    ax1.set_title('Average Waiting Time Comparison')
    ax1.set_xlabel('Scheduling Algorithms')
    ax1.set_ylabel('Average Waiting Time')

    # Bar chart comparing Average Turnaround Times
    ax2.bar(algorithms, turnaround_times, color='green')
    ax2.set_title('Average Turnaround Time Comparison')
    ax2.set_xlabel('Scheduling Algorithms')
    ax2.set_ylabel('Average Turnaround Time')

    plt.tight_layout()  # Adjust layout for better spacing
```

2022048_OS_Assignment#3.py > main

```python
    plt.show()  # Display the generated graphs

# Function to reset the remaining time for Round Robin
def reset_remaining_time(processes):
    for process in processes:
        process.remaining_time = process.burst_time  # Reset the remaining time to the burst time

# Main function to execute the scheduling algorithms and plot results
def main():
    file_path = 'data.txt'  # Path to the file containing process data
    processes = read_processes_from_file(file_path)  # Reading the processes from the file

    while True:
        print("\nSelect a CPU Scheduling Algorithm:")
        print("1. First-Come-First-Serve (FCFS)")
        print("2. Shortest Job First (SJF)")
        print("3. Round Robin (RR)")
        print("4. Comparative Analysis")
        print("5. Exit")

        option = int(input("Enter your choice: "))

        if option == 1:
            # Running FCFS scheduling algorithm
            fcfs_waiting, fcfs_turnaround = fcfs_sched(processes.copy())

        elif option == 2:
            # Running SJF scheduling algorithm
            sjf_waiting, sjf_turnaround = shortest_job_first_sched(processes.copy())

        elif option == 3:
            # Ask user for the time quantum
            time_quantum = int(input("Enter the Time Quantum for Round Robin Scheduling: "))
            reset_remaining_time(processes)  # Reset remaining time for Round Robin
            rr_waiting, rr_turnaround = round_robin_sched(processes.copy(), time_quantum)

        elif option == 4:
            # Run all scheduling algorithms for comparative analysis
            fcfs_waiting, fcfs_turnaround = fcfs_sched(processes.copy())

            time_quantum = int(input("Enter the Time Quantum for Round Robin Scheduling: "))
            reset_remaining_time(processes)  # Reset remaining time for Round Robin
            rr_waiting, rr_turnaround = round_robin_sched(processes.copy(), time_quantum)

                sjf_waiting, sjf_turnaround = shortest_job_first_sched(processes.copy())

                # Algorithm names and respective metrics for plotting
                algorithms = ['FCFS', 'RR', 'SJF']
                waiting_times = [fcfs_waiting, rr_waiting, sjf_waiting]
                turnaround_times = [fcfs_turnaround, rr_turnaround, sjf_turnaround]

                # Plotting the comparative graphs
                plot_comparative_graphs(algorithms, waiting_times, turnaround_times)

            elif option == 5:
                print("Exiting...")
                break
            else:
                print("Invalid choice. Please try again.")

# Entry point of the program
if __name__ == "__main__":
    main()  # Execute the main function
```

## 0.4 Conclusio

**Observations:**

- With interprocesses varying in arrival time, FCFS always has the highest average waiting and turnaround times because processes that arrive at a later time could have to wait long periods. For shorter processes SJF does best in terms with waiting and turnaround times and is generally the better strategy. But it also means that longer processes will starve. Using Round robin with smaller time quantum means less waiting time since there are almost no context switches, but lacks fairness since CPU time is shared equally.

- **Most Optimized Algorithm Justification:** In cases where the burst times are shorter, as is the case in SJF, its waiting and turnaround times are minimum. Round Robin is fair and minimize the starvation of longer processes with the correct time quan- tum.

## 0.5 References

https://github.com/

https://chat.openai.com/ -> for code optimization and editing

Thank you for having this much patience of reaching to this page.