A Project Report
On

CPU Scheduling Algorithms with a new proposed algorithm

For

Algorithm and Problem Solving Lab (15B17CI471)

Submitted by:

Aryan Patel(9921103156)
Satyam Gupta(9921103139)
Shivansh Pandey(9921103152)
Nitin Chaudhary(9921103163)

Submitted to:
Dr. Nitin Shukla
Dr. Neeraj Jain

Department of CSE/IT
Jaypee Institute of Information Technology University, Noida

May, 2023

TABLE OF CONTENT

1. Problem statement……....3

2. Introduction……...4

2.1 Motivation……...5
2.2 Objective ……5
2.3 Contribution……6

3. Detailed description of the project.……7

4. Implementation

4.1Formula Used…….……...9
4.2Workflow diagram……11

5. Results with snapshot of output ……...13

6. Conclusion…….……10

7. References……...23

Problem Statement

Effective CPU scheduling algorithms are essential for maximizing CPU utilization, throughput, and minimizing waiting time, response time, and turnaround time. With many CPU scheduling algorithms available, it can be challenging to determine the best algorithm for a given scheduling task with specific arrival time, burst time, and priority. The need for the best algorithm has led to the problem of finding the most suitable CPU scheduling algorithm.

To address this issue, our group aims to compare various CPU scheduling algorithms and propose a modified algorithm based on the Shortest Remaining Time First and Round Robin algorithm. Our objective is to design a new algorithm that provides optimal performance in all possible test cases.

Introduction

CPU scheduling is a critical component of modern operating systems that helps to manage the allocation of system resources efficiently. Selecting the best CPU scheduling algorithm is crucial in achieving optimal system performance. While there are many CPU scheduling algorithms available, it is difficult to select the most appropriate algorithm for a given scheduling task with a specific set of input parameters, such as Arrival time, Burst time, and Priority.

Therefore, the main objective of our project is to compare and evaluate different CPU scheduling algorithms and propose a new modified CPU scheduling algorithm based on the Shortest Remaining Time First and Round Robin algorithm. Our aim is to design an algorithm that will perform well on average across all possible test cases.

We plan to achieve this objective by first studying and analyzing the different CPU scheduling algorithms, including FCFS, SJF, Priority, SRTF, Priority Non-Preemptive, and Round Robin. Then, we will compare their performance in terms of metrics such as CPU utilization, throughput, waiting time, response time, and turnaround time.

Next, we will propose a new modified CPU scheduling algorithm that dynamically calculates the time quantum using a mean formula based on the average mean value of all burst times, the weighted average mean value of all burst times considering priorities as weights, the harmonic mean of all burst times, or the weighted harmonic mean of all burst times considering priorities as weights.

We will implement the new algorithm using the C++ programming language and use the C++ set data structure to implement the ready queue. This will allow us to sort the processes in the ready queue based on their burst time and assign priority to the process with minimum remaining burst time to be processed first. Finally, we will evaluate the performance of our proposed algorithm against the existing CPU scheduling algorithms and analyze its effectiveness based on various metrics. We anticipate that our proposed algorithm will show better performance compared to the existing algorithms in terms of average turnaround time, average waiting time, and average response time.

Motivation

As a group of computer science students, we recognize the critical importance of CPU scheduling algorithms in determining the performance and efficiency of computer systems. With so many different algorithms available, selecting the most appropriate algorithm for a particular system or application can be a challenging task.

Therefore, we have decided to undertake this project to gain a deeper understanding of CPU scheduling algorithms and evaluate their performance in various scenarios. By studying six commonly used algorithms, including FCFS, SJF, SRTF, Priority, Priority Preemptive, and Round Robin, we aim to identify the strengths and weaknesses of each algorithm and determine their suitability for different types of systems and applications.

Ultimately, by improving our understanding of CPU scheduling algorithms, we hope to make informed decisions when designing computer systems or developing software applications. We believe that this project will provide us with a unique opportunity to work collaboratively, apply our knowledge of computer science, and contribute to the ongoing research in this critical area.

Objective

The primary objective of our project is to propose a new CPU scheduling algorithm, which represents an improved version of Round Robin. The proposed algorithm dynamically calculates the time quantum using a mean formula, specifically, either the average mean value of all burst times, the weighted average mean value of all burst times with consideration of priorities as weights, the harmonic mean of all burst times, the weighted harmonic mean of all burst times with consideration of priorities as weights, or the mean of all means that includes the average mean, weighted average mean, harmonic mean, and weighted harmonic mean to calculate the dynamic time quantum.

Our implementation leverages a C++ set to manage the ready queue, where processes automatically get sorted based on the property of the binary search tree. As a result, all processes in the ready queue get sorted based on their burst time, allowing the algorithm to prioritize processes with the minimum remaining burst time for execution.

In evaluating the proposed algorithm, we compare it against six commonly used scheduling algorithms, including FCFS, SJF, SRTF, Priority, Priority Preemptive, and Round Robin. Our evaluation focuses on metrics such as average turnaround time, average waiting time, and average response time. The

experimental results reveal significant improvements in some cases, demonstrating the efficacy of our proposed algorithm in enhancing the performance of computer systems.

Through this project, we aim to contribute to the field of CPU scheduling algorithms and provide a valuable resource for researchers and professionals alike. By leveraging our knowledge and skills in computer science, we aspire to create a new algorithm that can improve the performance and efficiency of computer systems, thereby enhancing the user experience and enabling new applications and possibilities.

Contribution

Every group member contributes equally in the entire project.
We have also taken help from Nitin Shukla Sir and Neeraj Jain Sir whenever we felt any difficulties.

Description

In this section we have explained the related works for various types of CPU Scheduling. There are some CPU Scheduling algorithms for CPU Scheduling with varying complexity and effectiveness. All are discussed in brief below,

1. **First-Come, First-Served (FCFS) Scheduling :** It is the most simple scheduling technique. Processes are merely added to the ready queue using FCFS in the order that they arrive. In this approach, when a process is allocated CPU time, no other processes can obtain CPU time until the current process has finished.This characteristic of FCFS is known as Convoy Effect. The operating system becomes slower because of some slow processes.

2. **Non-preemptive Shortest-Job-First(SJF) scheduling:** The waiting process with the shortest execution time is chosen to execute first via the scheduling technique known as"shortest job first" (SJF). The advantage of adopting Shortest Job First is that, among all scheduling techniques, it has the shortest average waiting time. Starvation could occur if shorter processes are developed indefinitely. This problem can be resolved by using the concept of aging.

3. **Shortest-Job-First (SRTF) Preemptive Scheduling:** The "Shortest Job First" (SJF) algorithm chooses the process that will take the least time to complete for the next execution. This is called also (SRTF) . In this algorithm, jobs are added to the ready queue as they are received. The process with the shortest burst time begins to operate. If a process with a shorter burst time enters the system, the existing one is stopped or prohibited from continuing until the shorter job receives a CPU cycle.

4. **Non-preemptive Priority Scheduling:** One of the most widely used scheduling algorithms.Each procedure is given a priority. The process with the highest priority should be completed first, and so on. Similar priority processes are carried out in the order that they are received. In this type of scheduling strategy, the CPU has been allocated to a specific process. The CPU will become free when the activity using it changes context or Stops.

5. **Preemptive Priority Scheduling:** Priority scheduling refers to the scheduling of processes based on priority. In this algorithm, the scheduler selects the tasks to finish based on priority. Preemptive Scheduling often assigns tasks in accordance with their priority. On occasion, it's crucial to do the higher priority action first, even when the lower priority operation is still running. The lower priority task will halt for a while before continuing after the higher priority action has concluded its execution.

6. **Round Robin (RR) Scheduling:** Every task in a round-robin CPU schedule gets the same amount of processing time. Jobs are first placed in a circular queue, where they are shifted to the front of the queue when a new task is added and to the rear of the queue when their given CPU time expires. It is also possible to assign a variable maximum CPU time to each process. It is mostly used by programs and operating systems that manage resource requests from several consumers. To ensure that all processes and applications can access the same resources in the same amount of time and experience the same waiting time each cycle, all requests are handled in first-in, first-out (FIFO) order. Priority is avoided. In terms of average response time, it performs best. It works best for client-server architecture, interactive systems, and time-sharing systems. Time quantum has a significant impact on its performance.

The primary objective of our project is to propose a new CPU scheduling algorithm, which represents an improved version of Round Robin. The proposed algorithm dynamically calculates the time quantum using a mean formula, specifically, either the average mean value of all burst times, the weighted average mean value of all burst times with consideration of priorities as weights, the harmonic mean of all burst times, the weighted harmonic mean of all burst times with consideration of priorities as weights, or the mean of all means that includes the average mean, weighted average mean, harmonic mean, and weighted harmonic mean to calculate the dynamic time quantum.

Implementation

In this project, our proposed CPU scheduling algorithm is based on SRTF and Round Robin CPU scheduling algorithm. We calculated the time quantum as the formula at the beginning. We took the using a mean formula, specifically, either the average mean value of all burst times, the weighted average mean value of all burst times with consideration of priorities as weights, the harmonic mean of all burst times, the weighted harmonic mean of all burst times with consideration of priorities as weights, or the mean of all means that includes the average mean, weighted average mean, harmonic mean, and weighted harmonic mean to calculate the dynamic time quantum.

$$ArithmeticMean = \frac{\sum_{1}^{n} Burst\ time[i]}{n}\ where\ n\ is\ the\ number\ of\ processes$$

$$Harmonic\ Mean = \frac{n}{\frac{1}{\sum_{1}^{n} Burst\ time[i]}}$$

Proposed Algorithm

Pseudocode of our proposed algorithm:

1. Calculate the arithmetic mean, harmonic mean and find the maxburst time from the burst time of the processes.
2. Calculate the TQ = *ArithmeticMean OR WeightedArithmeticMean OR HarmonicMean OR WeightedHarmonicMean OR MeanOfAllMeans.*
3. Store the arrival time and process id in a pair of deque called arrivalQ.
4. Sort the arrivalQ based on the arrival time in ascending order.
5. ReadyQ is a 'set' which is sorted automatically in logN time, thus assigning priority based on the burst time of the processes after each insertion and Works as SJF.
6. Run a loop until ReadyQ is empty and the arrivalQ gets empty.

    6.1. If ReadyQ is not empty then pop the front process from the ReadyQ and reduce the Remaining BurstTime.

    6.2. If the burst time does not become zero, then push the process again in the ReadyQ.

    6.3. Else store current time in the gantt chart as the process gets ended. 6.3.1. Iterate over the arrivalQ.

    6.3.1.1. Check if the arrival time of the upcoming process is less than the current time then push in the ReadyQ.

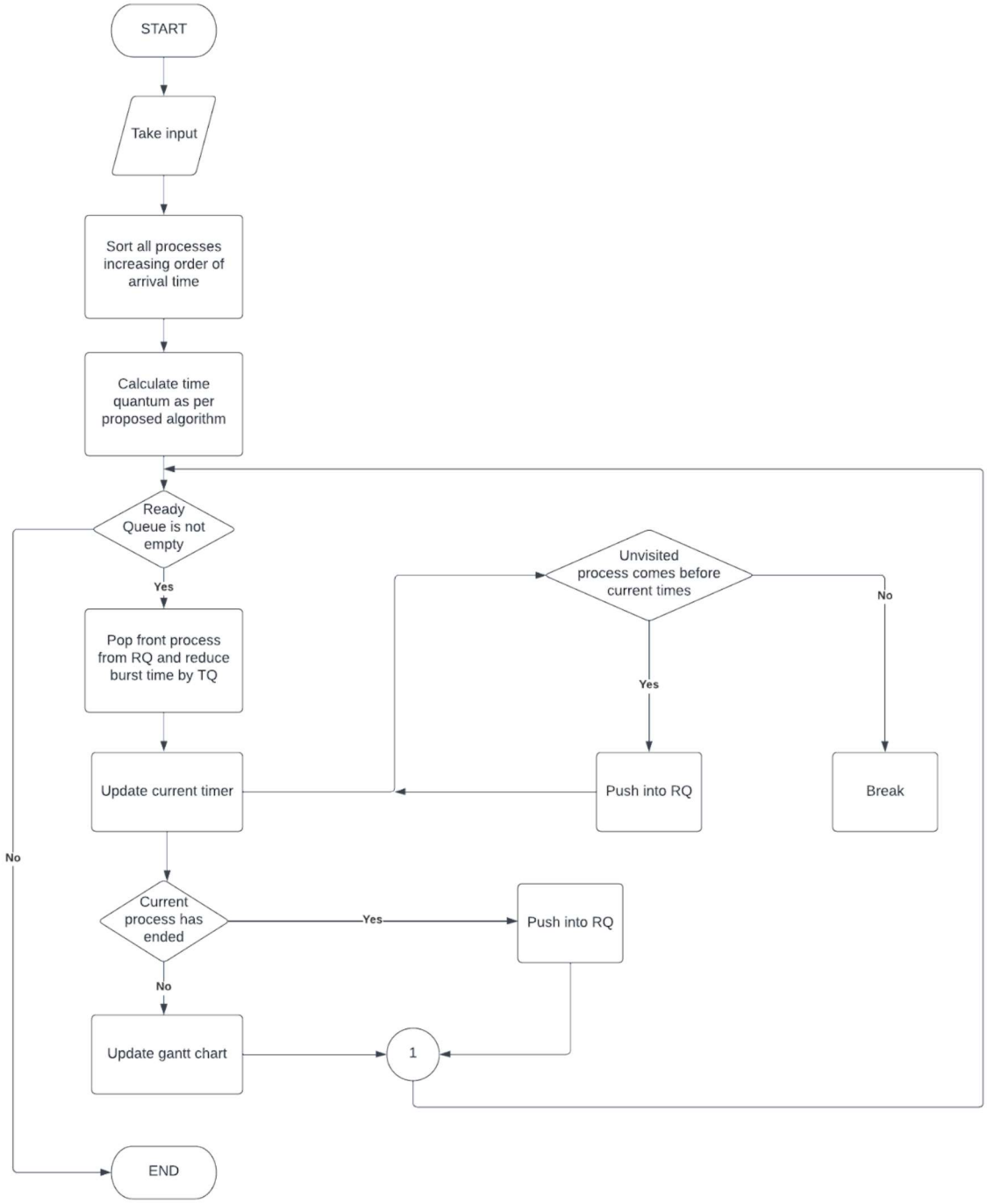
    6.3.1.2.Else break as the arrivalQ is in sorted fashion.

    6.4. Else take the first process with minimum arrival time from the arrivalQ and add the gap to the idle time as it indicates we are taking a process with arrival time greater than current time.

    6.5. Go back to 6. and Repeat

7. Print the answer and End the procedure.

Flowchart


START
Take input
Sort all processes increasing order of arrival time
Calculate time quantum as per proposed algorithm
Ready Queue is not empty
Yes
No
Pop front process from RQ and reduce burst time by TQ
Update current timer
Unvisited process comes before current times
Yes
No
Push into RQ
Break
Current process has ended
Yes
Push into RQ
No
Update gantt chart
1
END

Comparative Analysis:

We have performed operations on our new proposed algorithm by taking different sets of input. Every time it gives accurate results. Considering a set of inputs:

Process	Arrival Time	Burst Time	Priority
p1	3	4	5
p2	56	2	3
p3	1	5	2
p4	10	3	1
p5	5	5	4

Table : Input Test Case

Algorithm Name	Average Waiting Time	Average Turnaround time
FCFS	11.4	15.2
Non Preemptive SJF	11.2	15
Preemptive SJF	44.2	48
Non Preemptive Priority Scheduling	11.4	15.2
Preemptive Priority Scheduling	44.8	48.6
Round Robin	4.4	8.2
Proposed	2.2	6

RESULT OF PROPOSED

FCFS

```
1: FCFS
2: Non-Preemptive-SJF
3: Preemptive-SJF
4: Non-Preemptive-Priority
5: Preemptive-Priority
6: Round-Robin
7: Our-Proposed-AverageMean-Algorithm
8: Our-Proposed-WeightedAverageMean-Algorithm
9: Our-Proposed-HarmonicMean-Algorithm
10: Our-Proposed-WeightedHarmonicMean-Algorithm
11: Our-Proposed-MeanOfAllMeans-Algorithm
12: Compare-All
Choose Option: 1
Number of Process: 4
Enter the arrival time of P1:1
Enter the burst time of P1:2
Enter the arrival time of P2:3
Enter the burst time of P2:1
Enter the arrival time of P3:4
Enter the burst time of P3:7
Enter the arrival time of P4:3
Enter the burst time of P4:9

Gantt Chart
1 P1 3 P2 4 P4 13 P3

Process:P1  Finish Time: 3  Response Time: 0  Waiting Time: 0  Turnaround Time: 2
Process:P2  Finish Time: 4  Response Time: 0  Waiting Time: 0  Turnaround Time: 1
Process:P4  Finish Time: 13  Response Time: 1  Waiting Time: 1  Turnaround Time: 10
Process:P3  Finish Time: 20  Response Time: 9  Waiting Time: 9  Turnaround Time: 16

Average waiting time: 2.5
Average Turnaround time: 7.25
Idle Time: 1

Press Any Key for home page
```

NON PREEMPTIVE SJF

```
1: FCFS
2: Non-Preemptive-SJF
3: Preemptive-SJF
4: Non-Preemptive-Priority
5: Preemptive-Priority
6: Round-Robin
7: Our-Proposed-AverageMean-Algorithm
8: Our-Proposed-WeightedAverageMean-Algorithm
9: Our-Proposed-HarmonicMean-Algorithm
10: Our-Proposed-WeightedHarmonicMean-Algorithm
11: Our-Proposed-MeanOfAllMeans-Algorithm
12: Compare-All
Choose Option: 2
Number of Process: 4
Enter the arrival time of P1:1
Enter the burst time of P1:5
Enter the arrival time of P2:2
Enter the burst time of P2:4
Enter the arrival time of P3:3
Enter the burst time of P3:8
Enter the arrival time of P4:1
Enter the burst time of P4:6

Gantt Chart:
1 P1 6 P2 10 P4 16 P3

Process:P1  Finish Time: 6  Response Time: 0  Waiting Time: 0  Turnaround Time: 5
Process:P2  Finish Time: 10  Response Time: 4  Waiting Time: 4  Turnaround Time: 8
Process:P4  Finish Time: 16  Response Time: 9  Waiting Time: 9  Turnaround Time: 15
Process:P3  Finish Time: 24  Response Time: 13  Waiting Time: 13  Turnaround Time: 21

Average waiting time: 6.5
Average Turnaround time: 12.25
Idle Time: 1

Press Any Key for home page
```

PREEMPTIVE SJF

```
1: FCFS
2: Non-Preemptive-SJF
3: Preemptive-SJF
4: Non-Preemptive-Priority
5: Preemptive-Priority
6: Round-Robin
7: Our-Proposed-AverageMean-Algorithm
8: Our-Proposed-WeightedAverageMean-Algorithm
9: Our-Proposed-HarmonicMean-Algorithm
10: Our-Proposed-WeightedHarmonicMean-Algorithm
11: Our-Proposed-MeanOfAllMeans-Algorithm
12: Compare-All
Choose Option: 3
Number of Process: 4
Enter the arrival time of P1:1
Enter the burst time of P1:8
Enter the arrival time of P2:3
Enter the burst time of P2:8
Enter the arrival time of P3:2
Enter the burst time of P3:3
Enter the arrival time of P4:4
Enter the burst time of P4:5

Gantt Chart:
1 P1 2 P3 3 P3 4 P3 5 P4 10 P1 17 P2

Process:P3  Finish Time: 5  Response Time: 0  Waiting Time: 0  Turnaround Time: 3
Process:P4  Finish Time: 10  Response Time: 1  Waiting Time: 1  Turnaround Time: 6
Process:P1  Finish Time: 17  Response Time: 0  Waiting Time: 8  Turnaround Time: 16
Process:P2  Finish Time: 25  Response Time: 14  Waiting Time: 14  Turnaround Time: 22

Average waiting time: 5.75
Average Turnaround time: 11.75
Idle Time: 1

Press Any Key for home page
```

NON PREEMPTIVE PRIORITY SCHEDULING

```
7: Our-Proposed-AverageMean-Algorithm
8: Our-Proposed-WeightedAverageMean-Algorithm
9: Our-Proposed-HarmonicMean-Algorithm
10: Our-Proposed-WeightedHarmonicMean-Algorithm
11: Our-Proposed-MeanOfAllMeans-Algorithm
12: Compare-All
Choose Option: 4
Number of Process: 4
Enter the priority of P1:3
Enter the arrival time of P1:1
Enter the burst time of P1:4
Enter the priority of P2:2
Enter the arrival time of P2:3
Enter the burst time of P2:6
Enter the priority of P3:4
Enter the arrival time of P3:8
Enter the burst time of P3:2
Enter the priority of P4:5
Enter the arrival time of P4:2
Enter the burst time of P4:1
1: Higher Number Higher Priority
2: Lower Number Higher Priority
Choose Option: 1

Gantt Chart:
1 P1 5 P4 8 P3 10 P2

Process:P1  Finish Time: 5  Response Time: 0  Waiting Time: 0  Turnaround Time: 4
Process:P4  Finish Time: 6  Response Time: 3  Waiting Time: 3  Turnaround Time: 4
Process:P3  Finish Time: 10  Response Time: 0  Waiting Time: 0  Turnaround Time: 2
Process:P2  Finish Time: 16  Response Time: 7  Waiting Time: 7  Turnaround Time: 13

Average waiting time: 2.5
Average Turnaround time: 5.75
Idle Time: 3
```

PREEMPTIVE PRIORITY SCHEDULING

```
6: Round-Robin
7: Our-Proposed-AverageMean-Algorithm
8: Our-Proposed-WeightedAverageMean-Algorithm
9: Our-Proposed-HarmonicMean-Algorithm
10: Our-Proposed-WeightedHarmonicMean-Algorithm
11: Our-Proposed-MeanOfAllMeans-Algorithm
12: Compare-All
Choose Option: 5
Number of Process: 4
Enter the priority of P1:4
Enter the arrival time of P1:5
Enter the burst time of P1:3
Enter the priority of P2:3
Enter the arrival time of P2:2
Enter the burst time of P2:1
Enter the priority of P3:2
Enter the arrival time of P3:5
Enter the burst time of P3:6
Enter the priority of P4:1
Enter the arrival time of P4:2
Enter the burst time of P4:6
1: Higher Number Higher Priority
2: Lower Number Higher Priority
Choose Option: 1

Gantt Chart:
2 P2 3 P4 5 P1 6 P1 8 P3 14 P4

Process:P2  Finish Time: 3  Response Time: 0  Waiting Time: 0  Turnaround Time: 1
Process:P1  Finish Time: 8  Response Time: 0  Waiting Time: 0  Turnaround Time: 3
Process:P3  Finish Time: 14  Response Time: 3  Waiting Time: 3  Turnaround Time: 9
Process:P4  Finish Time: 19  Response Time: 1  Waiting Time: 11  Turnaround Time: 17

Average waiting time: 3.5
Average Turnaround time: 7.5
Idle Time: 3
```

ROUND ROBIN

```
3: Preemptive-SJF
4: Non-Preemptive-Priority
5: Preemptive-Priority
6: Round-Robin
7: Our-Proposed-AverageMean-Algorithm
8: Our-Proposed-WeightedAverageMean-Algorithm
9: Our-Proposed-HarmonicMean-Algorithm
10: Our-Proposed-WeightedHarmonicMean-Algorithm
11: Our-Proposed-MeanOfAllMeans-Algorithm
12: Compare-All
Choose Option: 6
Number of Process: 4
Enter the arrival time of P1:7
Enter the burst time of P1:1
Enter the arrival time of P2:4
Enter the burst time of P2:3
Enter the arrival time of P3:6
Enter the burst time of P3:1
Enter the arrival time of P4:1
Enter the burst time of P4:3
Enter the time quantum TQ for Round Robin Algorithm: 2

Gantt Chart
1 P4 3 P4 4 P2 6 P3 7 P2 8 P1

Process:P4  Finish Time: 4  Response Time: 0  Waiting Time: 0  Turnaround Time: 3
Process:P3  Finish Time: 7  Response Time: 0  Waiting Time: 0  Turnaround Time: 1
Process:P2  Finish Time: 8  Response Time: 0  Waiting Time: 1  Turnaround Time: 4
Process:P1  Finish Time: 9  Response Time: 1  Waiting Time: 1  Turnaround Time: 2

Average waiting time: 0.5
Average Turnaround time: 2.5
Idle Time: 1

Press Any Key for home page
```

PROPOSED ALGORITHM

```
6: Round-Robin
7: Our-Proposed-AverageMean-Algorithm
8: Our-Proposed-WeightedAverageMean-Algorithm
9: Our-Proposed-HarmonicMean-Algorithm
10: Our-Proposed-WeightedHarmonicMean-Algorithm
11: Our-Proposed-MeanOfAllMeans-Algorithm
12: Compare-All
Choose Option: 11
Number of Process: 4
Enter the priority of P1:5
Enter the arrival time of P1:1
Enter the burst time of P1:2
Enter the priority of P2:4
Enter the arrival time of P2:7
Enter the burst time of P2:1
Enter the priority of P3:3
Enter the arrival time of P3:8
Enter the burst time of P3:9
Enter the priority of P4:2
Enter the arrival time of P4:1
Enter the burst time of P4:15

Gantt Chart
1 P1 3 P4 7 P2 8 P3 12 P3 16 P3 17 P4 21 P4 25 P4

Process:P1  Finish Time: 3  Response Time: 0  Waiting Time: 0  Turnaround Time: 2
Process:P2  Finish Time: 8  Response Time: 0  Waiting Time: 0  Turnaround Time: 1
Process:P3  Finish Time: 17  Response Time: 0  Waiting Time: 0  Turnaround Time: 9
Process:P4  Finish Time: 28  Response Time: 2  Waiting Time: 12  Turnaround Time: 27

Average waiting time: 3
Average Turnaround time: 9.75
Idle Time: 1

Press Any Key for home page
```

COMPARISON RESULT IN CONSOLE

```
12: Compare-All
Choose Option: 12
Number of Process: 5
Enter the priority of P1:5
Enter the arrival time of P1:3
Enter the burst time of P1:4
Enter the priority of P2:3
Enter the arrival time of P2:56
Enter the burst time of P2:2
Enter the priority of P3:2
Enter the arrival time of P3:1
Enter the burst time of P3:5
Enter the priority of P4:1
Enter the arrival time of P4:10
Enter the burst time of P4:3
Enter the priority of P5:4
Enter the arrival time of P5:5
Enter the burst time of P5:5
Enter the time quantum TQ for Round Robin Algorithm: 4
Algorithm: 1    Average waiting time: 11.4      Average Turnaround time: 15.2
Algorithm: 2    Average waiting time: 11.2      Average Turnaround time: 15
Algorithm: 3    Average waiting time: 44.2      Average Turnaround time: 48
Algorithm: 4    Average waiting time: 11.4      Average Turnaround time: 15.2
Algorithm: 5    Average waiting time: 44.8      Average Turnaround time: 48.6
Algorithm: 6    Average waiting time: 4.4       Average Turnaround time: 8.2
Algorithm: 7    Average waiting time: 2.2       Average Turnaround time: 6
Algorithm: 8    Average waiting time: 2.2       Average Turnaround time: 6
Algorithm: 9    Average waiting time: 2.2       Average Turnaround time: 6
Algorithm: 10   Average waiting time: 2.2       Average Turnaround time: 6
Algorithm: 11   Average waiting time: 2.2       Average Turnaround time: 6

Press Any Key for home page
```

Complexity Analysis:

To calculate the time quantum for processes in the ready queue, our implemented algorithm employs a C++ set data structure. When a process is added to the queue, it is automatically sorted based on its burst time, which is a property of the binary search tree used by the C++ set. The insertion or deletion of a process in the set takes logN time, resulting in an overall time complexity of NlogN for N processes in the ready queue.

Furthermore, we used an arrival queue to maintain the order in which processes arrive. When a process arrives, it is added to the arrival queue, and when it is ready to run, it is moved to the ready queue. This process takes N iterations in total for N processes.

In summary, our proposed algorithm utilizes a C++ set for sorting processes in the ready queue by their burst time, resulting in an overall time complexity of O(NlogN). Additionally, we use an arrival queue to maintain the order in which processes arrive, which takes N iterations for N processes. By efficiently managing the order of processes, our algorithm provides an effective solution for CPU scheduling with improved performance.

Conclusion:

In our project work, we have proposed a new CPU scheduling algorithm based on Round Robin that effectively handles scenarios where multiple processes arrive at the ready queue simultaneously. Our algorithm shows better performance compared to other scheduling algorithms in terms of average turnaround time, waiting time, and response time. Despite some limitations, it produces more accurate results and reduces average turnaround and waiting times, making it a promising approach for CPU resource management.

While further enhancements can be made, our implemented algorithm offers several advantages over existing algorithms. It effectively manages CPU resources, reducing waiting time and enhancing overall system performance. The proposed algorithm can be useful in scenarios where multiple processes arrive at the ready queue simultaneously, such as in real-time systems or parallel processing. By reducing the overhead associated with context switching, our algorithm offers improved CPU utilization and faster processing times.

In conclusion, our proposed algorithm offers a competitive advantage over other scheduling techniques for managing CPU resources, particularly in scenarios where multiple processes arrive at the ready queue simultaneously.

References:

[1] M. Babbar, “Scheduling algorithms in operating system,” *Scaler Topics*, 20-Jan-2022. .

[2] L. Williams, “CPU scheduling algorithms in operating systems,” *Guru99*, 31-Jan-2021. [Online]. Available: https://www.guru99.com/cpu-scheduling-algorithms.html. [Accessed: 01-May-2023].

[3] A. Putera and U. Siahaan, “Comparison analysis of CPU scheduling : FCFS, SJF and round robin,” *Osf.io*. [Online]. Available: https://osf.io/preprints/inarxiv/6dq3p/download. [Accessed: 30-Apr-2023].

[4] Wikipedia contributors, “Hybrid Scheduling,” *Wikipedia, The Free Encyclopedia*, 07-Aug-2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Hybrid_Scheduling&oldid=909841583.

[5] R. N. Alhawadi, M. O. Nassar, and O. Al tarawneh, “Hybrid scheduling algorithms using round robin and short job first,” *Warse.org*. [Online]. Available: http://www.warse.org/IJSAIT/static/pdf/Issue/icsic2017sp02.pdf. [Accessed: 04-May-2023].