A Comparative Analysis of CPU Scheduling Algorithm with a Proposed SRTF like CPU Scheduler by Modifying the Time Quantum of Round Robin Algorithm.

Salman Farsi[1] ID:1804102

Md Sajidul Mowla[2] ID:1804100

Md Ali Hasan Emon[3] ID:1804104

Arpita Sarker[4] ID:180499

Fahad Hossain[5] ID:1804103

Md Nahid Imtiaz[6] ID:1804101

Department of Computer Science and Engineering

Chittagong University of Engineering and Technology, Bangladesh.

***Abstract*:** In this paper, we have proposed an algorithm that works based on the Round Robin with modified time quantum and setting priority on the burst time of the process in the ready queue. The basic idea was to calculate the arithmetic mean, harmonic mean, maximum burst time and then taking the average of these three parameters as time quantum. Thus behaves like preemptive SJF or SRTF. In the SRTF the burst time reduces by 1, but our proposed algorithm reduces the burst time by calculating time quantum before the procedure gets started from a round robin perspective. It performs quite well in most of the input results. Because the time quantum remains the later half of the processes currently on the ready queue. Though it also has some worst-case scenarios, it performs well in many cases. Our proposed algorithm runs in $O(NlogN)$ time complexity overall. However, we have studied some preemptive and non-preemptive CPU scheduling algorithms. Preemptive algorithms including FCFS, SJF, Priority preemptive, and non-preemptive including SRTF, Priority non-preemptive, and Round Robin. FCFS (First Come First Served) assigns the process based on their arrival time. SJF (Shortest Job First) and SRTF (Shortest Remaining Time First) both assign the process based on their burst time but the main difference between them is that SJF is preemptive whereas SRTF is non-preemptive. Priority preemptive and non-preemptive both assign processes based on priority. And Round Robin CPU scheduling algorithm assigns processes based on time quantum which means keeping every process in the CPU for some time which can be a fixed time quantum or dynamic. After this time, the process either goes to the ready queue and waits for another call or ends its execution. Every scheduling algorithm has its advantages and disadvantages.

Introduction:

CPU Scheduling Algorithms determine which processes will be selected to be executed by the CPU, holding the other processes. Selecting the perfect CPU scheduling algorithms is very important as it maximizes CPU utilization. CPU scheduling selects a process among multiple processes and assigns it to be executed by the CPU whether the other processes are waiting in the ready queue. This selection process is done based on some criteria such as arrival time, burst time, etc. CPU Scheduling algorithms are necessary as they maximize CPU utilization, and throughput and minimize waiting time, response time, and turnaround time. To select the best-suited CPU

Scheduling algorithms for our purpose, the CPU scheduling algorithms should be understood clearly. And we have studied here 6 types of CPU scheduling algorithms. Among them, they can be further classified into two types. One is preemptive and another is non-preemptive. Preemptive means it will assign a process to the CPU by keeping the other processes waiting until the completion of the execution of that running process. And non-preemptive means the CPU will not hold the running process till the execution, rather it will switch the processes after a while. Meanwhile, if the process is completed then it will end its execution and leave the CPU, otherwise, it will again go to the ready queue. The preemptive CPU scheduling algorithms we have learned are respectively FCFS (First Come First Served), SJF (Shortest Job First), and Priority preemptive. Among the non-preemptive, we have learned SRTF (Shortest Remaining Time First), Priority non-preemptive, and Round Robin. These CPU scheduling algorithms work based on different criteria, that's why they are good in some scenarios and perform badly in some scenarios. For example, FCFS performs based on the arrival time whereas SJF and SRTF both assign processes based on the burst time. And Priority works based on the priority of the processes and Round Robin works based on the time quantum. It runs a process for a selected amount of time which is known as time quantum and the switching of the processes after every time quantum is called context switching. Among all these CPU scheduling algorithms, Round Robin performs better in the maximum scenario. In most cases, the turnaround time, average waiting time, and average response time are all better in this scheduling algorithm.

And in this paper, we have proposed a new scheduling algorithm, an improved version of Round Robin. This proposed algorithm calculates the time quantum dynamically using a formula. It calculates the average value of all burst times, the highest value of burst time, and the harmonic mean of all burst times and then calculates the average of these values. We used a C++ set to implement the ready queue, when a particular process gets pushed in the ready queue it gets sorted automatically by the property of the binary search tree of C++ set. So all the processes in the ready queue are sorted based on their burst time. Therefore, it assigns a priority that the process with minimum remaining burst time is processed early. And if we look at the result part, we can see a huge improvement in some cases. Average turnaround time, average waiting time, average response time everything is quite better than the previously described scheduling algorithms.

Related work:

In this section related works are explained 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,

a) **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.

b) **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.

c) **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.

d) **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.

e) **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.

f) **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.

Methodology:

In this project, our proposed CPU scheduling algorithm is based on the Round Robin CPU scheduling algorithm. We calculated the time quantum as the formula at the beginning. We took the arithmetic average value of all burst times of the process in the ready queue. Also took the highest value of burst time and harmonic mean of all burst time. The average value of these three values taken as time quantum in our proposed algorithm.

$$Arithmetic\ Mean\ =\ \frac{\sum_{i=1}^{i=n} BurstTime[i]}{n}\ where\ n\ is\ the\ number\ of\ processes$$

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

$$MaximumBurstTime\ =\ Maximum\ of\ BurstTime[i]\ for\ 1\ <=\ i\ <=\ n$$

$$TQ\ =\ \frac{ArithmeticMean + HarmonicMean + MaxBurstTime}{3}$$

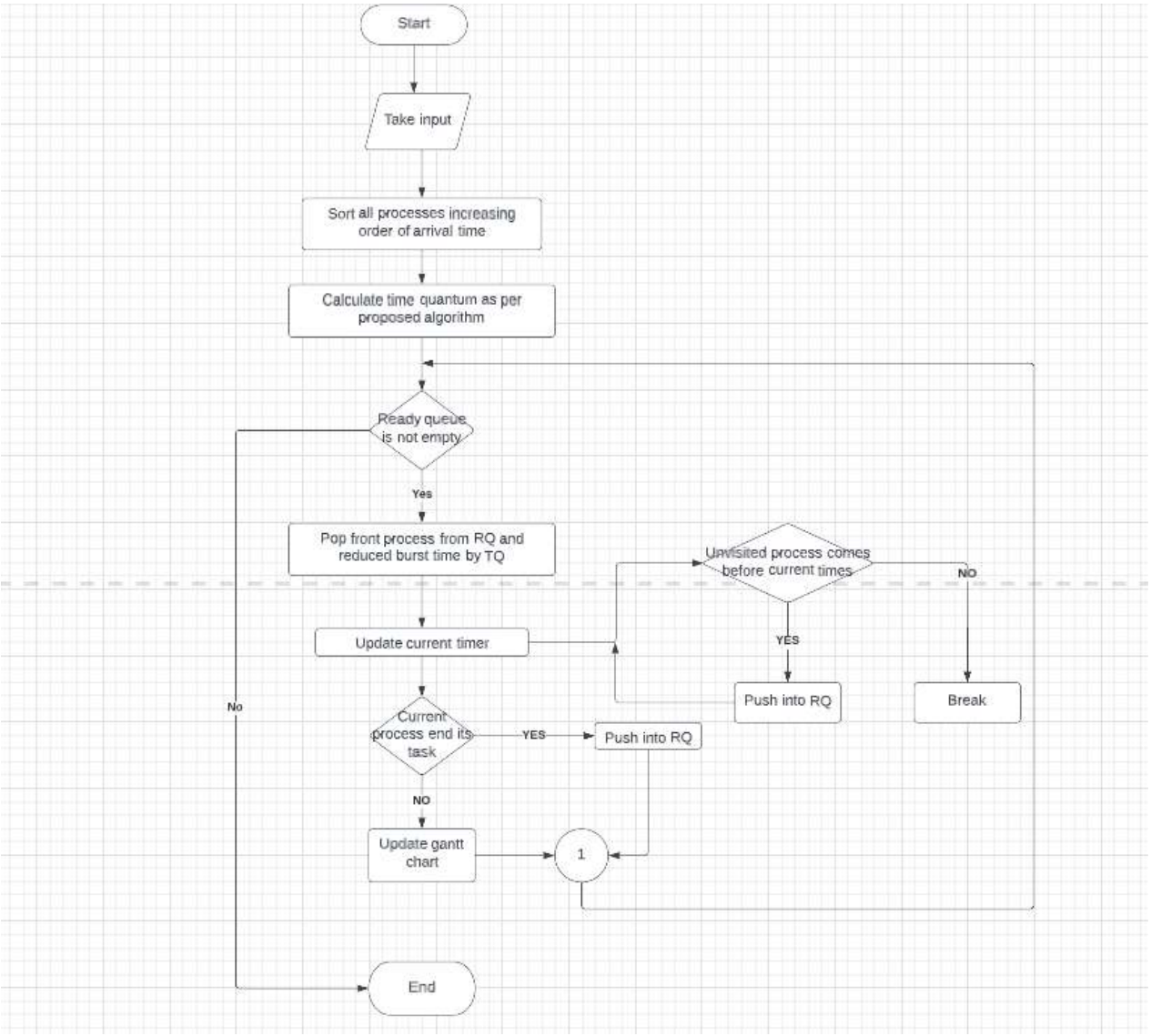
By this process, the time quantum value will be in the last half of the sorted ready queue of burst time. This is because if we would take only the average of arithmetic mean and harmonic mean then TQ would have been somewhere in the midst of the processes currently on the ready queue. So, the context switch will not be frequent and the waiting time decreases. Again, as the time quantum is not very large, so it will avoid FCFS behavior. That is, the chances of starvation is low in reality.

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 =** $\frac{ArithmeticMean + HarmonicMean + MaxBurstTime}{3}$
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 of our proposed algorithm:



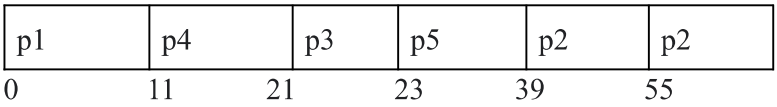
Result:

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
p1	0	11
p2	5	28
p3	12	2
p4	2	10
p5	9	16

Table : Input Test Case

Gantt Chart:



Process	End Time	Waiting Time	Turnaround Time	Response Time
p1	11	0	11	0
p4	21	9	19	9
p3	23	9	11	9
p5	39	14	30	14
p2	67	34	62	34

Table: Comparison between the Scheduling Algorithms

Result of Proposed Algorithm in the Console:

```
1: FCFS
2: Non-Preemptive-SJF
3: Preemptive-SJF
4: Non-Preemptive-Priority
5: Preemptive-Priority
6: Round-Robin
7: Our-Proposed-Algorithm
8: Compare-All
9: Exit
Choose Option: 7
Number of Process: 5
Enter the arrival time of P1:0
Enter the burst time of P1:11
Enter the arrival time of P2:5
Enter the burst time of P2:28
Enter the arrival time of P3:12
Enter the burst time of P3:2
Enter the arrival time of P4:2
Enter the burst time of P4:10
Enter the arrival time of P5:9
Enter the burst time of P5:16

Gantt Chart
0 P1 11 P4 21 P3 23 P5 39 P2 55 P2

Process:P1  Finish Time: 11  Response Time: 0  Waiting Time: 0  Turnaround Time: 11
Process:P4  Finish Time: 21  Response Time: 9  Waiting Time: 9  Turnaround Time: 19
Process:P3  Finish Time: 23  Response Time: 9  Waiting Time: 9  Turnaround Time: 11
Process:P5  Finish Time: 39  Response Time: 14  Waiting Time: 14  Turnaround Time: 30
Process:P2  Finish Time: 67  Response Time: 34  Waiting Time: 34  Turnaround Time: 62

Average waiting time: 13.2
Average Turnaround time: 26.6
Idle Time: 0

Press Any Key for home page
```

As in the project we also implemented all the CPU Scheduling algorithms, therefore we tested the above example for each of the CPU Scheduling Algorithms and got the following results in the console.

FCFS:

```
1: FCFS
2: Non-Preemptive-SJF
3: Preemptive-SJF
4: Non-Preemptive-Priority
5: Preemptive-Priority
6: Round-Robin
7: Our-Proposed-Algorithm
8: Compare-All
9: Exit
Choose Option: 1
Number of Process: 5
Enter the arrival time of P1:0
Enter the burst time of P1:11
Enter the arrival time of P2:5
Enter the burst time of P2:28
Enter the arrival time of P3:12
Enter the burst time of P3:2
Enter the arrival time of P4:2
Enter the burst time of P4:10
Enter the arrival time of P5:9
Enter the burst time of P5:16

Gantt Chart
0 P1 11 P4 21 P2 49 P5 65 P3

Process:P1  Finish Time: 11  Response Time: 0  Waiting Time: 0  Turnaround Time: 11
Process:P4  Finish Time: 21  Response Time: 9  Waiting Time: 9  Turnaround Time: 19
Process:P2  Finish Time: 49  Response Time: 16  Waiting Time: 16  Turnaround Time: 44
Process:P5  Finish Time: 65  Response Time: 40  Waiting Time: 40  Turnaround Time: 56
Process:P3  Finish Time: 67  Response Time: 53  Waiting Time: 53  Turnaround Time: 55

Average waiting time: 23.6
Average Turnaround time: 37
Idle Time: 0

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-Algorithm
8: Compare-All
9: Exit
Choose Option: 2
Number of Process: 5
Enter the arrival time of P1:0
Enter the burst time of P1:11
Enter the arrival time of P2:5
Enter the burst time of P2:28
Enter the arrival time of P3:12
Enter the burst time of P3:2
Enter the arrival time of P4:2
Enter the burst time of P4:10
Enter the arrival time of P5:9
Enter the burst time of P5:16

Gantt Chart:
0 P1 11 P4 21 P3 23 P5 39 P2

Process:P1  Finish Time: 11  Response Time: 0  Waiting Time: 0  Turnaround Time: 11
Process:P4  Finish Time: 21  Response Time: 9  Waiting Time: 9  Turnaround Time: 19
Process:P3  Finish Time: 23  Response Time: 9  Waiting Time: 9  Turnaround Time: 11
Process:P5  Finish Time: 39  Response Time: 14  Waiting Time: 14  Turnaround Time: 30
Process:P2  Finish Time: 67  Response Time: 34  Waiting Time: 34  Turnaround Time: 62

Average waiting time: 13.2
Average Turnaround time: 26.6
Idle Time: 0

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-Algorithm
8: Compare-All
9: Exit
Choose Option: 3
Number of Process: 5
Enter the arrival time of P1:0
Enter the burst time of P1:11
Enter the arrival time of P2:5
Enter the burst time of P2:28
Enter the arrival time of P3:12
Enter the burst time of P3:2
Enter the arrival time of P4:2
Enter the burst time of P4:10
Enter the arrival time of P5:9
Enter the burst time of P5:16

Gantt Chart:
0 P1 2 P1 5 P1 9 P1 12 P3 14 P1 21 P4 31 P5 47 P2

Process:P3  Finish Time: 14  Response Time: 0  Waiting Time: 0  Turnaround Time: 2
Process:P1  Finish Time: 21  Response Time: 0  Waiting Time: 10  Turnaround Time: 21
Process:P4  Finish Time: 31  Response Time: 19  Waiting Time: 19  Turnaround Time: 29
Process:P5  Finish Time: 47  Response Time: 22  Waiting Time: 22  Turnaround Time: 38
Process:P2  Finish Time: 75  Response Time: 42  Waiting Time: 42  Turnaround Time: 70

Average waiting time: 18.6
Average Turnaround time: 32
Idle Time: 8

Press Any Key for home page
```

Non-Preemptive Priority scheduling:

```
1: FCFS
2: Non-Preemptive-SJF
3: Preemptive-SJF
4: Non-Preemptive-Priority
5: Preemptive-Priority
6: Round-Robin
7: Our-Proposed-Algorithm
8: Compare-All
9: Exit
Choose Option: 4
Number of Process: 5
Enter the priority of P1:2
Enter the arrival time of P1:0
Enter the burst time of P1:11
Enter the priority of P2:0
Enter the arrival time of P2:5
Enter the burst time of P2:28
Enter the priority of P3:3
Enter the arrival time of P3:12
Enter the burst time of P3:2
Enter the priority of P4:1
Enter the arrival time of P4:2
Enter the burst time of P4:10
Enter the priority of P5:4
Enter the arrival time of P5:9
Enter the burst time of P5:16
1: Higher Number Higher Priority
2: Lower Number Higher Priority
Choose Option: 2

Gantt Chart:
0 P1 11 P2 39 P4 49 P3 51 P5

Process:P1  Finish Time: 11  Response Time: 0  Waiting Time: 0  Turnaround Time: 11
Process:P2  Finish Time: 39  Response Time: 6  Waiting Time: 6  Turnaround Time: 34
Process:P4  Finish Time: 49  Response Time: 37  Waiting Time: 37  Turnaround Time: 47
Process:P3  Finish Time: 51  Response Time: 37  Waiting Time: 37  Turnaround Time: 39
Process:P5  Finish Time: 67  Response Time: 42  Waiting Time: 42  Turnaround Time: 58

Average waiting time: 24.4
Average Turnaround time: 37.8
Idle Time: 0

Press Any Key for home page
```

Preemptive Priority scheduling:

```
1: FCFS
2: Non-Preemptive-SJF
3: Preemptive-SJF
4: Non-Preemptive-Priority
5: Preemptive-Priority
6: Round-Robin
7: Our-Proposed-Algorithm
8: Compare-All
9: Exit
Choose Option: 5
Number of Process: 5
Enter the priority of P1:2
Enter the arrival time of P1:0
Enter the burst time of P1:11
Enter the priority of P2:0
Enter the arrival time of P2:5
Enter the burst time of P2:28
Enter the priority of P3:3
Enter the arrival time of P3:12
Enter the burst time of P3:2
Enter the priority of P4:1
Enter the arrival time of P4:2
Enter the burst time of P4:10
Enter the priority of P5:4
Enter the arrival time of P5:9
Enter the burst time of P5:16
1: Higher Number Higher Priority
2: Lower Number Higher Priority
Choose Option: 2

Gantt Chart:
0 P1 2 P4 5 P2 9 P2 12 P2 38 P4 47 P1 57 P3 59 P5

Process:P2  Finish Time: 38  Response Time: 0  Waiting Time: 5  Turnaround Time: 33
Process:P4  Finish Time: 47  Response Time: 0  Waiting Time: 35  Turnaround Time: 45
Process:P1  Finish Time: 57  Response Time: 0  Waiting Time: 46  Turnaround Time: 57
Process:P3  Finish Time: 59  Response Time: 45  Waiting Time: 45  Turnaround Time: 47
Process:P5  Finish Time: 75  Response Time: 50  Waiting Time: 50  Turnaround Time: 66

Average waiting time: 36.2
Average Turnaround time: 49.6
Idle Time: 8

Press Any Key for home page
```

Round Robin:

```
1: FCFS
2: Non-Preemptive-SJF
3: Preemptive-SJF
4: Non-Preemptive-Priority
5: Preemptive-Priority
6: Round-Robin
7: Our-Proposed-Algorithm
8: Compare-All
9: Exit
Choose Option: 6
Number of Process: 5
Enter the arrival time of P1:0
Enter the burst time of P1:11
Enter the arrival time of P2:5
Enter the burst time of P2:28
Enter the arrival time of P3:12
Enter the burst time of P3:2
Enter the arrival time of P4:2
Enter the burst time of P4:10
Enter the arrival time of P5:9
Enter the burst time of P5:16
Enter the time quantum TQ: 3

Gantt Chart
0 P1 3 P4 6 P1 9 P2 12 P4 15 P5 18 P1 21 P3 23 P2 26 P4 29 P5 32 P1 34 P2 37 P4 38 P5 41 P2
 44 P5 47 P2 50 P5 53 P2 56 P5 57 P2 60 P2 63 P2 66 P2

Process:P3  Finish Time: 23  Response Time: 9  Waiting Time: 9  Turnaround Time: 11
Process:P1  Finish Time: 34  Response Time: 0  Waiting Time: 23  Turnaround Time: 34
Process:P4  Finish Time: 38  Response Time: 1  Waiting Time: 26  Turnaround Time: 36
Process:P5  Finish Time: 57  Response Time: 6  Waiting Time: 32  Turnaround Time: 48
Process:P2  Finish Time: 67  Response Time: 4  Waiting Time: 34  Turnaround Time: 62

Average waiting time: 24.8
Average Turnaround time: 38.2
Idle Time: 0

Press Any Key for home page
```

Comparative Analysis:

For comparative analysis among the seven scheduling algorithms, average turnaround time, average waiting time and average response time is considered. The comparative analysis of the algorithms along with our new proposed algorithm for the following test case is given below.

Process	Arrival Time	Burst time	Priority
p1	0	11	2
p2	5	28	0
p3	12	2	3
p4	2	10	1
p5	9	16	4

Table: Input Test case

Comparison between FCFS, non-preemptive SJF, preemptive SJF, non-preemptive priority scheduling, preemptive priority scheduling, round robin (RR) and our proposed algorithm.

Algorithm Name	Average Turnaround Time	Average Waiting Time
FCFS	23.6	37
Non-Preemptive SJF	13.2	26.6
Preemptive SJF	18.6	32
Non-Preemptive Priority Scheduling	24.4	37.8
Preemptive Priority Scheduling	36.2	49.6
Round Robin (RR)	24.8	38.2
Proposed Algorithm	13.2	26.6

Table: Comparison between the Scheduling Algorithms

Comparison Result in the Console:

```
1: FCFS
2: Non-Preemptive-SJF
3: Preemptive-SJF
4: Non-Preemptive-Priority
5: Preemptive-Priority
6: Round-Robin
7: Our-Proposed-Algorithm
8: Compare-All
9: Exit
Choose Option: 8
Number of Process: 5
Enter the priority of P1:2
Enter the arrival time of P1:0
Enter the burst time of P1:11
Enter the priority of P2:0
Enter the arrival time of P2:5
Enter the burst time of P2:28
Enter the priority of P3:3
Enter the arrival time of P3:12
Enter the burst time of P3:2
Enter the priority of P4:1
Enter the arrival time of P4:2
Enter the burst time of P4:10
Enter the priority of P5:4
Enter the arrival time of P5:9
Enter the burst time of P5:16
Enter the time quantum TQ: 3
Algorithm: 1    Average waiting time: 23.6      Average Turnaround time: 37
Algorithm: 2    Average waiting time: 13.2      Average Turnaround time: 26.6
Algorithm: 3    Average waiting time: 18.6      Average Turnaround time: 32
Algorithm: 4    Average waiting time: 24.4      Average Turnaround time: 37.8
Algorithm: 5    Average waiting time: 36.2      Average Turnaround time: 49.6
Algorithm: 6    Average waiting time: 24.8      Average Turnaround time: 38.2
Algorithm: 7    Average waiting time: 13.2      Average Turnaround time: 26.6

Press Any Key for home page
```

Complexity Analysis:

We calculate the time quantum before the process gets into the ready queue. We therefore implemented the ready queue using a C++ set, and when a specific process is pushed into the ready queue, it automatically sorts by a property of the binary search tree of the C++ set. Therefore, based on their burst time, all of the processes in the ready queue are sorted. A set insertion or deletion takes logN time, so for N processes it will take NlogN time in total. Besides, we used an arrival queue to know which process came before or within the current time. Everytime we pop processes from the arrival queue and push them into the ready queue, for N processes it will take only N iterations overall. So, our proposed algorithm runs in $O(NlogN + N)$ or asymptotically bounded to $O(NlogN)$ time complexity overall.

Conclusion:

In our project work, we have implemented a new scheduling algorithm based on the Round Robin (RR).Our suggested technique essentially yields the best outcome for several processes entering the ready queue at the same time. In the majority of test scenarios,our algorithm produces an effective outcome. In comparison to other algorithms, the average turnaround time, average waiting time, and average response time are all lower. Despite this limitation, it typically produces more accurate results and shortens average turnaround and waiting times. It can be enhanced with additional

work. In conclusion, compared to some of the other techniques for many processes arriving at the same time, our implemented algorithm is more CPU efficient.

References:

[1] A. Upadhyay and H. Hasija, "Optimization in Round Robin Process Scheduling Algorithm," in *Information Systems Design and Intelligent Applications*, 2016, pp. 457–467.

[2] M. S. Iraji, "Time Sharing Algorithm with Dynamic Weighted Harmonic Round Robin," *J. Asian Sci. Res.*, vol. 5, no. 3, pp. 131–142, 2015, doi: 10.18488/journal.2/2015.5.3/2.3.131.142.

[3] "ICCDE 2020: Proceedings of 2020 the 6th International Conference on Computing and Data Engineering," 2020.