

OPERATING SYSTEMS

TEAM MEMBERS:

DIKSHA SINGHANIA

KARTHIK CHALLA

PRIYANSHU MITTAL

DIVYA GAHLOT

SHUBH AGARWAL

17BCE0032

17BCE2255

17BCE0027

17BCE0509

17BCE2249

**PROJECT REVIEW 3**

PROJECT TITLE

SCHEDULING ALGORITHMS

An improved Round Robin CPU Scheduling Algorithm-Analysis and Implementation Using the IRRSJF Technique proposed.

**Contents covered in the Review:**

1. Title.
2. Slot.
3. Member names and Registration Numbers.
4. Abstract.
5. Keywords.
6. Introduction.
7. Literature Survey
8. Gaps identified and observations made from the literature survey.
9. Proposed Model-Explanation of the model with Architecture.
10. Parameters identified for evaluation of proposed model.
11. References (in APA format)
12. **PROJECT TITLE**

**SCHEDULING ALGORITHMS:**

**An improved Round Robin CPU Scheduling Algorithm**

**Ananlysis and Implementation in different fields.**

1. **SLOT**

**F2+TF2**

**Lab 11+12**

1. **MEMBER NAMES AND REGISTRATION NUMBERS**

TEAM MEMBERS

Diksha Singhania 17BCE0032

Karthik Challa 17BCE2255

Priyanshu Mittal 17BCE0027

Divya Gahlot 17BCE0509

Shubh Agarwal 17BCE2249

1. **ABSTRACT**

The Central Processing Unit (CPU) scheduling in today’s generation plays a deep-seated role during designing of latest processors and units. Not only it is the premium desire of today’s processing systems to efficiently switch between processes to maximize throughput, but also to do it in a way so that the idle time is reduced to a bare minimum. By switching the CPU among various processes, these targets are attempted to be achieved.

The performance of any CPU majorly depends on the scheduling algorithm used by the Processor. Primarily, the CPU is one of the most essential computer resources we have today. Not much can be done with the efficiency in executing the processes, since they take almost similar times, but a lot can be done with the scheduling. And, since the round robin scheduling algorithm is considered to be one of the most widely used algorithms, a new proposed variant of this algorithm is attempted to be established. We shall attempt improving upon the Round Robin CPU Scheduling System incorporating Shortest Job First Scheduling, achievement of stability in the Round Robin Scheduling, using the concept of dynamic Time Quantum.

1. **KEYWORDS**

* Operating System,
* Scheduling,
* Round Robin algorithms,
* Shortest Job First,
* Waiting time,
* Turnaround time,
* Context switch.

1. **INTRODUCTION**

As processor is the most important resource that essentially determines the working capacity of a machine, CPU scheduling becomes very important in accomplishing the operating system design goals.

The target of any Operating System should be **“To allow maximum processes running at all the times in order to make best use of the CPU.”** The efficiency of any CPU scheduler primarily depends on the design of the well-structured scheduling algorithms which goes well with the scheduling goals.

In this paper, we propose an algorithm which can handle all types of process with optimum scheduling criteria.

The efficiency of scheduling algorithm is measured by the following performance factors:

1. **Throughput:**

Defined as the number of processes executed in one unit time.

1. **Waiting- time:**

It is the time a process has to wait for CPU in ready-queue to come in main memory.

1. **Turn-around time:**

Defined as the amount of time required by any process to complete its execution.

1. **Response time:**

The time between generation of a request and the first response.

1. **Context Switching:**

The process of switching CPU between processes, also known as pre-emption.

The **CPU Utilization** is a measure of maximum usage of CPU, or how effectively CPU is made busy. The performance and effectiveness of the Round Robin algorithm is largely dependent on the value of **time quantum** selected.

If the value of time quantum is too small then the number of context switches will be more and algorithm will not be effective. If the value of the time quantum is too large, then the algorithm will work more or less like FCFS algorithm.

Improved Round Robin with dynamic Choosing an optimum time quantum can significantly decrease the number of context switches, maintaining the RR nature and also can improve performance. The number of context switches can further be reduced if there is a strategy to execute processes with smallest remaining burst- times.

CPU scheduling is the basis of all operating systems. **The technique used for controlling the order of job which is to be performed by a CPU of a computer is called Scheduling.**

**Most CPU scheduling algorithms concentrate on:**

* MAXIMIZING:

1. CPU utilization
2. Throughput

And,

* MINIMIZING:

1. Turnaround time
2. Response time
3. Waiting time
4. Number of context switching for a set of requests.

But we are concentrating on a **new Round Robin Scheduling which is designed with the Shortest job first along with using the concept of Dynamic Quantum Time.**

**This Scheduling gives better results compared to:**

1. First Come First Serve (FCFS)
2. Round Robin (RR)
3. Improved Round Robin (IRR)
4. Improved Round Robin with Mean Time Quantum (IRR-Mean-TQ)
5. Improved Round Robin with Min Time Quantum (IRR-Min-TQ)
6. Improved Round Robin

The Project is about the existing algorithms which are variants of RR algorithm. We attempt at briefing the proposed methodology, algorithmic procedure, pseudo code and flow chart, and illustrating it with an example, the working of the proposed algorithm. We further analyse the comparison of the proposed algorithm with other existing RR variants. Finally, we conclude and look into the future enhancements respectively.

1. **LITERATURE SURVEY**

The References have been listed in the References Section. An in-depth study of the papers and journals has been done, and the following survey has been established. We looked into the reviews, papers and journals presented by the earlier authors, and gathered following points of our own regarding the reviews.

**[1]**

This was one of the First greatest advancements made in the Round Robin Scheduling Algorithm. Proposed by Sir Manish Kumar Mishra in 2012, the paper describes a new improved version of RR. Improved Round Robin picks the first process from the ready queue and allocate the CPU to it for a time interval of up to 1 time quantum. Once a process’s time quantum is elapsed, it checks the remaining CPU burst times of the processes currently in execution. If the CPU burst time remaining of the current process is less than 1 TQ, the CPU shall again be allocated to the process currently in execution for the remaining burst time.

**[2]**

In year 2013, Aashna Bisht [2] performed an analysis, and proposed a work Enhanced Round Robin(ERR) model, in which, the time quantum of only those processes which require a slightly greater time than the allotted time quantum cycle were modified. The remaining process will be executed in the already proposed Round robin manner.

**[3]**

The Next advancement came in the year 2011, by Saroj Hiranwal. In this, first of all we arrange the processes according to the execution time/burst time in increasing order that is smallest the burst time higher the priority of the running process. The smart time slice, then calculated, is the median process burst time of all CPU burst times in the ready queue.

**[4]**

Many other advancements came along the years between 2011 and 2014. In year 2011, H.S. Behera proposed an improved process scheduling algorithm by using dynamic quantum time along with its weighted mean.

In year 2011, Rakash Mohanty & Manas Das performed a work in which a new variant of Round Robin scheduling algorithms was created by executing the processes according to the calculated Fit factor and they also used the concept of dynamic quantum time.

**[5]**

In year 2012,Ishwari Singh Rajput, Deepa Gupta [5] proposed priority based Round-robin CPU scheduling algorithms is based on the integration of round robin and priority scheduling. It still holds the advantage of round robin in reducing starvation and also integrates the advantage of sjf scheduling.

**[6]**

In year 2012, P.Surendra Varma[6] performed a work. In this paper the quantum time is computed with the help of median and highest burst time.

**[7]**

In year 2012, H.S. Behera & Brajendra Kumar Swain[7] performed a work it gives precedence to all processes according to their priority and burst time, then applies the Round Robin algorithm on it. This Proposed algorithm is developed by taking mean of dynamic quantum time in account.

1. **GAPS IDENTIFIED AND OBSERVATIONS MADE FROM THE LITERATURE SURVEY**

From the Analysis done above of the Research papers and journals by people, a lot of Observations have been made regarding the Implementation Challenges and Future possibilities of the combinative Approach of the Round Robin Algorithm with other Scheduling Algorithms.

1. Attempts at improving the Round Robin Algorithm using the Improved versions of other scheduling Algorithms like SRBRR.
2. Other improvement techniques like better Selection of time slice, priority scheduling has been done.
3. However, Researches oriented towards attempting to collaborate techniques like Integration of Improved Round Robin and SJF hasn’t seen much Advancement.
4. It is either RR attempted at union with the SJF or RR with SRTF. But, Uniting IMPROVED ROUND ROBIN TECHNIQUE hasn’t been implemented yet.
5. The Proposed techniques attempted at Either reducing the Average Waiting Time or the Turnaround time, which is exactly what should be the aim.
6. **PROPOSED MODEL AND ARCHITECTURE**

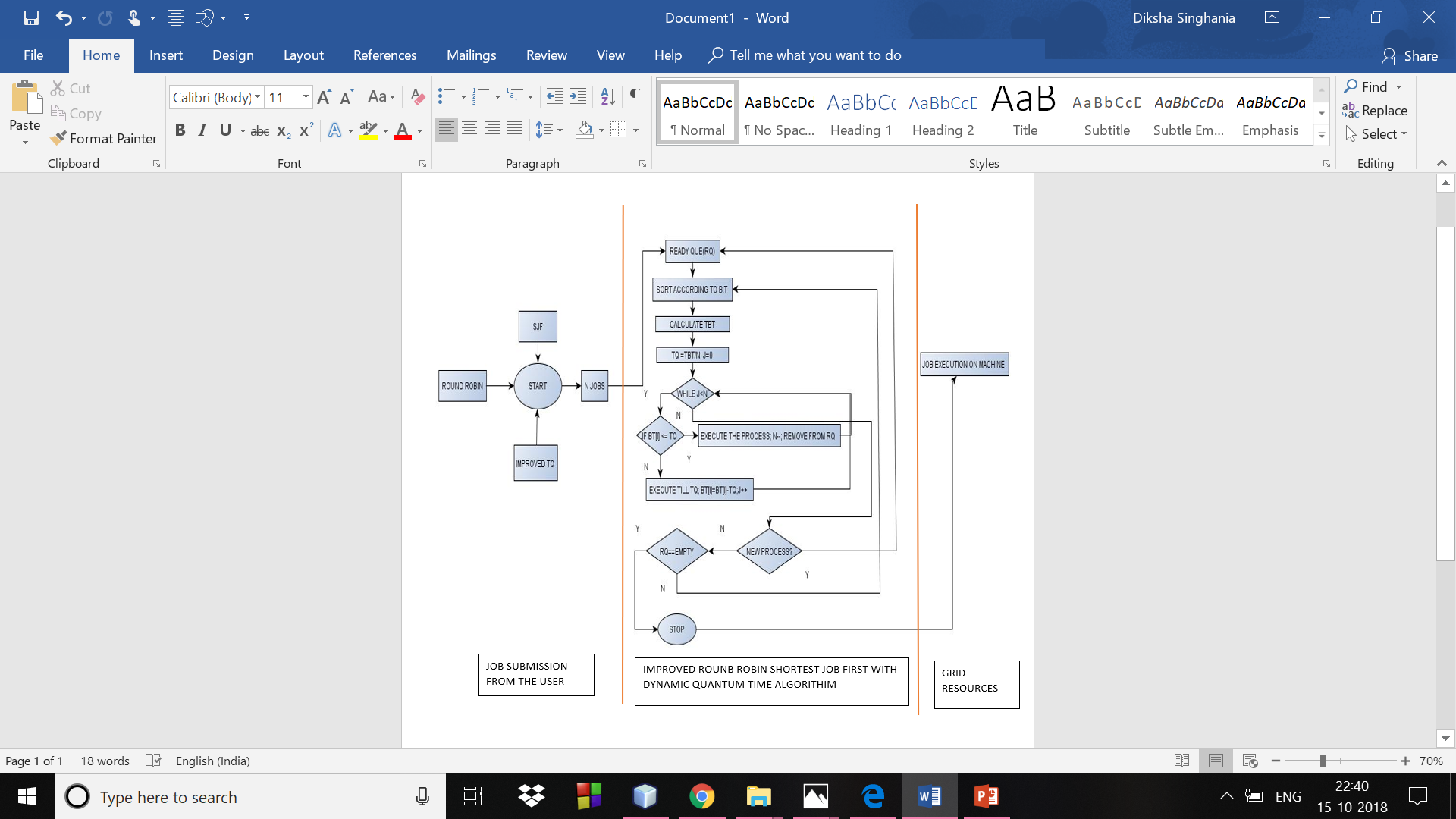
The project presents a vast scope for future advancements and improvements. Each and every day, the world is growing better with the advancements in Technology. To follow up with the ever-increasing efficiencies, new scheduling algorithms are need to be presented and developed. Better the CPU scheduling algorithms in their efficiency, faster would be the jobs taken up by the processor, and hence faster the execution.

**9.1 Architecture**

* Start
* Here we in our proposed new architecture we are going to take round robin algorithm and our improved round robin combined with sjf from the given user for the given amount of the jobs
* This part is the job submission from user
* From this jobs are been scheduled in the ready queue
* This assigning of the given jobs are to sorted in the order to the burst time in the ready queue.
* After sorting the given processes we are going to calculating the total burst time for the scheduled processes in the ready queue.
* Now we calculate the time quantum using total burst time of the scheduled processes / the no of the processes scheduled, where J=0 J considering it as a variable
* While J<N if (NO ) it will put fourth to new process
* If (yes) it checks whether burst time is greater than or equal to quantum time
* If the above condition is true then it will execute the process for N— times and it will remove the process from the ready queue
* If it fails then it will execute till quantum time where the BT[i]=BT[i]-TQ; J++
* If while J<N is false then it will assign to a new process
* If the given J<N is false then RQ==EMPTY
* If the above condition is true the stop
* If it is false then go to the process where the job is assigned and where all the processes are sorted in the order to the burst time which applies to the starting of the point of the ready queue
* If it satisfies the above condition then the job execution is complete for the given processes
* End

**9.2 Block Diagram of Architecture**

(Explained through the Block Diagram on the next page)



**9.3 PROPOSED MODEL**

* When operating system installed for the first time, it begins with a **default time quantum value**, which is subject to change after a period of time through which the operating system can identify the burst time for a predefined set of the programs that is used by the user.
* So, we assume that the system will not immediately take advantage of this method because it needs time to learn user behavior through the analysis of the burst time of the new processes. The determined time quantum represents real and optimal value because it based on real burst time unlike the other methods, which depend on fixed or possible time quantum value, determined by a variety methodologies such as guessing, fuzzy logic etc.
* Repeatedly, when a new process loaded to be executed the operating system tests the status of the specified program which can be either **1 or 0.**
* When the **STATUS** equals to:
  + **0:**

This means that the process is either being executed for the first time or it has been modified since the last analysis. In this case the operating system assigns a counter to find the burst time of the process and continues to executes the processes in the ready queue on the current round including the new arrival process using the current quantum time, otherwise.

and when status is equal to

* + **1:**

The operating system recalculates the time quantum Q depending on the remaining burst time of all ready processes including the new arrival process.

* **Important Point:**

In the previous versions of the round robin scheduling algorithm the context switch occurs even if there is only a single process in the ready queue, while the operating system assigns the process a specific time quantum Q, when time quantum expires the process interrupted and again assigned the same time quantum Q, regardless, whether the process alone in the ready queue or not, which means that, there will be additional unnecessary context switches, while this problem does not occur at all in this algorithm; because in this case the time quantum will equal to the remaining burst time of the process.

**9.4PSEUDOCODE**

1. **START**
2. Check the ready queue for null or not.

While (ready queue==null)

1. If new process Pi arrived then check its status.
2. If status == 0 then assign a new counter Ci  for this process

End if (and find the new quantum).

1. New\_Quantum (NQ) = median (for the remaining burst time of all processes in ready queue with status==1)
2. Make a REQUEUE for the processes which are ready to be executed
3. Input the number of processes, n and respective data
4. DO steps 4 to 12 WHILE REQUEUE is empty and count = n
5. Sort the REQUEUE according to Burst times
6. SET the time quantum as the rms value of burst times
7. Remove the first process from REQUEUE and execute it
8. if remaining burst time<=quantum time goto next step
9. Re execute the same process
10. If remaining burst time=0, count=count + 1
11. Add the current process at tail of REQUEUE
12. If new process has arrived goto step 12
13. Sort the REQUEUE according to Burst times
14. Continue executing the process using the new Quantum NQ.
15. Initialize the loop (for i=1 to n)

(here: n=the no. of all processes in the ready queue)

Now assign CPU to process Pi and give it slice of time=NQ

1. If Pi terminated normally and Pi.  status==0 the save the state Ci(Pi)

Now let Pi status =1

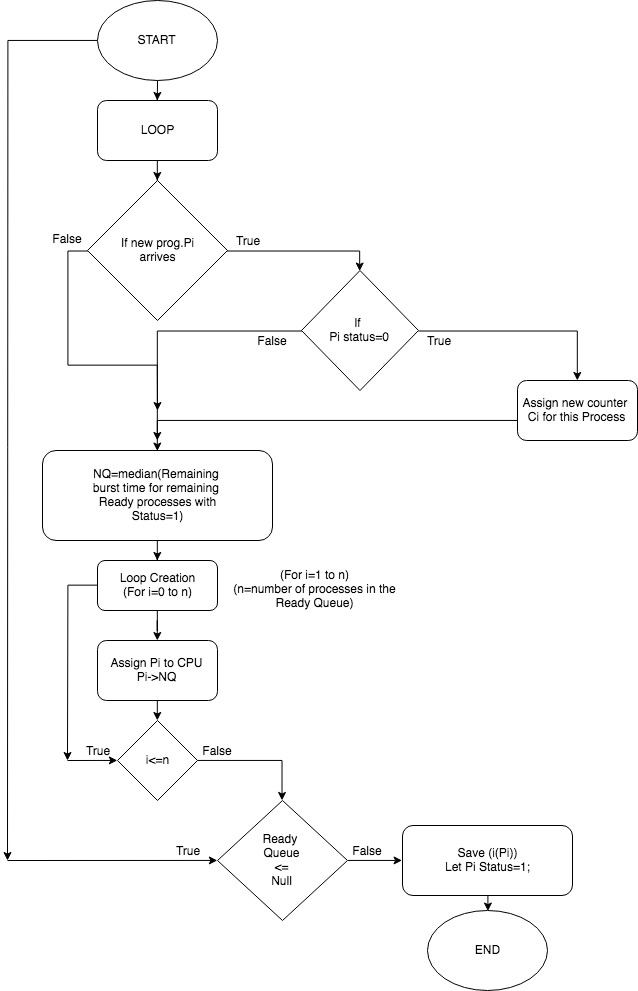
End of if

End of for

End while

**END**

* 1. **FLOWCHART**

****

**9.6 CASE STUDY**

To test how effective is the newly proposed model, we shall take up a scenario with 4 Processes to be executed by the CPU, with different burst times, and we shall study the respective:

1. **Average Waiting Time.**
2. **Average Turnaround Time.**

For the same.

Lesser the Average Waiting Time and turnaround time, better is the Scheduling Algorithm.

Let the jobs assigned to the CPU have the following **Arrival times and burst times**:

**CASE 1**

Let the Burst Time and the Arrival Times of the Processes be:

|  |  |  |
| --- | --- | --- |
| Process  Number | Arrival  Time | Burst  Time |
| P1 | **20** | **42** |
| P2 | **0** | **12** |
| P3 | **10** | **25** |
| P4 | **15** | **70** |

GANTT CHART FOR THE PROCESSES

1. FCFS

|  |  |  |  |
| --- | --- | --- | --- |
| P2 | P3 | P4 | P1 |

0 12 37 107 149

Average Waiting Time=27.75

Average Turnaround Time=65

1. RR

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P2 | P3 | P4 | P1 | P3 | P4 | P1 | P4 | P1 | P1 |

0 12 32 52 72 77 97 117 137 139 149

Average Waiting Time=39.5

Average Turnaround Time=76.75

1. IRR

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P2 | P3 | P3 | P4 | P1 | P4 | P1 | P1 | P4 |

0 12 32 37 57 77 97 107 109 149

Average Waiting Time=28.25

Average Turnaround Time=65.5

1. IRR-SJF

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| P2 | P3 | P3 | P1 | P4 | P1 | P1 | P4 |

0 12 32 37 57 77 97 99 149

Average Waiting Time=24.5

Average Turnaround Time=61.75

1. IRR-SJF with Dynamic TQ

|  |  |  |  |
| --- | --- | --- | --- |
| P2 | P3 | P1 | P4 |

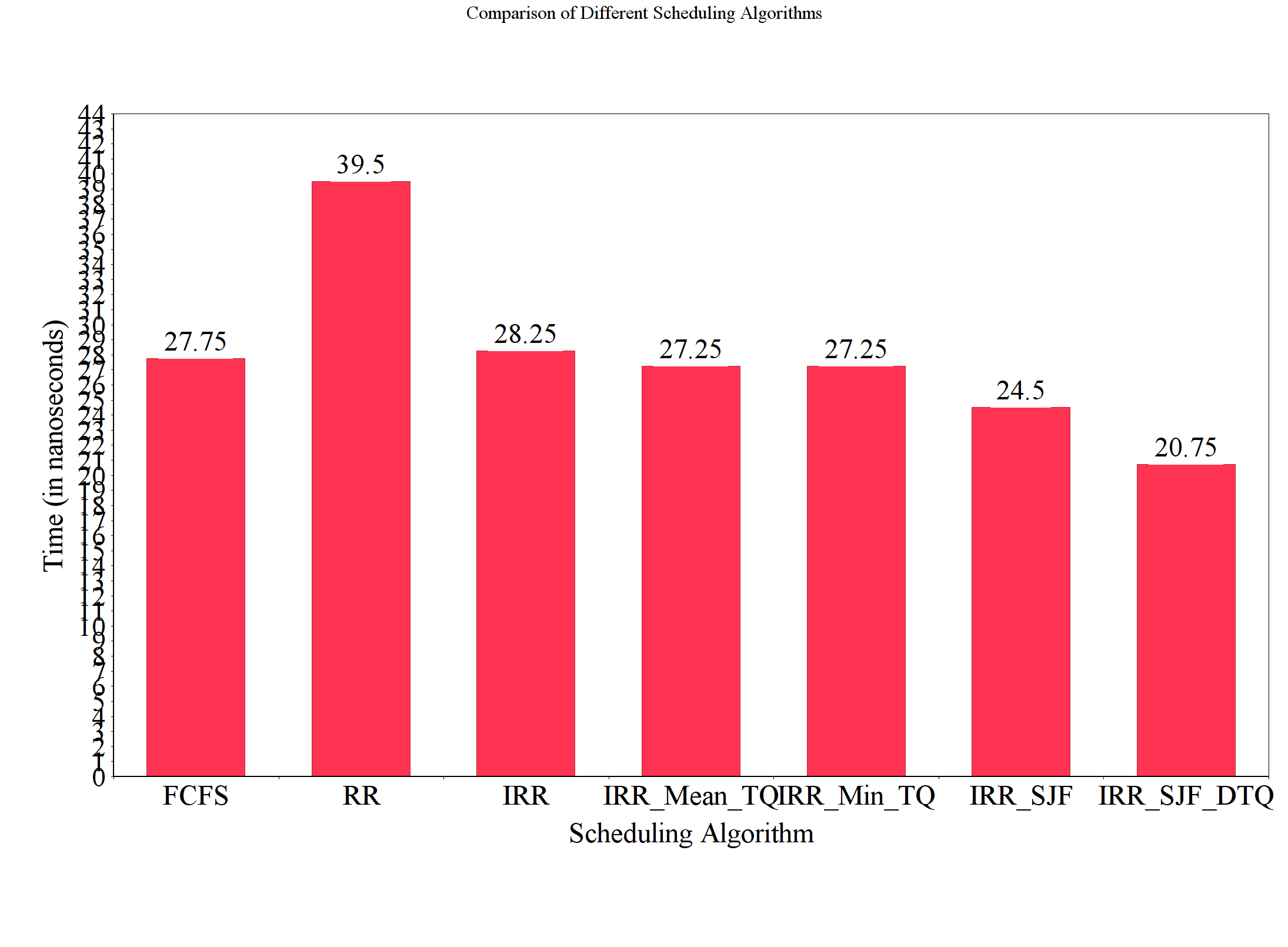
0 12 37 79 149

Average Waiting Time=20.75

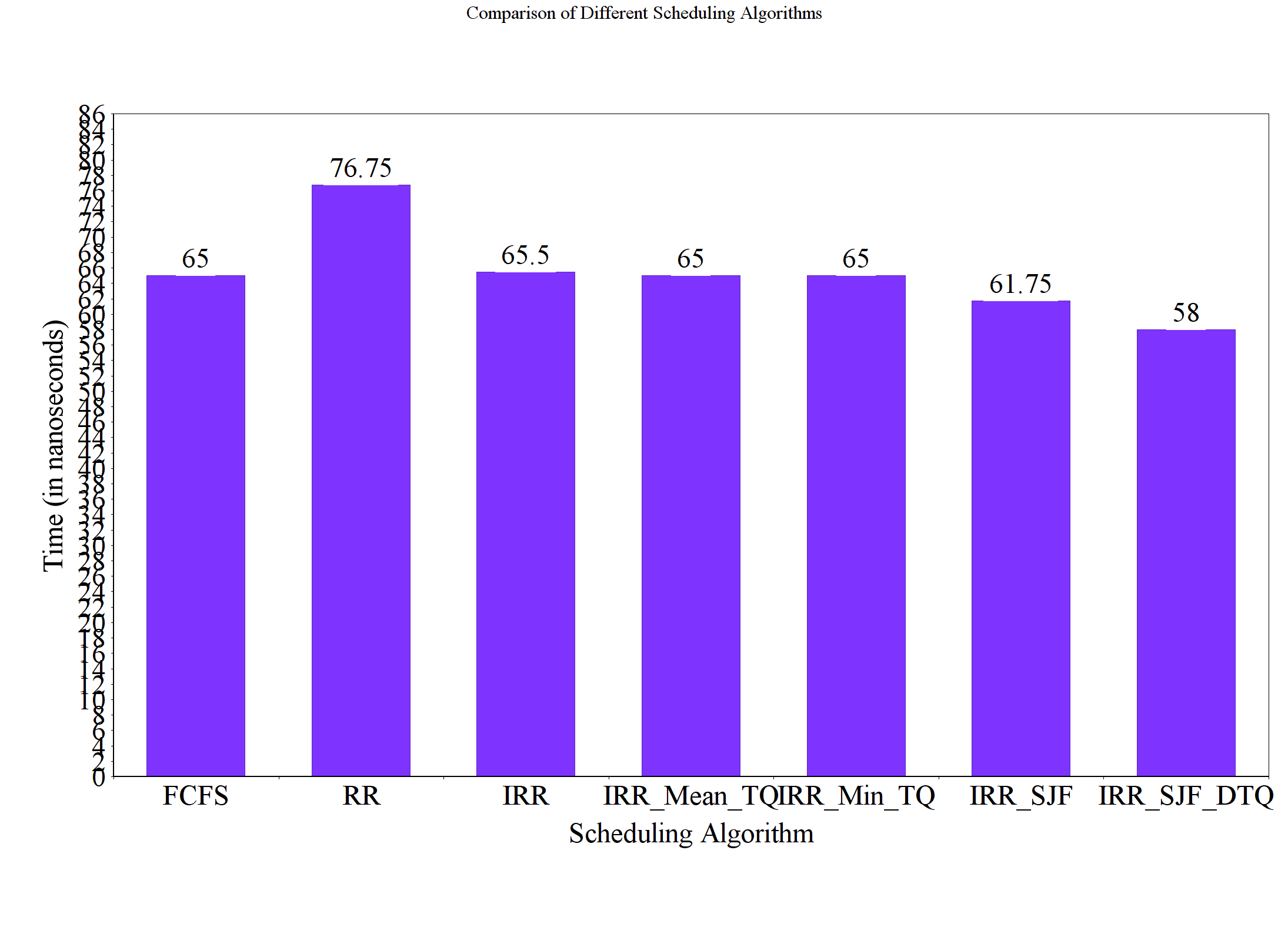
Average Turnaround Time=58

COMPARISON OF ALL THE SCHEDULING ALGORITHMS

|  |  |  |
| --- | --- | --- |
| ALGORITHM | AVG\_WT | AVG-TAT |
| FCFS | **27.75** | **65** |
| RR | **39.5** | **76.75** |
| IRR | **28.25** | **65.5** |
| IRR\_MEAN\_TQ | **27.25** | **65** |
| IRR\_MIN\_TQ | **27.25** | **65** |
| IRR\_SJF | **24.5** | **61.75** |
| IRR\_SJF\_DTQ | **20.75** | **58** |

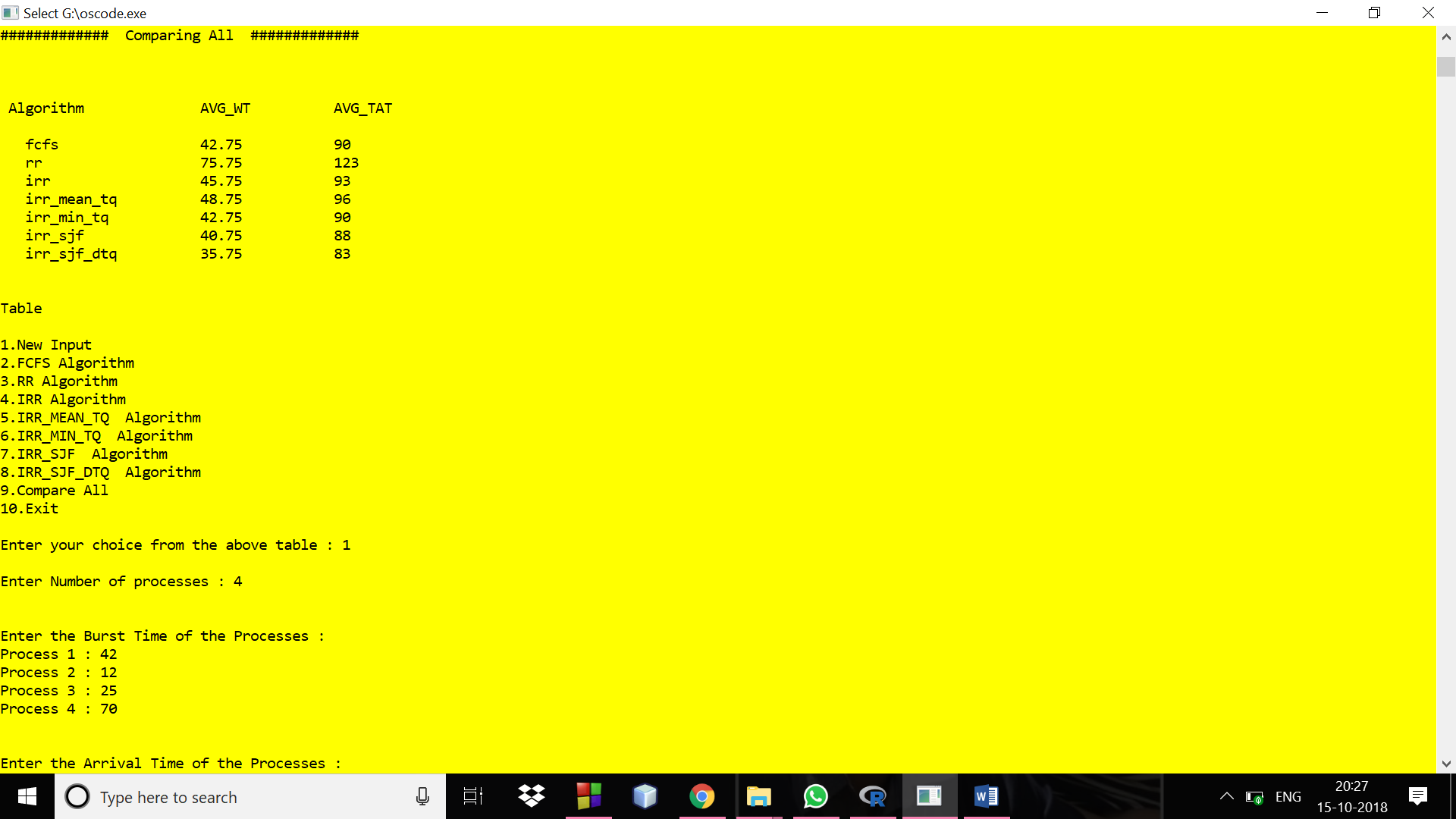
****

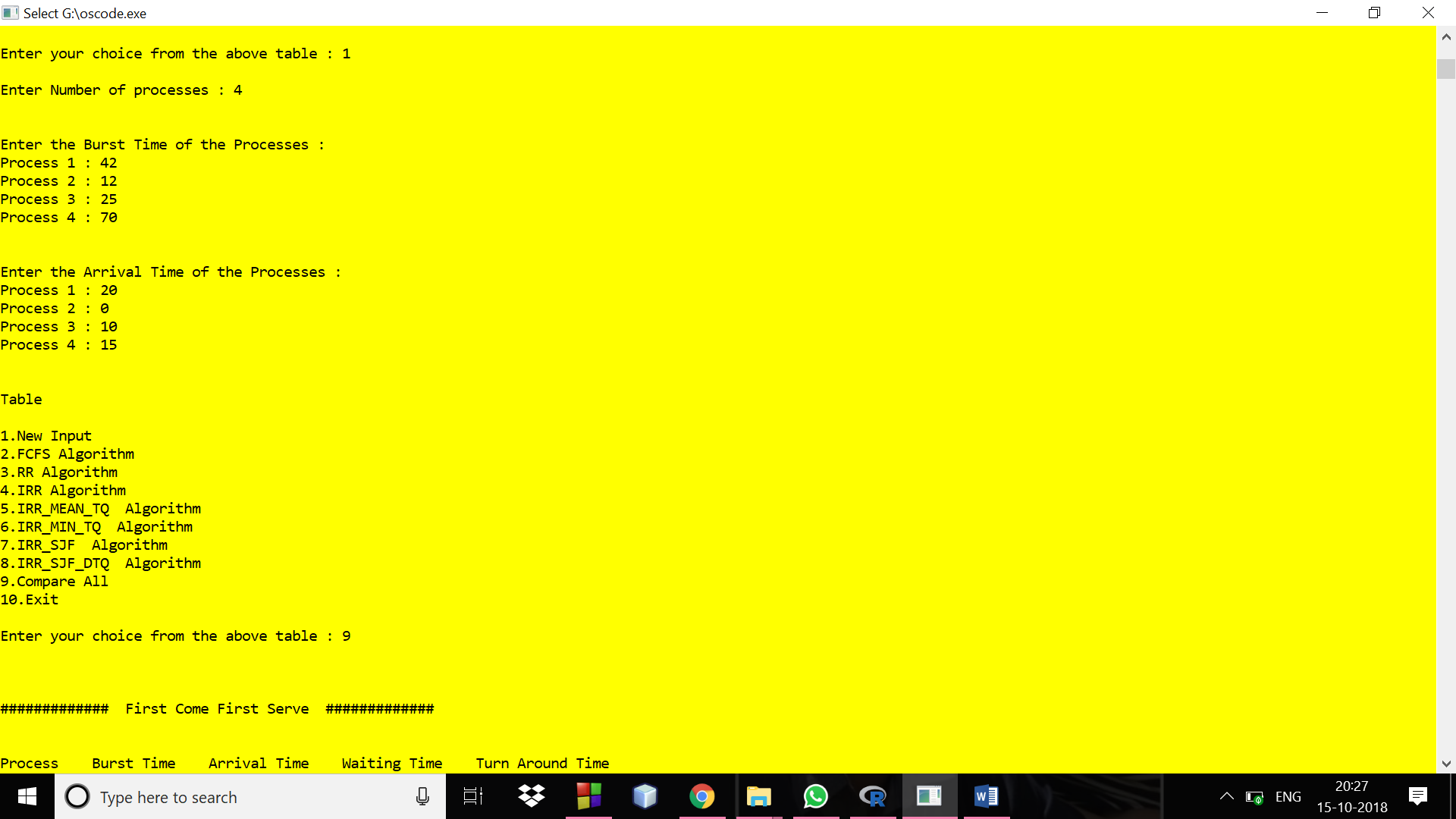
Comparison of the Average Waiting Time of different Scheduling Algorithms

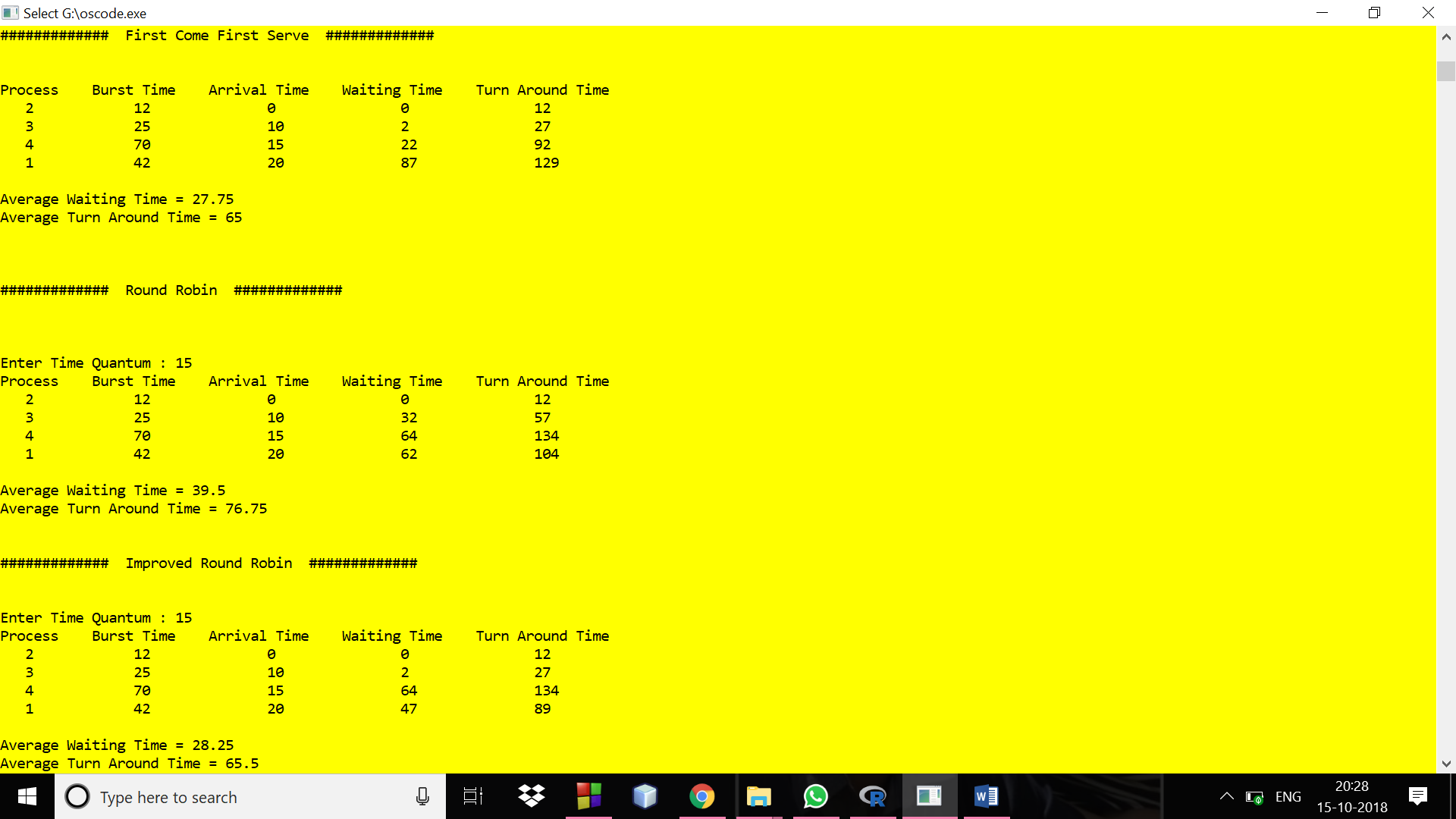
****

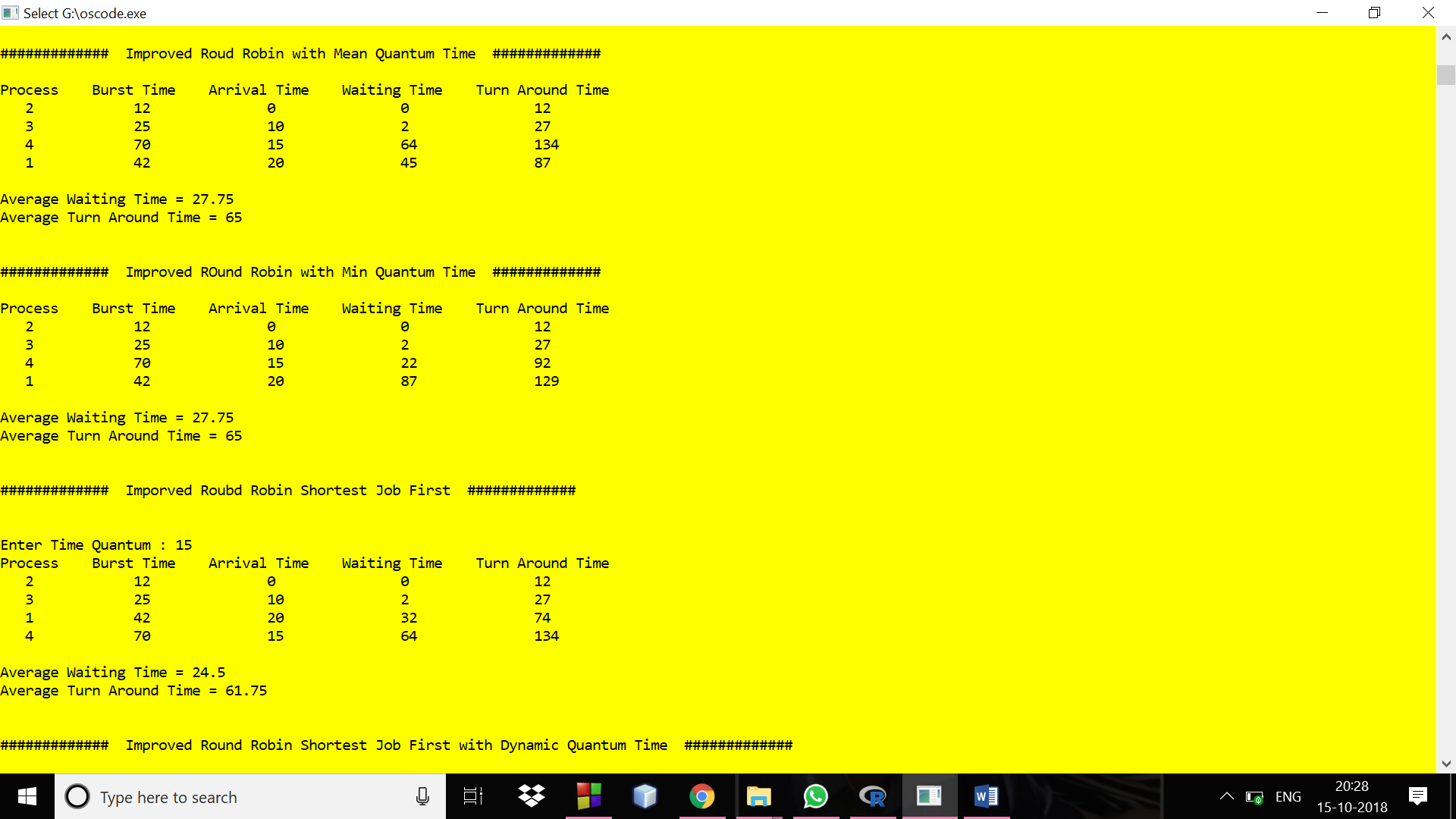
Comparison of the Average TurnAround Time of different Scheduling Algorithms

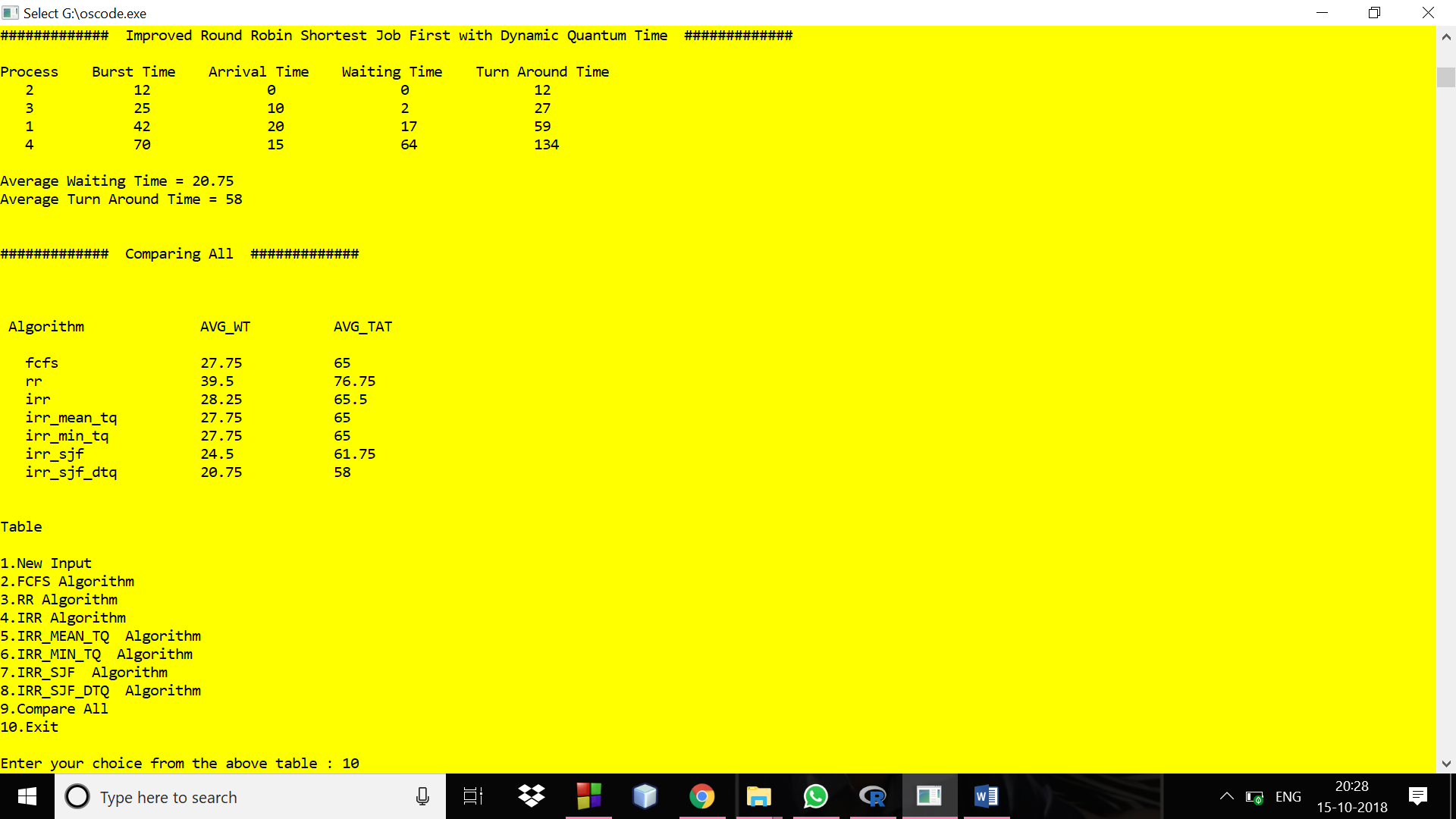
OUTPUT SCREENSHOTS OF THE IMPLEMENTATION

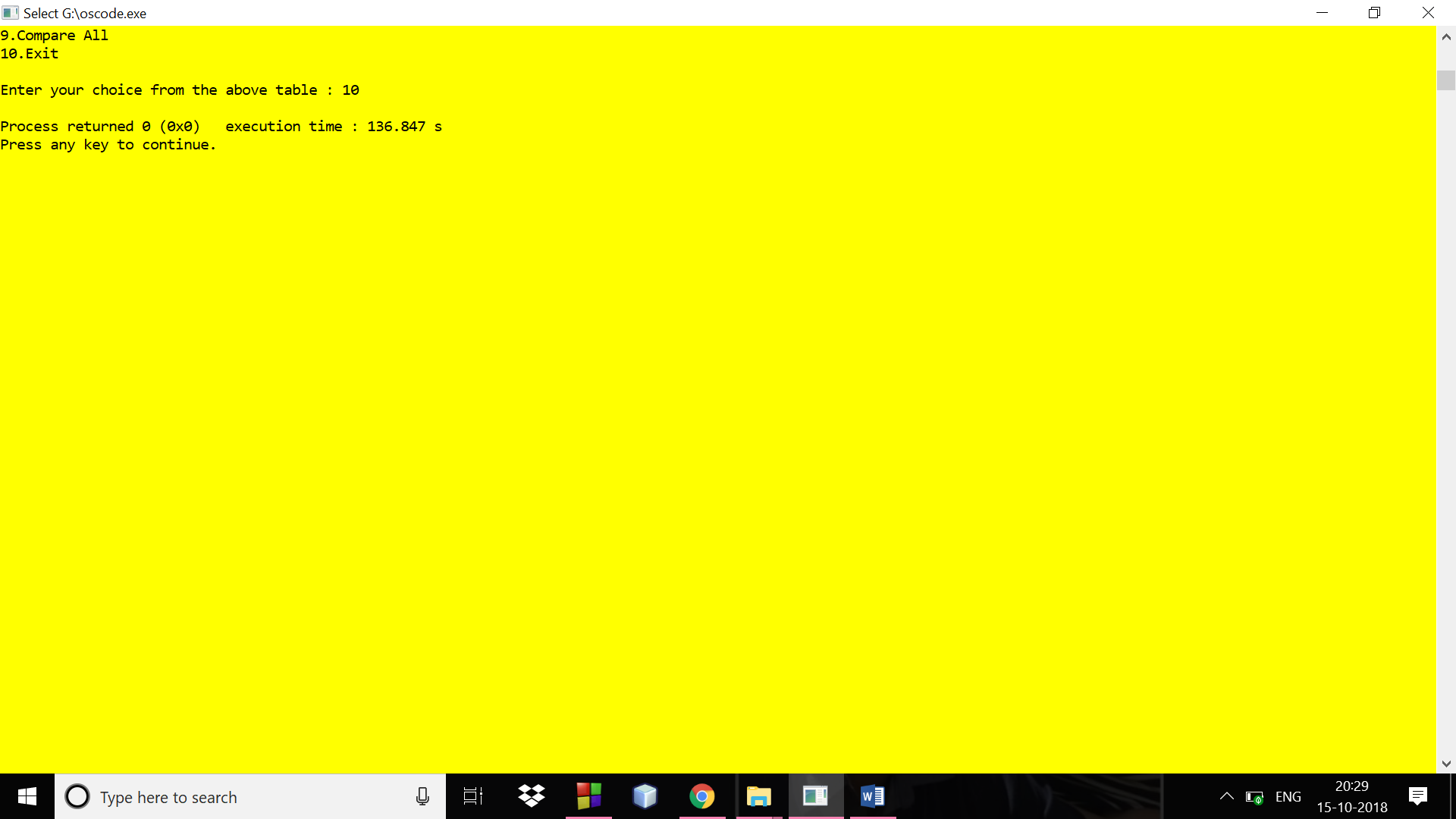












**CASE 2**

Let the Burst Time and the Arrival Times of the Processes be:

|  |  |  |
| --- | --- | --- |
| Process  Number | Arrival  Time | Burst  Time |
| P1 | 20 | 52 |
| P2 | 0 | 22 |
| P3 | 10 | 35 |
| P4 | 15 | 80 |

We shall compare the average Waiting and Turnaround time for the Newly

Proposed Algorithm and compare it with the previous existing Scheduling

Algorithms.

1. FCFS

|  |  |  |  |
| --- | --- | --- | --- |
| P2 | P3 | P4 | P1 |

0 22 57 137 189

Average Waiting Time=42.75

Average Turnaround Time=90

1. RR

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P2 | P3 | P4 | P1 | P2 | P3 | P4 | P1 | P4 | P4 |

0 20 40 60 80 82 97 117 132 159 189

Average Waiting Time=39.5

Average Turnaround Time=76.75

1. IRR

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P2 | P2 | P3 | P4 | P1 | P3 | P1 | P1 | P4 | P4 |

0 20 22 42 62 82 97 117 132 159 189

1. IRR-SJF

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P2 | P2 | P3 | P3 | P1 | P4 | P1 | P1 | P4 |

0 20 22 42 57 77 97 117 129 189

Average Waiting Time=40.75

Average Turnaround Time=88

1. IRR-SJF with dynamic TQ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P2 | P2 | P3 | P3 | P1 | P1 | P1 | P4 | P4 | P4 |

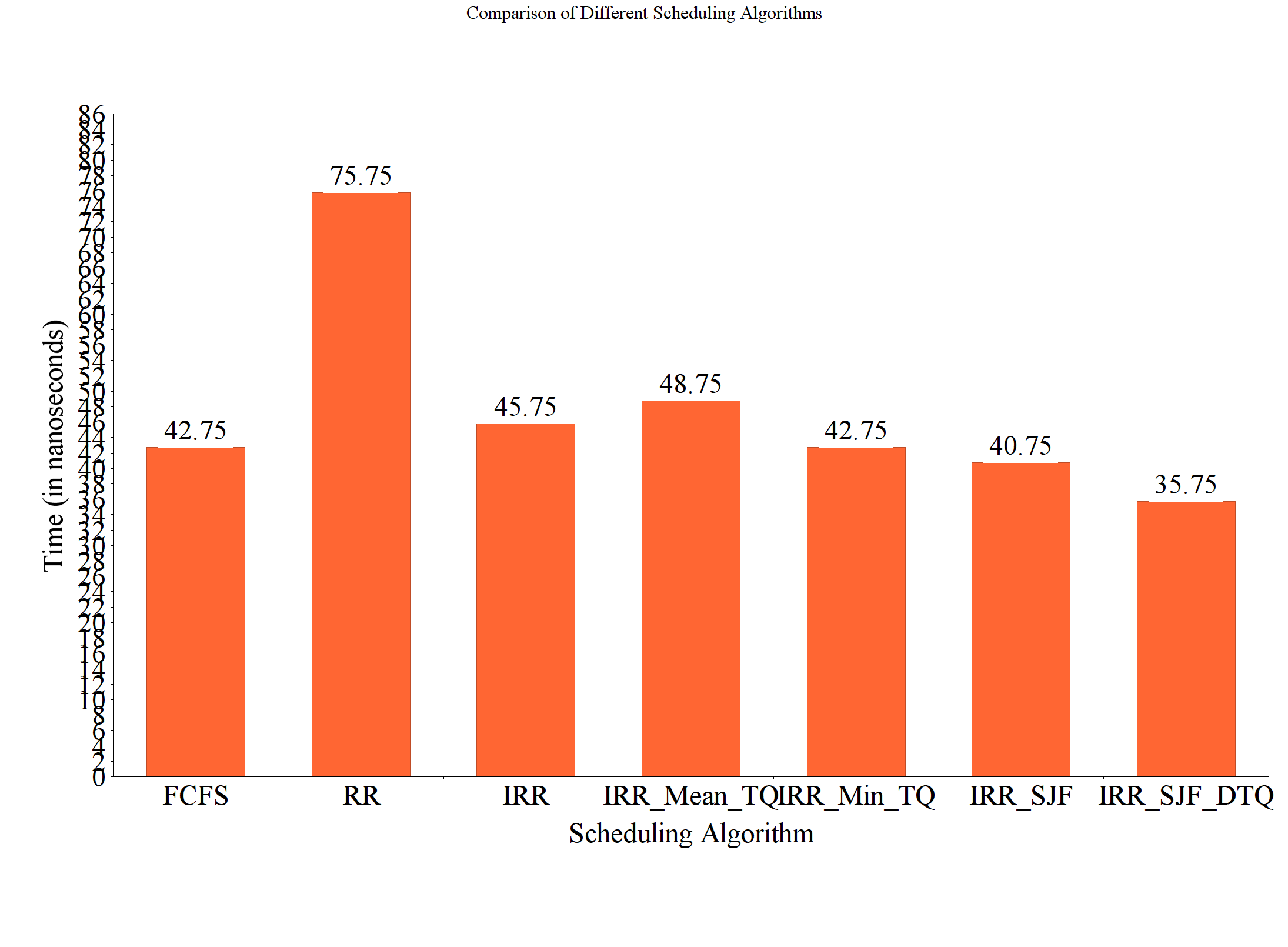
0 20 22 42 57 77 97 109 129 169 189

Average Waiting Time=35.75

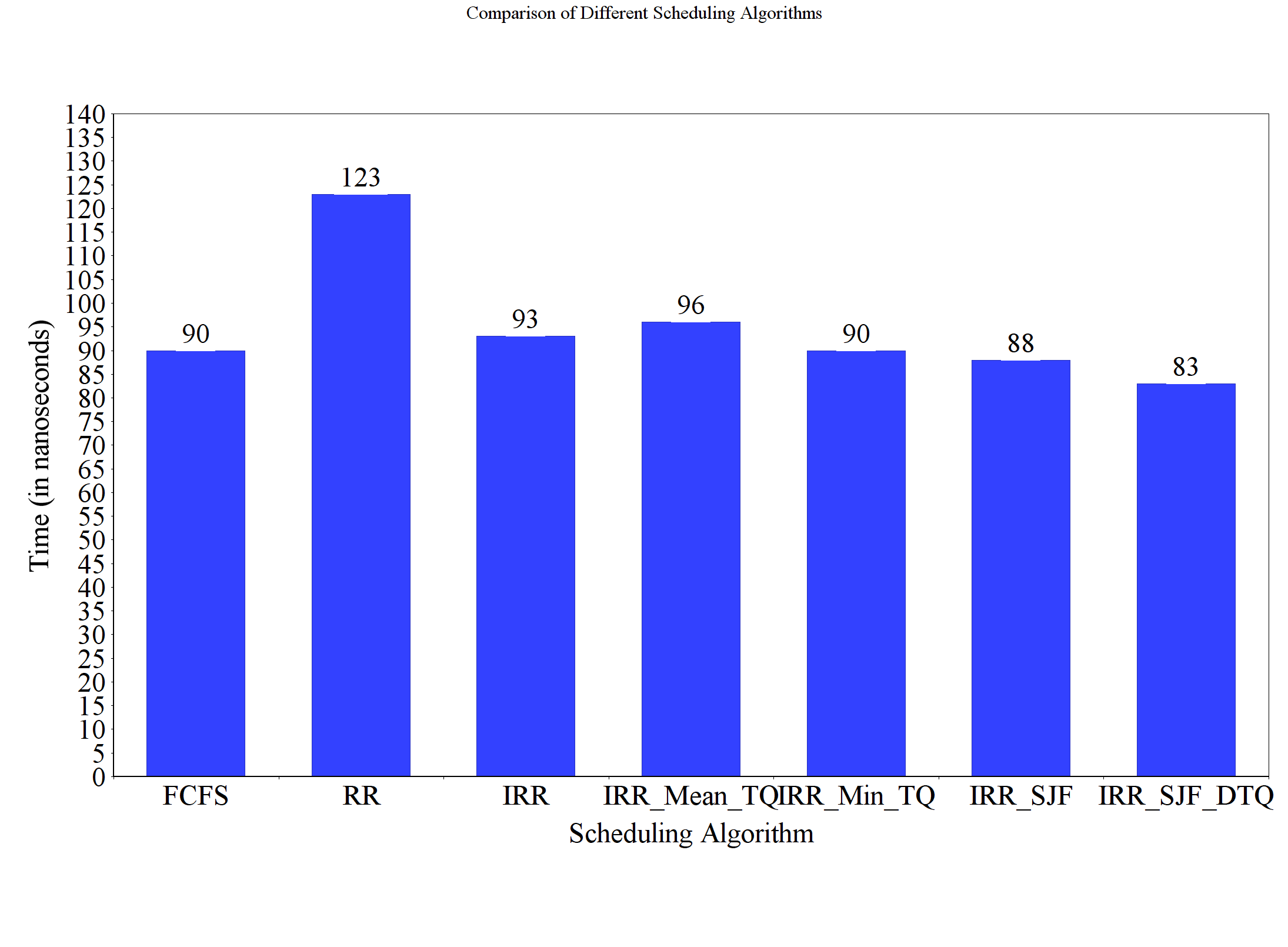
Average Turnaround Time=83

COMPARISON OF ALL THE SCHEDULING ALGORITHMS

|  |  |  |
| --- | --- | --- |
| ALGORITHM | AVG\_WT | AVG-TAT |
| FCFS | **42.75** | **90** |
| RR | **75.75** | **123** |
| IRR | **45.75** | **93** |
| IRR\_MEAN\_TQ | **48.75** | **96** |
| IRR\_MIN\_TQ | **42.75** | **90** |
| IRR\_SJF | **40.75** | **88** |
| IRR\_SJF\_DTQ | **35.75** | **83** |

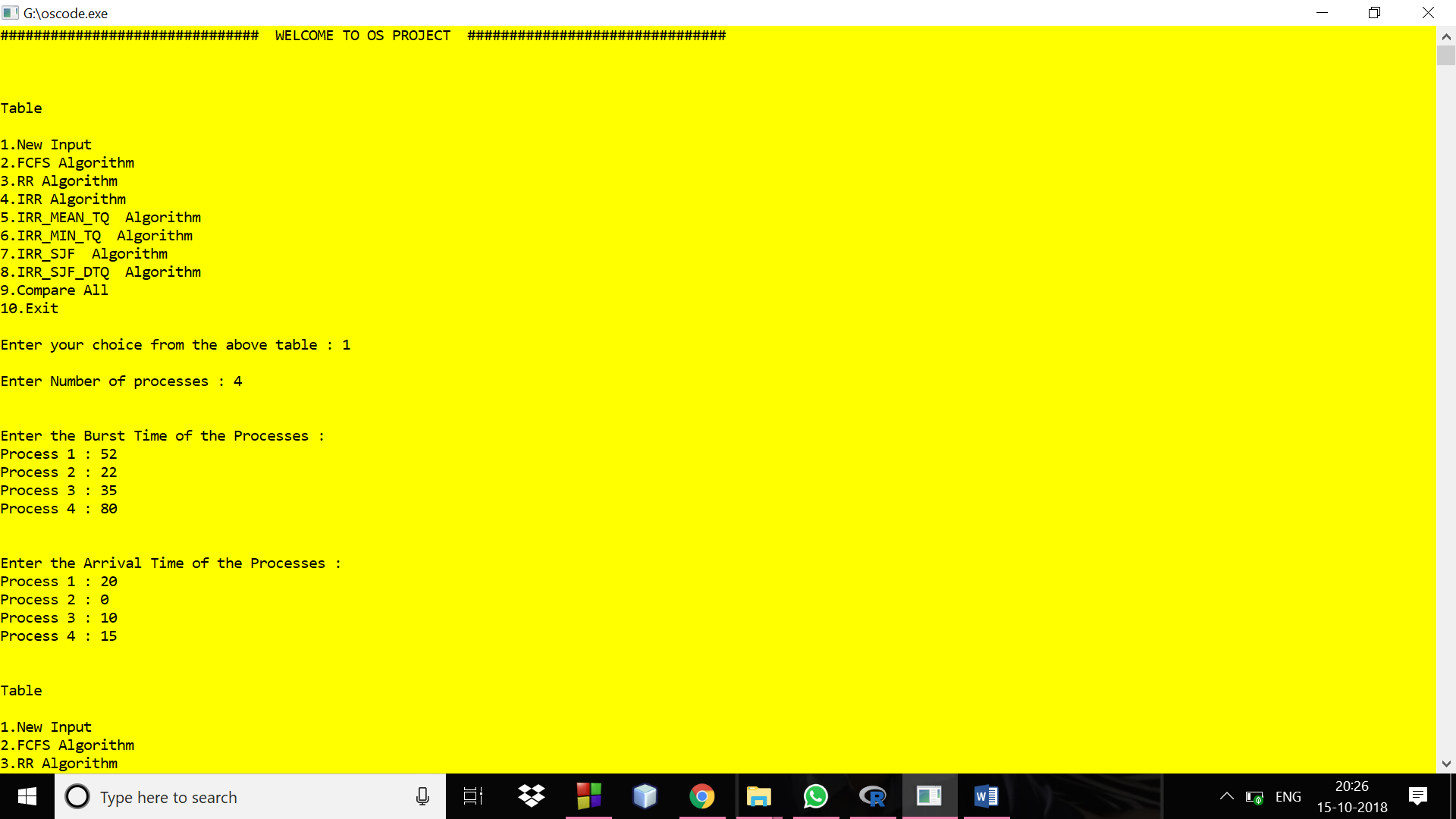


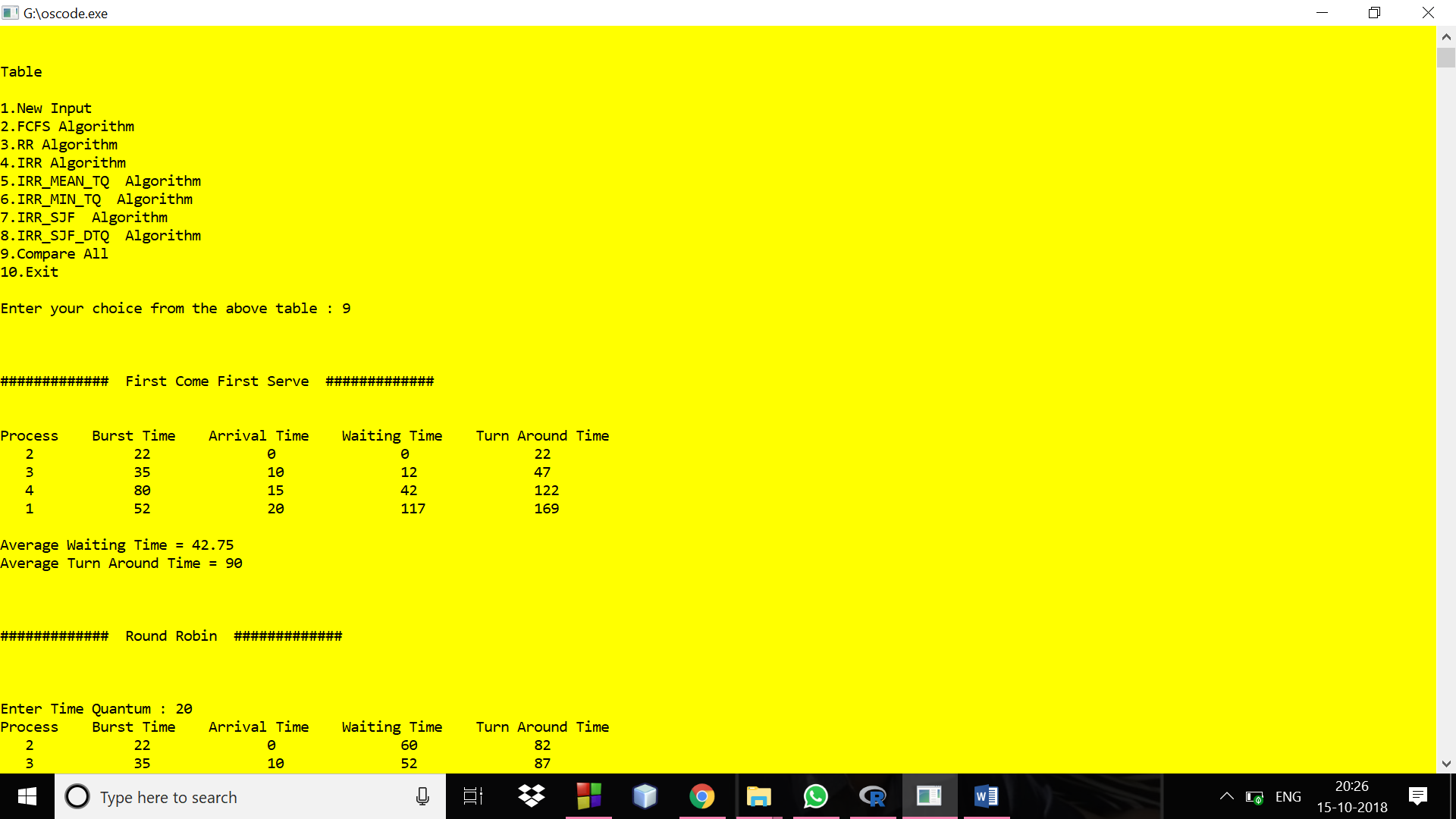
Comparison of the Average Waiting Time of different Scheduling Algorithms

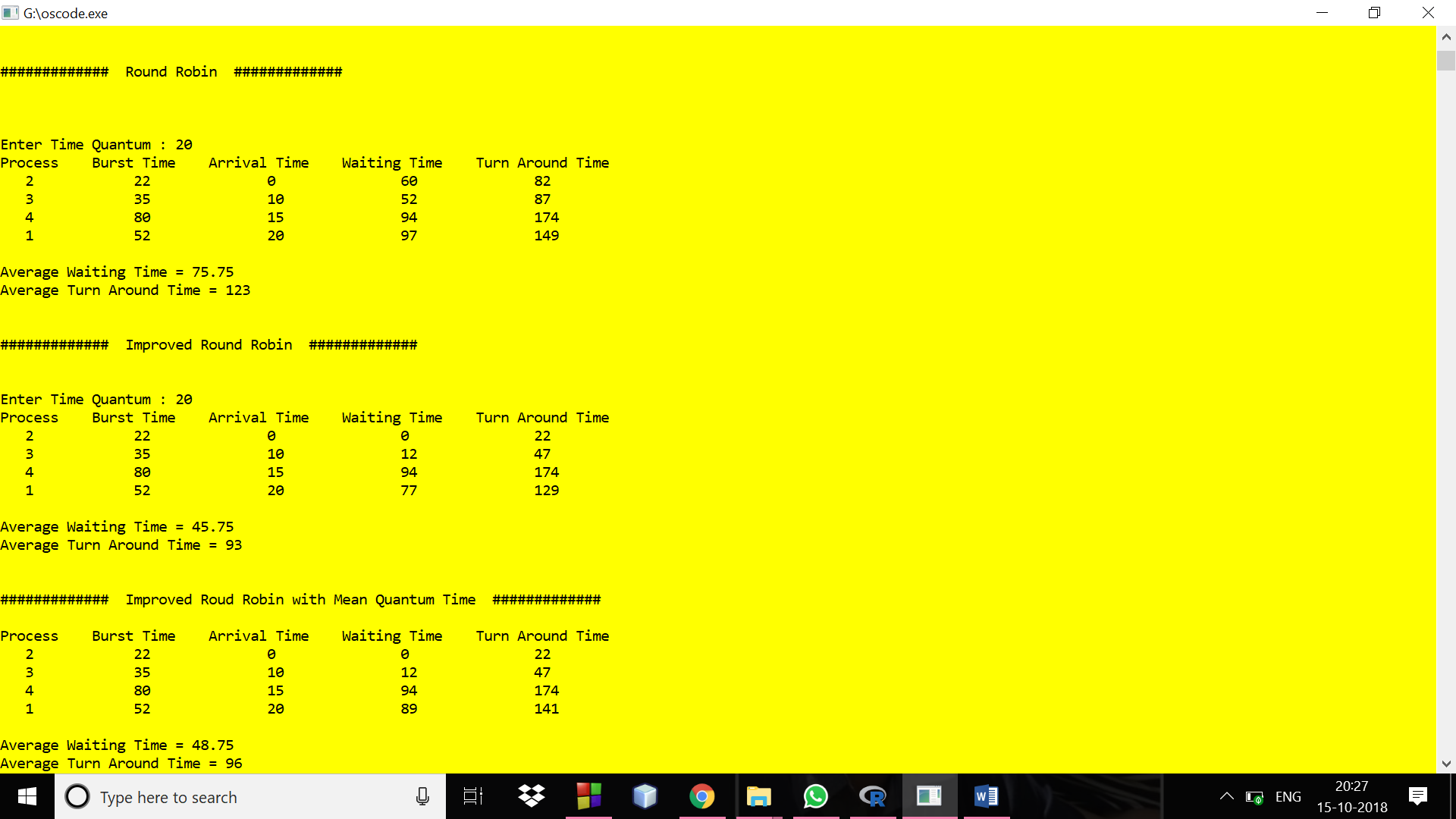


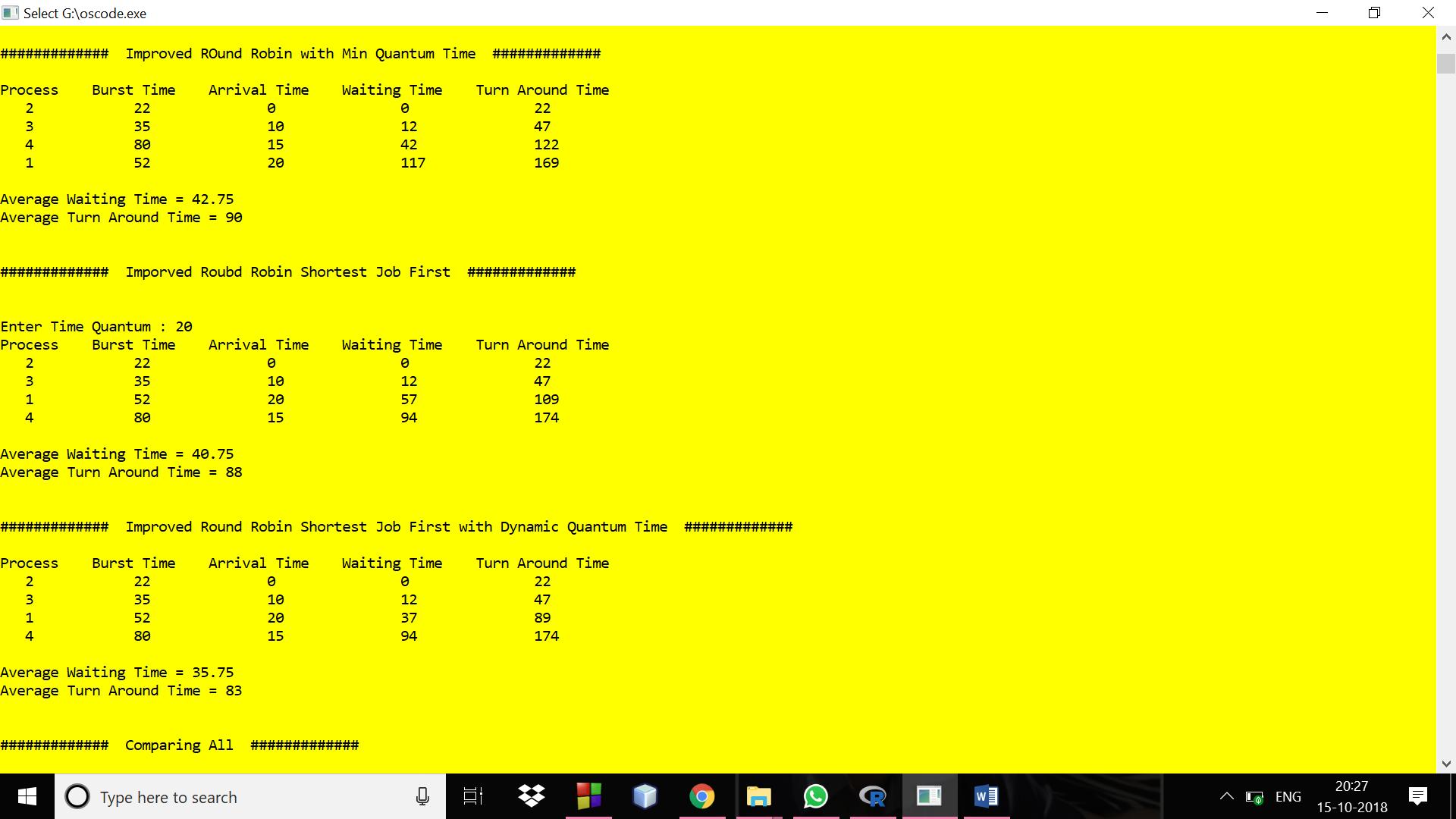
Comparison of the Average TurnAround Time of different Scheduling Algorithms

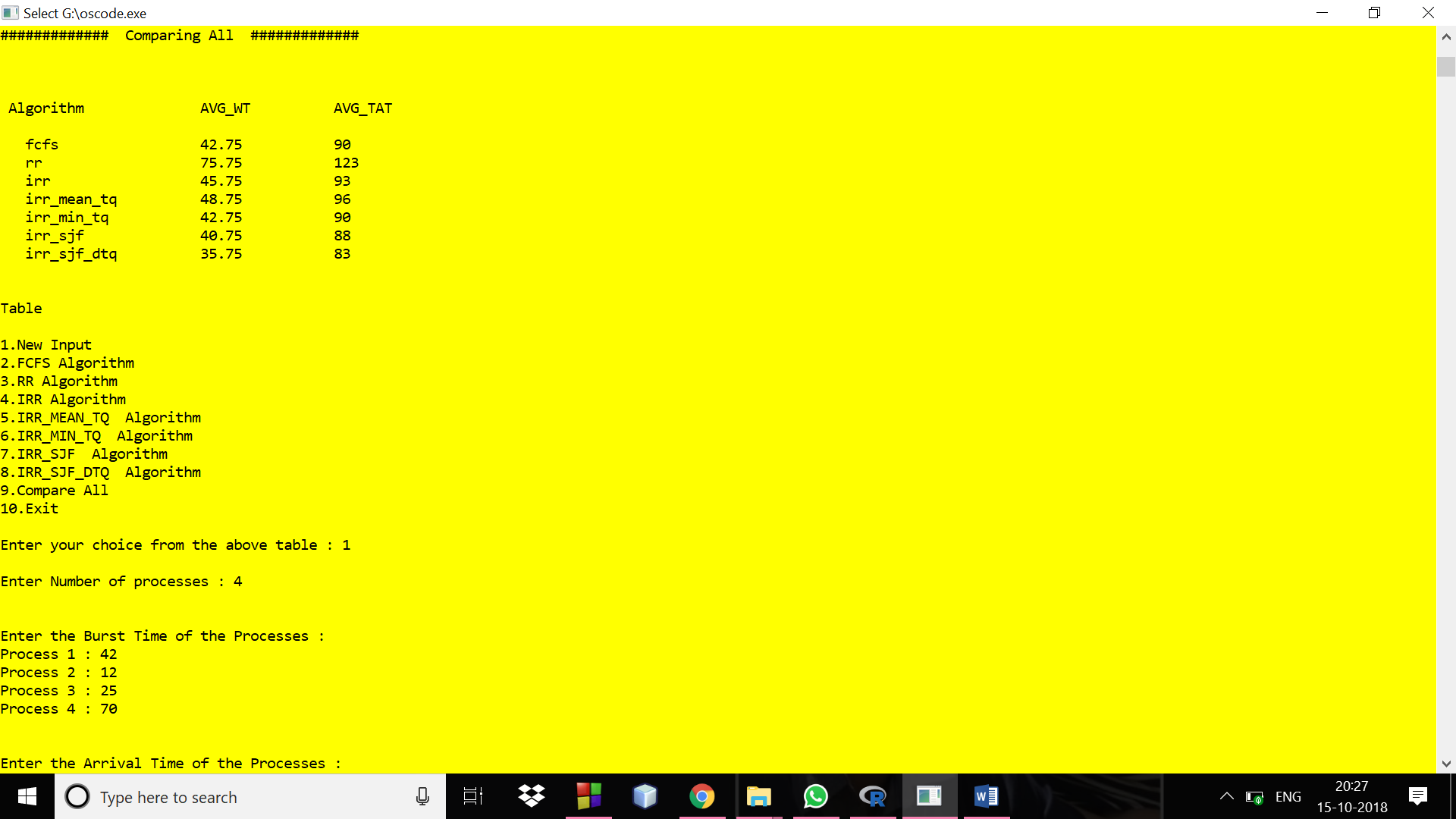
OUTPUT SCREENSHOTS OF THE IMPLEMENTATION











1. **PARAMETERS IDENTIFIED FOR EVALUATION**

**OF PROPOSED MODEL**

In order to evaluate over the existing round-robin algorithm, we follow a certain methodology where we see the time quantum makes the operating system according to the burst time of the already existing set of the process in the ready queue.

While the operating system is installed for the first time, it begins with time quantum which will be equal to the burst time of the first dispatched process and which will be subjected for a change after the end of the first time quantum. So, we assume that the system will immediately take an advantage of this method.

* **ONE PARAMETER TO BE NOTICED:**

Each job insertion will be achieved in O(1), but the given job selection (to run next) and its deletion would require O(n) time, where n is the number of process in the queue. Now simply maintaining all the ready tasks in a sorted priority queue that will be used as the linked list in the data structure.

Now when a task arrives we see a record for it can be inserted into linked list in O(n) time where n is the total number of processes in the priority queue.Therefore ,the time complexity will be equal to that of a typical linear sorting algorithm which is O(n).

**Whenever a task approaches, it will be sorted according to its burst time in ascending manner and then executed .**

**When a process completes its burst time , it gets deleted from the ready queue automatically** so in this case time quantum completes its execution. If the average time quantum is calculated and compared with the algorithm then the process will be continued till all the process were deleted from the ready queue after the scheduling is done for a given number of jobs, the **THROUGHPUT** is calculated.

Time slice is assumed to be not more than the maximum burst time.

**Attributes that would be Prerequisites for evaluation will be:**

1. **Burst time**
2. **Number of processes**
3. **Time slice**
4. **Arrival Time**

of all the process are known before submitting the processes to the processor. Large number of processes are assumed in the ready queue for the better efficiency and these cases are assumed to be ideal, the context switching time is equal to zero and we see that context switching overhead incurred in transferring from one job to the other.

**Parameters considered during the experiment:**

**Input Parameters**

* + - Burst Time
    - Arrival Time
    - Time Quantum
    - Number of Processes

**Output Parameters**

* + - Average Waiting Time
    - Average Turnaround Time
    - Number of Context Switches
    - Throughput
    - CPU Utilisation

We performed our experiment evaluating the performance new proposed algorithm considering a data set with different arrival time for the process.

**The algorithm works efficiently for a very large number of processes**.

In each case we have compared the experimental results of the round robin scheduling algorithm with a fixed time quantum with our proposed in our algorithm with dynamic time quantum. Here we will assume a constant time quantum which will be equal to round robin in all the cases and dynamic quantum will be calculated by the median formula and the second dynamic time quantum will be calculated by quartile formula when the CPU utilization is 100% in all the cases, since number of context switch cases = 0 as assumed in all ideal cases.

**11. REFERENCES**

1. Manish kumar Mishra, “Improved Round Robin CPU Scheduling Algorithm”, Journal of Global Research in computer science, ISSN - 2229-371X, vol. 3,No. 6, June 2012
2. Aashna Bisht, “ Enhanced Round Robin Algorithm for process scheduling using varying quantum precision”, IRAJ International Conference-proceedings of ICRIEST- AICEEMCS,29th Dec 2013, Pune, India.
3. Saroj Hiranwal, “ Adaptive Round Robin Scheduling using shortest Burst Approach Based on smart time slice ”, International Journal of Data Engineering(IJDE), volume2, Issue 3,2 011
4. Lalit Kishor& Dinesh Goyal, “Time Quantum Based Improved Scheduling Algorithms”, International Journal of Advanced Research in Computer science and Software Engineering, ISSN: 2277-128X, Volume 3, Issue 4, April 2013.
5. Ishwari Singh Rajput,Deepa Gupta, “A Priority based round robin CPU Scheduling Algorithms for real time systems”, International journal of Innovations in Engineering and Technology, ISSN:2319-1058, Vol. 1 ISSUE 3rd Oct 2012.
6. P.SurendraVarma, “A Best possible Time quantum for Improving Shortest Remaining Burst Round Robin (SRBRR) algorithms”, International Journal of advanced Research incomputer science and software Engineering, ISSN: 2277 128X, Vol. 2, ISSUE 11, November 2012.
7. H.S. Behera&Brajendra Kumar Swain, “ A New proposed precedence based Round Robin with dynamic time quantum Scheduling algorithm for soft real time systems”, International Journal of advanced Research in Computer science and software Engineering, ISSN:2277- 128X, Vol. 2, ISSUE 6, June 2012.

\*\*\*