# IMPROVED CPU SCHEDULING ALGORITHM

**Team Members:**
**19BCE0743 (B ADITYA KRISHNA)**
**19BCE0751(AMAN ANAND)**
**19BCE0866 (AADITYA PAREEK)**
**19BCE2151 (VIRAJ GUPTA)**
**19BCE2153 (SAHIL TIWARI)**

**Report submitted for the Project Review of:**
**Course Code: CSE2005 – Operating System**
**Slot: L19+L20**
**Professor:MANIKANDAN K**

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

# Abstract

Round Robin (RR) Method is an old, simple scheduling algorithm in Operating system, mainly for timesharing systems. Each process will be having equal priority and is having a time quantum after which the process is preempted. It performs in optimum way in timeshared systems as each and every process will be having an equal amount of static time quantum. Based on the Time quantum the effectiveness of RR algorithm is determined A comprehensive study and analysis of RR algorithm is done and discussed the new and improved version of Round Robin by assigning the processor to processes with shortest remaining burst in round robin manner using the best possible time quantum. Computation of the quantum is done by median and highest burst time. The experimental analysis also shows that this algorithm performs better than RR algorithm by reducing number of context switches, reducing average wait time and also the average turnaround time.

# INTRODUCTION

In computing, scheduling is the method by which work specified by some means is assigned to resources that complete the work. The work may be virtual computation elements such as threads, processes or data flows, which are in turn scheduled onto hardware resources such as processors, network links or expansion cards.

A scheduler may aim at one or more of many goals, for example: maximizing throughput (the total amount of work completed per time unit); minimizing wait time (time from work becoming enabled until the first point it begins execution on resources); minimizing latency or response time (time from work becoming enabled until it is finished in case of batch activity, or until the system responds and hands the first output to the user in case of interactive activity);or maximizing fairness (equal CPU time to each process, or more generally appropriate times according to the priority and workload of each process). . In practice, these goals often conflict (e.g. throughput versus latency), thus a scheduler will implement a suitable compromise.

A process is an instance of a program running in a computer. It includes the current values of the program counters, all the registers, and also the variables. The processes waiting to be assigned to a processor are put in a queue called ready queue. Burst time is amount of time for which a process is being held by the CPU. When a process arrives at the ready queue it is the arrival time. From the time a process is submitted to the time it is completed is the turnaround time. When a process waits in the ready queue that time is the waiting time. The number of times CPU gets switched from a process to another one is called context switching. The optimal and the best scheduling algorithm will have less waiting time, less turnaround time and less number of context switches.

This is what forms the basis of our project.

# LITERATURE REVIEW

| Authors | Title | Concept | Methodology used | Analysis | Finding | Limitations |
|---|---|---|---|---|---|---|
| Rakesh Kumar Yadav , Abhishek K Mishra , Navin Prakash and Himanshu Sharma (2010) | An Improved Round Robin Scheduling Algorithm for CPU scheduling | Cpu scheduling involves a careful examination of pending processes to determine the most efficient way to service the requests. | An Improved Round Robin CPU Scheduling Algorithm allocates the CPU to the processes for one time quantum and if the remaining burst time is less than one time quantum then CPU is again allocated to the currently running process for remaining burst time | The proposed algorithm reduces the number of context switches to provide a fair,efficient and methodical scheduling algorithm | This paper describes an improvement in RR. After improvement in RR it has been found that the waiting time and turnaround time have been reduced drastically. | Though this algorithm is superior than RR algorithm there is no significant difference in WT and TAT as compared to other algorithms. This algorithm can be modified in near future |
| [2]<br>Rami J.Matarneh (<br>2011) | “Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Processes ” | The performance and efficiency of multitasking operating systems mainly depends on the used CPU scheduling algorithm where the CPU is one of the primary computer resources | 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. Both the cases will have their own functioning. | This research provides definitive answer to the question of optimal timing by using dynamic time quantum instead of fixed time quantum. | Based on the experiments and calculations modified algorithm radically solves the fixed time quantum problem which is considered a challenge for round robin algorithm. | This dynamic time quantum based scheduling algorithm can be improved which will involve both SJF and round robin with dynamic time quantum. |

| | | | | | | |
|---|---|---|---|---|---|---|
| [3]<br>Lalit Kishor and Dinesh Goyal (2013) | “ Time Quantum Based Improved Scheduling Algorithm ” | We propose a Median based Time quantum based scheduling algorithm which is combination of SJF & RR . | *All the jobs in the queue are first aligned as per their burst time in ascending order and them Round robin is applied for improving the performance.* | The context switch, average waiting time , average turnaround time has been calculated and the results were compared . | The proposed algorithm shows better performance than the existing RR algorithm as it has less waiting time ,less turnaround time,<br>More CPU engagement and High Throughput. | Future work can be based on this algorithm modified and implemented for hard real time system |
| [4]<br>Ajit Singh, Priyanka Goyal, Sahil Batra (2010) | An Optimized Round Robin Scheduling Algorithm for CPU Scheduling | The main objective of this paper is to develop a new approach for round robin scheduling which help to improve the CPU efficiency in real time and time sharing operating system. | The proposed algorithm will be executed in three phase which help to minimize a number of performance parameters such as context switches, waiting time and average turnaround time. | The context switch, average waiting time , average turnaround time has been calculated and the results were compared . | It is concluded that the proposed algorithm is superior as it has less waiting response time, usually less pre-emption and context switching thereby reducing the overhead and saving of memory space. | In future a dynamic time quantum based scheduling algorithm can be designed which will involve both SJF and round robin with dynamic time quantum. |
| [5]<br>Radhe Shyam, Sunil Kumar Nandal, Dept. of Computer Science and Engineerin | Improved Mean Round Robin with Shortest Job First Scheduling | Some of the CPU scheduling algorithms are First-Come-First-Served (FCFS), Priority Scheduling , Shortest | Round Robin being the most popular in time shared operating system, but it may not be suitable for real time operating systems | They have compared the result of the proposed IMRRSJF method with Round Robin, | From the above experiments, IMRRSJF algorithm shows better results compare to RR algorithm, | As in the paper they have taken the ideal cases in calculating the turnaround time(TAT) and waiting time(WT). In |

| | | | | | | |
|---|---|---|---|---|---|---|
| g, GJUS&T, Hisar, Haryana, India | | Job First (SJF) and Round Robin (RR). FCFS is the simplest form of CPU scheduling algorithm. These algorithms are easy to implement, but it generally does not provide the best service | because of high turnaround time, waiting time and large number of context switches. This result paper describes an improvement in RR. A simple pseudo code has been designed | Improved RR, Enhanced RR and Self adjustment RR (SARR). The static time quantum (TQ) for RR, IRR and ERR algorithm is 20 taken. In SARR algo. median time quantum is used. They have also compared the result of the proposed IMRRSJF method with Time quantum based improved Scheduling algorithms TQBISA and Modified mean-deviation Round Robin MMDRR . | IRR algorithm, ERR algorithms and SARR algorithm. If no. of process is high than IMRRSJF gives better result compare to TQBISA and MMDRR algorithms. | future they can implement this algorithm in different arrival time of processes. |

| | | | | | | |
|---|---|---|---|---|---|---|
| [6] Manish Kumar Mishra and Dr. Faizur Rashid Department of Information Technology, Haramaya University, Dire Dawa, Ethiopia 2Department of Computer Science, Haramaya University, Dire Dawa, Ethiopia | An improved round robin CPU scheduling algorithm with varying time quantum | The improved Round Robin CPU scheduling algorithm with varying time quantum (IRRVQ) combines the features of SJF and RR scheduling algorithms with varying time quantum | Two different cases have been taken for performance evaluation of our proposed IRRVQ algorithm. | In the case 1, CPU burst time is in random orders and processes arrival time is assumed zero. In the case 2, CPU burst time is in random orders and processes arrival time is assumed non zero. The CPU burst time in ascending or descending orders have not been considered since it gives the same result as the CPU burst time in random orders. | An improved round robin CPU scheduling algorithm with varying time quantum proposed in this paper giving better performance than conventional RR algorithm. The waiting time and turnaround time have been reduced in the proposed IRRVQ scheduling algorithm and hence the system performance has been improved. Simulation results also prove the correctness of the theoretical results | The proposed algorithm can be integrated to improve the performance of the systems. |
| [7] H.S.Behera, R | A New Proposed Dynamic Quantum with Re- | The performance of RR algorithm depends | A time quantum is assigned to the processes and after each | Here ‘n’ processes are taken with input paramete | The values obtained of the output parameters are very low | Future work can be based on this algorithm |

| | | | | | | |
|---|---|---|---|---|---|---|
| Mohanty, Debashree Nayak (2010) | Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis | heavily on the size of the time quantum. For smaller time quantum, the context switching is more and for larger time quantum, response time is more.<br>So performance of RR may decrease for weak time quantum selection. Therefore, an optimal time quantum is necessary. | cycle it is recalculated taking the remaining burst time in account. The process which needs least CPU burst time is the first process and the process with highest is the second process and so on. | rs as arrival time(at), burst time(bt) and quantum time(qt). The output parameters are context switching, average waiting time and average turn around time. Processes are divided into two groups<br>1.with zero arrival time.<br>2.without zero arrival time. | as compared to the values of output parameters in simple round robin algorithm in both the cases i.e. with and without zero arrival time. | modified and implemented for hard real time system |
| [8] Harshal Bharatkumar Parekh, Sheetal Chaudhari (2016) | Improved Round Robin CPU Scheduling Algorithm | A fair time period is given to all the processes. The processes with low priority have smaller time periods than the medium priority processes | Initialise each process by setting flag as ‘false’ in job queue and calculate the time quantum required. Necessary conditions are applied to carry out smooth functioning of the processes. After finishing of the process | Processes are entered into the processor with predefined input parameters.<br>If a process with very low burst time arrives into the | Context switches and average waiting, turnaround time have substantially decreased in the proposed algorithm. | The algorithm can be operated within an algorithm of selecting the best scheduling algorithm dynamically i.e. based on the type of usage, different algorithms can be |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | and the medium priority processes have a smaller time period than the high priority processes. So this algorithm uses the priority of the processes to assign time periods to processes. | mark it as ‘true’ in job queue and calculate tat, awt and context switches and remove it from ready queue. | ready queue and has high priority, then it should perform its execution soon after the execution of the ongoing process, thus increasing the throughput. | | utilized together to give a better and efficient process scheduling. Other scheduling algorithms can also be used to modify the existing algorithm for effectiveness such as dynamically changing the time quantum. |
| [9]<br>Prof. Rakesh Mohanty, Prof. H. S. Behera Khusbu Patwari, Manas Ranjan Das, Monisha Dash, Sudhashree (*Department of Computer Science and Engineering*<br>*Veer Surendra Sai University of Technology, Burla,* | “Design and Performance Evaluation of a New Proposed Shortest Remaining Burst Round Robin (SRBRR) Scheduling Algorithm” | Round robin approach has a static time quantum that causes larger waiting and turnaround time that can be improved by using dynamic time quantum. | Implementation of RR with sorted remaining burst time with dynamic time quantum concept by taking judiciously the time quantum and ordering of processes. | Total six experiments were performed; three experiments with same arrival time and other three with different arrival time were considered with processes with burst time in increasing, decreasing and random order. Waiting time, turnaroun | In all the experiments, decrease in average waiting time, average turnaround time and number of context switches was observed. | Results with multiprocessor environment was not measured and all processes were assumed to be independent. |

| | | | | | | |
|---|---|---|---|---|---|---|
| *Sambalpur, Orissa, India*) | | | | d time and context switches were measured. | | |
| [10] Ishwari Singh Rajput, Deepa Gupta (*Department of Computer Science and Engineering Amity School of Engineering and Technology, Amity University, Noida, UP, India***)** | “A Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems” | Round robin approach has disadvantages like higher waiting time, turn around time and number of context switches. Proposed algorithm aims to implement a merged version of RR and priority scheduling in order to retain the advantages of RR and integrating the advantages of Priority scheduling | Allocating CPU to every process in Round Robin fashion, according to given priority, for the given time quantum for one time. Arranging processes in increasing order of remaining CPU burst time and assigning new priorities. | Five processes have been defined with CPU burst time and their priorities, these five processes are scheduled in round robin fashion and also according to the proposed algorithm. The context switch, average waiting time, average turnaround time has been calculated and the results were compared. | The proposed algorithm was found to have less average waiting time, turnaround time and number of context switches as compared to simple round robin algorithm. It also reduces the problem of starvation. | Performance of time-sharing systems can be improved with the proposed algorithm and can also be modified to enhance the performance of real time system. |

# PROBLEM STATEMENT:

**To make one scheduling algorithm of our own and compare its waiting time and turn around time with already existing algorithms.**

Some of the CPU scheduling algorithms are First-Come-First-Served (FCFS), Priority Scheduling, Shortest Job First (SJF) and Round Robin (RR). FCFS is the simplest form of CPU scheduling algorithm.

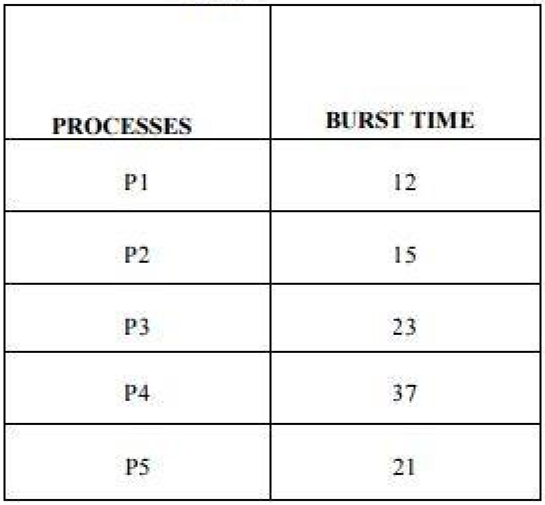
These algorithms are easy to implement, but it generally does not provide the best service. Round Robin being the most popular in time shared operating system, but it may not be suitable for real time operating systems because of high turnaround time, waiting time and large number of context switches. This project describes an improvement in Round Robin.

## EXISTING APPROACH

Round Robin (RR) Method is an old, simple scheduling algorithms in Operating system, mainly for time-sharing systems. Each process will be having equal priority and is having a time quantum and after that the process will be under preemption. The Operating Systems using RR, will first take the first process from the ready queue, will set a timer for interrupting after one-time quantum and gives the processor to that process. Now it will compare processor burst time and time quantum, If the processor burst time is smaller than that of time quantum, then either it will release the processor voluntarily, or it will terminate by issuing an I/O request. The OS starts with the next process which will be in the ready queue. On the other hand, if the processor burst time of a process exceeds the time quantum, then the timer will go off after there is an expiry of one Time Quantum, and it interrupts (preempts) the current process and puts its PCB to the end of the ready queue.

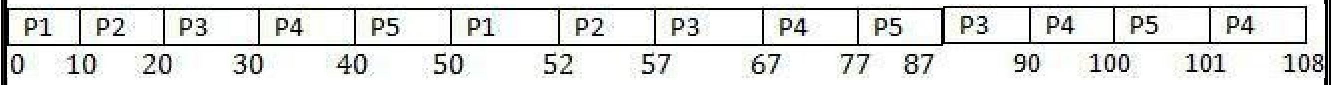
For example, suppose there are 5 different processes as given below:

Table 1



| PROCESSES | BURST TIME |
|---|---|
| P1 | 12 |
| P2 | 15 |
| P3 | 23 |
| P4 | 37 |
| P5 | 21 |

Let us Suppose for the above processes, the time quantum to be 10 ns then the Gantt Chart for Round robin scheduling is: -



| P1 | P2 | P3 | P4 | P5 | P1 | P2 | P3 | P4 | P5 | P3 | P4 | P5 | P4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

0 10 20 30 40 50 52 57 67 77 87 90 100 101 108

Total Wait time = [52-12] + [57-15] + [90-23] + [108-37] + [101-21] = 300
Now Average Waiting time will be: 300/5 = 60
Turn around time = 52 + 57 + 90 + 108 + 101 =408
Average turn around time = 408/5 =81

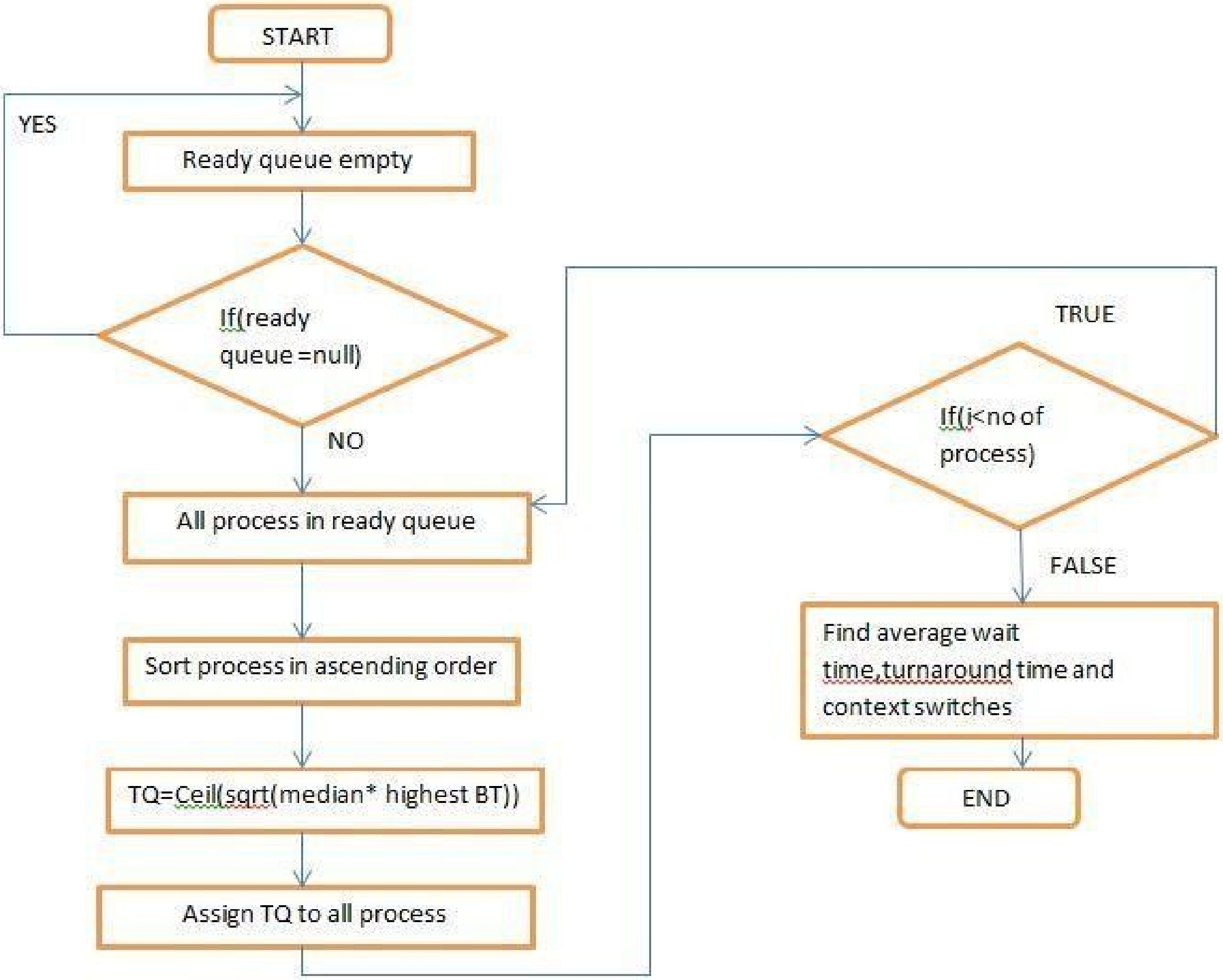
## OUR PROPOSED APPROACH TO IMPROVE PERFORMANCE OF RR SCHEDULING:

In order to get rid of the drawbacks faced in previous round robin scheduling algorithm and minimum context switches, maximum CPU utilization, maximum throughput, minimum turnaround time, minimum waiting time, we have proposed new method is proposed to find the best possible time quantum and make the algorithm an efficient one.

**The proposed algorithm is as follows:**

```
1. Start
2. Sort all the processes in increasing order those which are present in ready queue.
3. While (ready queue! = NULL)
4. Find TQ, where TQ Ceil (sqrt (median * highest Burst time))
5. Assign TQ to process
6. go to step 4 if (i<n)
7. Now update the counter n and go to step 2, If a new process is arrived.
8. End of while
9. Now calculate the Average waiting time, average turnaround time and total Number of context switches required for the process.
10. End
```

## Flowchart for the proposed algorithm:



### Code Implementation in C:

```
#include<stdio.h>
#include<math.h>
int st[10];
void fcfs(){
int bt[10]={0},at[10]={0},tat[10]={0},wt[10]={0},ct[10]={0};
int n,sum=0;
float totalTAT=0,totalWT=0;
printf("Enter number of processes: ");
scanf("%d",&n);
printf("Enter Arrival time and Burst time for each process:\n");
for(int i=0;i<n;i++)
{
     printf("Arrival time of process[%d]: ",i+1);
     scanf("%d",&at[i]);
     printf("Burst time of process[%d]: ",i+1);
     scanf("%d",&bt[i]);
}
for(int j=0;j<n;j++)
{
    sum+=bt[j];
    ct[j]+=sum;
}
for(int k=0;k<n;k++)
{
    tat[k]=ct[k]-at[k];
    totalTAT+=tat[k];
}
for(int k=0;k<n;k++)
{
     wt[k]=tat[k]-bt[k];
     totalWT+=wt[k];
}
printf("P#\t AT\t BT\t CT\t TAT\t WT\t\n\n");
for(int i=0;i<n;i++)
{
     printf("P%d\t %d\t %d\t %d\t %d\t %d\n",i+1,at[i],bt[i],ct[i],tat[i],wt[i]);
}
printf("\nAverage Turnaround Time = %f\n",totalTAT/n);
printf("Average WT = %f\n\n",totalWT/n);
}
void sjf(){
int i,n,p[10]={1,2,3,4,5,6,7,8,9,10},min,k=1,btime=0;
int bt[10],temp,j,at[10],wt[10],tt[10],ta=0,sum=0;
float wavg=0,tavg=0,tsum=0,wsum=0;
```

```
printf(" -------Shortest Job First Scheduling-------\n");
printf("\nEnter the No. of processes :");
scanf("%d",&n);

for(i=0;i<n;i++)
{
printf("\tEnter the burst time of %d process :",i+1);
scanf(" %d",&bt[i]);
printf("\tEnter the arrival time of %d process :",i+1);
scanf(" %d",&at[i]);
}

for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
if(at[i]<at[j])
{
temp=p[j];
p[j]=p[i];
p[i]=temp;
temp=at[j];
at[j]=at[i];
at[i]=temp;
temp=bt[j];
bt[j]=bt[i];
bt[i]=temp;
}
}
}
for(j=0;j<n;j++)
{
btime=btime+bt[j];
min=bt[k];
for(i=k;i<n;i++)
{
if (btime>=at[i] && bt[i]<min)
{
temp=p[k];
p[k]=p[i];
p[i]=temp;
temp=at[k];
at[k]=at[i];
at[i]=temp;
temp=bt[k];
bt[k]=bt[i];
bt[i]=temp;
}
}
k++;
}
wt[0]=0;
for(i=1;i<n;i++)
{
sum=sum+bt[i-1];
wt[i]=sum-at[i];
wsum=wsum+wt[i];
}

wavg=(wsum/n);
for(i=0;i<n;i++)
{
ta=ta+bt[i];
tt[i]=ta-at[i];
tsum=tsum+tt[i];
}

tavg=(tsum/n);

printf("************************");
printf("\n RESULT:-");
printf("\nProcess\t Burst\t Arrival\t Waiting\t Turn-around" );
for(i=0;i<n;i++)
{
printf("\n p%d\t %d\t %d\t\t %d\t\t\t%d",p[i],bt[i],at[i],wt[i],tt[i]);
}

printf("\n\nAVERAGE WAITING TIME : %f",wavg);
printf("\nAVERAGE TURN AROUND TIME : %f",tavg);
}
void rr(){
 int i, limit, total = 0, x, counter = 0, time_quantum;
   int wait_time = 0, turnaround_time = 0, arrival_time[10], burst_time[10],
temp[10];
      float average_wait_time, average_turnaround_time;
      printf("Enter Total Number of Processes: ");
      scanf("%d", &limit);
      x = limit;
      for(i = 0; i < limit; i++)
      {
            printf("\nEnter Arrival Time of Process[%d]: ",i+1);
            scanf("%d", &arrival_time[i]);
            printf("Enter Burst Time of Process[%d]: ",i+1);
            scanf("%d", &burst_time[i]);
            temp[i] = burst_time[i];
      }
      printf("Enter Time Quantum: ");
      scanf("%d", &time_quantum);
      printf("\nProcess ID\t\tBurst Time\t Turnaround Time\t Waiting Time\n");
      for(total = 0, i = 0; x != 0;)
      {
    if(temp[i] <= time_quantum && temp[i] > 0)
    {
                total = total + temp[i];
                 temp[i] = 0;
                 counter = 1;
            }
            else if(temp[i] > 0)
            {
                 temp[i] = temp[i] - time_quantum;
                 total = total + time_quantum;
            }
            if(temp[i] == 0 && counter == 1)
            {
               x--;
       printf("\nProcess[%d]\t\t%d\t\t %d\t\t\t %d", i + 1, burst_time[i], total -
arrival_time[i], total - arrival_time[i] - burst_time[i]);
               wait_time = wait_time + total - arrival_time[i] - burst_time[i];
               turnaround_time = turnaround_time + total - arrival_time[i];
               counter = 0;
            }
            if(i == limit - 1)
            {
                 i = 0;
            }
            else if(arrival_time[i + 1] <= total)
            {
                 i++;
            }
            else
            {
                 i = 0;
            }
      }
      average_wait_time = wait_time * 1.0 / limit;

      average_turnaround_time = turnaround_time * 1.0 / limit;

      printf("\n\nAverage Waiting Time: %f", average_wait_time);

      printf("\nAvg Turnaround Time: %f\n", average_turnaround_time);
}

int get_tq(int b[],int s)
```

```
{

int i,j,maxbt,tmp,hbt,median;
float k,l,m;
for(i=0;i<s;i++)
{

for(j=i+1;j<s;j++){

if(b[i]>b[j])

{

tmp=b[i];
b[i]=b[j];
b[j]=tmp;

}

}

}

int pos[s];

int nonzero = 0;

for(int i=0;i<s;i++)
{
if(b[i]!=0){

pos[nonzero] = b[i];

nonzero++;

}
}

if(nonzero == 0)
return 0;

hbt=pos[nonzero-1];

median=pos[nonzero/2];
for(i=0;i<s;i++)
st[i]=b[i];
l=(float)hbt;
```

```
m=(float)median;
k=sqrt(l*m);return(ceil(k));
}

void innovative(){

int turn[100],wait[100],burst[100];

int bt[100],wt[100],tat[100],n,tq;

int i,count=0,swt=0,stat=0,temp,sq=0;

float awt=0.0,atat=0.0;

printf("Enter the no. of processes:");
scanf("%d",&n);

printf(" burst time for all sequences:");
for(i=0;i<n;i++){
scanf("%d",&bt[i]);
st[i]=bt[i];
}

int cc = 0;
int exitflag = 0;
while(1){
tq=get_tq(st,n);

printf("\ntime quantum is ceilceil((highestbt*median)) = %d\n",tq);
if(tq == 0)
printf("All the processes has been executed\n");
for(i=0,count=0;i<n;i++){
if(cc == 0){
burst[i] = st[i];
}

temp=tq;
if(st[i]==0){
count++;
continue;
}

else if(st[i]>tq){
int aq = sq;
st[i]=st[i]-tq;
sq=sq+temp;
printf("Procoess %d from %d to %d\n",i,aq,sq);
}
else if(st[i]>=0){
int aq = sq;
temp=st[i];
st[i]=0;
sq=sq+temp;
turn[i] = sq;
printf("Procoess %d from %d to %d\n",i,aq,sq);
}
}

cc++;
if(n==count){
break;
}
}
for(i=0;i<n;i++)
{
wait[i] = turn[i] - burst[i];
swt=swt+wait[i];
stat=stat+turn[i];
}
awt=(float)swt/n;
atat=(float)stat/n;
printf("\nprocesses :");
for(int i=0;i<n;i++)
printf("%d ",i);

printf("\nTurn Around time for all processes : ");
for(int i=0;i<n;i++)
printf("%d ",turn[i]);
printf("\nBurst time for all processes :");

for(int i=0;i<n;i++)
printf("%d ",burst[i]);

printf("\nWaiting time for all processes :");
 for(int i=0;i<n;i++)
printf("%d ",wait[i]);
printf("\nAvg waiting time is %f\n",awt);
printf("\nAvg turn around time is %f\n",atat);
}

int main()
{
 printf("Welcome to CPU Scheduling:\n\n");
 int c,choice;
 printf("Choice\tAlgorithm used\n1\tFCFS Algorithm\n2\tSJF Algorithm\n3\tRound robin\n4\tOur innovative algorithm\n");
 do
 {
 printf("Enter your choice from the above table: ");
 scanf("%d",&c);
 switch(c)
 {
 case 1:fcfs();break;
 case 2:sjf();break;
 case 3:rr();break;
 case 4:innovative();break;
 default: printf("Please enter choice from 1 to 4 only\n");break;
 }
 printf("\n\nEnter 1 to continue 0 to stop");
 scanf("%d",&choice);
 }while(choice==1);
 }
```

## SYSTEM DESIGN/ARCHITECHTURE:

### HARDWARE REQUIREMENTS

- Processor: Pentium (IV)
- RAM: 256 MB
- Hard Disk: 40 GB

### SOFTWARE REQUIREMENTS

- Platform Used: Ubuntu(Terminal)
- Operating System: WINDOWS 10 & other versions, UBUNTU

Languages: C Program

## Result Analysis:-

We have used 2 datasets:

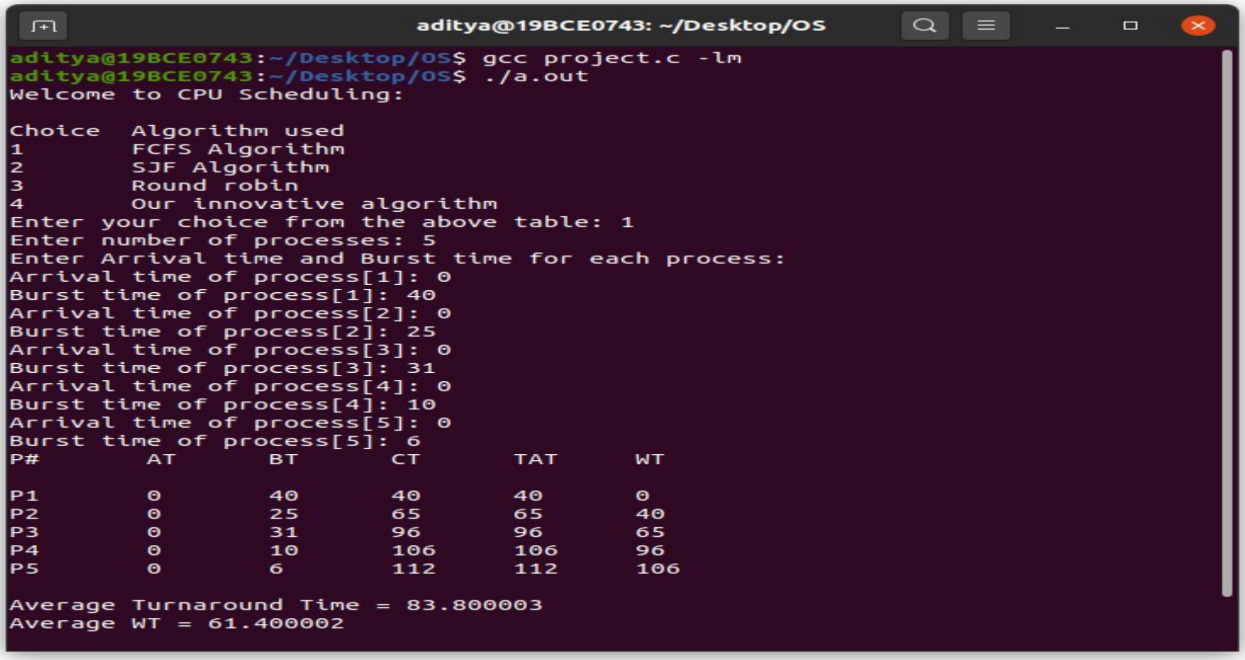
1. We have taken 5 processes as input (Burst Time) and compared our algorithm with other algorithms.
2. We have taken 10 processes as input (Burst Time) and compared our algorithm with other algorithms.

For each Dataset we have calculated the Average Waiting Time and Average Turnaround Time for each of the following processes:-

1. SJF
2. FCFS
3. RR
4. Our innovative algorithm

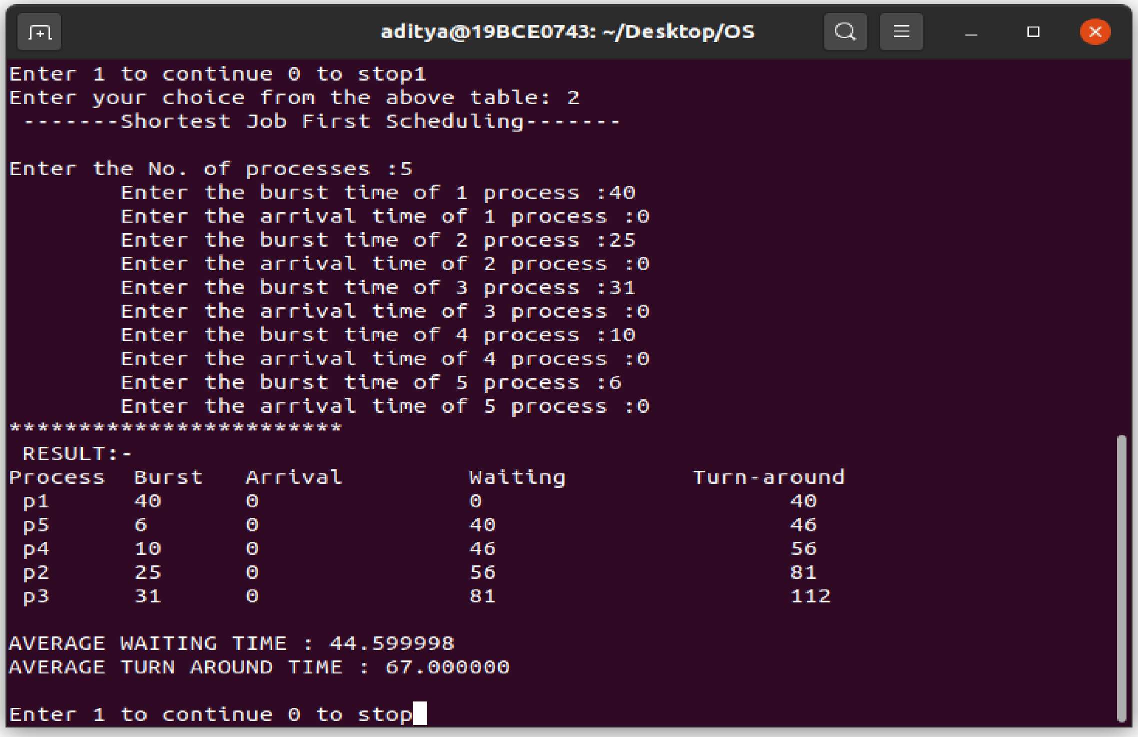
A) FOR 5 Processes:-

**FCFS:-**



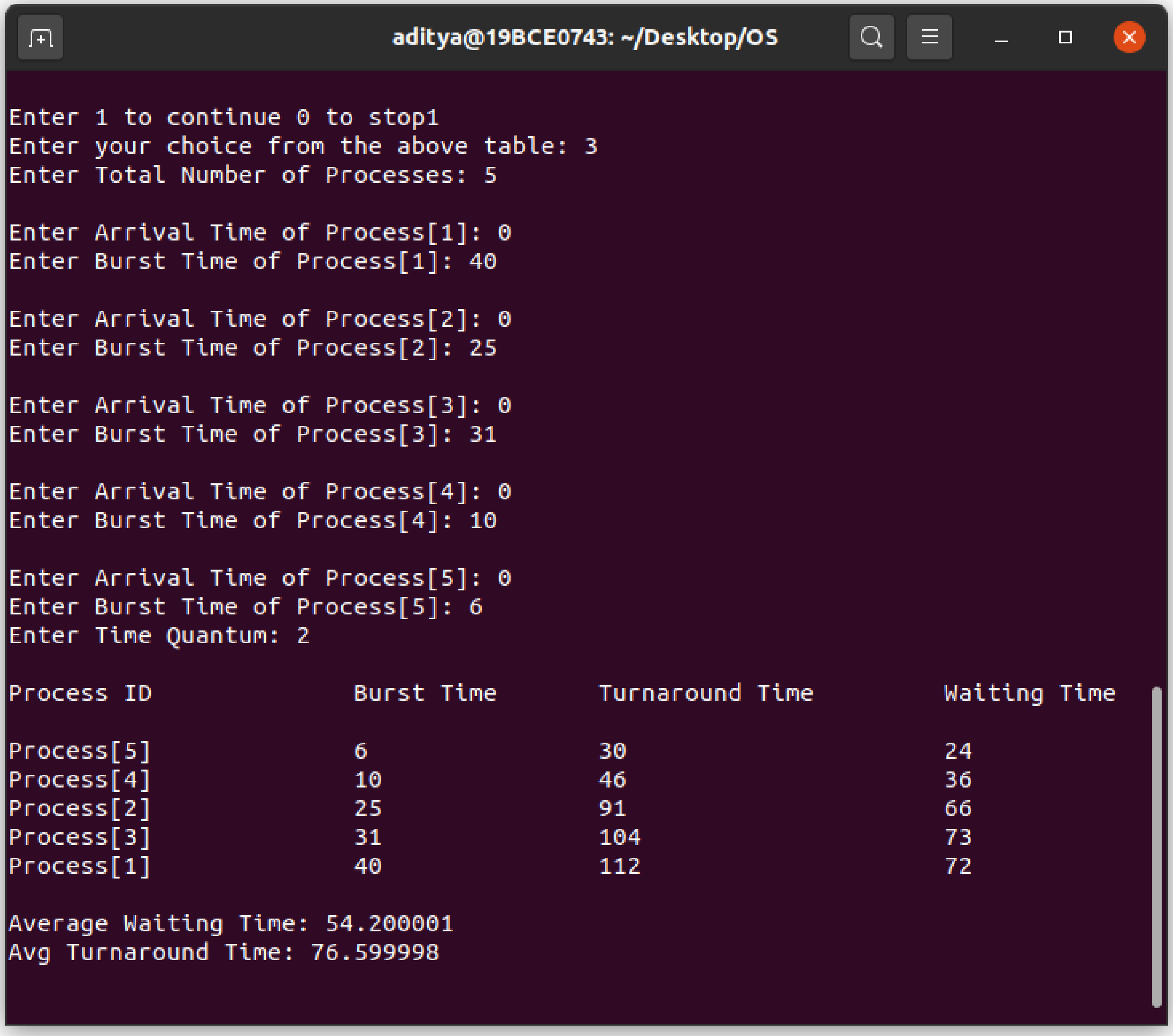
```
aditya@19BCE0743:~/Desktop/OS$ gcc project.c -lm
aditya@19BCE0743:~/Desktop/OS$ ./a.out
Welcome to CPU Scheduling:

Choice   Algorithm used
1        FCFS Algorithm
2        SJF Algorithm
3        Round robin
4        Our innovative algorithm
Enter your choice from the above table: 1
Enter number of processes: 5
Enter Arrival time and Burst time for each process:
Arrival time of process[1]: 0
Burst time of process[1]: 40
Arrival time of process[2]: 0
Burst time of process[2]: 25
Arrival time of process[3]: 0
Burst time of process[3]: 31
Arrival time of process[4]: 0
Burst time of process[4]: 10
Arrival time of process[5]: 0
Burst time of process[5]: 6
P#       AT       BT       CT       TAT      WT

P1       0        40       40       40       0
P2       0        25       65       65       40
P3       0        31       96       96       65
P4       0        10       106      106      96
P5       0        6        112      112      106

Average Turnaround Time = 83.800003
Average WT = 61.400002
```

## 2. SJF Algorithm



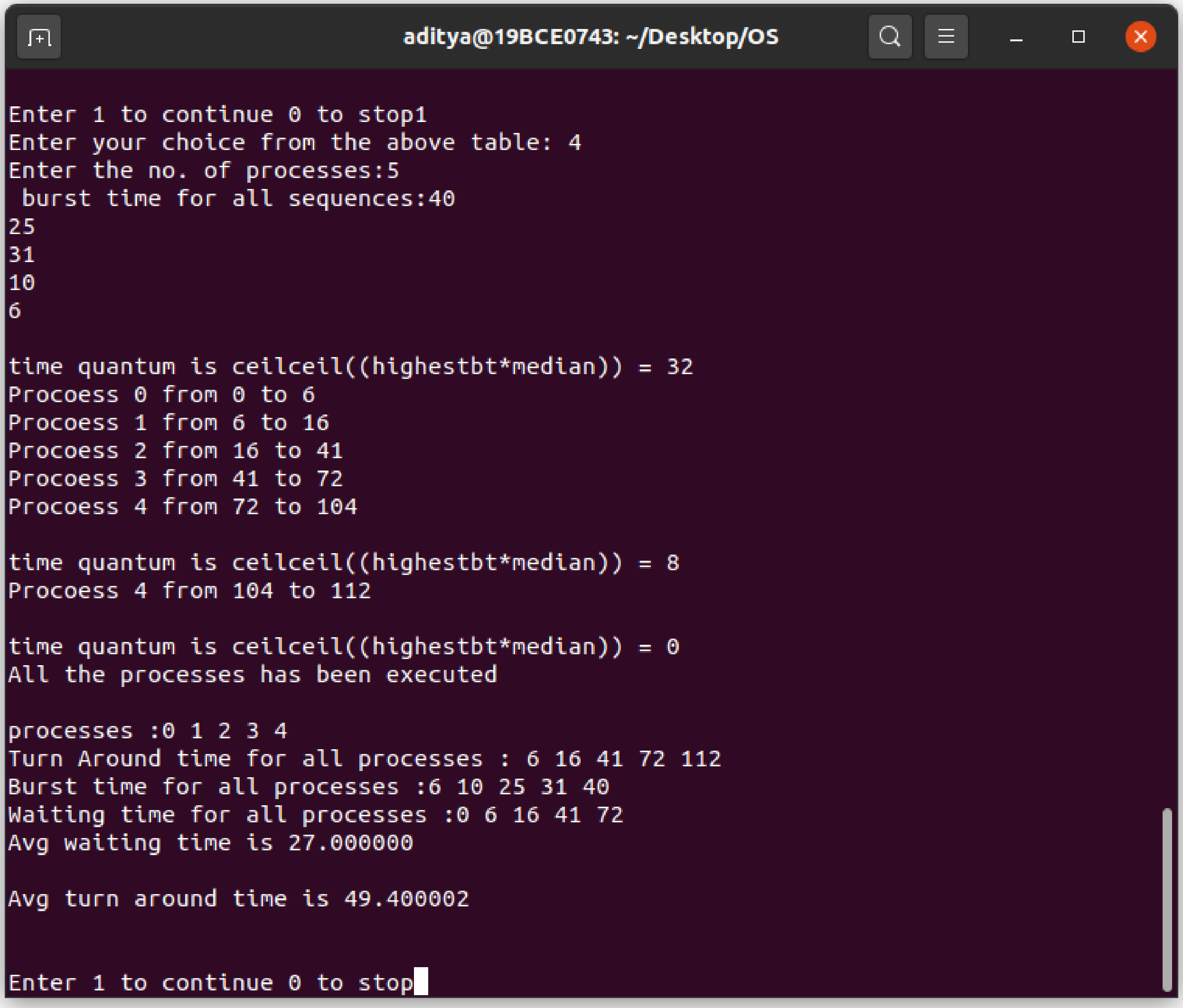
```
Enter 1 to continue 0 to stop1
Enter your choice from the above table: 2
 -------Shortest Job First Scheduling-------

Enter the No. of processes :5
        Enter the burst time of 1 process :40
        Enter the arrival time of 1 process :0
        Enter the burst time of 2 process :25
        Enter the arrival time of 2 process :0
        Enter the burst time of 3 process :31
        Enter the arrival time of 3 process :0
        Enter the burst time of 4 process :10
        Enter the arrival time of 4 process :0
        Enter the burst time of 5 process :6
        Enter the arrival time of 5 process :0
************************
 RESULT:-
Process  Burst   Arrival         Waiting         Turn-around
 p1      40      0               0                      40
 p5      6       0               40                     46
 p4      10      0               46                     56
 p2      25      0               56                     81
 p3      31      0               81                     112

AVERAGE WAITING TIME : 44.599998
AVERAGE TURN AROUND TIME : 67.000000

Enter 1 to continue 0 to stop
```

## 3. RR Algorithm



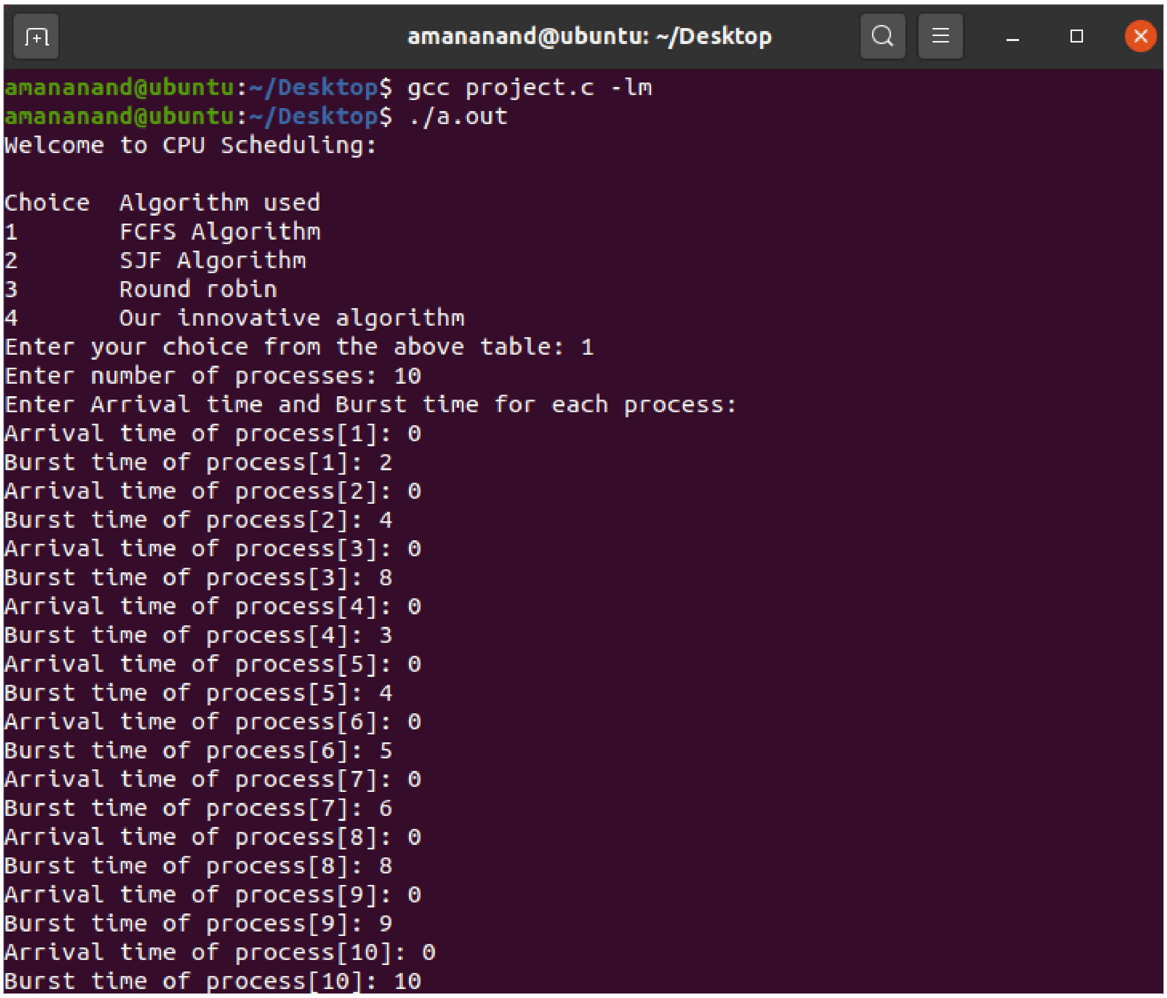
```
Enter 1 to continue 0 to stop1
Enter your choice from the above table: 3
Enter Total Number of Processes: 5

Enter Arrival Time of Process[1]: 0
Enter Burst Time of Process[1]: 40

Enter Arrival Time of Process[2]: 0
Enter Burst Time of Process[2]: 25

Enter Arrival Time of Process[3]: 0
Enter Burst Time of Process[3]: 31

Enter Arrival Time of Process[4]: 0
Enter Burst Time of Process[4]: 10

Enter Arrival Time of Process[5]: 0
Enter Burst Time of Process[5]: 6
Enter Time Quantum: 2

Process ID              Burst Time       Turnaround Time         Waiting Time

Process[5]              6                30                      24
Process[4]              10               46                      36
Process[2]              25               91                      66
Process[3]              31               104                     73
Process[1]              40               112                     72

Average Waiting Time: 54.200001
Avg Turnaround Time: 76.599998
```

## 4.OUR INNOVATIVE ALGORITHM

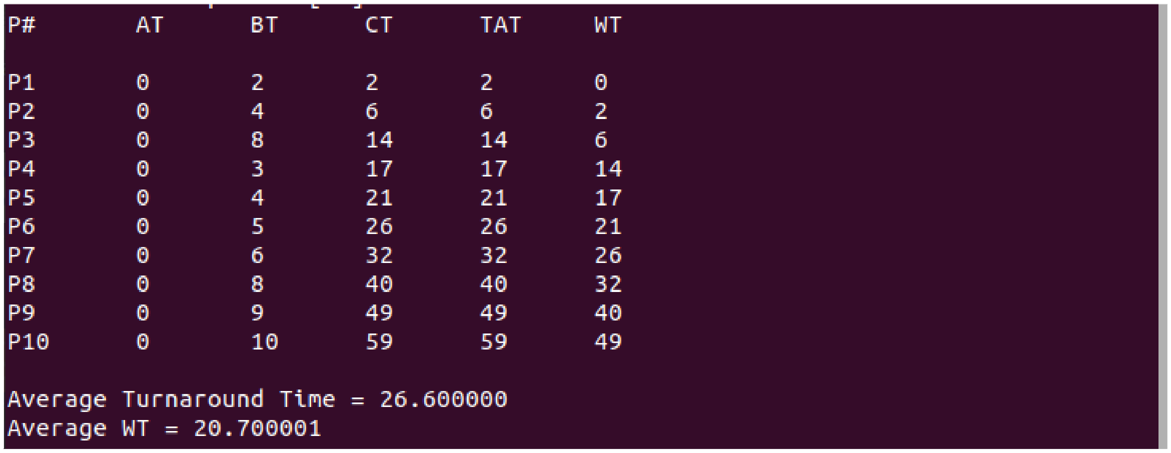


```
Enter 1 to continue 0 to stop1
Enter your choice from the above table: 4
Enter the no. of processes:5
 burst time for all sequences:40
25
31
10
6

time quantum is ceilceil((highestbt*median)) = 32
Procoess 0 from 0 to 6
Procoess 1 from 6 to 16
Procoess 2 from 16 to 41
Procoess 3 from 41 to 72
Procoess 4 from 72 to 104

time quantum is ceilceil((highestbt*median)) = 8
Procoess 4 from 104 to 112

time quantum is ceilceil((highestbt*median)) = 0
All the processes has been executed

processes :0 1 2 3 4
Turn Around time for all processes : 6 16 41 72 112
Burst time for all processes :6 10 25 31 40
Waiting time for all processes :0 6 16 41 72
Avg waiting time is 27.000000

Avg turn around time is 49.400002


Enter 1 to continue 0 to stop
```

**B). FOR 10 PROCESSES:-**
**1.FCFS**

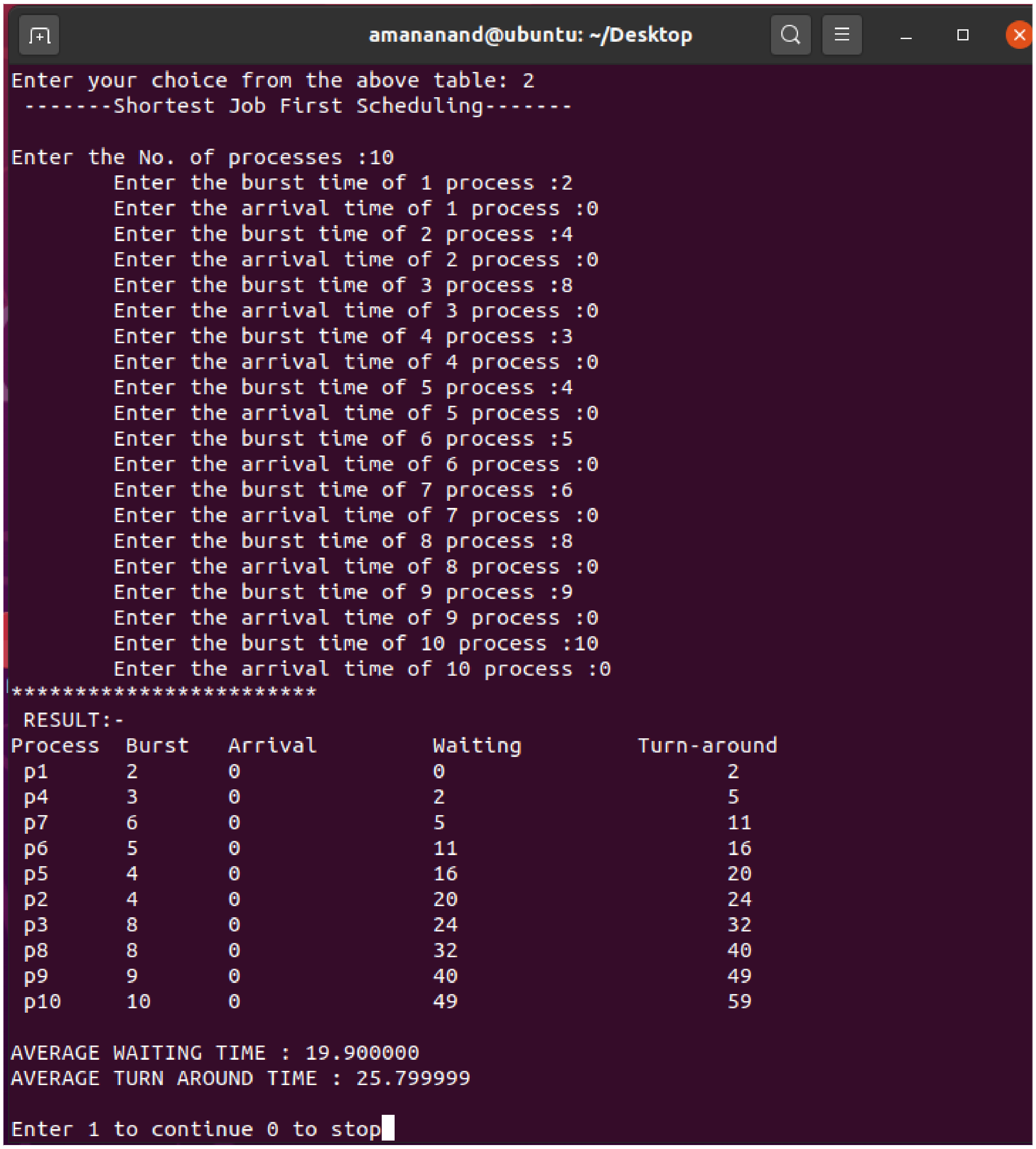


```
amananand@ubuntu:~/Desktop$ gcc project.c -lm
amananand@ubuntu:~/Desktop$ ./a.out
Welcome to CPU Scheduling:

Choice  Algorithm used
1       FCFS Algorithm
2       SJF Algorithm
3       Round robin
4       Our innovative algorithm
Enter your choice from the above table: 1
Enter number of processes: 10
Enter Arrival time and Burst time for each process:
Arrival time of process[1]: 0
Burst time of process[1]: 2
Arrival time of process[2]: 0
Burst time of process[2]: 4
Arrival time of process[3]: 0
Burst time of process[3]: 8
Arrival time of process[4]: 0
Burst time of process[4]: 3
Arrival time of process[5]: 0
Burst time of process[5]: 4
Arrival time of process[6]: 0
Burst time of process[6]: 5
Arrival time of process[7]: 0
Burst time of process[7]: 6
Arrival time of process[8]: 0
Burst time of process[8]: 8
Arrival time of process[9]: 0
Burst time of process[9]: 9
Arrival time of process[10]: 0
Burst time of process[10]: 10
```



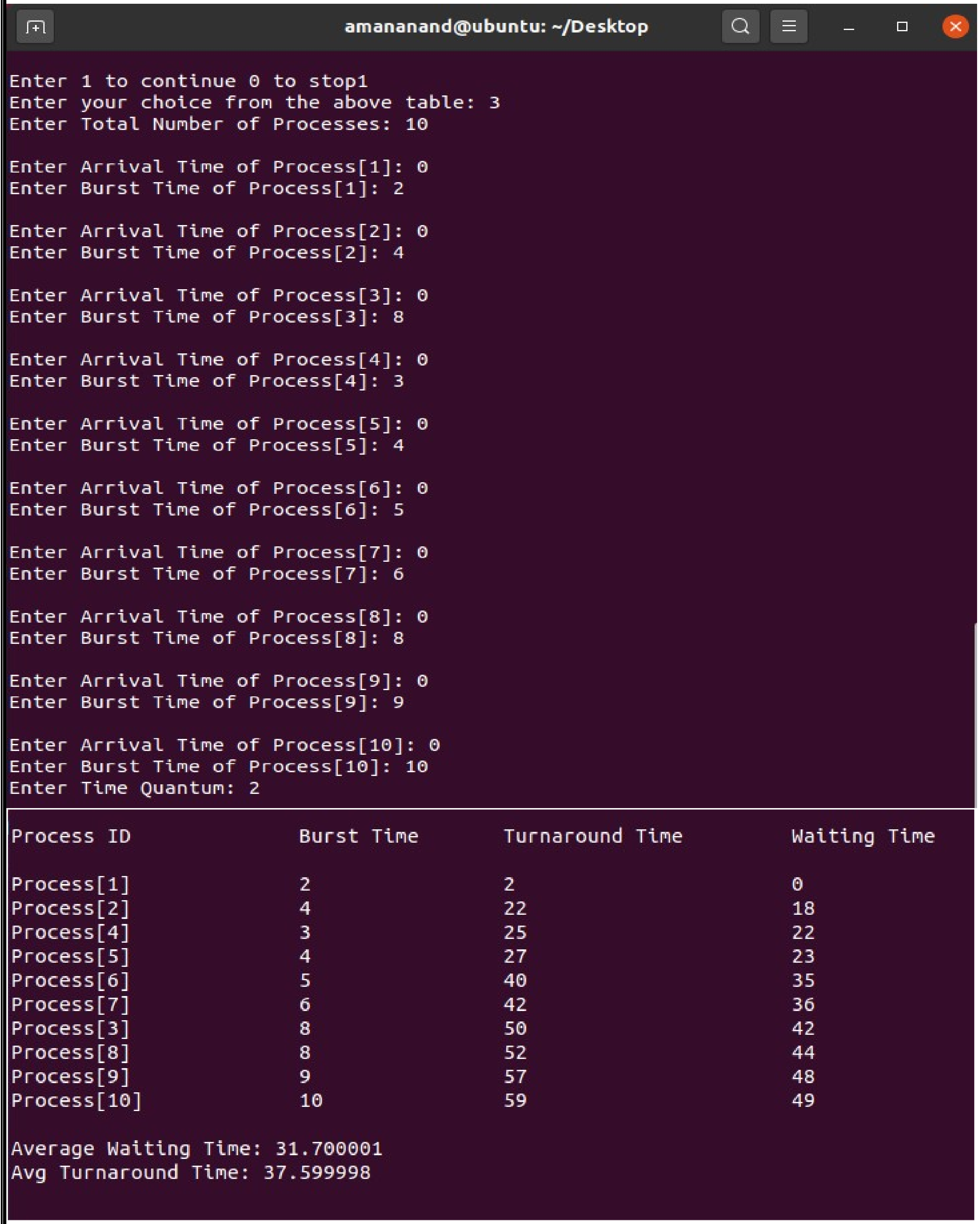
```
P#      AT      BT      CT      TAT     WT

P1      0       2       2       2       0
P2      0       4       6       6       2
P3      0       8       14      14      6
P4      0       3       17      17      14
P5      0       4       21      21      17
P6      0       5       26      26      21
P7      0       6       32      32      26
P8      0       8       40      40      32
P9      0       9       49      49      40
P10     0       10      59      59      49

Average Turnaround Time = 26.600000
Average WT = 20.700001
```

## 2. SJF



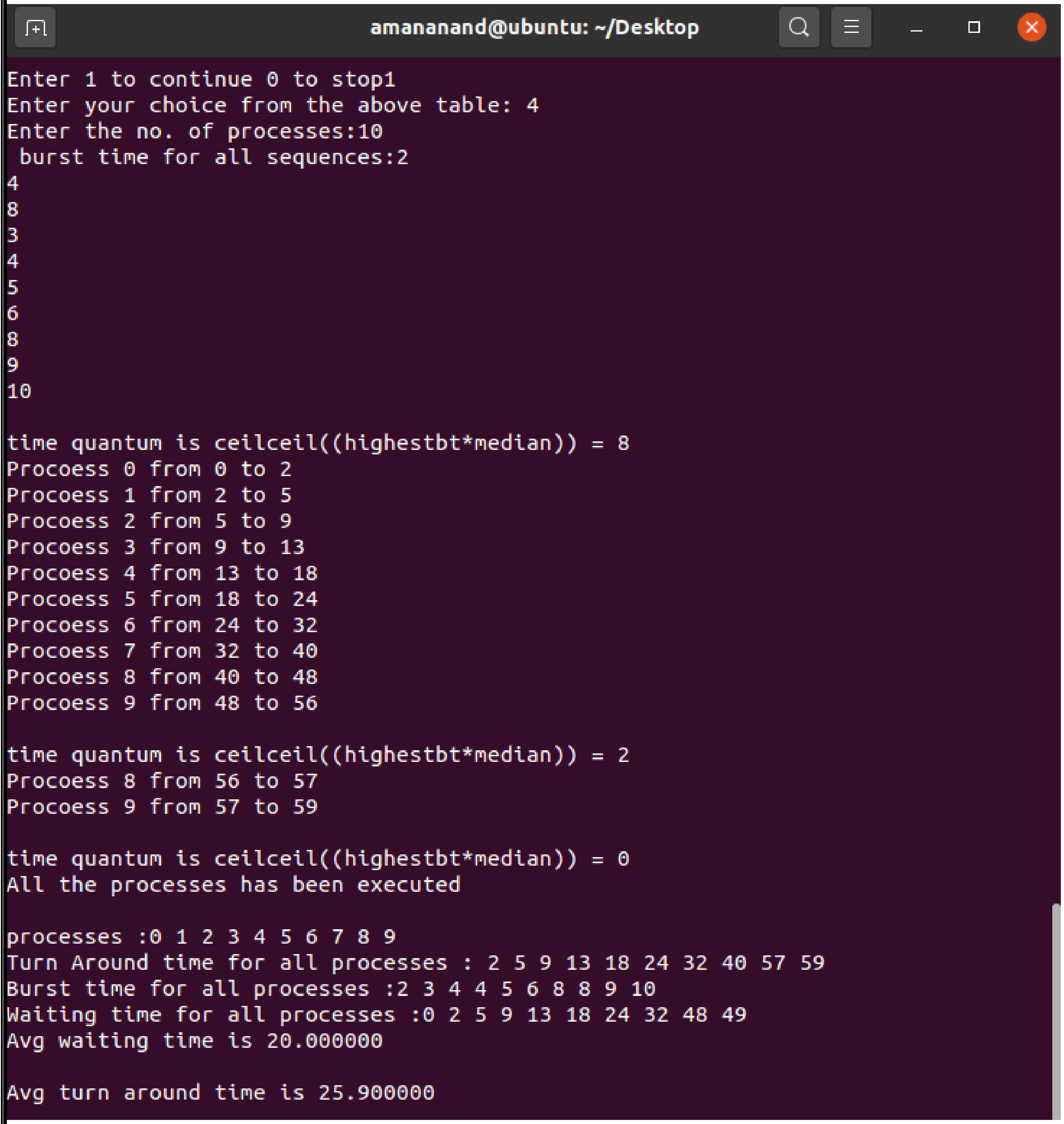
```
Enter your choice from the above table: 2
 -------Shortest Job First Scheduling-------

Enter the No. of processes :10
        Enter the burst time of 1 process :2
        Enter the arrival time of 1 process :0
        Enter the burst time of 2 process :4
        Enter the arrival time of 2 process :0
        Enter the burst time of 3 process :8
        Enter the arrival time of 3 process :0
        Enter the burst time of 4 process :3
        Enter the arrival time of 4 process :0
        Enter the burst time of 5 process :4
        Enter the arrival time of 5 process :0
        Enter the burst time of 6 process :5
        Enter the arrival time of 6 process :0
        Enter the burst time of 7 process :6
        Enter the arrival time of 7 process :0
        Enter the burst time of 8 process :8
        Enter the arrival time of 8 process :0
        Enter the burst time of 9 process :9
        Enter the arrival time of 9 process :0
        Enter the burst time of 10 process :10
        Enter the arrival time of 10 process :0
************************
 RESULT:-
Process  Burst   Arrival         Waiting         Turn-around
 p1      2       0               0                      2
 p4      3       0               2                      5
 p7      6       0               5                      11
 p6      5       0               11                     16
 p5      4       0               16                     20
 p2      4       0               20                     24
 p3      8       0               24                     32
 p8      8       0               32                     40
 p9      9       0               40                     49
 p10     10      0               49                     59

AVERAGE WAITING TIME : 19.900000
AVERAGE TURN AROUND TIME : 25.799999

Enter 1 to continue 0 to stop
```

### 3. RR Algorithm



```
Enter 1 to continue 0 to stop1
Enter your choice from the above table: 3
Enter Total Number of Processes: 10

Enter Arrival Time of Process[1]: 0
Enter Burst Time of Process[1]: 2

Enter Arrival Time of Process[2]: 0
Enter Burst Time of Process[2]: 4

Enter Arrival Time of Process[3]: 0
Enter Burst Time of Process[3]: 8

Enter Arrival Time of Process[4]: 0
Enter Burst Time of Process[4]: 3

Enter Arrival Time of Process[5]: 0
Enter Burst Time of Process[5]: 4

Enter Arrival Time of Process[6]: 0
Enter Burst Time of Process[6]: 5

Enter Arrival Time of Process[7]: 0
Enter Burst Time of Process[7]: 6

Enter Arrival Time of Process[8]: 0
Enter Burst Time of Process[8]: 8

Enter Arrival Time of Process[9]: 0
Enter Burst Time of Process[9]: 9

Enter Arrival Time of Process[10]: 0
Enter Burst Time of Process[10]: 10
Enter Time Quantum: 2

Process ID              Burst Time       Turnaround Time         Waiting Time

Process[1]              2                2                       0
Process[2]              4                22                      18
Process[4]              3                25                      22
Process[5]              4                27                      23
Process[6]              5                40                      35
Process[7]              6                42                      36
Process[3]              8                50                      42
Process[8]              8                52                      44
Process[9]              9                57                      48
Process[10]             10               59                      49

Average Waiting Time: 31.700001
Avg Turnaround Time: 37.599998
```

## 4. Our Innovative Algorithm



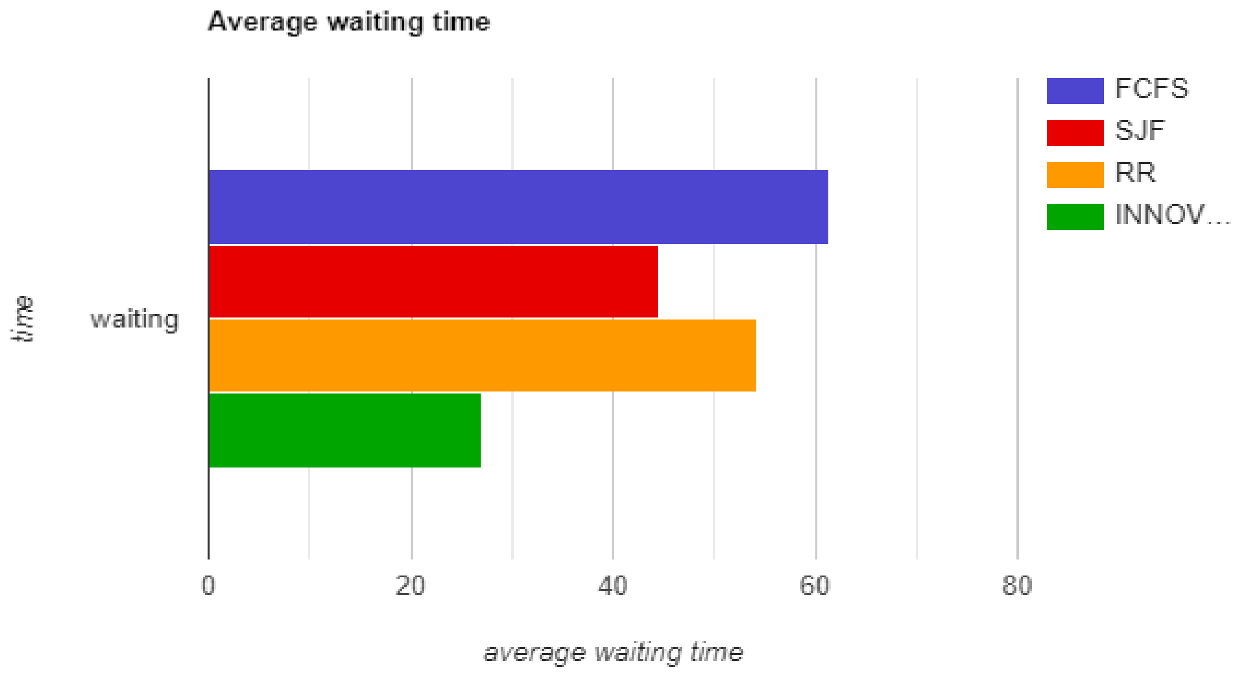
```
Enter 1 to continue 0 to stop1
Enter your choice from the above table: 4
Enter the no. of processes:10
 burst time for all sequences:2
4
8
3
4
5
6
8
9
10

time quantum is ceilceil((highestbt*median)) = 8
Procoess 0 from 0 to 2
Procoess 1 from 2 to 5
Procoess 2 from 5 to 9
Procoess 3 from 9 to 13
Procoess 4 from 13 to 18
Procoess 5 from 18 to 24
Procoess 6 from 24 to 32
Procoess 7 from 32 to 40
Procoess 8 from 40 to 48
Procoess 9 from 48 to 56

time quantum is ceilceil((highestbt*median)) = 2
Procoess 8 from 56 to 57
Procoess 9 from 57 to 59

time quantum is ceilceil((highestbt*median)) = 0
All the processes has been executed

processes :0 1 2 3 4 5 6 7 8 9
Turn Around time for all processes : 2 5 9 13 18 24 32 40 57 59
Burst time for all processes :2 3 4 4 5 6 8 8 9 10
Waiting time for all processes :0 2 5 9 13 18 24 32 48 49
Avg waiting time is 20.000000

Avg turn around time is 25.900000
```

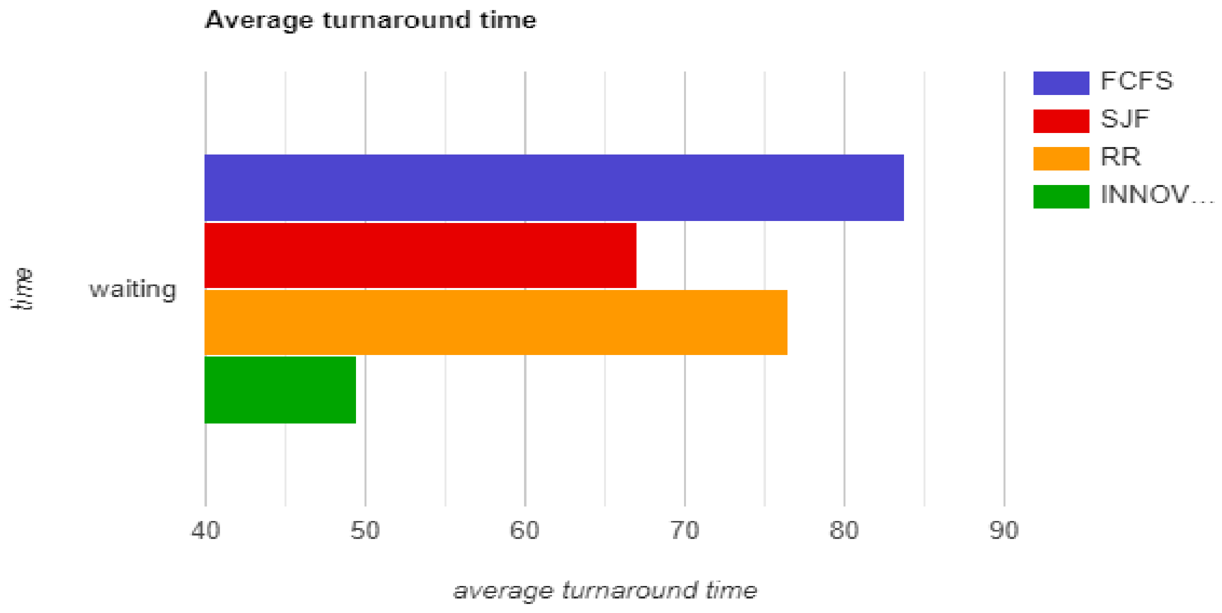
**Results:-**

| Algorithm | Waiting Time | TurnAround Time |
|---|---|---|
| FCFS | 61.4 | 83.8 |
| SJF | 44.5 | 67 |
| RR | 54.2 | 76.5 |
| INNOVATIVE ALGORITHM | 27 | 49.4 |

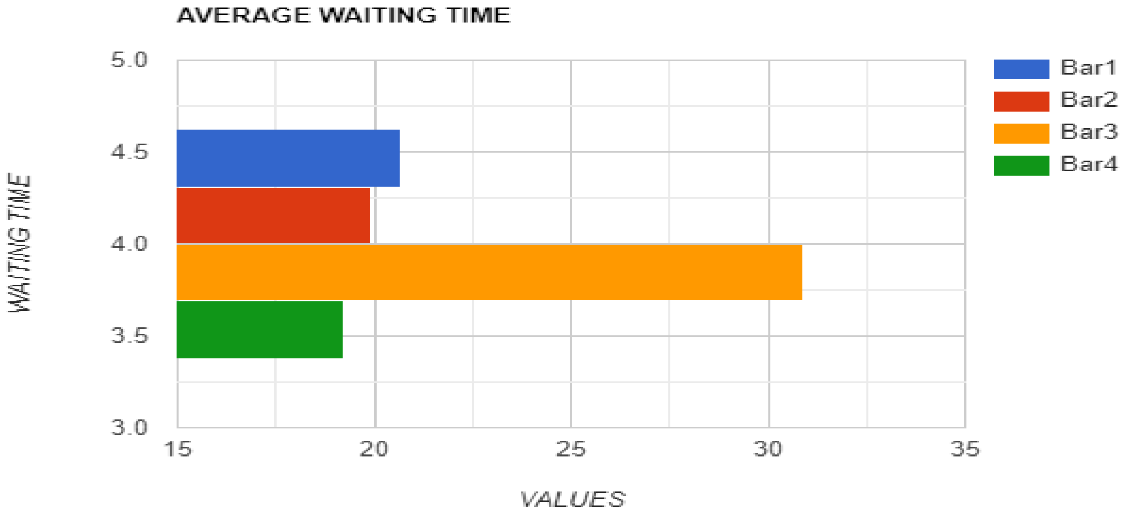
| Algorithm | Waiting Time | TurnAround Time |
|---|---|---|
| FCFS | 20.7 | 26.6 |
| SJF | 19.9 | 25.79 |
| RR | 31.7 | 37.5 |
| INNOVATIVE ALGORITHM | 20.0 | 25.9 |

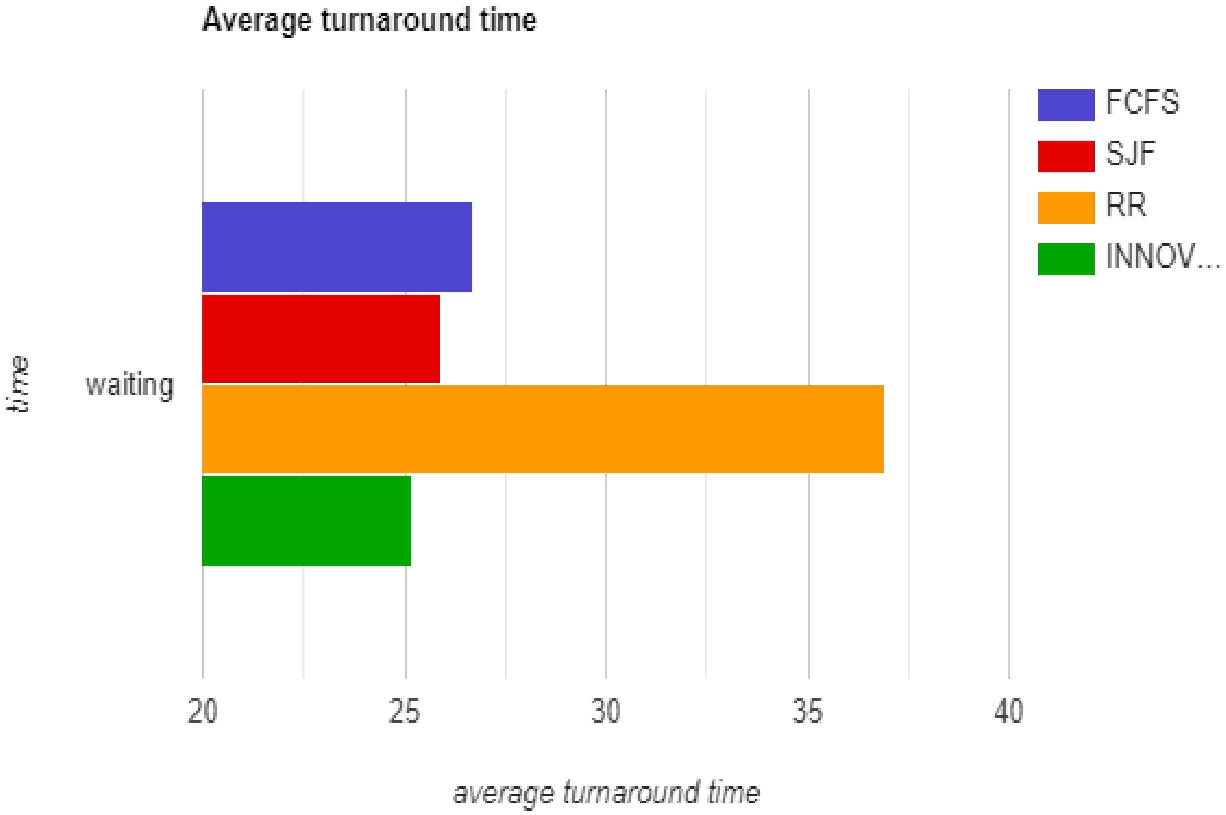
FOR 5 PROCESSES





FOR 10 PROCESSES





## Conclusion and Future Work:-

The average waiting time and turnaround time also  been reduced as compared to Round Robin algorithm which makes the algorithm an efficient one as we know the main goal of CPU scheduling is maximum CPU utilization, minimum waiting time, minimum turnaround time. Thus, we can say that if we calculate time quantum based on this algorithm then we can have a best possible time quantum for scheduling to make the CPU scheduling more efficient one. We also observe from the above stated algorithm, that in the newly proposed algorithm the number of context switches has been reduced from the general round robin algorithm. Clearly it can be concluded that the proposed algorithm will perform better than the static Round Robin Algorithm in terms of all measures like average waiting time, average turnaround time and number of context switches.

For the future work, processes at  different arrival of time can be considered for the proposed algorithm.

# REFERENCES:-

[1] Rakesh Kumar Yadav, Abhishek K Mishra, Navin Prakash and Himanshu Sharma, "An Improved Round Robin Schedduling Algorithm for CPU Scheduling", International Journal on Computer Science and Engineering, Vol. 02, No. 04, 2010, pp. 1064-1066.

[2] A. Silberschatz, P. B. Galvin, and G. Gagne, "Operating System Concepts", 7th Edn., John Wiley and Sons Inc, 2005, ISBN 0-47169466-5.

[3]Rami J. Matarneh,"Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of Now Running Processes", American J. of Applied Sciences 6(10):1831-1837,2009.

[4] H.S. Behera, R. Mohanty, and Debashree Nayak, "A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis", International Journal of Computer Applications, Vol. 5, No. 5, August 2010, pp. 10-15.

[5] Sunita Mohan,"Mixed Scheduling (A New Scheduling Policy)", Proceedings of Insight'09, 25-26 November 2009.

[6] Helmy, T. and A. Dekdouk, "Burst Round Robin as a Proportional-share Scheduling Algorithm", IEEEGCC, http://eprints.kfupm.edu.sa/1462/, 2007.

[7] Debashree Nayak, Sanjeev Kumar Malla, and Debashree Debadarshini, "Improved Round Robin Scheduling using Dynamic Time Quantum", International Journal of Computer Applications, Vol. 38, No. 5, January 2012, pp. 34-38.

[8] Mehdi Neshat, Mehdi Sargolzaei, Adel Najaran, and Ali Adeli, "The New method of Adaptive CPU Scheduling using Fonseca and Fleming's Genetic Algorithm", Journal of Theoretical and Applied Information Technology, Vol. 37, No. 1, March 2012, pp. 1-16.

[9]
Ashkan Emami Ale Agha, SomayyehJafaraliJassbi, "A New Method to Improve Round Robin Scheduling Algorithm with quantum time based on Harmonic-Arithmetic Mean*", International Journal of Information Technology and computer science,2013 ,07, 56-62*

[10] Manish Kumar Mishra and Dr. Faizur Rashid" An improved round robin CPU scheduling algorithm with varying time quantum"

[11]Radhe Shyam, Sunil Kumar Nandal "Improved Mean Round Robin With Shortest Job First Scheduling"

# IMPROVED CPU SCHEDULING ALGORITHM

**Team Members:**

| | |
|---|---|
| 19BCE0743 | (B ADITYA KRISHNA) |
| 19BCE0751 | (AMAN ANAND) |
| 19BCE0866 | (AADITYA PAREEK) |
| 19BCE2151 | (VIRAJ GUPTA) |
| 19BCE2153 | (SAHIL TIWARI) |

**Report submitted for the Project Review of:**

**Course Code: CSE2005 – Operating System**
**Slot: L19+L20**
**Professor:MANIKANDAN K**

# • ABSTRACT:

- New and improved version of Round Robin by assigning the processor to processes with shortest remaining burst in round robin manner using the best possible time quantum.
- Computation of the quantum is done by median and highest burst time. The experimental analysis also shows that this algorithm performs better than RR algorithm by reducing number of context switches, reducing average wait time and also the average turnaround time

# PROBLEM STATEMENT:

- To make one scheduling algorithm of our own and compare its waiting time and turnaround time with already existing algorithms.

**Some of the CPU scheduling algorithms are First-Come-First-Served (FCFS), Priority Scheduling, Shortest Job First (SJF) and Round Robin (RR). FCFS is the simplest form of CPU scheduling algorithm. These algorithms are easy to implement, but it generally does not provide the best service. Round Robin being the most popular in time shared operating system, but it may not be suitable for real time operating systems because of high turnaround time, waiting time and large number of context switches. This project describes an improvement in Round Robin.**

# EXISTING APPROACH

- Round Robin (RR) Method is an old, simple scheduling algorithms in Operating system, mainly for time-sharing systems.
- Each process will be having equal priority and is having a time quantum and after that the process will be under preemption.
- The Operating Systems using RR, will first take the first process from the ready queue, will set a timer for interrupting after one-time quantum and gives the processor to that process.

- Now it will compare processor burst time and time quantum, If the processor burst time is smaller than that of time quantum,then either it will release the processor voluntarily, or it will terminate by issuing an I/O request.
- The OS starts with the next process which will be in the ready queue. On the other hand, if the processor burst time of a process exceeds the time quantum, then the timer will go off after there is an expiry of one Time Quantum, and it interrupts (preempts) the current process and puts its PCB to the end of the ready queue.

# LIMITATIONS

- 1.Low value for time quantum can lead to frequent context switches.
- 2.High value of time quantum can lead to poor response times and make this algorithm behave as that of FCFS.

# PROPOSED APPROACH

- In order to get rid of the drawbacks faced in previous round robin scheduling algorithm and minimum context switches, maximum CPU utilization, maximum throughput, minimum turnaround time, minimum waiting time, we have proposed new method to find the best possible time quantum and make the algorithm an efficient one.

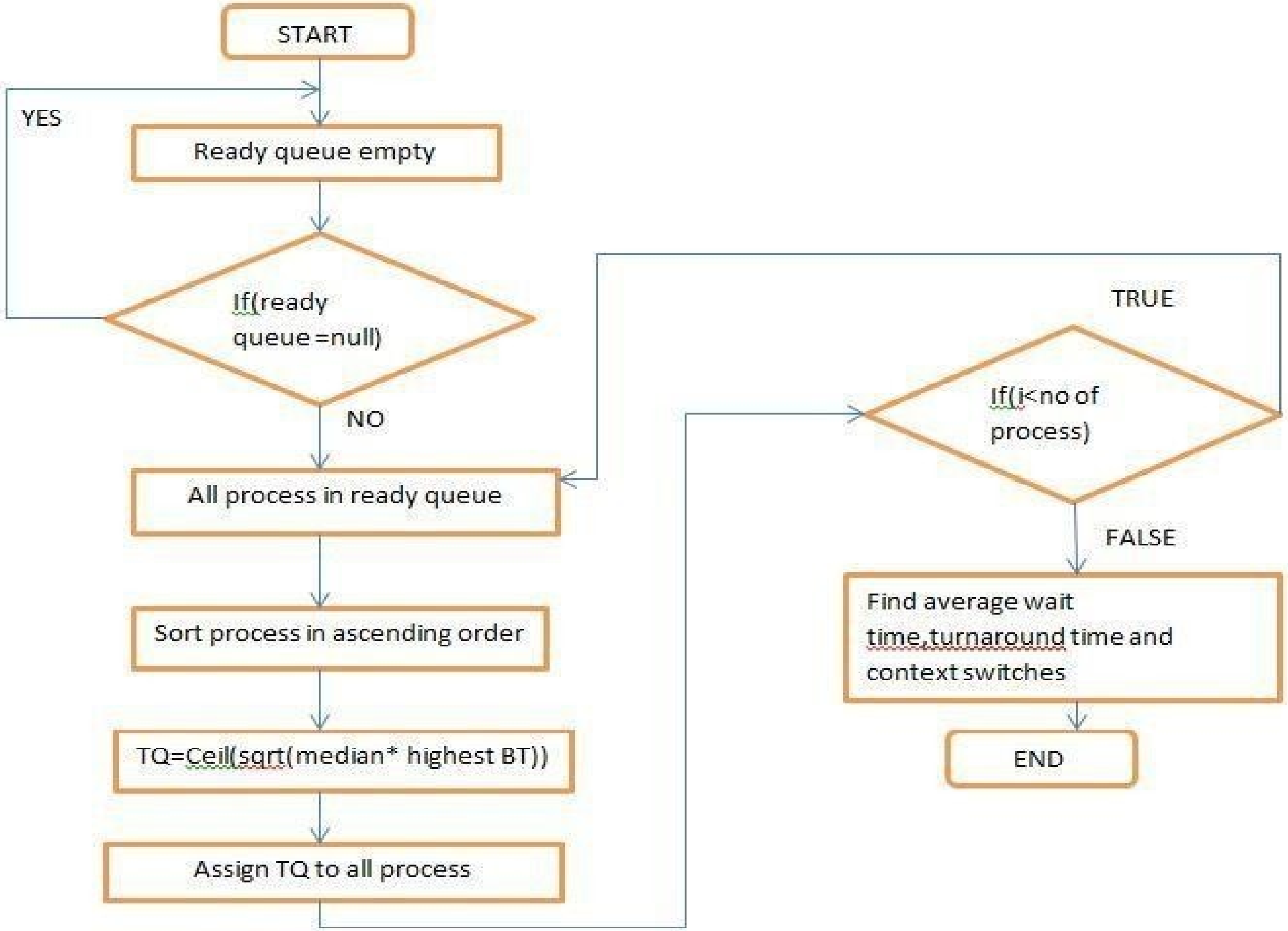
# ALGORITHM

- The proposed algorithm is as follows:
- 1. Start
- 2. Sort all the processes in increasing order those which are present in ready queue.
- 3. While (ready queue! = NULL)
- 4. Find TQ, where TQ Ceil (sqrt (median * highest Burst time))
- 5.Assign TQ to process

# CONTD...

- 6. go to step 4 if (i<n)
- 7. Now update the counter n and go to step 2, If a new process is arrived.
- 8. End of while
- 9. Now calculate the Average waiting time, average turnaround time and total Number of context switches required for the process.
- 10. End

# Flowchart



START
Ready queue empty
YES
If(ready queue =null)
NO
All process in ready queue
Sort process in ascending order
TQ=Ceil(sqrt(median* highest BT))
Assign TQ to all process
TRUE
If(i<no of process)
FALSE
Find average wait time,turnaround time and context switches
END

# SORTING THE PROCESSES

```
int get_tq(int b[],int s)
{
int i,j,maxbt,tmp,hbt,median;
float k,l,m;
for(i=0;i<s;i++)
{
for(j=i+1;j<s;j++){
if(b[i]>b[j])
{
tmp=b[i];
b[i]=b[j];
b[j]=tmp;
}
}
}
```

Sorting the processes on basis of burst time

# TIME QUANTUM CALCULATION

```
int pos[s];
int nonzero = 0;
for(int i=0;i<s;i++)
{
if(b[i]!=0){
pos[nonzero] = b[i];
nonzero++;
}
}if(nonzero == 0)
return 0;
hbt=pos[nonzero-1];
median=pos[nonzero/2];
for(i=0;i<s;i++)
st[i]=b[i];
l=(float)hbt;
m=(float)median;
k=sqrt(l*m);return(ceil(k));
}
```

Time quantum=ceil(sqrt(median* Highestbursttime))

```
void innovative(){
int turn[100],wait[100],burst[100];
int bt[100],wt[100],tat[100],n,tq;
int i,count=0,swt=0,stat=0,temp,sq=0;
float awt=0.0,atat=0.0;
printf("Enter the no. of processes:");
scanf("%d",&n);
printf(" burst time for all sequences:");
for(i=0;i<n;i++){
scanf("%d",&bt[i]);
st[i]=bt[i];
}
int cc = 0;
int exitflag = 0;
while(1){
tq=get_tq(st,n);
printf("\ntime quantum is ceilceil((highestbt*median)) = %d\n",tq);
}
if(tq == 0)
printf("All the processes has been executed\n");
```

# Result Analysis:-

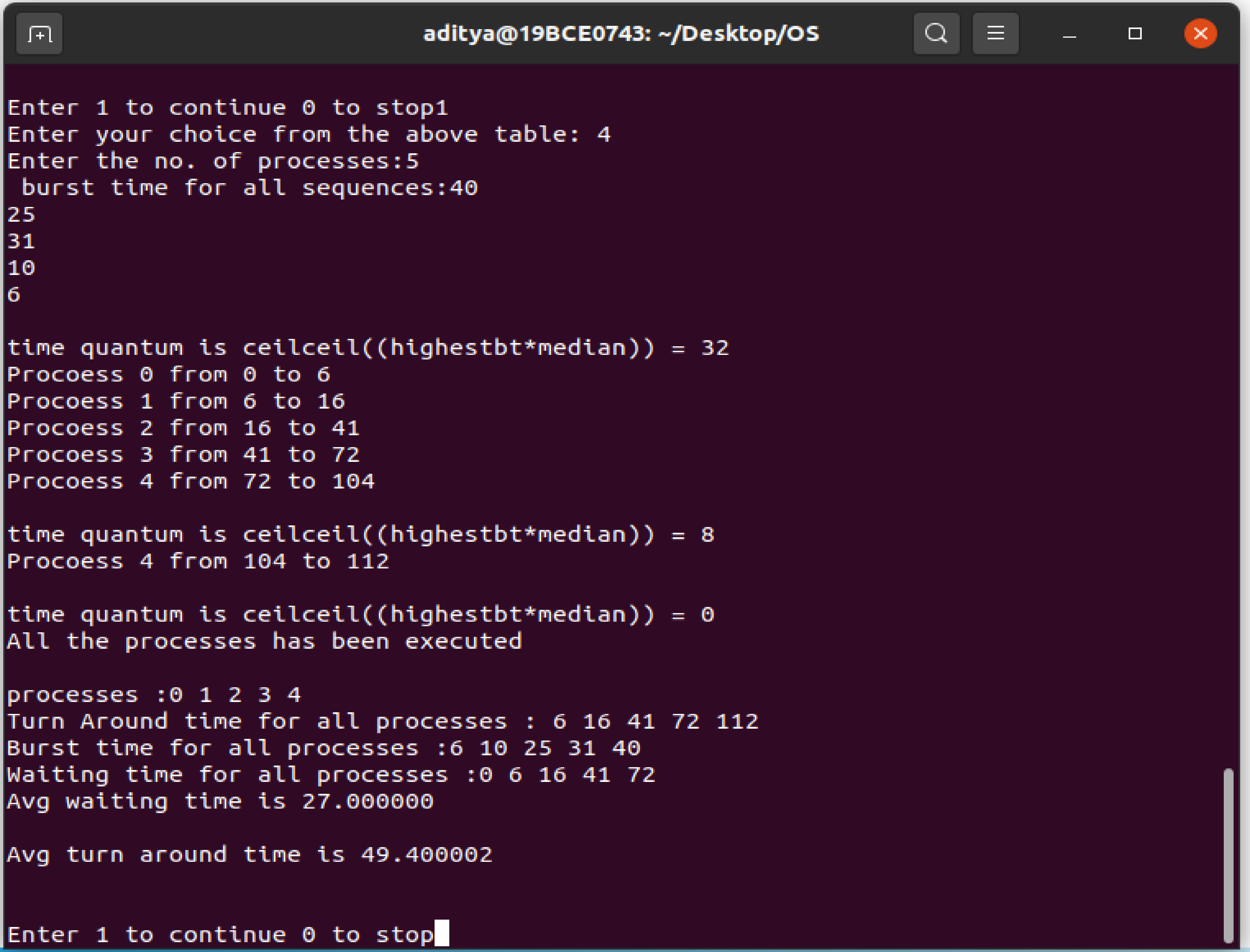
- We have used 2 datasets:
- 1. We have taken 5 processes as input (Burst Time) and compared our algorithm with other algorithms.
- 2. We have taken 10 processes as input (Burst Time) and compared our algorithm with other algorithms.
- For each Dataset we have calculated the Average Waiting Time and Average Turnaround Time for each of the following processes:-
- 1. SJF 2. FCFS 3. RR 4. Our Algorithm

# INPUT DATA

**A)FOR 5 Processes:-**

| PROCESS | BURST TIME |
|---|---|
| 1 | 40 |
| 2 | 25 |
| 3 | 31 |
| 4 | 10 |
| 5 | 6 |

# OUTPUT

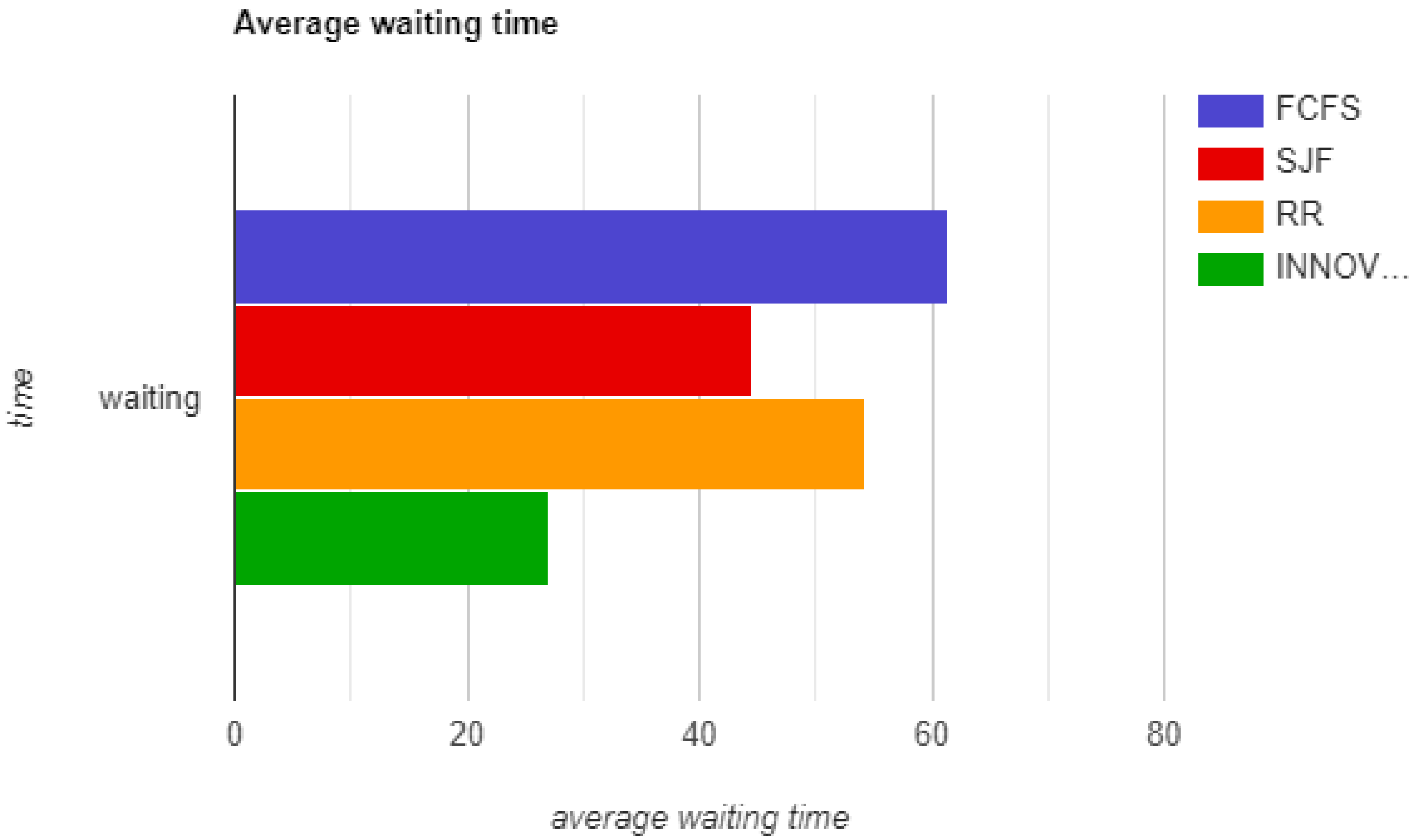


```
Enter 1 to continue 0 to stop1
Enter your choice from the above table: 4
Enter the no. of processes:5
 burst time for all sequences:40
25
31
10
6

time quantum is ceilceil((highestbt*median)) = 32
Procoess 0 from 0 to 6
Procoess 1 from 6 to 16
Procoess 2 from 16 to 41
Procoess 3 from 41 to 72
Procoess 4 from 72 to 104

time quantum is ceilceil((highestbt*median)) = 8
Procoess 4 from 104 to 112

time quantum is ceilceil((highestbt*median)) = 0
All the processes has been executed

processes :0 1 2 3 4
Turn Around time for all processes : 6 16 41 72 112
Burst time for all processes :6 10 25 31 40
Waiting time for all processes :0 6 16 41 72
Avg waiting time is 27.000000

Avg turn around time is 49.400002


Enter 1 to continue 0 to stop
```

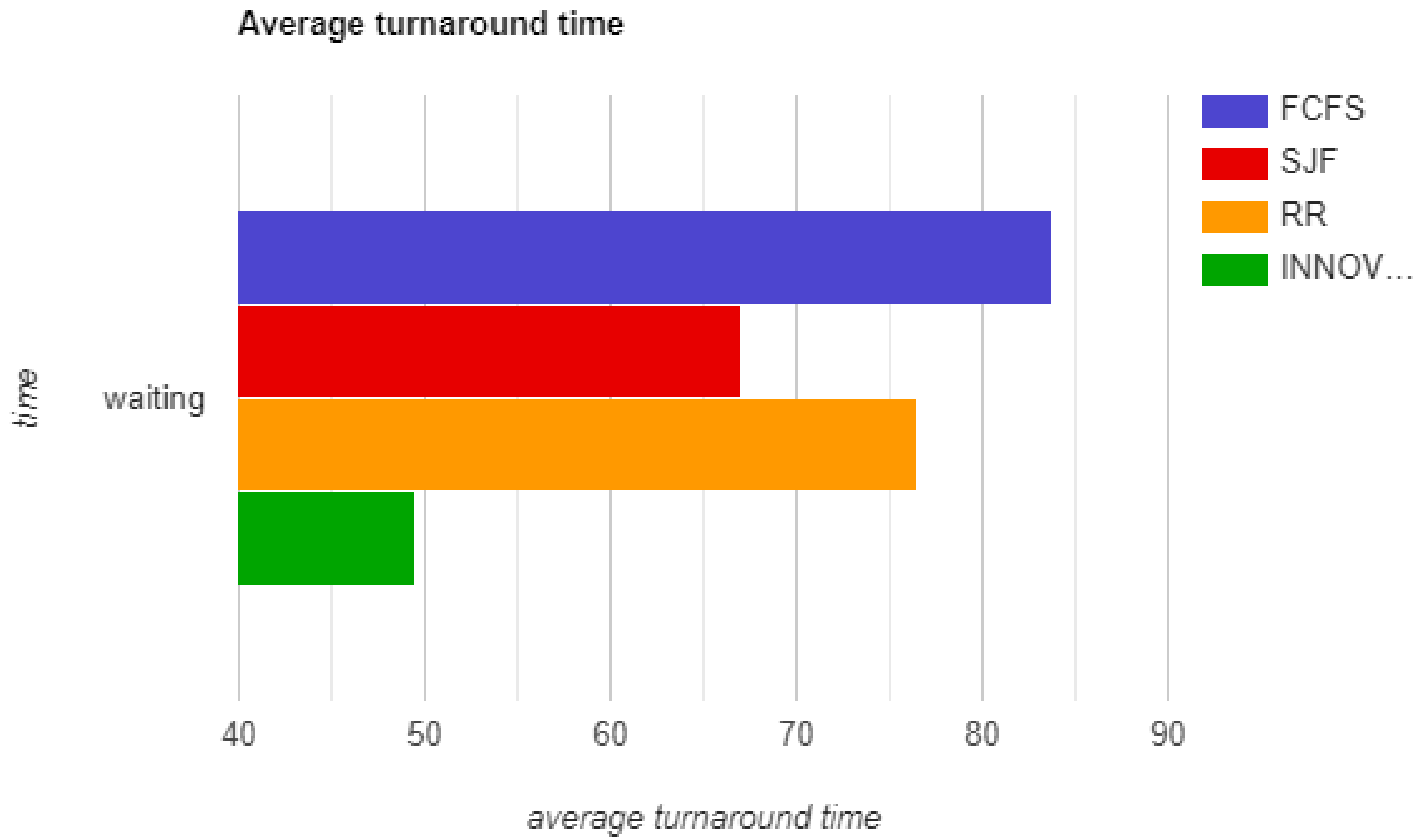
# COMPARISON

| ALGORITHM | WAITING TIME | TURN AROUND TIME |
|---|---|---|
| FCFS | 61.4 | 83.8 |
| SJF | 44.5 | 67 |
| RR | 54.2 | 76.5 |
| OUR ALGORITHM | 27 | 49.4 |

# AVERAGE WAITING TIME



# AVERAGE TURNAROUND TIME

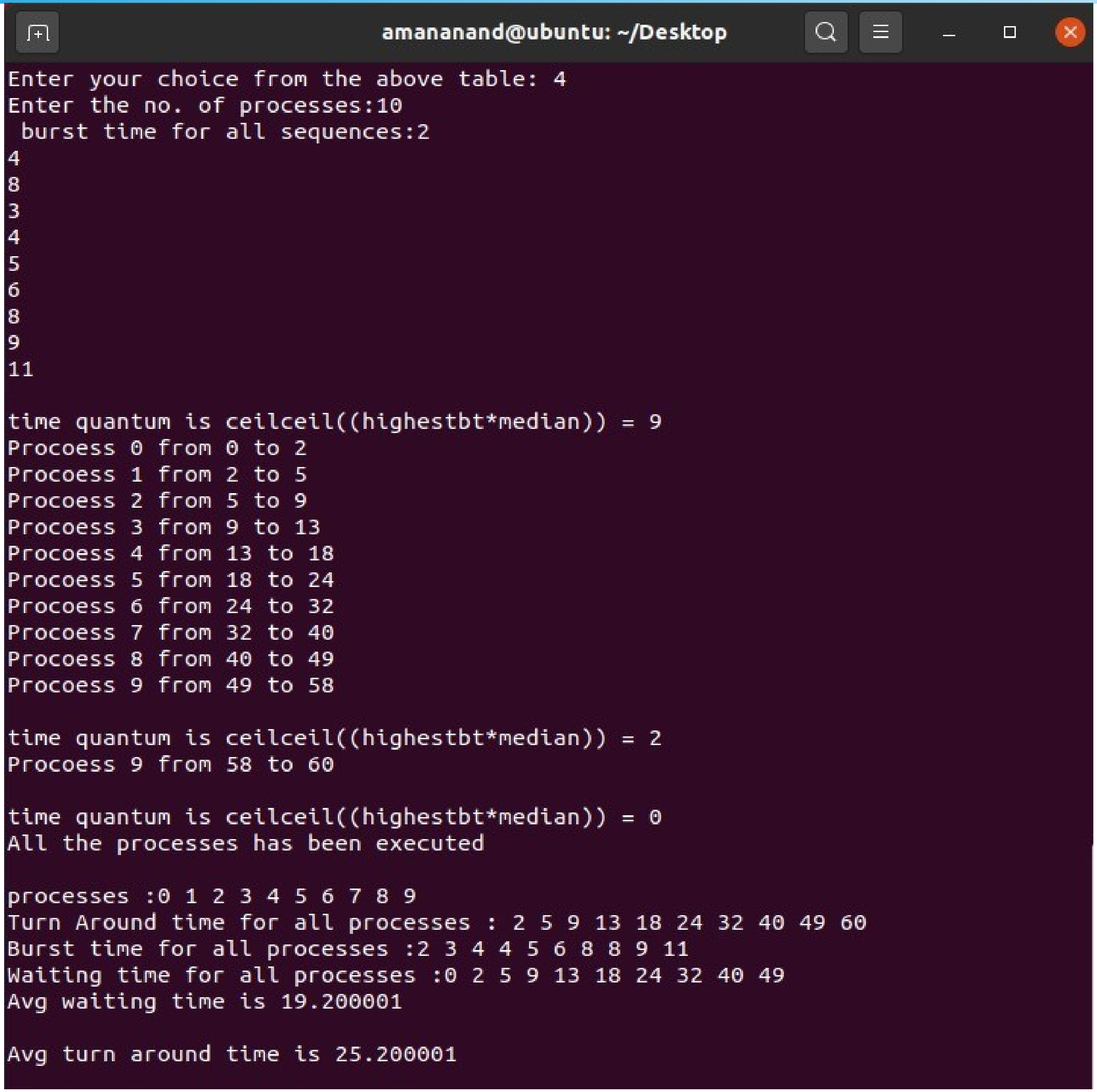


# INPUT DATA

- B)FOR 10 Processes:

| PROCESS | BURST TIME |
|---|---|
| 1 | 2 |
| 2 | 4 |
| 3 | 8 |
| 4 | 3 |
| 5 | 4 |
| 6 | 5 |
| 7 | 6 |
| 8 | 8 |
| 9 | 9 |
| 10 | 11 |

# OUTPUT

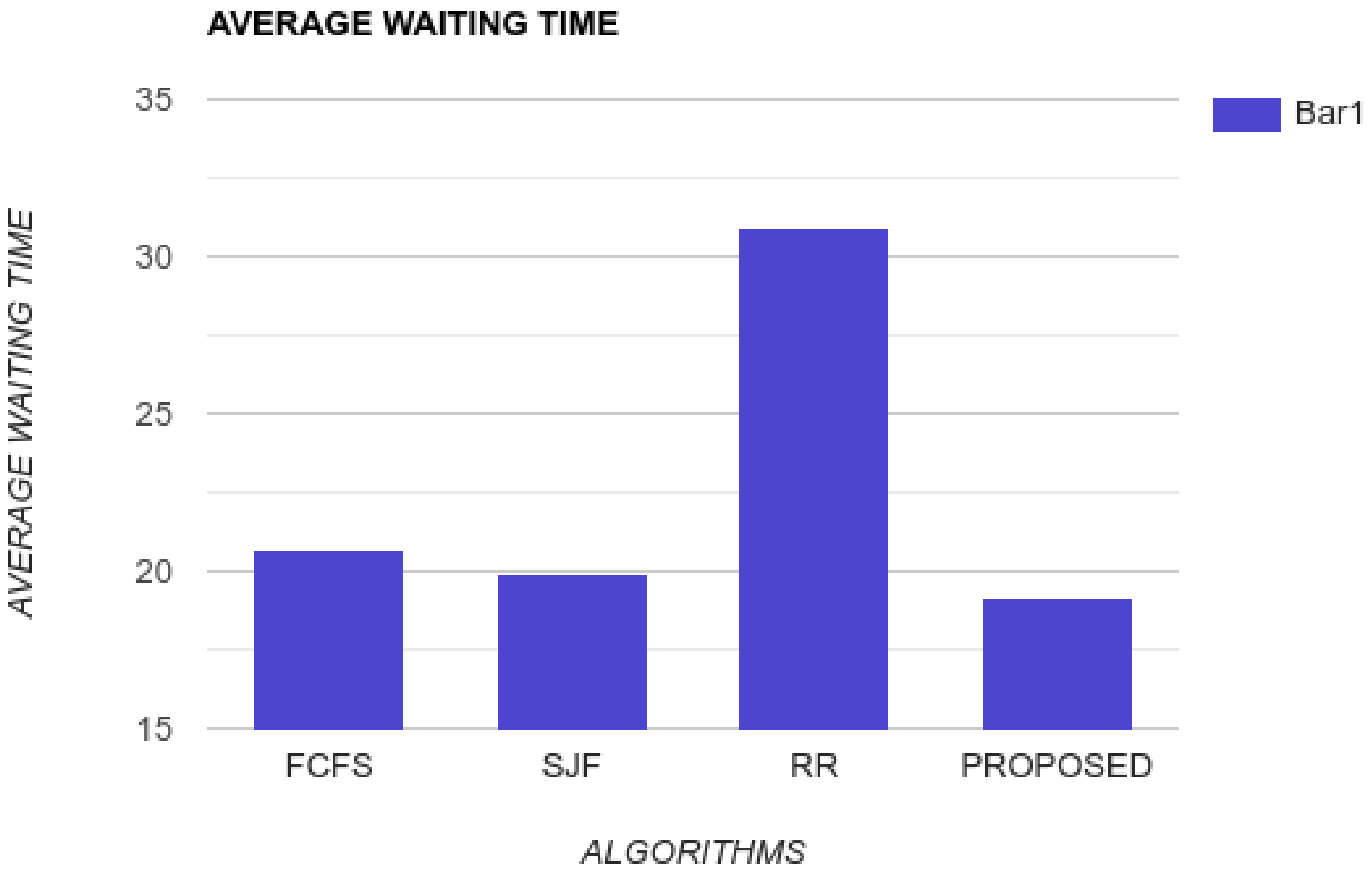


```
Enter your choice from the above table: 4
Enter the no. of processes:10
 burst time for all sequences:2
4
8
3
4
5
6
8
9
11

time quantum is ceilceil((highestbt*median)) = 9
Procoess 0 from 0 to 2
Procoess 1 from 2 to 5
Procoess 2 from 5 to 9
Procoess 3 from 9 to 13
Procoess 4 from 13 to 18
Procoess 5 from 18 to 24
Procoess 6 from 24 to 32
Procoess 7 from 32 to 40
Procoess 8 from 40 to 49
Procoess 9 from 49 to 58

time quantum is ceilceil((highestbt*median)) = 2
Procoess 9 from 58 to 60

time quantum is ceilceil((highestbt*median)) = 0
All the processes has been executed

processes :0 1 2 3 4 5 6 7 8 9
Turn Around time for all processes : 2 5 9 13 18 24 32 40 49 60
Burst time for all processes :2 3 4 4 5 6 8 8 9 11
Waiting time for all processes :0 2 5 9 13 18 24 32 40 49
Avg waiting time is 19.200001

Avg turn around time is 25.200001
```

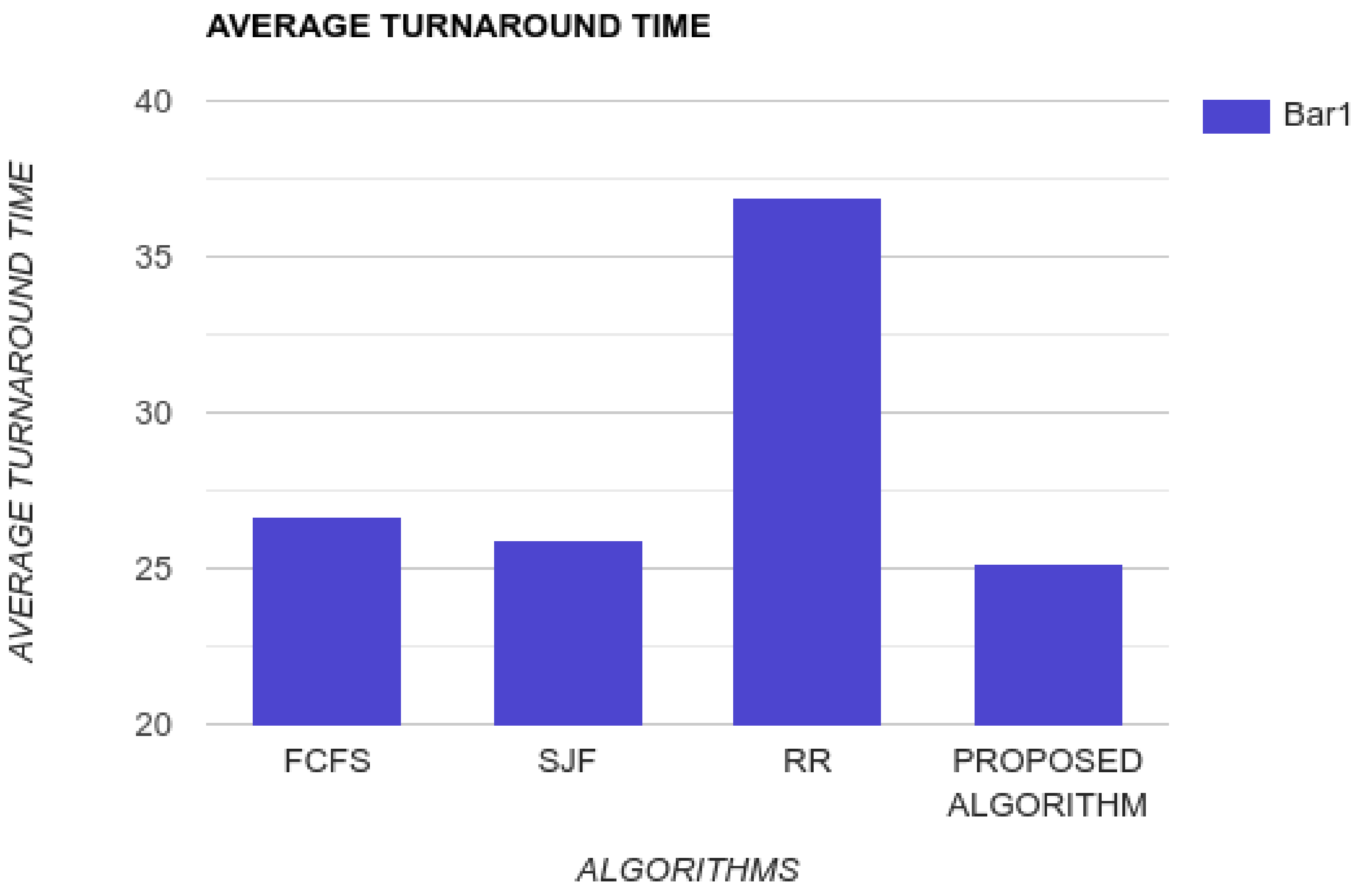
# COMPARISON

| ALGORITHM | WAITING TIME | TURN AROUND TIME |
|---|---|---|
| FCFS | 20.7 | 26.7 |
| SJF | 19.9 | 25.9 |
| RR | 30.9 | 36.9 |
| OUR ALGORITHM | 19.2 | 25.2 |

# WAITING TIME



# TURNAROUND TIME



# Conclusion and Future Work

- The average waiting time and turnaround time also been reduced as compared to Round Robin algorithm which makes the algorithm an efficient one as we know the main goal of CPU scheduling is maximum CPU utilization, minimum waiting time, minimum turnaround time.
- Thus, we can say that if we calculate time quantum based on this algorithm then we can have a best possible time quantum for scheduling to make the CPU scheduling more efficient one.

# CONTD...

- We also observe from the above stated algorithm, that in the newly proposed algorithm the number of context switches has been reduced from the general round robin algorithm.
- Clearly it can be concluded that the proposed algorithm will perform better than the static Round Robin Algorithm in terms of all measures like average waiting time, average turnaround time and number of context switches.
- For the future work, processes at different arrival of time can be considered for the proposed algorithm.

# REFERENCES:-

| AUTHOR | PROBLEM | SOLUTION | LIMITATION |
|---|---|---|---|
| [1]Rakesh Kumar Yadav ,Abhishek K Mishra , Navin Prakash and Himanshu Sharma (2010) “An Improved Round Robin Scheduling Algorithm for CPU Scheduling.” | To determine the most efficient way to service the requests. | allocates the CPU to the processes for one time quantum and if the remaining burst time is less than one time quantum then CPU is again allocated to the currently running process for remaining burst time. | Though this algorithm is superior than RR algorithm there is no significant difference in WT and TAT as compared to other algorithms. This algorithm can be modified in near future. |
| [2]Rami J.Matarneh(2011) “Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Processes” | The performance and efficiency of multitasking operating systems depends on the used CPU scheduling algorithm and as round robin scheduling algorithm is considered most used scheduling algorithms in this | 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. Both the cases will have their own functioning. | This dynamic time quantum based scheduling algorithm can be improved which will involve both SJF and round robin with dynamic time quantum. |

# CONTD….

| AUTHOR | PROBLEM | SOLUTION | LIMITATION |
|---|---|---|---|
| [3]Ajit Singh, Priyanka Goyal, Sahil Batra<br>“An Optimized Round Robin Scheduling Algorithm for CPU Scheduling”<br>Vol. 02, No. 07, 2010 | The main objective of this paper is to develop a new approach for round robin scheduling which help to improve the CPU efficiency in real time and time sharing operating System and to improve all the drawbacks of simple round robin architecture. | After first cycle perform the following steps:<br>a) Double the initial time quantum (2k units).<br>b) Select the shortest process from the waiting queue and assign to CPU.<br>c) select the next shortest process for execution by excluding the already executed one in this phase. | Future work can be based on this algorithm modified and implemented for hard real time system |
| [4]Lalit Kishor and Dinesh Goyal<br>“Time Quantum Based Improved Scheduling Algorithm”” (2013) | To make one scheduling algorithm of our own and compare its waiting time and turnaround time with already existing algorithms. | We propose a Median based Time quantum based scheduling algorithm which is combination of SJF & RR | In future a dynamic time quantum based scheduling algorithm can be designed which will involve both SJF and round robin with dynamic time quantum. |

| AUTHOR | PROBLEM | SOLUTION | LIMITATION |
|---|---|---|---|
| [5]Radhe Shyam, Sunil Kumar Nandal “Improved Mean Round Robin With Shortest Job First Scheduling”. | To make one scheduling algorithm of our own and compare its waiting time and turnaround time with already existing algorithms. | Round Robin being the most popular in time shared operating system, but it may not be suitable for real time operating systems because of high turnaround time, waiting time and large number of context switches | As in the paper they have taken the ideal cases in calculating the turnaround time(TAT) and waiting time(WT). In future they can implement this algorithm in different arrival time of processes |
| [6] Manish Kumar Mishra and Dr. Faizur Rashid” An improved round robin CPU scheduling algorithm with varying time quantum” | To make one scheduling algorithm of our own and compare its waiting time and turnaround time with already existing algorithms. | The improved Round Robin CPU scheduling algorithm with varying time quantum (IRRVQ) combines the features of SJF and RR scheduling algorithms with varying time quantum | Future work can be based on this algorithm modified and implemented for hard real time system |

| AUTHOR | PROBLEM | SOLUTION | LIMITATION |
|---|---|---|---|
| [7]H.S.Behera, R Mohanty, Debashree Nayak (2010)” | To make one scheduling algorithm of our own and compare its waiting time and turnaround time with already existing algorithms. | A time quantum is assigned to the processes and after each cycle it is recalculated taking the remaining burst time in account. | The proposed algorithm performs better than RR scheduling algorithm with respect to average waiting time, turnaround time and context switching. |
| [8]Harshal Bharatkumar Parekh, Sheetal Chaudhari (2016) | To make one scheduling algorithm of our own and compare its waiting time and turnaround time with already existing algorithms. | Initialise each process by setting flag as ‘false’ in job queue and calculate the time quantum required. Necessary conditions are applied to carry out smooth functioning of the processes | The algorithm can be operated within an algorithm of selecting the best scheduling algorithm dynamically i.e. based on the type of usage, different algorithms can be utilized |

| AUTHOR | PROBLEM | SOLUTION | LIMITATION |
|---|---|---|---|
| [9]Prof. Rakesh Mohanty, Prof. H. S. Behera<br>Khusbu Patwari, Manas Ranjan Das, Monisha Dash, Sudhashree | To make one scheduling algorithm of our own and compare its waiting time and turnaround time with already existing algorithms. | Round robin approach has a static time quantum that causes larger waiting and turnaround time that can be improved by using dynamic time quantum. | Results with multiprocessor environment was not measured and all processes were assumed to be independent |
| [10]Ishwari Singh Rajput,<br>Deepa Gupta “A Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems” | To make one scheduling algorithm of our own and compare its waiting time and turnaround time with already existing algorithms. | Allocating CPU to every process in Round Robin fashion, according to given priority, for the given time quantum for one time. Arranging processes in increasing order of remaining CPU burst time and assigning new priorities. | Performance of time-sharing systems can be improved with the proposed algorithm and can also be modified to enhance the performance of real time system. |

# Thank You!