FCFS , SJF , SRTF SCHEDULING ALGORITHM

GROUP MEMBER:
AAKASH SUDAN(2021A1R159)
MANIK SHARMA(2021A1R161)
RAGHAV ANTHAL(2021A1R162)
NAVJYOT(2021A1R163)

SUBMITTED TO:
MR.SAURABH SHARMA

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our operating system lab teacher “Mr.Saurabh Sharma” for their able guidance and support in completing our project.

We came to know about so many new things that we were unaware of.

"Simulation of First Come First Serve(FCFS) Shortest Job First(SJF) and Shortest Remaining Time First (SRTF)"

Abstract - Development of scheduling algorithms is directly related with the development of operating system which brings difficulties in implementation. Any modification on the scheduling algorithm will appear as modification on the operating system kernel code. Processor is an important source of cpu scheduling process, so it becomes very important on accomplishing of the operating system design goals. A delicate problem of the well-functioning of SO is the case when in CPU comes two or more processes which wants to be executed. Scheduling includes a range of mechanisms and policies that SO has to follows in order that all processes take the service. In this Paper we will discuss about two main batches algorithms, such as FCFS and SJF, and i will show a manner how to improve these algorithms in the future work.

Keywords - CPU-Scheduling, Scheduler, FCFS, SJF,SRTF.

1. Introduction

CPU scheduling is the process of deciding which process will own the CPU to use while another process is suspended. The main function of the CPU scheduling is to ensure that whenever the CPU remains idle, the OS has at least selected one of the processes available in the ready-to-use line.

We have three types of schedulers:

(A)Long-term scheduler – This type of scheduler decides which jobs or processes would be admitted to the ready queue. Also this Scheduler dictates what processes are to run on a system.

(B)Mid-term Scheduler - One second type scheduler it's mid-term scheduler who removes process from main memory and moves on secondary memory.

(C)Short-term Scheduler (also known as dispatcher) - Dispatcher module gives control of the CPU to the process selected by the short-term scheduler.

If most operating systems change their status from performance to waiting then there may always be a chance of failure in the system. So in order to minimize this excess, the OS needs to schedule tasks in order to make full use of the CPU and avoid the possibility of deadlock.

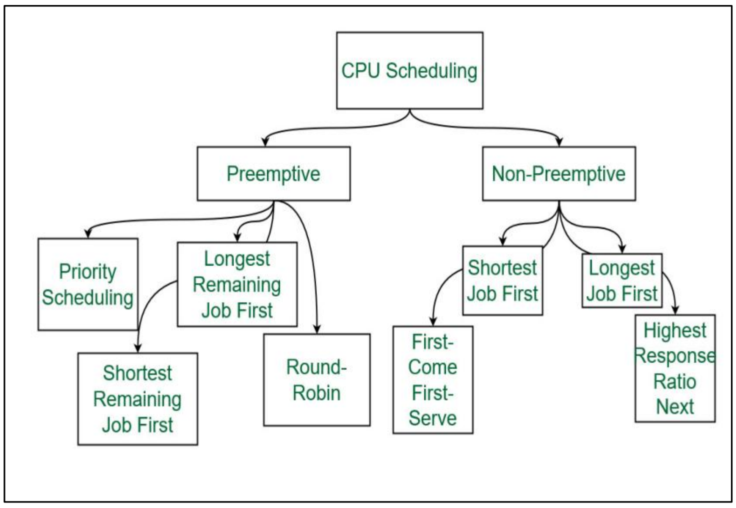
The different terminologies in any CPU Scheduling algorithm

Arrival Time: Time at which the process arrives in the ready queue.
Completion Time: Time at which process completes its execution.
Burst Time: Time required by a process for CPU execution.
Turn Around Time: Time Difference between completion time and arrival time.
Waiting Time(W.T): Time Difference between turn around time and burst time.

2. Types of CPU Scheduling Algorithms

There are mainly two types of scheduling methods:

1. **Preemptive Scheduling**: Preemptive scheduling is used when a process switches from running state to ready state or from the waiting state to the ready state.
2. **Non-Preemptive Scheduling**: Non-Preemptive scheduling is used when a process terminates , or when a process switches from running state to waiting state.



Below will give an overview of algorithms for both construction and operation.

2.1 **First Come First Serve:**

FCFS considered to be the simplest of all operating system scheduling algorithms. First come first serve scheduling algorithm states that the process that requests the CPU first is allocated the CPU first and is implemented by using FIFO queue.

Characteristics of FCFS:

- 1.FCFS supports non-preemptive and preemptive CPU scheduling algorithms.
- 2.Tasks are always executed on a First-come, First-serve concept.
- 3.FCFS is easy to implement and use.
- 4.This algorithm is not much efficient in performance, and the wait time is quite high.

Advantages of FCFS:

- Easy to implement
- First come, first serve method.

Disadvantages of FCFS:

- 1.FCFS suffers from **Convoy effect**.
- 2.The average waiting time is much higher than the other algorithms.
- 3.FCFS is very simple and easy to implement and hence not much efficient.

2.2 Shortest Job First(SJF):

Shortest job first (SJF) is a scheduling process that selects the waiting process with the smallest execution time to execute next. This scheduling method may or may not be preemptive. Significantly reduces the average waiting time for other processes waiting to be executed. The full form of SJF is Shortest Job First.

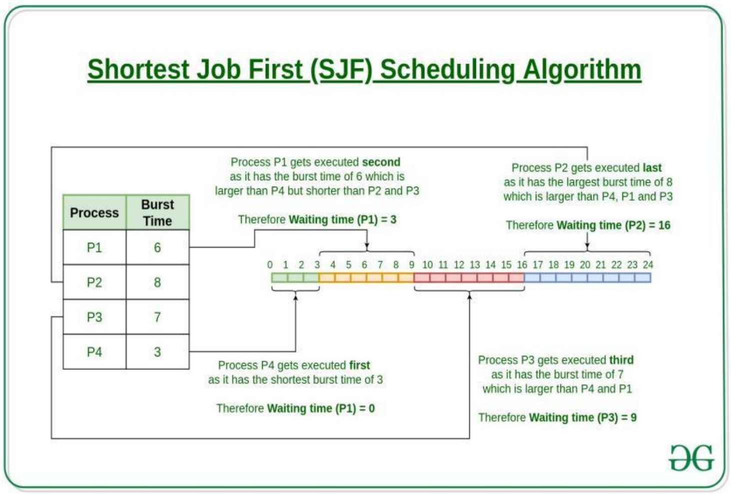
Characteristics of SJF:

- 1.Shortest Job first has the advantage of having a minimum average waiting time among all operating system scheduling algorithms.
- 2.It is associated with each task as a unit of time to complete.
- 3.It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of ageing.

Advantages of Shortest Job first:

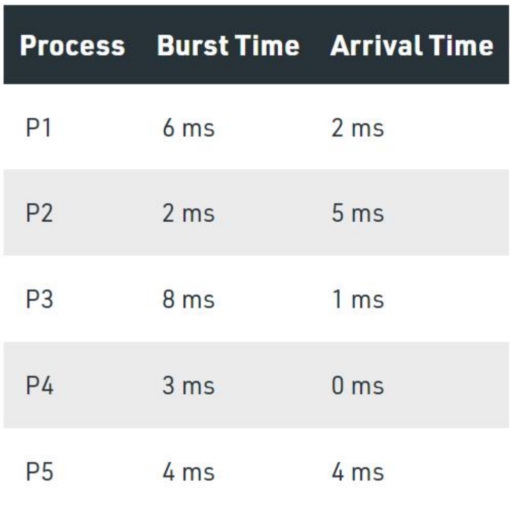
1.As SJF reduces the average waiting time thus, it is better than the first come first serve scheduling algorithm.
2.SJF is generally used for long term scheduling

Disadvantages of SJF:
1.One of the demerit SJF has is starvation.
2.Many times it becomes complicated to predict the length of the upcoming CPU request


Shortest Job First (SJF) Scheduling Algorithm
Process
Burst Time
P1
6
P2
8
P3
7
P4
3
Process P1 gets executed second as it has the burst time of 6 which is larger than P4 but shorter than P2 and P3
Therefore Waiting time (P1) = 3
Process P2 gets executed last as it has the largest burst time of 8 which is larger than P4, P1 and P3
Therefore Waiting time (P2) = 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Process P4 gets executed first as it has the shortest burst time of 3
Therefore Waiting time (P1) = 0
Process P3 gets executed third as it has the burst time of 7 which is larger than P4 and P1
Therefore Waiting time (P3) = 9

Examples to show working of Non-Preemptive Shortest Job First CPU Scheduling Algorithm:

Question :Consider the following table of arrival time and burst time for five processes **P1, P2, P3, P4** and **P5**.



Process	Burst Time	Arrival Time
P1	6 ms	2 ms
P2	2 ms	5 ms
P3	8 ms	1 ms
P4	3 ms	0 ms
P5	4 ms	4 ms

The Shortest Job First CPU Scheduling Algorithm will work on the basis of steps as mentioned below:

At time = 0,

❖ Process P4 arrives and starts executing.

At time= 1,

- Process P3 arrives.
- But, as P4 still needs 2 execution units to complete.
- Thus, P3 will wait till P4 gets executed.

At time =2,

- Process P1 arrives and is added to the waiting table
- P4 will continue its execution.

At time = 3,

❖ Process P4 will finish its execution.
❖ Then, the burst time of P3 and P1 is compared.
❖ Process P1 is executed because its burst time is less as compared to P3.

At time = 4,

❖ Process P5 arrives and is added to the waiting Table.
❖ P1 will continue execution.

At time = 5,

❖ Process P2 arrives and is added to the waiting Table.
❖ P1 will continue execution.

At time = 6,

❖ Process P1 will finish its execution.
❖ The burst time of P3, P5, and P2 is compared.
❖ Process P2 is executed

At time=9,

- ❖ Process P2 is executing and P3 and P5 are in the waiting Table.

At time = 11,

- ❖ The execution of Process P2 will be done.
- ❖ The burst time of P3 and P5 is compared.
- ❖ Process P5 is executed because its burst time is lower than P3.

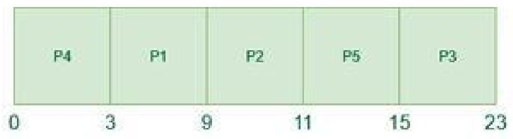
At time = 15,

- ❖ Process P5 will finish its execution.

At time = 23,

- ❖ Process P3 will finish its execution.

Gantt chart for above execution:



Now, let's calculate the average waiting time for above example:

P4 = 0 – 0 = 0

P1 = 3 – 2 = 1

P2 = 9 – 5 = 4

P5 = 11 – 4 = 7

P3 = 15 – 1 = 14

Average Waiting Time = 0 + 1 + 4 + 7 + 14/5 = 26/5 = 5.2

2.3 Shortest Remaining Time First:

Shortest remaining time first is the preemptive version of the Shortest job first which we have discussed earlier where the processor is allocated to the job closest to completion. In SRTF the process with the smallest amount of time remaining until completion is selected to execute.

Characteristics of Shortest remaining time first:

- ❖ SRTF algorithm makes the processing of the jobs faster than SJF algorithm, given it's overhead charges are not counted.
- ❖ The context switch is done a lot more times in SRTF than in SJF and consumes the CPU's valuable time for processing. This adds up to its processing time and diminishes its advantage of fast processing.

Advantages of SRTF:

- ❖ In SRTF the short processes are handled very fast.
- ❖ The system also requires very little overhead since it only makes a decision when a process completes or a new process is added.

Disadvantages of SRTF:

- ❖ Like the shortest job first, it also has the potential for process starvation.
- ❖ Long processes may be held off indefinitely if short processes are continually added.
- ❖ To learn about how to implement this CPU scheduling algorithm, please refer to our detailed article on the Shortest remaining time first.

<u>Examples to show working of Preemptive Shortest Job First CPU Scheduling Algorithm:</u>

Example-1: Consider the following table of arrival time and burst time for five processes **P1, P2, P3, P4** and **P5**.

Process	Burst Time	Arrival Time
P1	6 ms	2 ms
P2	2 ms	5 ms
P3	8 ms	1 ms
P4	3 ms	0 ms
P5	4 ms	4 ms

The Shortest Job First CPU Scheduling Algorithm will work on the basis of steps as mentioned below:

At time = 0,

- ❖ Process P4 arrives and starts executing

At time= 1,

- ❖ Process P3 arrives.
- ❖ But, as P4 has a shorter burst time. It will continue execution.
- ❖ Thus, P3 will wait till P4 gets executed.

At time =2,

- ❖ Process P1 arrives with burst time = 6
- ❖ As the burst time of P1 is more than that of P4
- ❖ Thus, P4 will continue its execution.

At time = 3,

- ❖ Process P4 will finish its execution.
- ❖ Then, the burst time of P3 and P1 is compared.
- ❖ Process P1 is executed because its burst time is less as compared to P3.

At time = 4,

- Process P5 arrives.
- Then the burst time of P3, P5, and P1 is compared.
- Process P5 gets executed first among them because its burst time is lowest, and process P1 is **preempted**.

At time = 5,

- ❖ Process P2 arrives.
- ❖ The burst time of all processes are compared,
- ❖ Process P2 gets executed as its burst time is lowest among all.
- ❖ Process P5 is preempted.

At time = 6,

- ❖ Process P2 will keep executing.
- ❖ It will execute till time = 8 as the burst time of P2 is 2ms

At time=7,

- ❖ The process P2 finishes its execution.
- ❖ Then again the burst time of all remaining processes is compared.
- ❖ The Process P5 gets executed because its burst time is lesser than the others.
- ❖ **At time = 10,**
- ❖ The process P5 will finish its execution.
- ❖ Then the burst time of the remaining processes P1 and P3 is compared.
- ❖ Thus, process P1 is executed as its burst time is less than P3

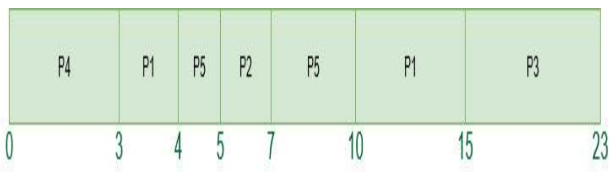
At time = 15,

- ❖ The process P1 finishes its execution and P3 is the only process left.
- ❖ P3 will start executing.

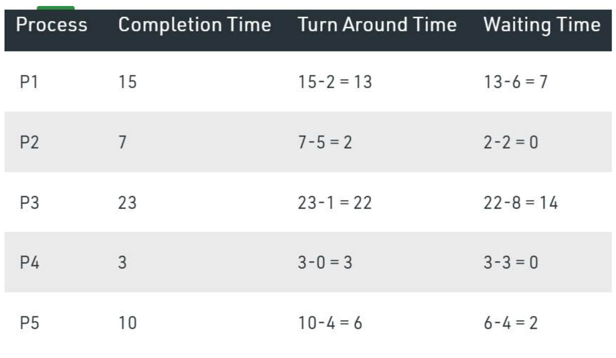
At time = 23,

- ❖ Process P3 will finish its execution.

Gantt chart for above execution:



Average turnaround time and waiting time:



Process	Completion Time	Turn Around Time	Waiting Time
P1	15	15-2 = 13	13-6 = 7
P2	7	7-5 = 2	2-2 = 0
P3	23	23-1 = 22	22-8 = 14
P4	3	3-0 = 3	3-3 = 0
P5	10	10-4 = 6	6-4 = 2

Now,

- **Average Turn around time** = (13 + 2 + 22 + 3 + 6)/5 = 9.2
- **Average waiting time** = (7 + 0 + 14 + 0 + 2)/5 = 23/5 = 4.6

Conclusions

After execution of simulation codes of FCFS and SJF I come to the conclusion that SJF algorithm is more competitive than FCFS If we talk about performance because It has minimal average waiting that is very important.SJF main flaw is : a very long process to be not executed at the time required, thing that does not happen to FCFS algorithm.In SRTF long processes may be held off indefinitely if short processes are continually added.

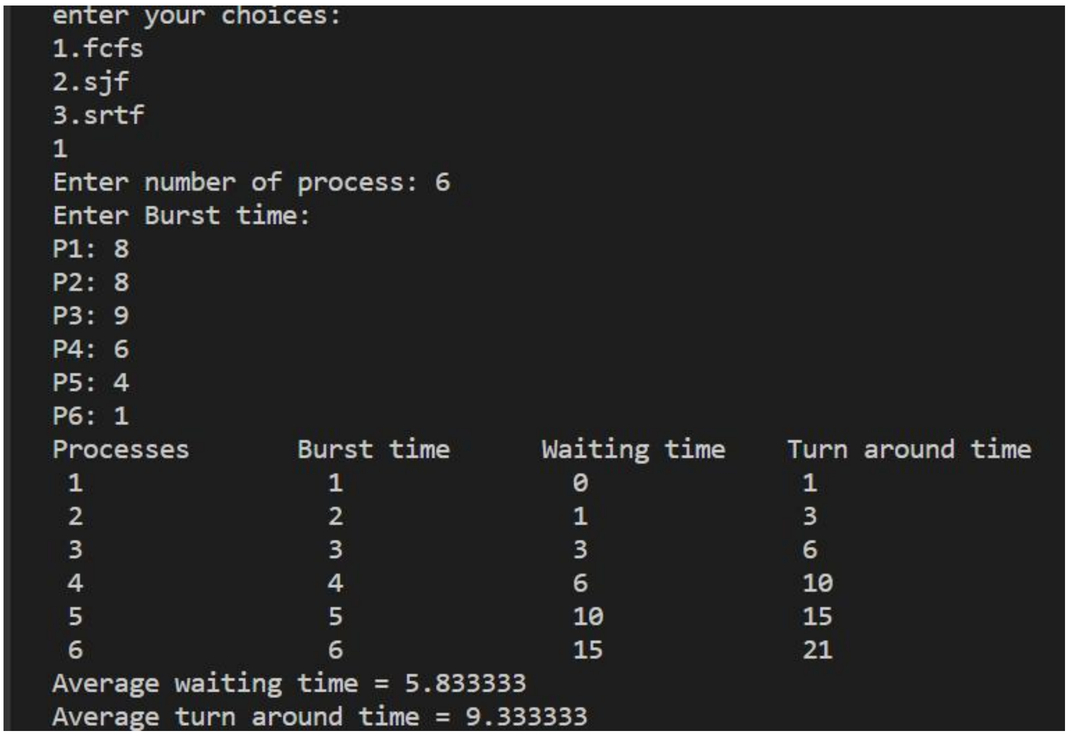
CODE

```
#include<stdio.h>
int choice;
//fcfs code{

// Function to find the waiting time for all processes
void findWaitingTime(int processes[], int n,
                        int bt[], int wt[])
{
    // waiting time for first process is 0
    wt[0] = 0;
    // calculating waiting time
    for (int i = 1; i < n ; i++ )
        wt[i] = bt[i-1] + wt[i-1] ;
}
// Function to calculate turn around time
void findTurnAroundTime( int processes[], int n,
                int bt[], int wt[], int tat[])
{
    // calculating turnaround time by adding
    // bt[i] + wt[i]
    for (int i = 0; i < n ; i++)
        tat[i] = bt[i] + wt[i];
}
//Function to calculate average time
void findavgTime( int processes[], int n, int bt[])
{
    int wt[n], tat[n], total_wt = 0, total_tat = 0;

    //Function to find waiting time of all processes
    findWaitingTime(processes, n, bt, wt);
    //Function to find turn around time for all processes
    findTurnAroundTime(processes, n, bt, wt, tat);
    //Display processes along with all details
    printf("Processes\tBurst time\tWaiting time\tTurn around time\n");
    // Calculate total waiting time and total turn around time
    for (int i=0; i<n; i++)
    {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        printf(" %d\t\t ",(i+1));
        printf(" %d\t\t ", bt[i] );
        printf(" %d\t\t",wt[i] );
        printf(" %d\t\t \n",tat[i] );
    }
float s=(float)total_wt / (float)n;
    float t=(float)total_tat / (float)n;
    printf("Average waiting time = %f",s);
    printf("\n");
    printf("Average turn around time = %f ",t);
}
int fcfs()
{
    int n; // no. of process.
    //process id's
    int processes[10];
    //Burst time of all processes
    int burst_time[10];
    printf("Enter number of process: ");
    scanf("%d", &n);
    printf("Enter Burst time:\n");
    // User Input Burst Time and alloting Process Id.
    for (int i = 0; i < n; i++) {
        printf("P%d: ", i + 1);
        scanf("%d", &burst_time[i]);
        burst_time[i] = i + 1;
        processes[i]=i+1;
    }

    findavgTime(processes, n, burst_time);
    return 0;
}
// }

int sjf()
{
    int A[100][4]; // Matrix for storing Process Id, Burst
                // Time, Average Waiting Time & Average
                // Turn Around Time.
    int i, j, n, total = 0, index, temp;
    float avg_wt, avg_tat;
    printf("Enter number of process: ");
    scanf("%d", &n);
    printf("Enter Burst Time:\n");
    // User Input Burst Time and alloting Process Id.
    for (i = 0; i < n; i++) {
        printf("P%d: ", i + 1);
        scanf("%d", &A[i][1]);
        A[i][0] = i + 1;
    }
    // Sorting process according to their Burst Time.
    for (i = 0; i < n; i++) {
        index = i;
        for (j = i + 1; j < n; j++)
            if (A[j][1] < A[index][1])
                index = j;
        temp = A[i][1];
        A[i][1] = A[index][1];
        A[index][1] = temp;

        temp = A[i][0];
        A[i][0] = A[index][0];
        A[index][0] = temp;
    }
A[0][2] = 0;
    // Calculation of Waiting Times
    for (i = 1; i < n; i++) {
        A[i][2] = 0;
        for (j = 0; j < i; j++)
            A[i][2] += A[j][1];
        total += A[i][2];
    }
```

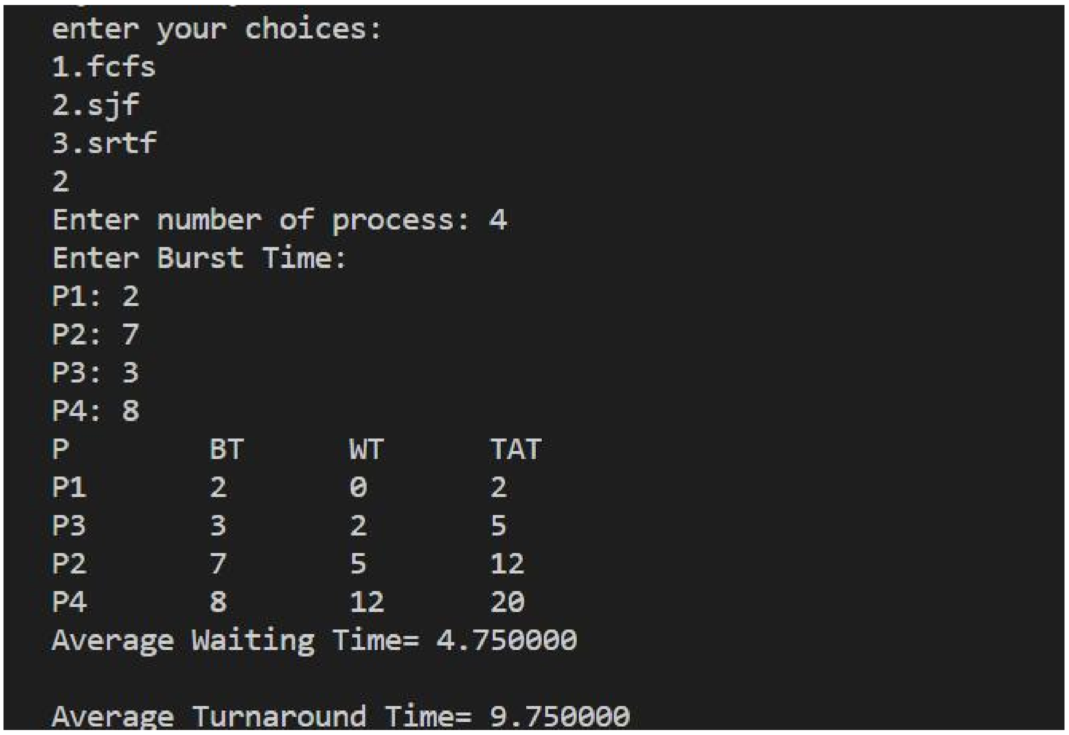
```
avg_wt = (float)total / n;
    total = 0;
    printf("P    BT  WT  TAT\n");
    // Calculation of Turn Around Time and printing the data.
    for (i = 0; i < n; i++) {
        A[i][3] = A[i][1] + A[i][2];
        total += A[i][3];
        printf("P%d  %d  %d  %d\n", A[i][0],
            A[i][1], A[i][2], A[i][3]);
    }
    avg_tat = (float)total / n;
    printf("Average Waiting Time= %f\n", avg_wt);
    printf("\nAverage Turnaround Time= %f", avg_tat);
}


int srtf()
{
      int arrival_time[10], burst_time[10], temp[10];
      int i, smallest, count = 0, time, limit;
      double wait_time = 0, turnaround_time = 0, end;
      float average_waiting_time, average_turnaround_time;
      printf("Enter the Total Number of Processes:");
      scanf("%d", &limit);
     // printf("Enter Details of %d Processes", limit);
      for(i = 0; i < limit; i++)
      {
            printf("Enter Arrival Time of P%d:",i+1);
            scanf("%d", &arrival_time[i]);
            printf("Enter Burst Time:of P%d:",i+1);
            scanf("%d", &burst_time[i]);
            temp[i] = burst_time[i];
      }
      burst_time[9] = 9999;
      for(time = 0; count != limit; time++)
      {
            smallest = 9;
            for(i = 0; i < limit; i++)
            {
                  if(arrival_time[i] <= time && burst_time[i] < burst_time[smallest] && burst_time[i] >
0)
                  {
                        smallest = i;
                  }
            }
            burst_time[smallest]--;
            if(burst_time[smallest] == 0)
            {
                  count++;
                  end = time + 1;
                  wait_time = wait_time + end - arrival_time[smallest] - temp[smallest];
                  turnaround_time = turnaround_time + end - arrival_time[smallest];
            }
      }
      average_waiting_time = wait_time / limit;
      average_turnaround_time = turnaround_time / limit;
      printf("Average Waiting Time:%f\n", average_waiting_time);

      printf("Average Turnaround Time:%f", average_turnaround_time);
      return 1;
}
```

```
int main()
{
    printf("enter your choices:\n1.fcfs\n2.sjf\n3.srtf\n");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1:
        {
            fcfs();
            break;
        }
        case 2:
        {
            sjf();
            break;
        }
        case 3:
        {
            srtf();
            break;
        }
        default:
        printf("invalid choice\n");
        break;

}

return 0;
}
```

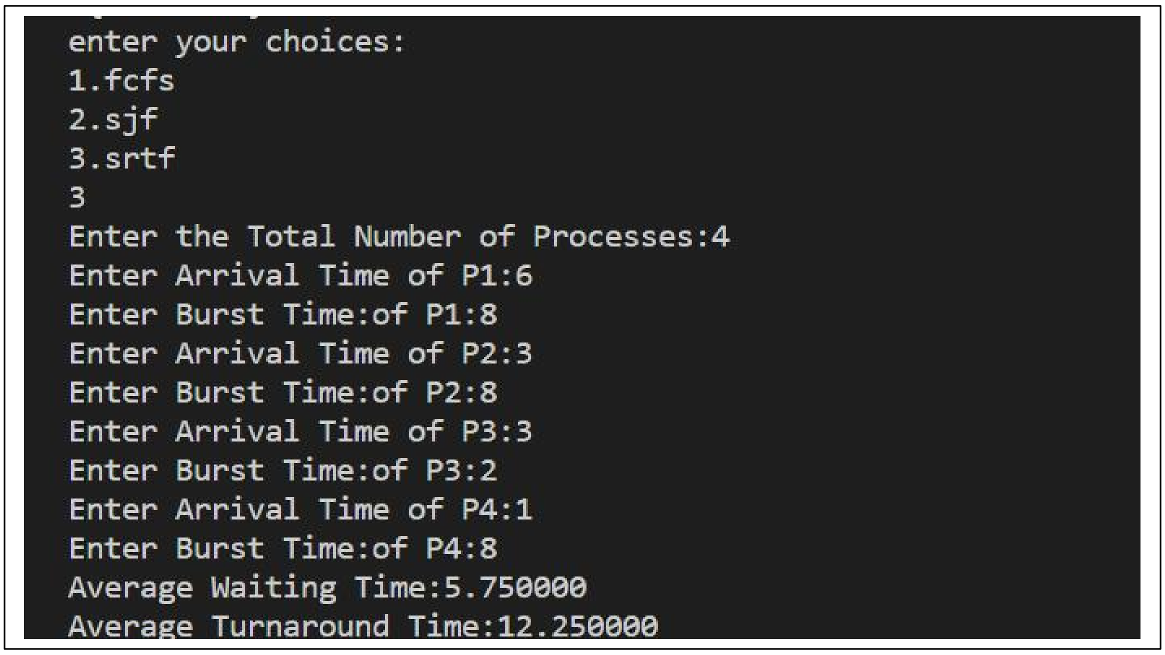
OUTPUT:



```
enter your choices:
1.fcfs
2.sjf
3.srtf
1
Enter number of process: 6
Enter Burst time:
P1: 8
P2: 8
P3: 9
P4: 6
P5: 4
P6: 1
Processes       Burst time      Waiting time    Turn around time
 1               1               0               1
 2               2               1               3
 3               3               3               6
 4               4               6               10
 5               5               10              15
 6               6               15              21
Average waiting time = 5.833333
Average turn around time = 9.333333
```



```
enter your choices:
1.fcfs
2.sjf
3.srtf
2
Enter number of process: 4
Enter Burst Time:
P1: 2
P2: 7
P3: 3
P4: 8
P        BT      WT      TAT
P1       2       0       2
P3       3       2       5
P2       7       5       12
P4       8       12      20
Average Waiting Time= 4.750000

Average Turnaround Time= 9.750000
```



```
enter your choices:
1.fcfs
2.sjf
3.srtf
3
Enter the Total Number of Processes:4
Enter Arrival Time of P1:6
Enter Burst Time:of P1:8
Enter Arrival Time of P2:3
Enter Burst Time:of P2:8
Enter Arrival Time of P3:3
Enter Burst Time:of P3:2
Enter Arrival Time of P4:1
Enter Burst Time:of P4:8
Average Waiting Time:5.750000
Average Turnaround Time:12.250000
```

Reference:

We take reference from various articles as well as various websites

1) https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/

2) http://ijcsn.org/IJCSN-2014/3-6/Simulation-of-First-Come-First-Served-_FCFS_-and-Shortest-Job-First-_SJF_-Algorithms.pdf

Links of GitHub accounts:

1) https://github.com/AakashSudan/operating_system_project

2) https://github.com/RAGHAV-223/project-os

3) https://github.com/manik141/project-os

4) https://github.com/Navjyot201/project-os