**Student Name :** Ritik Singh

**Student ID :** 11804911

**Email Address :** ritikbandwal@gmail.com

**GitHub Link :** https://github.com/RitikBandwal/OS\_Scheduling\_Prog\_ShortestExeTime1st.git

**Code :**

#include<iostream>

using namespace std;

main()

{

int n,T=3,l=0,CT=3,j=0;

int AT[10],BT[10],RT[10],TAT[10],WT[10];

int lbt=INT\_MAX;

float TTAT=0,TWT=0;

bool counter=false;

cout<<"\n\n\n\t\t\t\t\tEnter the number of processes : ";

cin>>n;

cout<<"\n";

for(int i=0;i<n;i++)

{

cout<<"\t\tEnter the arrival time and burst time of process "<<i+1<<" respectively.\n";

cout<<"\t\t\tArrival Time : ";

cin>>AT[i];

cout<<"\t\t\tBurst Time : ";

cin>>BT[i];

RT[i]=BT[i];

if((AT[i]<3)||(BT[i]>10))

{

cout<<"\n\t\tInvalid input...Arrival Time should be more than 3 Time Units AND"

<<" Burst Time should be less than 10 Time Units.\n";

i--;

}

}

system("cls");

cout<<"\n\n\n\t\tComputing Average Waiting Time And Turn Around Time...";

cout<<"\n\n\n\tQueue of order of execution of all processes(1 Time Unit Each) : \n\t\t|";

while(j<n)

{ a:

for(int i=0;i<n;i++)

{

if((AT[i]<=T)&&(RT[i]<lbt)&&(RT[i]>0))

{

lbt=RT[i];

l=i;

counter=true;

}

}

if(counter==false)

{

T++;

goto a;

}

cout<<"P"<<l+1<<"|";

RT[l]--;

lbt=RT[l];

if (lbt == 0)

{

lbt = INT\_MAX;

}

if(RT[l]==0)

{

j++;

counter=false;

CT=T+1;

WT[l]=CT-BT[l]-AT[l];

if (WT[l] < 0)

{

WT[l] = 0;

}

}

T++;

}

cout<<"\n\n\t\tProcess\t\tAT\t\tBT\t\tWT\t\tTAT\n";

cout<<"\t\t-------------------------------------------------------------------\n";

for(int i=0;i<n;i++)

{

TAT[i]=BT[i]+WT[i];

cout<<"\t\t"<<i+1<<"\t\t"<<AT[i]<<"\t\t"<<BT[i]<<"\t\t"<<WT[i]<<"\t\t"<<TAT[i]<<"\n";

TTAT=TTAT+TAT[i];

TWT =TWT +WT[i] ;

}

cout<<"\n\t\tAverage Waiting Time is : "<<TWT/n;;

cout<<"\n\t\tAverage Turn Around Time is : "<<TTAT/n<<"\n";;

}

1.Explain the problem in terms of operating system concept? (Max 200 word)

**Description:**

This problem is the **preemptive version** of **Shortest Job First scheduling**. In Shortest Execution Time First Scheduling, the execution of a process can be stopped and execution of another process can be started after a certain amount of time. At the arrival of each process, the short-term scheduler schedules that process which have less remaining burst time among the list of processes in the ready queue and the currently running process. If each and every of the given processes are accessible in the **ready queue**, pre-emption will be not be done and after that the algorithm will run as **Shortest Job First Scheduling**. The state of the currently running process is stored in the **Process Control Block**when that process is about to removed from the execution period and then next process is scheduled for execution. This Process Control Block is accessed on the **next execution** of that process if required. In the above problem the CPU is idle for the first 3 Time Unit and will not accept any process having arrival time prior this. The processes having time unit more than 10 are not acceptable.

2.Write the algorithm for proposed solution of the assigned problem.

**Algorithm:**

1- Traverse each and every process.

a) Find that process having the least burst time after each time unit.

b) Gradually decrement it’s burst time by 1.

c) Check whether the process with least burst time has reached 0.

d) If it has reached 0 then

: Increase the counter for process completion and execute next process.

Else

: Keep that process running.

e) Calculate the completion time of currently running process by Completion Time = Current Time + 1;

f) Calculate the Waiting Time for each completed process.

Waiting Time = Completion time - Arrival time – Burst time

And then add all them up to find the Average Waiting Time.

2- Find the Turnaround time.

Waiting Time + Burst Time

And then add all them up to find the Average Turn Around Time.

3.Calculate complexity of implemented algorithm. (Student must specify complexity of each line of code along with overall complexity)

**Description (purpose of use):**

For first ‘for’ loop which is for taking the inputs time complexity is O(n).

n is the number of processes.

For finding the Waiting time we need to loop inside a loop so that each of the process is traversed accurately till it’s completion. So, it has time complexity O(n^2).

Then for finding the Average Waiting and Average Turn Around Time we need a single ‘for’ loop. So, it has time complexity O(n).

Therefore, the complexity of the given algorithm is O(n)+O(n^2)+O(n) i.e.,

O(n^2).

4.Explain all the constraints given in the problem. Attach the code snippet of the implemented constraint.

**Code snippet:**

Constraints are:

Arrival time should be greater than 3 time-units and burst time should be less than 10 time-units.

for(int i=0;i<n;i++)

{

if((AT[i]<3)||(BT[i]>10))

{

cout<<"\n\t\tInvalid input...Arrival Time should be more than 3 Time Units AND"

<<" Burst Time should be less than 10 Time Units.\n";

i--;

}

}

5. Explain the boundary conditions of the implemented code.

**Description:**

The boundary conditions for the implemented code are:

We will be finding that process which have the least arrival time and burst time at a given point of time. If the arrival time of a process is less than or equal to the time T at a given time and the burst time of a process is the least but not zero than all the processes at that given time. Then we will be choosing that process and decrementing it’s burst time by 1 time-unit.

Code Snippet: if((AT[i]<=T)&&(RT[i]<lbt)&&(RT[i]>0))

As that process burst time reaches zero OR if we get a process having less burst time at the given point of time then we will be incrementing our counter and then switching for the next process.

6.Explain all the test cases applied on the solution of assigned problem**.**

**Description:**

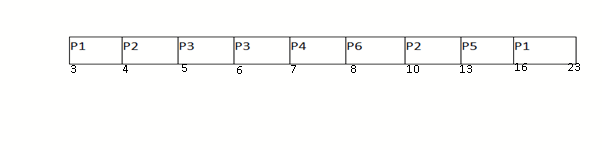
**TEST CASE 1:**

|  |  |  |
| --- | --- | --- |
| Process ID | Arrival Time | Burst Time |
| 1 | **3** | **8** |
| 2 | **4** | **4** |
| 3 | **5** | **2** |
| 4 | **6** | **1** |
| 5 | **7** | **3** |
| 6 | **8** | **2** |

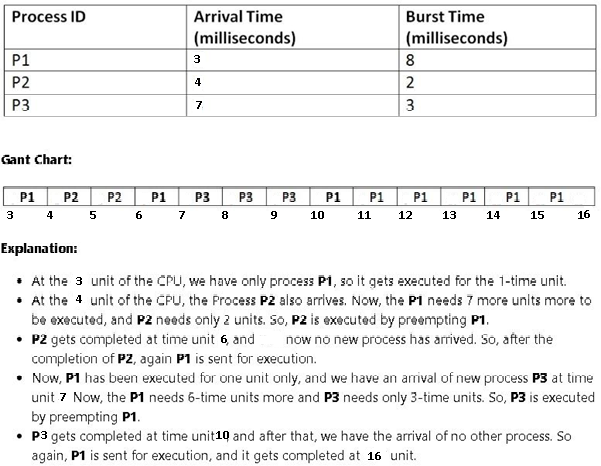
1. Since, at time 3, the only available process is P1 with CPU burst time 8. This is the only available process in the list therefore it is scheduled.
2. The next process arrives at time unit 4. Since the algorithm we are using is SRTF which is a pre-emptive one, the current execution is stopped and the scheduler checks for the process with the least burst time.  
   Till now, there are two processes available in the ready queue. The OS has executed P1 for one unit of time till now; the remaining burst time of P1 is 7 units. The burst time of Process P2 is 4 units. Hence Process P2 is scheduled on the CPU according to the algorithm.
3. The next process P3 arrives at time unit 5. At this time, the execution of process P3 is stopped and the process with the least remaining burst time is searched. Since the process P3 has 2 unit of burst time hence it will be given priority over others.
4. The Next Process P4 arrives at time unit 6. At this arrival, the scheduler will stop the execution of P4 and check which process is having least burst time among the available processes (P1, P2, P3 and P4). P1 and P2 are having the remaining burst time 7 units and 3 units respectively.

P3 and P4 are having the remaining burst time 1 unit each. Since, both are equal hence the scheduling will be done according to their arrival time. P3 arrives earlier than P4 and therefore it will be scheduled again.

1. The Next Process P5 arrives at time unit 7. Till this time, the Process P3 has completed its execution and it is no more in the list. The scheduler will compare the remaining burst time of all the available processes. Since the burst time of process P4 is 1 which is least among all hence this will be scheduled.
2. The Next Process P6 arrives at time unit 8, till this time, the Process P4 has completed its execution. We have 4 available processes till now, that are P1 (7), P2 (3), P5 (3) and P6 (2). The Burst time of P6 is the least among all hence P6 is scheduled. Since, now, all the processes are available hence the algorithm will now work same as SJF. P6 will be executed till its completion and then the process with the least remaining time will be scheduled.



**TEST CASE 2:**



**GitHub Link :**

https://github.com/RitikBandwal/OS\_Scheduling\_Prog\_ShortestExeTime1st.git