Report On

SIMULATION BASED PROCESS SCHEDULER

For Course

Operating Systems

(CSE 316)

BY:	
Registration No	11804431
Name of Student	Vaibhav Patel
Roll No	24
Section	K18SP
Email ID:	Vniranjan26@gmail.com
GitHub Link:	github.com/vniranjan26/Operating-System-CSE316

Submitted To: **Ms. Isha**

Faculty of: **Lovely Professional University**

Jalandhar, Punjab, India

Contents

Problem Description

Problem states that consider n number of processes and design a process management scheduler, which assign a processor to each process on the basics given conditions. Problem say's take arrival time of the process and the required burst time of that process as input from the user at run time and calculate the completion time, turnaround time and waiting time of all the process and also calculates the average turnaround time and waiting time of the processes. While calculating the completion time of the process, we have to follow some set of rules.

- The scheduler schedules the processes by interrupting the processor after every 3 units of time and does consider the completion of the process in this iteration.(First iteration)
- The schedulers then checks for the number of processes waiting for the processor and allots the processor to the process but interrupting the processor after every 6 units of time and considers the completion of the process in this iteration. (Second iteration)
- The scheduler after the second iteration checks for the number of processes waiting for the processor and now provides the processor to the process with the least time requirement to go in the terminated state.

Basically we have to implement 2 algorithms for design such type scheduler,

1. Round-Robin
2. Shortest job first

Round-Robin: Round-Robin algorithm having a fixed time quantum to execute each process in this type of scheduling approach every process will execute at a certain time and no process will go to starvation.

Shortest job First: It is a non-primitive scheduling algorithm where it works on the basis of arrival time and burst time it assigned CPU to that very process which required less time to fully execute and then move to next process in this process large process may be go at starvation.

Test Cases

Test Case: 1

Sample Input:

Process	Arrival time	Burst time
P_1	0	18
P_2	2	23
P_3	4	13
P_4	13	10

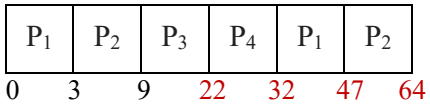
Expected Output:

Process	Arrival time	Burst time	Completion time	Turnaround time	Waiting time
P_1	0	18	47	47	29
P_2	2	23	64	62	39
P_3	4	13	22	18	5
P_4	13	10	32	19	9

Average Turnaround time: 36.5
Average Waiting time: 20.5

Explanation: The time quantum of 1 iteration is 3 unit of time, the time quantum of 2 iteration is 6 unit of time and after 2 iteration provide CPU of that process which required less time to go terminated state.

Gantt chart of Process Completion:



Test Case: 2

Sample Input:

Process	Arrival time	Burst time
P_1	0	20
P_2	3	36
P_3	13	19

P_4	20	42
P_5	36	69
P_6	41	35

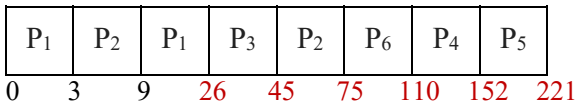
Expected Output:

Process	Arrival time	Burst time	Completion time	Turnaround time	Waiting time
P_1	0	20	26	26	6
P_2	3	36	75	72	36
P_3	13	19	45	32	13
P_4	20	42	152	132	90
P_5	36	69	221	185	116
P_6	41	35	110	69	34

Average Turnaround time: 86
Average Waiting time: 49.1667

Explanation: The time quantum of 1 iteration is 3 unit of time, the time quantum of 2 iteration is 6 unit of time and after 2 iteration provide CPU of that process which required less time to go terminated state.

Gantt chart of Process Completion:



Test Case: 3

Sample Input:

Process	Arrival time	Burst time
P_1	0	6
P_2	3	16
P_3	5	2
P_4	9	8
P_5	10	7

P_6	22	5
P_7	14	13
P_8	26	4
P_9	17	19
P_{10}	28	3

Expected Output:

Process	Arrival time	Burst time	Completion time	Turnaround time	Waiting time
P_1	0	6	14	14	8
P_2	3	16	51	48	32
P_3	5	2	11	6	4
P_4	9	8	29	20	12
P_5	10	7	21	11	4
P_6	22	5	41	19	14
P_7	14	13	64	50	37
P_8	26	4	36	10	6
P_9	17	19	83	66	47
P_{10}	28	3	32	4	1

Average Turnaround time: 24.8
Average Waiting time: 16.5

Explanation: The time quantum of 1 iteration is 3 unit of time, the time quantum of 2 iteration is 6 unit of time and after 2 iteration provide CPU of that process which required less time to go terminated state.

Gantt chart of Process Completion:

P_1	P_2	P_3	P_1	P_5	P_4	P_{10}	P_8	P_6	P_2	P_7	P_9

0 3 9 11 14 21 29 32 36 41 51 62 83

Algorithm

Step -1: Declare array arival_time[ ], busrt_time[ ], completion_time[ ], waiting_time[ ], trunaround_time[ ] and **int** variable size.

Step -2: Take input number of process in size.

Step -3: Repeat for int I = 0,1,2…. Size-1
. Take arrival time and burst time of the process as input.

Step -4: Remaining process = size , time_quantam_i1=3, time_quantam_i2=6;
. current_time=0 and Repeat step -5 to step - 9 while remaining process != 0

Step -5: if(remanning_bt[Process_no]<time_quantam_i1 && remanning_bt
. [Process_no]>=0&&current_time<9)

current_time+=remanning_bt[Process_no];

remanning_bt[Process_no]=0

indicator = 1

time_quantam_itration++

remmaning_process--

Step -6: else if(remanning_bt[Process_no]>0&&current_time<9)

if(time_quantam_itration==1)

remanning_bt[Process_no]-=time_quantam_i1

time_quantam_itration++

current_time+=time_quantam_i1

else if(time_quantam_itration==2)

remanning_bt[Process_no]-=time_quantam_i2

current_time+=time_quantam_i2

Step -7: else if(remanning_bt[Process_no]<9 && remanning_bt
. [Process_no]>=3&&current_time<9)

current_time+=remanning_bt[Process_no];

remanning_bt[Process_no]=0;

remmaning_process-- , indicator = 1

Step -8: else if(remanning_bt[Process_no]>3&&current_time<9)

remanning_bt[Process_no]-=time_quantam_i2

current_time+=time_quantam_i2

Step -9: if(remanning_bt[Process_no]==0 && indicator==1)

Remening_proc--

completion_time[Process_no]=current_time

trunaround_time[Process_no]=completion_time[Process_no]-
. arival_time[Process_no]

waiting_time[Process_no]=trunaround_time[Process_no]-
. bust_time[Process_no]

indicator = 0

Step -10: End of step 4 loop, set:- Process_no=0

Repeat Step -11 to Step - 12 while remaining process != 0

Step -11: Pick a job whose burst time is lesser among all of them.

Step -12: Execute that job and set

completion_time[Process_no]=current_time

trunaround_time[Process_no]=completion_time[Process_no]-
. arival_time[Process_no]

waiting_time[Process_no]=trunaround_time[Process_no]-
. bust_time[Process_no]

Step -13: End of step step 10 loop.

Step -14: Repeat for int I = 0,1,2…. Size-1

. **A)** Calculate average waiting time and average burst time.

. **B)** Print all the information.

Time Complexity

Step -1 and Step -2 having time complexity constant time complexity : O(1).

Step -3 having time complexity no of process : O(n).

Step -4 to Step-9 having time complexity two iteration : O(m).

Step -10 to Step 13 having time complexity : $O(n^2)$.

Step -11 having time complexity no of process : O(n).

Overall Time Complexity is : **$O(n^2)$.**

Boundary Conditions

1. The first iteration should be follow Round Robin scheduling algorithm with time quantum of 3.
    - Arrival time must be greater and equal to 0
    - If any process having burst time less than 3, then set its completion time is burst time and remove from ready queue.
2. Second iteration also should be follow the Round Robin scheduling algorithm but having time quantum of 6.
    - Arrival time must be greater and equal to 0
    - And also check the ready queue if any process waiting already then execute first that very process.
    - If any process having burst time less than 9, then set its completion time is burst time and remove from ready queue.
3. After the second iteration follow then Shortest Job First scheduling algorithm.
    - Current time quantum must be greater or equals to 9
    - And also check the ready queue if any process waiting already then execute first that very process according to which have less time required to fully execute.
    - While picking up any shortest job to execute must be check that its arrival time equals of greater than current execution time.