Smt. Chandibai Himathmal Mansukhani College

Contents

Smt. Chandibai Himathmal Mansukhani College

USCS3P01:USCS303 – OPERATING SYSYEM (OS)
PRACTIAL 3

Practical – 01 : Implement RR (Round-Robin) Scheduling Algorithm in Java

Practical Date : 27 – 07 – 2021

Practical Aim : Implement RR Scheduling Algorithm In Java.

CPU Scheduling Algorithms

Round – Robin (RR) scheduling algorithm is mainly designed for time – sharing system.

This algorithm is similar to FCFS scheduling, but in Round – Robin (RR) scheduling, pre-emption is added which enables the system to switch between processes.

Round – Robin scheduling algorithm is used to schedule process fairly each job a time slot or quantum and the interrupting the job if it is not competed by then the job come after the job which is arrived in the quantum time that makes these scheduling fairly .

Algorithm :

Step 1: Input the number of processes and time quanta or time slice required to be scheduled using RR, burst time for each process.

Step 2: Choose the first process in the ready queue, set a timer to interrupt it after time quantum and dispatches it. Check if any other process request has arrived. If a process request arrives during the quantum time in which another process is executing, then add the new process to the Ready queue.

Step 3: After the quantum time has passed, check for any processes in the Ready queue. If the ready queue is empty then continue the current process. If the queue not empty and the current process is not complete, then add the current process to the end of the ready queue.

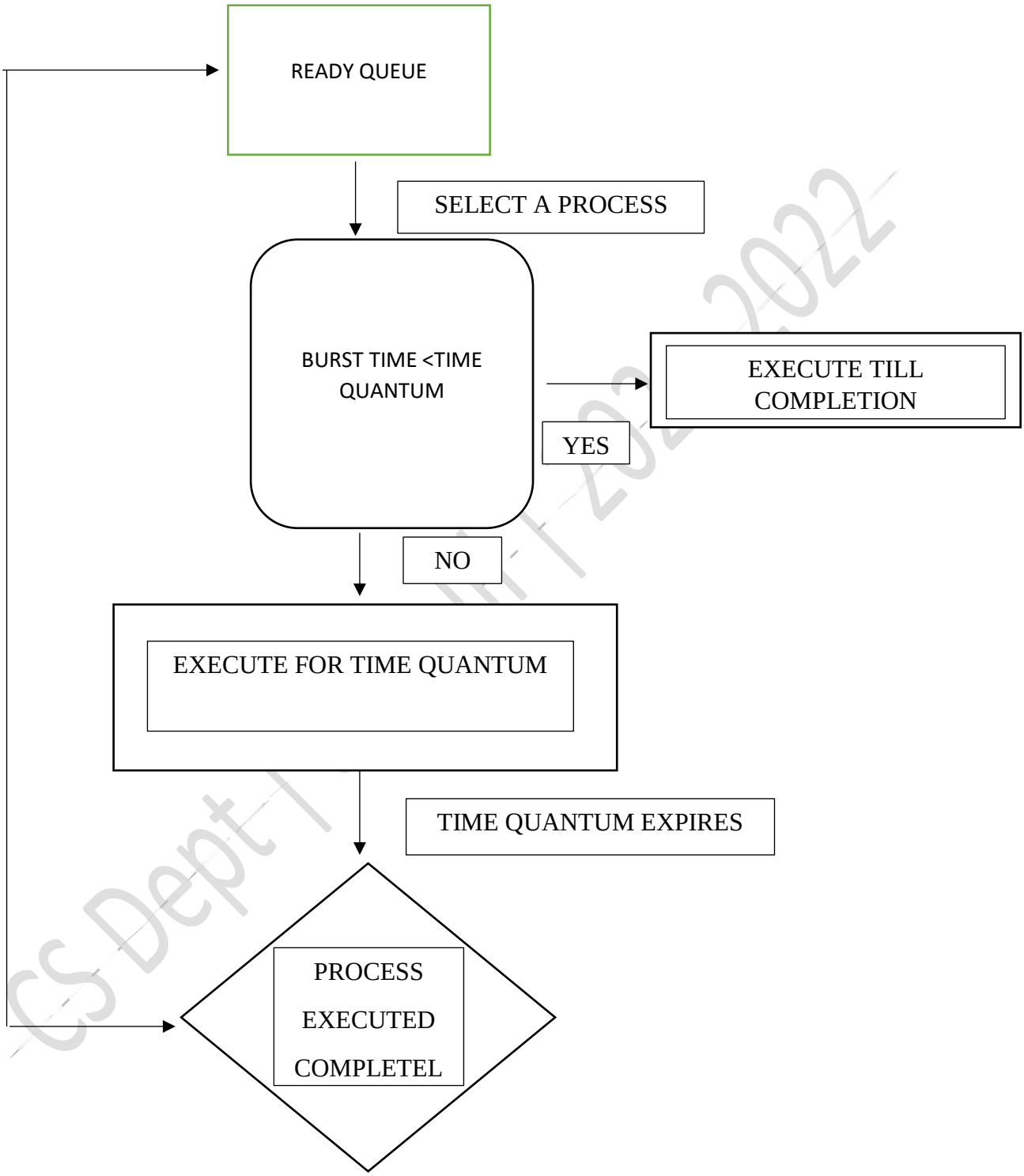
Step 4: Take the first process from the Ready queue and start executing it. Calculate the Turn Around Time and Waiting Time for each process using RR.

Step 5: Repeat all steps above from Step 2 to Step 4.

Step 6: If the process is complete and the ready queue is empty then the task is complete.

Step 7: Calculate the Average Waiting Time and Average Turn Around Time.

Flow Chart :


READY QUEUE
SELECT A PROCESS
BURST TIME <TIME QUANTUM
EXECUTE TILL COMPLETION
YES
NO
EXECUTE FOR TIME QUANTUM
TIME QUANTUM EXPIRES
PROCESS EXECUTED COMPLETEL

Solved Examples :

Example 1:

Consider the following example containing three processes arriving at time t = 0 ms.

Process ID	Burst Time
P0	24
P1	3
P2	3

Step 1: Consider the time quanta / time slice - 4 ms. Step 2: Following shows the scheduling and execution of processes.

Step 2.1: PO process arrives at 0 with 24 ms as the burst time which is greaterthan time quanta - 4 ms. So PO executes for 4 ms and goes in waiting queue.

System Time	:	0
Process Scheduled	:	P0
Remaining Time	:	24 – 4 = 20
Waiting Time	:	0 + 0 = 0
Turn Around Time		0 + 4 = 4

Step 2.2: Next P1 process executes for 3 ms which is greater than quanta time. So P1 executes and gets terminated.

System Time	:	0
Process Scheduled	:	P0 , P1
Remaining Time	:	3 – 4 = -1 =0
Waiting Time	:	4 – 0 = 4
Turn Around Time		4 + 3 = 7

Step 2.3: Next P2 process executes for 3 ms which is greater than quanta time. So P2 executes and gets terminated.

System Time	:	7
Process Scheduled	:	P0 , P1 , P2
Remaining Time	:	3 – 4 = -1 =0
Waiting Time	:	7– 0 = 7
Turn Around Time		7 + 3 = 10

Step 2.4: Now PO turns comes again and it's the only process for execution so for 4 ms of quanta it gets executed.

System Time	:	10
Process Scheduled	:	P0 , P1 , P2 , P0
Remaining Time	:	20 – 4 = 16
Waiting Time	:	0
Turn Around Time		10+ 4 = 14

Step 2.5: Again, PO continues to execute for next 4 ms. Waiting for PO will be zero.

System Time	:	7
Process Scheduled	:	P0 , P1 , P2 , P0 , P0
Remaining Time	:	16 – 4 = 12
Waiting Time	:	0
Turn Around Time		14 + 4 = 18

Step 2.6: PO continues to execute for next 4 ms.

System Time	:	7
Process Scheduled	:	P0 , P1 , P2 , P0 , P0 , P0
Finish Time	:	12 – 4 = 12
Turn Around Time	:	18 + 4 = 22

Step 2.7: PO continues to execute for next 4 ms.

System Time	:	7
Process Scheduled	:	P0 , P1 , P2 , P0 , P0 , P0 , P0
Finish Time	:	8 – 4 = 4
Turn Around Time	:	22 + 4 = 26

Step 2.8: PO continues to execute for next 4 ms.

System Time	:	7
Process Scheduled	:	P0 , P1 , P2 , P0 , P0 , P0 , P0 , P0
Finish Time	:	4 – 4 = 0
Turn Around Time	:	26 + 4 = 30

Step 3: Calculate Average Waiting Time and Average Turn Around Time.

Average Waiting Time	=	(6 + 4 + 7) / 3
	=	17 / 3
	=	5.666667

Average Turn Around Time	=	(30 + 7 + 10) / 3
	=	47 / 3
	=	16

Step 4: After scheduling of all provided processes:

Process ID	**Burst Time**	**Turn Around Time (Completion Time - Arrival Time)**	**Waiting Time (Turn Around Time – Burst Time)**
P0	24	30 – 0 = 30	30 – 24 = 6
P1	3	4 + 3 = 7	7 – 3 = 4
P2	3	7 + 3 = 10	10 – 3 = 7
Average		15.666667	5.666667

Gnatt Chart :

Process ID	Burst Time	Turn Around Time (Completion Time - Arrival Time)	Waiting Time (Turn Around Time – Burst Time)
P0	24	30 – 0 = 30	30 – 24 = 6
P1	3	4 + 3 = 7	7 – 3 = 4
P2	3	7 + 3 = 10	10 – 3 = 7
Average		15.666667	5.666667

P0	P1	P2	P0	P0	P0	P0	P0
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 25 26	27 28 29 30

Example 2:

Consider the following example containing three processes arrive at same time having time slice as 1ms

Process ID	Burst Time
P0	2
P1	1
P2	6

Solution:

Process ID	Burst Time	Turn Around Time (Completion Time - Arrival Time)	Waiting Time (Turn Around Time – Burst Time)
P0	2	4 – 0 = 4	4 – 2 = 2
P1	1	2 - 0 = 2	2 – 1 = 1
P2	6	9 – 0 = 9	9 – 6 = 3
Average		5.000000	2.000000

P0	P1	P2	P0	P2	P2	P2	P2	P2	P2
0	1	2	3	4	5	6	7	8	9

Example 3:

Consider the following example containing three processes arrive at same time having same time. Time Quanta = 3

Process ID	Burst Time
P0	7
P1	3
P2	2
P3	10
P4	8

Solution:

Process ID	Burst Time	Turn Around Time (Completion Time - Arrival Time)	Waiting Time (Turn Around Time – Burst Time)
P0	7	24 – 0 = 24	24 – 7 =17
P1	3	6 – 0 = 6	6 – 3 =3
P2	2	8 – 0 = 8	8 – 2 = 6
P3	10	30 – 0 =30	30 – 10 =20
P4	8	29 – 0 =29	29 – 8 =21
Average		19.4000000	13.4000000

P0	P1	P2	P3	P4	P0	P3	P4	P0	P3	P4	P3
0	3	6	8	11	14	17	20	23	24	29	30

Implementation :

```
//Name: ABHISHEKNIKAM
//Batch:B2
//PRN: 2020016400805951
//Date:28/7/2021
//Prac-03: RR(with no preemption)Algorithm

import java.util.Scanner;
class P3_RR_AN
{
	public static void main(String args[])
	{
		Scanner input = new Scanner(System.in);
		int i, j, k, q, sum = 0;
		System.out.print("Enter number of process: ");
		int n = input.nextInt();
		int burstTime[] = new int[n];
		int waitingTime[] = new int[n];
		int turnAroundTime[] = new int[n];
		int a[] = new int[n];
		System.out.println("Enter Burst Time of each process: ");
		for (i = 0; i < n; i++)
		{
			System.out.print("Enter Burst Time for Process - P" + (i) +" : ");
			burstTime[i] = input.nextInt();
			a[i] = burstTime[i];
		}
```

```
System.out.print("Enter Time quantum: ");
q=input.nextInt();
for (i = 0; i<n; i++)
        waitingTime[i] = 0;
int timer = 0; // Current time
// Keep traversing processes in round robin manner until all of them are not done.
do
{
        for (i = 0; i<n; i++)
        {
// If burst time of a process is greater than 0 then only need to process further

        if (burstTime[i] > q)
                {
// Increase the value of ti.e. shows how much time a process has been processed
                        timer += q;
// Decrease the burst time of current process by quantum
                        burstTime[i] -= q;
                        for (j = 0; j<n; j++)
                        {
                                if ((j!= i) && (burstTime[j] != 0))
                                waitingTime[j] += q;
                        }
                } // if ends
// If burst time is smaller than or equal to quantum. Last cycle for this process
                        else
                        {
// Increase the value of t i.e. shows how much time a process has been processed
                        timer += burstTime[i];
                        for (j = 0; j<n; j++)
```

```
                    {
                        if ((j != i) && (burstTime[j] != 0))
                    waitingTime[j] += burstTime[i];
                    }
// As the process gets fully executed make its remaining burst time = 0
                    burstTime[i] = 0;
                    } // else ends
                }
            sum = 0;
            for (k = 0; k<n; k++)
            sum += burstTime[k];
            } while (sum != 0);
// calculating turnaround time by adding waiting Time + burst Time
            for (i = 0; i<n; i++)
            turnAroundTime[i] = waitingTime[i] + a[i];
            float total = 0;
            for (int x : waitingTime)
            {
            total += x;
            }
            float averageWaitingTime = total / n;
            total = 0;
            for (int y : turnAroundTime)
            {
                total += y;
            }
            float averageTurnAroundTime = total / n;
// print on console the order of processes scheduled using Round-robin Algorithm
            System.out.println("RR Algorithm: ");
            System.out.format("%20s %20s %20s %20s\n","Processld","BurstTime","Waiting
Time","TurnAroundTime");
```

```
                for (i = 0; i<n; i++)
                {
                System.out.format("%20s %20d %20d %20d\n","P"+(i),a[i],
waitingTime[i],turnAroundTime[i]);
                }
                System.out.format("%40s %20f
%20f\n","Average",averageWaitingTime,averageTurnAroundTime);
        }
}
```

Input :

```
Enter number of process: 3
Enter burst Time of each process:
Enter burst Time for Process-P0: 24
Enter burst Time for Process-P1: 3
Enter burst Time for Process-P2: 3
Enter Time quantum: 4
```

Output :

ProcessId	burstTime	waitingTime	turnAroundTime
P0	24	6	30
P1	3	4	7
P2	3	7	10
	Average	5.666667	15.666667

Sample Output 1 :

```
Enter number of process: 3
Enter burst Time of each process:
Enter burst Time for Process-P0: 24
Enter burst Time for Process-P1: 3
Enter burst Time for Process-P2: 3
Enter Time quantum: 4
RR Algorithm:
          ProcessId          burstTime         waitingTime      turnAroundTime
                 P0                 24                   6                  30
                 P1                  3                   4                   7
                 P2                  3                   7                  10
                                Average           5.666667           15.666667
```

Sample Output 2 :

```
Enter number of process: 3
Enter burst Time of each process:
Enter burst Time for Process-P0: 2
Enter burst Time for Process-P1: 1
Enter burst Time for Process-P2: 6
Enter Time quantum: 1
RR Algorithm:
          ProcessId          burstTime        waitingTime      turnAroundTime
                 P0                  2                  2                  4
                 P1                  1                  1                  2
                 P2                  6                  3                  9
                            Average           2.000000           5.000000
```

Sample Output 3 :

```
Enter number of process: 5
Enter burst Time of each process:
Enter burst Time for Process-P0: 7
Enter burst Time for Process-P1: 3
Enter burst Time for Process-P2: 2
Enter burst Time for Process-P3: 10
Enter burst Time for Process-P4: 8
Enter Time quantum: 3
RR Algorithm:
          ProcessId          burstTime        waitingTime      turnAroundTime
                P0                  7                 17                 24
                P1                  3                  3                  6
                P2                  2                  6                  8
                P3                 10                 20                 30
                P4                  8                 21                 29
                               Average          13.400000          19.400000
```