|  |  |  |
| --- | --- | --- |
| **CL-2006 Operating Systems** | **LAB - 8**  **Simulation of Round Robin CPU scheduling & Highest Response Ration Next (HRRN) .** | |
|  | |  |

**Round Robin Scheduling Algorithm**

Round Robin scheduling algorithm is one of the most popular scheduling algorithm which can

actually be implemented in most of the operating systems. The Algorithm focuses on Time Sharing.

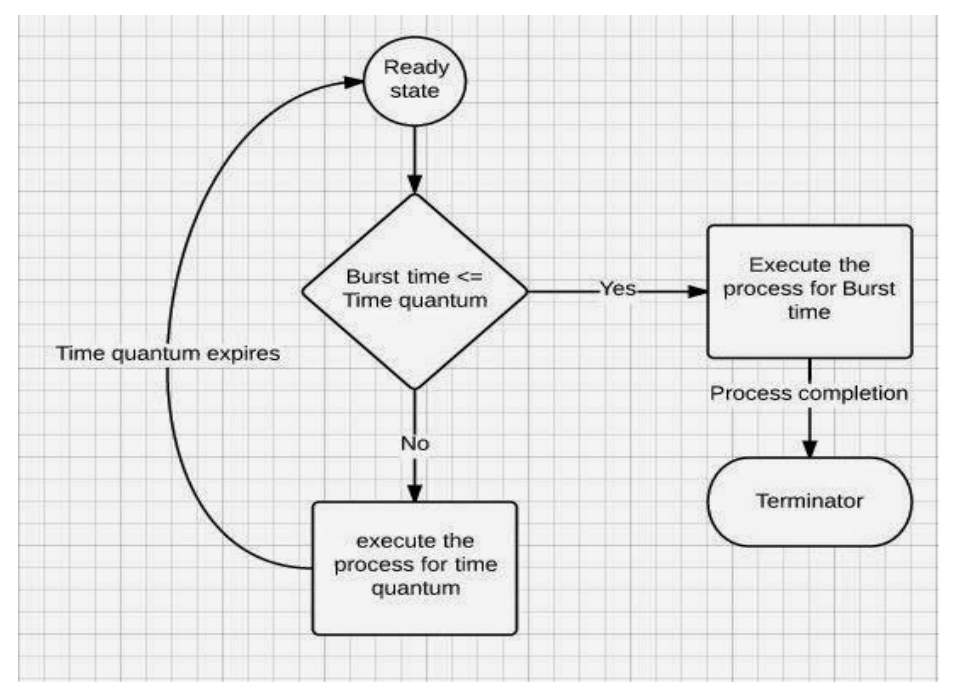
In this algorithm, every process gets executed in a cyclic way. A certain time slice is defined in the

system which is called time quantum. Each process present in the ready queue is assigned the CPU

for that time quantum, if the execution of the process is completed during that time then the process

will terminate else the process will go back to the ready queue and waits for the next turn to

complete the execution.



Implementation

For round robin scheduling algorithm, read the number of processes/jobs in the system, their CPU

burst times, and the size of the time slice. Time slices are assigned to each process in equal portions

and in circular order, handling all processes execution. This allows every process to get an equal

chance. Calculate the waiting time and turnaround time of each of the processes accordingly.

Steps to find Completion times of all processes:

**1- Create an array rem\_bt[] to keep track of remaining burst time of processes. This array is initially a copy of bt[] (burst times array)**

**2- Create another array ct[] to store completion times of processes. Initialize this array as 0.**

**3- Initialize time : t = 0**

**4- Keep traversing the all processes while all processes are not done. Do following for i'th process if it is not done yet.**

**a- If rem\_bt[i] > quantum**

**(i) t = t + quantum**

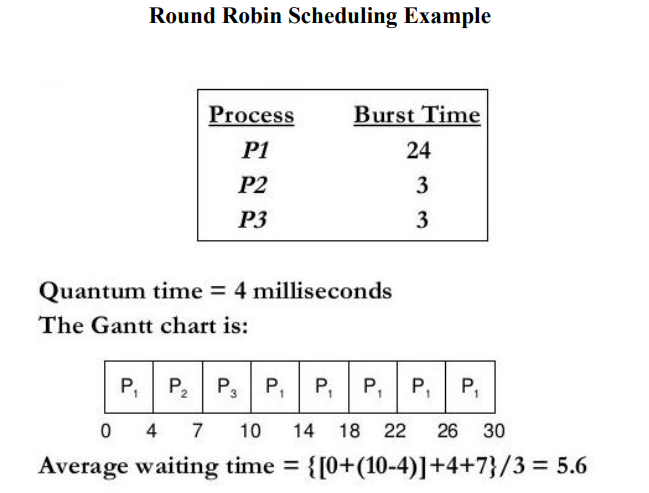
**(ii) bt\_rem[i] -= quantum;**

**b- Else // Last cycle for this process**

**(i) t = t + bt\_rem[i];**

**(ii) ct[i]=t;**

**(iii) bt\_rem[i] = 0; // This process is over**

****

**Highest Response Ratio Next (HRRN) Scheduling:**

**Highest Response Ratio Next (HRNN) is one of the most optimal scheduling algorithms. This is a non-preemptive algorithm in which, the scheduling is done on the basis of an extra parameter called Response Ratio. A Response Ratio is calculated for each of the available jobs and the Job with the highest response ratio is given priority over the others.**

**Response Ratio is calculated by the given formula.**

1. **Response Ratio = (W+S)/S**

**Where,**

1. **W → Waiting Time**
2. **S → Service Time or Burst Time**

**If we look at the formula, we will notice that the job with the shorter burst time will be given priority but it is also including an extra factor called waiting time. Since,**

1. **HRNN α W**
2. **HRNN α (1/S)**

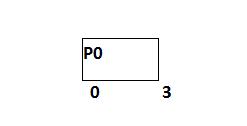
**Hence,**

1. **This algorithm not only favors shorter job but it also concern the waiting time of the longer jobs.**
2. **Its mode is non preemptive hence context switching is minimal in this algorithm.**

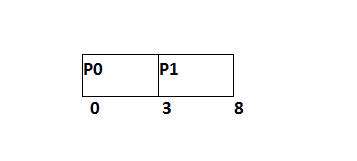
**In the following example, there are 5 processes given. Their arrival time and Burst Time are given in the table.**

|  |  |  |
| --- | --- | --- |
| **Process ID** | **Arrival Time** | **Burst Time** |
| **0** | **0** | **3** |
| **1** | **2** | **5** |
| **2** | **4** | **4** |
| **3** | **6** | **1** |
| **4** | **8** | **2** |

**At time 0, The Process P0 arrives with the CPU burst time of 3 units. Since it is the only process arrived till now hence this will get scheduled immediately.**

****

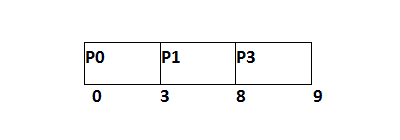
**P0 is executed for 3 units, meanwhile, only one process P1 arrives at time 3. This will get scheduled immediately since the OS doesn't have a choice.**

****

**P1 is executed for 5 units. Meanwhile, all the processes get available. We have to calculate the Response Ratio for all the remaining jobs.**

1. **RR (P2) = ((8-4) +4)/4 = 2**
2. **RR (P3) = (2+1)/1 = 3**
3. **RR (P4) = (0+2)/2 = 1**

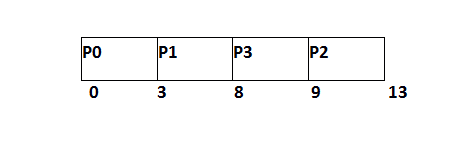
**Since, the Response ratio of P3 is higher hence P3 will be scheduled first.**

****

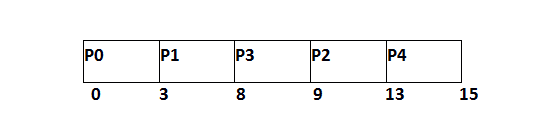
**P3 is scheduled for 1 unit. The next available processes are P2 and P4. Let's calculate their Response ratio.**

1. **RR ( P2) = (5+4)/4 = 2.25**
2. **RR (P4) = (1+2)/2 = 1.5**

**The response ratio of P2 is higher hence P2 will be scheduled.**

****

**Now, the only available process is P4 with the burst time of 2 units, since there is no other process available hence this will be scheduled.**

****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process ID** | **Arrival Time** | **Burst Time** | **Completion Time** | **Turn Around Time** | **Waiting Time** |
| **0** | **0** | **3** | **3** | **3** | **0** |
| **1** | **2** | **5** | **8** | **6** | **1** |
| **2** | **4** | **4** | **13** | **9** | **5** |
| **3** | **6** | **1** | **9** | **3** | **2** |
| **4** | **8** | **2** | **15** | **7** | **5** |

**Average Waiting Time = 13/5**

**Tasks:**

**01: Implement the round robin scheduling algorithm with a time quantum of 2 and use the above table for reference. Calculate the waiting time and average waiting time.**

**02: Use the same table as Q1 and again implement the same scheduling algorithm but this time implement a dynamic approach to time quantum (Keep changing the value of it). Compare the results of both a static time quantum vs a dynamic time quantum by comparing its average waiting times.**

**03: Implement the Highest Response Ratio Next Scheduling algorithm using a new table (You would have to design your own table) and calculate the wait times and compare it to the other two questions wait times to yield which one is better.**