CPU Scheduling

- A process execution consists of a cycle of CPU execution or wait and i/o execution or wait. Normally a process alternates between two states. Process execution begin with the CPU burst that may be followed by a i/o burst, then another CPU and i/o burst and so on. eventually in the last will end up on CPU burst. So, process keep switching between the CPU and i/o during execution.

Terminology

- **Arrival Time (AT):** Time at which process enters a ready state.
- **Burst Time (BT):** Amount of CPU time required by the process to finish its execution.
- **Completion Time (CT):** Time at which process finishes its execution.
- **Turn Around Time (TAT):** Completion Time (CT) – Arrival Time (AT), WT + BT
- **Waiting Time:** Turn Around Time (TAT) – Burst Time (BT)
- **Response Time:** Is the time it takes to start responding, not the time it takes to output the response.

Scheduling criteria

Different CPU-scheduling algorithms have different properties, and the choice of a particular algorithm may favour one class of processes over another. So, in order to efficiently select the scheduling algorithms following criteria should be taken into consideration:

- **CPU utilization:** Keeping the CPU as busy as possible.
- **Throughput:** If the CPU is busy executing processes, then work is being done. One measure of work is the number of processes that are completed per time unit, called throughput.
- **Turnaround time:** The interval from the time of submission of a process to the time of completion is the turnaround time.
- **Waiting time:** Waiting time is the sum of the periods spent waiting in the ready queue.
- **Response Time:** Is the time it takes to start responding, not the time it takes to output the response.
- Note: The CPU-scheduling algorithm does not affect the amount of time during which a process executes I/0; it affects only the amount of time that a process spends waiting in the ready queue. It is desirable to maximize CPU utilization and throughput and to minimize turnaround time, waiting time, and response time.

Type of scheduling

- CPU scheduling decision may take place under the following circumstances:
- **Non-Pre-emptive:** Under Non-Pre-emptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state. i.e. A process will leave the CPU willingly it can't be forced out. They are rigid as even if a critical process enters the ready queue the process running CPU is not disturbed.
  - When a process completes its execution
  - When a process leaves CPU voluntarily to perform some i/o or other operations and enters waiting state.
- **Pre-emptive**
  - If a new process enters in the ready state, in case of high priority
  - When process switches from running to ready state because of time quantum expire.

FCFS (FISRT COME FIRST SERVE)

- FCFS is the simplest scheduling algorithm, as the name suggest, the process that requests the CPU first is allocated the CPU first. Implementation is managed by FIFO Queue. It is always non pre-emptive in nature.

Example: Consider the following table of arrival time and burst time for three processes P_0, P_1, P_2, P_3, P_4. What is the average Waiting Time and Turnaround Time for the five processes?

P. No	Arrival Time (AT)	Burst Time (BT)
P_0	0	4
P_1	1	3
P_2	2	1
P_3	3	2
P_4	4	5

Ans. We make a Gantt chart (Bar chart that illustrates a particular Schedule).

P_1(0-4)	P_2(4-7)	P_3(7-8)	P_4(8-10)	P_5(10-15)

Calculating the various parameters:

P. No	AT	BT	CT	TAT	WT)
P_0	0	4	4	4	0
P_1	1	3	7	6	3
P_2	2	1	8	6	5
P_3	3	2	10	7	5
P_4	4	5	15	11	6
Total				34	19

Average TAT = 34/5 = 6.8
Average WT = 19/5 = 3.8

Example: Calculating the various parameters:

P. No	AT	BT	CT	TAT	WT
P_0	0	3			
P_1	2	2			
P_2	6	4			
Average					

Example: Calculating the various parameters:

P. No	AT	BT	CT	TAT	WT
P_0	2	4			
P_1	1	2			
P_2	0	3			
P_3	4	2			
P_4	3	1			
Average					

Example: Calculating the various parameters:

P. No	AT	BT	CT	TAT	WT
P_0	6	4			
P_1	2	5			
P_2	3	3			
P_3	1	1			
P_4	4	2			
P_5	5	6			
Average					

- **Advantage**
  - Easy to understand, and can easily be implemented using Queue data structure.
  - Can be used for Background processes where execution is not urgent.
- **Disadvantage**
  - FCFS suffers from convoy which means smaller process have to wait larger process, which result into large average waiting time. This effect results in lower CPU and device utilization than might be possible if the shorter processes were allowed to go first.
  - This algo fails with time sharing system or interactive systems
  - Higher average waiting time and TAT compared to other algorithms.
  - The FCFS algorithm is thus particularly troublesome for time-sharing systems (due to its non-pre-emptive nature), where it is important that each user get a share of the CPU at regular intervals.

Shortest Job First (SJF)(non-pre-emptive)
Shortest Remaining Time First (SRTF)/ (Pre-emptive)

- Whenever we make a decision of selecting the next process for CPU execution, out of all available process, CPU is assigned to the process having smallest burst time requirement. When the CPU is available, it is assigned to the process that has the smallest next CPU burst. If there is a tie, FCFS is used to break tie
- It supports both version non-pre-emptive and pre-emptive (purely greedy approach)
  - In Shortest Job First (SJF)(non-pre-emptive) once a decision is made and among the available process, the process with the smallest CPU burst is scheduled on the CPU, it cannot be pre-empted even if a new process with the smaller CPU burst requirement then the remaining CPU burst of the running process enter in the system
  - In Shortest Remaining Time First (SRTF) (Pre-emptive) whenever a process enters in ready state, again we make a scheduling decision weather, this new process with the smaller CPU burst requirement then the remaining CPU burst of the running process and if it is the case then the running process is pre-empted and new process is scheduled on the CPU.
  - This version (SRTF) is also called optimal is it guarantee minimal average waiting time.

Example on SJF (Non-Pre-emptive), Calculate Average TAT and WT?

P. No	A.T	B.T
P_1	1	7
P_2	2	5
P_3	3	1
P_4	4	2
P_5	5	8

Ans. Constructing the Gantt chart

P_0 (0-1)	P_1 (1-8)	P_3 (8-9)	P_4 (9-11)	P_2 (11-16)	P_5 (16-24)

P. No	A.T	B.T	C.T	TAT	WT
P1	1	7	8	7	0
P2	2	5	16	14	9

P3	3	1	9	6	5
P4	4	2	11	7	5
P5	5	8	24	19	11
Total				53	30

Average Waiting Time: 30/5 = 6
Average Turn Around Time: 53/5 = 10.6

Example: Consider the following processes:

P. No	AT	BT
P1	0	10
P2	3	6
P3	7	1
P4	8	3

Calculate Average WT and TAT?
Ans.

P1 (0-3)	P2 (3-7)	P3 (7-8)	P2 (8-10)	P4 (10-13)	P1 (13-20)

P. No	AT	BT	CT	TAT	WT
P1	0	10	20	20	10
P2	3	6	10	7	1
P3	7	1	8	1	0
P4	8	3	13	5	2
Total				33	13

Average TAT: 33/4 = 8.25
Average WT: 13/4 = 3.25

Example:

Process	AT	BT	CT	TAT	WT
P_1	0	6			
P_2	1	4			

P_3	2	3			
P_4	3	1			
P_5	4	2			
P_6	5	1			

Example:

Process	AT	BT	CT	TAT	WT
P_1	3	4			
P_2	4	2			
P_3	5	1			
P_4	2	6			
P_5	1	8			
P_6	2	4			

Advantage:

- Pre-emptive version guarantees minimal average waiting time so some time also referred as optimal algorithm.
- Provide a standard for other algo in terms of average waiting time
- Provide better average response time compare to FCFs

Q Consider an arbitrary set of CPU-bound processes with unequal CPU burst lengths submitted at the same time to a computer system. Which one of the following process scheduling algorithms would minimize the average waiting time in the ready queue? **(GATE - 2016) (1 Marks)**

(a) Shortest remaining time firs

(b) Round-robin with time quantum less than the shortest CPU burst

(c) Uniform random

(d) Highest priority first with priority proportional to CPU burst length

Answer: (A)

Q For the processes listed in the following table, which of the following scheduling schemes will give the lowest average turnaround time? **(GATE-2015) (2 Marks)**

Process	Arrival Time	Processing Time
A	0	3
B	1	6
C	4	4
D	6	2

(A) First Come First Serve
(B) Non-pre-emptive Shortest Job First
(C) Shortest Remaining Time
(D) Round Robin with Quantum value two
Answer: (C)

Disadvantage

- This algo cannot be implemented as there is no way to know the length of the next CPU burst.
- Here process with the longer CPU burst requirement goes into starvation.
- No idea of priority, longer process has poor response time.

Q Consider the following four processes with arrival times (in milliseconds) and their length of CPU bursts (in milliseconds) as shown below: **(GATE - 2019) (2 Marks)**

Process	Arrival Time	CPU Time
P_1	0	3
P_2	1	1
P_3	3	3
P_4	4	Z

These processes are run on a single processor using pre-emptive Shortest Remaining Time First scheduling algorithm. If the average waiting time of the processes is 1 millisecond, then the value of Z is _____.

Answer: 2

Q Consider the following three processes with the arrival time and CPU burst time given in milliseconds: **(NET-July-2018)**

Process	Arrival Time	Burst Time
P1	0	7
P2	1	4
P3	2	8

The Gantt Chart for pre-emptive SJF scheduling algorithm is _____.

a)

P_1(0-7)	P_2(7-13)	P_3(13-21)

b)

P_1(0-1)	P_2(1-5)	P_1(5-11)	P_3(11-19)

c)

P_1(0-7)	P_2(7-11)	P_3(11-19)

d)

P_2(0-4)	P_3(4-12)	P_1(12-19)

Answer: (b)

Q Consider the following four processes with the arrival time and length of CPU burst given in milliseconds: **(NET-Nov-2018)**

Process	Arrival Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

The average waiting time for pre-emptive SJF scheduling algorithm is _____.

a) 6.5 **b)** 7.5 **c)** 6.75 **d)** 7.75

Answer: (A)

Q Consider the following set of processes, with the arrival times and the CPU-burst times given in milliseconds **(GATE - 2017) (2 Marks)**

Process	Arrival Time	CPU Time
P_1	0	5
P_2	1	3
P_3	2	3
P_4	4	1

What is the average turnaround time for these processes with the pre-emptive shortest remaining processing time first (SRPT) algorithm?

(A) 5.50 **(B)** 5.75 **(C)** 6.00 **(D)** 6.25

Answer: (A)

Q Consider the following CPU processes with arrival times (in milliseconds) and length of CPU bursts (in milliseconds) as given below: **(GATE - 2017) (1 Marks)**

Process	Arrival Time	CPU Time
P_1	0	7
P_2	3	3
P_3	5	5
P_4	6	2

If the pre-emptive shortest remaining time first scheduling algorithm is used to schedule the processes, then the average waiting time across all processes is _____ milliseconds.

Answer: 3

Q Consider three CPU intensive processes P1, P2, P3 which require 20,10 and 30 units of time, arrive at times 1,3 and 7 respectively. Suppose operating system is implementing Shortest Remaining Time first (pre-emptive scheduling) algorithm, then _____ context switches are required (suppose context switch at the beginning of Ready queue and at the end of Ready queue are not counted). **(NET-Aug-2016)**

a) 3 **b)** 2 **c)** 4 **d)** 5

Answer: (A)

Q Consider the following processes, with the arrival time and the length of the CPU burst given in milliseconds. The scheduling algorithm used is pre-emptive shortest remaining-time first.

Process	Arrival Time	CPU Time
P_1	0	10
P_2	3	6
P_3	7	1
P_4	8	3

The average turnaround time of these processes is _____ milliseconds. **(GATE - 2016) (2 Marks)**

Answer: 8.25

Q Consider the following set of processes that need to be scheduled on a single CPU. All the times are given in milliseconds? **(GATE-2014) (2 Marks)**

Process Name	Arrival Time	Execution Time
A	0	6
B	3	2
C	5	4
D	7	6
E	10	3

Using the *shortest remaining time first* scheduling algorithm, the average process turnaround time (in msec) is _____.

Answer: 7.2

Q An operating system uses shortest remaining time first scheduling algorithm for pre-emptive scheduling of processes. Consider the following set of processes with their arrival times and CPU burst times (in milliseconds) **(GATE-2014) (2 Marks)**

Process	Arrival Time	Burst Time
P_1	0	12
P_2	2	4
P_3	3	6
P_4	8	5

The average waiting time (in milliseconds) of the processes is _____.
Answer: 5.5

Q Consider the following table of arrival time and burst time for three processes P_0, P_1 and P_2.

Process	Arrival Time	Burst Time
P_0	0	9
P_1	1	4
P_2	2	9

The pre-emptive shortest job first scheduling algorithm is used. Scheduling is carried out only at arrival or completion of processes. What is the average waiting time for the three processes? **(GATE-2011) (2 Marks)**
(A) 5.0 ms **(B)** 4.33 ms **(C)** 6.33 **(D)** 7.33
Answer: (A)

Q An operating system uses Shortest Remaining Time first (SRT) process scheduling algorithm. Consider the arrival times and execution times for the following processes:

Process	Arrival Time	CPU Time
P_1	0	20
P_2	15	25
P_3	30	10
P_4	45	15

What is the total waiting time for process P2? **(GATE - 2007) (2 Marks)**

(A) 5 **(B)** 15 **(C)** 40 **(D)** 55

Answer: (B)

Q Consider three CPU-intensive processes, which require 10, 20 and 30 time units and arrive at times 0, 2 and 6, respectively. How many context switches are needed if the operating system implements a shortest remaining time first scheduling algorithm? Do not count the context switches at time zero and at the end? **(GATE-2006) (1 Marks)**

(A) 1 **(B)** 2 **(C)** 3 **(D)** 4

Answer: (B)

- **Note: -** There are several approaches either to make SJF implementable or to improve it in terms of starvation or in other sense. Will discuss them later.
- As SJF is not implementable, we can use the one technique where we try to predict the CPU burst of the next coming process. The method is used as exponential averaging technique, where we consider the previous value and previous prediction.
- $Tau_{(n+1)}$ = alpa t_n + (1- alpa) tau_n. This idea is also more of theoretical importance as most of the time the burst requirement of the coming process may vary by a large extent and if the burst time requirement of all the process is approximately same then there is no advantage of using this scheme.
  - **Example:**

Process	t	tau
P_1	10	20
P_2	12	
P_3	14	
P_4		

Priority Scheduling

- Here a priority is associated with each process. At any instance of time out of all available process, CPU is allocated to the process which possess highest priority (may be higher or lower number). Tie is broken using FCFS order. No importance to senior or burst time. It supports both non-pre-emptive and pre-emptive versions.
- In Priority (non-pre-emptive) once a decision is made and among the available process, the process with the highest priority is scheduled on the CPU, it cannot be pre-empted even if a new process with higher priority more than the priority of the running process enter in the system.
- In Priority (pre-emptive) once a decision is made and among the available process, the process with the highest priority is scheduled on the CPU, and if it a new process with higher priority more than the priority of the running process enter in the system, then we do a context switch and the processor is provided to the new process with higher priority.
- Priorities are generally indicated by some fixed range of numbers, such as 0 to 7 or 0 to 4,095. There is no general agreement on whether 0 is the highest or lowest priority, it can vary from systems to systems.

Example: We will assume in example that low numbers have higher priority. Non-Pre-emptive Priority Scheduling

P. No	AT	BT	Priority
P_1	0	10	3
P_2	0	1	1
P_3	0	2	4
P_4	0	1	5
P_5	0	5	2

Gantt chart:

P_2 (0-1)	P_5 (1-6)	P_1 (6-16)	P_3 (16-18)	P_4 (18-19)

P. No	AT	BT	Priority	CT	TAT	WT
P_1	0	10	3	16	16	6
P_2	0	1	1	1	1	0
P_3	0	2	4	18	18	16
P_4	0	1	5	19	19	18
P_5	0	5	2	6	6	1

Total					60	41

Average WT: 41/5 = 8.2 ms
Average TAT: 60/5 = 12 ms

Q Example: Pre-emptive priority scheduling

P. No	AT	BT	Priority
P_1	1	4	4
P_2	2	2	5
P_3	2	3	7
P_4	3	5	8(H)
P_5	3	1	5
P_6	4	2	6

Ans. Gantt chart

P_0 (0-1)	P_1 (1-2)	P_3 (2-3)	P_4 (3-4)	P_4 (4-8)	P_3 (8-10)	P_6 (10-12)	P_2 (12-14)	P_5 (14-15)	P_1 (15-18)

P. No	AT	BT	Priority	CT	TAT	WT
P1	1	4	4	18	17	13
P2	2	2	5	14	12	10
P3	2	3	7	10	8	5
P4	3	5	8	8	5	0
P5	3	1	5	15	12	11
P6	4	2	6	12	8	6
Total					62	45

Average TAT: 62/6 = 10.33
Average WT: 45/6: 7.5

Q Example: Pre-emptive priority scheduling

Process	AT	BT	Priority	CT	TAT	WT
P_1	0	50	4			
P_2	20	20	1(H)			
P_3	40	100	3			
P_4	60	40	2			

Q Example: Pre-emptive priority scheduling

Process	AT	BT	Priority	CT	TAT	WT
P_1	1	4	5			
P_2	2	5	2			
P_3	3	6	6			
P_4	0	1	4			
P_5	4	2	1			
P6	5	3	8(H)			

- **Advantage**
  - Gives a facility specially to system process.
  - Allow us to run important process even if it is a user process.
- **Disadvantage**
  - Here process with the smaller priority may starve for the CPU
  - No idea of response time or waiting time.
- Note: - Specially use to support system process or important user process
- Ageing: - a technique of gradually increasing the priority of processes that wait in the system for long time. E.g. priority will increase after every 10 mins

Q Consider the set of processes with arrival time (in milliseconds), CPU burst time (in milliseconds), and priority (0 is the highest priority) shown below. None of the processes have I/O burst time.

Process	AT	BT	Priority
P_1	0	11	2
P_2	5	28	0
P_3	12	2	3
P_4	2	10	1
P_5	9	16	4

The average waiting time (in milliseconds) of all the processes using preemptive priority scheduling algorithm is ____. **(GATE-2017) (2 Marks)**
Note: This question appeared as Numerical Answer Type.
Answer: 29

Q Some of the criteria for calculation of priority of a process are:
a. Processor utilization by an individual process.
b. Weight assigned to a user or group of users.
c. Processor utilization by a user or group of processes
In fair share scheduler, priority is calculated based on: **(NET-Jan-2017)**
a) only (a) and (b) **b)** only (a) and (c) **c)** (a), (b) and (c) **d)** only (b) and (c)
Answer: C

Q Consider a uniprocessor system executing three tasks T1, T2 and T3, each of which is composed of an infinite sequence of jobs (or instances) which arrive periodically at intervals of 3, 7 and 20 milliseconds, respectively. The priority of each task is the inverse of its period and the available tasks are scheduled in order of priority, with the highest priority task scheduled first. Each instance of T1, T2 and T3 requires an execution time of 1, 2 and 4 milliseconds, respectively. Given that all tasks initially arrive at the beginning of the 1st milliseconds and task preemptions are allowed, the first instance of T3 completes its execution at the end of _____ milliseconds. **(GATE-2015) (2 Marks)**
Answer: 12

Q Consider a preemptive priority-based scheduling algorithm based on dynamically changing priority. Larger priority number implies higher priority. When the process is waiting for CPU in the ready queue (but not yet started execution), its priority changes at a rate a = 2. When it starts running, its priority changes at a rate b = 1. All the processes are assigned priority value 0 when they enter ready queue. Assume that the following processes want to execute: **(NET-Dec-2013)**

Process	Arrival Time	Burst Time
P1	0	4
P2	1	1
P3	2	2
P4	3	1

The time quantum q = 1. When two processes want to join ready queue simultaneously, the process which has not executed recently is given priority. The finish time of processes P1, P2, P3 and P4 will respectively be

(A) 4, 5, 7 and 8 **(B)** 8, 2, 7 and 5

(C) 2, 5, 7 and 8 **(D)** 8, 2, 5 and 7

Answer: (B)

Q The problem of indefinite blockage of low-priority jobs in general priority scheduling algorithm can be solved using: **(NET-Dec-2012)**

a) Parity bit **b)** Aging **c)** Compaction **d)** Timer

Answer: (B)

Q We wish to schedule three processes P1, P2 and P3 on a uniprocessor system. The priorities, CPU time requirements and arrival times of the processes are as shown below.

Process	Priority	CPU time required	Arrival time (hh: mm: ss)
P1	10(highest)	20 sec	00:00:05
P2	9	10 sec	00:00:03
P3	8 (lowest)	15 sec	00:00:00

We have a choice of pre-emptive or non-pre-emptive scheduling. In pre-emptive scheduling, a late-arriving higher priority process can pre-empt a currently running process with lower priority. In non-pre-emptive scheduling, a late-arriving higher priority process must wait for the currently executing process to complete before it can be scheduled on the processor. What are the turnaround times (time from arrival till completion) of P2 using pre-emptive and non-pre-emptive scheduling respectively? **(GATE-2005) (2 Marks)**

(A) 30 sec, 30 sec **(B)** 30 sec, 10 sec

(C) 42 sec, 42 sec **(D)** 30 sec, 42 sec

Answer: (D)

Round robin

- This algo is designed for time sharing systems, where it is not, necessary to complete one process and then start another, but to be responsive and divide time of CPU among the process in the ready state. Here ready queue is treated as a circular queue (FIFO). It is similar to FCFS scheduling, but pre-emption is added to enable the system to switch between processes.
- The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval equivalent 1 Time quantum (where value of TQ can be anything).
- We fix a time quantum, up to which a process can hold the CPU in one go, with in which either a process terminates or process must release the CPU and enter the ready queue and wait for the next chance.
- The process may have a CPU burst of less than given time quantum. In this case, the process itself will release the CPU voluntarily. CPU Scheduler will select the next process for execution. OR The CPU burst of the currently running process is longer than 1-time quantum, the timer will go off and will cause an interrupt to the operating system. A context switch will be executed, and the process will be put at the tail of the ready queue. The CPU scheduler will then select the next process in the ready queue.
- RR is always pre-emptive in nature.
- If there are n processes in the ready queue and the time quantum is q, then each process gets 1/n of the CPU time in chunks of at most q time units. Each process must wait no longer than (n - 1) x q time units until its next time quantum.
- If the time quantum is extremely large, the RR policy is the same as the FCFS policy. If the time quantum is extremely small (say, 1 millisecond), the RR approach is called processor sharing and (in theory) creates the appearance that each of n processes has its own processor running at 1/n the speed of the real processor. We also need also to consider the effect of context switching on the performance of RR scheduling.
- We have only one process of 10-time units. If the quantum is 12-time units, the process finishes in. less than 1-time quantum, with no overhead. If the quantum is 6-time units, however, the process requires 2 quanta, resulting in a context switch. If the time quantum is 1-time unit, then nine context switches will occur, slowing the execution of the process accordingly. In General, If TQ is less the number of Context Switch increases, response time will be less. If TQ is large the number of Context Switch will decrease and response time increases.
- Optimal Time Quantum, so that every problem gets CPU after t seconds: $q \leq t - ns / n-1$. A rule of thumb is that 80 percent of the CPU bursts should be shorter than the time quantum.

Q **Example:** Let Time Quantum (TQ) = 2

P. No	AT	BT
P_1	0	4
P_2	1	5
P_3	2	2
P_4	3	1
P_5	4	6
P_6	6	3

Gantt Chart:

P1 (0-2)	P2 (2-4)	P3 (4-6)	P1 (6-8)	P4 (8-9)	P5 (9-11)	P2 (11-13)	P6 (13-15)	P5 (15-17)	P2 (17-18)	P6 (18-19)	P5 (19-21)

P. No	AT	BT	CT	TAT	WT
P_1	0	4	8	8	4
P_2	1	5	18	17	12
P_3	2	2	6	4	2
P_4	3	1	9	6	5
P_5	4	6	21	17	11
P_6	6	3	19	13	10
Total				65	44

Average TAT: 65/6 = 10.83 ms
Average WT = 44/6 = 7.33 ms

Process	Arrival Time	Burst Time	CT	TAT	WT
P_1	5	5			
P_2	4	6			
P_3	3	7			
P_4	1	9			
P_5	2	2			
P6	6	3			

- **Advantage**
  - Perform best in terms of average response time

- Works will in case of time-sharing systems, client server architecture and interactive system
- kind of SJF implementation

- **Disadvantage**
  - Longer process may starve
  - Performance depends heavily on time quantum - If value of the time quantum is very less, then it will give lesser average response time (good but total no of context switches will be more, so CPU utilization will be less), If time quantum is very large then average response time will be more bad, but no of context switches will be less, so CPU utilization will be good.
  - No idea of priority

Q consider the following set of processes and the length of CPU burst time given in milliseconds:

Process	CPU Burst Time
P1	5
P2	7
P3	6
P4	4

Assume that processes being scheduled with round robin scheduling algorithm with time quantum 4ms. Then the waiting for p4 is _____. **(NET-DEC-2018)**

a) 0 **b) 4** **c) 6** **d) 12**

Answer: (D)

Q Consider the following set of processes with the length of CPU burst time in milliseconds (ms):

Process	Burst Time	Priority
A	6	3
B	1	1
C	2	3
D	1	4
E	5	2

Assume that processes are stored in ready queue in following order: A – B – C – D – E

Using round robin scheduling with time slice of 1 ms, the average turnaround time is **(NET-Spet-2013)**

(A) 8.4 ms **(B)** 12.4 ms **(C)** 9.2 ms **(D)** 9.4 ms

Answer: A

Q A scheduling algorithm assigns priority proportional to the waiting time of a process. Every process starts with priority zero (the lowest priority). The scheduler re-evaluates the process priorities every T time units and decides the next process to schedule. Which one of the following is TRUE if the processes have no I/O operations and all arrive at time zero? **(GATE-2013) (1 Marks)**

(A) This algorithm is equivalent to the first-come-first-serve algorithm

(B) This algorithm is equivalent to the round-robin algorithm.

(C) This algorithm is equivalent to the shortest-job-first algorithm..

(D) This algorithm is equivalent to the shortest-remaining-time-first algorithm

Answer: (B)

Q In round robin CPU scheduling as time quantum is increased the average turnaround time **(NET-June-2012)**

(A) increases **(B)** decreases

(C) remains constant **(D)** varies irregularly

Answer: (D)

Q Consider n processes sharing the CPU in a round-robin fashion. Assuming that each process switch takes s seconds, what must be the quantum size q such that the overhead resulting from process switching is minimized but at the same time each process is guaranteed to get its turn at the CPU at least every t seconds? **(GATE-1998) (1 Marks)** **(NET-Dec-2012)**

a) $q \leq (t-ns)/(n-1)$ **b)** $q \geq (t-ns)/(n-1)$

c) $q \leq (t-ns)/(n+1)$ **d)** $q \geq (t-ns)/(n+1)$

Answer: (A)

Q If the time-slice used in the round-robin scheduling policy is more than the maximum time required to execute any process, then the policy will **(GATE-2008) (1 Marks)**

(A) degenerate to shortest job first **(B)** degenerate to priority scheduling

(C) degenerate to first come first serve **(D)** none of the above

Answer: (C)

Q In processor management, round robin method essentially uses the pre-emptive version of ..... **(NET-JUNE-2006)**

(A) FILO **(B)** FIFO **(C)** SJF **(D)** Longest time first

Answer: B

Q Four jobs to be executed on a single processor system arrive at time 0+ in the order A, B, C, D. their burst CPU time requirements are 4, 1, 8, 1 time units respectively. The completion time of A under round robin scheduling with time slice of one-time unit is. **(GATE-1996) (2 Marks) (ISRO-2008)**

(a) 10 **(b) 4** **(c) 8** **(d) 9**

Answer: (D)

Q Which scheduling policy is most suitable for a time-shared operating system? **(GATE-1995) (1 Marks)**

(a) Shortest Job First **(b)** Round Robin **(c)** First Come First Serve **(d)** Elevator

Answer: (B)

Q Assume that the following jobs are to be executed on a single processor system

Job Id	CPU Burst Time
P	4
Q	1
R	8
S	1
T	2

The jobs are assumed to have arrived at time 0+ and in the order p, q, r, s, t. calculate the departure time (completion time) for job p if scheduling is round robin with time slice 1. **(GATE-1993) (2 Marks)**

(a) 4 **(b)** 10 **(c)** 11 **(d)** 12

Answer: (C)

Q In which of the following scheduling criteria, context switching will never take place? **(NET-July-2018)**

a) Round Robin
b) Pre-emptive SJF
c) Non-Pre-emptive SJF
d) Preemptive priority

Answer: (c)

Q Which of the following scheduling algorithms may cause starvation? **(NET-Jan-2017)**

a. First-come-first-served **b.** Round Robin **c.** Priority
d. Shortest process next **e.** Shortest remaining time first

a) a, c and e
b) c, d and e
c) b, d and e
d) b, c and d

Answer: B

Q A scheduling Algorithm assigns priority proportional to the waiting time of a process. Every process starts with priority zero (lowest priority). The scheduler reevaluates the process priority for every 'T' time units and decides next process to be scheduled. If the process has no I/O operations and all arrive at time zero, then the scheduler implements _____ criteria. **(NET-June-2016)**

a) Priority scheduling
b) Round Robin Scheduling
c) Shortest Job First
d) FCFS

Answer: B

Q Three processes A, B and C each execute a loop of 100 iterations. In each iteration of the loop, a process performs a single computation that requires t_c CPU milliseconds and then initiates a single I/O operation that lasts for $t_{i/o}$ milliseconds. It is assumed that the computer where the processes execute has sufficient number of I/O devices and the OS of the computer assigns different I/O devices to each process. Also, the scheduling overhead of the OS is negligible. The processes have the following characteristics:

Process Id	T_c	$T_{i/o}$
A	**100**	**500**
B	**350**	**500**
C	**200**	**500**

The processes A, B, and C are started at times 0, 5 and 10 milliseconds respectively, in a pure time-sharing system (round robin scheduling) that uses a time slice of 50 milliseconds. The time in milliseconds at which process C would complete its first I/O operation is _____.

(GATE-2014) (2 Mark)
Answer: 1000

Q Consider the following processes with time slice of 4 milliseconds (I/O requests are ignored):

Process	Arrival Time	Processing Time
A	0	8
B	1	4
C	2	9
D	3	5

The average turnaround time of these processes will be **(NET-June-2013)**
(A) 19.25 milliseconds
(B) 18.25 milliseconds
(C) 19.5 milliseconds
(D) 18.5 milliseconds
Answer: B

Q Pre-emptive scheduling is the strategy of temporarily suspending a running process **(NET-June-2012)**
a) before the CPU time slice expires
b) to allow starving processes to run
c) when it requires I/O
d) to avoid collision
Answer: B

Q Consider the 3 processes, P_1, P_2 and P_3 shown in the table. **(GATE-2012) (2 Marks)**

Process	Arrival Time	Time Unit Required
P_1	0	5
P_2	1	7
P_3	3	4

The completion order of the 3 processes under the policies FCFS and RR2 (round robin scheduling with CPU quantum of 2-time units) are
(A) FCFS: P_1, P_2, P_3 RR: P_1, P_2, P_3
(B) FCFS: P_1, P_3, P_2 RR: P_1, P_3, P_2
(C) FCFS: P_1, P_2, P_3 RR: P_1, P_3, P_2
(D) FCFS: P_1, P_3, P_2 RR: P_1, P_2, P_3
Answer: (C)

Q Which of the following statements are true? **(GATE-2010) (2 Marks)**
1) Shortest remaining time first scheduling may cause starvation

2) Pre-emptive scheduling may cause starvation

3) Round robin is better than FCFS in terms of response time

(A) 1 only **(B)** 1 and 3 only **(C)** 2 and 3 only **(D)** 1, 2 and 3

Answer: (D)

Q An example of a non-preemptive CPU scheduling algorithm is: **(NET-June-2008)**

(A) Shortest job first scheduling **(B)** Round robin scheduling.

(C) Priority scheduling **(D)** Fair share scheduling.

Answer: A

Q Consider three processes, all arriving at time zero, with total execution time of 10, 20 and 30 units, respectively. Each process spends the first 20% of execution time doing I/O, the next 70% of time doing computation, and the last 10% of time doing I/O again. The operating system uses a shortest remaining compute time first scheduling algorithm and schedules a new process either when the running process gets blocked on I/O or when the running process finishes its compute burst. Assume that all I/O operations can be overlapped as much as possible. For what percentage of time does the CPU remain idle? **(GATE-2006) (2 Marks)**

(A) 0% **(B)** 10.6% **(C)** 30.0% **(D)** 89.4%

Answer: (B)

Q The arrival time, priority, and duration of the CPU and I/O bursts for each of three processes P_1, P_2 and P_3 are given in the table below. Each process has a CPU burst followed by an I/O burst followed by another CPU burst. Assume that each process has its own I/O resource. **(GATE-2006) (2 Marks)**

Process	Arrival Time	Priority	Burst duration, CPU, I/O CPU
A	0	2	1,5,3
B	2	3(L)	3,3,1
C	3	1(H)	2,3,1

The multi-programmed operating system uses preemptive priority scheduling. What are the finish times of the processes P_1, P_2 and P_3 ?

(A) 11, 15, 9 **(B)** 10, 15, 9 **(C)** 11, 16, 10 **(D)** 12, 17, 11

Answer: (B)

Q .....is one of pre-emptive scheduling algorithm. **(NET-Dec-2006)**

(A) Shortest-Job-first **(B)** Round-robin

(C) Priority based **(D)** Shortest-Job-next

Answer: B

Q A uniprocessor computer system only has two processes, both of which alternate 10 ms CPU bursts with 90 ms I/O bursts. Both the processes were created at nearly the same time. The I/O of both processes can proceed in parallel. Which of the following scheduling strategies will result in the least CPU utilization (over a long period of time) for this system? **(GATE-2003) (2 Marks)**

(A) First come first served scheduling

(B) Shortest remaining time first scheduling

(C) Static priority scheduling with different priorities for the two processes

(D) Round robin scheduling with a time quantum of 5 ms

Answer: (D)

Longest Job First

- Process having longest Burst Time will get scheduled first.
- It can be both pre-emptive and non-pre-emptive in nature.
- The pre-emptive version is referred to as Longest Remaining Time First (LRTF) Scheduling Algorithm.

Example:

P. No	AT	BT
P_1	0	3
P_2	1	2
P_3	2	4
P_4	3	5
P_5	4	6

P_1 (0-3)	P_4 (3-8)	P_5 (8-14)	P_3 (14-18)	P_2 (18-20)

P. No	AT	BT	CT	TAT	WT
P_1	0	3	3	3	0
P_2	1	2	20	19	17
P_3	2	4	18	16	12
P_4	3	5	8	5	0
P_5	4	6	14	10	4
Total				53	33

Average TAT: 53/5 = 10.6 ms
Average WT: 33/5 = 6.6 ms

Longest Remaining Time First (LRTF)

- It is pre-emptive in nature.
- Longest Burst time process will get scheduled first.

Example:

P. No	AT	BT
P_1	0	2
P_2	0	4
P_3	0	8

P_1 (0-1)	P_3 (1-2)	P_3 (2-3)	P_3 (3-4)	P_3 (4-5)	P_2 (5-6)	P_3 (6-7)	P_2 (7-8)

P_3 (8-9)	P_2 (9-10)	P_3 (10-11)	P_1 (11-12)	P_2 (12-13)

P. No	AT	BT	CT	TAT	WT
P_1	0	2	12	12	10
P_2	0	4	13	13	9
P_3	0	8	14	14	6

Average TAT: 39/3 = 13 ms
Average WT: 25/3 = 8.33 ms

Q Consider three processes (process id 0, 1, 2 respectively) with compute time bursts 2, 4- and 8-time units. All processes arrive at time zero. Consider the longest remaining time first (LRTF) scheduling algorithm. In LRTF ties are broken by giving priority to the process with the lowest process id. The average turnaround time is? **(GATE-2006) (2 Marks)**

(A) 13 units **(B)** 14 units **(C)** 15 units **(D)** 16 units

Answer: (A)

Highest response ratio next (HRRN)

- Scheduling is a non-pre-emptive discipline, similar to shortest job next (SJN), in which the priority of each job is dependent on its estimated run time, and also the amount of time it has spent waiting.
- Jobs gain higher priority the longer they wait, which prevents indefinite waiting or in other words what we say starvation. In fact, the jobs that have spent a long time waiting compete against those estimated to have short run times.
- Response Ratio = (W + S)/S
- Here, **W** is the waiting time of the process so far and **S** is the Burst time of the process.
- So, the conclusion is it gives priority to those processes which have less burst time (or execution time) but also takes care of the waiting time of longer processes, thus preventing starvation.

Q Consider a set of n tasks with known runtimes r_1, r_2, …. r_n to be run on a uniprocessor machine. Which of the following processor scheduling algorithms will result in the maximum throughput? **(GATE-2001) (1 Marks)**

(A) Round-Robin
(B) Shortest-Job-First
(C) Highest-Response-Ratio-Next
(D) First-Come-First-Served

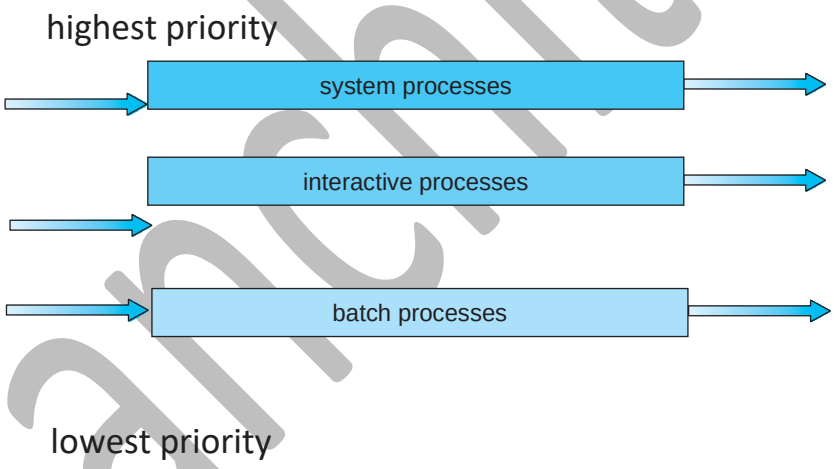
Answer: (C)

Q highest response ratio next scheduling policy favours ------- jobs, but it also limits the waiting time of ---------- jobs. **(GATE-1990) (1 Marks)**

Answer: "shorter, longer"

MULTIVEL QUEUE SCGEDULING

- Another class of scheduling algorithm has been created for situations in which processes are easily classified into different groups. For example, a common division is made between foreground (interactive) processes and background (batch) processes. These two types of processes have different response-time requirements and so may have different scheduling needs.
- A multilevel queue scheduling algorithm partitions the ready queue into several separate queues. The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority, or process type.
- Each queue has its own scheduling algorithm. For example, separate queues might be used for foreground and background processes. The foreground queue might be scheduled by an RR algorithm, while the background queue is scheduled by an FCFS algorithm.
- In addition, there must be scheduling among the queues, which is commonly implemented as fixed-priority preemptive scheduling. For example, the foreground queue may have absolute priority over the background queue.



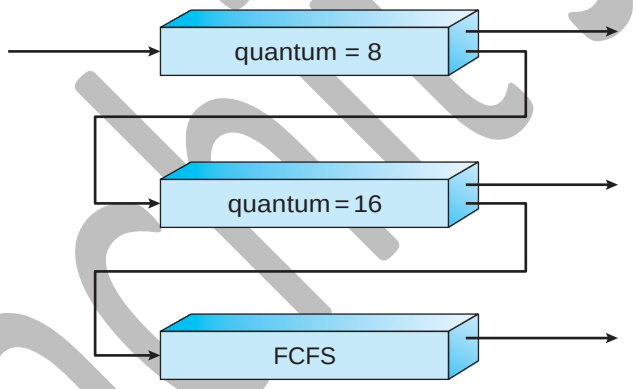
- Each queue has absolute priority over lower-priority queues. No process in the batch queue, for example, could run unless the queues for system processes, interactive processes, and interactive editing processes were all empty. If an interactive editing process entered the ready queue while a batch process was running, the batch process would be preempted.
- Another possibility is to time-slice among the queues. Here, each queue gets a certain portion of the CPU time, which it can then schedule among its various processes. For instance, in the foreground– background queue example, the foreground queue can be given 80 percent of the CPU time for RR scheduling among its processes, while the background queue receives 20 percent of the CPU

to give to its processes on an FCFS basis.

Q Consider the following justifications for commonly using the two-level CPU scheduling:
I. It is used when memory is too small to hold all the ready processes.
II. Because its performance is same as that of the FIFO.
III. Because it facilitates putting some set of processes into memory and a choice is made from that.
IV. Because it does not allow to adjust the set of in-core processes.
Which of the following is true? **(NET-Dec-2014)**
(A) I, III and IV **(B)** I and II **(C)** III and IV **(D)** I and III
Answer: D

MULTIVEL QUEUE SCHEDULING WITH FEEDBACK

- Normally, when the multilevel queue scheduling algorithm is used, processes are permanently assigned to a queue when they enter the system. If there are separate queues for foreground and background processes, for example, processes do not move from one queue to the other, since processes do not change their foreground or background nature. This setup has the advantage of low scheduling overhead, but it is inflexible.
- The multilevel feedback queue scheduling algorithm, in contrast, allows a process to move between queues. The idea is to separate processes according to the characteristics of their CPU bursts. If a process uses too much CPU time, it will be moved to a lower-priority queue. This scheme leaves I/O-bound and interactive processes in the higher-priority queues. In addition, a process that waits too long in a lower-priority queue may be moved to a higher-priority queue. This form of aging prevents starvation.



- A process entering the ready queue is put in queue 0.A process in queue 0 is given a time quantum of 8 milliseconds. If it does not finish within this time, it is moved to the tail of queue 1. If queue 0 is empty, the process at the head of queue 1 is given a quantum of 16 milliseconds. If it does not complete, it is preempted and is put into queue 2. Processes in queue 2 are run on an FCFS basis but are run only when queues 0 and 1 are empty.
- This scheduling algorithm gives highest priority to any process with a CPU burst of 8 milliseconds or less. Such a process will quickly get the CPU, finish its CPU burst, and go off to its next I/O burst. Processes that need more than 8 but less than 24 milliseconds are also served quickly, although with lower priority than shorter processes. Long processes automatically sink to queue 2 and are served in FCFS order with any CPU cycles left over from queues 0 and 1.

- In general, a multilevel feedback queue scheduler is defined by the following parameters:
    - The number of queues
    - The scheduling algorithm for each queue
    - The method used to determine when to upgrade a process to a higher-priority queue
    - The method used to determine when to demote a process to a lower-priority queue
    - The method used to determine which queue a process will enter when that process needs service
- The definition of a multilevel feedback queue scheduler makes it the most general CPU-scheduling algorithm. It can be configured to match a specific system under design. Unfortunately, it is also the most complex algorithm since defining the best scheduler requires some means by which to select values for all the parameters.

Q Which of the following statements is not true for Multi-Level Feedback Queue processor scheduling algorithm? **(NET-June-2015)**
a) Queues have different priorities
b) Each queue may have different scheduling algorithm
c) Processes are permanently assigned to a queue
d) This algorithm can be configured to match a specific system under design
Answer: C

Q Match the following:

List – I	List – II
a. Multilevel feedback queue	**i.** Time-slicing
b. FCFS	**ii.** Criteria to move processes between queues
c. Shortest process next	**iii.** Batch processing
d. Round robin scheduling	**iv.** Exponential smoothening

Codes: **(NET-June-2014)**

	a	b	c	d
(A)	i	iii	ii	iv
(B)	iv	iii	ii	i
(C)	iii	i	iv	i
(D)	ii	iii	iv	i

Answer: D

Q An example of a non-pre-emptive scheduling algorithm is: **(NET-Dec-2008)**

(A) Round Robin
(B) Priority Scheduling
(C) Shortest job first
(D) 2 level scheduling

Answer: C

Q Which of the following scheduling algorithms is non-preemptive? **(GATE-2002) (1 Marks)**

(A) Round Robin
(B) First-In First-Out
(C) Multilevel Queue Scheduling
(D) Multilevel Queue Scheduling with Feedback

Answer: (B)

Q Which of the following scheduling algorithms is non-pre-emptive? **(GATE-2001) (1 Marks)**

(A) Round Robin
(B) First-In First-Out
(C) Multilevel Queue Scheduling
(D) Multilevel Queue Scheduling with Feedback

Answer: (B)

Q An Operating System (OS) crashes on the average once in 30 days, that is, the Mean Time Between Failures (MTBF) = 30 days. When this happens, it takes 10 minutes to recover the OS, that is, the Mean Time To Repair (MTTR) = 10 minutes. The availability of the OS with these reliability figures is approximately: **(NET-AUG-2016)**

a) 96.97% **b)** 97.97% **c)** 99.009% **d)** 99.97%

Ans: d

Q Consider a uniprocessor system where new processes arrive at an average of five processes per minute and each process needs an average of 6 seconds of service time. What will be the CPU utilization? **(NET-JUNE-2014)**

a) 80 % **b)** 50 % **c)** 60 % **d)** 30 %

Ans: b