L-2.1: Process Scheduling Algorithms (Preemption Vs Non-Preemption) | CPU Scheduling in OS

In an operating system, the scheduling algorithm is responsible for selecting a process from the ready queue and placing it in the CPU. The goal is to keep multiple processes in the ready queue as much as possible to maximize the degree of multi-programming. Scheduling algorithms are classified into two categories: pre-emptive and non-pre-emptive.

Pre-emptive scheduling involves stopping a process in the running state and returning it to the ready queue to allow another process to be executed. Non-pre-emptive scheduling, on the other hand, completes the execution of a process before moving on to another one.

The pre-emptive scheduling technique has several advantages, such as responsiveness and priority. Time quantum is one reason for pre-emption. The CPU is programmed to execute a process for a specific period, and when it reaches the end of the time quantum, it returns the process to the ready queue, allowing another process to be executed. This approach ensures that all processes receive a fair share of the CPU's processing time.

Pre-emption is also used to prioritize some processes over others. The scheduler places higher-priority processes in the running queue first, ensuring that they receive the CPU's maximum attention. The lower-priority processes are moved to the ready queue and executed later.

In an operating system, the scheduling algorithm is responsible for selecting a process from the ready queue and placing it in the CPU. The goal is to keep multiple processes in the ready queue as much as possible to maximize the degree of multi-programming. Scheduling algorithms are classified into two categories: pre-emptive and non-pre-emptive.

Pre-emptive scheduling involves stopping a process in the running state and returning it to the ready queue to allow another process to be executed. Non-pre-emptive scheduling, on the other hand, completes the execution of a process before moving on to another one.

The pre-emptive scheduling technique has several advantages, such as responsiveness and priority. Time quantum is one reason for pre-emption. The CPU is programmed to execute a process for a specific period, and when it reaches the end of the time quantum, it returns the process to the ready queue, allowing another process to be executed. This approach ensures that all processes receive a fair share of the CPU's processing time.

Pre-emption is also used to prioritize some processes over others. The scheduler places higher-priority processes in the running queue first, ensuring that they receive the CPU's maximum attention. The lower-priority processes are moved to the ready queue and executed later.

L-2.2: What is Arrival, Burst, Completion, Turnaround, Waiting and Response time in CPU Scheduling

In operating systems, CPU scheduling is the process of allocating CPU time to processes that are waiting in the ready queue. The CPU scheduler selects one process from the ready queue and assigns the CPU to it for execution.

There are different CPU scheduling algorithms used in operating systems such as First-Come, First-Served (FCFS), Shortest-Job-First (SJF), Priority Scheduling, Round Robin, etc.

Let's take an example to understand the scheduling of CPU to processes in an operating system. Suppose we have three processes P1, P2, and P3 in the ready queue with their respective burst times and arrival times as shown in the table below:

Process	Burst Time	Arrival Time
P1	6	0
P2	4	1
P3	8	2

Now, let's apply different scheduling algorithms to schedule these processes:

1. First-Come, First-Served (FCFS) Scheduling: In this algorithm, the process that arrives first will get the CPU first. The order of execution will be P1, P2, P3. The waiting time, turnaround time, and completion time for each process are calculated using the formulas discussed earlier.
2. Shortest-Job-First (SJF) Scheduling: In this algorithm, the process with the shortest burst time will get the CPU first. The order of execution will be P2, P1, P3. The waiting time, turnaround time, and completion time for each process are calculated using the formulas discussed earlier.
3. Priority Scheduling: In this algorithm, each process is assigned a priority, and the process with the highest priority will get the CPU first. Let's say the priority of P1, P2, and P3 is 2, 1, and 3 respectively. The order of execution will be P3, P1, P2. The waiting time, turnaround time, and completion time for each process are calculated using the formulas discussed earlier.
4. Round Robin Scheduling: In this algorithm, each process gets a fixed time slice, and after that, the CPU is given to the next process in the ready queue. Let's say the time slice is 3 units. The order of execution will be P1, P2, P3, P1, P3, P1, P3, P3. The waiting time, turnaround time, and completion time for each process are calculated using the formulas discussed earlier.

Thus, different scheduling algorithms give different results in terms of waiting time, turnaround time, and completion time for each process. The choice of scheduling algorithm depends on the system's requirements and the nature of the processes being executed.

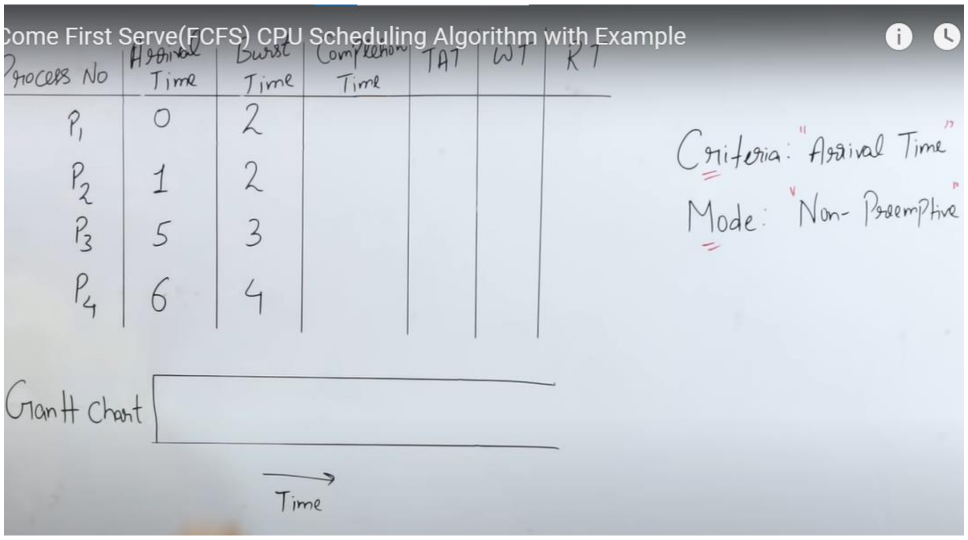
Let's say you want to go to the bank to deposit money. You enter the bank at 11:00 am, so the arrival time is 11:00 am. The amount of time it takes you to deposit the money depends on the clerk who is working with you, so that time is the burst time. Suppose you tell the clerk that you will take 15 minutes to deposit the money, so the burst time is 15 minutes.

After 15 minutes, you complete the transaction and leave the bank at 12:00 pm, so the completion time is 12:00 pm. The turnaround time is the total time you spent in the bank, which is 60 minutes (12:00 pm - 11:00 am).

However, the useful time you spent in the bank was only 15 minutes, which is the burst time. The waiting time is the difference between the turnaround time and the burst time, which is 45 minutes (60 minutes - 15 minutes). This waiting time could be due to factors such as waiting in a queue, filling out forms, or other delays.

- CPU scheduling involves several times, including arrival time, burst time, completion time, turnaround time, and waiting time.
- Arrival time is the time when a process enters the ready queue from the ready state.
- Burst time is the time required by a process to get executed on a CPU.
- Completion time is the point in time at which a process gets executed.
- Turnaround time is the time required for a process to complete, which is calculated as completion time minus arrival time.
- Waiting time is the time a process spends waiting in the ready queue, which is calculated as turnaround time minus burst time.
- Processes can be either CPU-bound or I/O-bound.

L-2.3: First Come First Serve(FCFS) CPU Scheduling Algorithm with Example



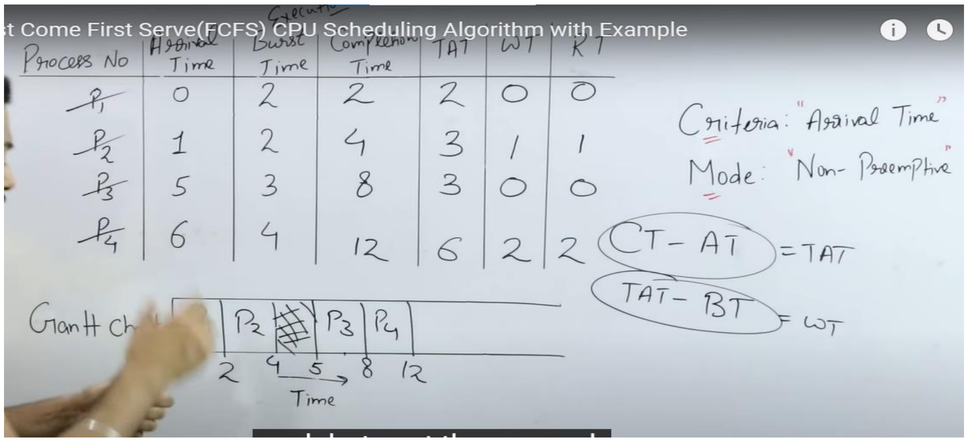
Process No	Arrival Time	Burst Time	Completion Time	TAT	WT	RT
P_1	0	2				
P_2	1	2				
P_3	5	3				
P_4	6	4				

Criteria: "Arrival Time"

Mode: "Non-Preemptive"

Gantt Chart

Time


t Come First Serve(FCFS) CPU Scheduling Algorithm with Example
Execution

Process No	Arrival Time	Burst Time	Completion Time	TAT	WT	RT
~~P_1~~	0	2	2	2	0	0
~~P_2~~	1	2	4	3	1	1
~~P_3~~	5	3	8	3	0	0
~~P_4~~	6	4	12	6	2	2

Criteria: "Arrival Time"

Mode: "Non-Preemptive"

CT − AT = TAT

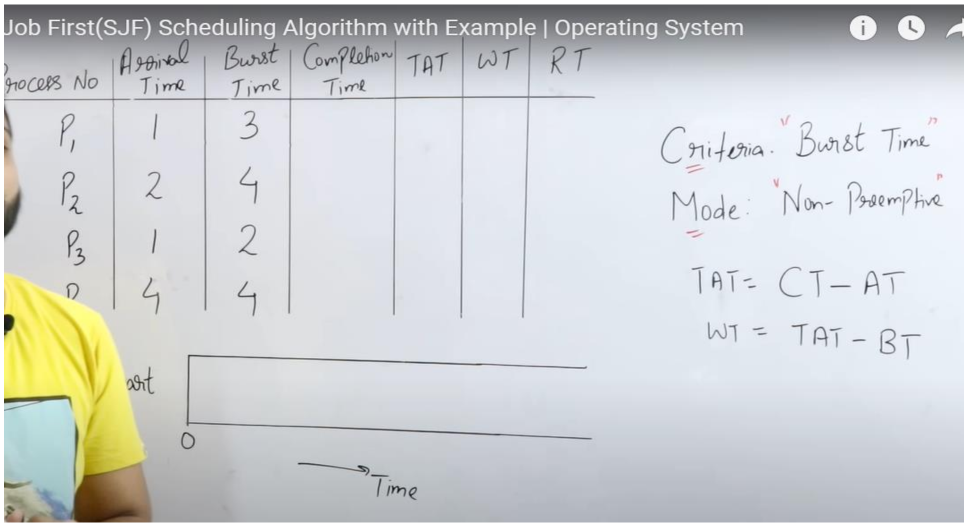
TAT − BT = WT

Gantt Chart

	P_2	▨	P_3	P_4
2	4	5	8	12

Time

L-2.4: Shortest Job First(SJF) Scheduling Algorithm with Example | Operating System



Job First(SJF) Scheduling Algorithm with Example | Operating System

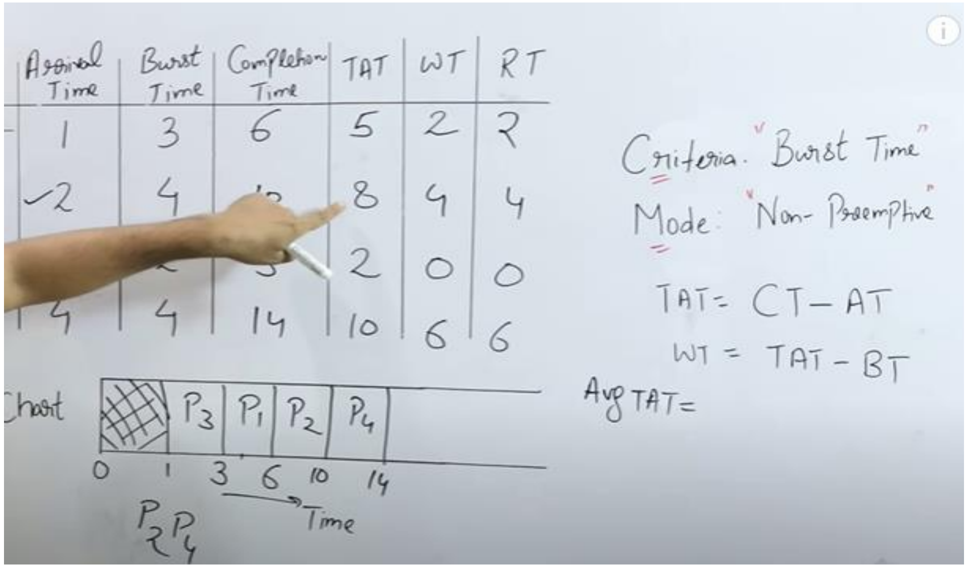
Process No	Arrival Time	Burst Time	Completion Time	TAT	WT	RT
P_1	1	3				
P_2	2	4				
P_3	1	2				
P	4	4				

Criteria: "Burst Time"

Mode: "Non-Preemptive"

TAT = CT − AT

WT = TAT − BT



Arrival Time	Burst Time	Completion Time	TAT	WT	RT
1	3	6	5	2	2
2	4	[illegible]	8	4	4
[illegible]	[illegible]	[illegible]	2	0	0
4	4	14	10	6	6

Criteria: "Burst Time"

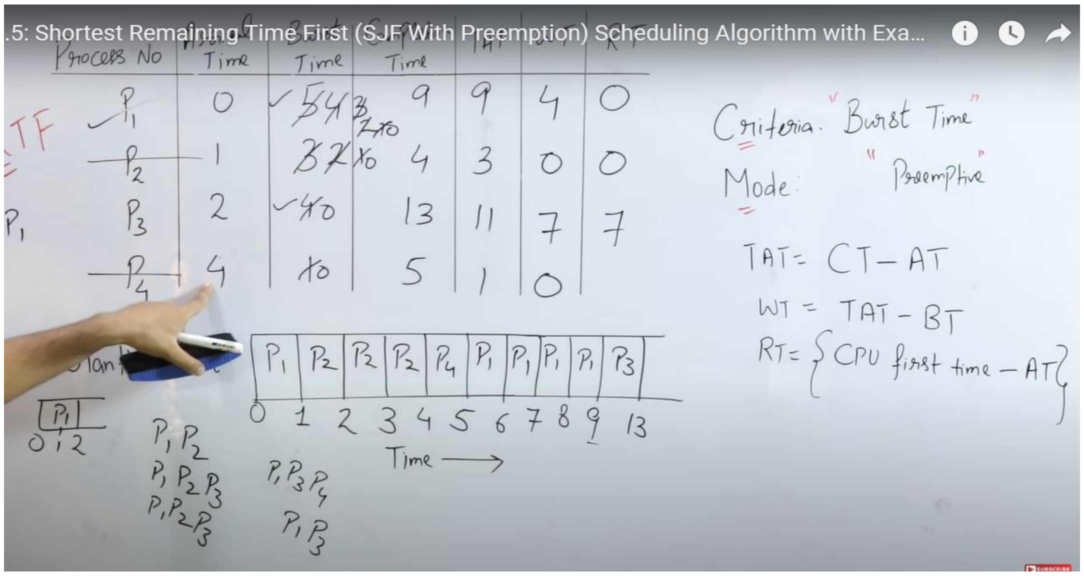
Mode: "Non-Preemptive"

TAT = CT − AT

WT = TAT − BT

Avg TAT =

L-2.5: Shortest Remaining Time First (SJF With Preemption) Scheduling Algorithm with Example



.5: Shortest Remaining Time First (SJF With Preemption) Scheduling Algorithm with Exa...

Process No	Time	Time	Time			
P1	0	~~5~~ ~~4~~ ~~3~~ ~~2~~ ~~1~~ 0	9	9	4	0
P2	1	~~3~~ ~~2~~ ~~1~~ 0	4	3	0	0
P3	2	~~4~~ 0	13	11	7	7
P4	4	~~1~~ 0	5	1	0	0

P1	P2	P2	P2	P4	P1	P1	P1	P1	P3

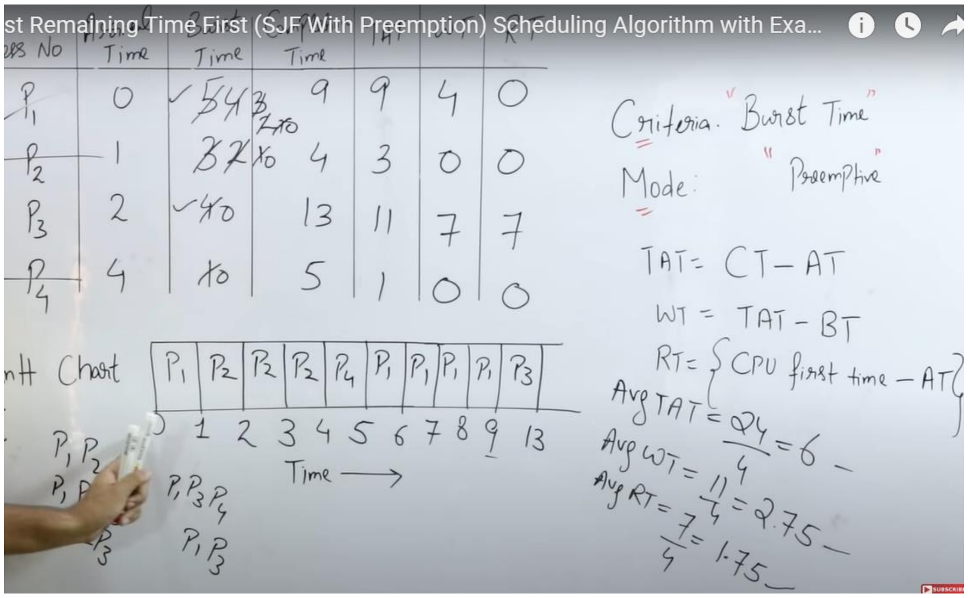
0 1 2 3 4 5 6 7 8 9 13

Time ⟶

P1 P2
P1 P2 P3
P1 P2 P3
P1 P3 P4
P1 P3

Criteria: "Burst Time"
Mode: "Preemptive"

TAT = CT − AT
WT = TAT − BT
RT = {CPU first time − AT}



st Remaining Time First (SJF With Preemption) Scheduling Algorithm with Exa...

Process No	Time	Time	Time			
P1	0	~~5~~ ~~4~~ ~~3~~ ~~2~~ ~~1~~ 0	9	9	4	0
P2	1	~~3~~ ~~2~~ ~~1~~ 0	4	3	0	0
P3	2	~~4~~ 0	13	11	7	7
P4	4	~~1~~ 0	5	1	0	0

Gantt Chart

P1	P2	P2	P2	P4	P1	P1	P1	P1	P3

0 1 2 3 4 5 6 7 8 9 13

Time ⟶

Criteria: "Burst Time"
Mode: "Preemptive"

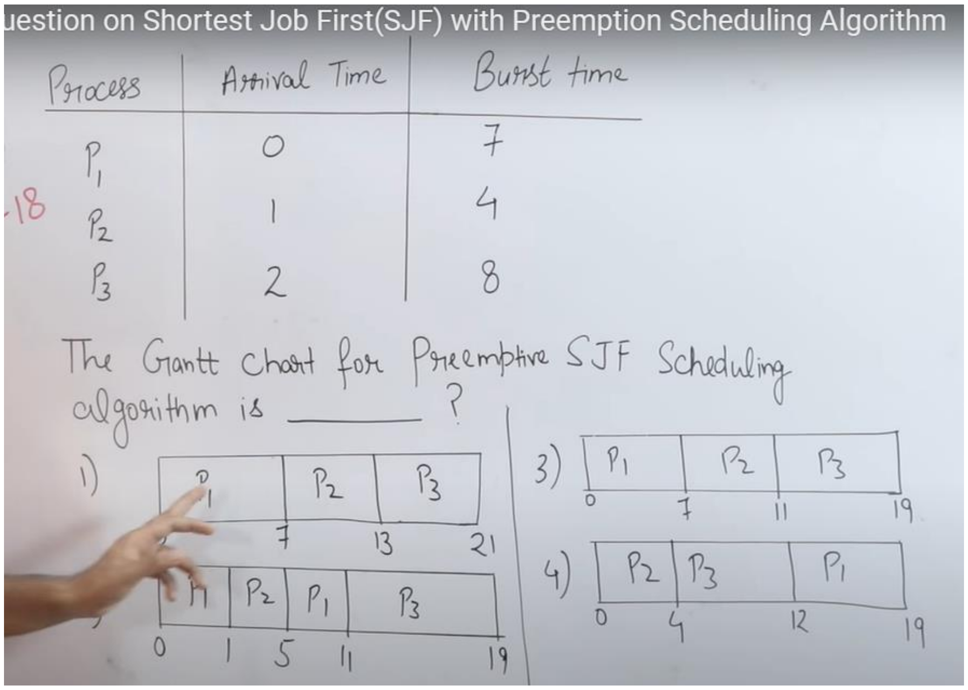
TAT = CT − AT
WT = TAT − BT
RT = {CPU first time − AT}

$\text{Avg TAT} = \frac{24}{4} = 6$

$\text{Avg WT} = \frac{11}{4} = 2.75$

$\text{Avg RT} = \frac{7}{4} = 1.75$

L-2.6: Question on Shortest Job First(SJF) with Preemption Scheduling Algorithm



uestion on Shortest Job First(SJF) with Preemption Scheduling Algorithm

-18

Process	Arrival Time	Burst time
P_1	0	7
P_2	1	4
P_3	2	8

The Gantt Chart for Preemptive SJF Scheduling algorithm is ______ ?

1)

P_1	P_2	P_3

7 13 21

3)

P_1	P_2	P_3

0 7 11 19

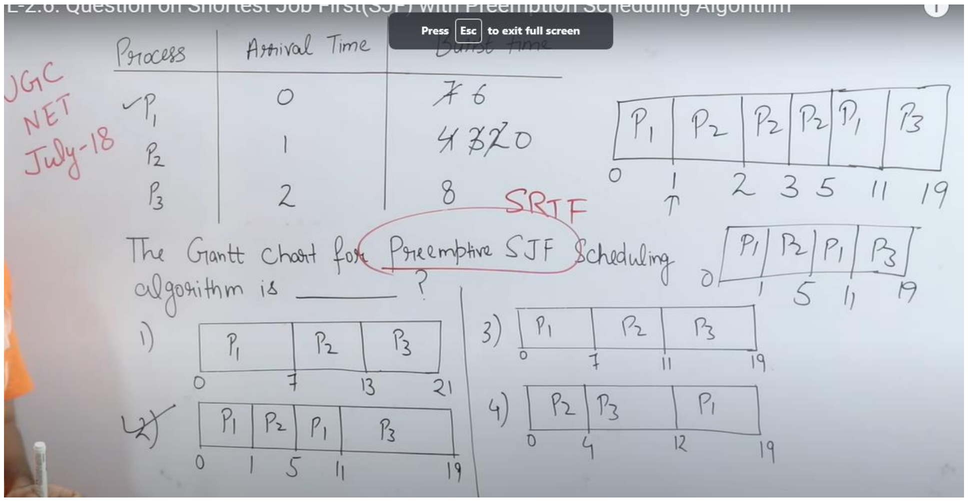
P_1	P_2	P_1	P_3

0 1 5 11 19

4)

P_2	P_3	P_1

0 4 12 19



L-2.6: Question on Shortest Job First(SJF) with Preemption Scheduling Algorithm

Press Esc to exit full screen

UGC NET July-18

Process	Arrival Time	Burst time
✓P_1	0	~~7~~ 6
P_2	1	~~4~~ ~~3~~ ~~2~~ ~~1~~ 0
P_3	2	8

SRTF

The Gantt Chart for Preemptive SJF Scheduling algorithm is ______ ?

P_1	P_2	P_2	P_2	P_1	P_3

0 1 2 3 5 11 19

P_1	P_2	P_1	P_3

0 1 5 11 19

1)

P_1	P_2	P_3

0 7 13 21

✓2)

P_1	P_2	P_1	P_3

0 1 5 11 19

3)

P_1	P_2	P_3

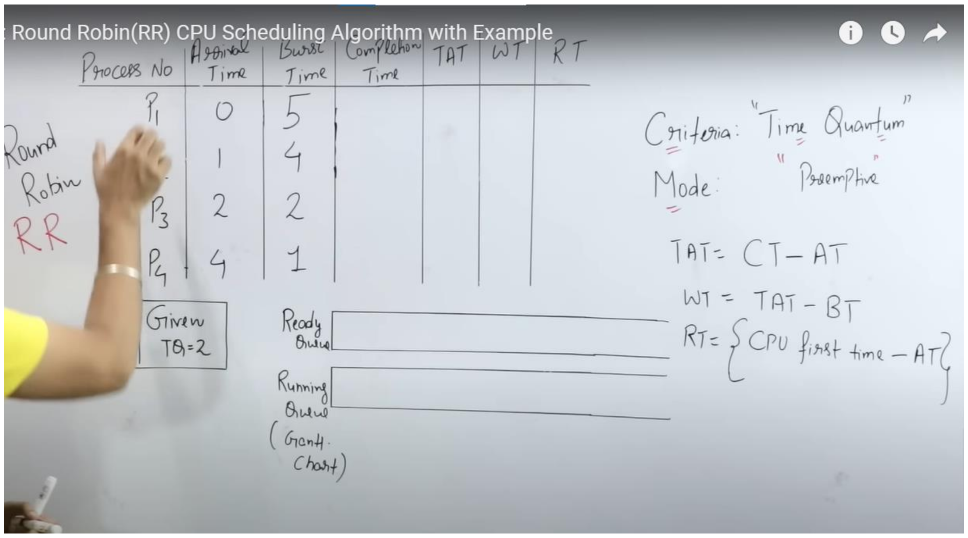
0 7 11 19

4)

P_2	P_3	P_1

0 4 12 19

L-2.7: Round Robin(RR) CPU Scheduling Algorithm with Example



Round Robin(RR) CPU Scheduling Algorithm with Example

Process No	Arrival Time	Burst Time	Completion Time	TAT	WT	RT
P_1	0	5				
	1	4				
P_3	2	2				
P_4	4	1				

Round Robin RR

Given TQ=2

Ready Queue

Running Queue (Gantt Chart)

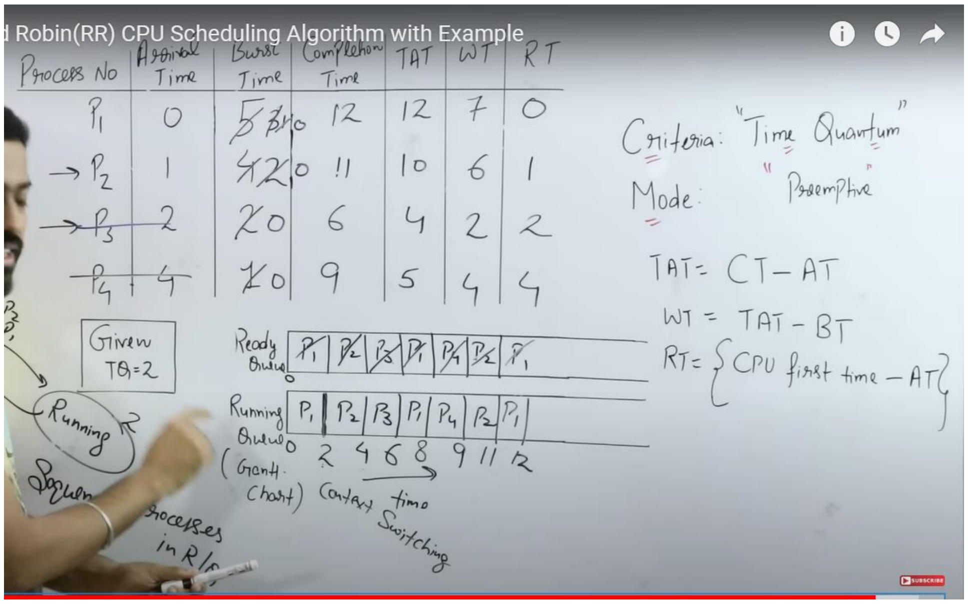
Criteria: "Time Quantum"

Mode: "Preemptive"

TAT = CT − AT

WT = TAT − BT

RT = {CPU first time − AT}



Robin(RR) CPU Scheduling Algorithm with Example

Process No	Arrival Time	Burst Time	Completion Time	TAT	WT	RT
P_1	0	~~5~~ ~~3~~ ~~1~~ 0	12	12	7	0
$\rightarrow P_2$	1	~~4~~ ~~2~~ 0	11	10	6	1
$\rightarrow$ ~~P_3~~	~~2~~	~~2~~ 0	6	4	2	2
~~P_4~~	~~4~~	~~1~~ 0	9	5	4	4

Given TQ=2

Ready Queue: ~~P_1~~ | ~~P_2~~ | ~~P_3~~ | ~~P_1~~ | ~~P_4~~ | ~~P_2~~ | ~~P_1~~

Running Queue (Gantt Chart): P_1 | P_2 | P_3 | P_1 | P_4 | P_2 | P_1

0 2 4 6 8 9 11 12

time

Context Switching

Running

processes in R/Q

Criteria: "Time Quantum"

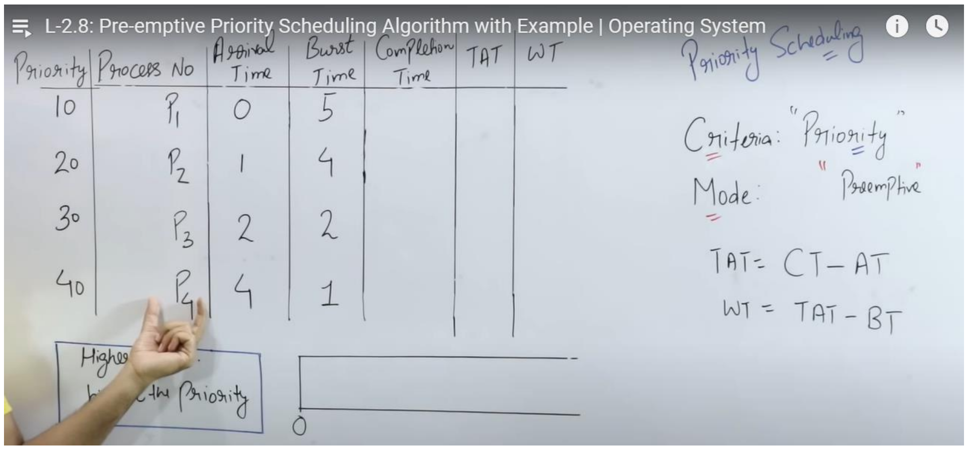
Mode: "Preemptive"

TAT = CT − AT

WT = TAT − BT

RT = {CPU first time − AT}

L-2.8: Pre-emptive Priority Scheduling Algorithm with Example | Operating System



L-2.8: Pre-emptive Priority Scheduling Algorithm with Example | Operating System

Priority Scheduling

Priority	Process No	Arrival Time	Burst Time	Completion Time	TAT	WT
10	P_1	0	5			
20	P_2	1	4			
30	P_3	2	2			
40	P_4	4	1			

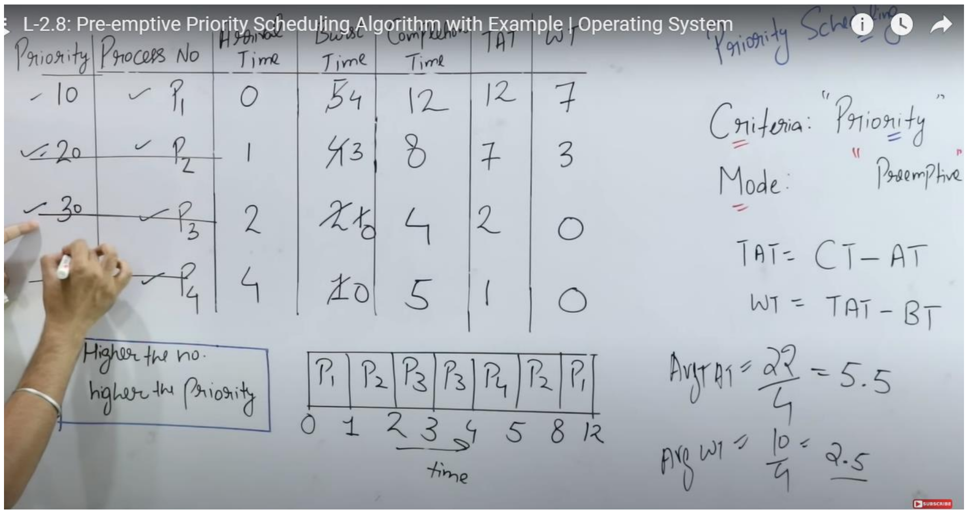
Criteria: "Priority"

Mode: "Preemptive"

TAT = CT − AT

WT = TAT − BT

Higher [illegible] the Priority



L-2.8: Pre-emptive Priority Scheduling Algorithm with Example | Operating System

Priority Scheduling

Priority	Process No	Arrival Time	Burst Time	Completion Time	TAT	WT
10	P_1	0	~~5~~ 4	12	12	7
~~20~~	~~P_2~~	1	~~4~~ 3	8	7	3
~~30~~	~~P_3~~	2	~~2~~ ~~1~~ 0	4	2	0
	~~P_4~~	4	~~1~~ 0	5	1	0

Criteria: "Priority"

Mode: "Preemptive"

TAT = CT − AT

WT = TAT − BT

Higher the no. higher the Priority

P_1	P_2	P_3	P_3	P_4	P_2	P_1

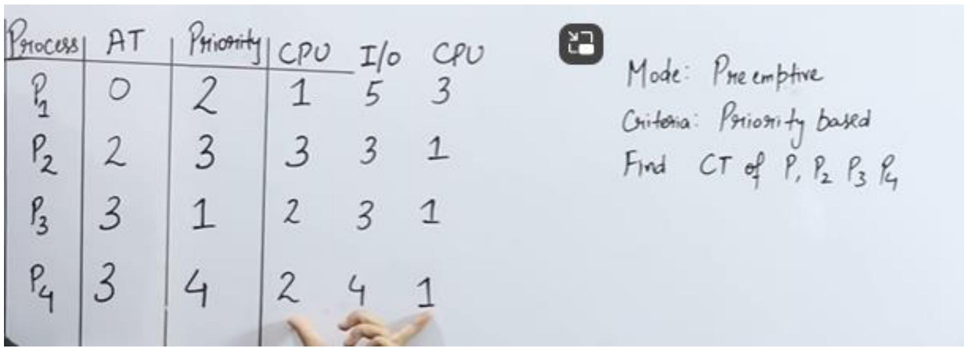
0 1 2 3 4 5 8 12

$\xrightarrow{\text{time}}$

$\text{Avg TAT} = \frac{22}{4} = 5.5$

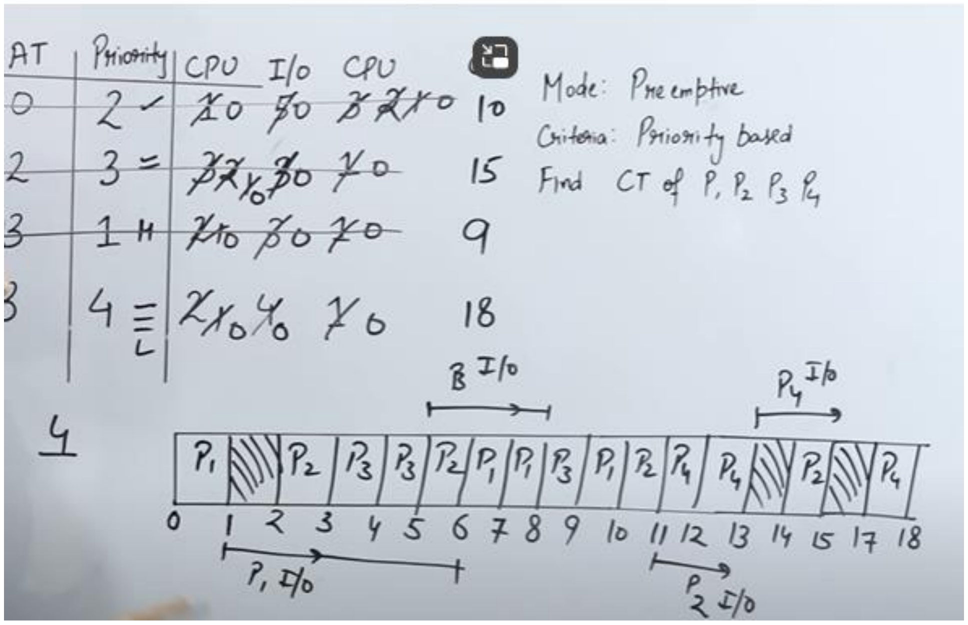
$\text{Avg WT} = \frac{10}{4} = 2.5$

L-2.9: Example of Mix Burst Time(CPU & I/O both) in CPU Scheduling | Tough Question



Process	AT	Priority	CPU	I/O	CPU
P_1	0	2	1	5	3
P_2	2	3	3	3	1
P_3	3	1	2	3	1
P_4	3	4	2	4	1

Mode: Preemptive
Criteria: Priority based
Find CT of P_1 P_2 P_3 P_4



AT	Priority	CPU	I/O	CPU	
~~0~~	~~2~~ ✓	~~1~~ 0	~~5~~ 0	~~3~~ ~~2~~ ~~1~~ 0	10
~~2~~	~~3~~ =	~~3~~ ~~2~~ ~~1~~ 0	~~3~~ 0	~~1~~ 0	15
~~3~~	~~1~~ H	~~2~~ ~~1~~ 0	~~3~~ 0	~~1~~ 0	9
3	4 ≡	~~2~~ ~~1~~ 0	~~4~~ 0	~~1~~ 0	18

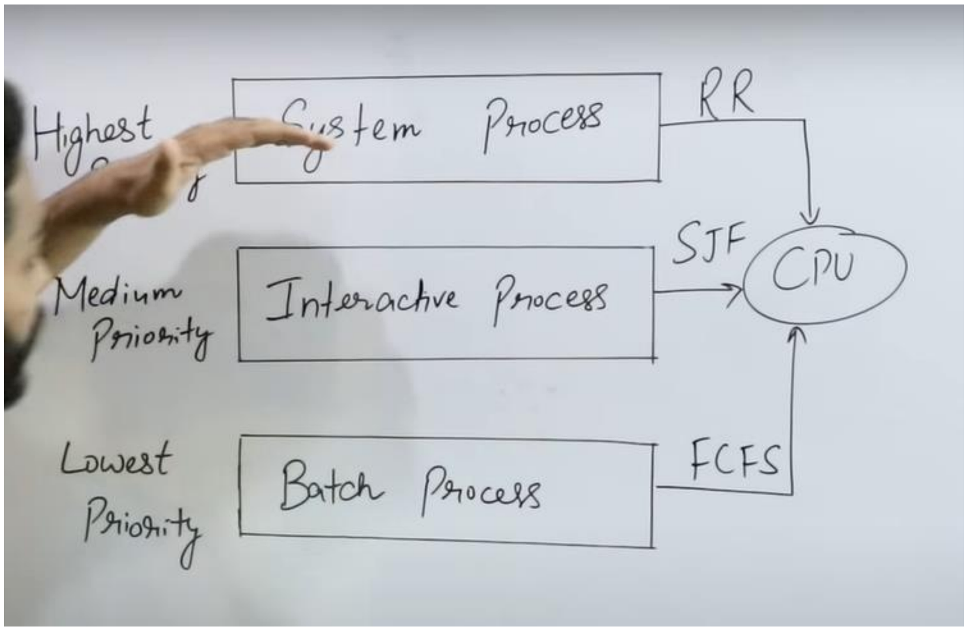
Mode: Preemptive
Criteria: Priority based
Find CT of P_1 P_2 P_3 P_4

4
P1 | ▨ | P2 | P3 | P3 | P2 | P1 | P1 | P3 | P1 | P2 | P4 | P4 | ▨ | P2 | ▨ | P4
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 17 18
P3 I/O
P4 I/O
P1 I/O
P2 I/O

Multi-level queue scheduling is a concept where different types of processes are categorized based on their priority, such as system processes, interactive processes, and batch processes. Each type of process is assigned its own ready queue, and each queue can have its own scheduling algorithm. For instance, system processes can use the round-robin scheduling algorithm, while interactive processes can use the shortest job first algorithm.

The advantage of using multi-level queue scheduling is that it can better handle processes of different types with varying priorities, making the system more efficient. However, there is a risk of starvation for lower priority processes if higher priority processes are constantly added to the queue. To avoid this, multi-level feedback queue scheduling can be used.

Overall, multi-level queue scheduling is a theoretical concept that does not have practical or numerical applications.



Topic: Multi-level feedback queue

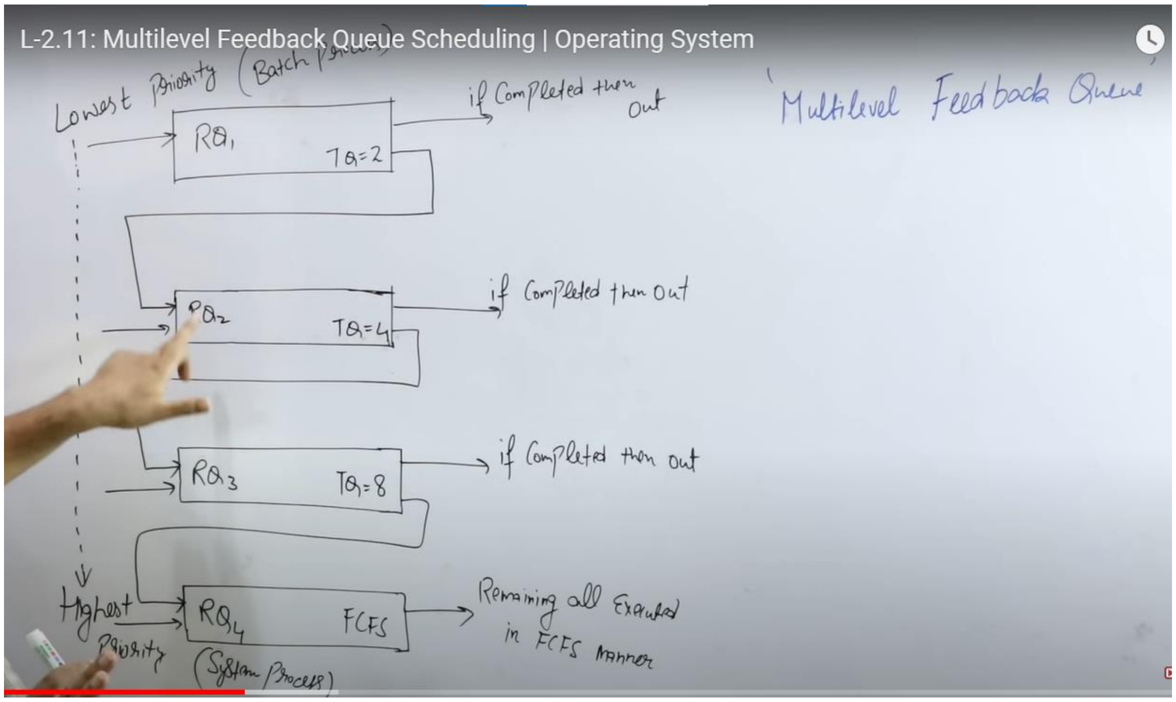
- The multi-level queue has a problem of starvation, where high priority processes like system processes can cause the lowest priority processes to wait for a long time for their turn to execute in the CPU.
- To solve the problem of starvation, feedback is used, where the lowest priority process gives feedback and is gradually upgraded to higher priority queues.
- There are different algorithms that can be used for upgrading and feedback, and the values may vary in numerical examples.
- The concept of multi-level feedback queue involves multiple queues for different types of processes with different priorities.When a process enters the lowest priority queue, it waits for its turn to execute when the high priority processes end. However, if a new process arrives, it may take its turn, causing the lowest priority process to wait longer.
- Feedback is given to the lowest priority process, and it is upgraded to higher priority queues to get the CPU faster. This process is repeated until the process completes its execution.
- High priority processes do not require feedback, and they execute using the chosen algorithm directly.
- Multi-level feedback queue solves the problem of starvation for lower priority processes, ensuring they get their turn faster, and there is no waiting for a long time.

Example:

- Suppose a batch process of lowest priority (P1) has an execution time of 19 units. It enters the lowest priority queue and needs to wait for its turn.

- The process is given a time quantum of 2, and after running for 2 units, it is upgraded to a higher priority queue to get the CPU faster. It now has 17 units left.
- The process is again upgraded after running for 4 units in the second queue and has 13 units left.
- It is then upgraded again after running for 8 units in the third queue and has 5 units left.
- The leftover 5 units are executed using the first come first serve algorithm in the lowest priority queue.

Summary: Multi-level feedback queue solves the problem of starvation for lower priority processes by using feedback to upgrade the lowest priority process to higher priority queues until it completes execution. High priority processes do not require feedback and execute using the chosen algorithm directly.


L-2.11: Multilevel Feedback Queue Scheduling | Operating System
Lowest Priority (Batch Process)
RQ1
TQ=2
if Completed then out
Multilevel Feed back Queue
RQ2
TQ=4
if Completed then out
RQ3
TQ=8
if Completed then out
Highest Priority (System Process)
RQ4
FCFS
Remaining all Executed in FCFS manner