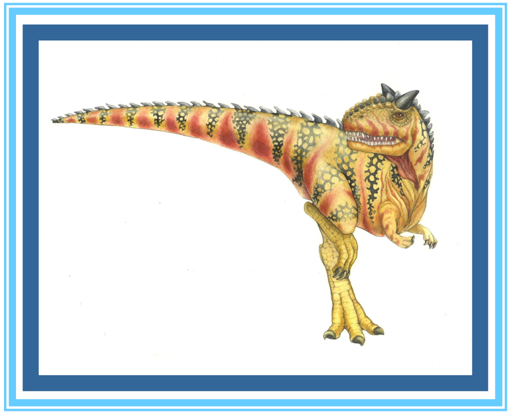
Lecture 5: CPU Scheduling



Outline

- Basic Concepts
- Why CPU Scheduling
- Preemptive Vs Non Preemptive Scheduling
- First Come first Serve
- Shortest Job First
- Priority Scheduling
- Round robin Scheduling

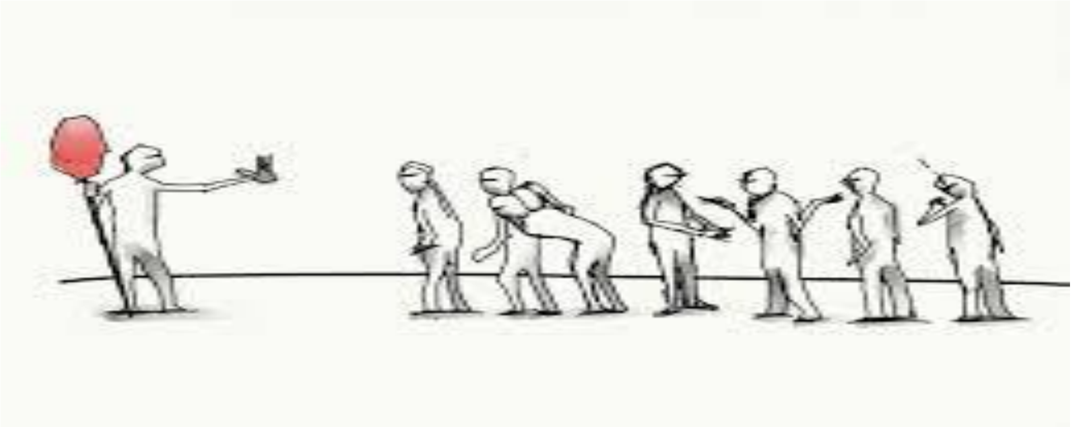
Why CPU scheduling?

In a single processor system, only one process can run at a time.

Any others must wait until CPU is free and can be rescheduled.

The objective of multiprogramming is to have some processes running at all times, to maximize CPU utilization.

CPU waits for the completion of IO request. All this waiting time is Waste of CPU resources, no useful work is accomplished.



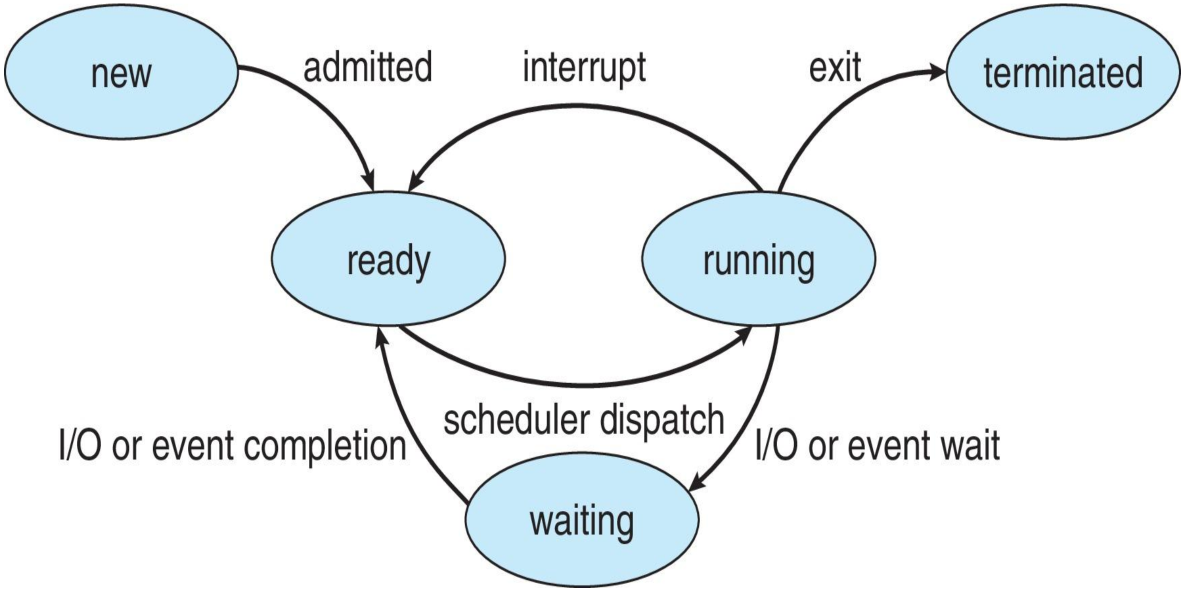
With Multiprogramming, we try to use this time productively.

Several processes are kept in memory at one time.

When one process has to wait, the operating system take the CPU away from that process and gives the CPU to another process and this pattern continues.

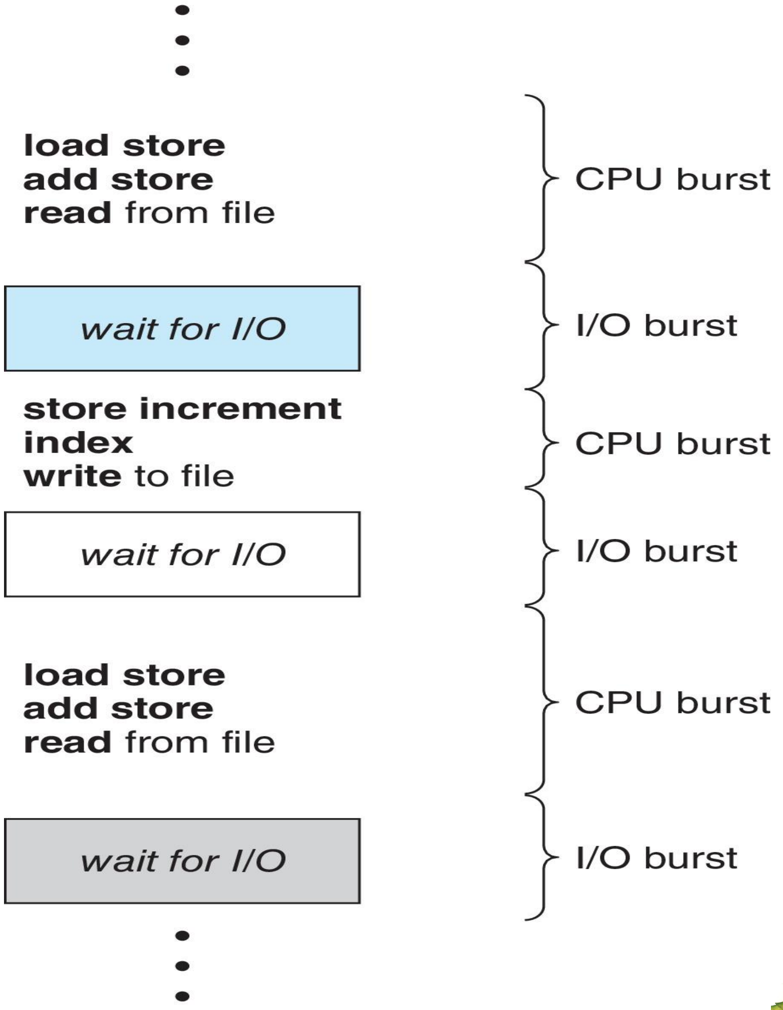
CPU Scheduling

- CPU scheduling is the basis of multiprogrammed operating system.
- By switching the CPU among processes, the OS can make the computer more productive.
- We will discuss various CPU scheduling algorithms.



Lets get the basics right

- Maximum CPU utilization obtained with multiprogramming
- CPU–I/O Burst Cycle – Process execution consists of a **cycle** of CPU execution and I/O wait
- **CPU burst** followed by **I/O burst**
- CPU burst distribution is of main concern
- During an I/O burst, a process or program waits for data to be read from or written to external storage devices, such as disks or network resources. I/O bursts are common in scenarios where data needs to be fetched from or delivered to external sources.



Types of CPU scheduling

Preemptive and non-Preemptive scheduling

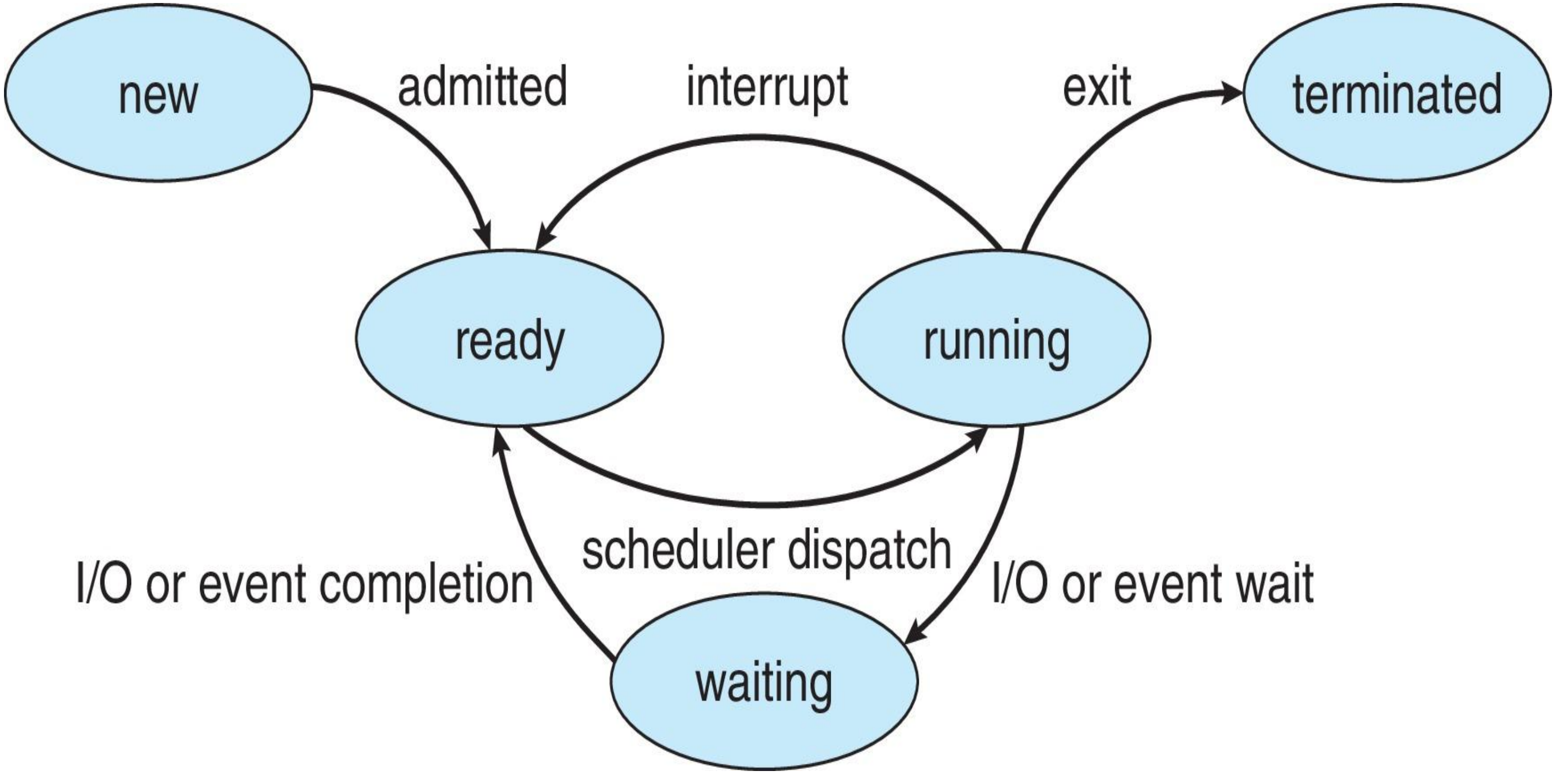
Preemptive scheduling:

The running process can be interrupted and must release the CPU (can be forced to give up CPU)

Non-preemptive scheduling:

The running process keeps the CPU until it voluntarily gives up the CPU

Revisit the process diagram



CPU Scheduler

- The **CPU scheduler** selects from among the processes in ready queue, and allocates a CPU core to one of them
- CPU scheduling decisions may take place in these four cases:

  1. Switches from running to waiting state
  2. Switches from running to ready state (interrupt occurs)
  3. Switches from waiting to ready (completion of IO)
  4. Process Terminates

- For situations 1 and 4, there is no choice in terms of scheduling. A new process (if one exists in the ready queue) must be selected for execution.
- For situations 2 and 3, however, there is a choice.

- Multiple processes are in ready state/queue and any of them can be scheduled based on the scheduling algorithm.
- Under Nonpreemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases it either by terminating or by switching to the waiting state.
- **Virtually all modern operating systems including Windows, MacOS, Linux, and UNIX use preemptive scheduling algorithms.**

Scheduling Criteria

- **CPU utilization** – keep the CPU as busy as possible
- **Throughput** – number of processes that complete their execution per time unit.
- **Turnaround time** – Interval from time of submission of a process to the time of completion of the same process.
- **Waiting time** – amount of time a process has been waiting in the ready queue

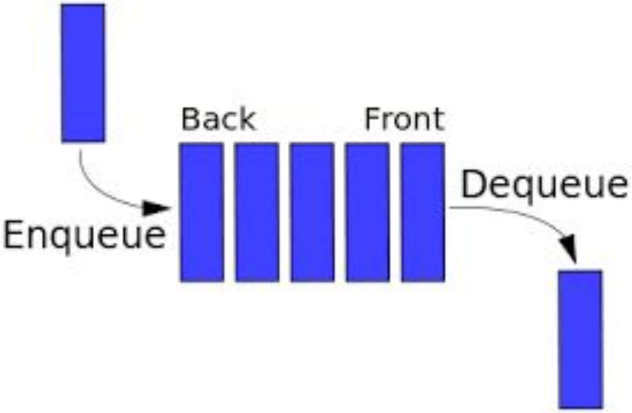
- Turnaround time =Completion time−Arrival Time
- Waiting time= Turnaroundtime – Burst time

Scheduling Algorithm Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time

First Come First Serve Scheduling

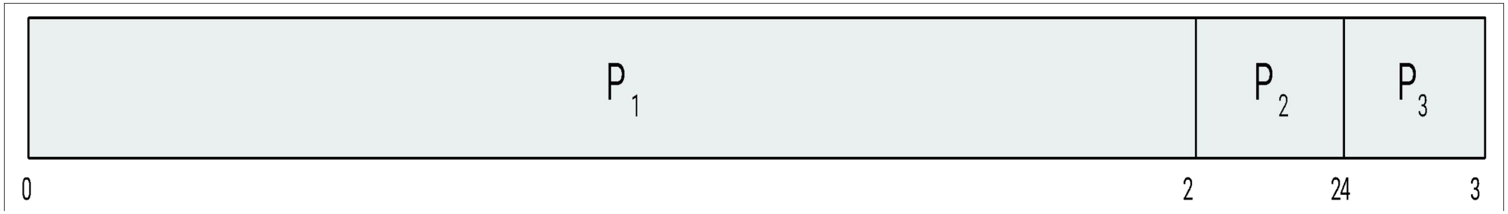
- Simplest scheduling algorithm.
- The process that requests the CPU first is allocated the CPU first.
- FCFS easily managed with a FIFO queue.



The average waiting time in FIFO is long

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3
The Gantt Chart for the schedule is:



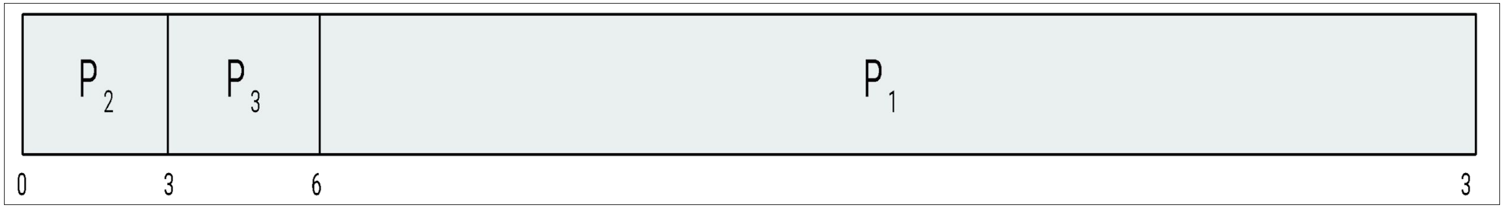
- Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
- Average waiting time: (0 + 24 + 27)/3 = 17

Lets change the order of arrival of process from previous example

Suppose that the processes arrive in the order:

P_2 , P_3 , P_1

- The Gantt chart for the schedule is:



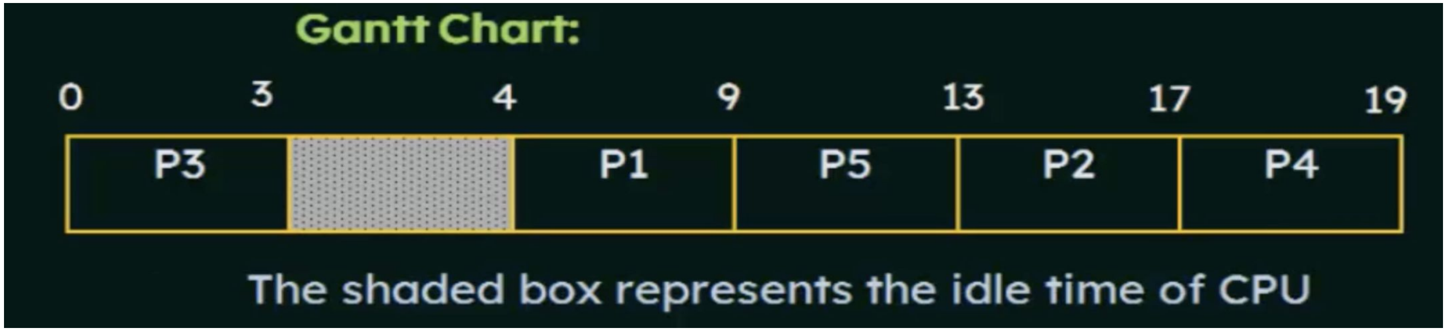
- Waiting time for $P_1 = 6$; $P_2 = 0$; $P_3 = 3$
- Average waiting time: (6 + 0 + 3)/3 = 3
- Much better than previous case
- **Convoy effect** - short process behind long process. So that smaller burst time process don’t have to wait.

Example 2 FCFS Algo

Consider set of 5 processes. Calculate the average waiting time and average turnaround time, if FCFS scheduling is followed.

Process ID	Arrival Time	Burst Time
P1	4	5
P2	6	4
P3	0	3
P4	6	2
P5	5	4

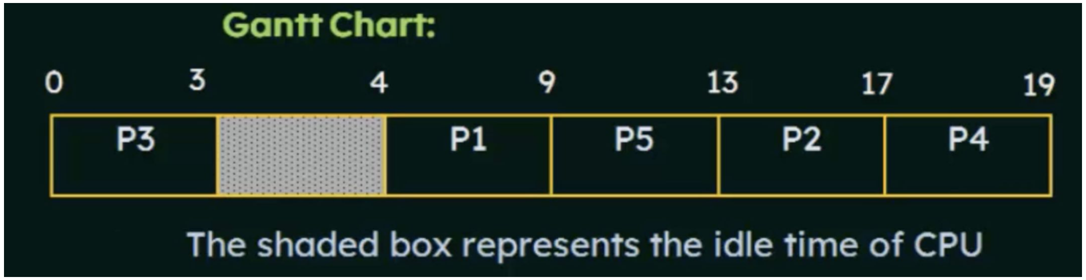
Gantt Chart:



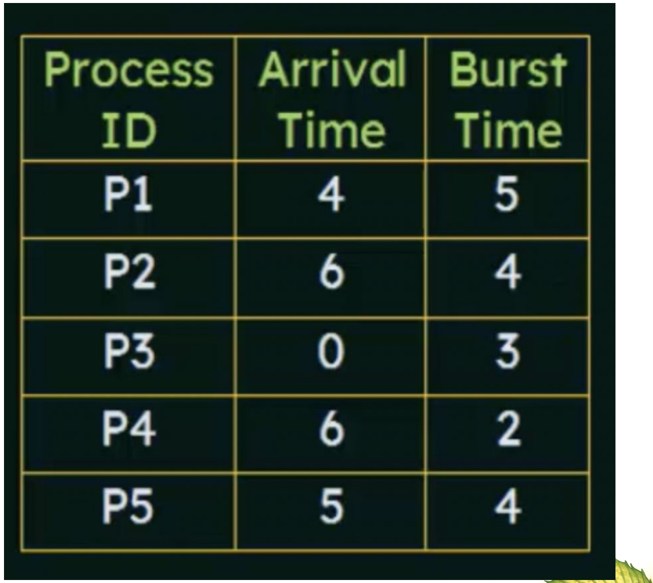
The shaded box represents the idle time of CPU

Turnaround time and waiting time

- Turnaround time = Completion time – Arrival Time
- P1 completed at 9th unit of time and arrived at 4th unit of time.

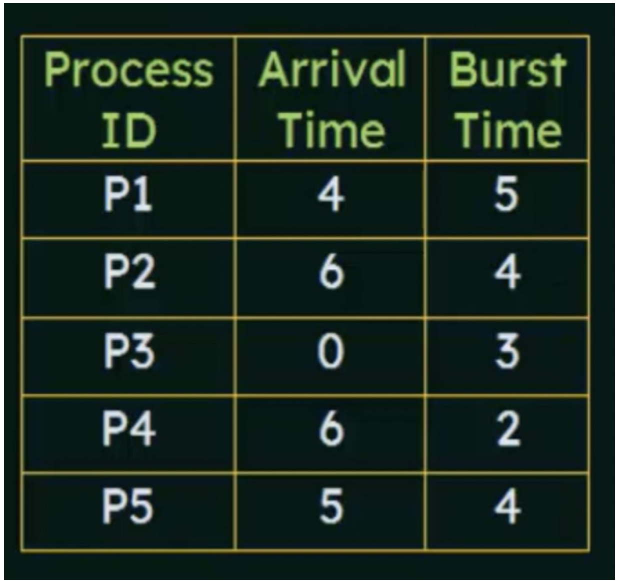


- Waiting time= Turnaroundtime – Burst time
- Burst time is the time the process actually needs.



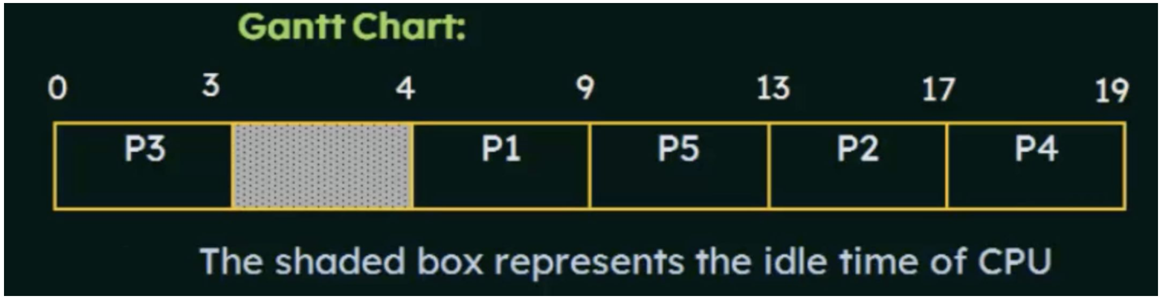
Process ID	Arrival Time	Burst Time
P1	4	5
P2	6	4
P3	0	3
P4	6	2
P5	5	4

Turnaround time and waiting time

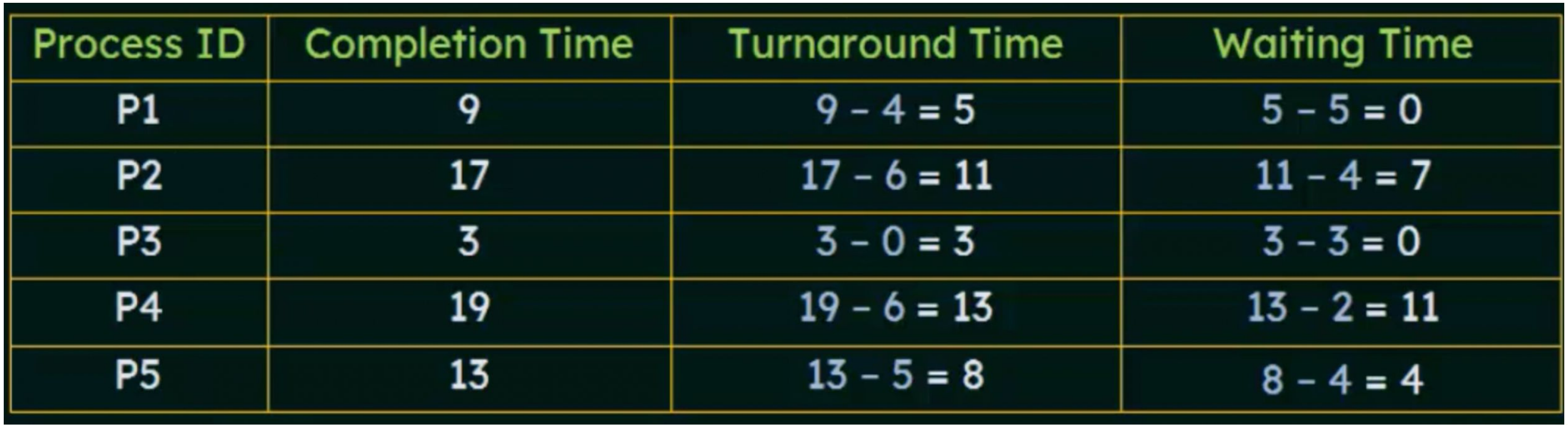


Process ID	Arrival Time	Burst Time
P1	4	5
P2	6	4
P3	0	3
P4	6	2
P5	5	4

Turnaround time =Completion time–Arrival Time
Waiting time= Turnaroundtime – Burst time



The shaded box represents the idle time of CPU

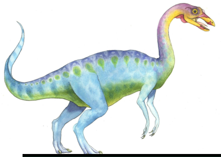


Process ID	Completion Time	Turnaround Time	Waiting Time
P1	9	9 - 4 = 5	5 - 5 = 0
P2	17	17 - 6 = 11	11 - 4 = 7
P3	3	3 - 0 = 3	3 - 3 = 0
P4	19	19 - 6 = 13	13 - 2 = 11
P5	13	13 - 5 = 8	8 - 4 = 4

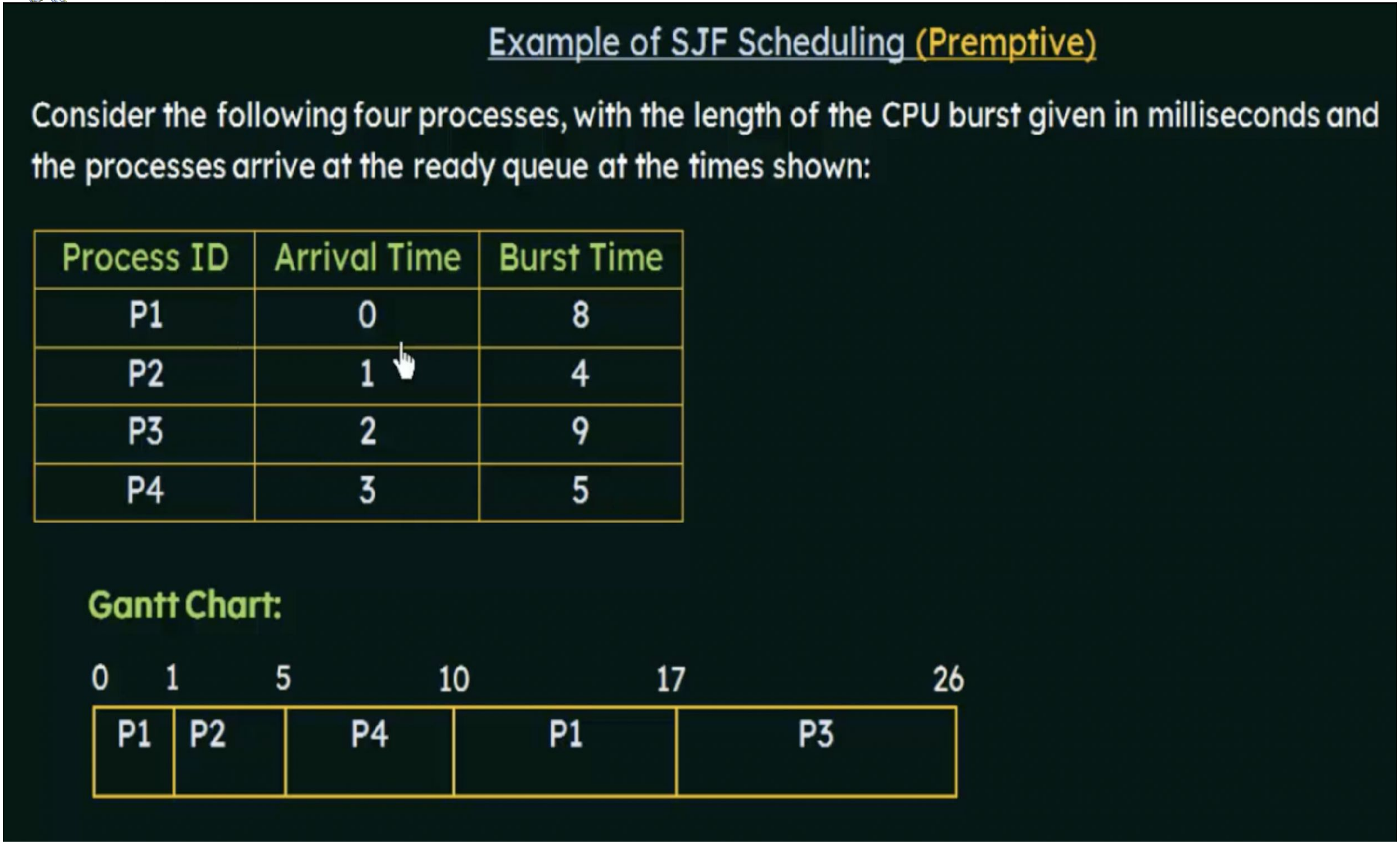
- Average Turnaround time= (5+11+3+13+8)/5 = 40/5= 8 units
- Average waiting time = (0+7+0+11+4)/5 = 22/5 = 4.4 units

Shortest-Job-First (SJF) Scheduling

- When the CPU is available, assigned to process that has the smallest next CPU burst.
- SJF is optimal – gives minimum average waiting time for a given set of processes. How?
- If the next CPU burst of two processes are the same, FCFS scheduling is used to break the tie.
- SJF can be preemptive or non-preemptive.



SJF(Preemptive)



Example of SJF Scheduling (Premptive)

Consider the following four processes, with the length of the CPU burst given in milliseconds and the processes arrive at the ready queue at the times shown:

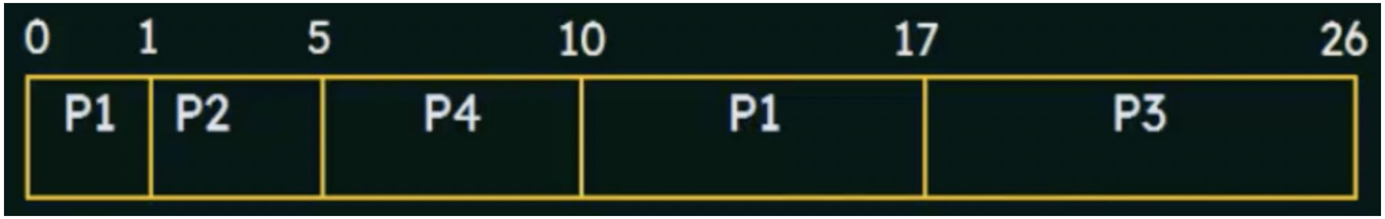
Process ID	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	9
P4	3	5

Gantt Chart:

0 1 5 10 17 26

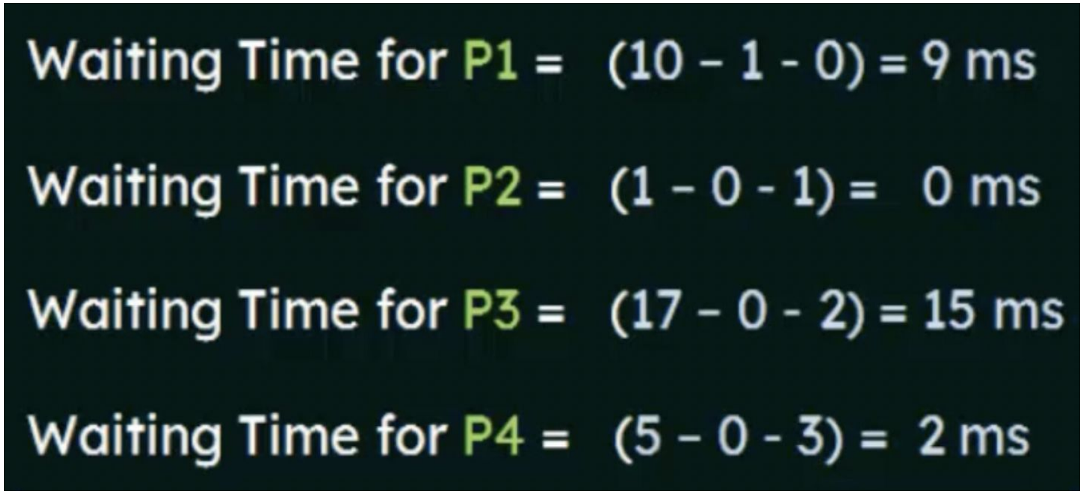
P1	P2	P4	P1	P3

Average waiting time of SJF?

- Waiting time=
- Total waiting time - no. of milliseconds process executed - Arrival time.
- 

0 1 5 10 17 26

P1	P2	P4	P1	P3



Waiting Time for P1 = (10 – 1 - 0) = 9 ms

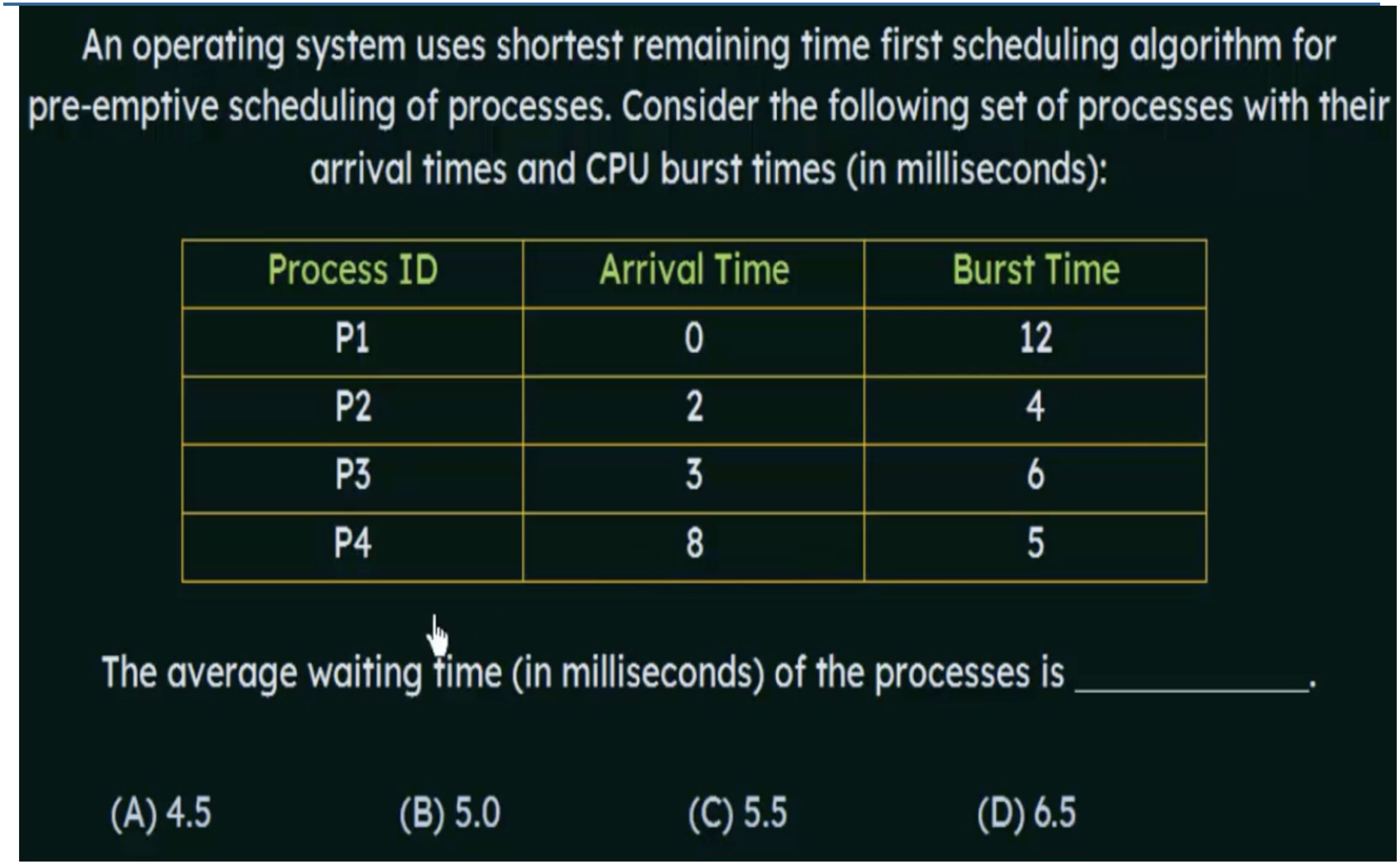
Waiting Time for P2 = (1 – 0 - 1) = 0 ms

Waiting Time for P3 = (17 – 0 - 2) = 15 ms

Waiting Time for P4 = (5 – 0 - 3) = 2 ms

Average waiting time= (9+0+15+2)/4 = 6.5

Try yourself



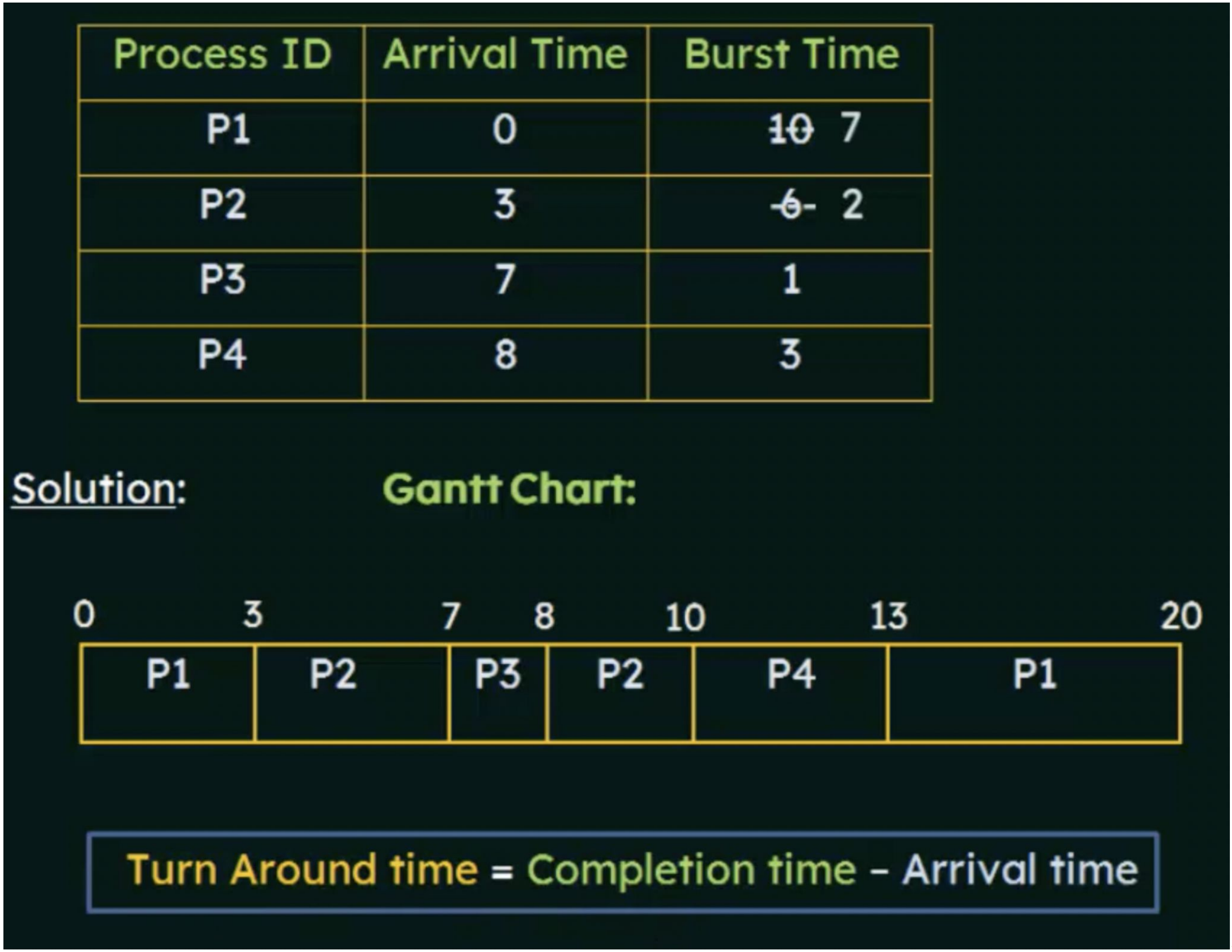
An operating system uses shortest remaining time first scheduling algorithm for pre-emptive scheduling of processes. Consider the following set of processes with their arrival times and CPU burst times (in milliseconds):

Process ID	Arrival Time	Burst Time
P1	0	12
P2	2	4
P3	3	6
P4	8	5

The average waiting time (in milliseconds) of the processes is ____________.

(A) 4.5 (B) 5.0 (C) 5.5 (D) 6.5

Example: Calculate Turnaround Time



Process ID	Arrival Time	Burst Time
P1	0	~~10~~ 7
P2	3	~~6~~ 2
P3	7	1
P4	8	3

Solution:

Gantt Chart:

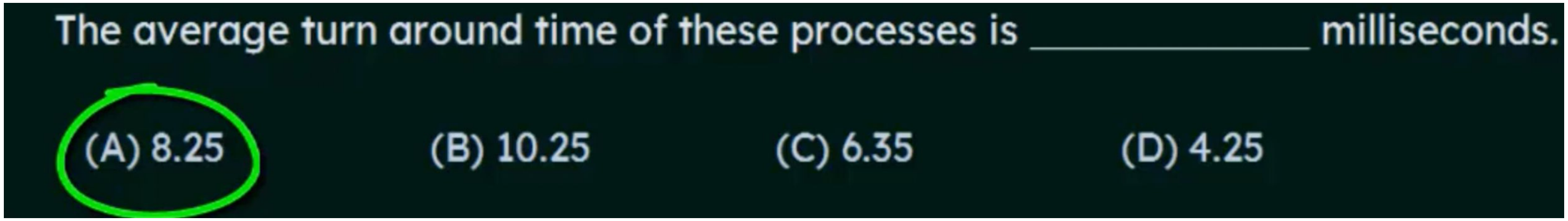
0 3 7 8 10 13 20

P1	P2	P3	P2	P4	P1

Turn Around time = Completion time – Arrival time

Turnaround time for the example?

- Calculate the Turnaround time for all the processes and find the average TAT.



The average turn around time of these processes is ______________ milliseconds.

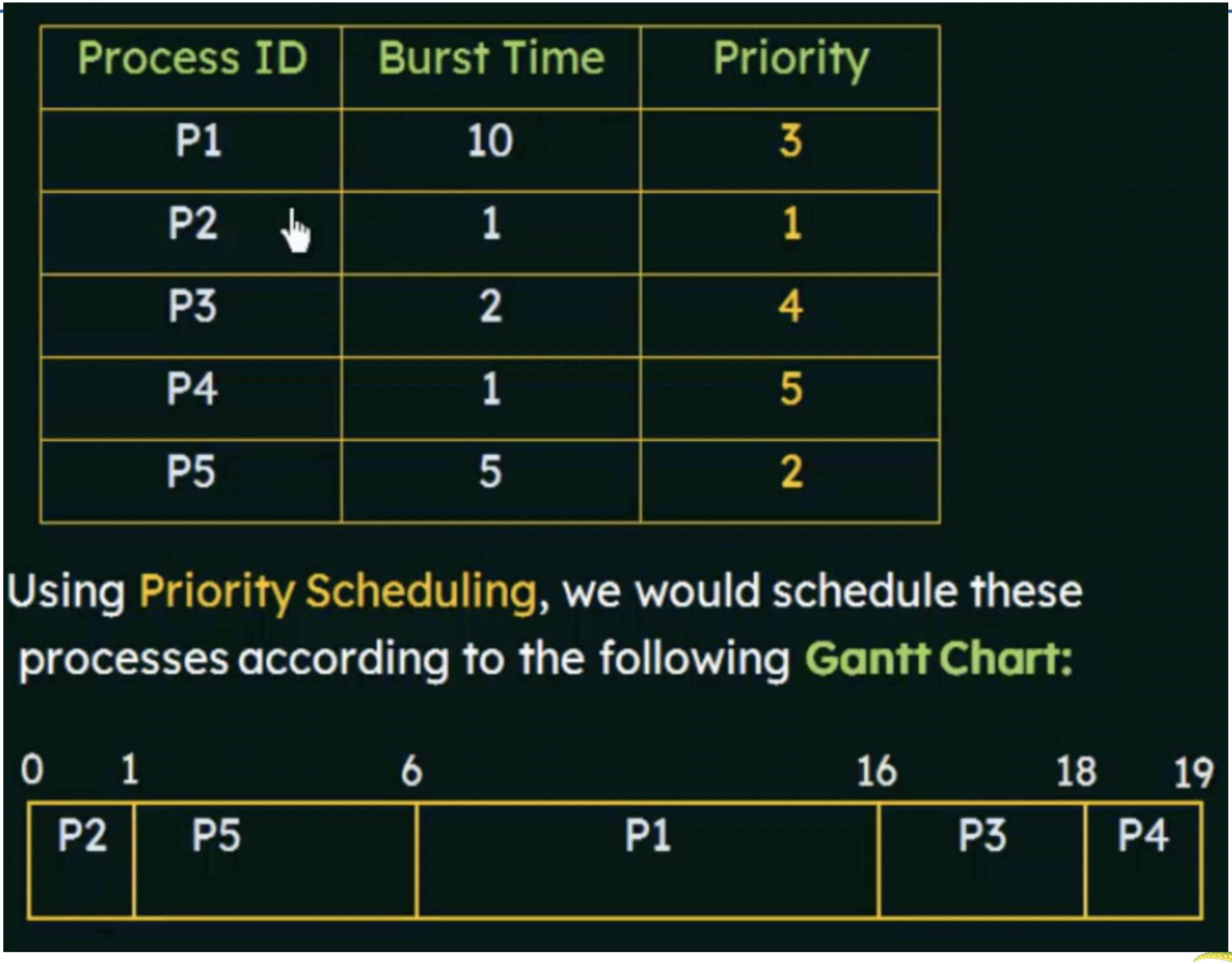
[✓ (A) 8.25] (B) 10.25 (C) 6.35 (D) 4.25

Priority Scheduling

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority)
  - Preemptive
  - Non- preemptive
- SJF is simply a priority scheduling where priority is the inverse of predicted next CPU burst time.
- The larger the CPU burst, the lower the priority, and vice versa.

- Preemptive priority scheduling algorithm will preempt the CPU if the priority of the newly arrived process is higher than the priority of the currently running process.
- Non-preemptive priority scheduling will simply put the process at the head of the ready queue.

Priority Scheduling example

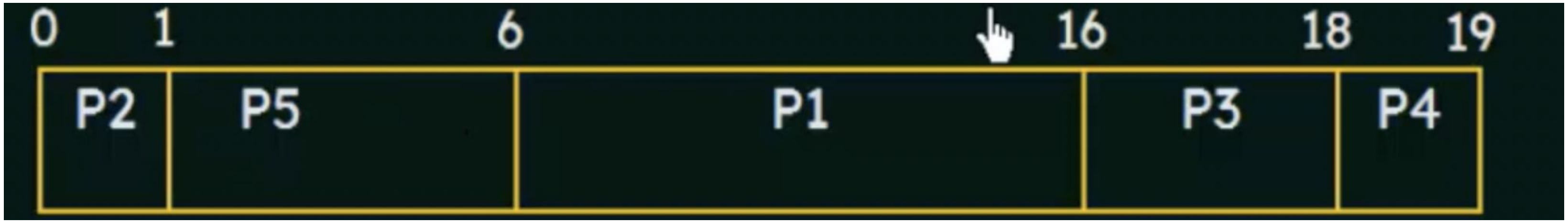


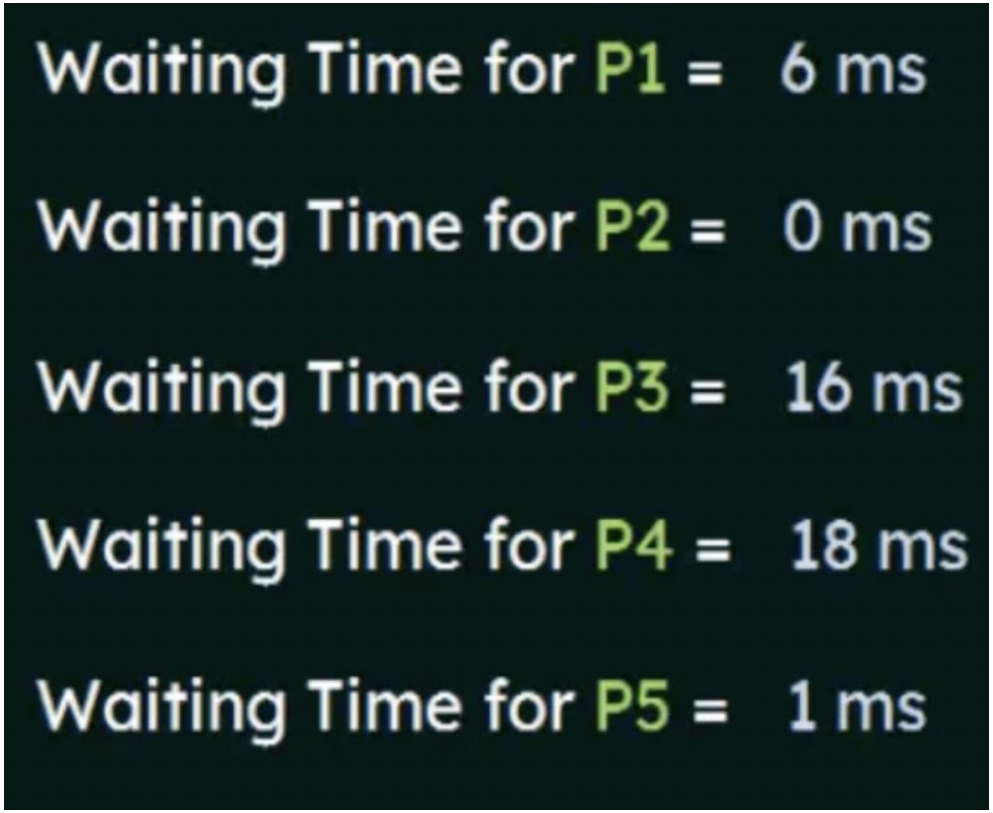
Process ID	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

Using Priority Scheduling, we would schedule these processes according to the following Gantt Chart:

0 1 6 16 18 19

P2 P5 P1 P3 P4





Waiting Time for P1 = 6 ms

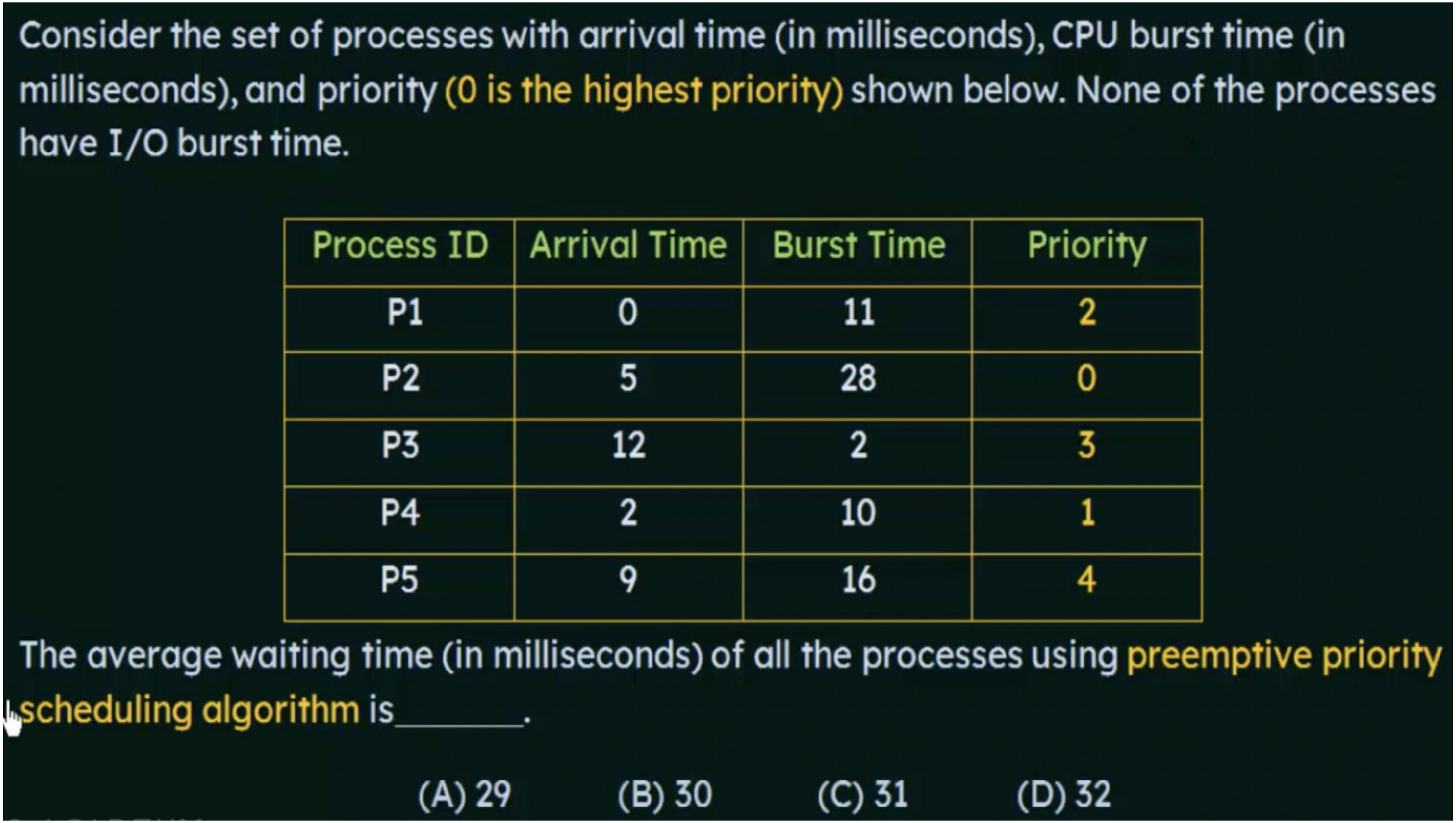
Waiting Time for P2 = 0 ms

Waiting Time for P3 = 16 ms

Waiting Time for P4 = 18 ms

Waiting Time for P5 = 1 ms

Example 2



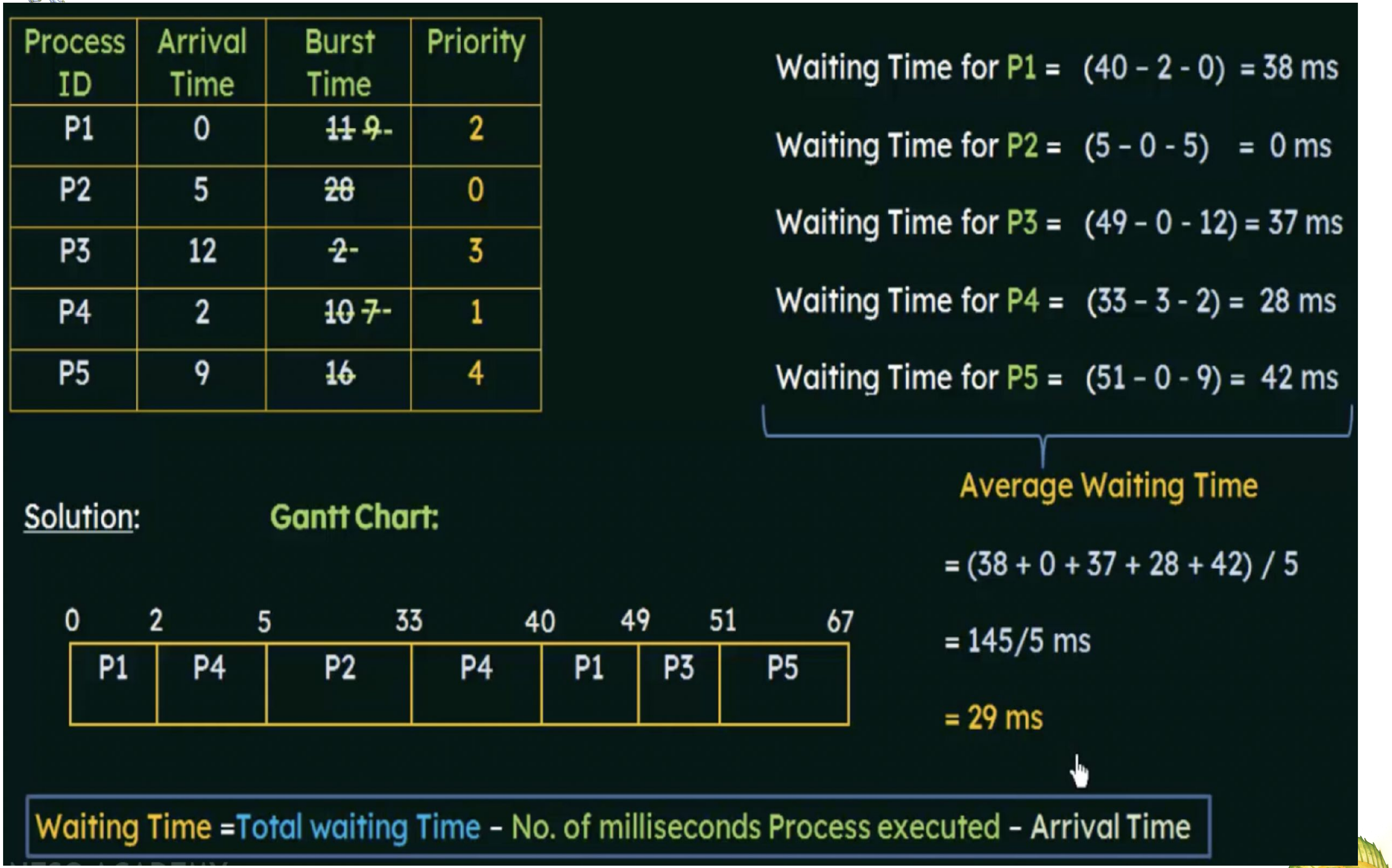
Consider the set of processes with arrival time (in milliseconds), CPU burst time (in milliseconds), and priority (0 is the highest priority) shown below. None of the processes have I/O burst time.

Process ID	Arrival Time	Burst Time	Priority
P1	0	11	2
P2	5	28	0
P3	12	2	3
P4	2	10	1
P5	9	16	4

The average waiting time (in milliseconds) of all the processes using preemptive priority scheduling algorithm is_______.

(A) 29 (B) 30 (C) 31 (D) 32

Solution



Process ID	Arrival Time	Burst Time	Priority
P1	0	~~11~~ ~~9~~	2
P2	5	~~28~~	0
P3	12	~~2~~	3
P4	2	~~10~~ ~~7~~	1
P5	9	~~16~~	4

Waiting Time for P1 = (40 – 2 - 0) = 38 ms

Waiting Time for P2 = (5 – 0 - 5) = 0 ms

Waiting Time for P3 = (49 – 0 - 12) = 37 ms

Waiting Time for P4 = (33 – 3 - 2) = 28 ms

Waiting Time for P5 = (51 – 0 - 9) = 42 ms

Average Waiting Time

= (38 + 0 + 37 + 28 + 42) / 5

= 145/5 ms

= 29 ms

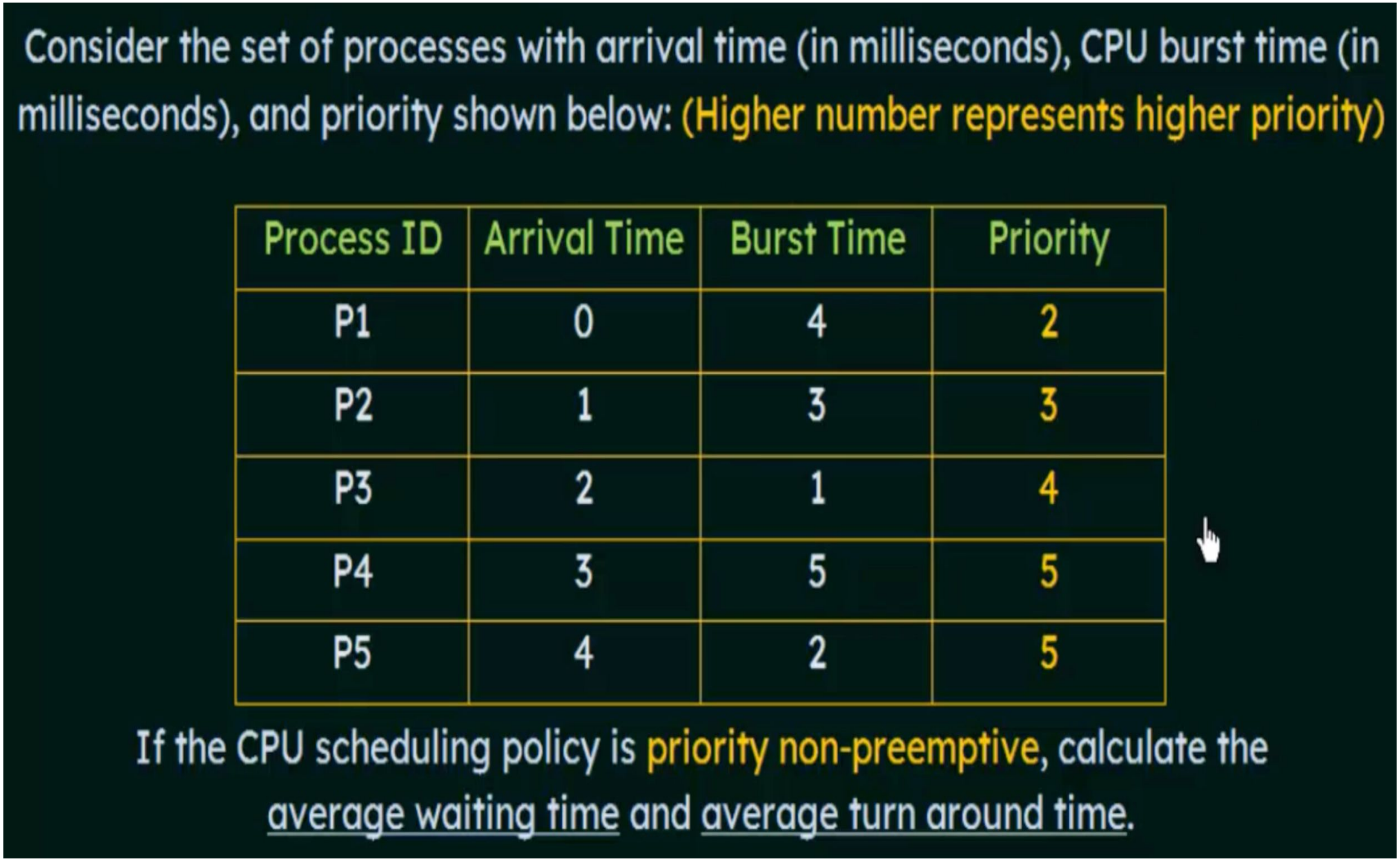
Solution:

Gantt Chart:

0	2	5	33	40	49	51	67
P1	P4	P2	P4	P1	P3	P5	

Waiting Time = Total waiting Time – No. of milliseconds Process executed – Arrival Time

Try yourself



Consider the set of processes with arrival time (in milliseconds), CPU burst time (in milliseconds), and priority shown below: (Higher number represents higher priority)

Process ID	Arrival Time	Burst Time	Priority
P1	0	4	2
P2	1	3	3
P3	2	1	4
P4	3	5	5
P5	4	2	5

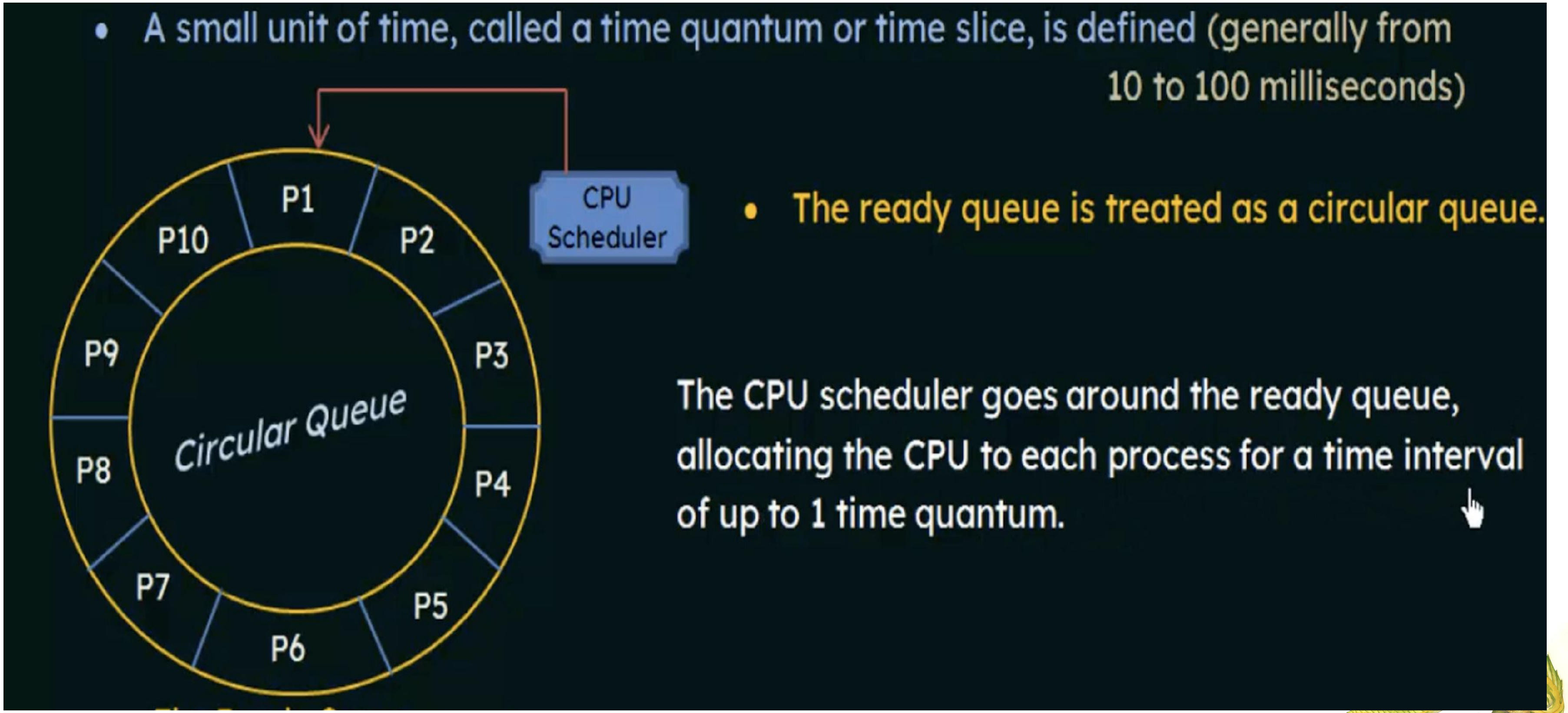
If the CPU scheduling policy is priority non-preemptive, calculate the average waiting time and average turn around time.

Problem with Priority Scheduling

- Indefinite blocking
- Leaving some low priority starved.
- Problem ≡ **Starvation** – low priority processes may never execute
- Solution ≡ **Aging** – as time progresses increase the priority of the process

Round Robin (RR) Scheduling

- Designed especially for time sharing systems.
- Similar to FCFS scheduling, but preemption is added to switch between processes



- A small unit of time, called a time quantum or time slice, is defined (generally from 10 to 100 milliseconds)

- 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 of up to 1 time quantum.