Lecture 12

ROUND ROBIN

Question

12

Suppose a system contains n processes and system uses the round-robin algorithm for CPU scheduling then which data structure is best suited ready queue of the process

A. stack
B. queue
C. circular queue
D. tree

https://gateoverflow.in/51692/isro2015-33

18

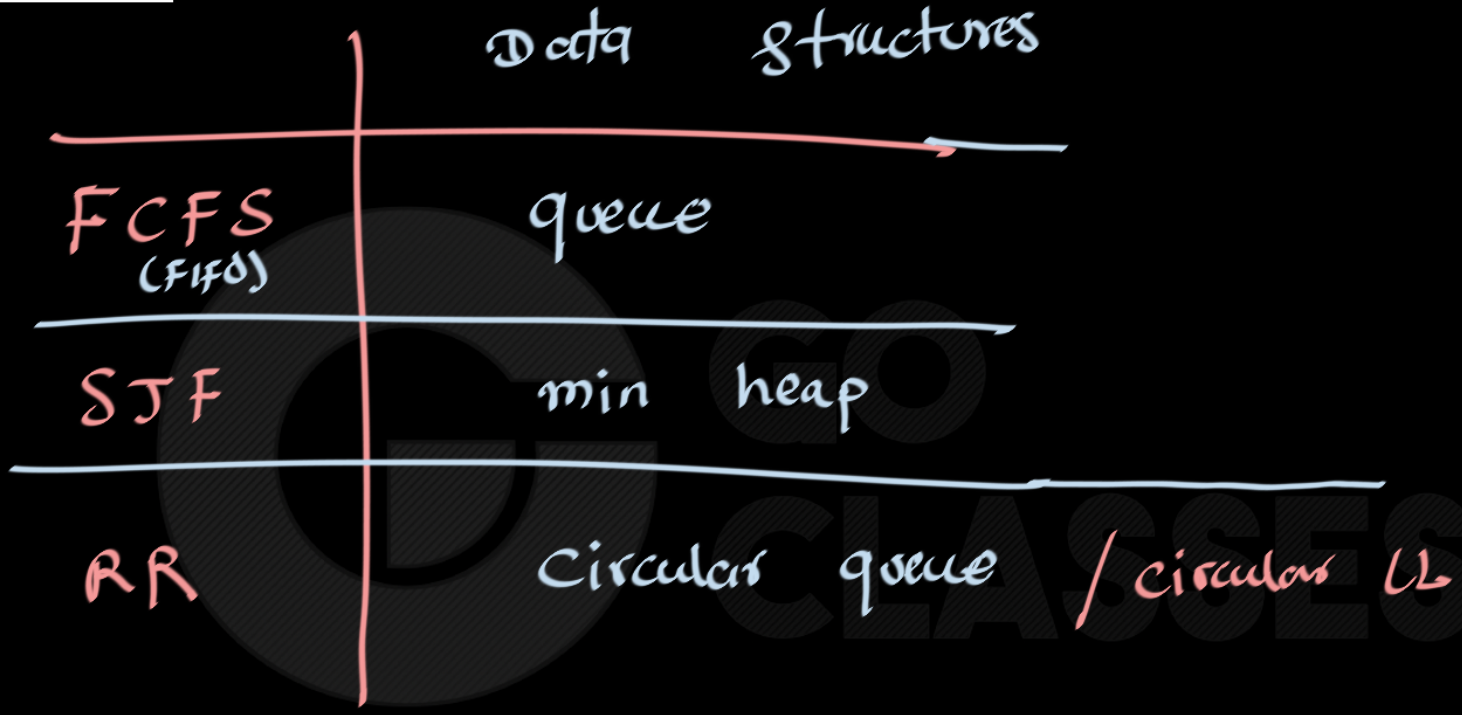
Circular queue is the best data structure for round-robin CPU scheduling algorithm .

In round-robin CPU scheduling if the timer goes off first, then the process is swapped out of the CPU and moved to the back end of the ready queue.

The ready queue is maintained as a **circular queue**, so when all processes have had a turn, then the scheduler gives the first process another turn, and so on.

Best answer

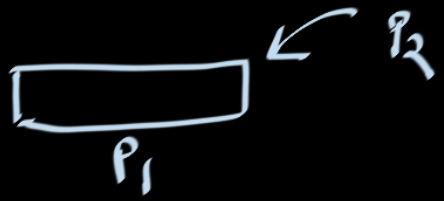
So option C is correct.


Data structures
FCFS
(FIFO)
queue
SJF
min heap
RR
Circular queue / circular LL

Question

Consider a system with two processes, P_1 and P_2. Process P_1 is running and process P_2 is ready to run. The operating system uses preemptive round robin scheduling. Consider the situation in which P_1's scheduling quantum is about to expire, and P_2 will be selected to run next.

List the sequence of events that will occur in this situation, in the order in which they will occur. Create your list by choosing from the events shown below, using the event numbers to identify events. For example, a valid answer might be: "7,5,4,8,5,2". If an event occurs more than once, you may include it more than once in your list.



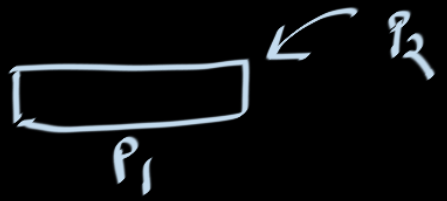
1. P_1's user-level state is saved into a trap frame
2. P_2's user-level state is saved into a trap frame
3. a timer interrupt occurs
4. the operating system's scheduler runs
5. there is a context switch from thread T_1 (from process P_1) to thread T_2 (from process P_2).
6. P_1's user-level state is restored from a trap frame
7. P_2's user-level state is restored from a trap frame
8. a "system call" instruction is executed
9. thread T_2 (for process P_2) is created
10. thread T_1 (from process P_1 is destroyed

https://student.cs.uwaterloo.ca/~cs350/common/old-exams/W08midsol.pdf

Question

Consider a system with two processes, P_1 and P_2. Process P_1 is running and process P_2 is ready to run. The operating system uses preemptive round robin scheduling. Consider the situation in which P_1's scheduling quantum is about to expire, and P_2 will be selected to run next.

List the sequence of events that will occur in this situation, in the order in which they will occur. Create your list by choosing from the events shown below, using the event numbers to identify events. For example, a valid answer might be: "7,5,4,8,5,2". If an event occurs more than once, you may include it more than once in your list.



1. P_1's user-level state is saved into a trap frame
2. P_2's user-level state is saved into a trap frame
3. a timer interrupt occurs
4. the operating system's scheduler runs
5. there is a context switch from thread T_1 (from process P_1) to thread T_2 (from process P_2).
6. P_1's user-level state is restored from a trap frame
7. P_2's user-level state is restored from a trap frame
8. a "system call" instruction is executed
9. thread T_2 (for process P_2) is created
10. thread T_1 (from process P_1 is destroyed

3, 1, 4, 5, 7

https://student.cs.uwaterloo.ca/~cs350/common/old-exams/W08midsol.pdf


fork()
assembly code
Mov Rax 57
Syscall
Ret

Write your answer here:

3,1,4,5,7

Question

throughput = ? (in RR)

context switch(each .5 units)

quanta = 3 units

	Burst time
A	5
B	3

https://www.cs.virginia.edu/~cr4bd/4414/F2019/slides/20190912--slides-1up.pdf

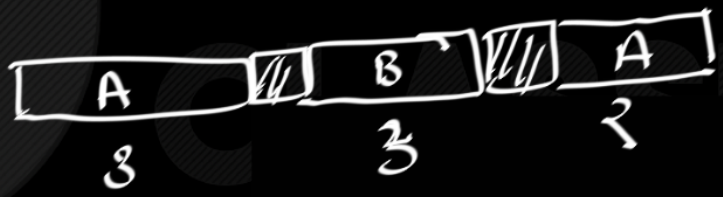
Question

throughput = ? (in RR)

context switch(each .5 units)

quanta = 3 units

	Burst time
A	5
B	3



CPU utilization $= \frac{8}{9} = 88.89\%$

throughput $= \frac{2}{3+3+2+0.5+0.5} = \frac{2}{9}$ Procs/Sec.

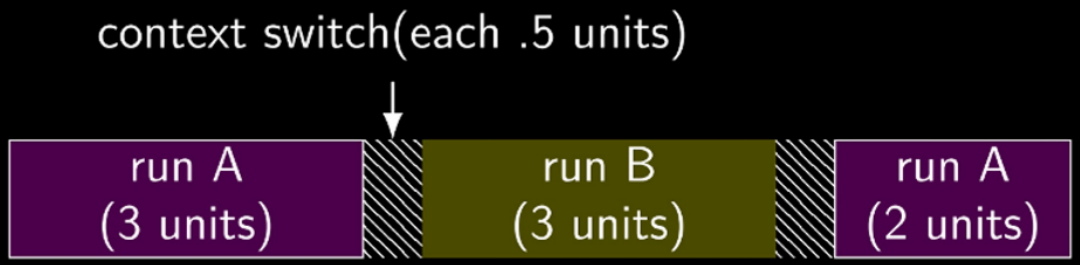
https://www.cs.virginia.edu/~cr4bd/4414/F2019/slides/20190912--slides-1up.pdf

Question

throughput = ? (in RR)

quanta = 3 units

	Burst time
A	5
B	3



throughput: useful work done per unit time

https://www.cs.virginia.edu/~cr4bd/4414/F2019/slides/20190912--slides-1up.pdf

Question True/False

1 If all jobs have identical run lengths, a RR scheduler (with a time-slice much shorter than the jobs' run lengths) provides better average turnaround time than FIFO.

2 The longer the time slice, the more a RR scheduler gives similar results to a FIFO scheduler.

https://pages.cs.wisc.edu/~dusseau/Classes/CS537/Fall2016/Exams/f16-e2-answers.pdf

4) If all jobs have identical run lengths, a RR scheduler (with a time-slice much shorter than the jobs' run lengths) provides better average turnaround time than FIFO.

False – with RR, identical jobs will all finish at nearly the same time (at the very end of the workload time), which has very poor average turnaround time.

5) The longer the time slice, the more a RR scheduler gives similar results to a FIFO scheduler.

True – In the extreme, when the time slice is >= the length of the job, RR degenerates to FIFO (or FCFS).

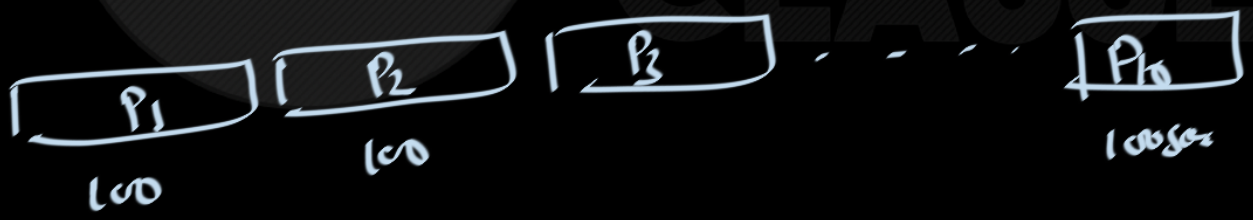
https://pages.cs.wisc.edu/~dusseau/Classes/CS537/Fall2016/Exams/f16-e2-answers.pdf

FCFS vs. Round Robin

- Example
  - 10 jobs and each takes 100 seconds
- Round Robin
  - time slice 1sec

$TAT = CT - AT$

- FCFS



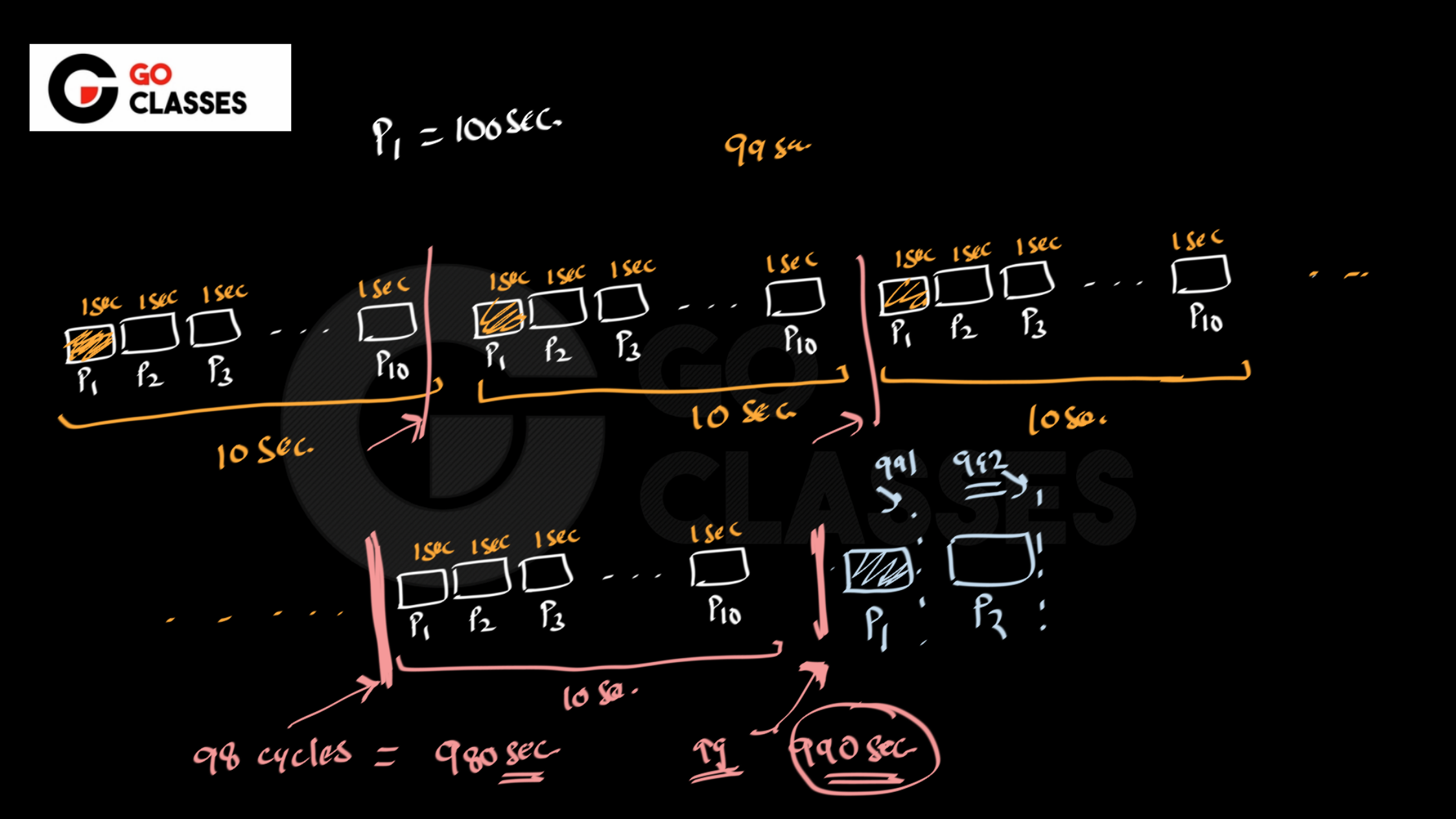
$$\text{Avg TAT FCFS} \quad \frac{100 + 200 + 300 + 400 + \cdots 1000}{10}$$

FCFS vs. Round Robin

- Example
  - 10 jobs and each takes 100 seconds
- Round Robin
  - time slice 1sec

TAT = CT − AT

when will P_1 get finished ?


GO CLASSES
P1 = 100 sec.
99 sec
1sec 1sec 1sec 1sec
P1 P2 P3 P10
10 sec.
10 sec.
10 sec.
991
992
98 cycles = 980 sec
T9
990 sec

	CT	TAT
P_1	991	→
P_2	992	→
P_3	993	→
⋮	⋮	
P_{10}	1000	→

$$\text{Avg TAT} = \frac{991 + 992 + \cdots + 1000}{10}$$

$$= 995.5$$

FCFS vs. Round Robin

- Example
  - 10 jobs and each takes 100 seconds
- FCFS (non-preemptive scheduling)
  - job 1: 100s, job2: 200s, ... , job10: 1000s

→ Avg TAT = $\frac{100 + 200 + \cdots + 1000}{10}$

- Round Robin (preemptive scheduling)
  - time slice 1sec and no overhead
  - job1: 991s, job2: 992s, ... , job10: 1000s

→ Avg TAT = $\frac{991 + 992 + \cdots + 1000}{10}$

- Comparisons
  - Round robin is much worse (turnaround time) for jobs about the same length
  - Round robin is better for short jobs

Question

WT = TAT − BT

Suppose that two processes, P_a and P_b, are running in a uniprocessor system. P_a has three threads. P_b has two threads. All threads in both processes are CPU-intensive, i.e., they never block for I/O. The operating system uses simple round-robin scheduling.

a. Suppose that all of the threads are user-level threads, and that user-level threads are implemented using a single kernel thread per process. What percentage of the processor's time will be spent running P_a's threads? For full credit, briefly justify your answer.

b. Suppose instead that all of the threads are kernel threads. What percentage of the processor's time will be spent running P_a's threads? For full credit, briefly justify your answer.

we already solved in last class.

https://trg-cc.com/portfolio/uw-cs/cs350-notes/midsol.pdf

Suppose that two processes, P_a and P_b, are running in a uniprocessor system. P_a has three threads. P_b has two threads. All threads in both processes are CPU-intensive, i.e., they never block for I/O. The operating system uses simple round-robin scheduling.

a. Suppose that all of the threads are user-level threads, and that user-level threads are implemented using a single kernel thread per process. What percentage of the processor's time will be spent running P_a's threads? For full credit, briefly justify your answer.

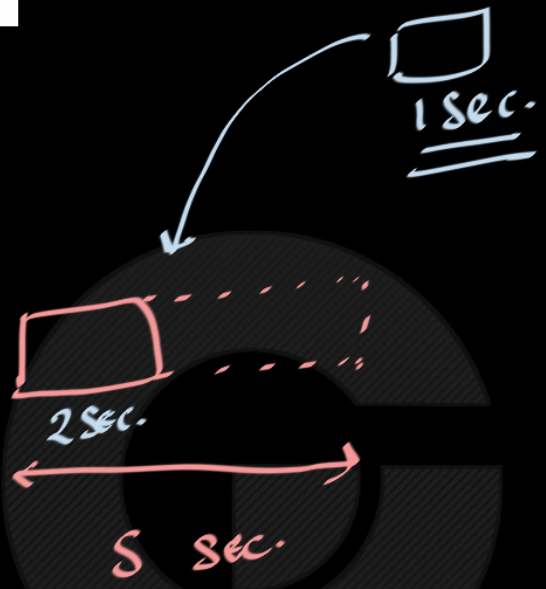
50% of the processor's time will be spent running P_a's threads.
Each process has a single kernel thread. Threads never block, so they will always consume their entire scheduling quantum. One a thread has consumed its quantum, it will be preempted in to allow the thread from the other process to run.

b. Suppose instead that all of the threads are kernel threads. What percentage of the processor's time will be spent running P_a's threads? For full credit, briefly justify your answer.

60% of the process's time will be spent running P_a's threads.
In this case there are five kernel threads, each of which will receive 20% of the processor's time. Three of these are associated with process P_a.

Shortest-remaining-time-first (SRTF)

- SRTF is the **preemptive** version of SJF
  - If the newly arrived process has a shorter running time than what is left of the currently executing process, then the OS will preempt the current process.


1 sec.
2 sec.
5 sec.

Question

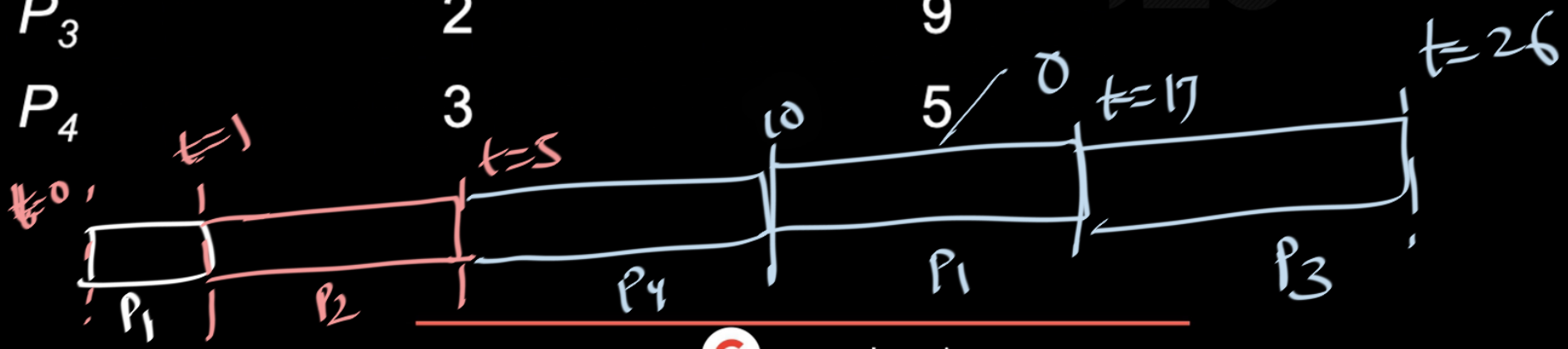
Shortest Remaining time first

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

Question

Shortest Remaining time first

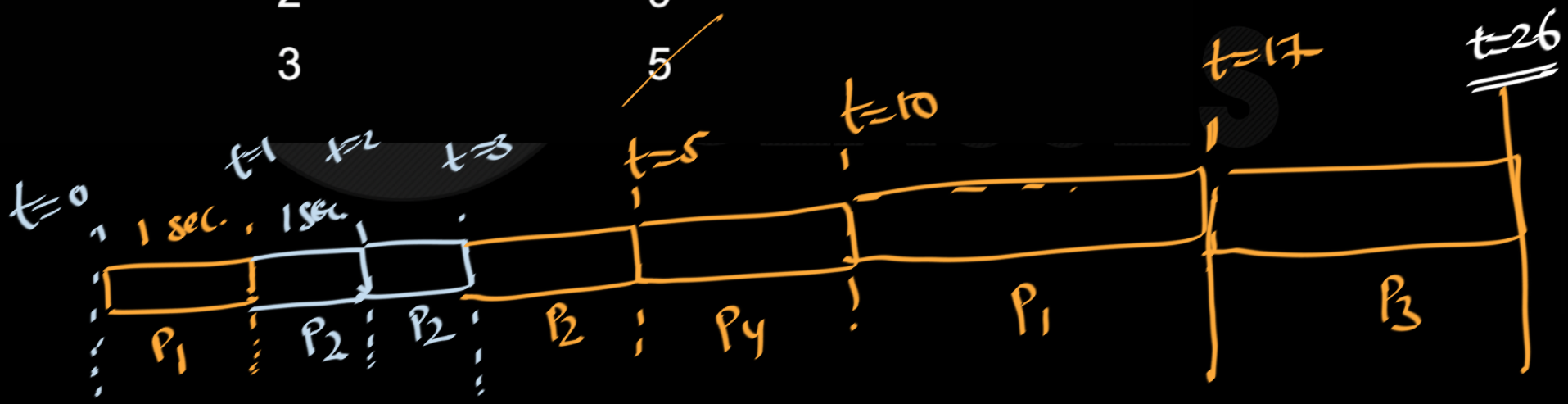
Process	Arrival Time	Burst Time
P_1	0	~~8~~ 7
P_2	1	~~4~~ 0
P_3	2	9
P_4	3	5



Question

Shortest Remaining time first

Process	*Arrival* Time	Burst Time
P_1	0	8 7
P_2	1	4 3 2 0
P_3	2	9
P_4	3	5



GO CLASSES

We can use min heap

- whenever new job enter into ready queue we insert that into min heap also decrease key with remaining burst time for current scheduled job

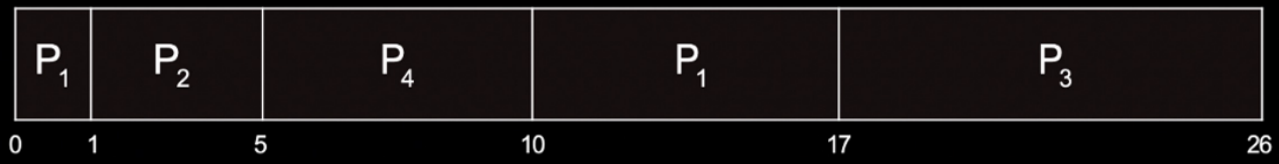
- Now we add the concepts of varying arrival times to the analysis

CT − AT = TAT

TAT − BT = WT

Process	*Arrival* Time	Burst Time	CT	TAT	WT
P_1	0	8	17	17	9
P_2	1	4	5	4	0
P_3	2	9	26	24	15
P_4	3	5	10	7	2

- *Preemptive* SJF Gantt Chart



$$\frac{9+0+15+2}{4} = \frac{26}{4}$$

$$= 6.5$$

- Now we add the concepts of varying arrival times to the analysis

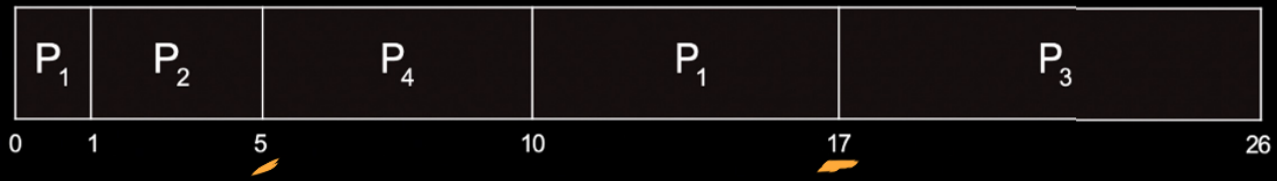
Process	*Arrival* Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

CT − AT = TAT

TAT − BT = WT

	CT	TAT	WT
	17	17	9
	5	4	0
	26	24	15
	10	7	2

- *Preemptive* SJF Gantt Chart



- Average waiting time = [(10-1)+(1-1)+(17-2)+5-3)]/4 = 26/4 = 6.5 msec

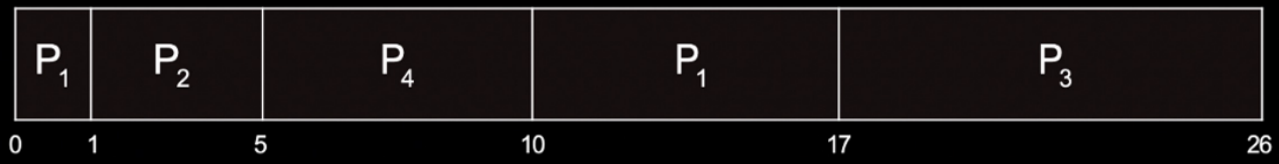
$\frac{9+0+15+2}{4} = \frac{26}{4}$

$= 6.5$

- Now we add the concepts of varying arrival times to the analysis

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

- *Preemptive* SJF Gantt Chart



- Average waiting time = [(10-1)+(1-1)+(17-2)+5-3)]/4 = 26/4 = 6.5 msec

Shortest Job first (Non preemptive version)

Shortest Remaining Job first (time) (Preemptive version)

Longest Job First (LJF)

- Opposite of Shortest Job First

- Criteria: Burst Time
- Mode: Non Preemptive
- Data Structure: Max Heap

longer burst Jobs will be Scheduled first

Question

Calculate following numbers for each process using Longest Job First (LJF)

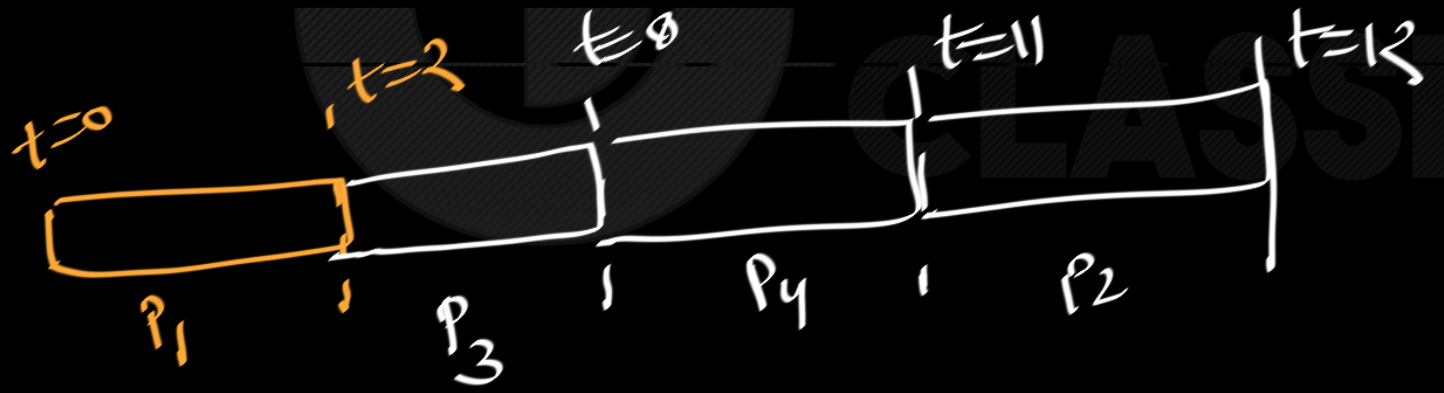
- Completion Time
- Turn around time
- Waiting Time

Process ID	Arrival Time (milliseconds)	Burst Time (milliseconds)
P1	0	2
P2	1	1
P3	2	6
P4	3	3

Question

Process ID	Arrival Time (milliseconds)	Burst Time (milliseconds)
P1	0	2
P2	1	1
P3	2	6
P4	3	3

	CT	TAT
P1	2	2
P2	13	11
P3	8	6
P4	11	8



0 P1	1 P1	2 P3	3 P3	4 P3	5 P3	6 P3	7 P3	8 P4	9 P4	10 P4	11 P2 12

TAT − BT

P ID	Arrival Time	Burst Time	Completion time (milliseconds)	Turn Around Time (milliseconds)	Waiting Time (milliseconds)
P1	0	2	2	2	0
P2	1	1	12	11	10
P3	2	6	8	6	0
P4	3	3	11	8	5

Total Turn Around Time = 2 + 11 + 6 + 8 = 27 milliseconds

Average Turn Around Time= Total Turn Around Time / Total No. of Processes = 27 / 4 = 6.75 milliseconds

Total Waiting Time = 0 + 10 + 0 + 5 = 15 milliseconds

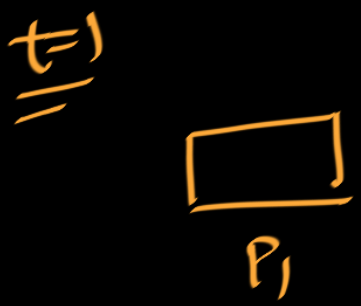
Average Waiting Time = Total Waiting Time / Total No. of Processes = 15 / 4 = 3.75 milliseconds

Longest Remaining Time First (LRTF)

- It is a preemptive version of the LJF algorithm

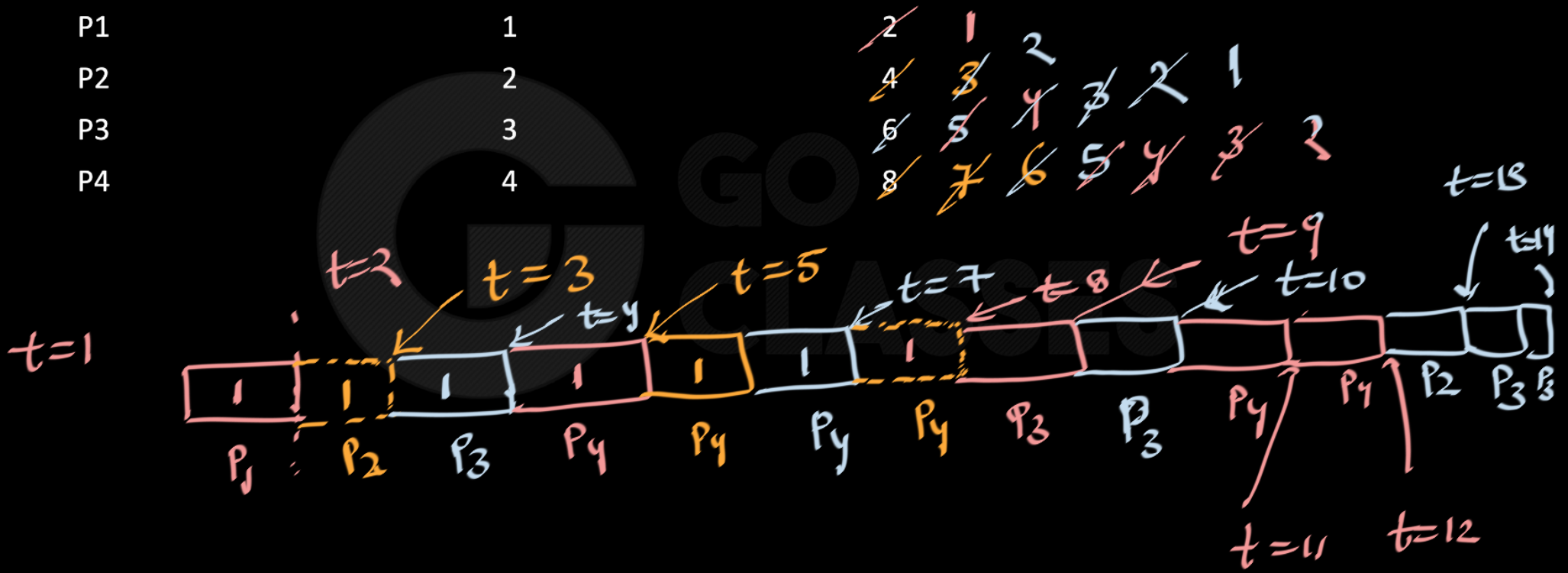
LRTF

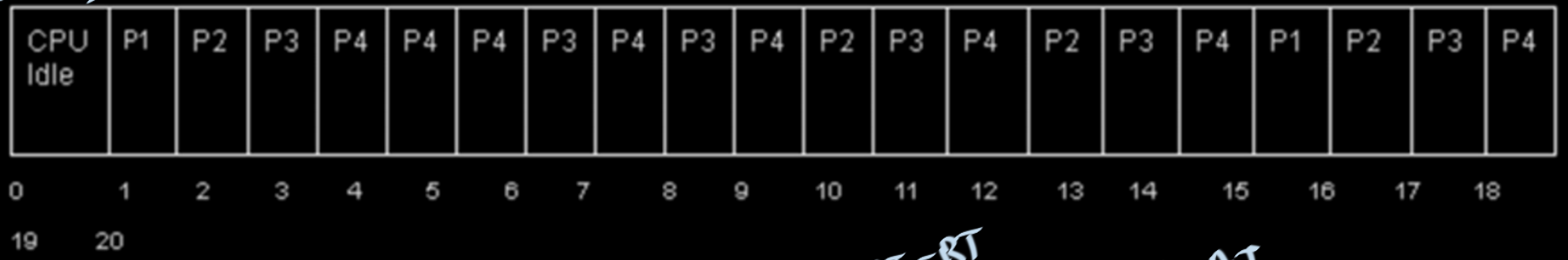
PROCESS	ARRIVAL TIME	BURST TIME
P1	1	2
P2	2	4
P3	3	6
P4	4	8



LRTF

PROCESS	ARRIVAL TIME	BURST TIME
P1	1	2
P2	2	4
P3	3	6
P4	4	8





= TAT - BT

= CT - AT

P. NO.	ARRIVAL TIME	BURST TIME	WAITING TIME(WT)	TURN AROUND TIME(TAT)	COMPLETION TIME (CT)
P1	1	2	15	17	18
P2	2	4	13	17	19
P3	3	6	11	17	20
P4	4	8	9	17	21

GATE CSE 2006 | Question: 64

33

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 turn around time is:

A. 13 units
B. 14 units
C. 15 units
D. 16 units

H.W.

https://gateoverflow.in/1842/gate-cse-2006-question-64

41

Best answer

A.

Gantt Chart is as follows.

P_2	P_2	P_2	P_2	P_1	P_2	P_1	P_2	P_0	P_1	P_2	P_0	P_1	P_2

Scheduling Table

P.ID	A.T	B.T	C.T	T.A.T.	W.T.
P0	0	2	12	12	10
P1	0	4	13	13	9
P2	0	8	14	14	6
TOTAL				39	25

A.T.= Arrival Time

B.T.= Burst Time

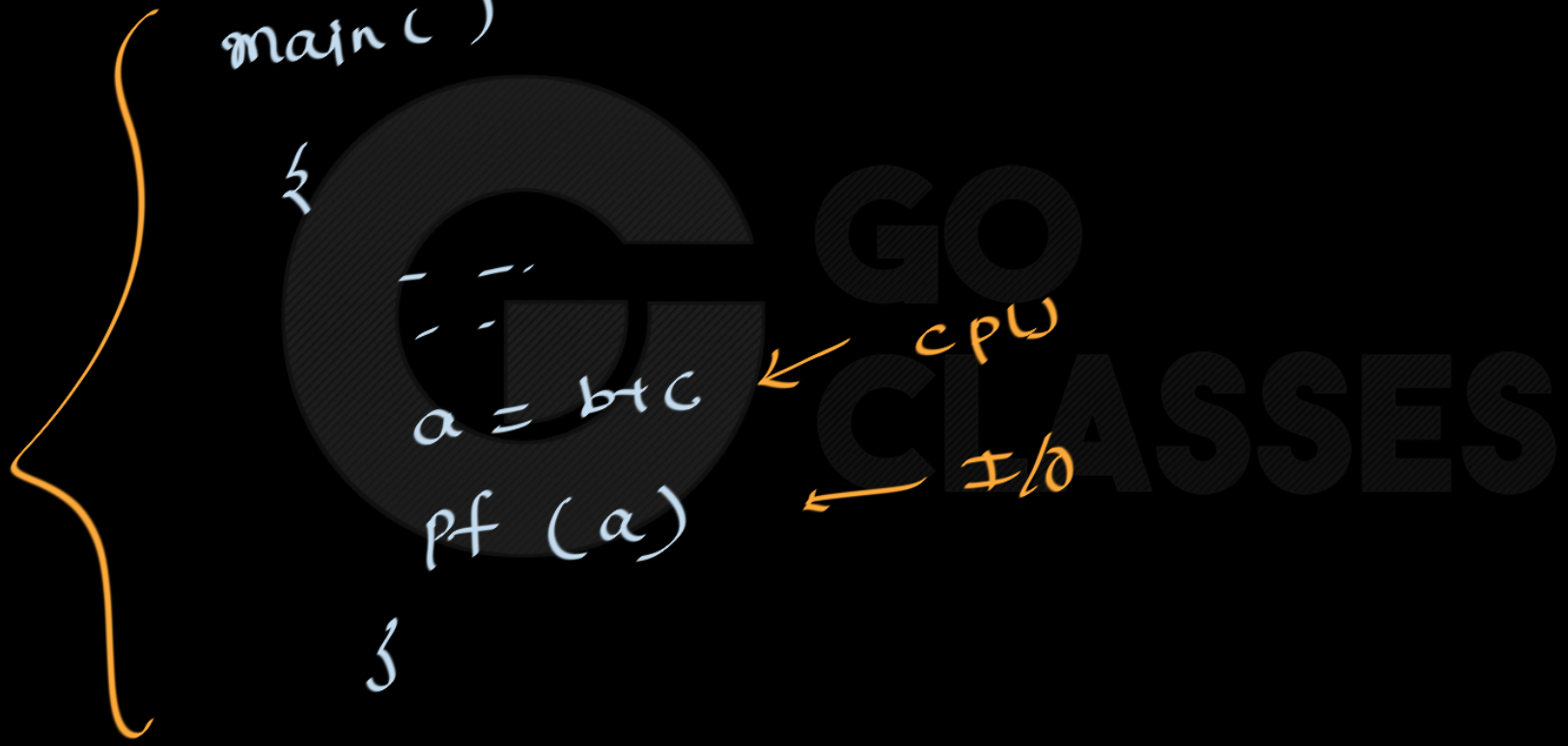
C.T.= Completion Time.

T.A.T.= Turn Around Time

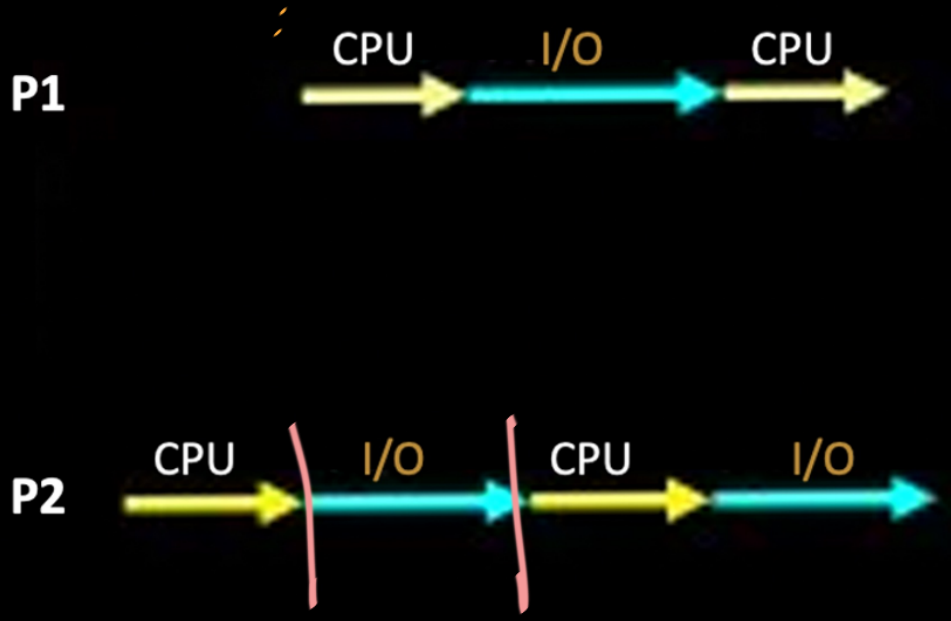
W.T.= Waiting Time.

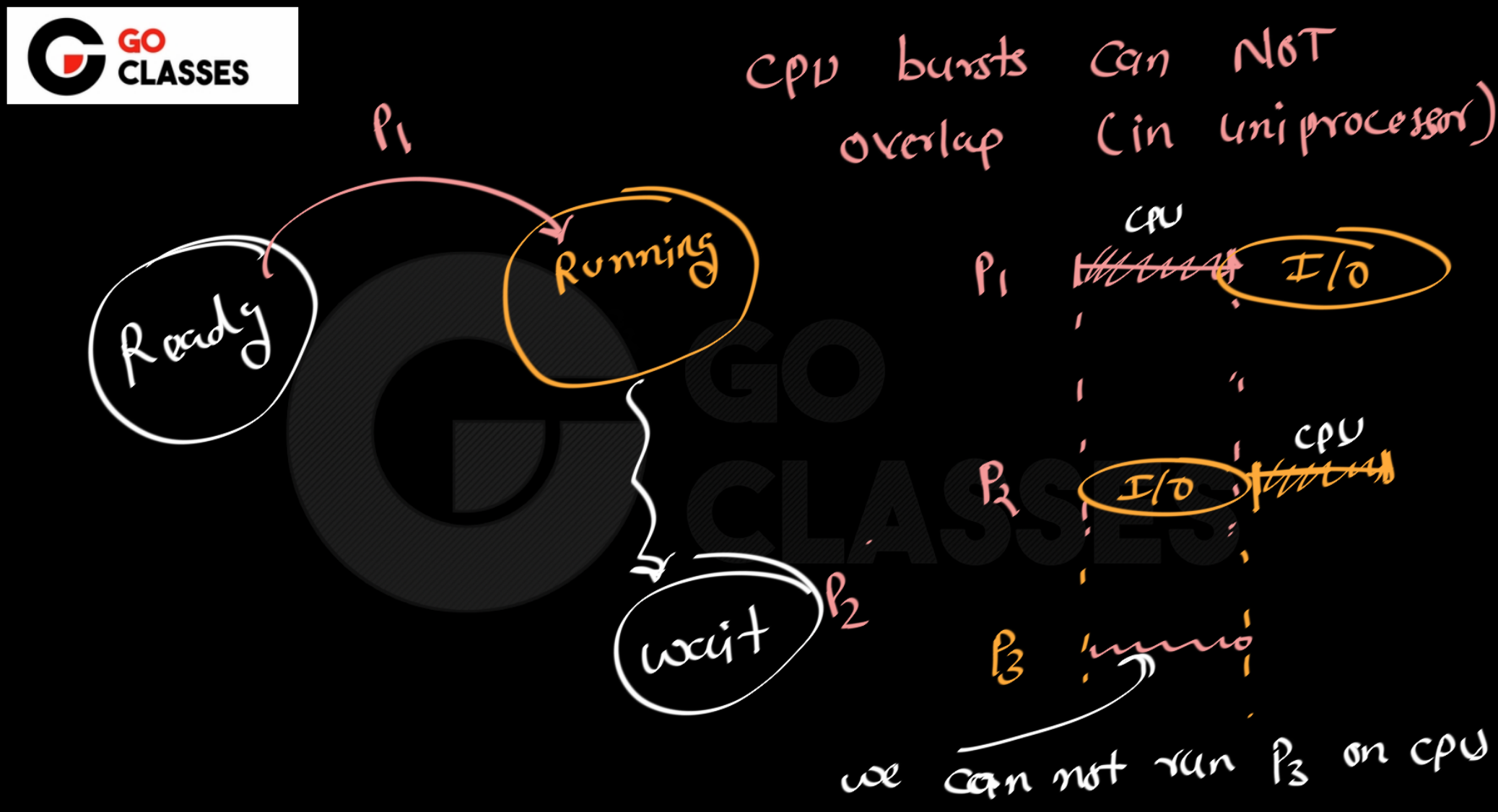
Average TAT $= 39/3 = 13$ units.

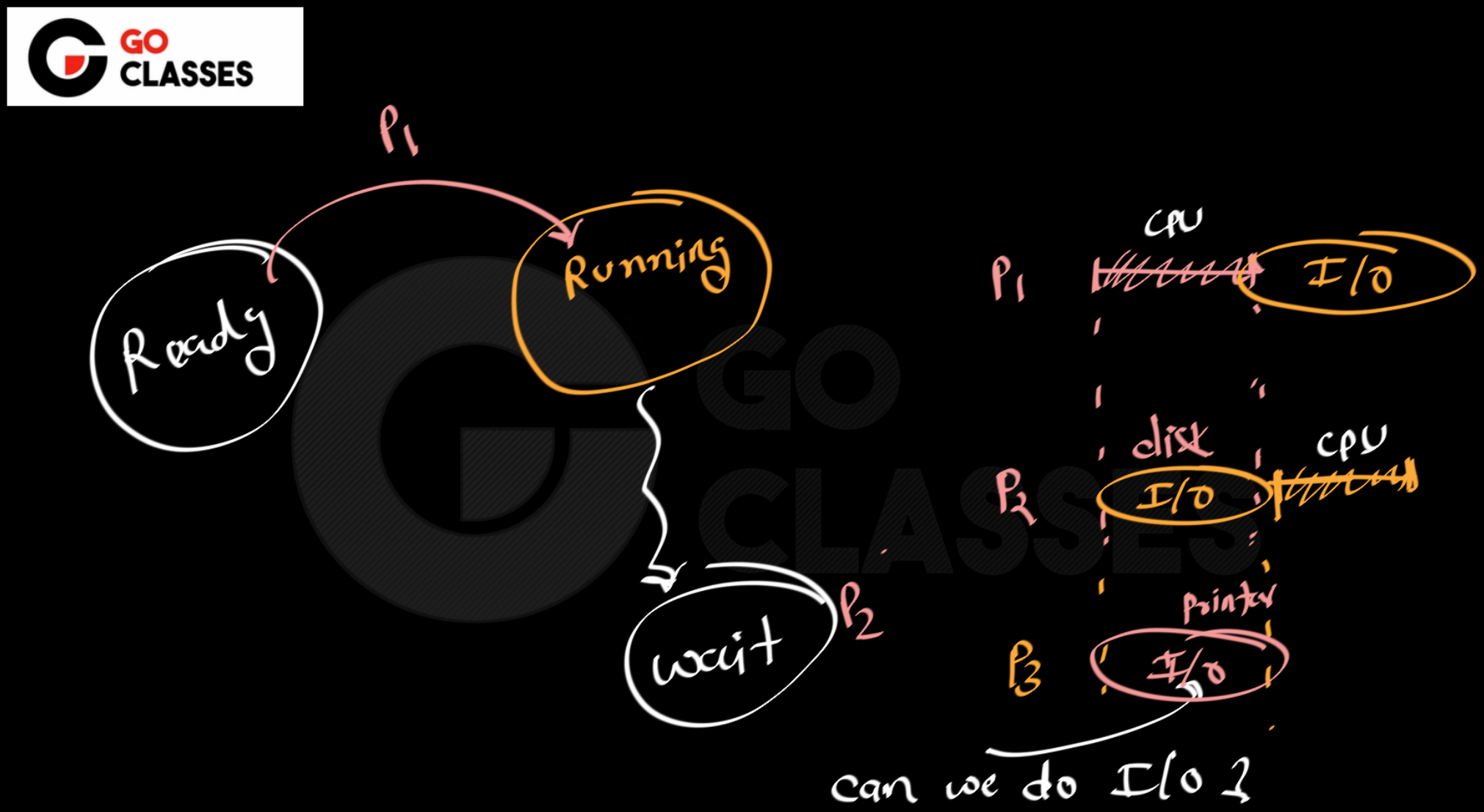
Scheduling with mixed workload

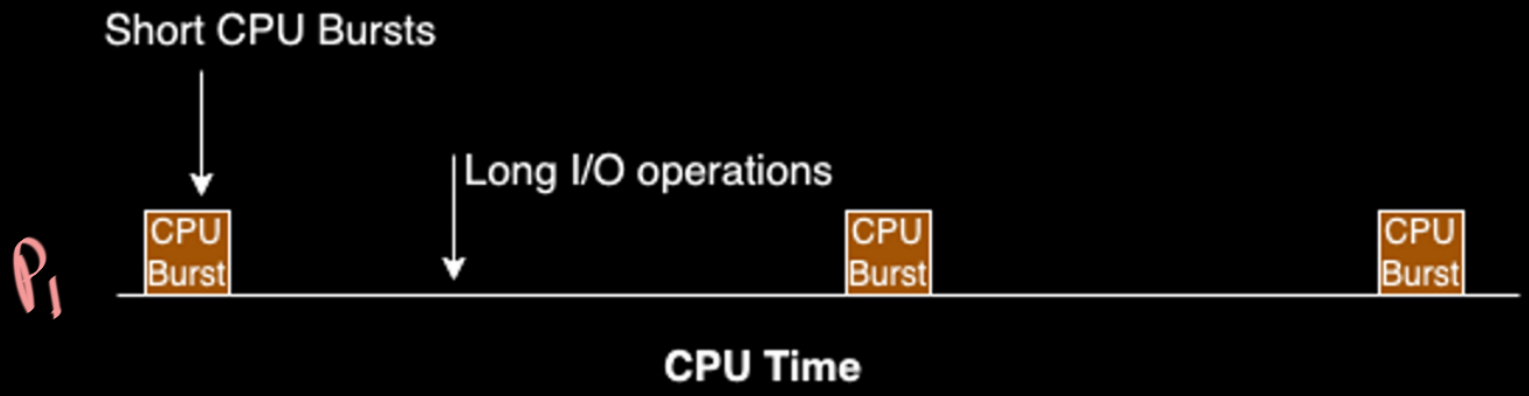

main( )
{
a = b+c
pf (a)
}
CPU
I/O

Scheduling with mixed workload

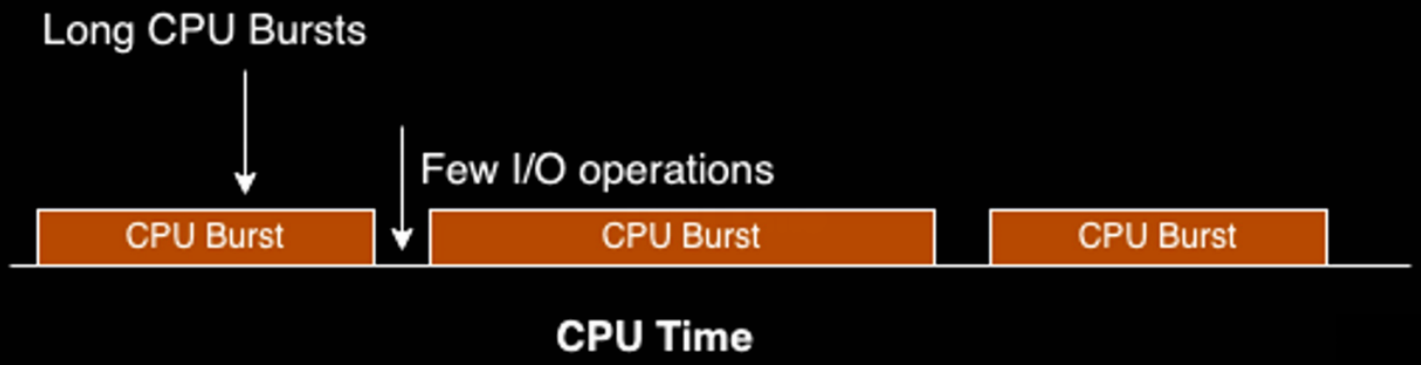



GO CLASSES
CPU bursts can NOT overlap (in uniprocessor)
P1
Ready
Running
wait
P2
CPU
P1
I/O
P2
I/O
CPU
P3
we can not run P3 on CPU


GO CLASSES
P1
Ready
Running
wait
P2
CPU
P1
I/O
disk
CPU
P2
I/O
printer
P3
I/O
can we do I/O ?

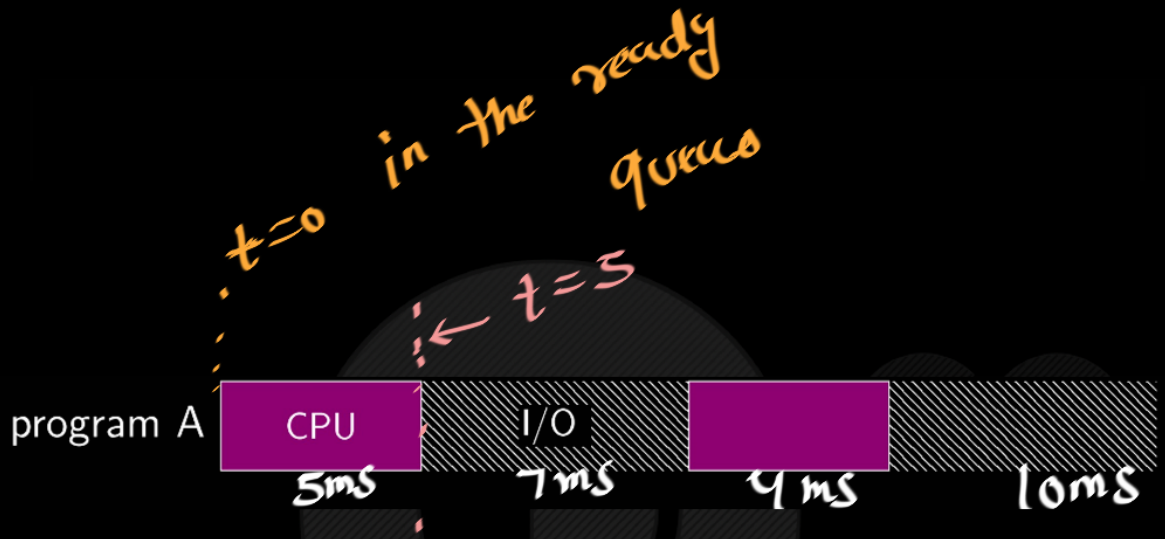


P1 is called I/O bound process because it is doing more I/O



P_2 =

P_2 is called CPU bound process because it is doing more CPU.

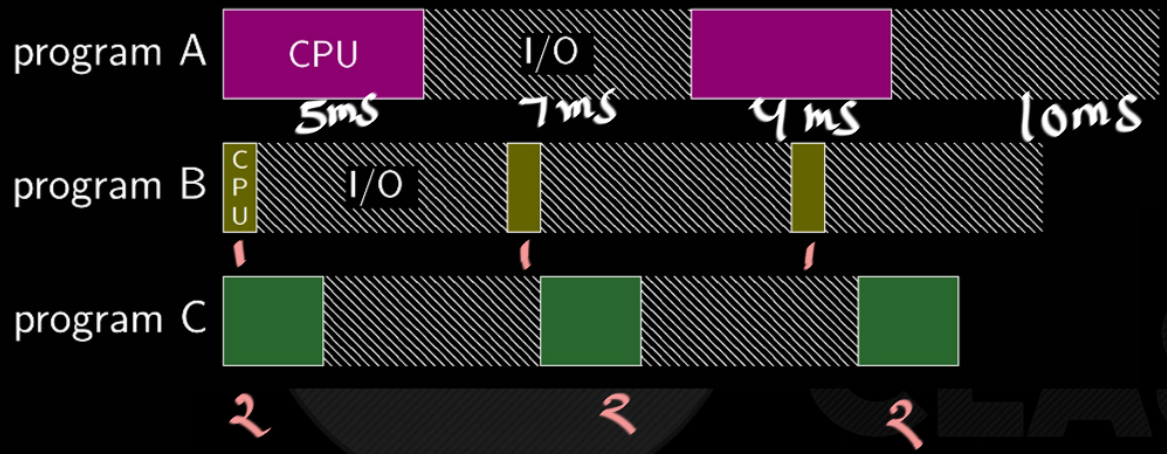


[we can not rearrange I/O or CPU bursts

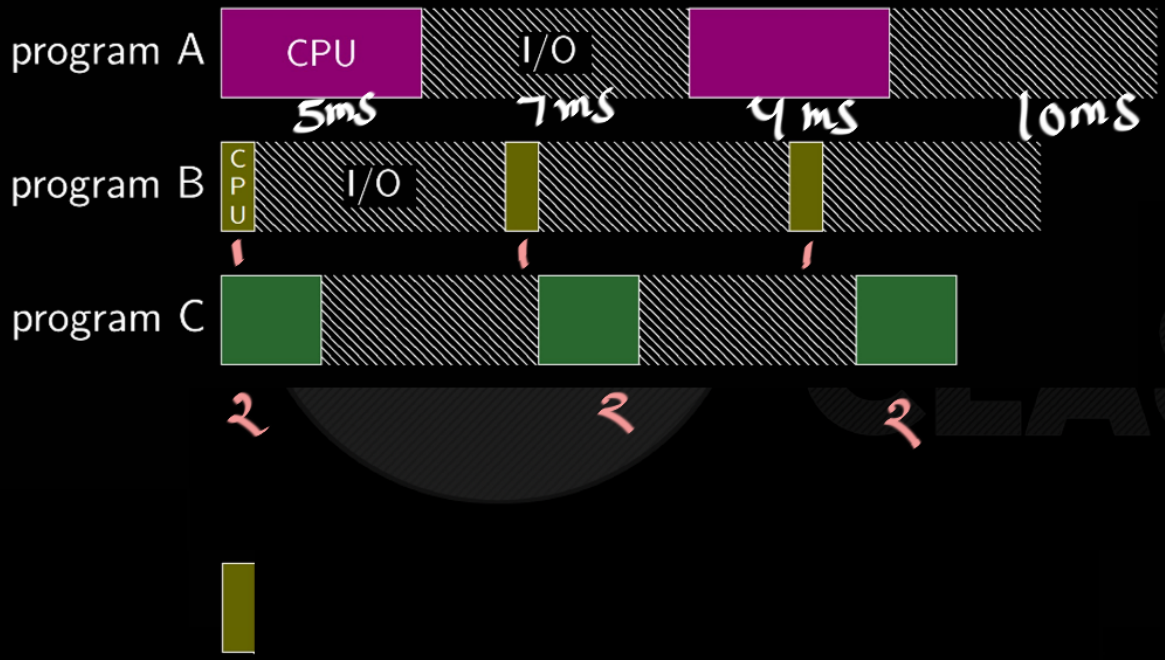
whater is given sequence it is fixed

```
a=b²+c
pf( )
{
}
```

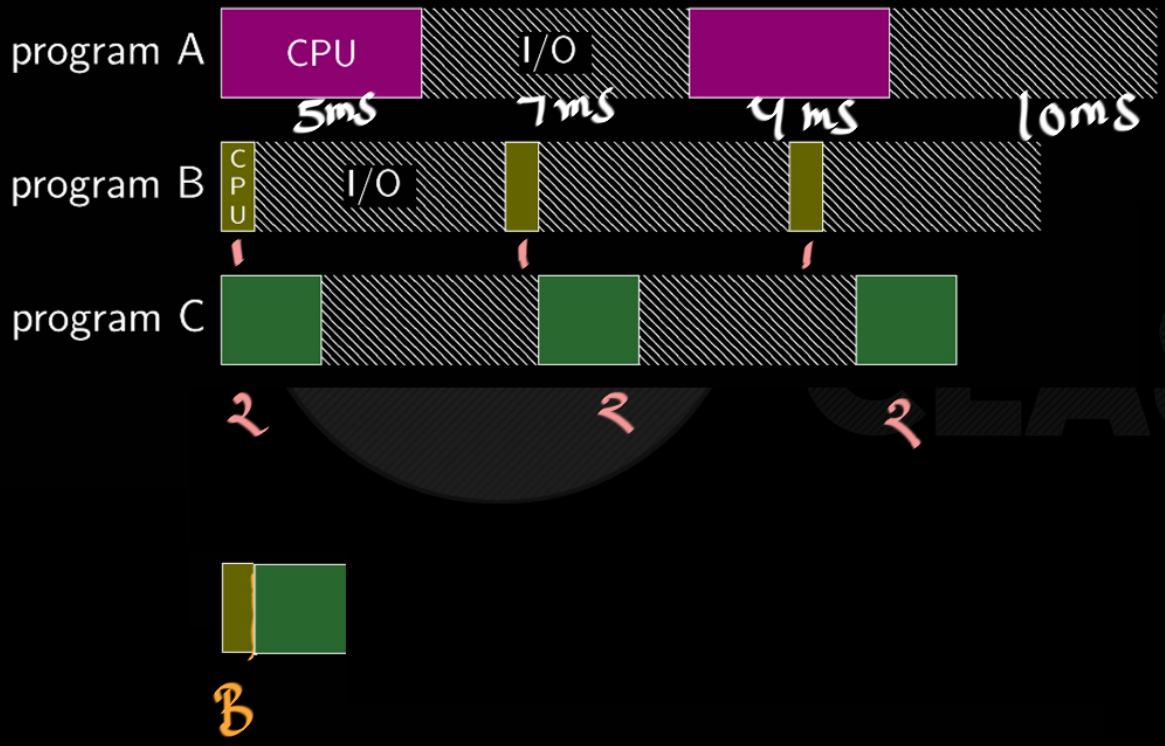
alternating I/O and CPU: SJF



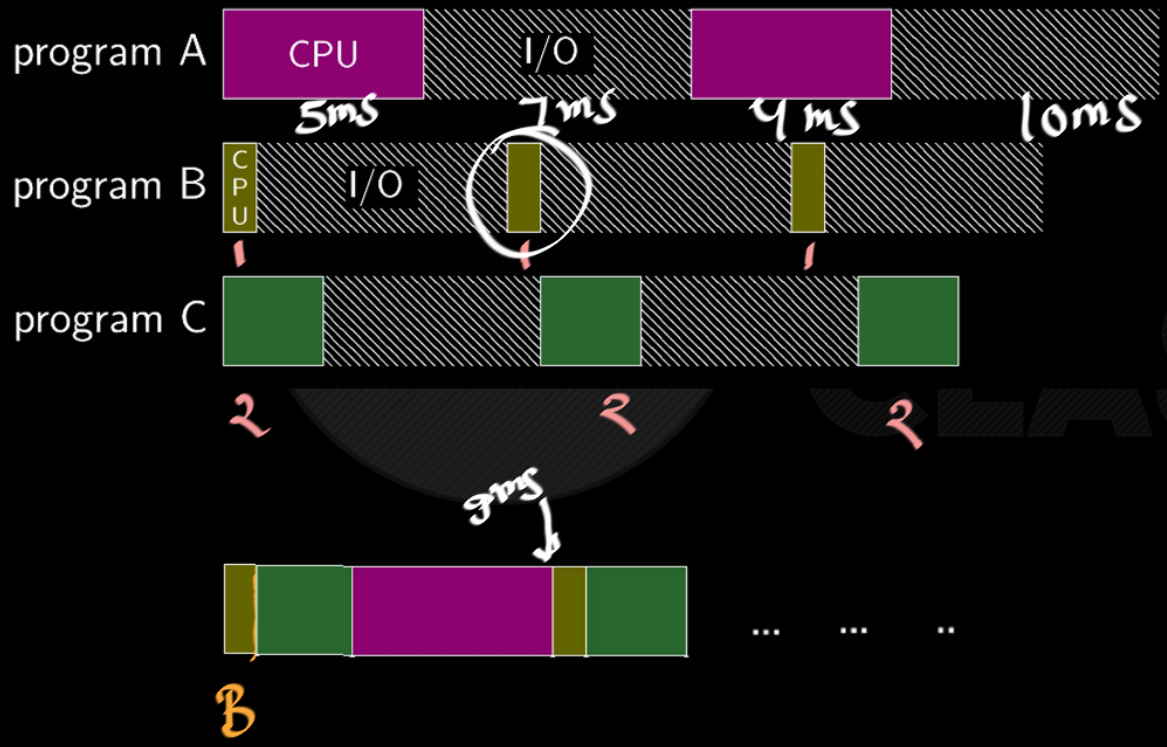
alternating I/O and CPU: SJF



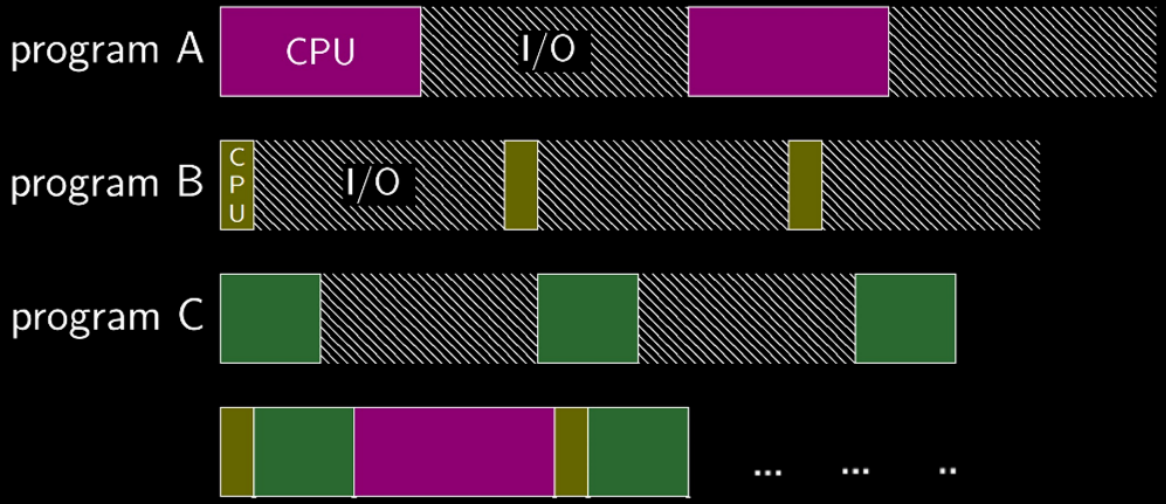
alternating I/O and CPU: SJF



alternating I/O and CPU: SJF



alternating I/O and CPU: SJF



Question

CT TAT WT

P1: CPU-7, IO-3, CPU-3
P2: CPU-5, IO-8, CPU-7

a. First-come/first-served
b. Shortest job first
c. Preemptive shortest job first
d. Round robin with a quantum of 2

https://people.umass.edu/tongping/teaching/ece670/Midterm-Review.pdf

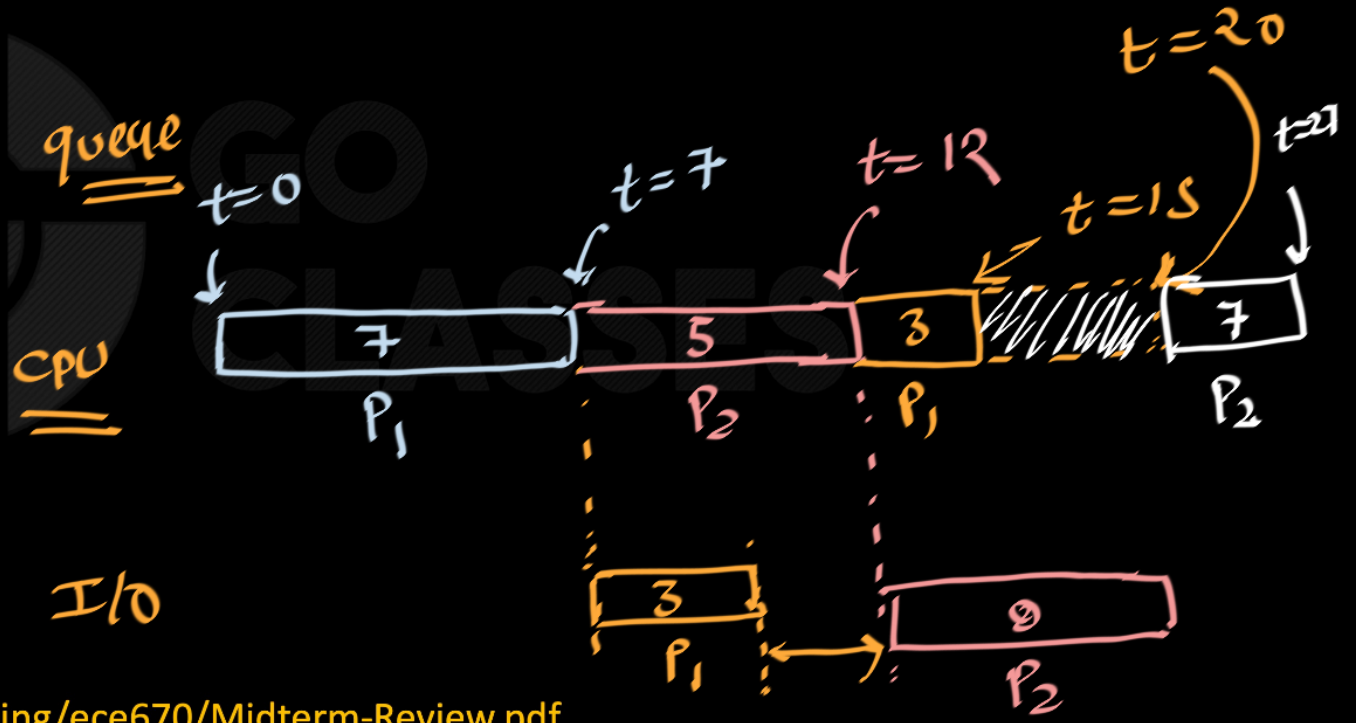
Question

AT = 0

Draw GANNT Chart

P1: CPU-7, IO-3, CPU-3
P2: CPU-5, IO-8, CPU-7

a. First-come/first-served
b. Shortest job first
c. Preemptive shortest job first
d. Round robin with a quantum of 2



https://people.umass.edu/tongping/teaching/ece670/Midterm-Review.pdf

a. First-come/first-served

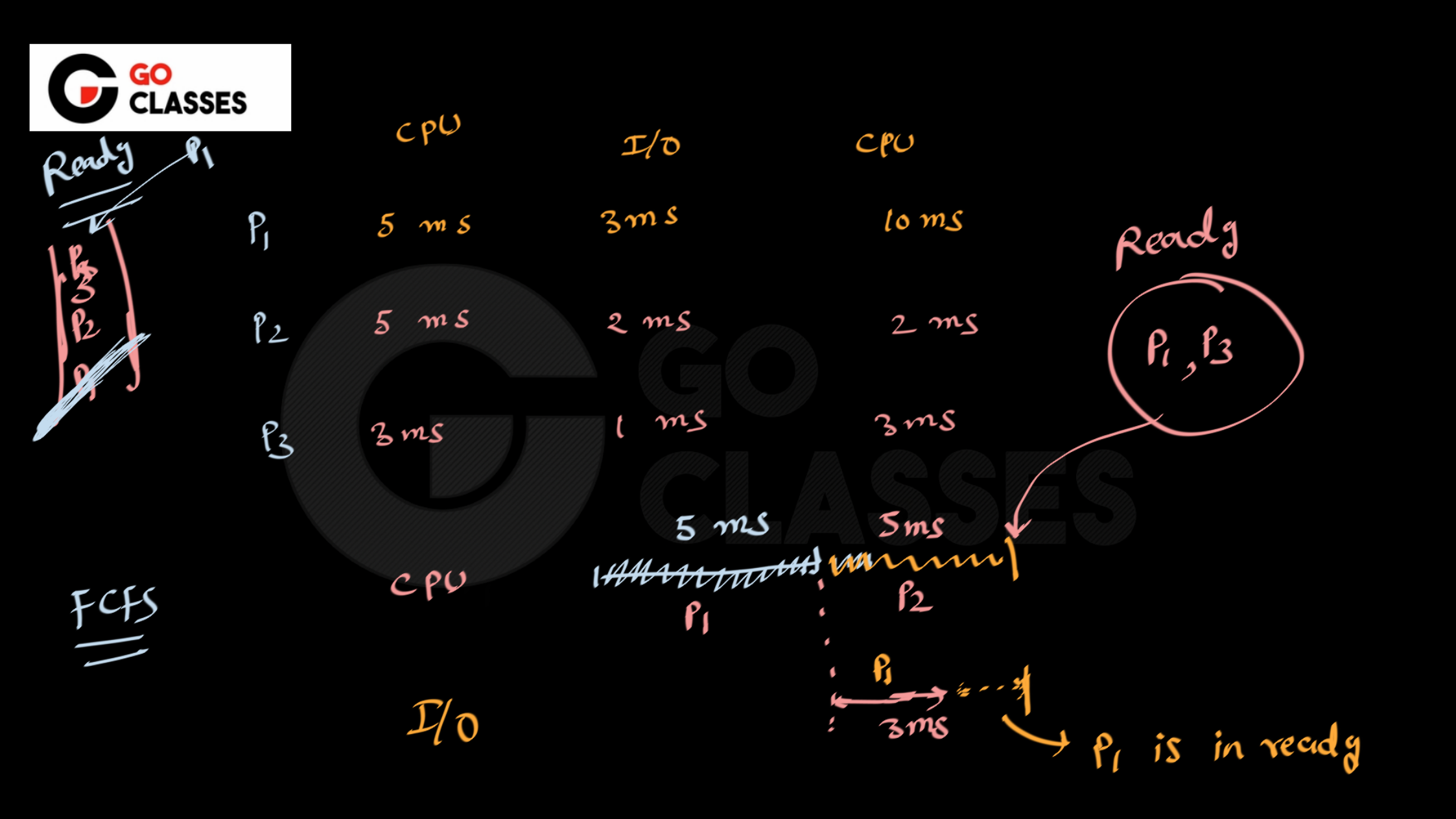
Algorithm	P1 WT	P2 WT	Avg WT	P1 FT	P1 FT	Schd Len	CPU Ut
FCFS	2	7	4.5	15	27	27	81.48%

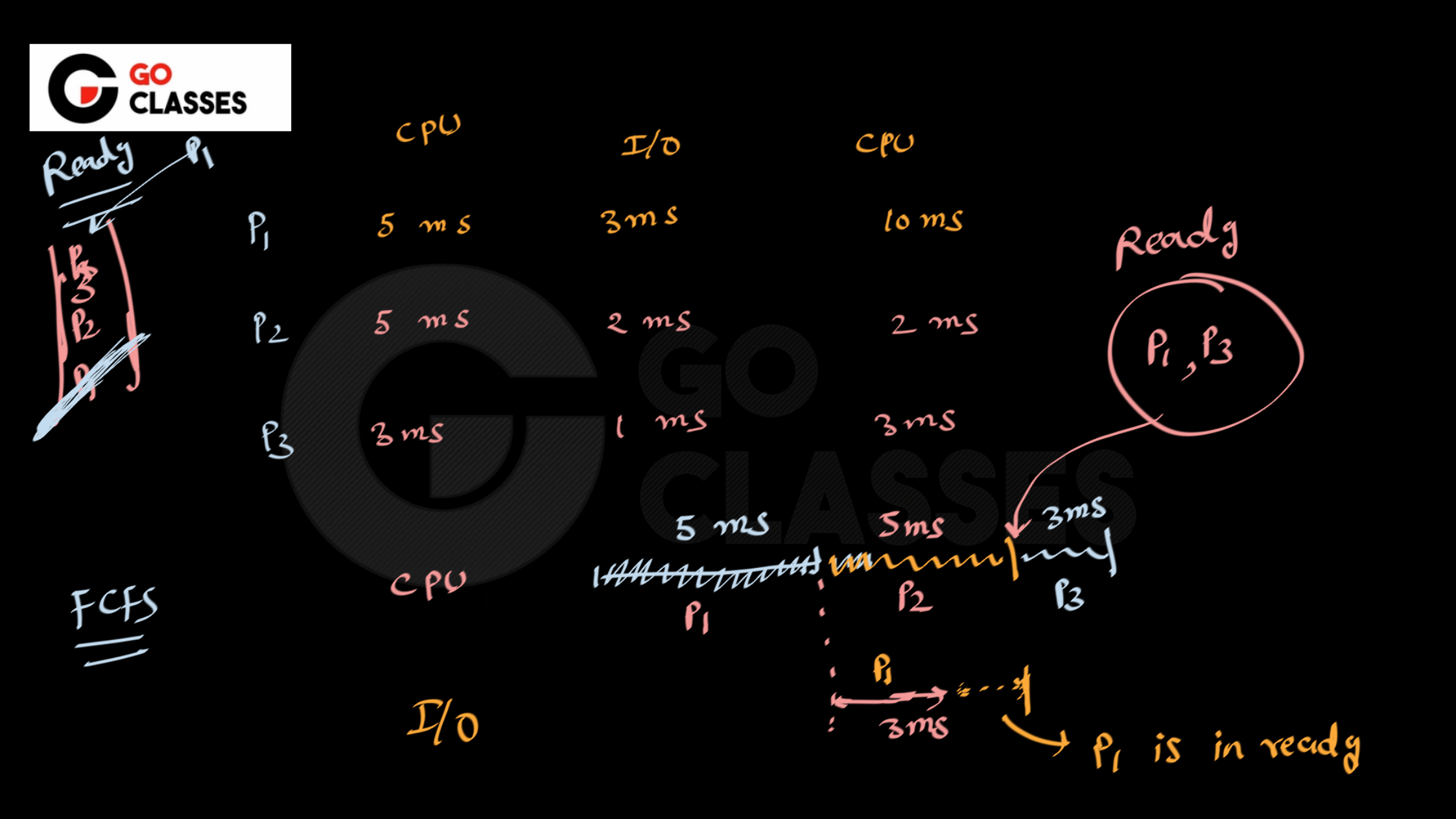
	0					5		7			10		12			15					20					25		27		30					35
P1	R	R	R	R	R	R	R	w	w	w	r	r	R	R	R																			
P2	r	r	r	r	r	r	r	R	R	R	R	R	w	w	w	w	w	w	w	w	R	R	R	R	R	R	R	R						

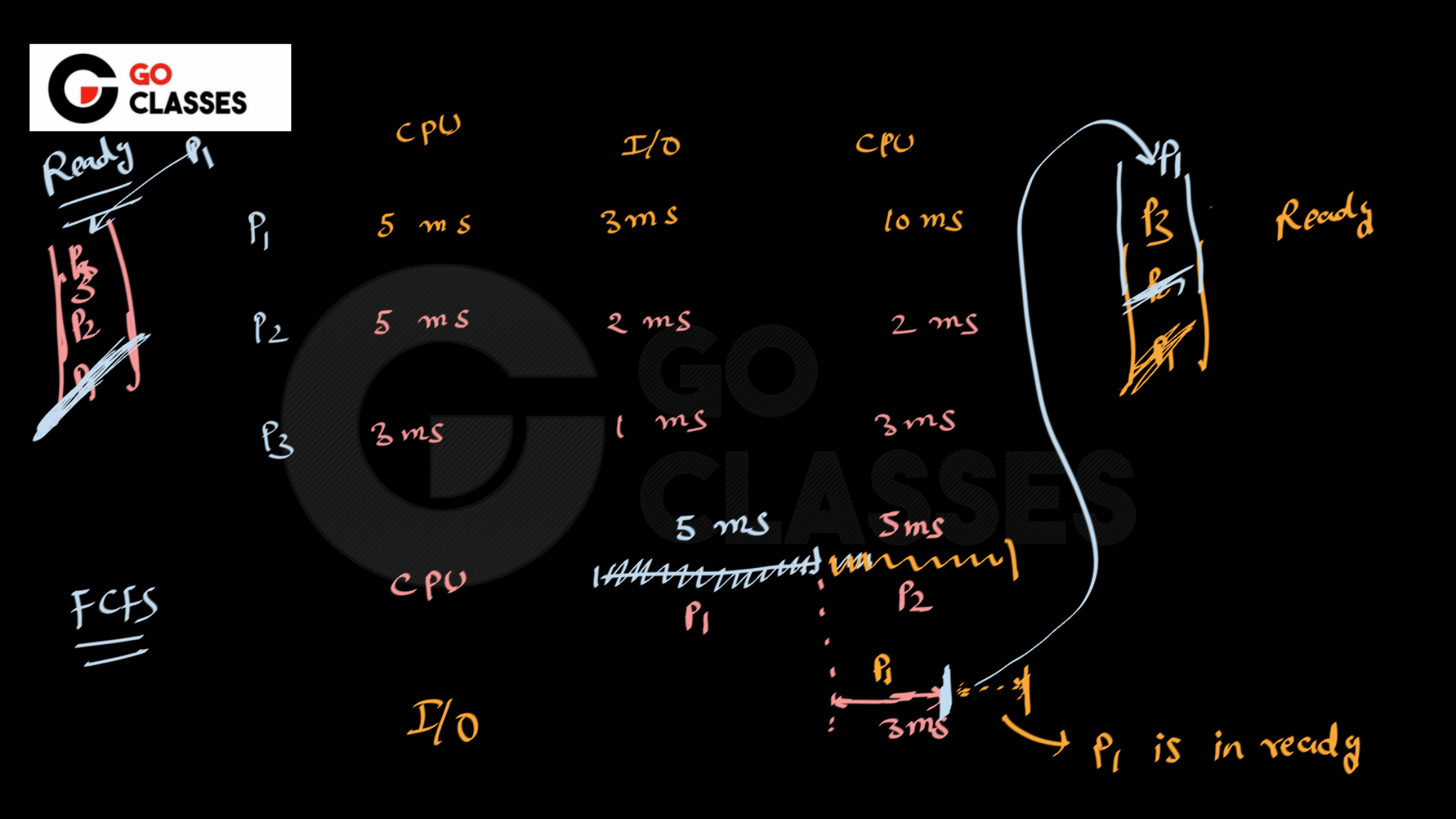
R: Running r: ready w: waiting queue (I/O)

FCFS : P_1, P_2, P_3 AT = 0

P_1 is running


GO CLASSES
Ready
P1
P3
P2
FCFS
CPU
I/O
CPU
P1
5 ms
3ms
10 ms
P2
5 ms
2 ms
2 ms
P3
3ms
1 ms
3ms
Ready
P1 , P3
5 ms
3ms
P1
P2
P1
3ms
P1 is in ready


GO CLASSES
Ready
P1
P3
P2
CPU
I/O
CPU
P1
5 ms
3ms
10 ms
P2
5 ms
2 ms
2 ms
P3
3ms
1 ms
3ms
Ready
P1 , P3
5 ms
5ms
3ms
P1
P2
P3
FCFS
CPU
I/O
P1
3ms
P1 is in ready

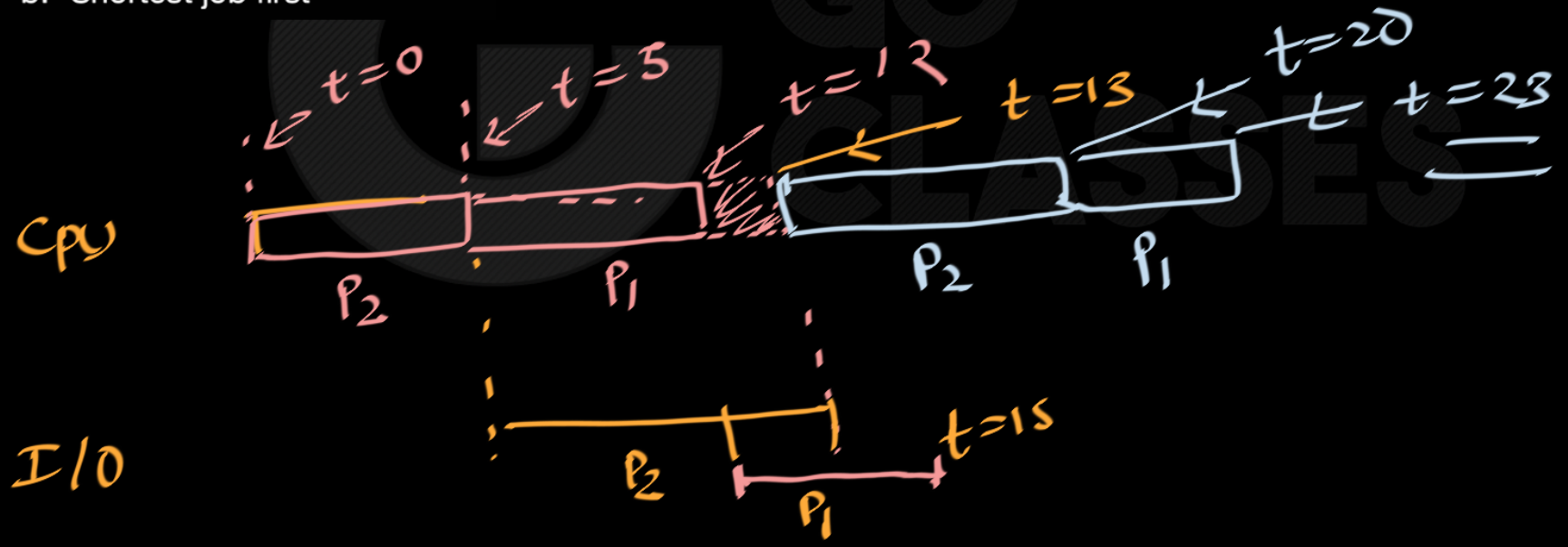

GO CLASSES
Ready
P1
P3
P2
FCFS
CPU
I/O
CPU
P1 5 ms 3ms 10 ms
P2 5 ms 2 ms 2 ms
P3 3ms 1 ms 3ms
5 ms
3ms
P1
P2
P1
3ms
P1 is in ready
P1
P3
Ready

GO CLASSES

P1: CPU-7, IO-3, CPU-3
P2: CPU-5, IO-8, CPU-7

I/O can overlap.

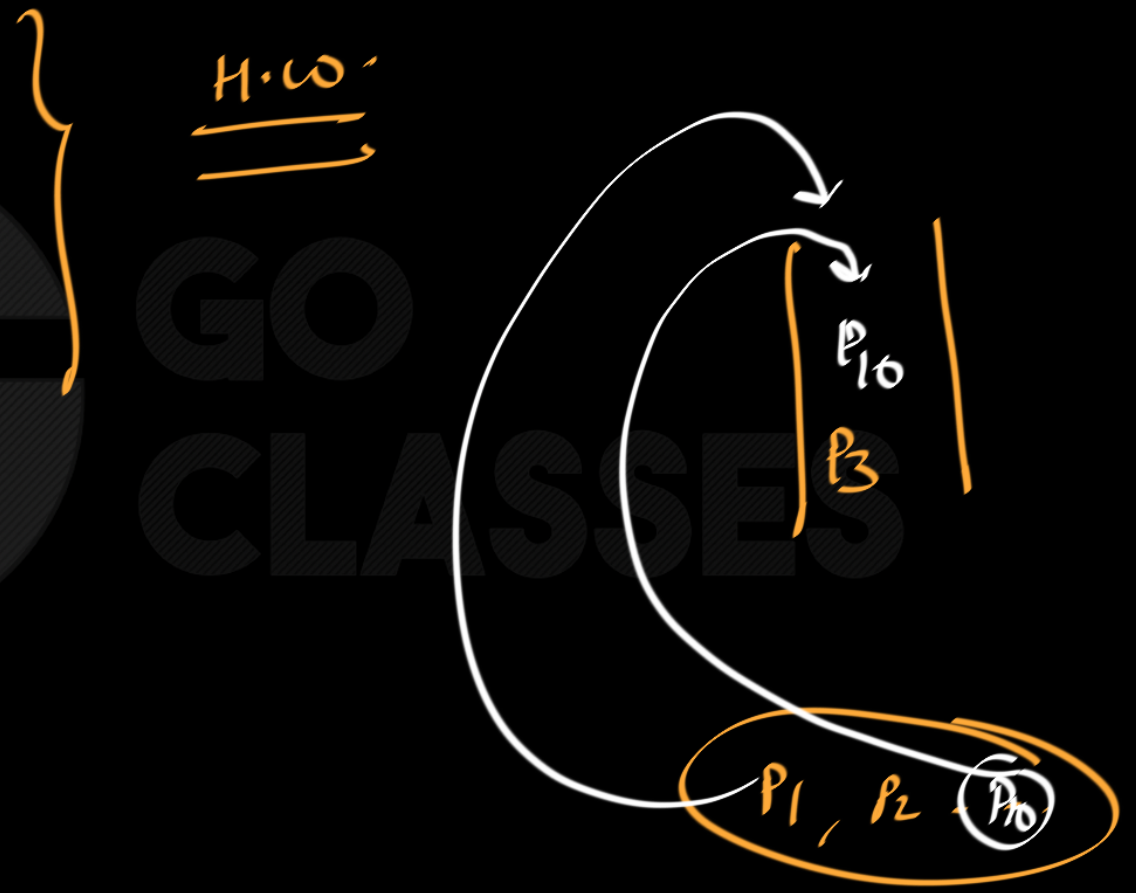
b. Shortest job first



P1: CPU-7, IO-3, CPU-3
P2: CPU-5, IO-8, CPU-7

c. Preemptive shortest job first

d. Round robin with a quantum of 2



a. First-come/first-served

Algorithm	P1 WT	P2 WT	Avg WT	P1 FT	P1 FT	Schd Len	CPU Ut
FCFS	2	7	4.5	15	27	27	81.48%

0					5					10					15					20					25					30					35
P1	R	R	R	R	R	R	R	w	w	w	r	r	R	R	R																			
P2	r	r	r	r	r	r	r	R	R	R	R	R	w	w	w	w	w	w	w	w	R	R	R	R	R	R	R							

b. Shortest job first

					5					10					15					20					25					30					35
P1	r	r	r	r	r	R	R	R	R	R	R	R	w	w	w	r	r	r	r	r	R	R	R											
P2	R	R	R	R	R	w	w	w	w	w	w	w	w	R	R	R	R	R	R	R														

c. Preemptive shortest job first

Algorithm	P1 WT	P2 WT	Avg WT	P1 FT	P1 FT	Schd Len	CPU Ut
PSJF	5	3	4	18	23	23	95.65%

					5					10					15					20					25					30					35
P1	r	r	r	r	r	R	R	R	R	R	R	R	w	w	w	R	R	R																
P2	R	R	R	R	R	w	w	w	w	w	w	w	w	R	R	r	r	r	R	R	R	R	R											

d. Round robin with a quantum of 2

					5					10					15					20					25					30					35
P1	R	R	r	r	R	R	r	r	R	R	r	R	w	w	w	R	R	R																
P2	r	r	R	R	r	r	R	R	r	r	R	w	w	w	w	w	w	w	w	R	R	R	R	R	R	R								

Algorithm	P1 WT	P2 WT	Avg WT	P1 FT	P1 FT	Schd Len	CPU Ut
RR 2	5	6	5.5	18	26	26	84.62%