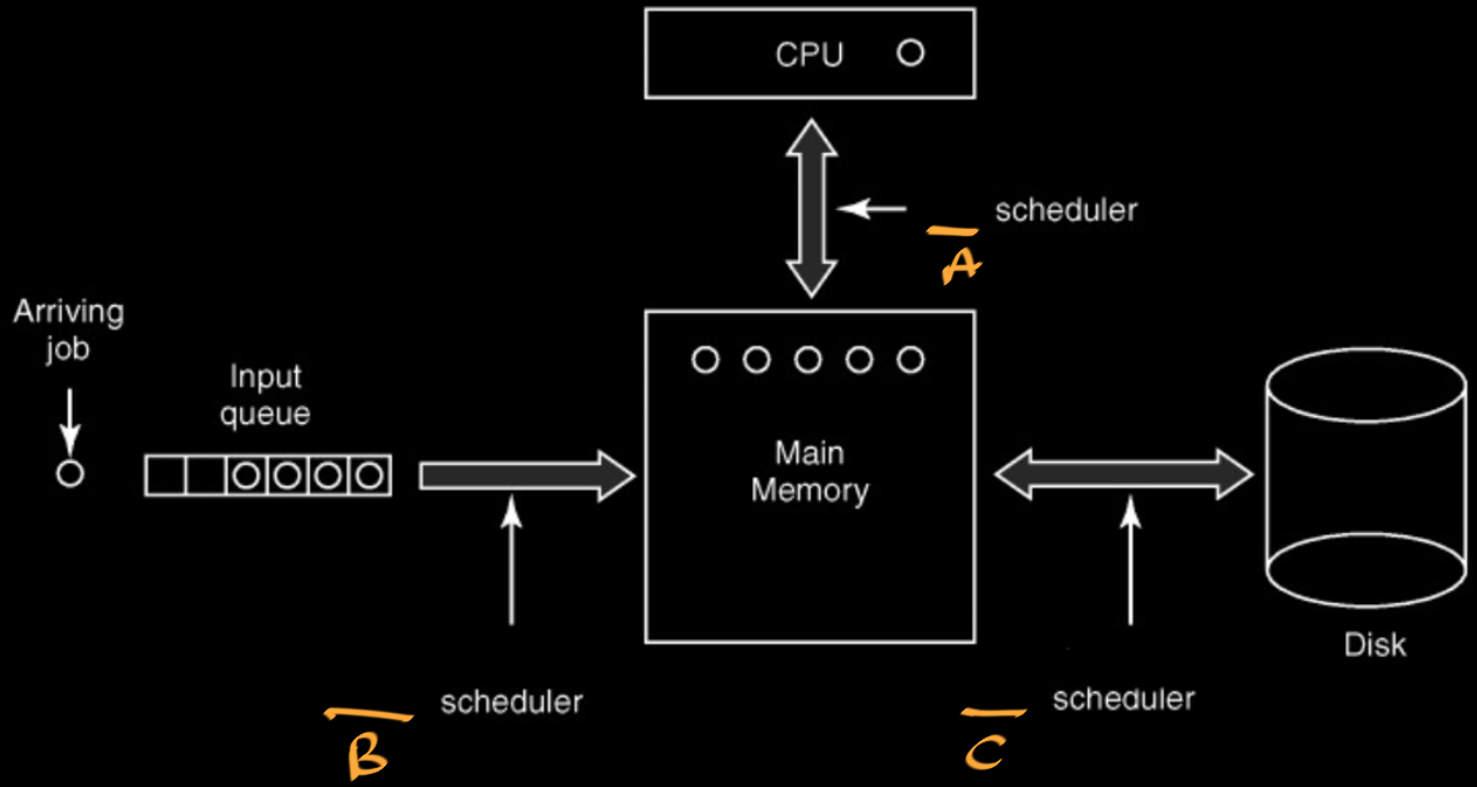
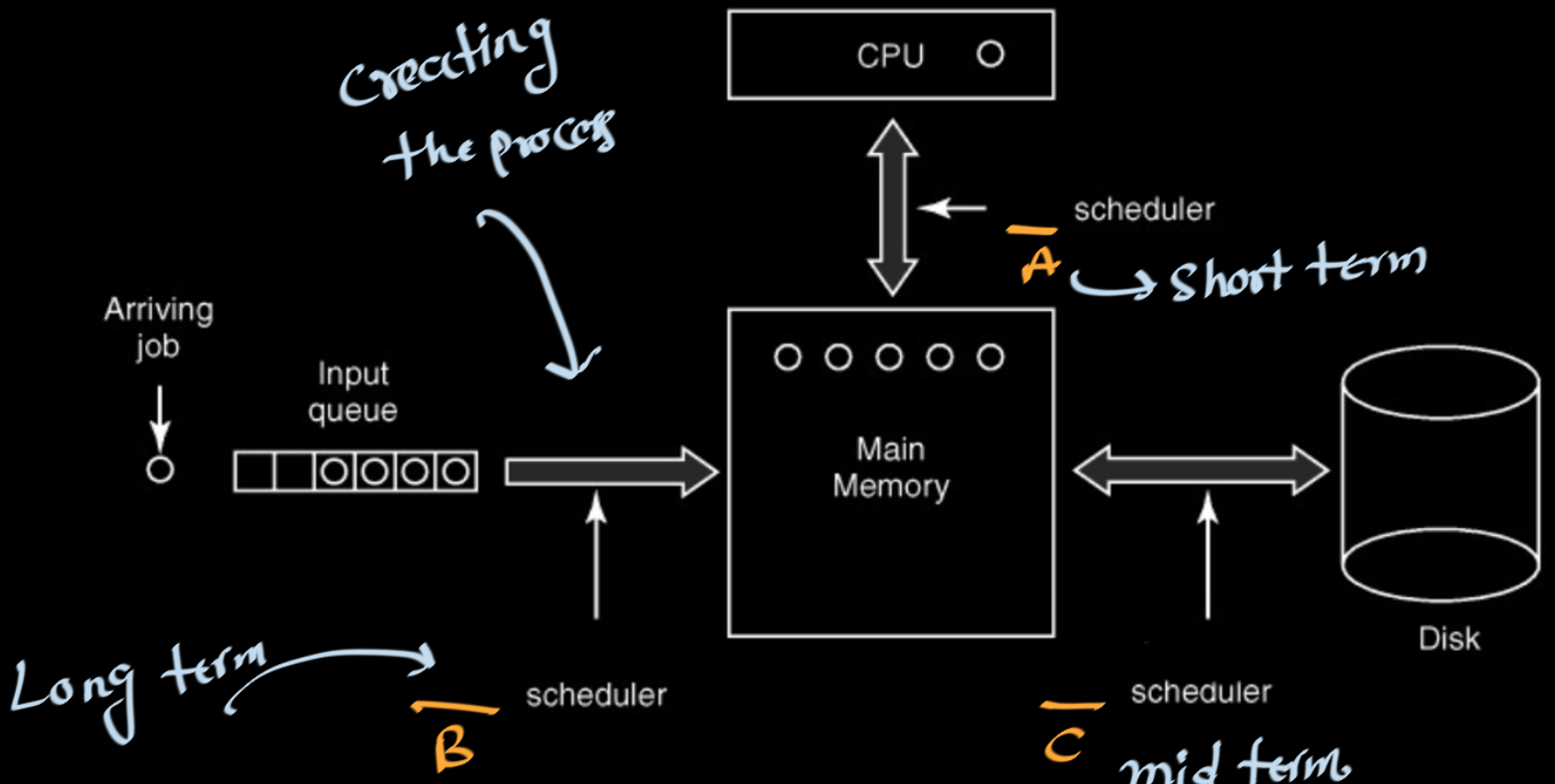
Lecture 10

Three Types of Schedulers



Three Types of Schedulers



Scheduling Algorithms

First-Come First-Serve (FCFS)

- By far the simplest CPU-scheduling algorithm

First-Come First-Serve (FCFS)

- By far the simplest CPU-scheduling algorithm

Process	Burst Time
P_1	24
P_2	3
P_3	3

First-Come, First-Served (FCFS) Scheduling

- Example

Process	Burst Time
P1	24
P2	3
P3	3

Gantt Chart for Schedule

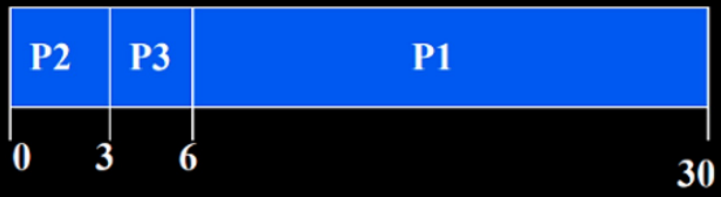
- Suppose all processes arrived at around time 0 in the following order:
  - P1, P2, P3
- Waiting time
  - P1 = 0;
  - P2 = 24;
  - P3 = 27;
- Average waiting time
  - (0+24+27)/3 = 17

FCFS Scheduling (cont.)

- Example

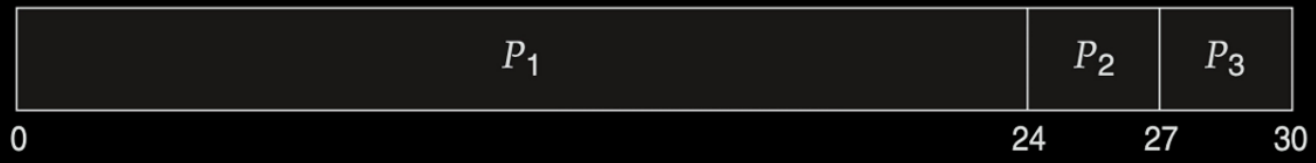
Process	Burst Time
P1	24
P2	3
P3	3

Gantt Chart for Schedule

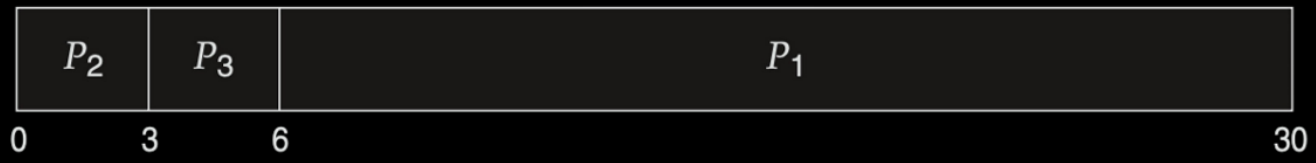


- Suppose the arrival order for the processes is
  - P2, P3, P1
- Waiting time
  - P1 = 6; P2 = 0; P3 = 3;
- Average waiting time
  - (6+0+3)/3 = 3 , better..
- *Convoy Effect*:
  - short processes waiting behind long process..

If the processes arrive in the order P_1, P_2, P_3, and are served in FCFS order, we get the result shown in the following **Gantt chart**, which is a bar chart that illustrates a particular schedule, including the start and finish times of each of the participating processes:



The waiting time is 0 milliseconds for process P_1, 24 milliseconds for process P_2, and 27 milliseconds for process P_3. Thus, the average waiting time is $(0 + 24 + 27)/3 = 17$ milliseconds. If the processes arrive in the order P_2, P_3, P_1, however, the results will be as shown in the following Gantt chart:



The average waiting time is now $(6 + 0 + 3)/3 = 3$ milliseconds. This reduction is substantial. Thus, the average waiting time under an FCFS policy is generally not minimal and may vary substantially if the processes' CPU burst times vary greatly.

Source: Galvin

Convoy Effect

because of one (slow or) long process everyone else is suffering

https://compas.cs.stonybrook.edu/~nhonarmand/courses/fa17/cse306/slides/09-sched.pdf

Convoy effect in FCFS

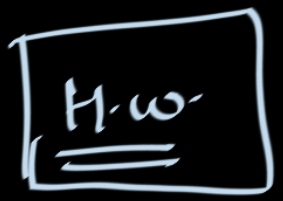
Convoy effect - short process behind long process

Convoy effect in FCFS

There is a **convoy effect** as all the other processes wait for the one big process to get off the CPU. This effect results in lower CPU and device utilization than might be possible if the shorter processes were allowed to go first.

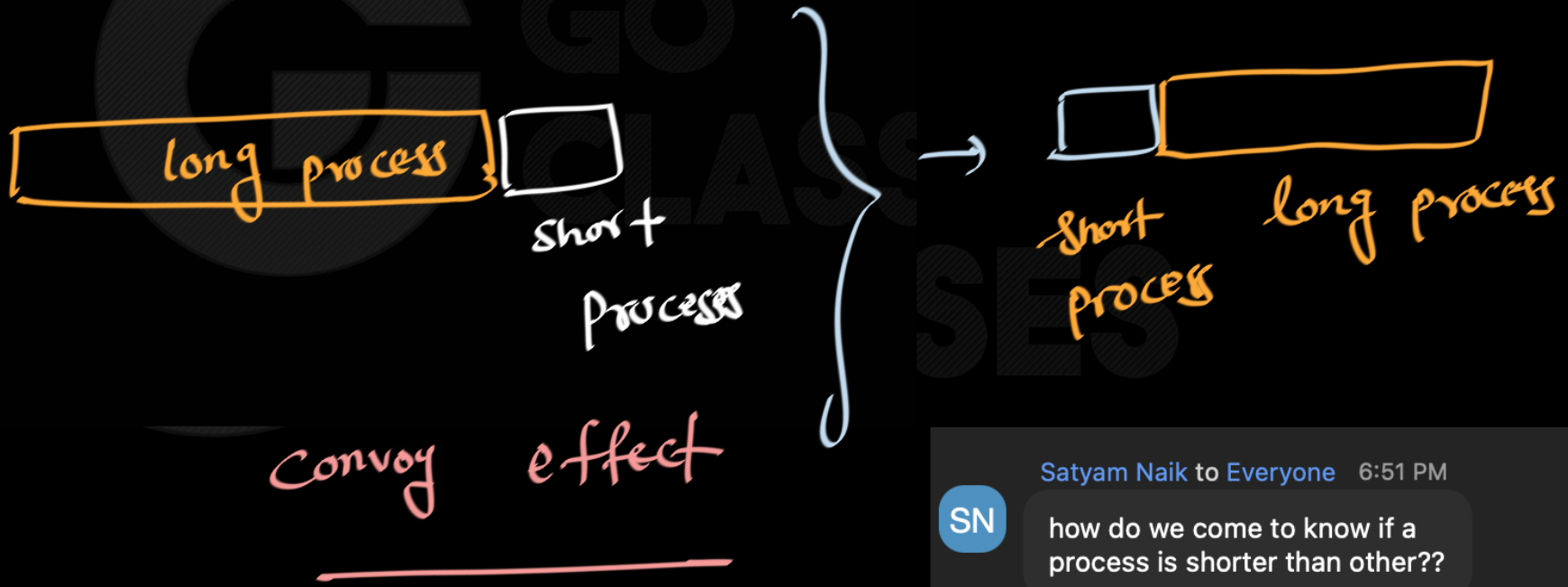
Source: Galvin

Example 2:



Calculate Avg waiting time and Avg TAT

Sr. No	Process	Execution Time/ Burst Time(ms)
1	p0	10
2	p1	4
3	p2	8
4	p3	6



Satyam Naik to Everyone 6:51 PM

SN

how do we come to know if a process is shorter than other??

3

idea: Put short process first than long processes.

Shortest Job first

(will it really help us to decrease avg waiting time?)

Shortest Job First (SJF)

Shortest-Job-First Scheduling (SJF)

- When the CPU is available, it is assigned to the process that has the smallest next CPU burst.

- If the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie.

Example :

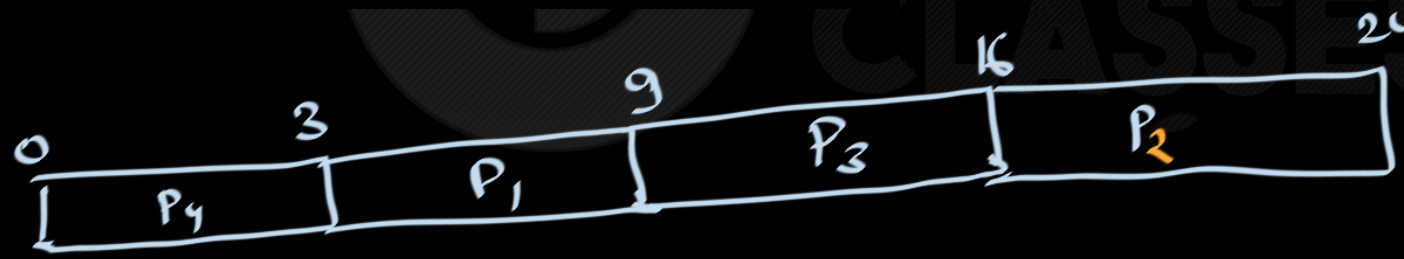
Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

arrival time = 0 for all processes

Example :

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

arrival time = 0 for all processes



Avg waiting time = $\frac{0 + 3 + 9 + 16}{4} = 7$ ms

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

- SJF scheduling chart

- Average waiting time = (3 + 16 + 9 + 0) / 4 = 7

Non-preemptive SJF: Example

Process	Duration	Arrival Time
P1	6	0
P2	8	0
P3	7	0
P4	3	0

Do it yourself

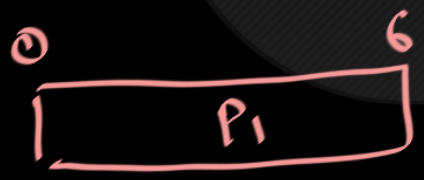
P4 waiting time: 0
P1 waiting time: 3
P3 waiting time: 9
P2 waiting time: 16

The total time is: 24
The average waiting time (AWT):
(0+3+9+16)/4 = 7

Comparing to FCFS

Process	Duration	Arrival Time
P1	6	0
P2	8	0
P3	7	0
P4	3	0

Do it yourself



Comparing to FCFS

Process	Duration	Arrival Time
P1	6	0
P2	8	0
P3	7	0
P4	3	0

Do it yourself

avg waiting time = $\frac{0 + 6 + 14 + 21}{4} = \frac{41}{4} = 10.25$

GO CLASSES

we have just seen that for one example SJF is giving lesser avg wait time than FCFS.

Question: will this always happen?

Comparing to FCFS

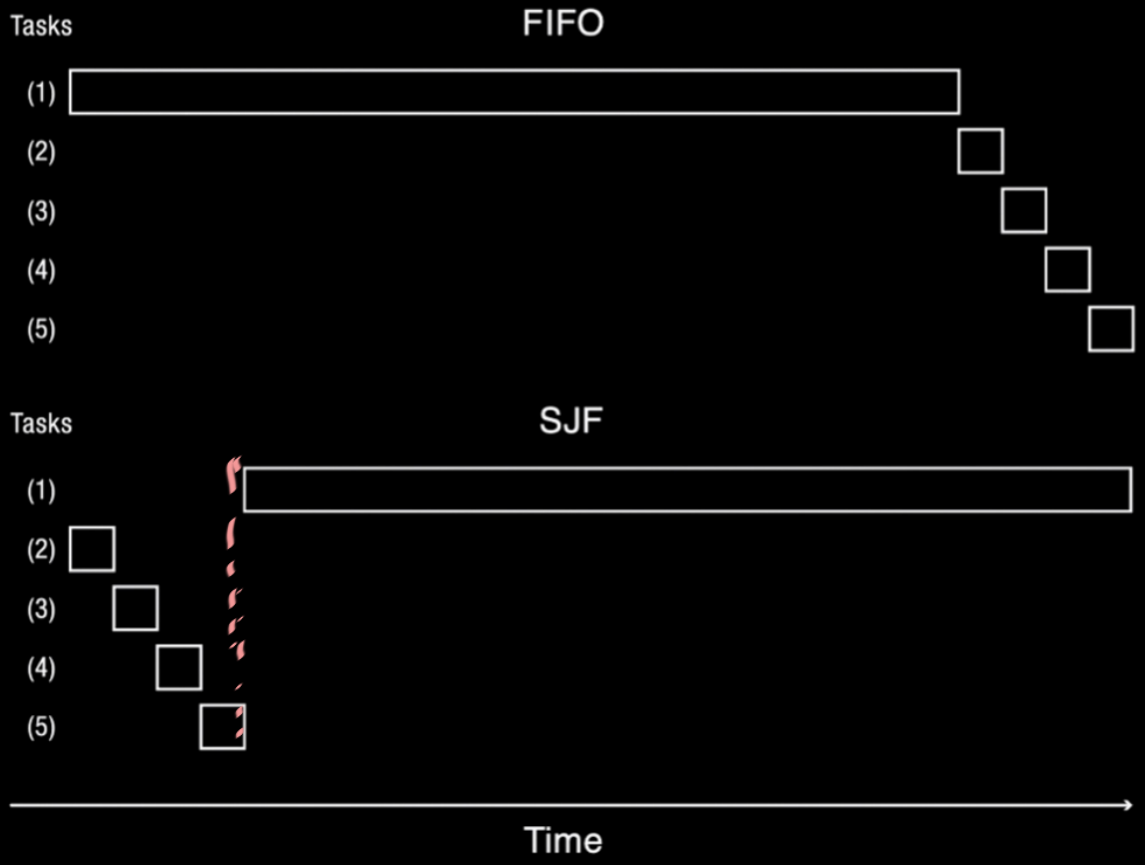
Process	Duration	Arrival Time
P1	6	0
P2	8	0
P3	7	0
P4	3	0

Do it yourself

P1 waiting time: 0
P2 waiting time: 6
P3 waiting time: 14
P4 waiting time: 21

The total time is the same.
The average waiting time (AWT):
(0+6+14+21)/4 = 10.25
(compared to 7)

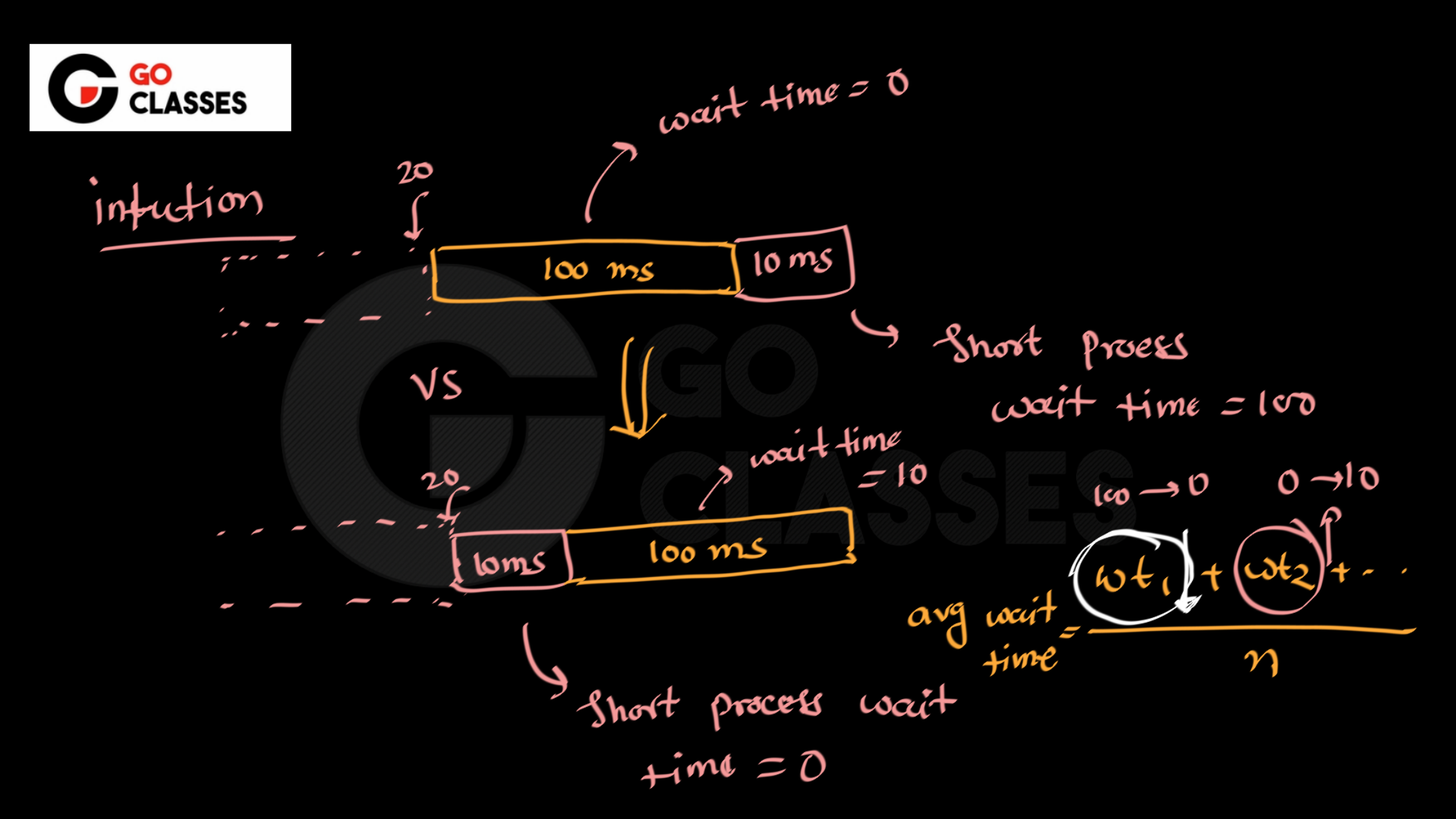
FIFO vs. SJF



- SJF is optimal – gives minimum average waiting time for a given set of processes
  - The difficulty is knowing the length of the next CPU request
  - Could ask the user

Intuition

- Moving a short process before a long one decreases the waiting time of the short process more than it increases the waiting time of the long process. Consequently, the average waiting time decreases.


GO CLASSES
intuition
20
100 ms
10 ms
wait time = 0
short process wait time = 100
VS
20
10ms
100 ms
wait time = 10
short process wait time = 0
100 → 0
0 → 10
avg wait time = wt1 + wt2 + . . . / n

GO CLASSES

OPTIONAL PROOF

SJF has scheduled in this order :-

P1 P2 P3 P4 P5

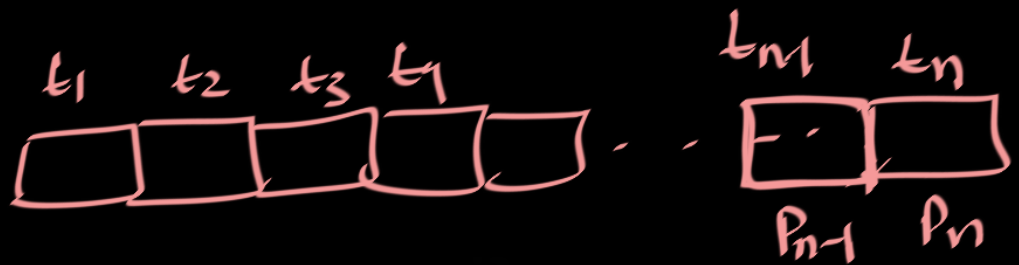
GO CLASSES

OPTIONAL PROOF

SJF has scheduled in this order :-

avg waiting time = $\dfrac{0 + t_1 + (t_1+t_2) + (t_1+t_2+t_3) + (t_1+t_2+t_3+t_4)}{5}$

(P_3 finishes at $t_1+t_2+t_3$)

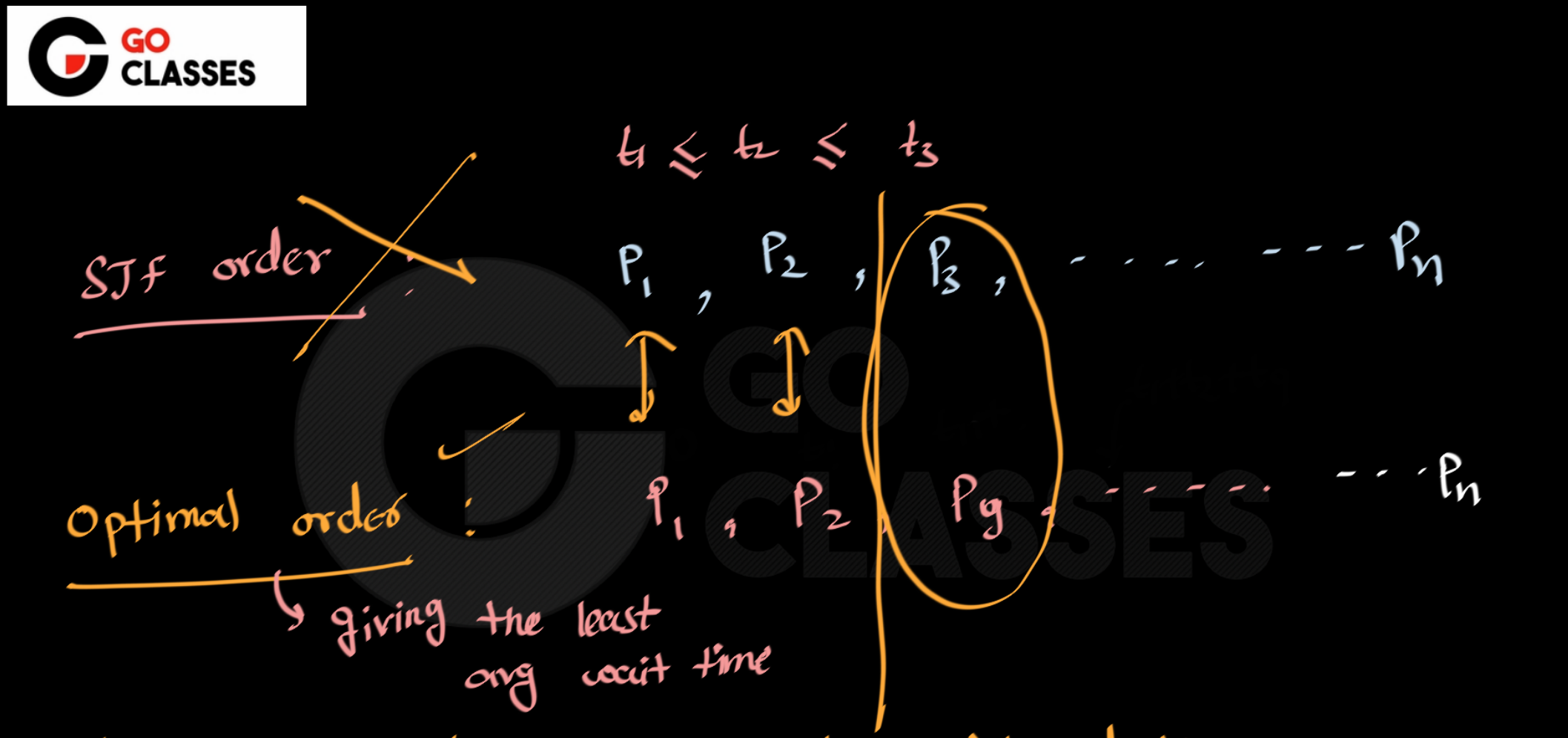


avg wait time for SJF

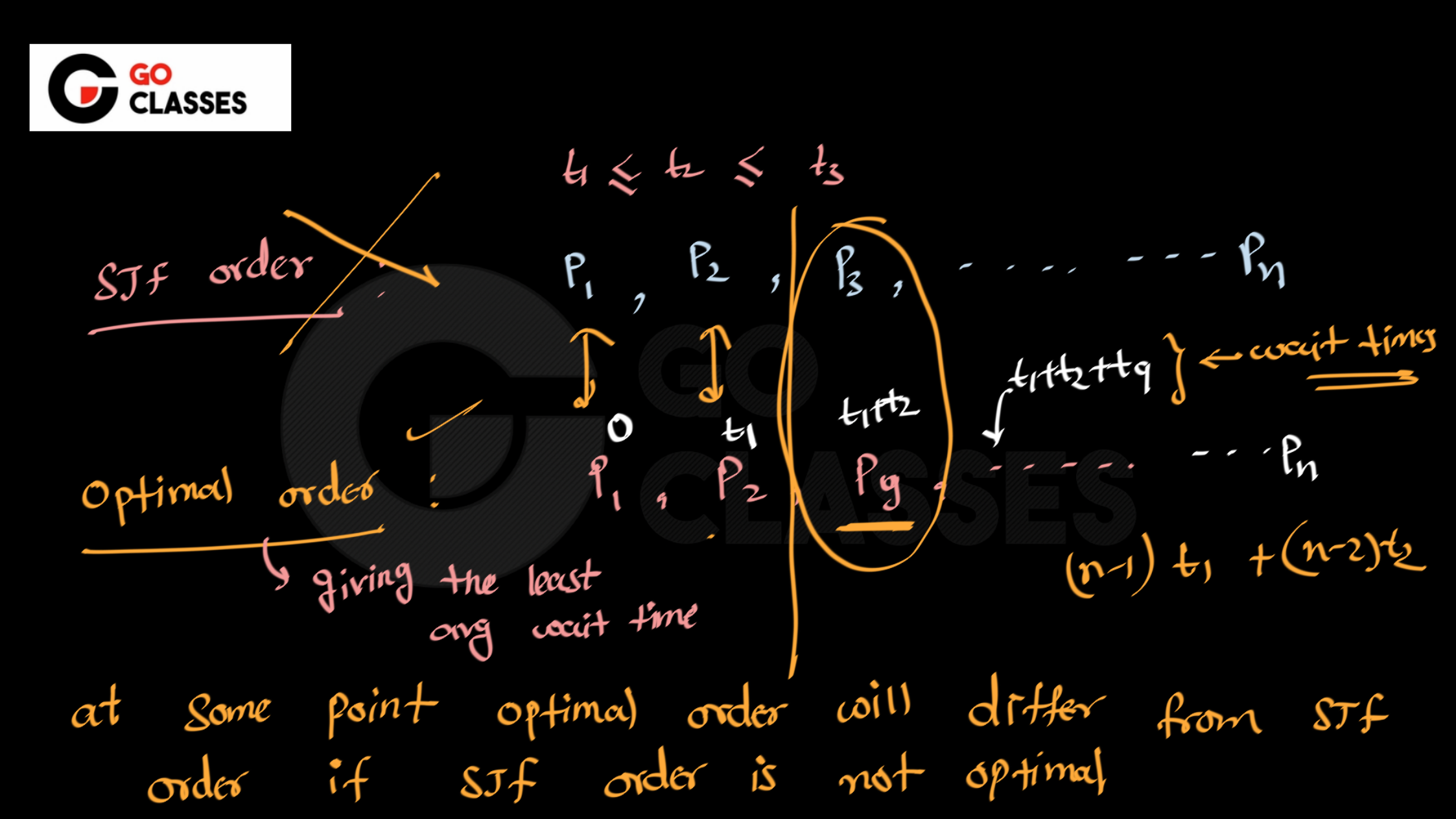
$$= \frac{0 + t_1 + (t_1+t_2) + (t_1+t_2+t_3) + (t_1+t_2+t_3+t_4)}{5}$$

$t_1 < t_2 < t_3$

$$= \frac{(n-1)t_1 + (n-2)t_2 + (n-3)t_3 + (n-4)t_4 + \cdots 1 \cdot t_{n-1}}{n}$$



at some point optimal order will differ from SJF order if SJF order is not optimal


GO CLASSES
$t_1 \le t_2 \le t_3$
SJF order
P_1 , P_2 , P_3 , ... P_n
0
t_1
t_1+t_2
$t_1+t_2+t_3$
← wait times
Optimal order :
P_1 , P_2 P_3 ..... P_n
giving the least avg wait time
$(n-1)\,t_1 + (n-2)t_2$
at some point optimal order will differ from SJF order if SJF order is not optimal

GO CLASSES

Exchange trick

$t_1 \leq t_2 \leq t_3$

SJF order

$P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9 \ldots P_n$

Optimal order :

$P_1, P_2, P_9, P_4, P_5, P_6, P_7, P_8, P_3 \ldots P_n$

giving the least avg wait time

0, t_1, t_1+t_2, t_4

$(n-1)t_1 + (n-2)t_2$

at some point optimal order will differ from SJF order if SJF order is not optimal

$t_1 \leq t_2 \leq t_3 \leq t_4 \leq$

Avg wait time of SJF $= \dfrac{(n-1)t_1 + (n-2)t_2 + (n-3)t_3 + (n-4)t_4 + (n-5)t_5 + \cdots + 1 \cdot t_{n-1}}{n}$

Avg wait time of optimal $= \dfrac{(n-1)t_1 + (n-2)t_2 + (n-3)t_3 + (n-4)t_4 \cdot t + \cdots (n-5)t_5}{n}$

$t_1 \leq t_2 \leq t_3 \leq t_4 \leq$

Avg wait time of SJF $= \dfrac{(n-1)t_1 + (n-2)t_2 + \boxed{(n-3)t_3} + (n-4)t_4 + \underbrace{(n-5)t_5 + \cdots 1 \cdot t_{n-1}}_{n}}{n}$

Avg wait time of optimal $= \dfrac{(n-1)t_1 + (n-2)t_2 + \boxed{(n-3)t_5} + (n-4)t_4 + \underbrace{(n-5)t_3 \cdots 1 \cdot t_{n-1}}_{n}}{n}$

$$= \begin{cases} \underbrace{(n-3)t_3 + (n-5)t_5}_{SJF} \leq \underbrace{(n-3)t_5 + (n-5)t_3}_{\text{optimal}} \end{cases}$$

GO CLASSES

Rishesh Tiwari to Everyone 7:37 PM

RT

how to find the job which is shortest?

[without Running the Jobs on cpu we can not find the Burst times.

GO CLASSES

[we need the burst times to implement SJF.

[we need to predict the burst time

⇓

the only logical way seems to use history of previous processes.

GO CLASSES

in COA, we predict

Branch taken or not taken.

SJF is NOT Practical to implement

because burst times are not known

↓

we can try to predict.

Example 2:

H.W.

find out Avg wait time and Avg turn arround time.

Process	Arrival Time (ms)	Processing Time (ms)
P1	8	3
P2	2	1
P3	1	3
P4	3	2
P5	4	4

| P3 | P2 | P4 | P5 | P1 |

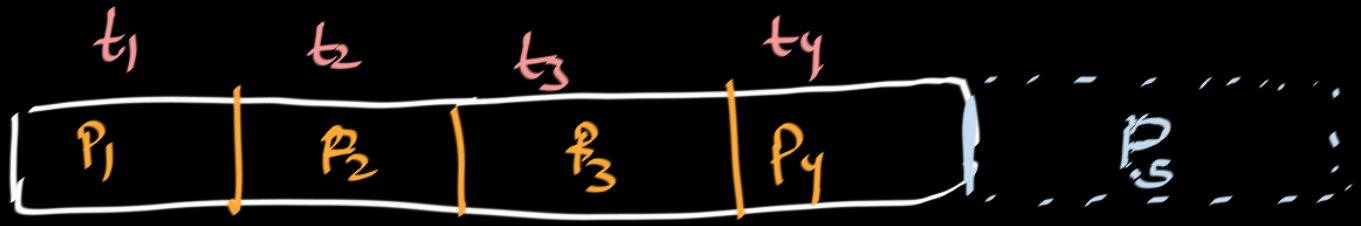
1 4 5 7 11 14

Average waiting time=(0+2+2+3+3)/5=2

Average Turnaround time=(3+3+4+7+6)/5=4.6

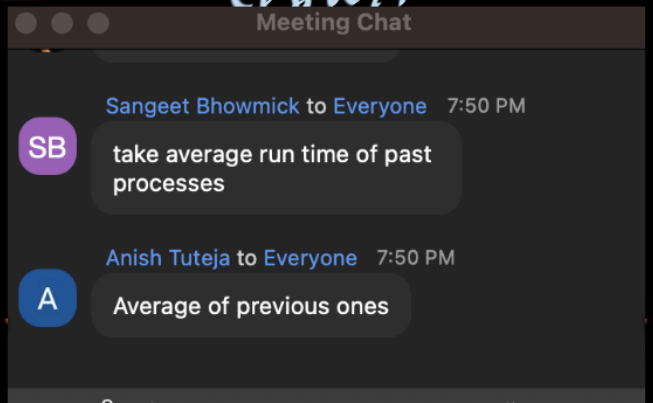
we have already run P_1, P_2, P_3, P_4.

Now want to "approximate" that for how long P_5 may run.

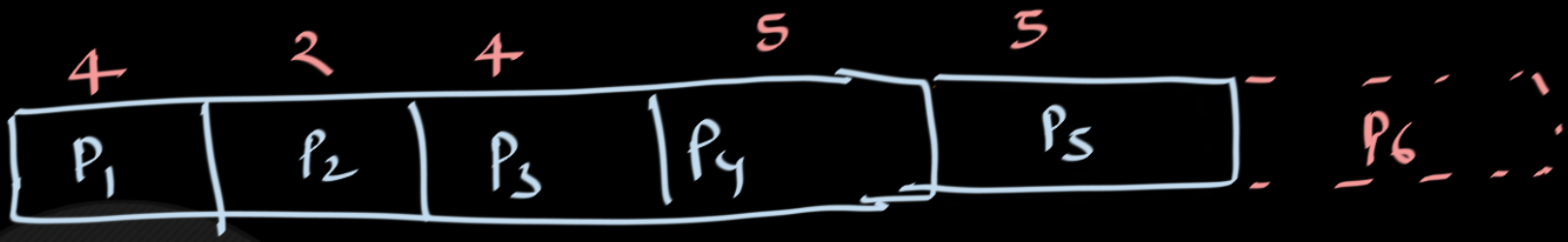


we have already run P_1, P_2, P_3, P_4.

Now want to "approximate" that for how long P_5 may run.



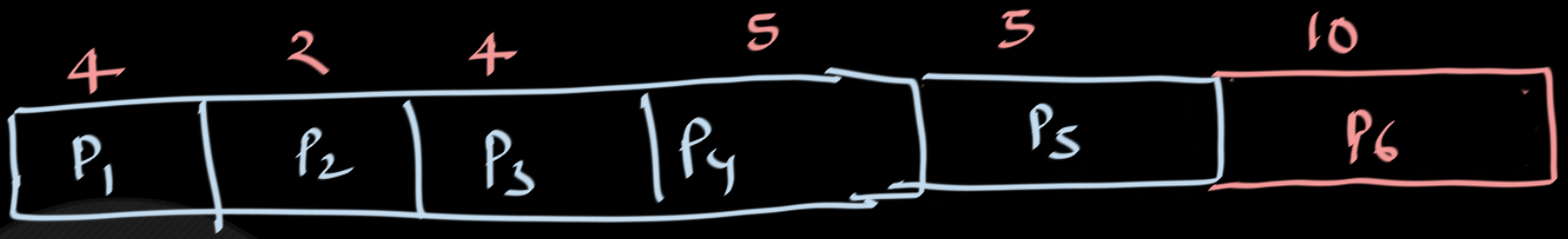
approximately P_5 will run for $\frac{t_1+t_2+t_3+t_4}{4}$ time



Approximately P_6 will run $= \frac{4+2+4+5+5}{5} = \frac{20}{5} = 4$

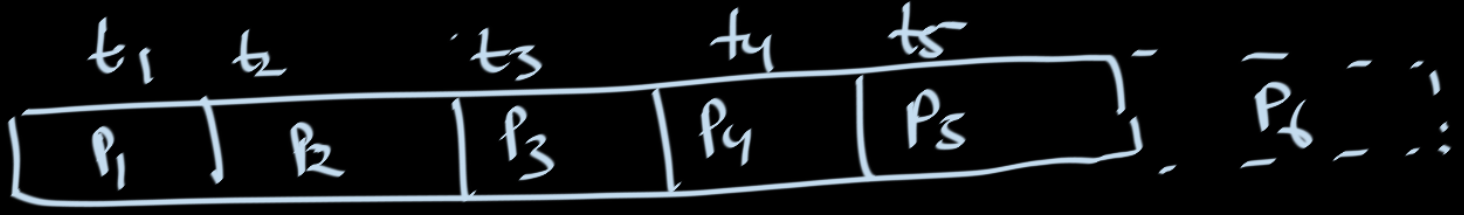
Predicted time for P_6

Actual time for P_6 = 10 ms.

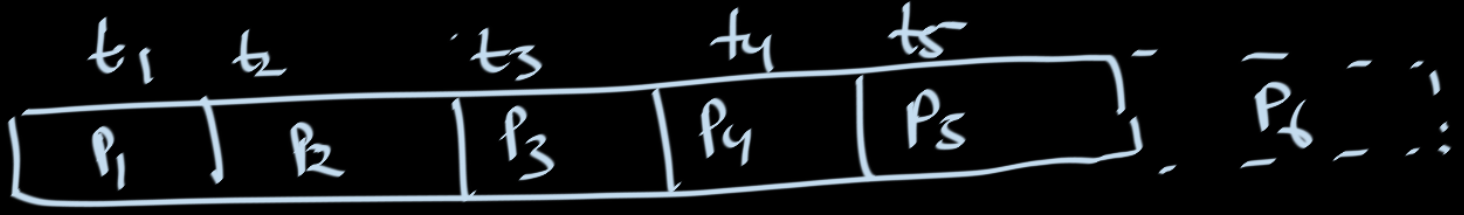


Approximately $= \frac{4+2+4+5+5+10}{6} = 5$

P_7 will run

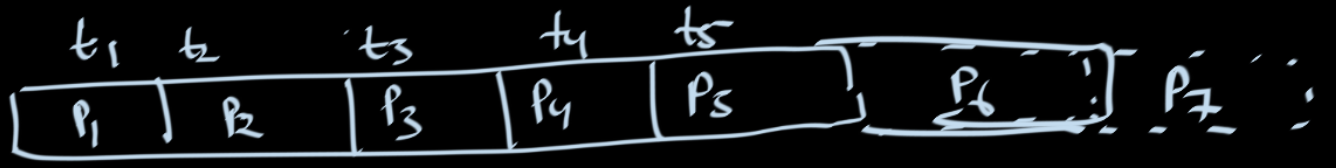


(Predicted time for P_6) $S_6 = \dfrac{t_1 + t_2 + t_3 + t_4 + t_5}{5}$



(Predicted time for P_6) $S_6 = \dfrac{t_1 + t_2 + t_3 + t_4 + t_5}{5}$

Actual time for $P_6 = t_6$



$$\left(\begin{array}{c}\text{Predicted}\\ \text{time for}\\ P_6\end{array}\right) \quad S_6 = \frac{t_1 + t_2 + t_3 + t_4 + t_5}{5}$$

Actual time for $P_6 = t_6$

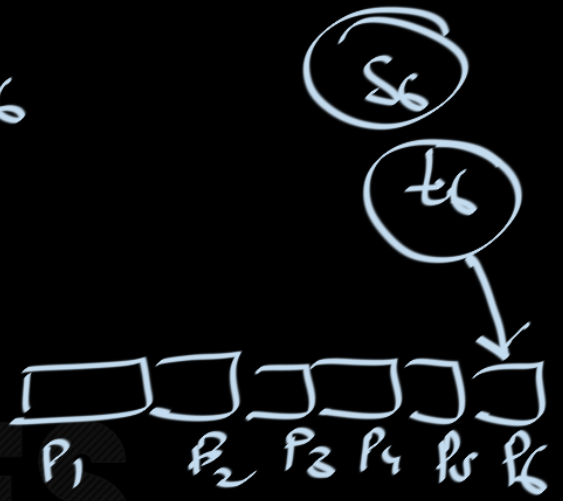
$$\left(\begin{array}{c}\text{Predict time}\\ \text{for } P_7\end{array}\right) \quad S_7 = \frac{t_1 + t_2 + t_3 + t_4 + t_5 + t_6}{6}$$

GO CLASSES

$S_6 = \dfrac{t_1 + t_2 + t_3 + t_4 + t_5}{5}$

$S_7 = \dfrac{t_1 + t_2 + t_3 + t_4 + t_5 + t_6}{6}$

$5\,S_6$



P_7

$S_7 = \dfrac{5\,S_6 + t_6}{6} = \dfrac{5}{6}(S_6) + \dfrac{1}{6}(t_6)$

S_6 → Actual time

t_6 → Actual time

$$S_7 = \frac{5}{6} S_6 + \frac{1}{6} t_6$$

If SAME

$$S_7 = \frac{t_1 + t_2 + t_3 + t_4 + t_5 + t_6}{6}$$

(moving average)

$$S_7 = \frac{5}{6} S_6 + \frac{1}{6} t_6$$

we are giving more weight to the previous PREDICTION and less weight to the previous REAL Burst time. ← it is ONE of the idea.

$$s_7 = \frac{5}{6} s_6 + \frac{1}{6} t_6$$

$$s_n = \frac{n-2}{n-1} s_{n-1} + \frac{1}{n-1} t_{n-1}$$

these weights are static.

$$S_7 = \frac{5}{6} S_6 + \frac{1}{6} t_6$$

$$S_n = \frac{n-2}{n-1} S_{n-1} + \frac{1}{n-1} t_{n-1}$$

these weights are static (S_{n-1} always have higher weight)

$$S_n = \alpha S_{n-1} + (1-\alpha) t_{n-1}$$

GO CLASSES

earlier, we were rigid in the sense that we were ALWAYS giving less weight to the recent actual time and more weight to the predicted time.

$$S_n = \alpha S_{n-1} + (1-\alpha) t_{n-1} \qquad 0 \leq \alpha \leq 1$$

(i) $\alpha = 0$ $\quad S_n = t_{n-1}$

(ii) $\alpha = 1$ $\quad S_n = S_{n-1}$

$$S_n = \alpha S_{n-1} + (1-\alpha)\, t_{n-1} \qquad 0 \le \alpha \le 1$$

$$S_{n-1} = \alpha S_{n-2} + (1-\alpha)\, t_{n-2}$$

$$S_n = \alpha S_{n-1} + (1-\alpha)\, t_{n-1} \qquad 0 \le \alpha \le 1$$

$$S_{n-1} = \alpha S_{n-2} + (1-\alpha)\, t_{n-2}$$

$$\Rightarrow S_n = \alpha \left(\alpha S_{n-2} + (1-\alpha)\, t_{n-2} \right) + (1-\alpha)\, t_{n-1}$$

$$S_n = \alpha S_{n-1} + (1-\alpha)\, t_{n-1} \qquad 0 \le \alpha \le 1$$

$$S_{n-1} = \alpha S_{n-2} + (1-\alpha)\, t_{n-2}$$

$$\Rightarrow S_n = \alpha \left(\alpha S_{n-2} + (1-\alpha)\, t_{n-2} \right) + (1-\alpha)\, t_{n-1}$$

$$\Rightarrow S_n = \alpha^2 S_{n-2} + \alpha (1-\alpha)\, t_{n-2} + (1-\alpha)\, t_{n-1}$$

$$S_n = \alpha^2 S_{n-2} + \underbrace{\alpha(1-\alpha)\, t_{n-2} + (1-\alpha)\, t_{n-1}}$$

$$\Rightarrow S_n = \alpha^3 S_{n-3} + \underbrace{\alpha^2(1-\alpha)\, t_{n-3} + \alpha(1-\alpha)\, t_{n-2} + (1-\alpha)\, t_{n-1}}$$

$$\alpha^n S_0 + \text{GP}(1-\alpha)\, t_n$$

$0 \le \alpha \le 1$

$0 \le 1-\alpha \le 1$

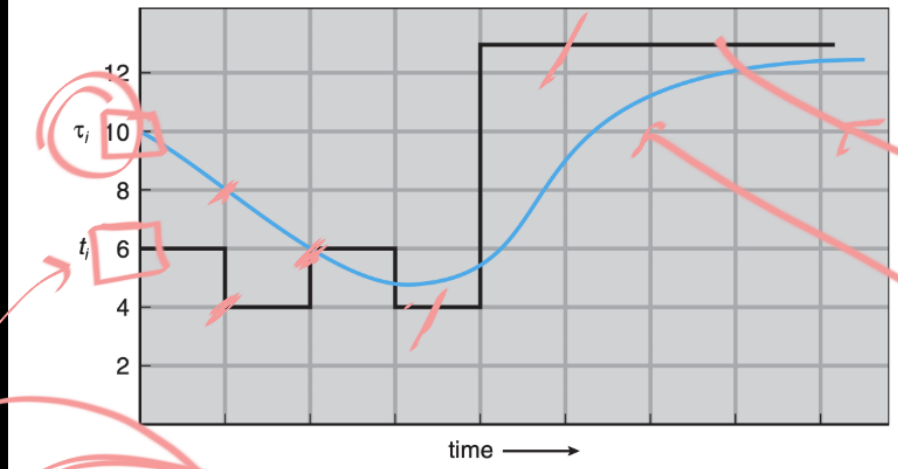
$2^3 < 2^5$

$\left(\frac{1}{2}\right)^3 > \left(\frac{1}{2}\right)^5$

$$S_n = \alpha^2 S_{n-2} + \alpha(1-\alpha) t_{n-2} + (1-\alpha) t_{n-1}$$

$$\Rightarrow S_n = \alpha^3 S_{n-3} + \alpha^2(1-\alpha) t_{n-3} + \alpha(1-\alpha) t_{n-2} + (1-\alpha) t_{n-1}$$

this formula gives more weight to the recent time, and lesser and lesser weights to the previous times.



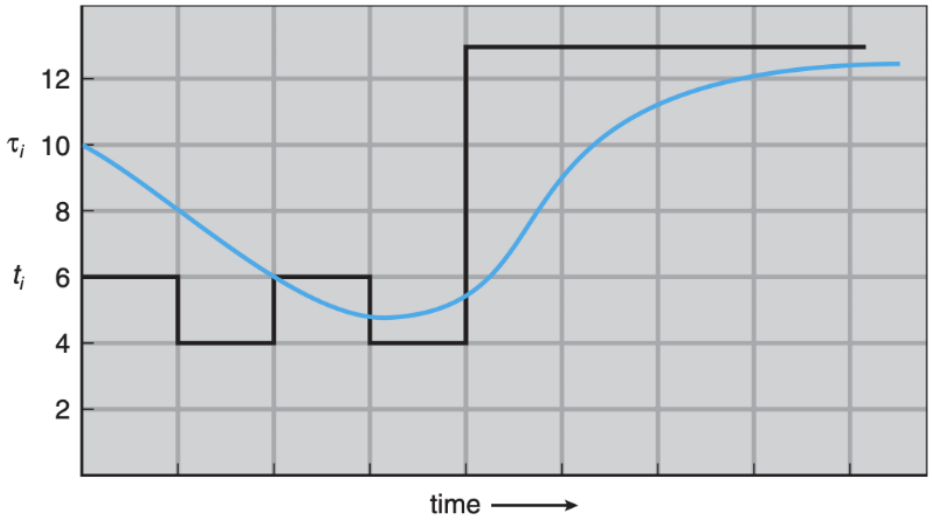
CPU burst (t_i)	6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	5	9	11	12	...

Figure 6.3 Prediction of the length of the next CPU burst.

Random

Actual time

Guess



CPU burst (t_i)	6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	5	9	11	12	...

Figure 6.3 Prediction of the length of the next CPU burst.

GO CLASSES

SJF :

- SJF is optimal for Avg wait time
- SJF is not practical as we don't know burst times in advance
- we can predict Burst time using

$$S_n = \alpha S_{n-1} + (1-\alpha)\, t_{n-1}$$

GO CLASSES

Sujith to Everyone 8:27 PM

Sir one small doubt, curious to know whether is there someway Short term Scheduler can find Time complexity of process to guess the Burst TIme as this method wldnt need the process to be executed, and we also wld be able to compare it

Same as SJF