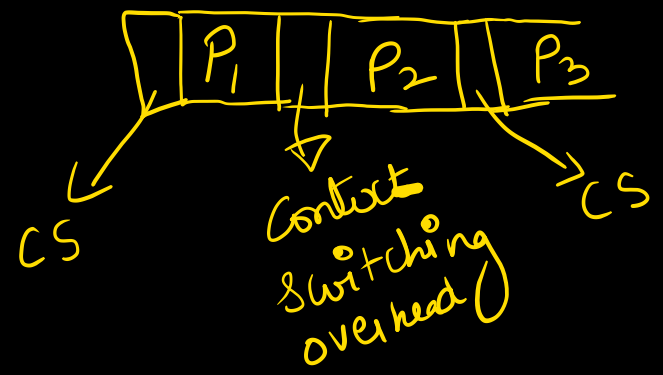
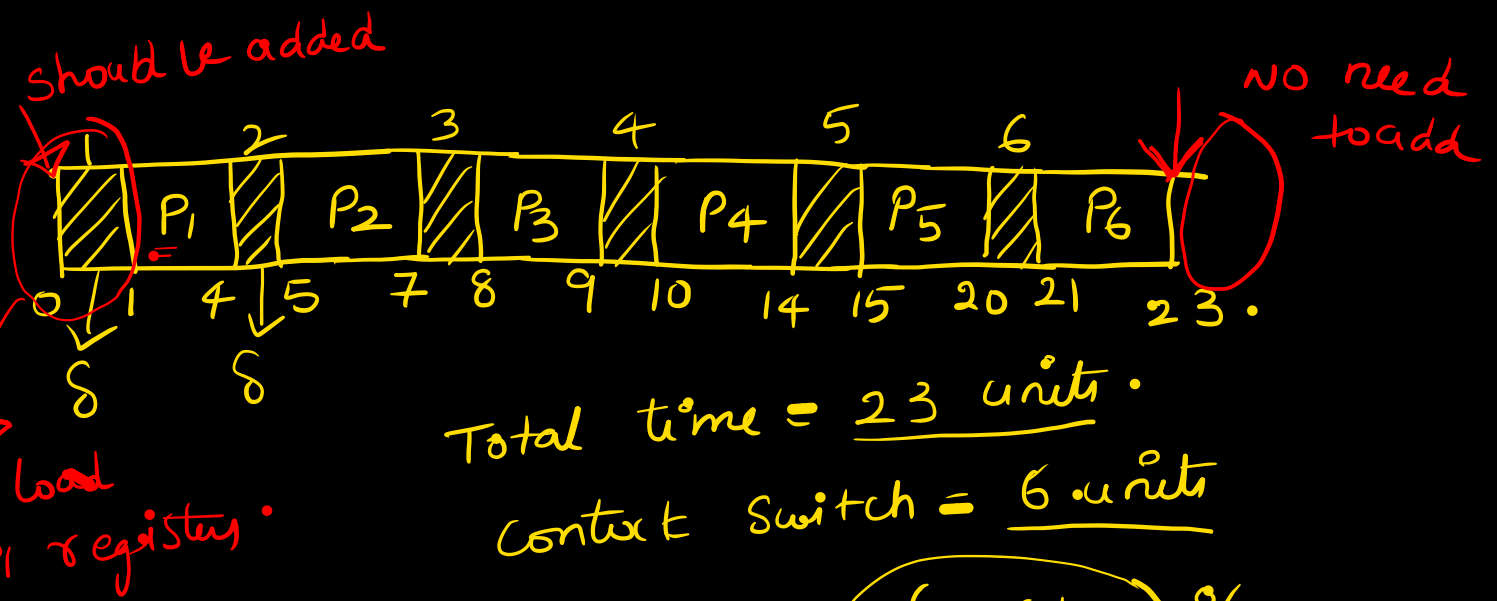


Context switching is a overhead.

Let do an ex on Contx switching in FCFS

FCFS with Context switching overhead. Let Context Switching time δ = 1 unit

PNO	AT	BT
1	0	3
2	1	2
3	2	1
4	3	4
5	4	5
6	5	2



Total time = 23 units.

Context Switch = 6 units

$\therefore$ %efficiency = $\frac{6}{23} \times 100$ %.

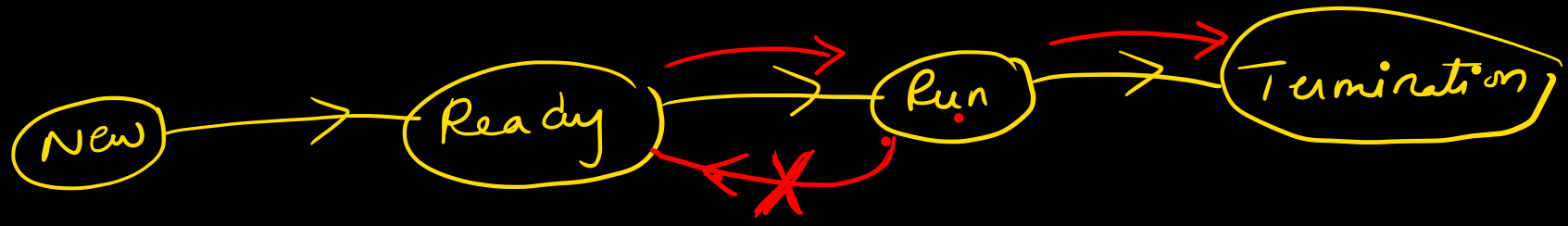
efficiency = $\frac{17}{23} \times 100$ %.

Shortest Job First (SJF)

Out of all processes available in ready state, we will schedule the process will shortest burst time.

Criteria: Burst time → we decide based on burst time.

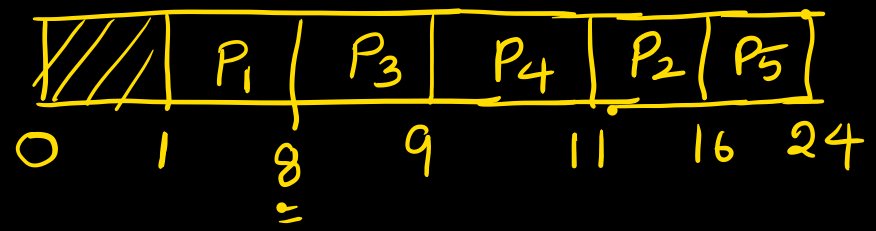
mode: non preemptive



ignore wait/block.

Gate:

PNO	AT	BT	CT	TAT	WT
1	1	~~7~~ 1	8	7	0
2	2	~~5~~ 5	16	14	9
3	3	~~1~~ 1	9	6	5
4	4	~~2~~ 2	11	7	5
5	5	8	24	19	11



what is avg WT?

we will do all parameters

$$\text{Avg WT} = \frac{9+5+5+11}{5} \Rightarrow$$

Consider context switching time only if it is mentioned

SJF also has Convoy effect:

PNO	AT	BT	CT	TT	WT
1	0	20	20	20	0
2	1	1	21	20	19
3	2	1	22	20	19

Avg WT = $\frac{38}{3}$ = 12.83 → very high.
Because big process arrived first.

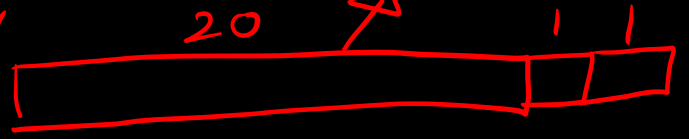
P1	P2	P3

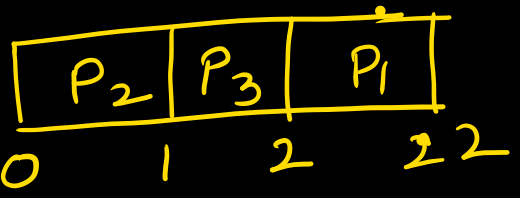
0 20 21 22

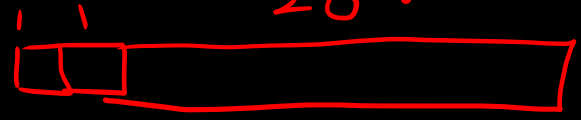
when BT is same → see arrival time (lower) → PNO (low)

PNO	AT	BT	CT	TT	WT
1	2	20	22	20	0
2	0	1	1	1	0
3	1	1	2	1	0

Avg WT = 0

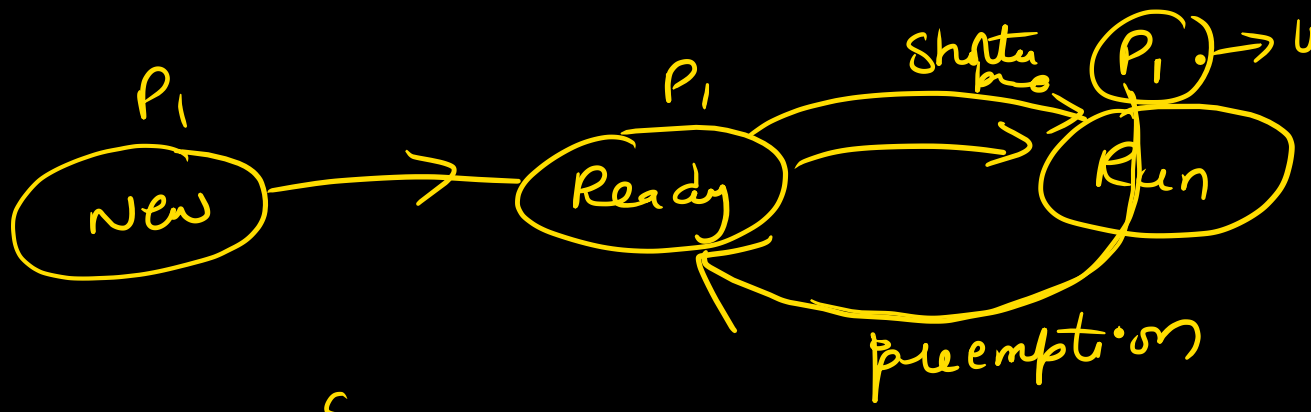


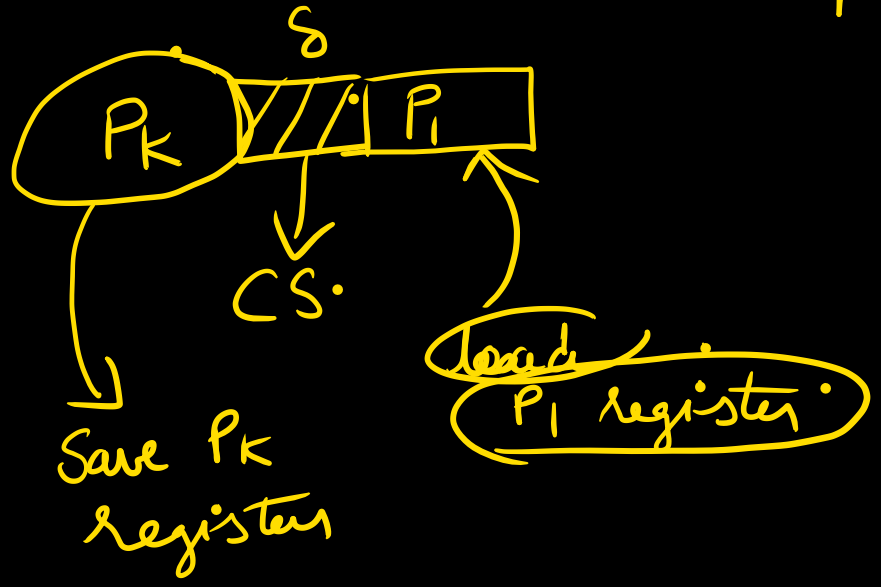




Preemption can solve Convoy effect.

But FCFS and SJF are non preemptive algo.





FCFS

↓

arrival time for scheduling

↓

if arrival time is same

↓

See the process number (lower)

SJF ⟶ Be careful, check BT of processes which have arrived.

↓

Burst time for scheduling.

↓

if BT is same

↓

see AT (lower)

↓ AT is same

↓

Process id. (low)

SJF → Based on Burst time

↓

Time needed by a process to complete

↓

We Cant know that before we execute

↓

But SJF needs it before we execute

↓

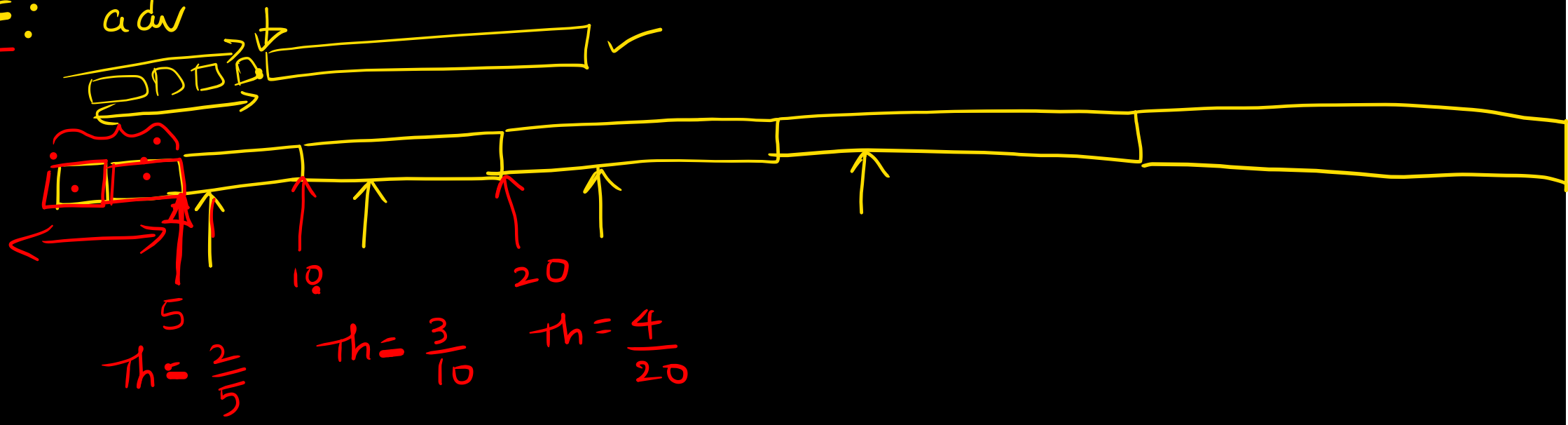
∴ SJF is not practical.

Disad:

SJF tells us what best is possible.
So we use SJF to benchmark (compare) performance with other scheduling algo

Any scheduling algo based on BT cannot be practically implemented.
To overcome this, we use some prediction methods.
We can't know exact BT but we can guess.
It may be right along.

SJF: adv



All scheduling algo will have same thoupat at the end.

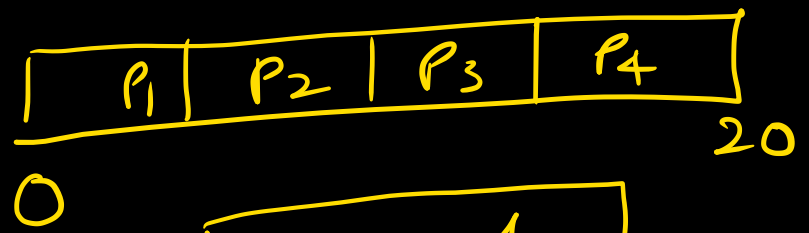
Thoupat of SJF is very high in between & at any point during execution.

Similary: Avg TAT and Avg WT of SJF is minimum except convoy case.

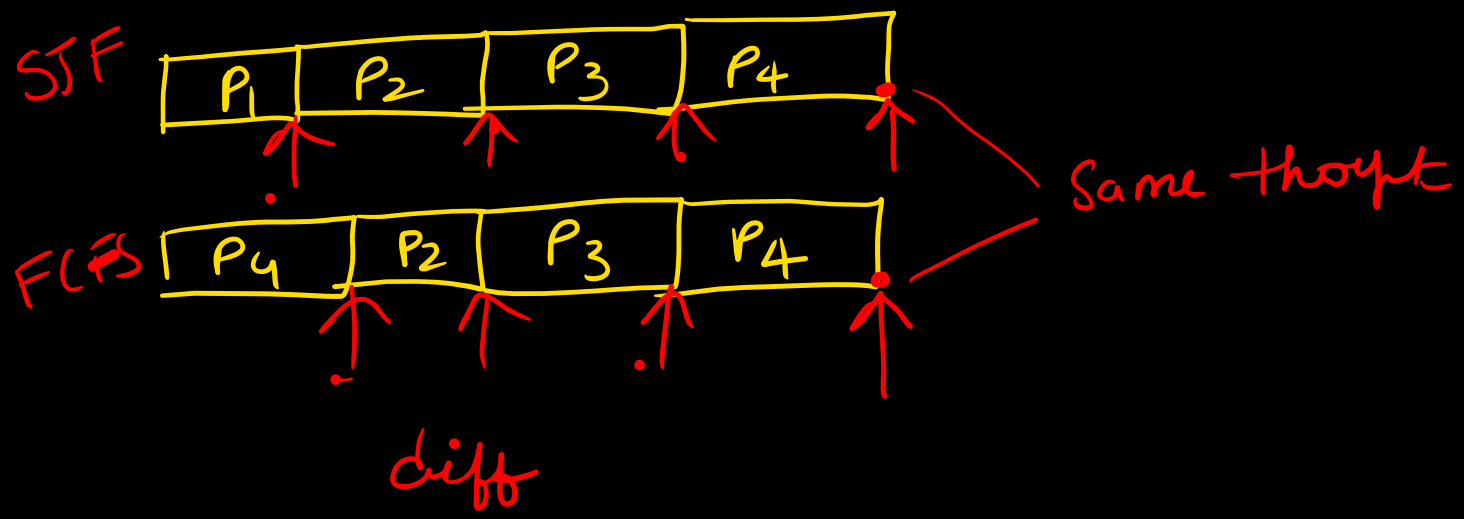
Through put = num of process completed per second.

$$= \frac{\text{Total no of processes}}{\text{Schedule length}}$$

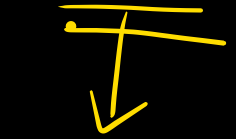
Ex:



$$Th = \frac{4}{20}$$


SJF
P1
P2
P3
P4
FCFS
P4
P2
P3
P4
Same thought
diff

STF is best algo but useless.



we can't use it practically.

To make algorithms based on BT work, people have invented BT prediction.

algorithms → like SJF

P_1 → Before running → Guess or how long it will take!!!

Guess → Prediction.

Prediction techniques:

- Static
  1. Process size ✓
  2. Process type ✓
- Dynamic
  1. simple averaging
  2. Exponential averaging

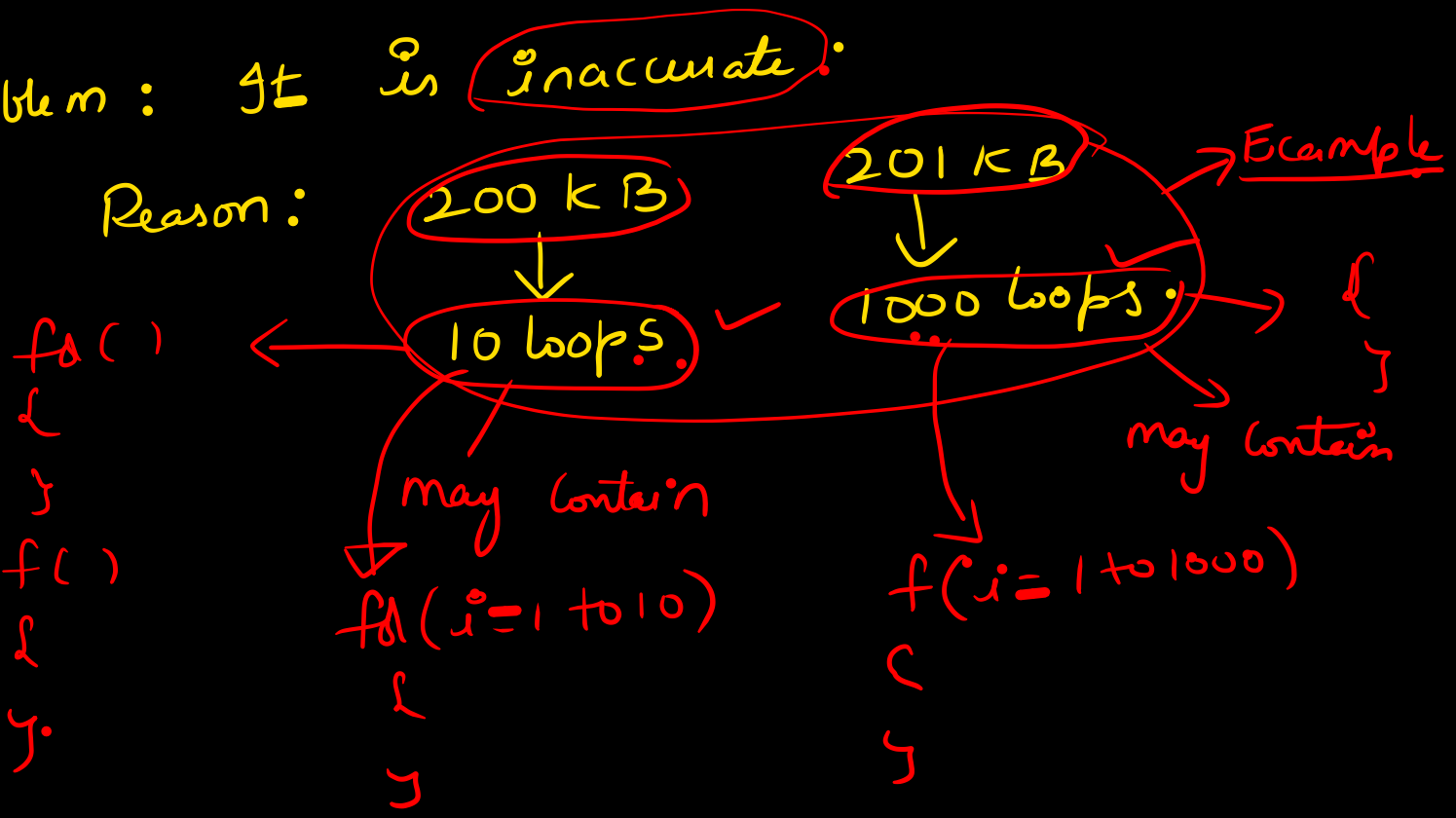
4 methods

1) Process size : Based on past experience, we will guess BT.

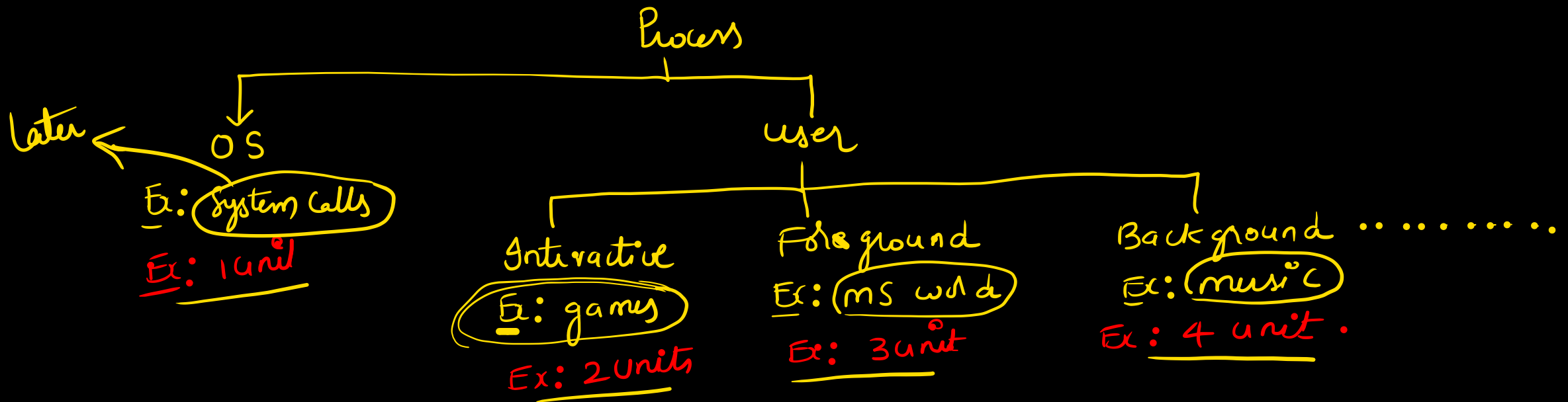
Ex: In past, Process of 200 KB took 20 unit.
If we get a process of 201 KB → guess → 20 unit.

Problem : It is inaccurate.

Reason :



2) Process type: we classify processes into various types and we give BT for each type.



This is also inaccurate. X

Simple average:

Let us assume we executed n processes $(P_1, P_2, P_3 \cdots P_n)$ and we know exactly their BT.

P_1	P_2	P_3	P_n
1	10	20	30

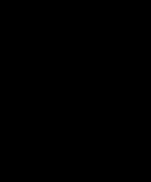
Then we guess $BT(P_{n+1}) = \dfrac{BT_1 + BT_2 + BT_3 \cdots BT_n}{n}$.

$$\frac{1 + 10 + 20 \cdots + 30}{n}$$

This is also not accurate.

Reason is $(n+1)^{th}$ process can run for anytime independent of previous processes.

These methods are used just to make algo like SJF work.
Only theoritically. At practice mainly Round Robin is used



Exponential averaging:

$$\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$$

- τ_{n+1} → Predicted BT of n+1
- t_n → actual BT of n (after leaving the process)
- τ_n → Predicted value of n.

P_n → Prediction = 10 units = τ_n
→ actual = 20 units = t_n.

Ex:

$$\tau_{n+1} = \alpha(20) + (1-\alpha)(10)$$

α → Smoothening factor.

$$0 \le \alpha \le 1$$ ✓

Gate: $\alpha = 0.5$, $\tau_1 = 10$, actual BT $(t_1, t_2, t_3, t_4) = (4, 8, 6, 7)$, $\tau_5 = ?$

$\tau_2 = 0.5 \times 4 + 0.5 \times 10 = 7$

$\tau_3 = 0.5 \times 8 + 0.5 \times 7 = 7.5$

$\tau_4 = 0.5 \times 6 + 0.5 \times 7.5 = 6.75$

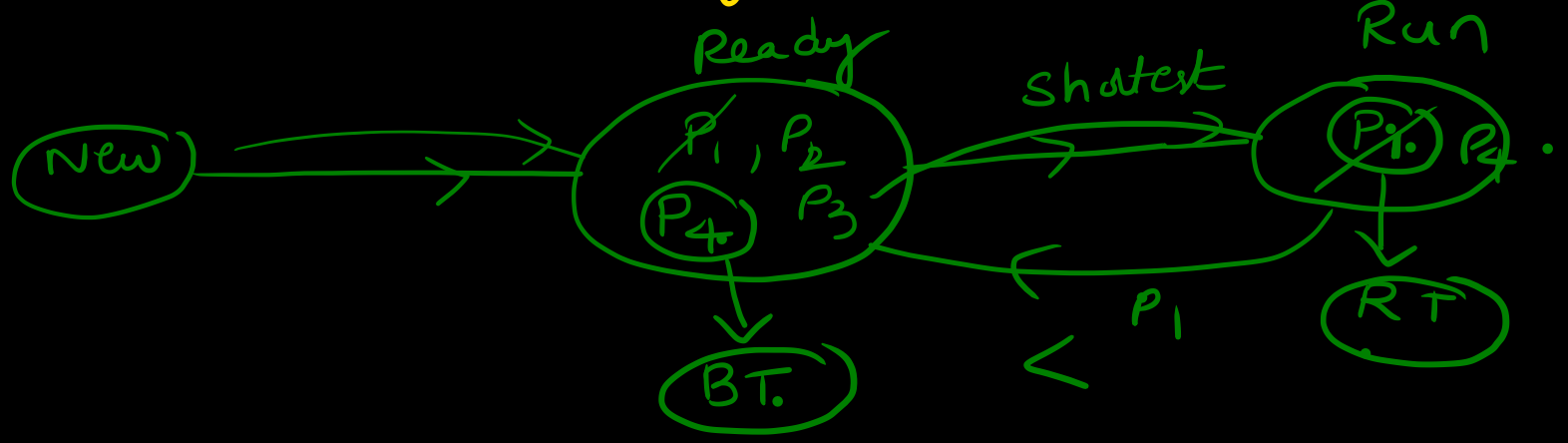
$\tau_5 = 0.5 \times 7 + 0.5 \times 6.75 = 6.875$ ✓

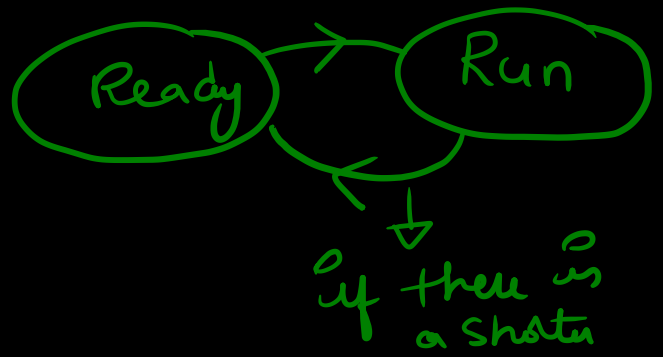
Shortest Remaining Time First: (SRTF)

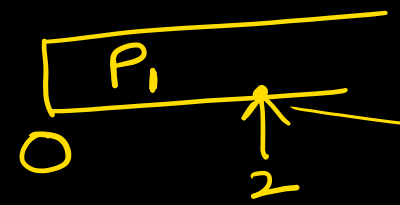
→ modification of SJF

Criteria → Burst time → when BT is same → AT (lower)

mode → Preempting.







Remaining Time of P_1

Burst time of P_2

$P_2 \Rightarrow AT = 2$

RT of P_1 < BT of P_2

↓

P_1 will continue

if RT of P_1 > BT of P_2

↓ ↓

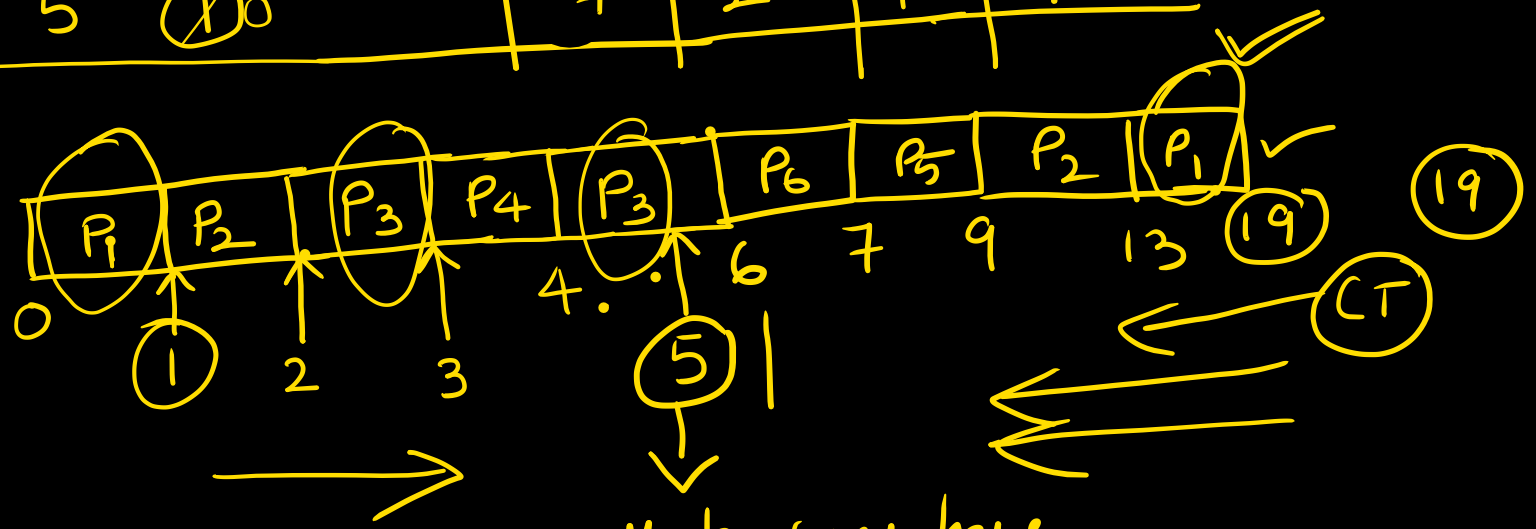
Preempted P_2 will be schedule

Gate: what is Completion time of P_1 in SRTF

PNO	AT	BT	CT	TAT	WT	RT
1	0	~~7~~ 6	19	19	12	0
2	1	~~5~~ 4	13	12	~~7~~ 0	0
3	2	~~3~~ ~~2~~ ~~1~~ 0	6	4	1	0
4	3	~~1~~	4	1	0	0
5	4	~~2~~	9	5	3	3
6	5	~~1~~ 0	7	2	1	1

one more example

ans WT = 4



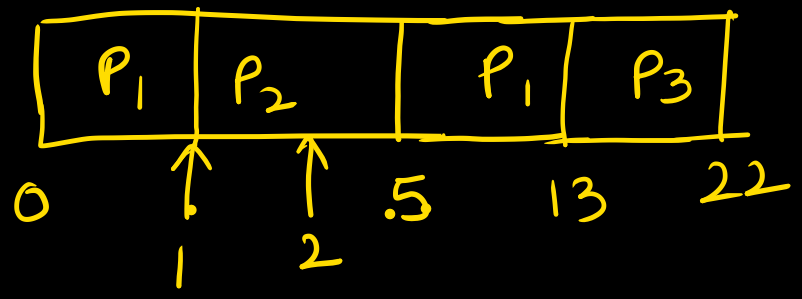
all process have arrived. So it becomes STF

Gate 2011

PNO	AT	BT	CT	TAT	WT
1.	0	~~9~~ ~~8~~ 0	13	13	4
2	(1)	~~(4)~~ ~~(3)~~ 0	5	4	0
3.	(2)	9.	22	20	11

what is avg waiting time using SRTF?

$\frac{15}{3} = 5.$

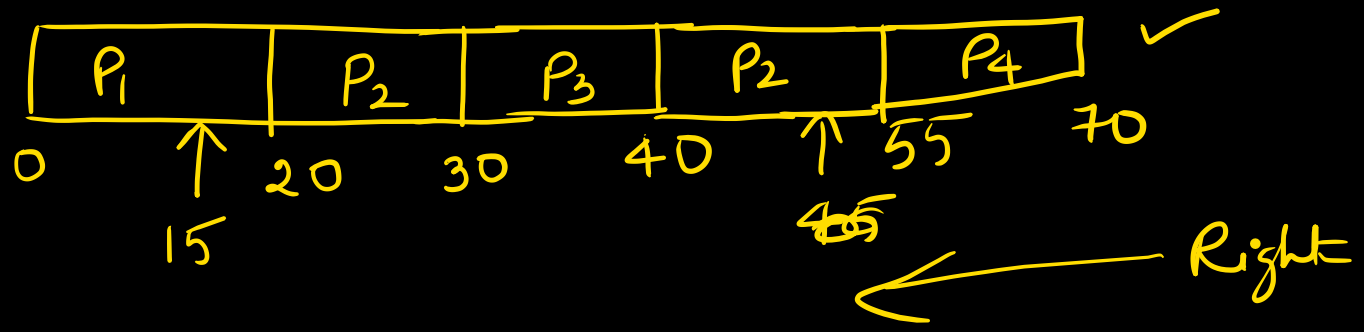


Gate 07: waiting time of P_2 using SRTF

PNO	AT	BT		CT	TAT	WT
1	0	~~20~~	~~50~~	20	20	0
2	15	~~25~~	~~15~~ 100	55	40	15
3	30	~~10~~		40	10	0
4	45	~~15~~		70	25	10

15

R

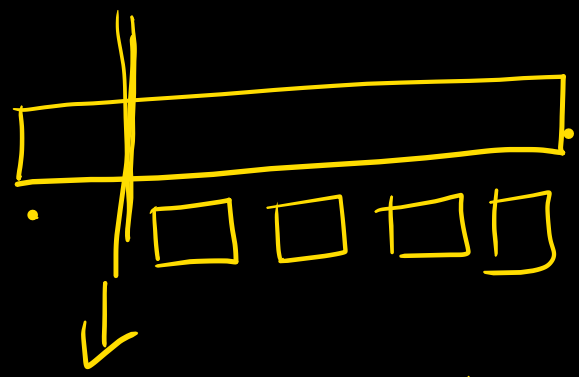


SRTF → better than SJF

Because no Convoy effect.

SRTF, SJF

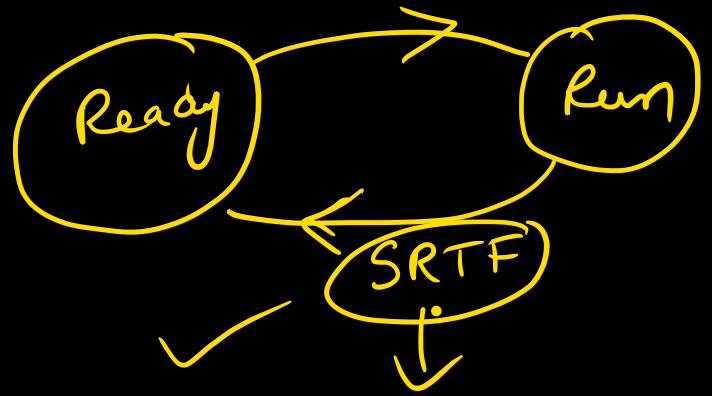
→ longer job — Starvation.



Convoy

→ Starvation.

✗ Practical ✗



Preemption

BT cannot be known