COSI 131a: Operating Systems

CPU Scheduling (2)

Chapter 6

Agenda

1. Scheduling Overview
   1. *First Example of Resource (CPU) Management (Sharing)*
   2. *Non-Preemptive (N) vs. Preemptive Scheduling (P)*
   3. *Metrics: Ways to Assess Effectiveness of Scheduling Policies*

2. Scheduling Policies

   1. *First-Come-First-Served (FCFS) (N)*
   2. *Shortest-Job-First (SJF) (N, P)*

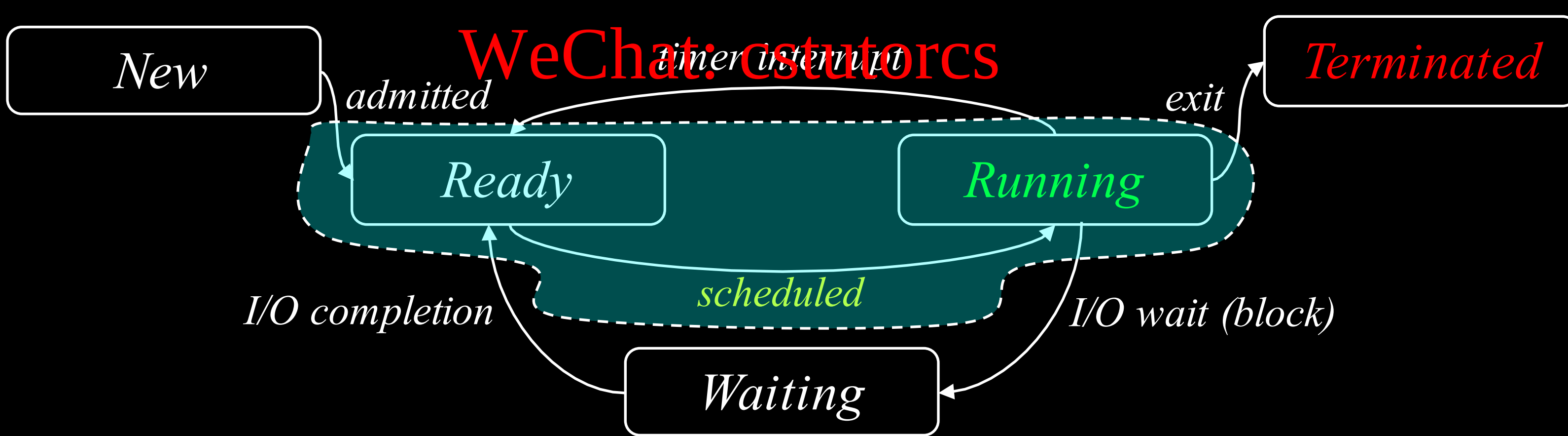
   3. *Priority (N, P)*
   4. *Round-Robin (RR) (P)*
   5. *Multilevel Queues (MLQ) (P) , Lottery*
   6. *Real-time*

3. Examples

Review: CPU Scheduler

CPU (Short-Term) Scheduler

- *first example of resource <u>manager</u> (manages sharing of CPU)*
- *applicable to either multiprocessing or multithreading (we'll assume former)*
- *What? A kernel process that is always active (daemon)*
- *responsible for choosing process to run from ready queue*



Review: CPU Scheduling

Two Types of Scheduling:

Non-Preemptive: *processes only give up CPU voluntarily*

- *simpler to implement (no timer interrupts)*
- *greedy or buggy process can starve others*

Preemptive: *processes also may be preempted by an interrupt*

- *adds complexity*
- *adds protection, better at ensuring fairness, as well as doing better in other scheduling metrics*

Review: Scheduling Metrics

Can’t Meet All Performance Goals With 1 Policy

$\therefore$ *Pick 1 or 2 that matter most*

Typically Important Metrics:

1. *Throughput / Turnaround time*:

   Throughput = n processes / sec $\Rightarrow$ *Turnaround = 1/n sec per process*

2. *Waiting time*

3. *Response time*: *(especially for highly interactive systems)*

Also Care About:

1. *Dispatch time*:

   Time it takes to choose next running process including schedule time + context switch time. If lengthy, effects effectiveness of scheduler

2. *Fairness*: *Want to avoid process starvation*

Review: First-Come First-Serve

Non-Preemptive Policy

Example

Process	CPU Burst Time
P_1	24
P_2	3
P_3	3

Suppose processes arrive in order: P_1; P_2; P_3

(*Gantt Chart*)

Waiting Time

P_1 : 0
P_2 : 24
P_3 : 27

Average Wait Time = 17

CPU Scheduling Policies Summary

Policy	*Throughput*	*Waiting*	*Response*	*Fairness*	*Overhead*	*Comments*
FCFS	✗	✗	✗	✓	✓	

Schedule Policy #2:

SJF: Shortest Job First

2a. Shortest-Job-First (SJF)

Non-Preemptive Policy

Idea: *Rank processes by CPU time requests.*
Optimizes (minimizes) average waiting time

Example

Process	CPU Burst
P_1	24
P_2	3
P_3	3
P_4	7

Average wait = (0 + 3 + 6 + 13) / 4 = 5.5

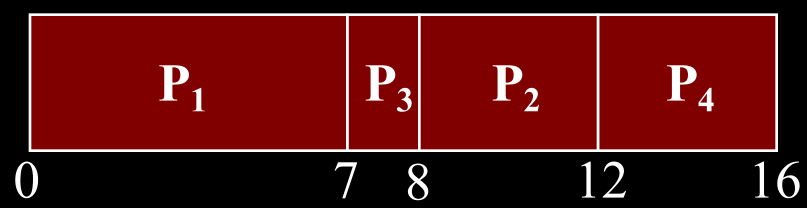
2a. Shortest-Job-First (SJF)

Q: What if processes don't arrive at same time?

A: Whenever process finishes, choose next process with shortest CPU burst

Example

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



Average wait time: ?

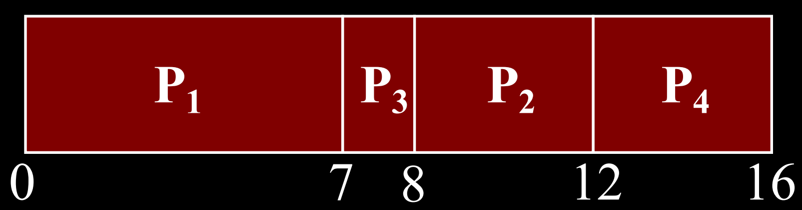
2a. Shortest-Job-First (SJF)

Q: What if processes don't arrive at same time?

A: *Whenever process finishes, choose next process with shortest CPU burst*

Example

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



Average wait time: (0+6+3+7) / 4 = 4

Preemptive version of SJF does even better …

2b. Preemptive SJF

Preemptive Version of SJF

Idea: *Currently running process can be preempted if new process arrives with shorter remaining CPU burst*

Example

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

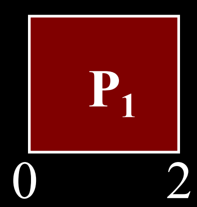
2b. Preemptive SJF

Preemptive Version of SJF

Idea: *Currently running process can be preempted if new process arrives with shorter remaining CPU burst*

Example

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



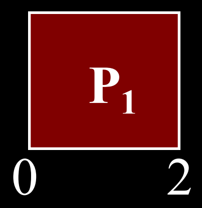
2b. Preemptive SJF

Preemptive Version of SJF

Idea: *Currently running process can be preempted if new process arrives with shorter remaining CPU burst*

Example

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



$P_1 = 5$, $P_2 = 4$

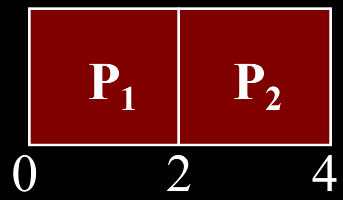
2b. Preemptive SJF

Preemptive Version of SJF

Idea: *Currently running process can be preempted if new process arrives with shorter remaining CPU burst*

Example

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



$P_1 = 5,\ P_2 = 4$

$P_1 = 5,\ P_2 = 2,\ P_3 = 1$

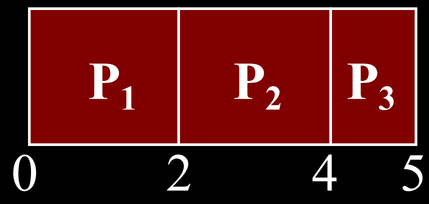
2b. Preemptive SJF

Preemptive Version of SJF

Idea: *Currently running process can be preempted if new process arrives with shorter remaining CPU burst*

Example

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



$P_1 = 5,\ P_2 = 4$

$P_1 = 5,\ P_2 = 2,\ P_3 = 1$ $P_1 = 5,\ P_2 = 2,\ P_4 = 4$

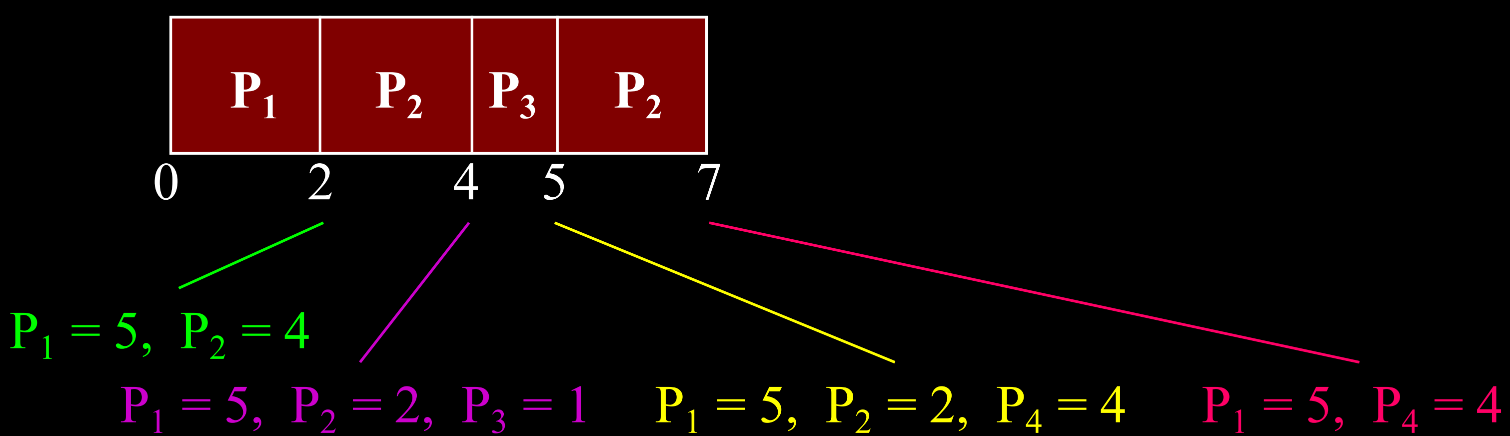
2b. Preemptive SJF

Preemptive Version of SJF

Idea: *Currently running process can be preempted if new process arrives with shorter remaining CPU burst*

Example

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



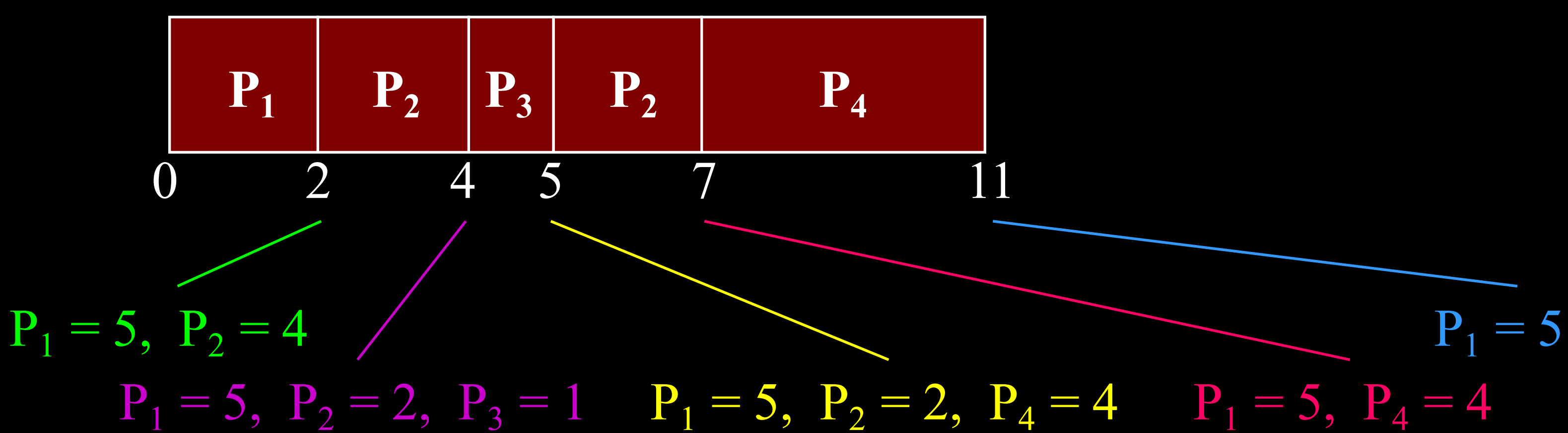
2b. Preemptive SJF

Preemptive Version of SJF

Idea: *Currently running process can be preempted if new process arrives with shorter remaining CPU burst*

Example

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



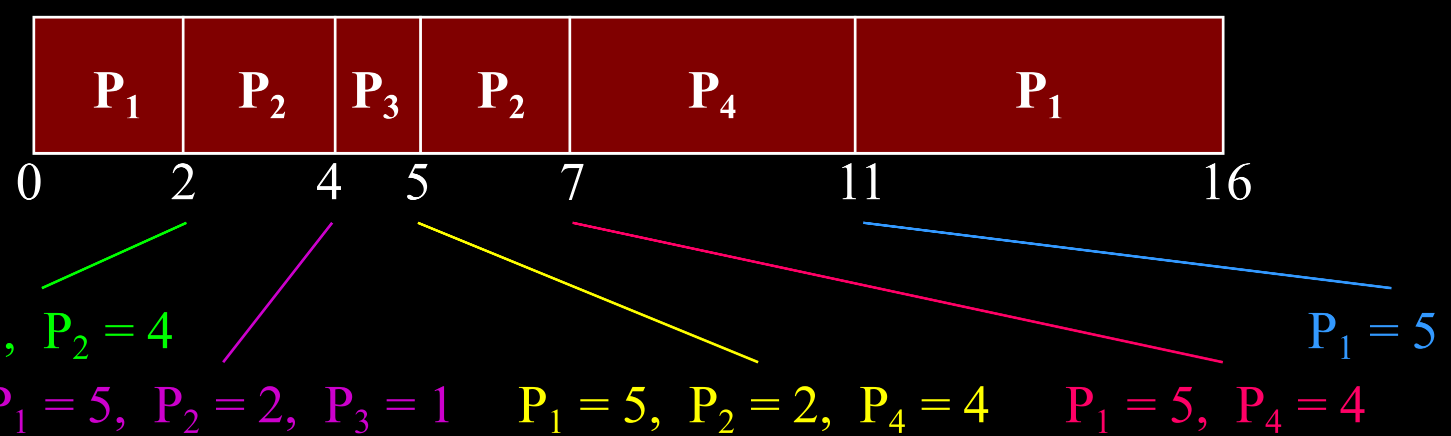
2b. Preemptive SJF

Preemptive Version of SJF

Idea: *Currently running process can be preempted if new process arrives with shorter remaining CPU burst*

Example

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4



2b. Preemptive SJF

Example:

Process	Arrival Time	CPU Burst
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

Average wait time:

P_1 : 9
P_2 : 1
P_3 : 0
P_4 : 2

Average = 3

(Compare to non-preemptive SJF average = 4)

2a, 2b: SJF

Issue: Predicting how much is next CPU burst

Exponential Averaging

Uses "history" of previous CPU bursts to predict length of next CPU burst

t_n: *Actual length of nth CPU burst*

τ_{n+1}: *Predicted length of next CPU burst*

α: *"Weight":*

→ *How much emphasis to put on recent past*
→ $0 \leq \alpha \leq 1$

Policies # 2a,2b: SJF (cont.)

Exponential Averaging

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

τ_{n+1}: *next guess*

τ_n: *previous guess (history)*

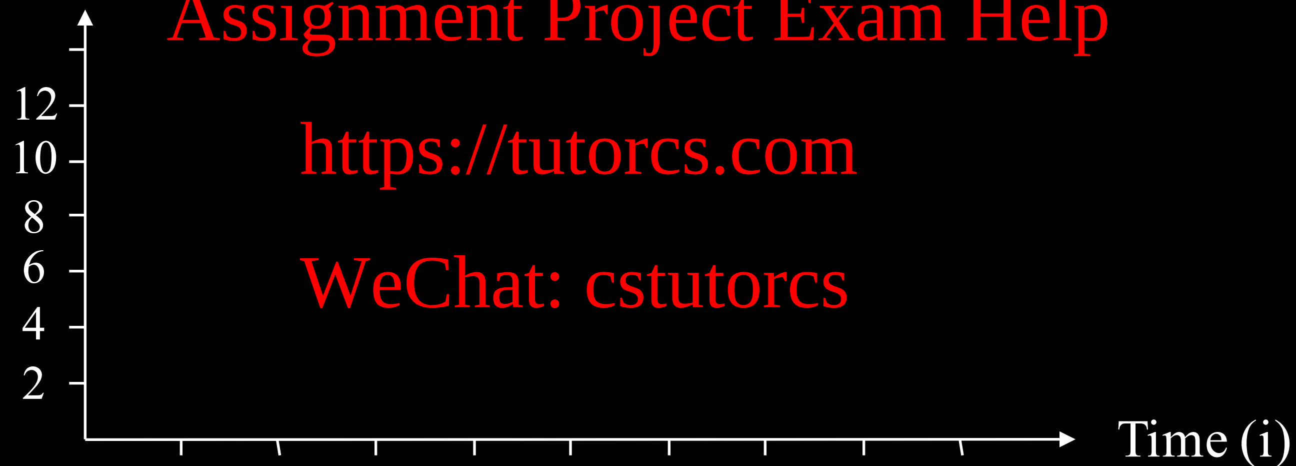
t_n: *previous actual burst time (most recent information)*

α: *weighting factor (O $\leq \alpha \leq$ 1)*

Policies # 2a,2b: SJF (cont.)

Exponential Averaging (cont.)

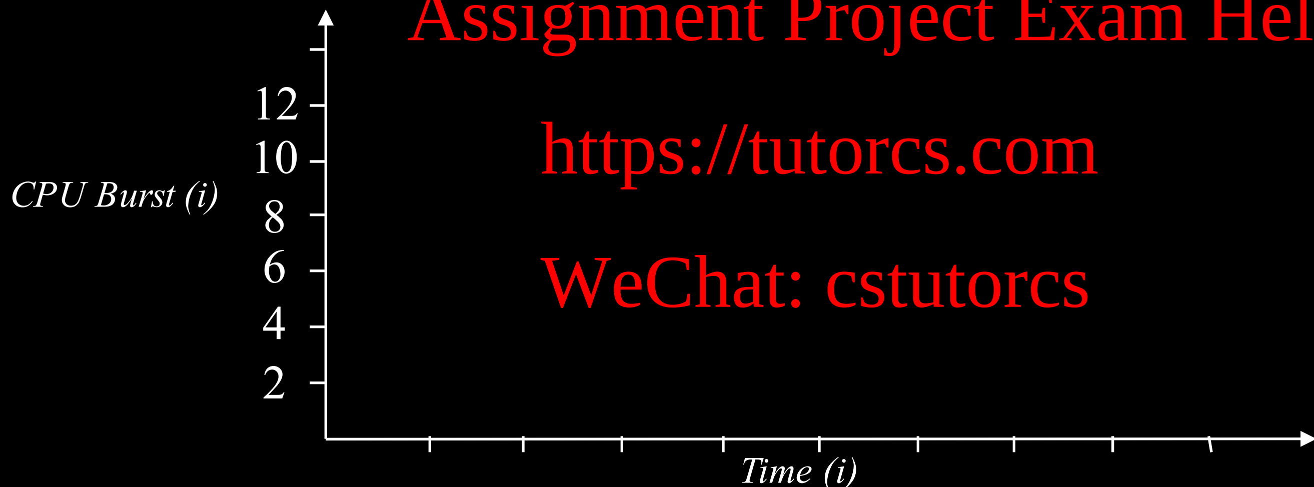
Example: $\alpha = 0.5$ $\tau_{n+1} = (\alpha \times t_n) + ((1 - \alpha) \times \tau_n)$



Policies # 2a,2b: SJF (cont.)

Exponential Averaging (cont.)

Example: $\alpha = 0.5$ $\qquad \tau_{n+1} = (0.5 \times t_n) + (0.5 \times \tau_n)$

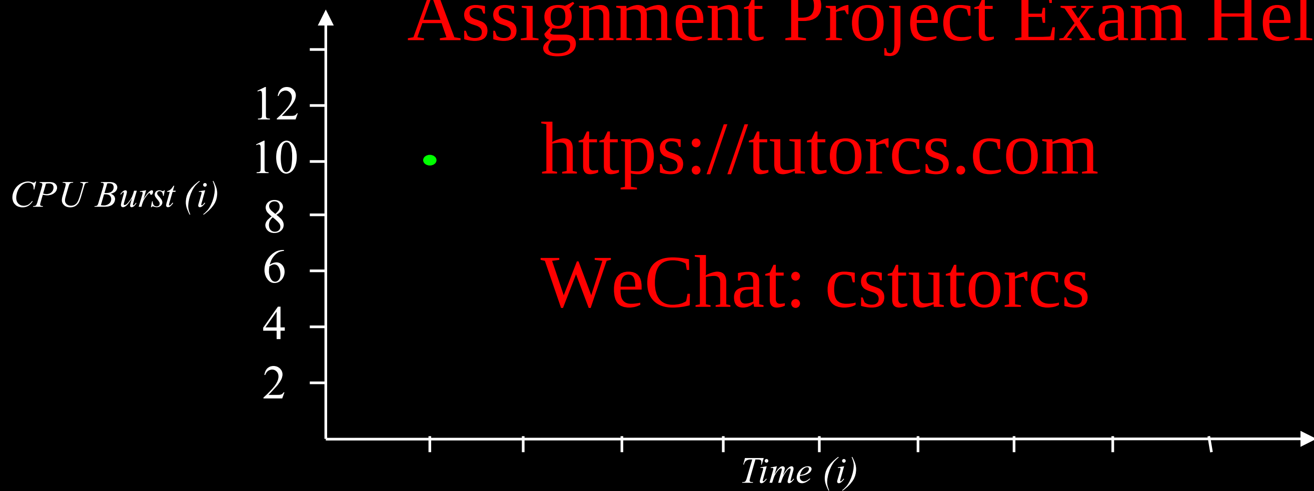


Actual (t_i):	
Guess (τ_i):	

Policies # 2a,2b: SJF (cont.)

Exponential Averaging (cont.)

Example: $\alpha = 0.5$ $\tau_{n+1} = (0.5 \times t_n) + (0.5 \times \tau_n)$

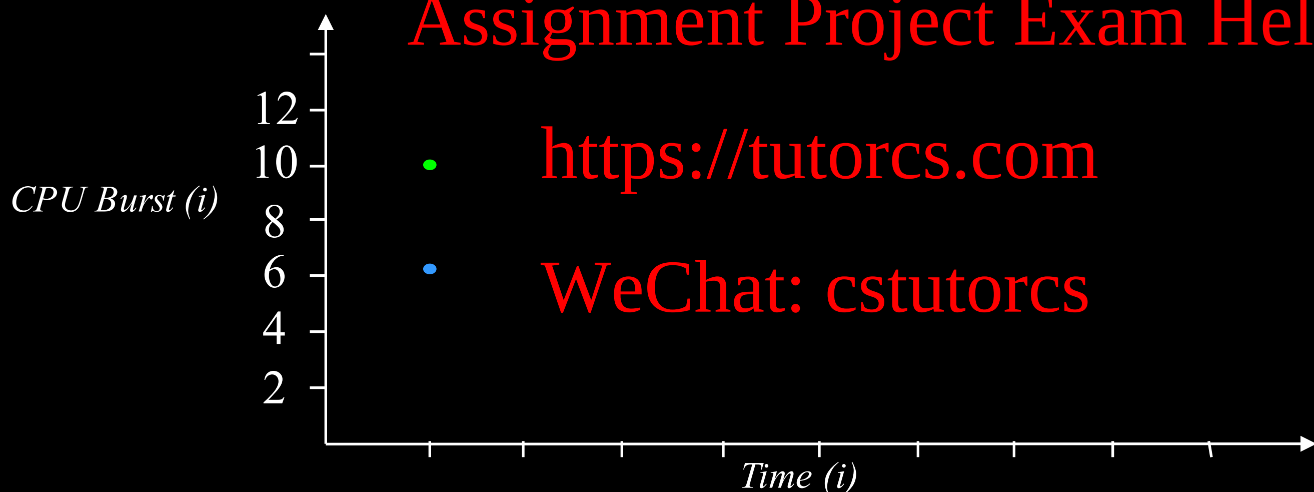


Actual (t_i):	
Guess (τ_i):	10

Policies # 2a,2b: SJF (cont.)

Exponential Averaging (cont.)

Example: $\alpha = 0.5$ $\qquad \tau_{n+1} = (0.5 \times t_n) + (0.5 \times \tau_n)$

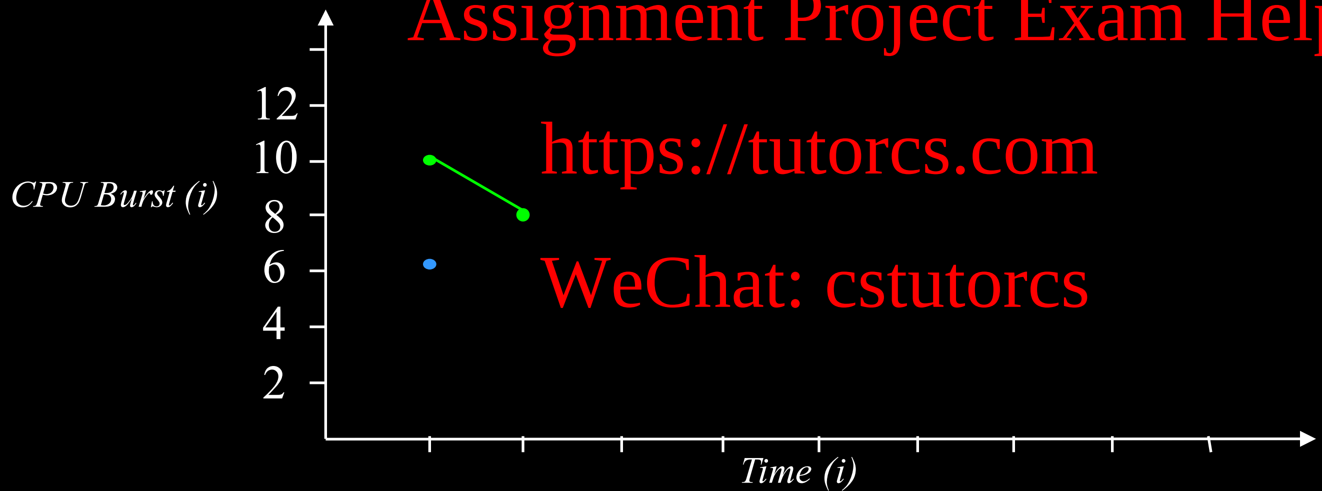


Actual (t_i):	6
Guess (τ_i):	10

Policies # 2a,2b: SJF (cont.)

Exponential Averaging (cont.)

Example: $\alpha = 0.5$ $\tau_{n+1} = (0.5 \times t_n) + (0.5 \times \tau_n)$

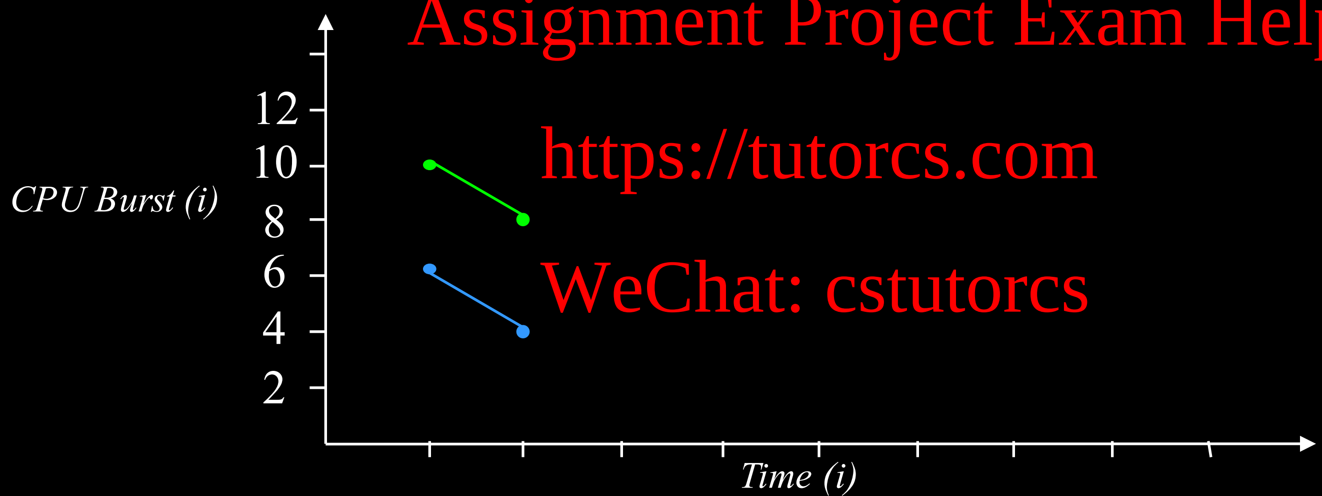


Actual (t_i):	6	
Guess (τ_i):	10	8

Policies # 2a,2b: SJF (cont.)

Exponential Averaging (cont.)

Example: $\alpha = 0.5$ $\qquad \tau_{n+1} = (0.5 \times t_n) + (0.5 \times \tau_n)$

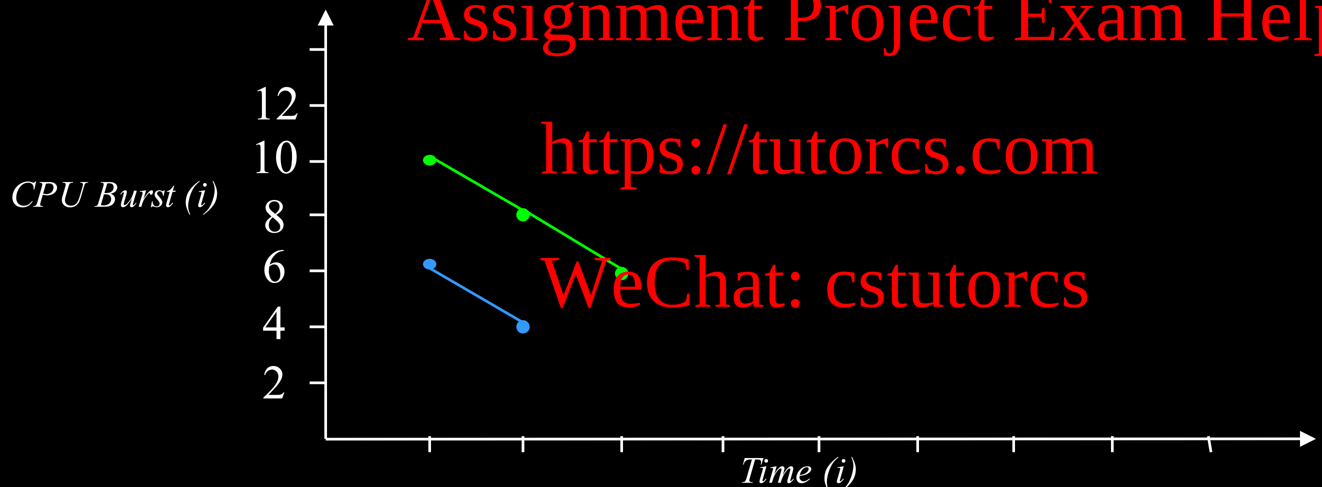


Actual (t_i):	6	4
Guess (τ_i):	10	8

Policies # 2a,2b: SJF (cont.)

Exponential Averaging (cont.)

Example: $\alpha = 0.5$ $\qquad \tau_{n+1} = (0.5 \times t_n) + (0.5 \times \tau_n)$

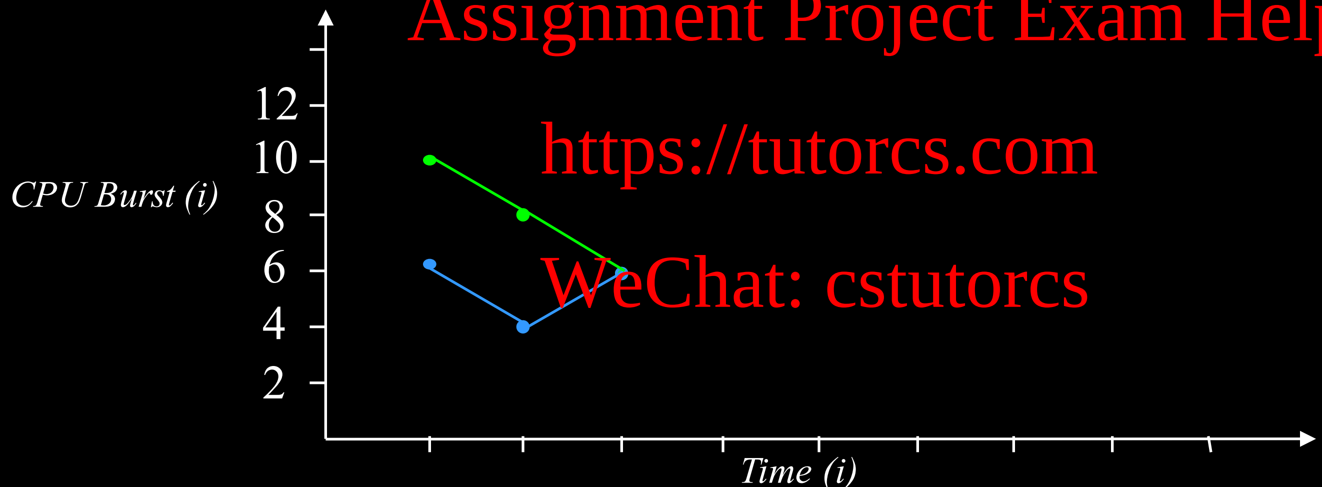


Actual (t_i):	6	4	
Guess (τ_i):	10	8	6

Policies # 2a,2b: SJF (cont.)

Exponential Averaging (cont.)

Example: $\alpha = 0.5$ $\qquad \tau_{n+1} = (0.5 \times t_n) + (0.5 \times \tau_n)$

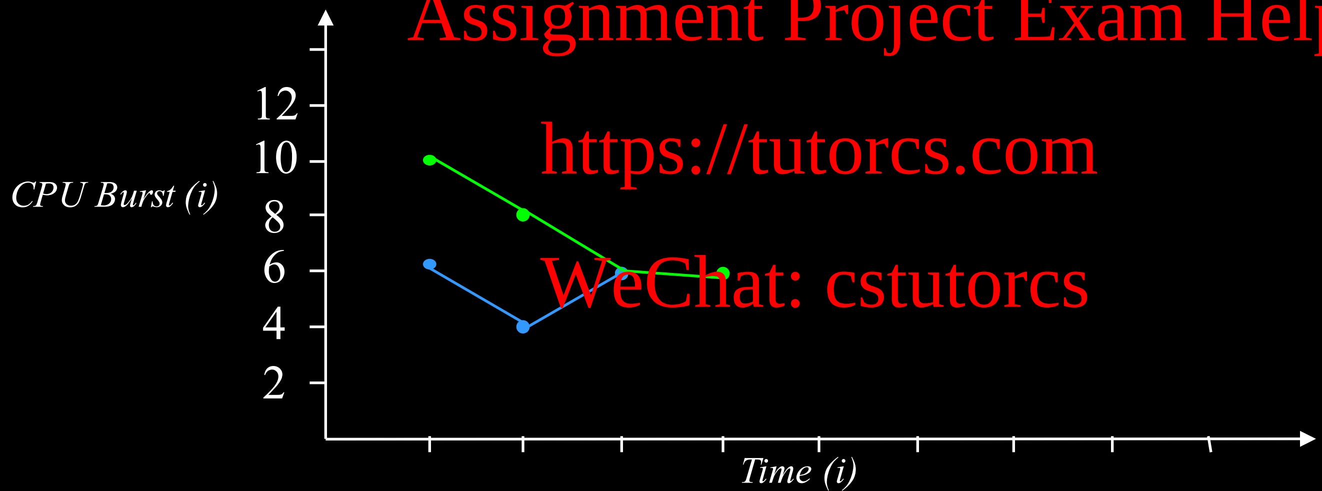


Actual (t_i):	6	4	6
Guess (τ_i):	10	8	6

Policies # 2a,2b: SJF (cont.)

Exponential Averaging (cont.)

Example: $\alpha = 0.5$ $\tau_{n+1} = (0.5 \times t_n) + (0.5 \times \tau_n)$

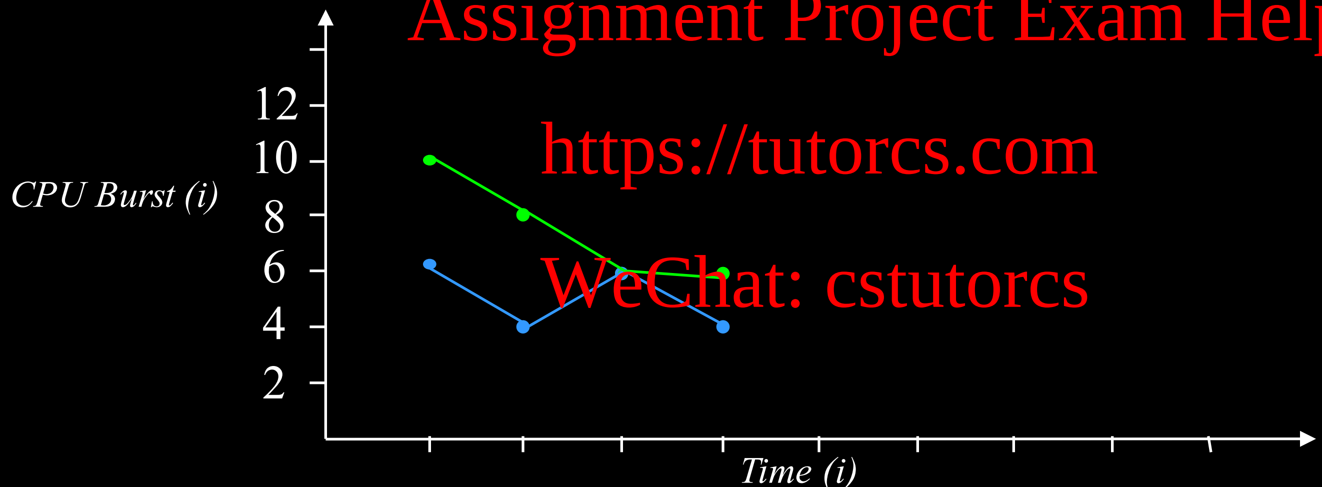


Actual (t_i):	6	4	6	
Guess (τ_i):	10	8	6	6

Policies # 2a,2b: SJF (cont.)

Exponential Averaging (cont.)

Example: $\alpha = 0.5$ $\qquad \tau_{n+1} = (0.5 \times t_n) + (0.5 \times \tau_n)$

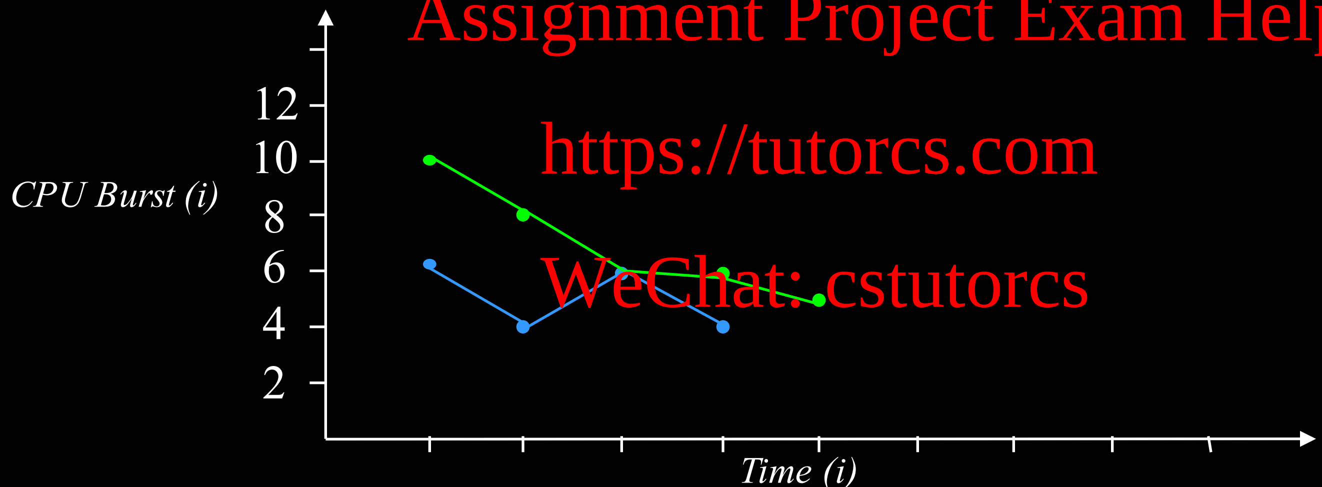


Actual (t_i):	6	4	6	4
Guess (τ_i):	10	8	6	6

Policies # 2a,2b: SJF (cont.)

Exponential Averaging (cont.)

Example: $\alpha = 0.5$ $\quad\quad \tau_{n+1} = (0.5 \times t_n) + (0.5 \times \tau_n)$

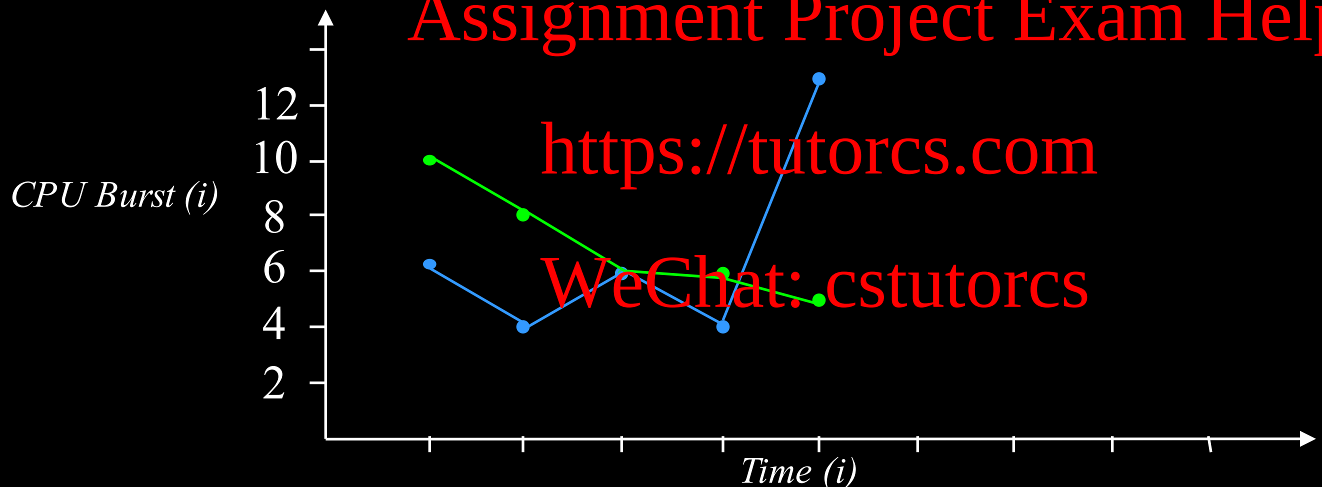


Actual (t_i):	6	4	6	4	
Guess (τ_i):	10	8	6	6	5

Policies # 2a,2b: SJF (cont.)

Exponential Averaging (cont.)

Example: $\alpha = 0.5$ $\quad\quad \tau_{n+1} = (0.5 \times t_n) + (0.5 \times \tau_n)$

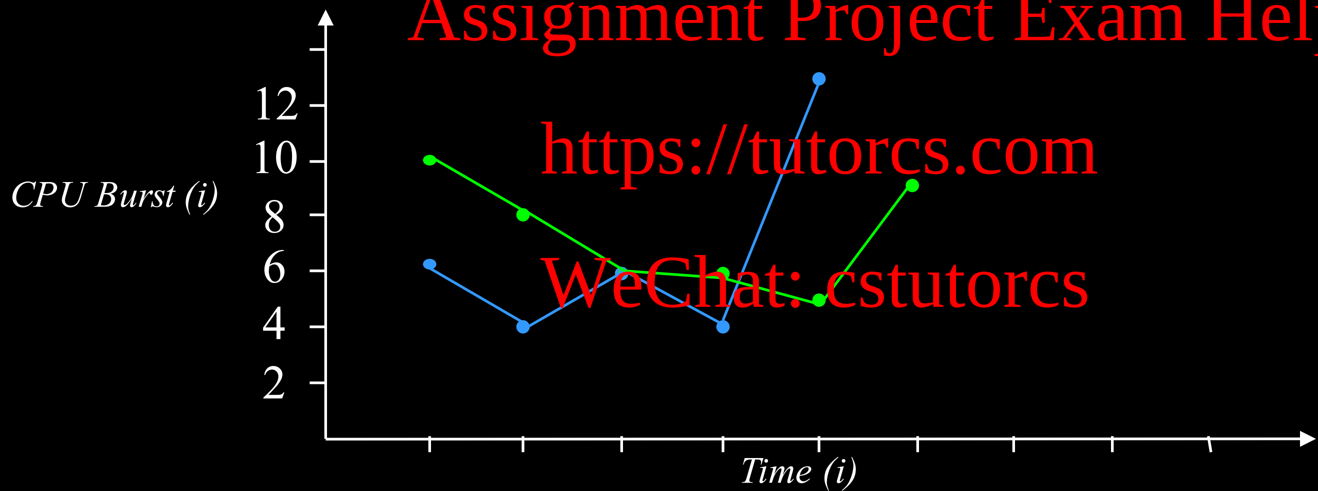


Actual (t_i):	6	4	6	4	13
Guess (τ_i):	10	8	6	6	5

Policies # 2a,2b: SJF (cont.)

Exponential Averaging (cont.)

Example: $\alpha = 0.5$ $\qquad \tau_{n+1} = (0.5 \times t_n) + (0.5 \times \tau_n)$

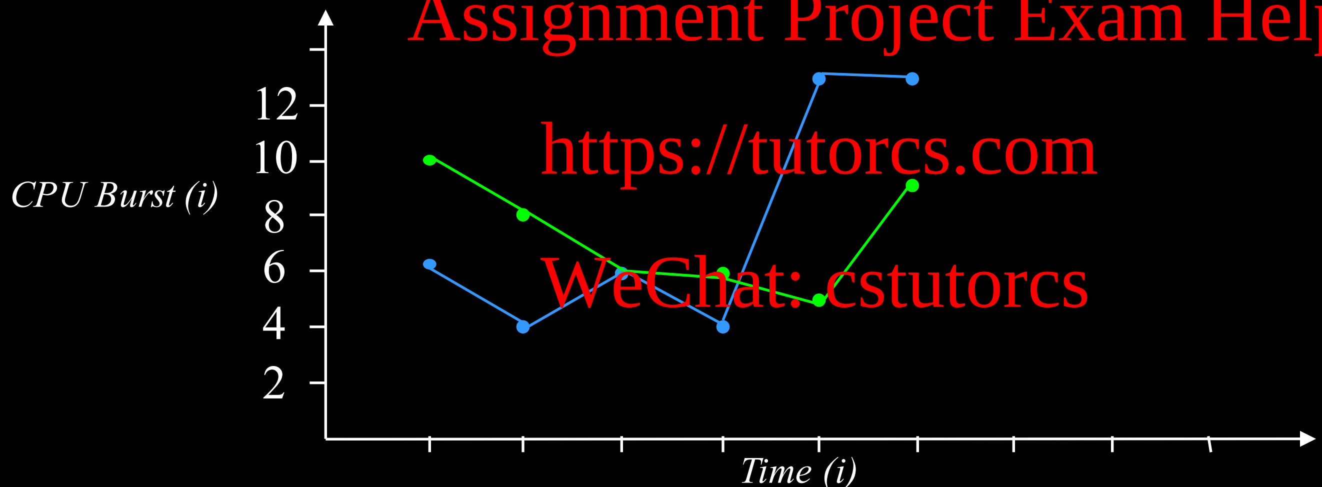


Actual (t_i):	6	4	6	4	13	
Guess (τ_i):	10	8	6	6	5	9

Policies # 2a,2b: SJF (cont.)

Exponential Averaging (cont.)

Example: $\alpha = 0.5$ $\qquad \tau_{n+1} = (0.5 \times t_n) + (0.5 \times \tau_n)$

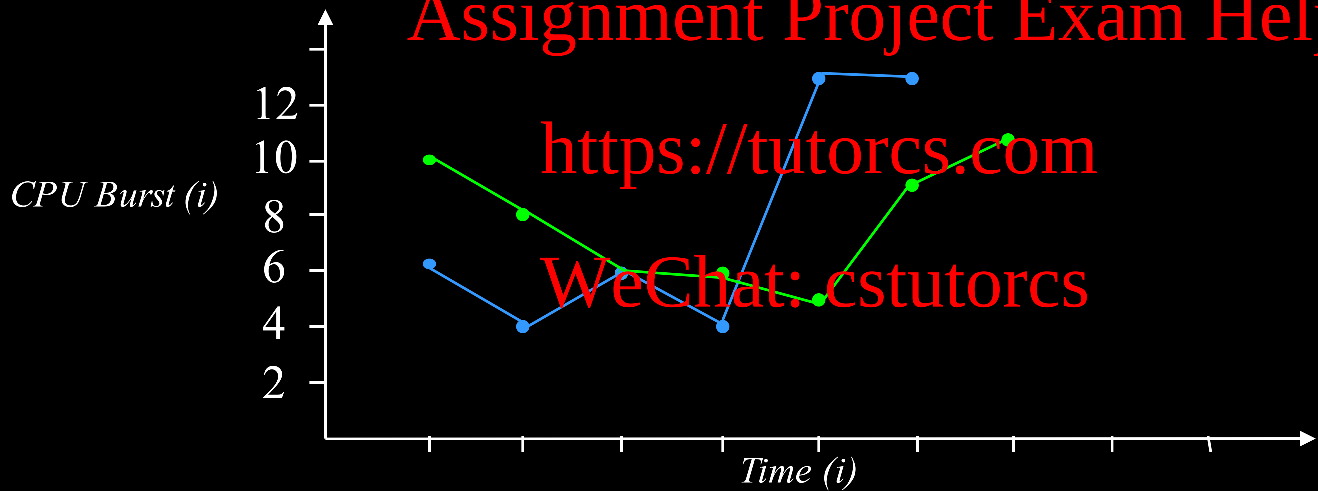


Actual (t_i):	6	4	6	4	13	13
Guess (τ_i):	10	8	6	6	5	9

Policies # 2a,2b: SJF (cont.)

Exponential Averaging (cont.)

Example: $\alpha = 0.5$ $\tau_{n+1} = (0.5 \times t_n) + (0.5 \times \tau_n)$

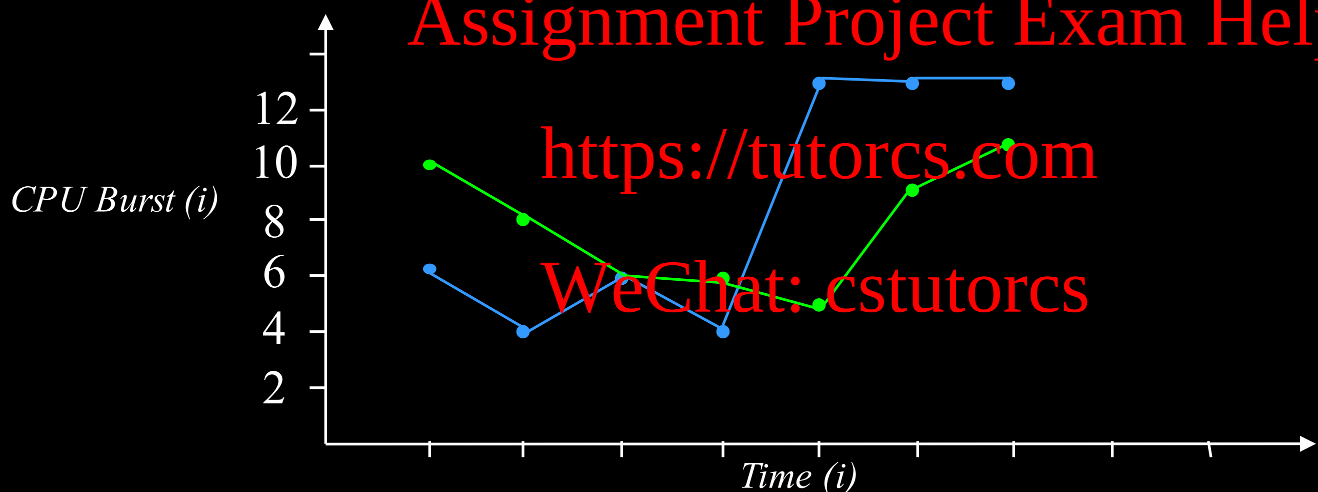


Actual (t_i):	6	4	6	4	13	13	
Guess (τ_i):	10	8	6	6	5	9	11

Policies # 2a,2b: SJF (cont.)

Exponential Averaging (cont.)

Example: $\alpha = 0.5$ $\qquad \tau_{n+1} = (0.5 \times t_n) + (0.5 \times \tau_n)$

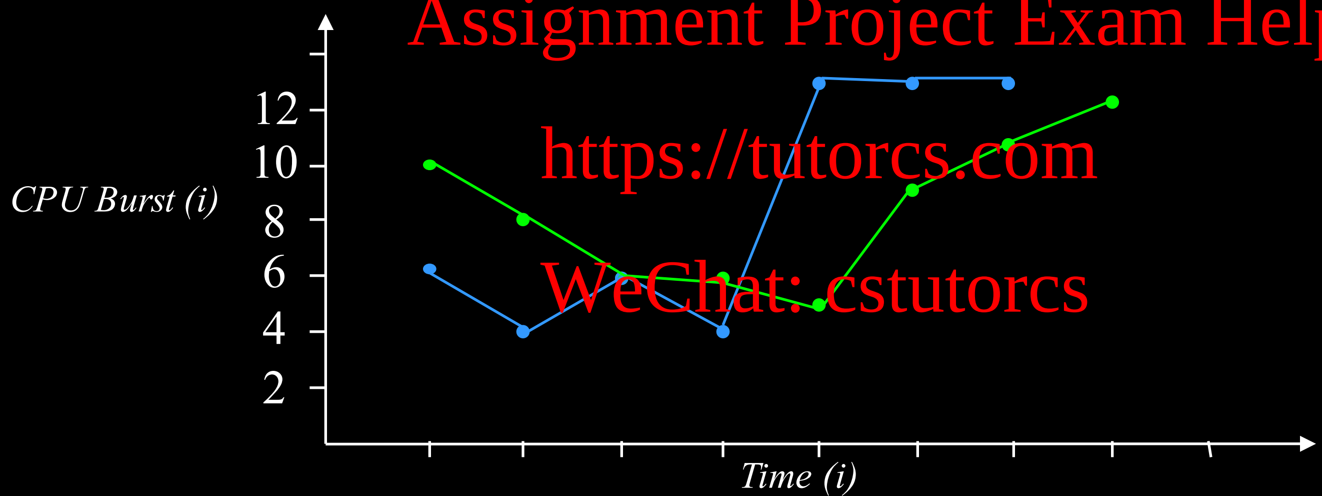


Actual (t_i):	6	4	6	4	13	13	13
Guess (τ_i):	10	8	6	6	5	9	11

Policies # 2a,2b: SJF (cont.)

Exponential Averaging (cont.)

Example: $\alpha = 0.5$ $\tau_{n+1} = (0.5 \times t_n) + (0.5 \times \tau_n)$



Actual (t_i):	6	4	6	4	13	13	13	
Guess (τ_i):	10	8	6	6	5	9	11	12

Policies # 2a,2b: SJF (cont.)

Exponential Average:

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

Choosing α*:*

1. $\alpha = 0 \Rightarrow \tau_{n+1} = \tau_n$
   - *Recent history doesn't count*

2. $\alpha = 1 \Rightarrow \tau_{n+1} = t_n$
   - *Only most recent burst counts*

3. *In general:* $\tau_{n+1} = (\alpha \times t_n) +$

$$(1 - \alpha) \times \alpha \, t_{n-1} +$$
$$(1 - \alpha)^2 \times \alpha \, t_{n-2} +$$
$$(1 - \alpha)^3 \times \alpha \, t_{n-3} +$$
$$\ldots$$
$$(1 - \alpha)^n \times \alpha \, t_0 +$$

Weights decrease since $(1 - \alpha) \leq 1$

Policies # 2a,2b: SJF (cont.)

+: *Approximates optimal schedule for*

- *Average wait time*
- *Average turnaround time*

–: *Fairness*

- *Starves processes with expected long CPU bursts*

CPU Scheduling Policies Summary

Policy	*Throughput*	*Waiting*	*Response*	*Fairness*	*Overhead*	*Comments*
FCFS	✗	✗	✗	✓	✓	+ *easy implementation* – *convoy effect*

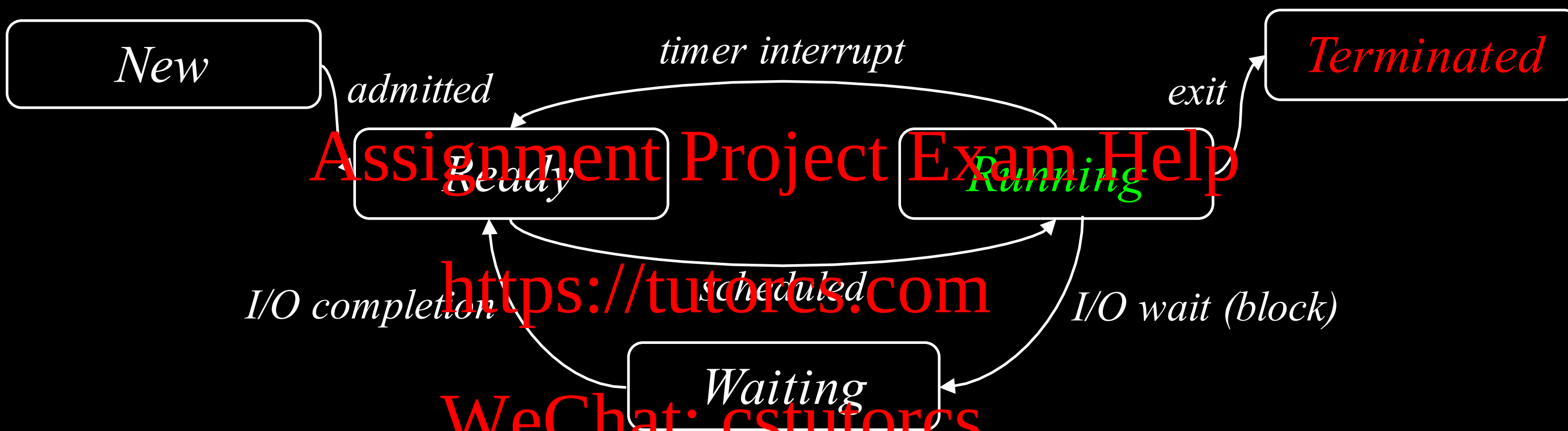
CPU Scheduling Policies Summary

Policy	*Throughput*	*Waiting*	*Response*	*Fairness*	*Overhead*	*Comments*
FCFS	✗	✗	✗	✓	✓	+ *easy implementation* -- *convoy effect*
SJF	✓✓	✓✓		✗		+ *waiting turnaround time* -- *starvation hard to predict CPU time*

Schedule Policy #3: Priority-Based

Scheduling the Ready Queue

Processes Not Always Running



New:	*Process is being created*
Running:	*Instructions are being executed*
Waiting:	*Process is waiting for some event to occur*
Ready:	*Process is waiting to be assigned the CPU*
Terminated:	*Process has finished execution*

Policies # 3a,3b: Priority

Idea:

- *Priority Number (Integer) Associated With Each Process*
- *CPU Allocated to Process With Highest Priority (Lowest Priority Number) …*

3a. Non-preemptive:

Whenever running process blocks or terminates

3b. Preemptive:

Whenever new process arrives or running process blocks or terminates

Example …

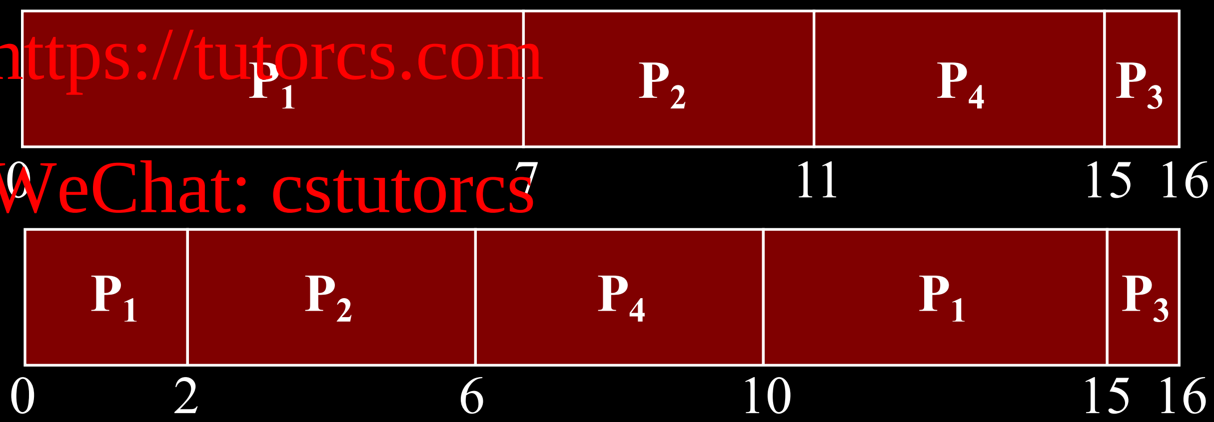
Policies # 3a,3b: Priority

Example

Process	Arrival Time	Burst	Priority
P_1	0.0	7	2
P_2	2.0	4	0
P_3	4.0	1	3
P_4	5.0	4	1

Non-preemptive

Preemptive



Q: How does priority scheduling generalize SJF scheduling?

A: SJF = Priority scheduling where shortest CPU burst = highest priority

Policies # 3a,3b: Priority

Assessing Priority Scheduling

+:

Reflects relative importance of processes (e.g.: kernel > user)

-:

Turnaround, wait times (process burst length ignored)

Starvation of low-priority processes

→ *Can counter with "aging"*
(process gets increasing priority the more it waits)

CPU Scheduling Policies Summary

Policy	***Throughput***	***Waiting***	***Response***	***Fairness***	***Overhead***	***Comments***
FCFS	✗	✗	✗	✓	✓	+ *easy implementation* -- *convoy effect*
SJF	✓✓	✓✓		✗		+ *waiting turnaround time* -- *starvation hard to predict CPU time*

CPU Scheduling Policies Summary

Policy	***Throughput***	***Waiting***	***Response***	***Fairness***	***Overhead***	***Comments***
FCFS	✗	✗	✗	✓	✓	+ *easy implementation* -- *convoy effect*
SJF	✓✓	✓✓		✗		+ *waiting turnaround time* -- *starvation hard to predict CPU time*
Priority				✗		

Schedule Policy #4: Round-Robin

Policy #4: Round-Robin

Idea:

- *Each process gets small unit of CPU time (time quantum).*
- *After time quantum has elapsed, process is preempted and added to end of ready queue*
- *Preemptive only: Process switch based on timer interrupts*

Goal: Fairness and Response Time

Suppose Time Quantum = q, & n Processes in Ready Queue

Each process gets 1/n of CPU time in chunks of q

No process waits more than (n – 1) × q time units

Policy # 4: Round-Robin

Example: *Time Quantum = 20*

Process	CPU Burst
P_1	0
P_2	0
P_3	0
P_4	0

Waiting time:

P_1 : 17 + 20 + 20 + 20 +4 = 81
P_2 : 20
P_3 : 20 + 17 + 20 + 20 + 4 + 13 = 94
P_4 : 20 + 17 + 20 + 20 + 20 = 97

Average: 73

Policy #4: Round-Robin

+: *Fairness*

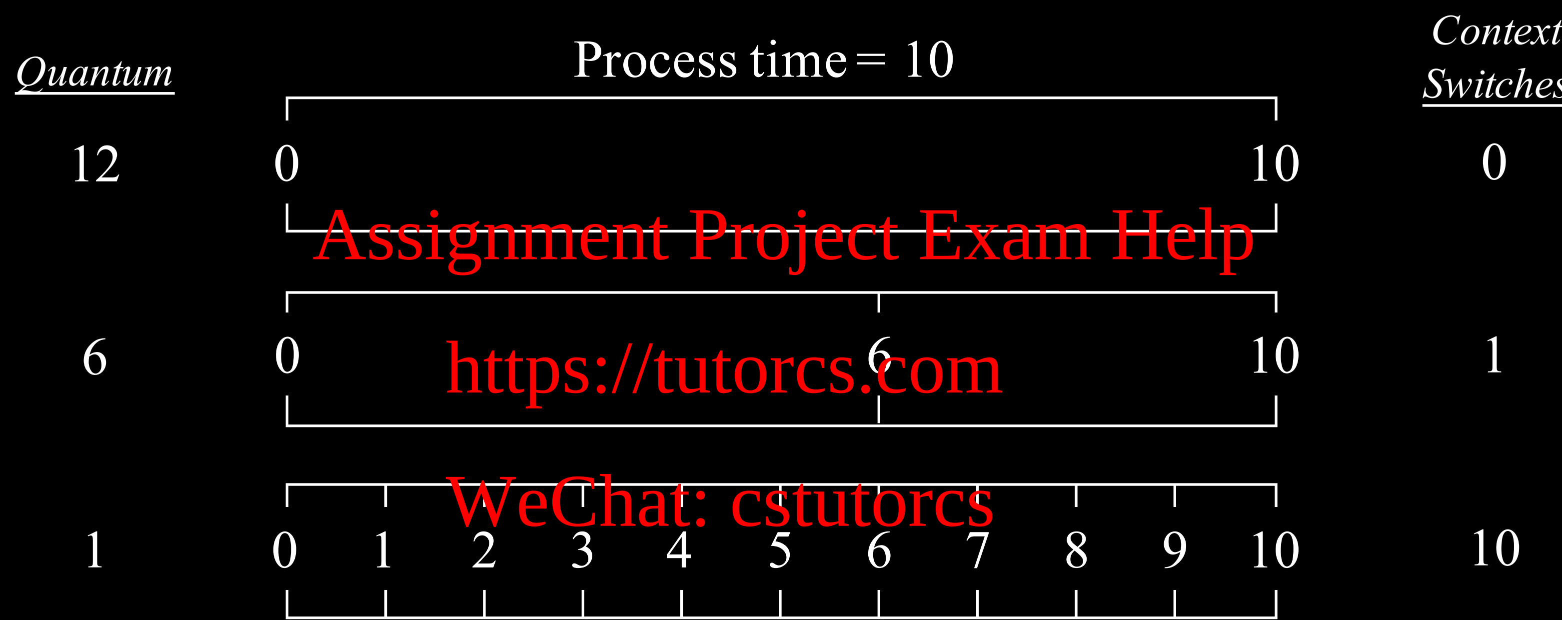
Response time

–: *Turnaround time*

Cost of context switches (Must choose quantum carefully)

Policy #4: Round-Robin

Choosing a Time Quantum, q:



1. *Large q $\Rightarrow$ few context switches.*
   q > all CPU bursts = FIFO

2. *Small q $\Rightarrow$ many context switches.*
   Too small $\Rightarrow$ overhead can compromise performance

Policy #4: Round-Robin

Choosing a Time Quantum, q

Should be large with respect to context switch time

$$\frac{\texttt{context-switch-time}}{\texttt{context-switch-time + q}} = \texttt{\% of CPU spent switching}$$

Should also consider process performance (e.g.: Average turnaround time)

Q: Given:

Process	CPU Burst
P_1	6
P_2	3
P_3	1
P_4	7

What is Average Turnaround Time Assuming q = 2?

Policy #4: Round-Robin

q = 2:

Process	CPU Burst
P_1	0
P_2	0
P_3	0
P_4	0

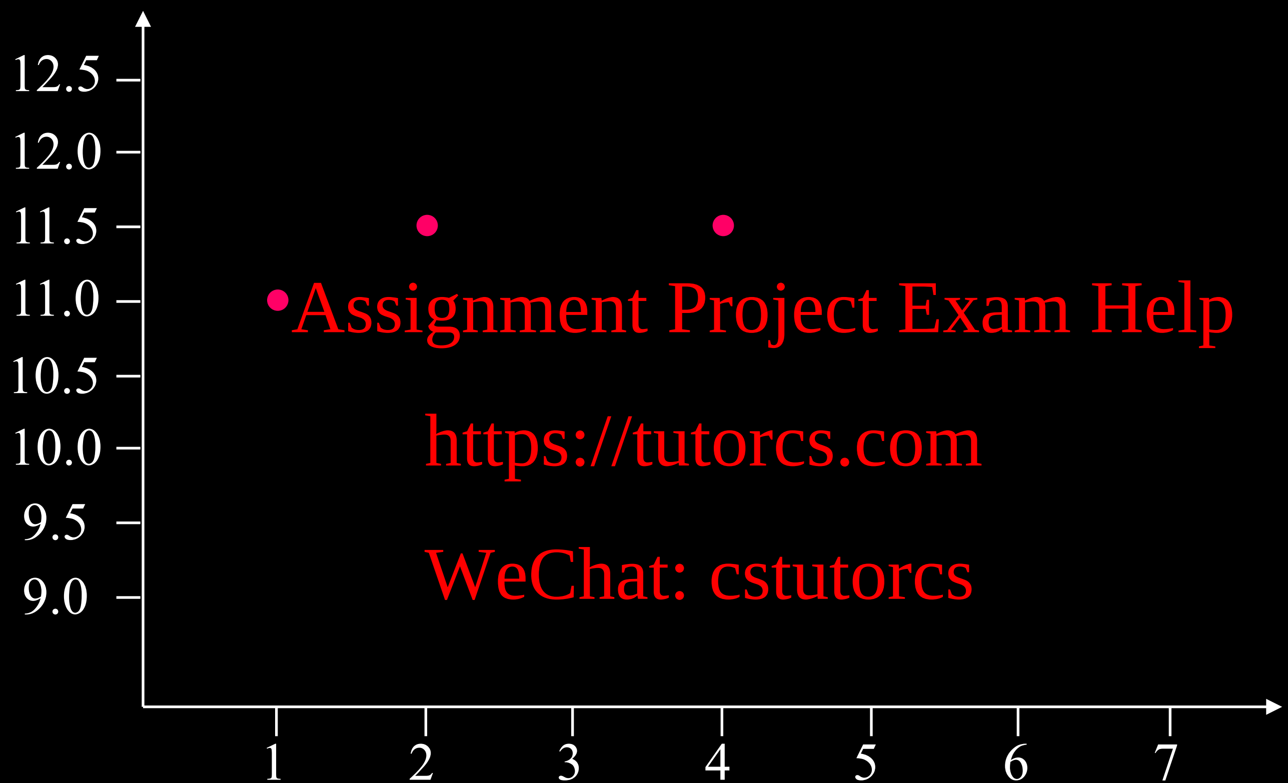
P_1	P_2	P_3	P_4	P_1	P_2	P_4	P_1	P_4	P_4
0–2	2–4	4–5	5–7	7–9	9–10	10–12	12–14	14–16	16–17

Turnaround time:

P_1 : 14
P_2 : 10
P_3 : 5
P_4 : 17

Average = 11.5

Policy #4: Round-Robin



Q: Determine the Average Turnaround Time Assuming q = …

a) 3 *b) 5* *c) 6* *d) 7*

Policy #4: Round-Robin

q = 3, 5, 6, 7:

Process	CPU Burst
P_1	6
P_2	3
P_3	1
P_4	7

Turnaround time?

Policy #4: Round-Robin

$q = 3$:

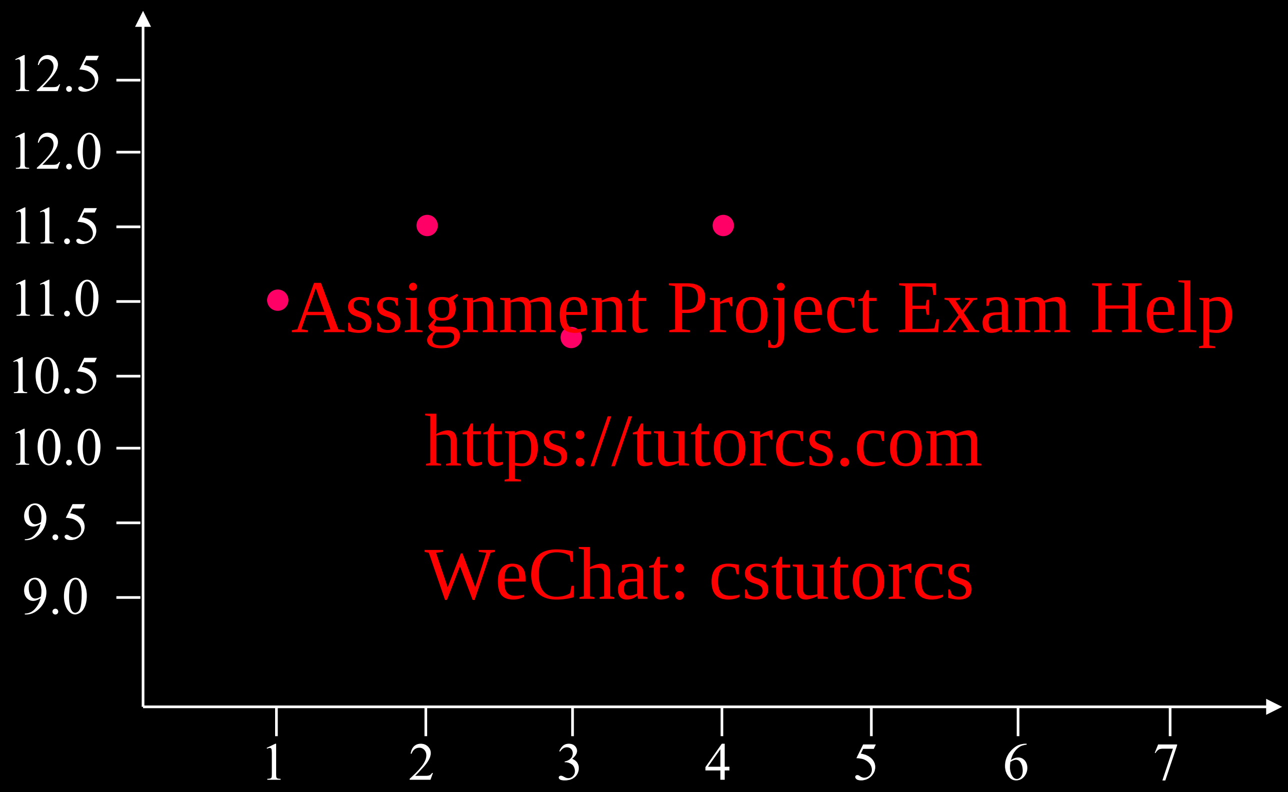
Process	CPU Burst
P_1	6
P_2	3
P_3	1
P_4	7

Turnaround time:

P_1 : 13
P_2 : 6
P_3 : 7
P_4 : 17

Average = 10.75

Policy #4: Round-Robin



Policy #4: Round-Robin

q = 5

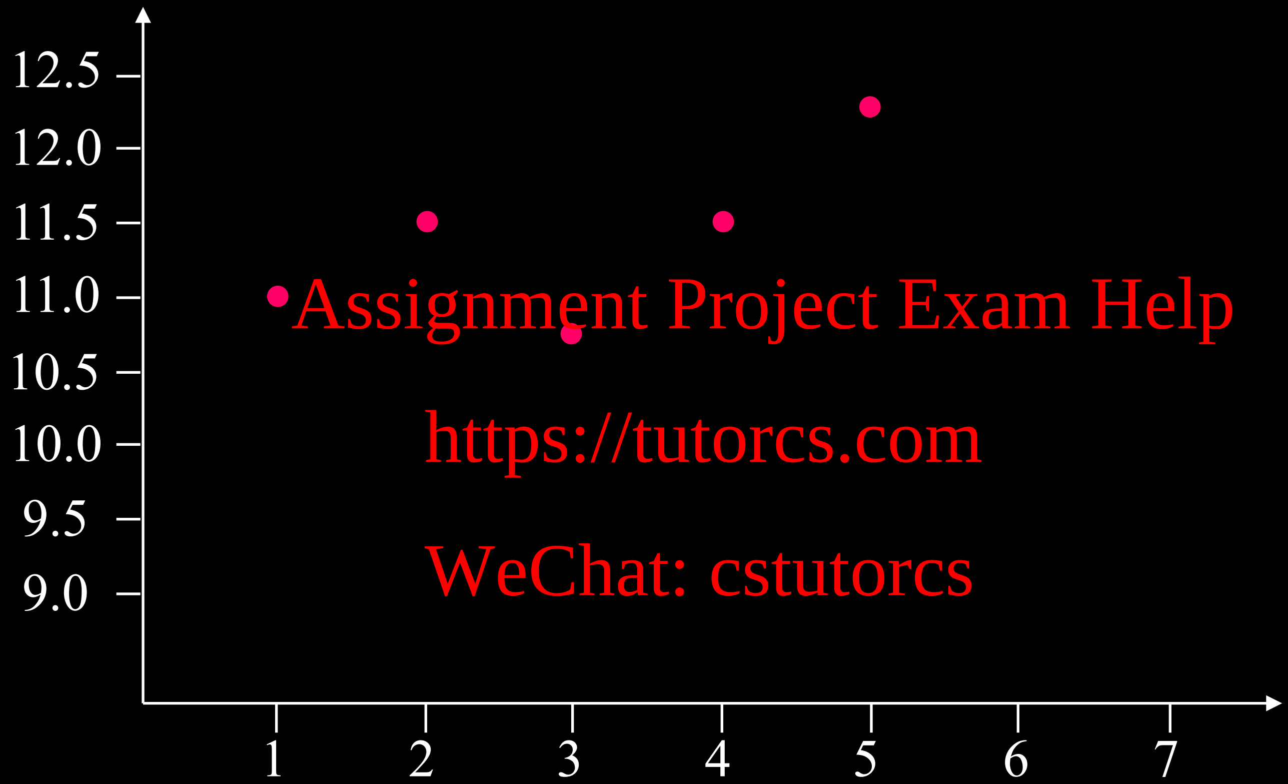
Process	CPU Burst
P_1	6
P_2	3
P_3	1
P_4	7

Turnaround time:

P_1 : 15
P_2 : 8
P_3 : 9
P_4 : 17

Average = 12.25

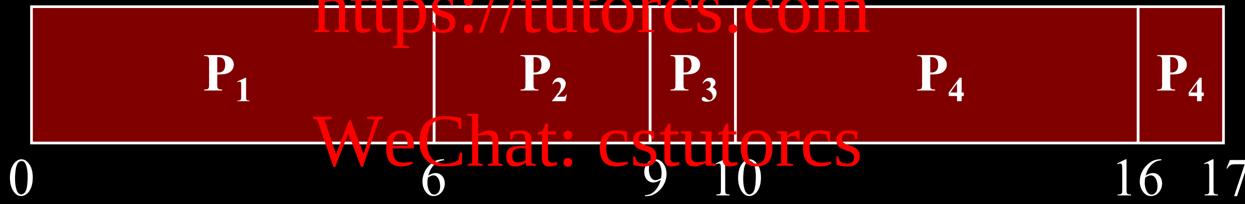
Policy #4: Round-Robin



Policy #4: Round-Robin

q = 6,
q = 7

Process	CPU Burst
P_1	6
P_2	3
P_3	1
P_4	7

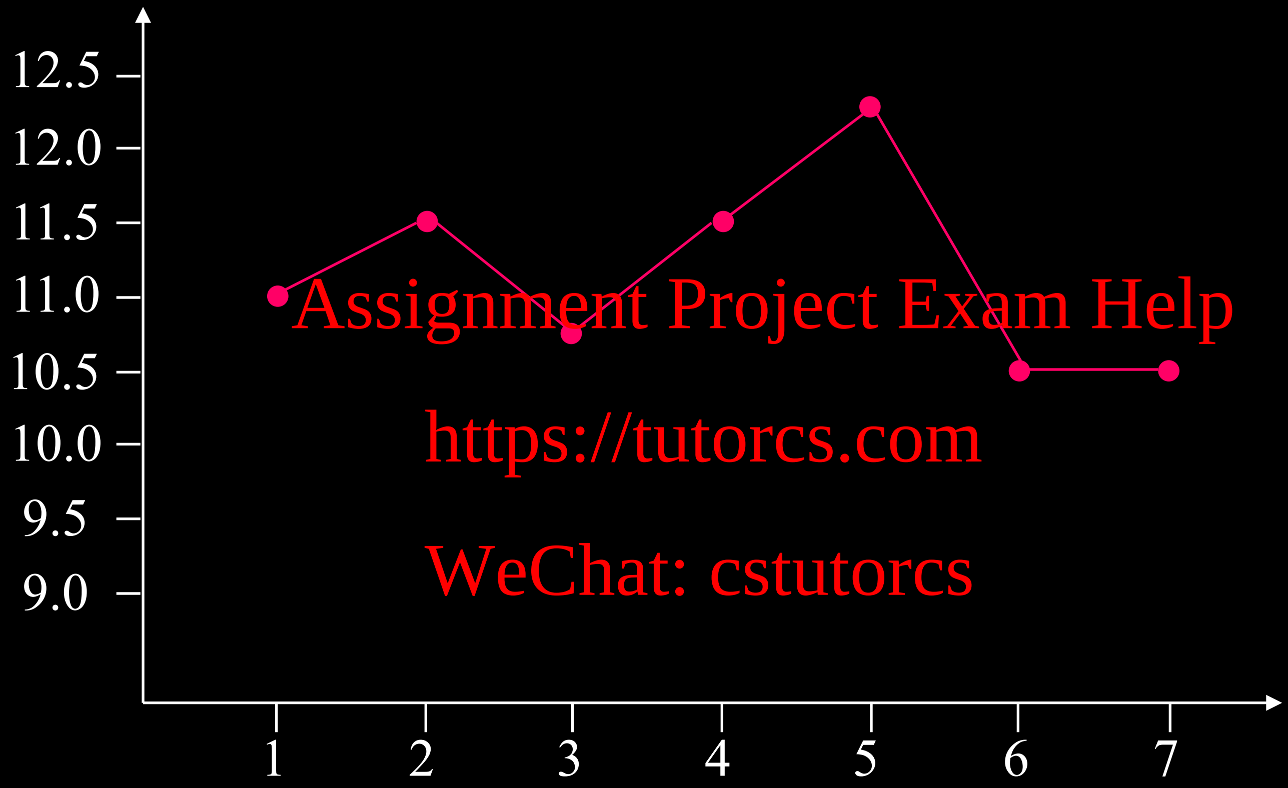


Turnaround time:

P_1 : 6
P_2 : 9
P_3 : 10
P_4 : 17

Average = 10.5

Policy #4: Round-Robin



CPU Scheduling Policies Summary

Policy	***Throughput***	***Waiting***	***Response***	***Fairness***	***Overhead***	***Comments***
FCFS	✗	✗	✗	✓	✓	+ *easy implementation* -- *convoy effect*
SJF	✓✓	✓✓		✗		+ *waiting* *turnaround time* -- *starvation* *hard to predict CPU time*
Priority				✗		+ *importance considered* -- *poor metrics for low priority*

CPU Scheduling Policies Summary

Policy	***Throughput***	***Waiting***	***Response***	***Fairness***	***Overhead***	***Comments***
FCFS	✗	✗	✗	✓	✓	+ *easy implementation* -- *convoy effect*
SJF	✓✓	✓✓		✗		+ *waiting turnaround time* -- *starvation hard to predict CPU time*
Priority				✗		+ *importance considered* -- *poor metrics for low priority*
Round-Robin			✓	✓	✗	

Revise: Throughput vs Turnaround

Analogy: Plane vs Train for Shipping Tables (MA-CA)

Plane: 6 hours of travel each way, 48 tables per trip

Train: 2 days of travel each way, 60 x 96 = 5760 tables per trip

Processing: Can assemble 60 tables / hour

Turnaround Time = max time to "process" a table

lowest *Plane: 12 (wait) + 6 (travel)+ 0.8 hrs (assembling) ~ 19 hrs*

Train: 96 (wait) + 48 (travel) + 96 (processing) = 10 days

Throughput = number of tables processed / day

	Day 1	*Day 2*	*Day 3*	*Day 4*	*Day 5*	*Day 6*	*Throughput*
Plane	*96*	*96*	*96*	*96*	*96*	*96*	*96 / day*
highest *Train*	*0*	*0*	*1440*	*1440*	*1440*	*1440*	*~1440 / day*

Schedule Policy #5: Multilevel Queues

Policy #5: Multilevel Queues

Idea:

- *Ready queue is partitioned into separate queues, each with own scheduling algorithm*

  → *e.g., Queue 1 (for interactive processes): RR,*
  Queue 2 (for others): FCFS

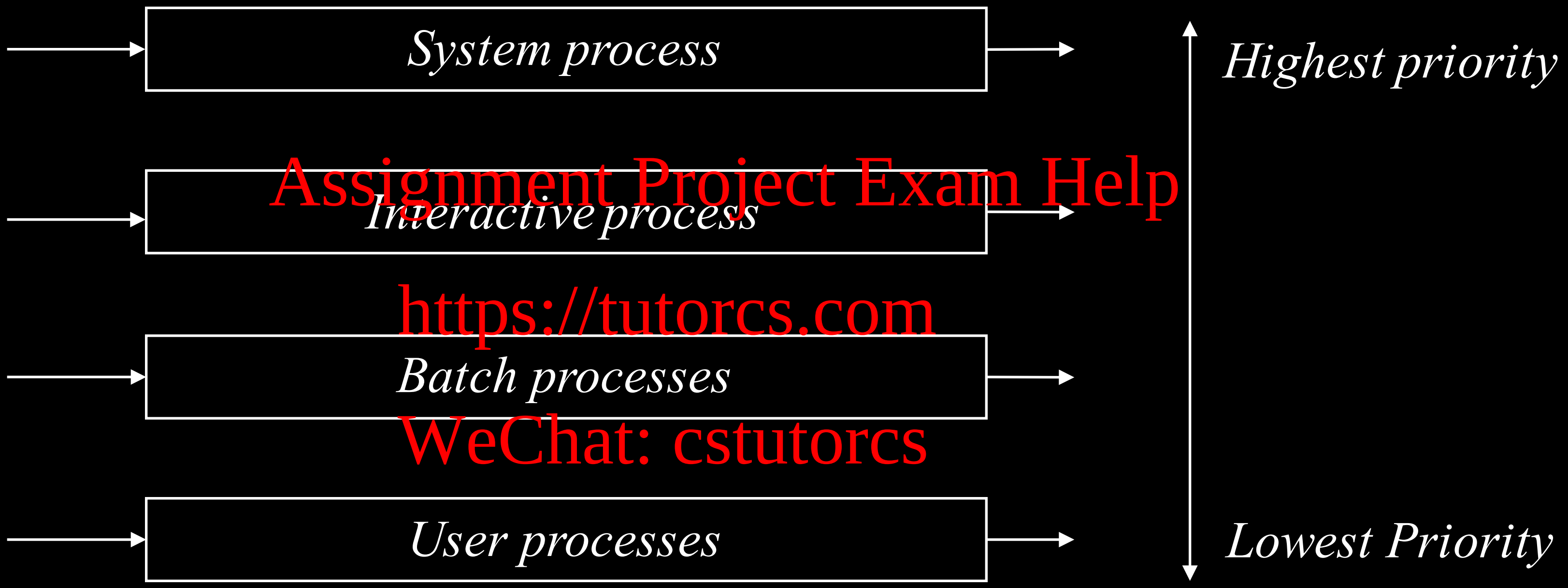
Requires Interqueue Scheduling Too (Scheduling of Queues)

1. *Fixed Priority:*
   Serve all from high-priority queue before servicing lower-priority queue (–: starvation)

2. *Time-slice:*
   Each queue gets reserved slice of CPU time to schedule amongst its processes (e.g.: 80% for hi-priority, 20% low)

Policy #5: Multilevel Queues

An Example:



Policy #5: Multilevel Queues

In UNIX:

32 queues

→ *0…7:* *System (kernel) queues*
(e.g.: 0 reserved for swapper)

→ *8…31:* *User-level queues*
Processes move from queue to queue
1. `Nice`*: Downward*
2. Aging: Upward

Policy #5: Multilevel Queues

Variation: "Multilevel Feedback Queue"

- *Process can move between queues*

MFQ Scheduler Must Have Following Parameters:

1. *Number of queues*
2. *Scheduling algorithms for each queue*
3. *Scheduling algorithm between queues (or CPU time-slices)*
4. *Upgrade process (e.g.: Aging)*
5. *Downgrade process*
6. *Initial queue determination process*

Scheduling Design 101

Suppose Scheduling Goal Is: Fairness

(i.e.: Want to turnaround time to be proportional to burst time)

Accomplished Using MFQ's instead of Burst Prediction

A: *Queues*

"Reverse aging"

1. *High priority: RR with low quantum*
2. *Low priority: RR with high quantum*

Process gets one go on each queue, getting lower priority as it gets older

Policy #5: Multilevel Queues

An Example of MFQ:

Q_0^{75}: *RR, quantum = 75;* *initial queue of entering process*

Q_1^{150}: *RR, quantum = 150;* *process ages to Q_1 after Q_0*

Q_2: *FCFS*

Fixed priority

Interqueue Scheduling: FCFS (Serve Q_i when $Q_0 \ldots Q_{i-1}$ empty)

Process	Burst time	Arrival time
P_1	125	0
P_2	50	0
P_3	500	0
P_4	175	0
P_5	200	250
P_6	50	250

Policy #5: Multilevel Queues

Q_0^{75}: *RR, quantum = 75;*

Q_1^{150}: *RR, quantum = 150;*

Q_2: *FCFS*

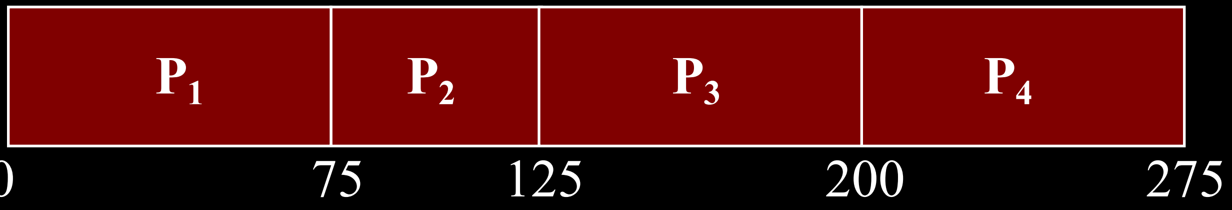
Process	Burst time	Arrival time
P_1	125	0
P_2	50	0
P_3	500	0
P_4	175	0
P_5	200	250
P_6	50	250

Time = 0:

Q_0^{75}: $P_1^{125} \rightarrow P_2^{50} \rightarrow P_3^{500} \rightarrow P_4^{175}$

Q_1^{150}: –

Q_2: –



Policy #5: Multilevel Queues

Process	Burst time	Arrival time
P_1	50	0
P_2	0	0
P_3	425	0
P_4	100	0
P_5	200	250
P_6	50	250

Q_0^{75}: *RR, quantum = 75;*

Q_1^{150}: *RR, quantum = 150;*

Q_2: *FCFS*

Time = 0:

Q_0^{75}: $P_1^{125} \rightarrow P_2^{50} \rightarrow P_3^{500} \rightarrow P_4^{175}$

Q_1^{150}: –

Q_2: –

P_1	P_2	P_3	P_4
0–75	75–125	125–200	200–275

Policy #5: Multilevel Queues

Q_0^{75}: *RR, quantum = 75;*

Q_1^{150}: *RR, quantum = 150;*

Q_2: *FCFS*

Process	Burst time	Arrival time
P_1	50	0
P_2	0	0
P_3	425	0
P_4	100	0
P_5	200	250
P_6	50	250

Time = 0:

Q_0^{75}:

Q_1^{150}: $P_1^{50} \rightarrow P_3^{425} \rightarrow P_4^{100}$

Q_2: –

P_1	P_2	P_3	P_4

0 75 125 200 275

Policy #5: Multilevel Queues

Q_0^{75}: *RR, quantum = 75;*

Q_1^{150}: *RR, quantum = 150;*

Q_2: *FCFS*

Process	Burst time	Arrival time
P_1	50	0
P_2	0	0
P_3	425	0
P_4	100	0
P_5	200	250
P_6	50	250

Time = 275:

Q_0^{75}: $P_5^{200} \rightarrow P_6^{50}$

Q_1^{150}: $P_1^{50} \rightarrow P_3^{425} \rightarrow P_4^{100}$

Q_2: –

Policy #5: Multilevel Queues

Q_0^{75}: *RR, quantum = 75;*

Q_1^{150}: *RR, quantum = 150;*

Q_2: *FCFS*

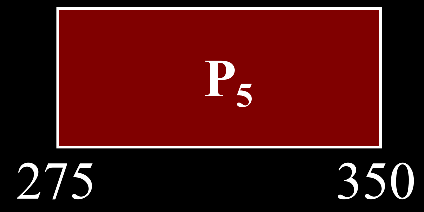
Process	Burst time	Arrival time
P_1	50	0
P_2	0	0
P_3	425	0
P_4	100	0
P_5	200	250
P_6	50	250

Time = 275:

Q_0^{75}: $P_5^{200} \rightarrow P_6^{50}$

Q_1^{150}: $P_1^{50} \rightarrow P_3^{425} \rightarrow P_4^{100}$

Q_2: –



Policy #5: Multilevel Queues

Q_0^{75}: *RR, quantum = 75;*

Q_1^{150}: *RR, quantum = 150;*

Q_2: *FCFS*

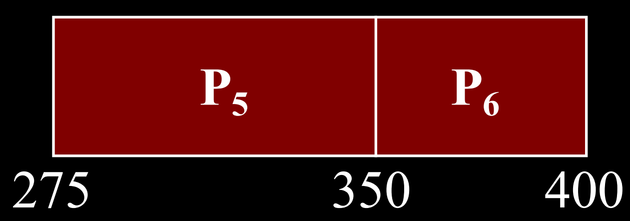
Process	Burst time	Arrival time
P_1	50	0
P_2	0	0
P_3	425	0
P_4	100	0
P_5	200	250
P_6	50	250

Time = 275:

Q_0^{75}: $P_5^{200} \rightarrow P_6^{50}$

Q_1^{150}: $P_1^{50} \rightarrow P_3^{425} \rightarrow P_4^{100}$

Q_2: –



Policy #5: Multilevel Queues

Process	Burst time	Arrival time
P_1	50	0
P_2	0	0
P_3	425	0
P_4	100	0
P_5	125	250
P_6	0	250

Q_0^{75}: *RR, quantum = 75;*

Q_1^{150}: *RR, quantum = 150;*

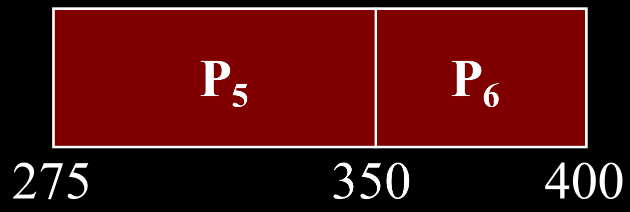
Q_2: *FCFS*

Time = 275:

Q_0^{75}: $P_5^{200} \rightarrow P_6^{50}$

Q_1^{150}: $P_1^{50} \rightarrow P_3^{425} \rightarrow P_4^{100}$

Q_2: –



Policy #5: Multilevel Queues

Q_0^{75}: *RR, quantum = 75;*

Q_1^{150}: *RR, quantum = 150;*

Q_2: *FCFS*

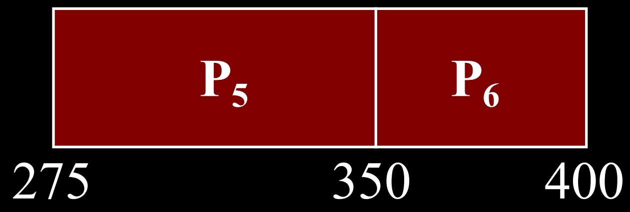
Process	Burst time	Arrival time
P_1	50	0
P_2	0	0
P_3	425	0
P_4	100	0
P_5	125	250
P_6	0	250

Time = 400:

Q_0^{75}: –

Q_1^{150}: $P_1^{50} \rightarrow P_3^{425} \rightarrow P_4^{100} \rightarrow P_5^{125}$

Q_2: –



Policy #5: Multilevel Queues

Q_0^{75}: *RR, quantum = 75;*

Q_1^{150}: *RR, quantum = 150;*

Q_2: *FCFS*

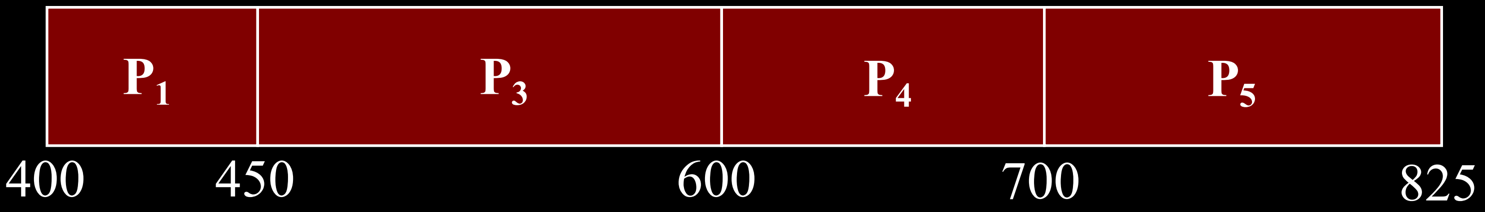
Process	Burst time	Arrival time
P_1	50	0
P_2	0	0
P_3	425	0
P_4	100	0
P_5	125	250
P_6	0	250

Time = 400:

Q_0^{75}: –

Q_1^{150}: $P_1^{50} \rightarrow P_3^{425} \rightarrow P_4^{100} \rightarrow P_5^{125}$

Q_2: –



Policy #5: Multilevel Queues

Q_0^{75}: *RR, quantum = 75;*

Q_1^{150}: *RR, quantum = 150;*

Q_2: *FCFS*

Process	Burst time	Arrival time
P_1	0	0
P_2	0	0
P_3	275	0
P_4	0	0
P_5	0	250
P_6	0	250

Time = 400:

Q_0^{75}: –

Q_1^{150}: $P_1^{50} \rightarrow P_3^{425} \rightarrow P_4^{100} \rightarrow P_5^{125}$

Q_2: –

P_1	P_3	P_4	P_5
400–450	450–600	600–700	700–825

Policy #5: Multilevel Queues

Process	Burst time	Arrival time
P_1	0	0
P_2	0	0
P_3	275	0
P_4	0	0
P_5	0	250
P_6	0	250

Q_0^{75}: *RR, quantum = 75;*

Q_1^{150}: *RR, quantum = 150;*

Q_2: *FCFS*

Time = 825:

Q_0^{75}: –

Q_1^{150}: –

Q_2: P_3^{275}

P_1	P_3	P_4	P_5
400–450	450–600	600–700	700–825

Policy #5: Multilevel Queues

Q_0^{75}: *RR, quantum = 75;*

Q_1^{150}: *RR, quantum = 150;*

Q_2: *FCFS*

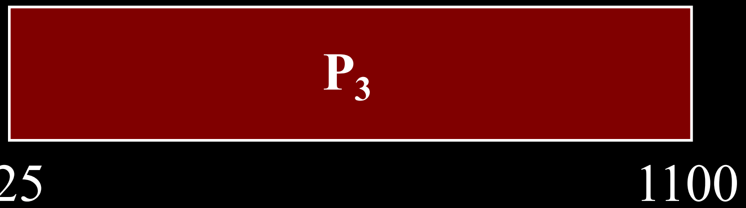
Process	Burst time	Arrival time
P_1	0	0
P_2	0	0
P_3	275	0
P_4	0	0
P_5	0	250
P_6	0	250

Time = 825:

Q_0^{75}: –

Q_1^{150}: –

Q_2: P_3^{275}



Policy #5: Multilevel Queues

Q_0^{75}: *RR, quantum = 75;*

Q_1^{150}: *RR, quantum = 150;*

Q_2: *FCFS*

Process	Burst time	Arrival time
P_1	0	0
P_2	0	0
P_3	0	0
P_4	0	0
P_5	0	250
P_6	0	250

Time = 1100:

Q_0^{75}: –

Q_1^{150}: –

Q_2: –

Policy #5: Multilevel Queues

Process	Arrival time	Completion time	Turnaround Time
P_1 (125)	0	450	450
P_2 (50)	0	125	125
P_3 (500)	0	1100	1100
P_4 (175)	0	700	700
P_5 (200)	250	825	575
P_6 (50)	250	400	150

Observe:

1. *Average turnaround time: 3100 / 6 = 516.7*
2. *Shorter jobs tend to have shorter turnaround times*

CPU Scheduler Summary

- *First Example of Resource Management (Sharing)*

Scheduling Metrics

- *CPU utilization, throughput, turnaround time, wait time, response time*
- *also care about: fairness, dispatch time*

Scheduling Policies

Policy	***Type***	+	-
FCFS	*N*	• *easy implementation*	• *convoy effect*

CPU Scheduler Summary

- *First Example of Resource Management (Sharing)*

Scheduling Metrics

- *CPU utilization, throughput, turnaround time, wait time, response time*
- *also care about: fairness, dispatch time*

Scheduling Policies

Policy	***Type***	+	-
FCFS	*N*	• *easy implementation*	• *convoy effect*
SJF	*N,P*	• *waiting, turnaround time*	• *starvation* • *need to predict CPU bursts*

CPU Scheduler Summary

- *First Example of Resource Management (Sharing)*

Scheduling Metrics

- *CPU utilization, throughput, turnaround time, wait time, response time*
- *also care about: fairness, dispatch time*

Scheduling Policies

Policy	*Type*	+	-
FCFS	*N*	• *easy implementation*	• *convoy effect*
SJF	*N,P*	• *waiting, turnaround time*	• *starvation* • *need to predict CPU bursts*
Priority	*N,P*	• *importance of process considered*	• *waiting, turnaround time* • *starvation*

CPU Scheduler Summary

- *First Example of Resource Management (Sharing)*

Scheduling Metrics

- *CPU utilization, throughput, turnaround time, wait time, response time*
- *also care about: fairness, dispatch time*

Scheduling Policies

Policy	***Type***	+	-
FCFS	*N*	• *easy implementation*	• *convoy effect*
SJF	*N,P*	• *waiting, turnaround time*	• *starvation* • *need to predict CPU bursts*
Priority	*N,P*	• *importance of process considered*	• *waiting, turnaround time* • *starvation*
Round-Robin	*P*	• *fairness* • *response time*	• *turnaround time* • *context switch overhead*

CPU Scheduler Summary

- *First Example of Resource Management (Sharing)*

Scheduling Metrics

- *CPU utilization, throughput, turnaround time, wait time, response time*
- *also care about: fairness, dispatch time*

Scheduling Policies

Policy	***Type***	+	-
FCFS	*N*	• *easy implementation*	• *convoy effect*
SJF	*N,P*	• *waiting, turnaround time*	• *starvation* • *need to predict CPU bursts*
Priority	*N,P*	• *importance of process considered*	• *waiting, turnaround time* • *starvation*
Round-Robin	*P*	• *fairness* • *response time*	• *turnaround time* • *context switch overhead*
MLQ	*P*	• *can mix and match priority with other policies*	• *complex implementation* • *inherits weaknesses of scheduling policies it uses*