

R

بسم الله الرحمن الرحيم

سیستم عامل

جلسه یازدهم – الگوریتم‌های زمان‌بندی

جلسہ ی گذشتہ

پن پست

Deadlock Avoidance

- Detection – *optimistic* approach
 - *Allocate resources*
 - *Break system to fix the problem if necessary*
- Avoidance – *pessimistic* approach
 - *Don't allocate resource if it may lead to deadlock*
 - *If a process requests a resource make it wait until you are sure it's OK*

Banker's Algorithm

- Look for a row, R , whose unmet resource needs are all smaller than or equal to A . If no such row exists, the system will eventually deadlock since no process can run to completion
- Assume the process of the row chosen requests all the resources that it needs (which is guaranteed to be possible) and finishes. Mark that process as terminated and add all its resources to A vector
- Repeat steps 1 and 2, until either all process are marked terminated, in which case the initial state was safe, or until deadlock occurs, in which case it was not

Safety Algorithm

1. Let **Work** and **Finish** be vectors of length m and n , respectively. Initialize:

Work = Available

Finish [i] = false for $i = 0, 1, \dots, n-1$

2. Find an i such that both:

(a) Finish [i] = false

(b) Need_i ≤ Work

If no such i exists, go to step 4

3. ***Work = Work + Allocation_i***
Finish[i] = true
go to step 2

4. If ***Finish [i] == true*** for all i , then the system is in a safe state

Resource-Request Algorithm for Process

P_i

$Request_i$ = request vector for process T_i . If **$Request_i[j] = k$** then process T_i wants k instances of resource type R_j

1. If **$Request_i \leq Need_i$** , go to step 2. Otherwise, raise error condition
2. If **$Request_i \leq Available$** , go to step 3. Otherwise T_i must wait, since resources are not available
3. Pretend to allocate requested resources to T_i by modifying the state as follows:

$Available = Available - Request_i$;

$Allocation_i = Allocation_i + Request_i$;

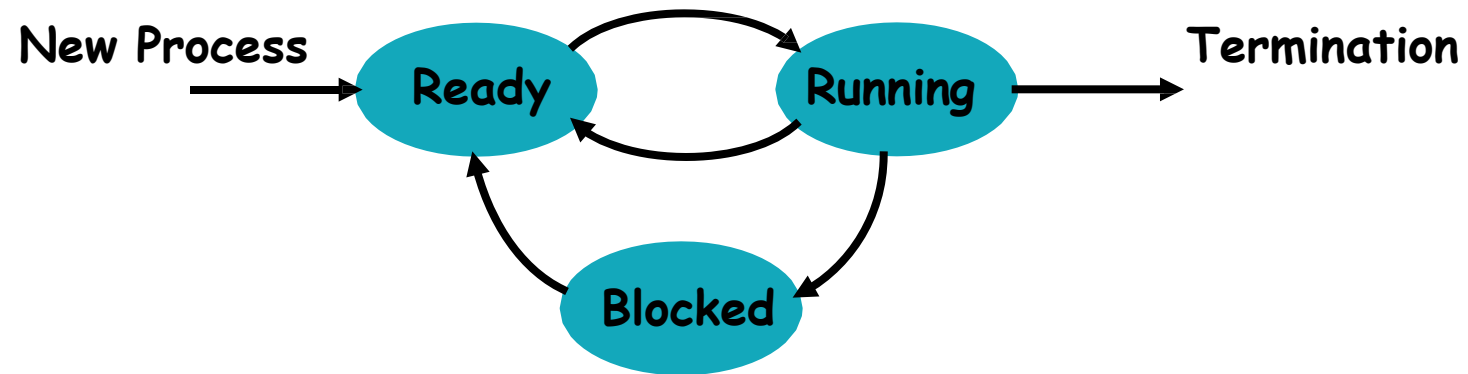
$Need_i = Need_i - Request_i$;

- If safe \Rightarrow the resources are allocated to T_i
- If unsafe $\Rightarrow T_i$ must wait, and the old resource-allocation state is restored

جلسه‌ی جدید

زمان‌بندی پرداخت‌ها

Process State Model



CPU Scheduling Criteria

- CPU Utilization
 - *how busy is the CPU?*
- Throughput
 - *how many jobs finished/unit time?*
- Turnaround Time
 - *how long from job submission to job termination?*
- Response Time
 - *how long does it take to get a response*
- Missed deadlines

Scheduler Options

■ Priorities

- May use priorities to determine who runs next
- Dynamic vs. Static algorithms
 - Dynamically alter the priority of the tasks while they are in the system (possibly with feedback)
 - Static algorithms typically assign a fixed priority when the job is initially started

■ Preemptive vs. Nonpreemptive

- Preemptive systems allow the task to be interrupted at any time so that the O.S. can take over again






Scheduling Policies

- First-Come, First Served (FIFO)
- Shortest Job First (non-preemptive)
- Shortest Job First (with preemption)
- Round-Robin Scheduling
- Priority Scheduling
- Real-Time Scheduling






First-Come, First-Served (FIFO)

- Start jobs in the order they arrive (FIFO queue)
Run each job until completion






First-Come, First-Served (FIFO)

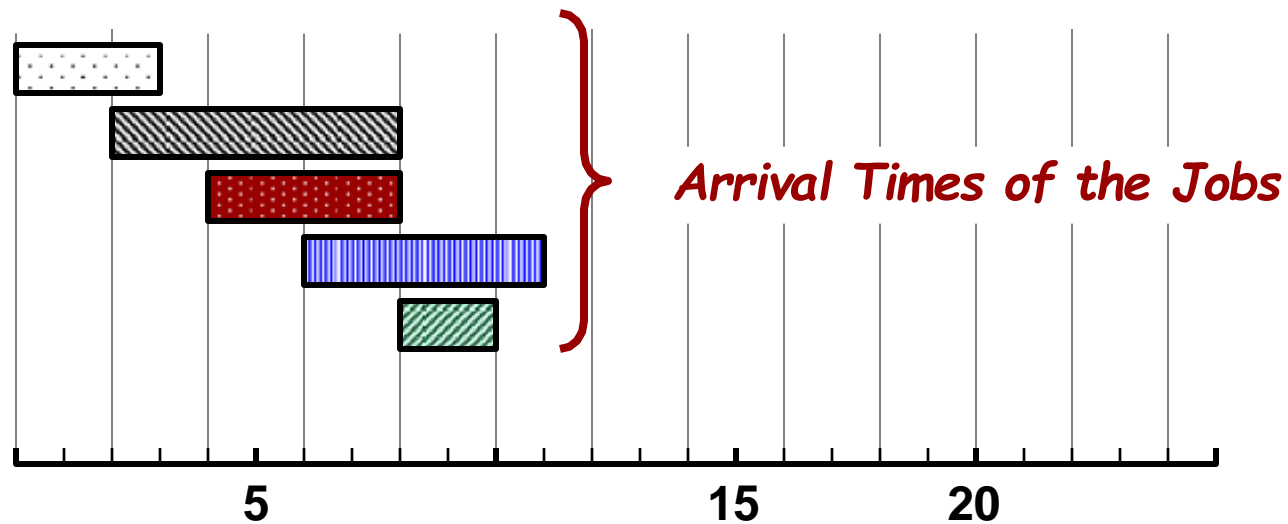
Process		Arrival Time	Processing Time
	1	0	3
	2	2	6
	3	4	4
	4	6	5
	5	8	2

First-Come, First-Served (FIFO)






Process		Arrival Time	Processing Time	Delay	Turnaround Time
	1	0	3		
	2	2	6		
	3	4	4		
	4	6	5		
	5	8	2		

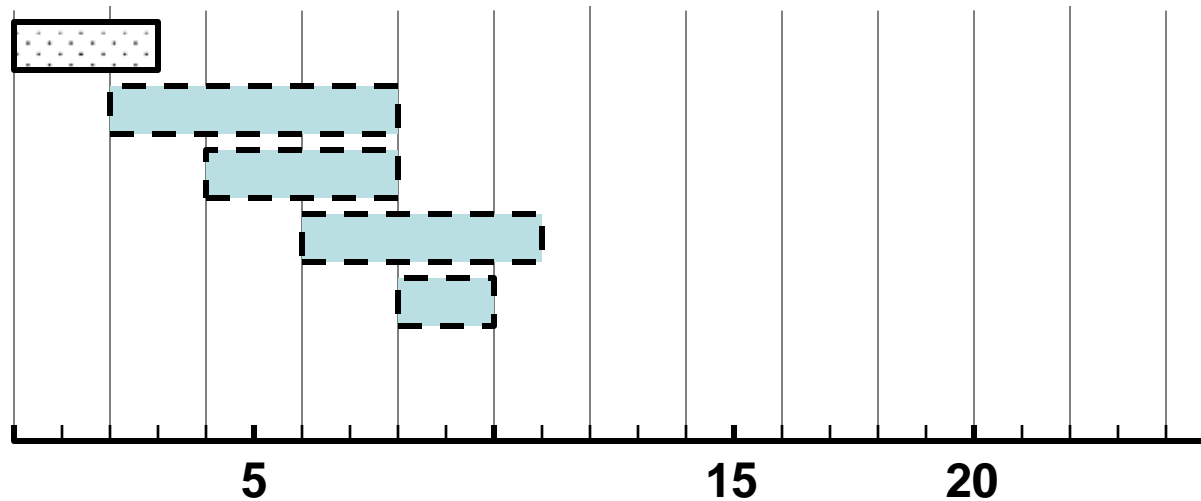
First-Come, First-Served (FIFO)

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








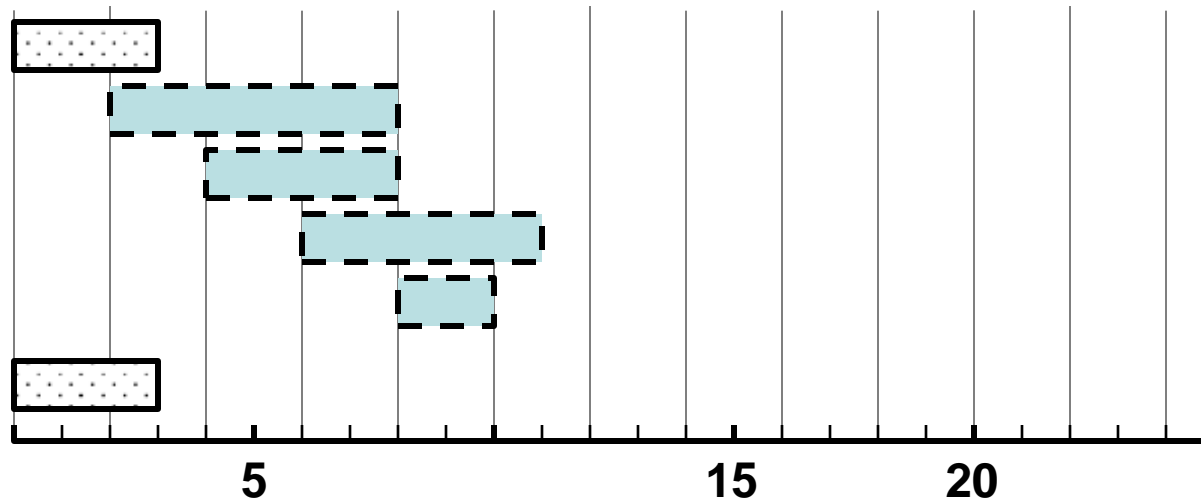
First-Come, First-Served (FIFO)

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








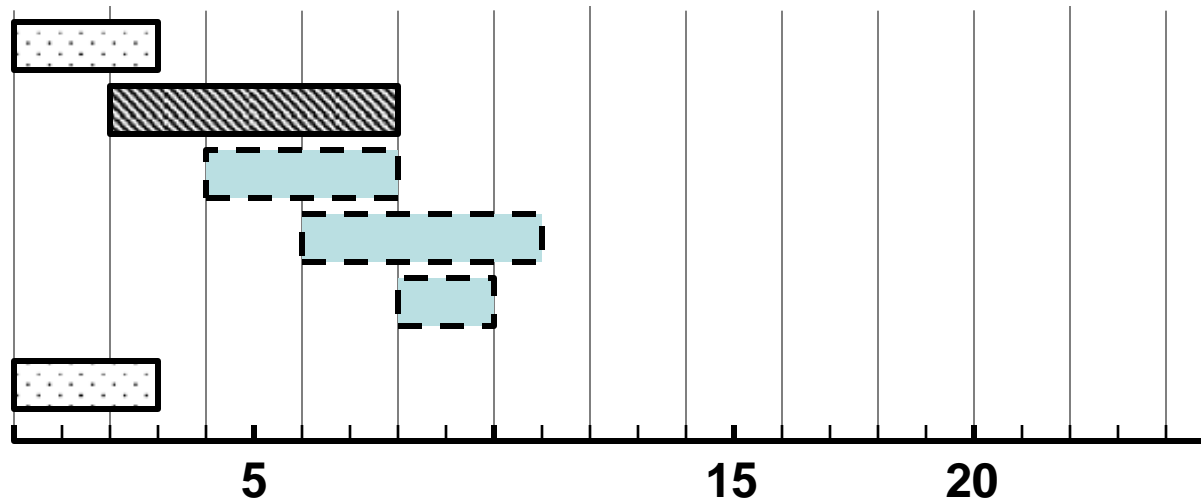
First-Come, First-Served (FIFO)

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








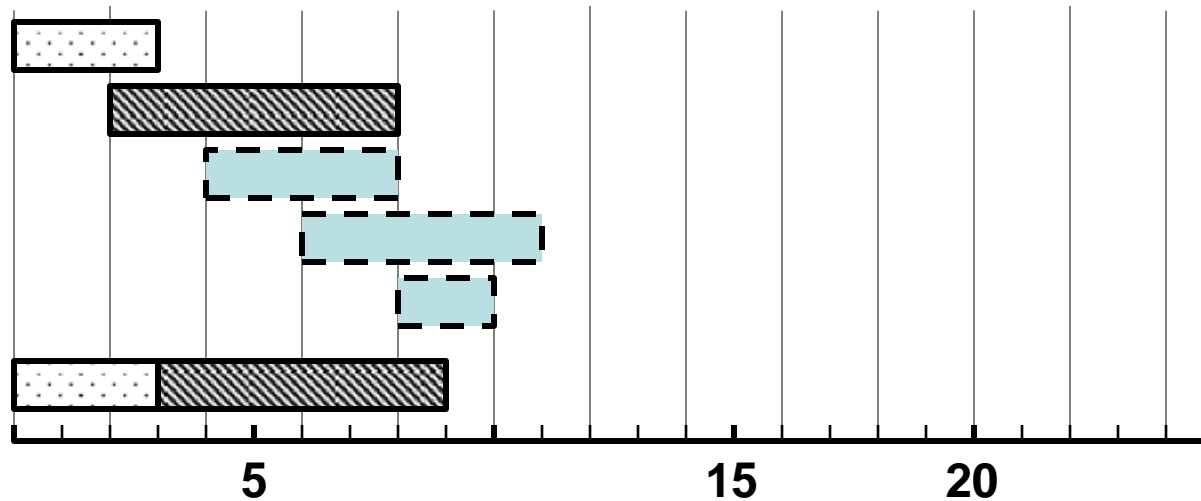
First-Come, First-Served (FIFO)

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








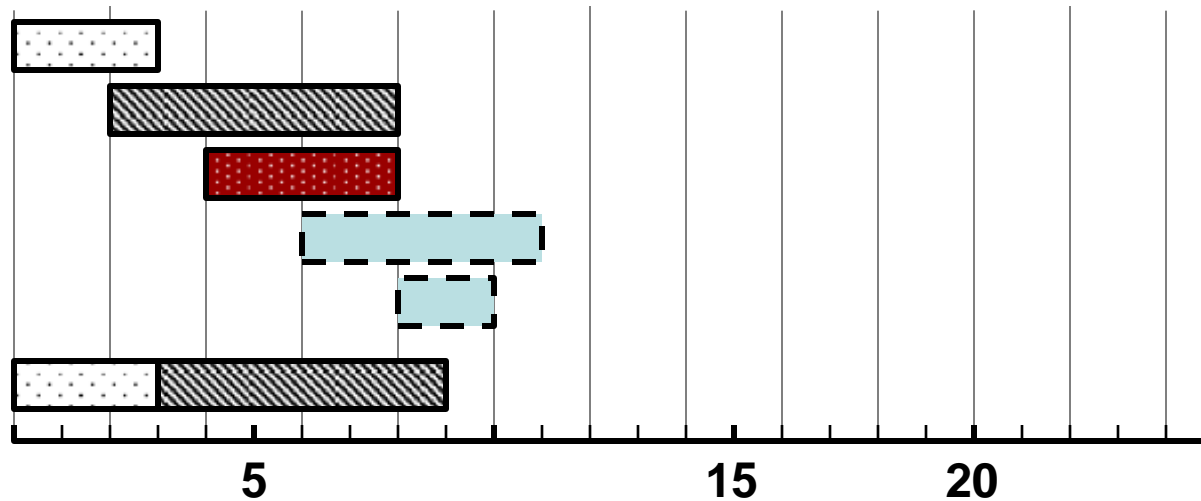
First-Come, First-Served (FIFO)

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








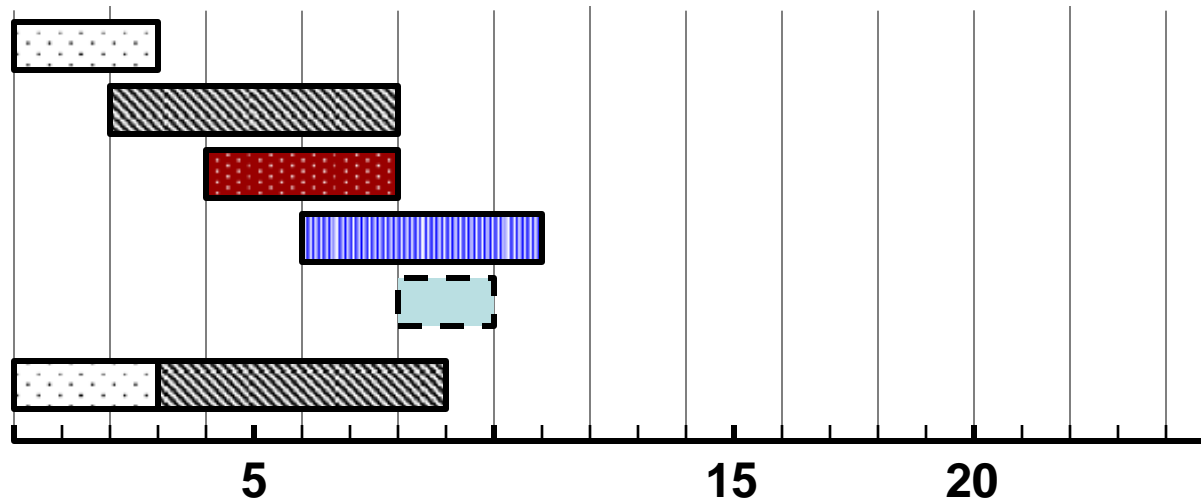
First-Come, First-Served (FIFO)

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








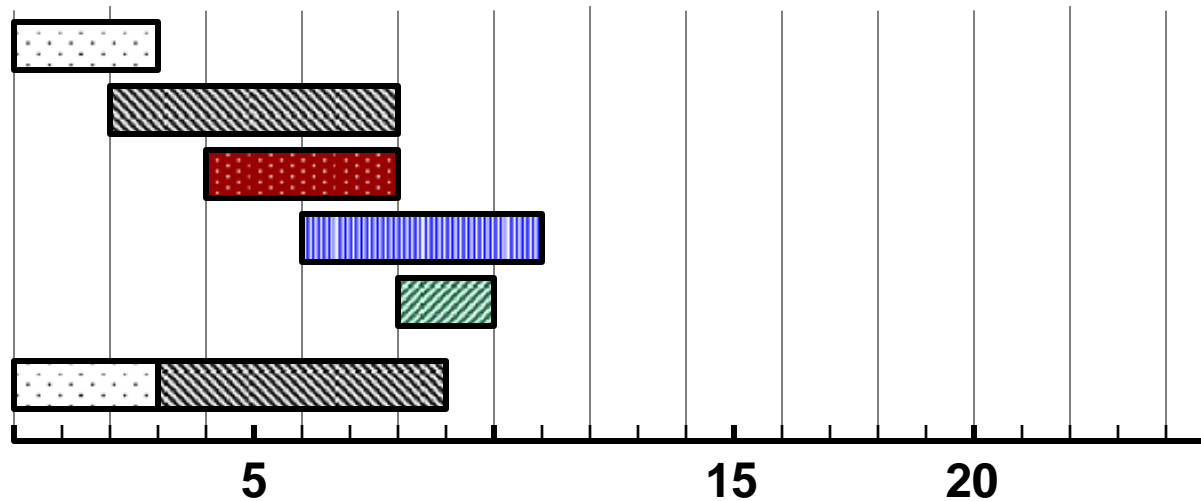
First-Come, First-Served (FIFO)

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








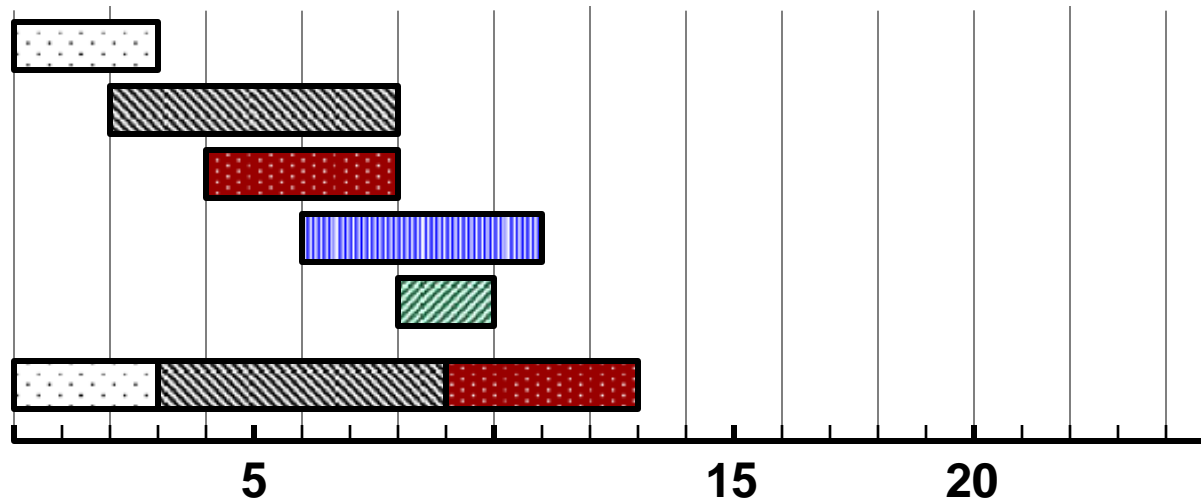
First-Come, First-Served (FIFO)

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








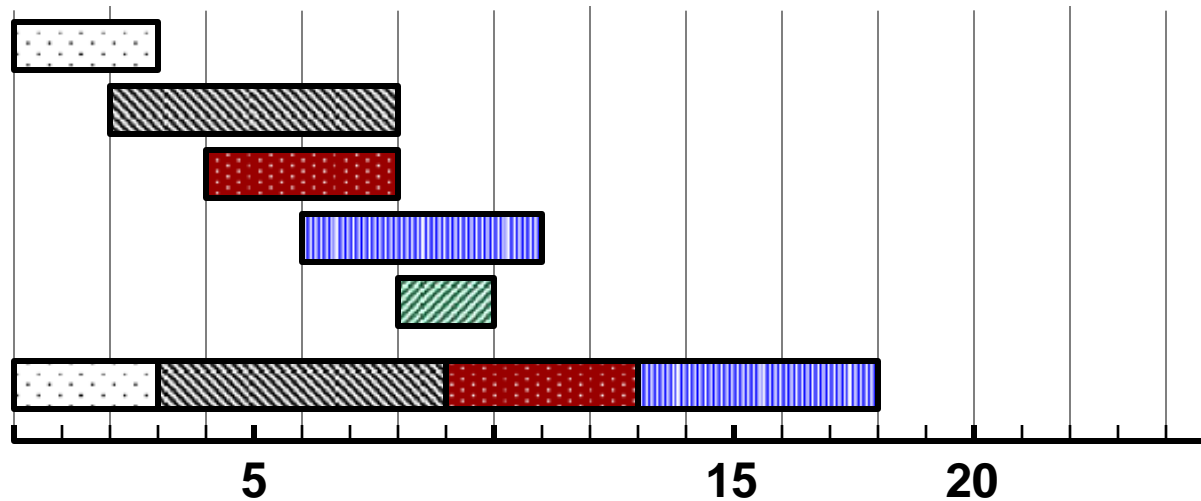
First-Come, First-Served (FIFO)

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








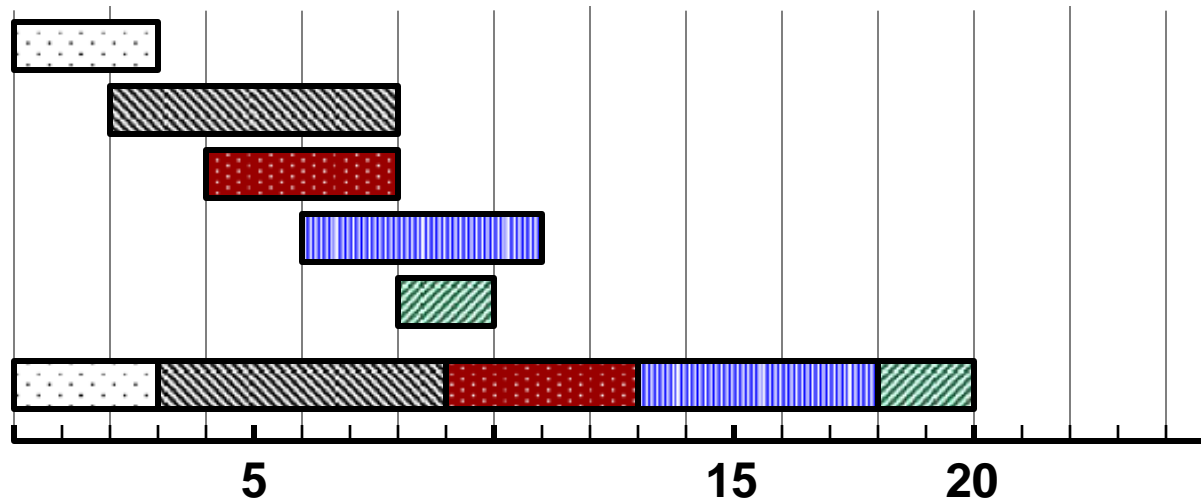
First-Come, First-Served (FIFO)

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








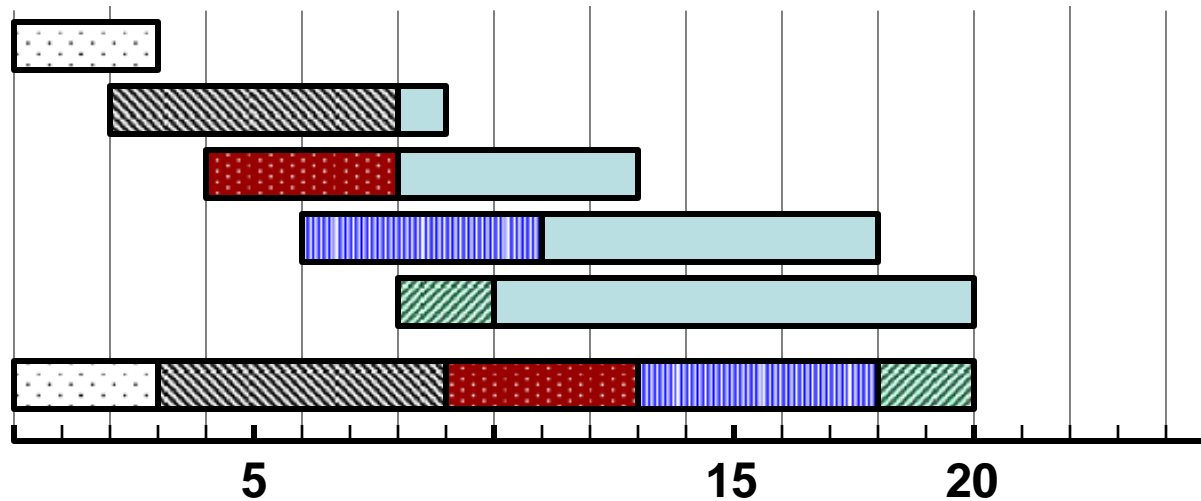
First-Come, First-Served (FIFO)

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








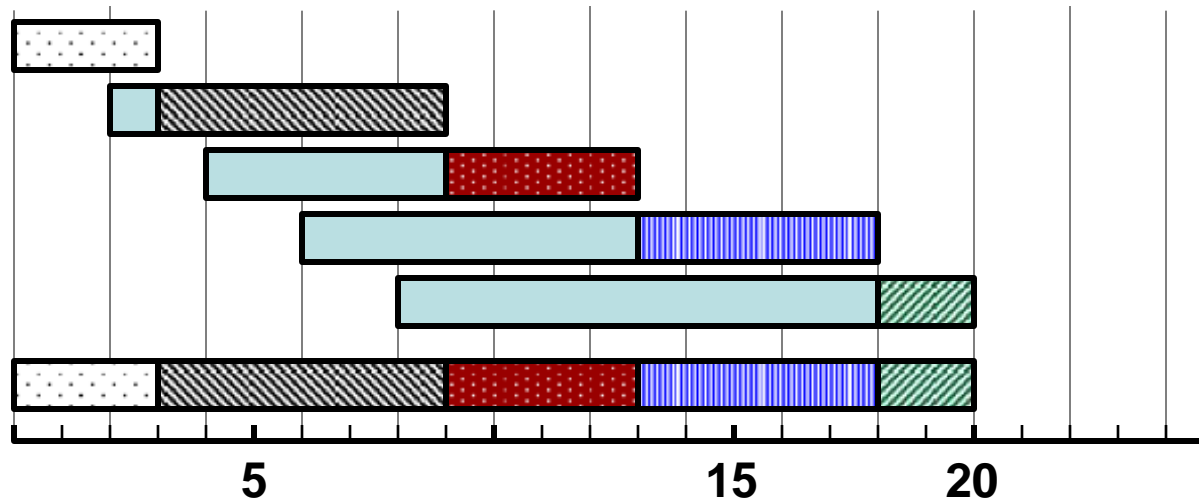
First-Come, First-Served (FIFO)

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








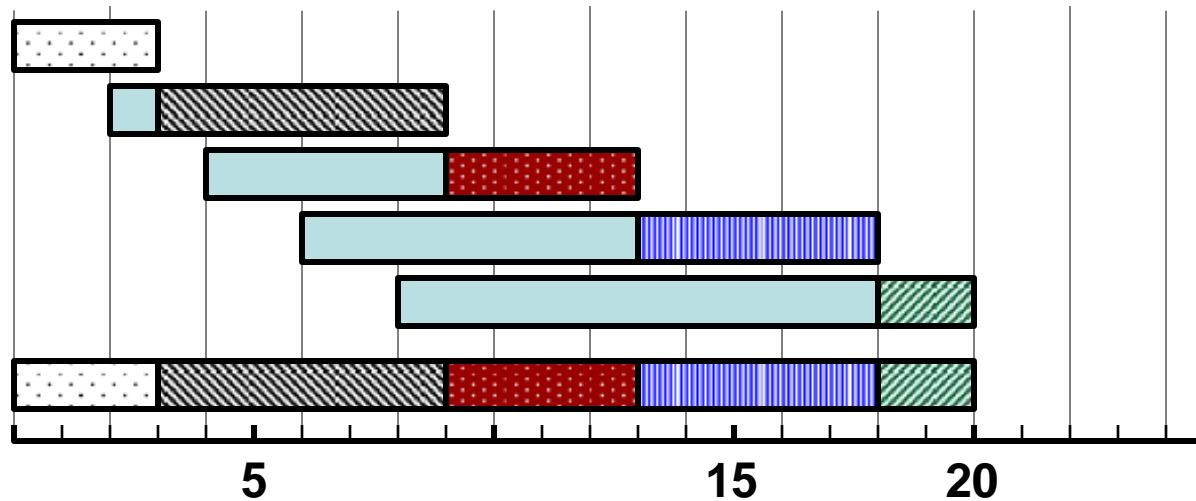
First-Come, First-Served (FIFO)

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








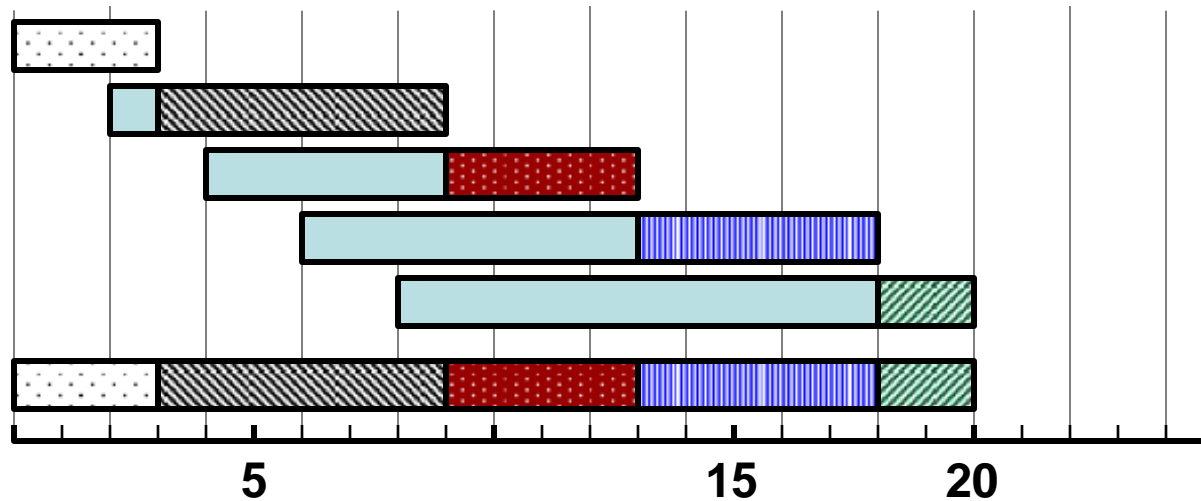
First-Come, First-Served (FIFO)

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3	0	
 2	2	6	1	
 3	4	4	5	
 4	6	5	7	
 5	8	2	10	



First-Come, First-Served (FIFO)






		Arrival	Processing	Turnaround	
Process		Time	Time	Delay	Time
	1	0	3	0	3
	2	2	6	1	7
	3	4	4	5	9
	4	6	5	7	12
	5	8	2	10	12

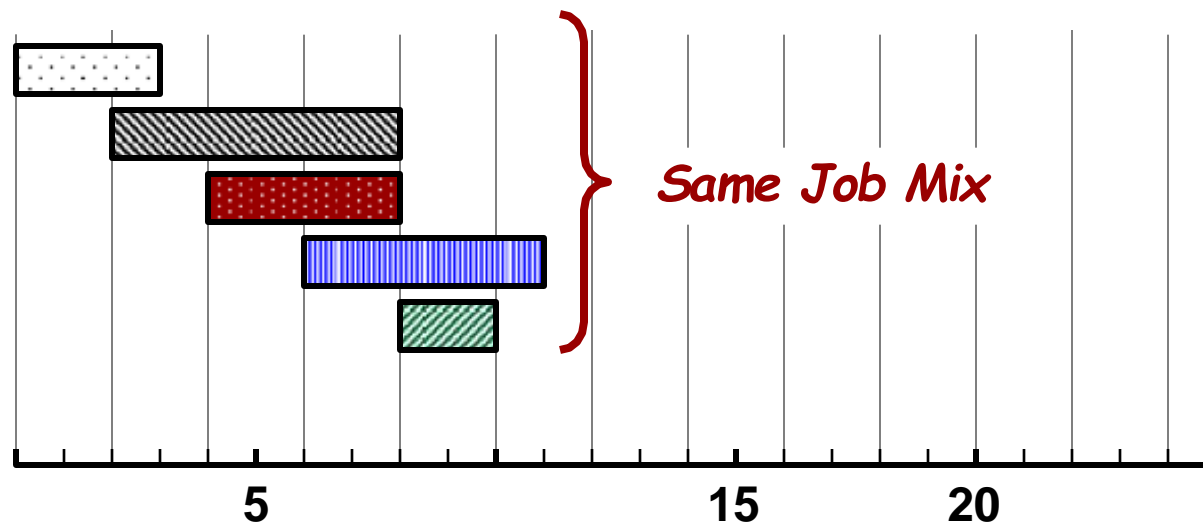


Shortest Job First






- ☐ Select the job with the shortest (expected) running time
- ☐ Non-Preemptive

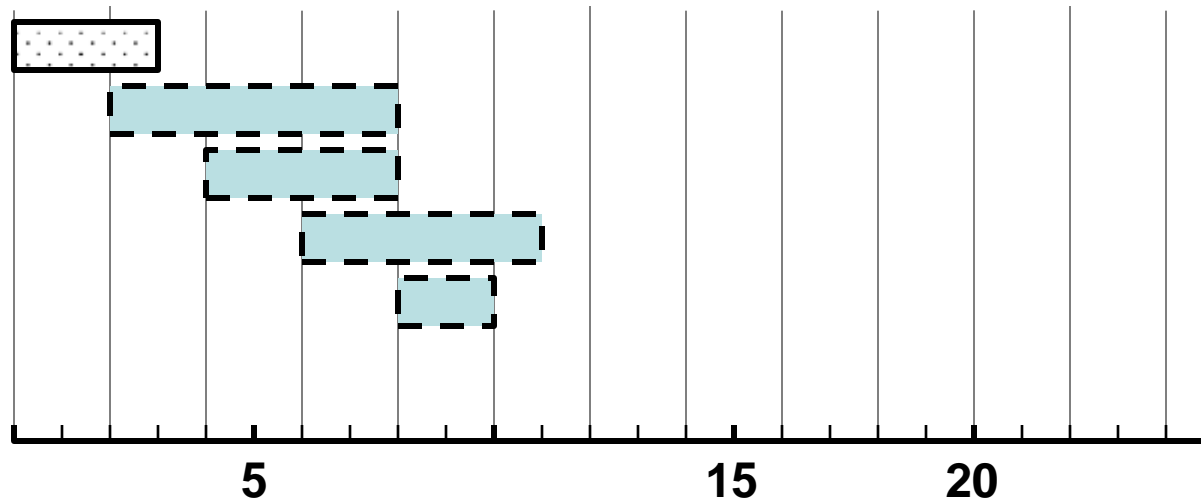
Shortest Job First

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








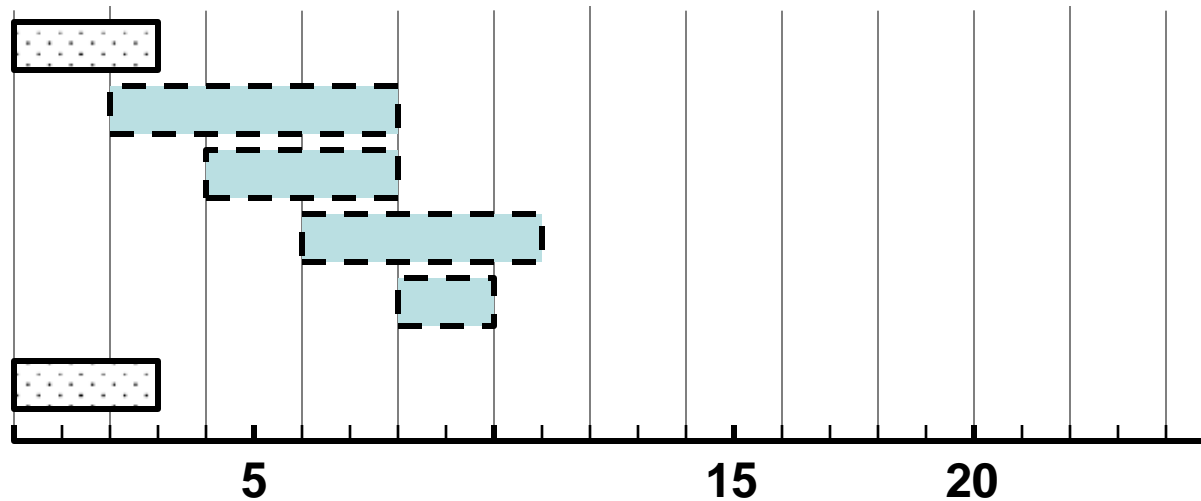
Shortest Job First

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








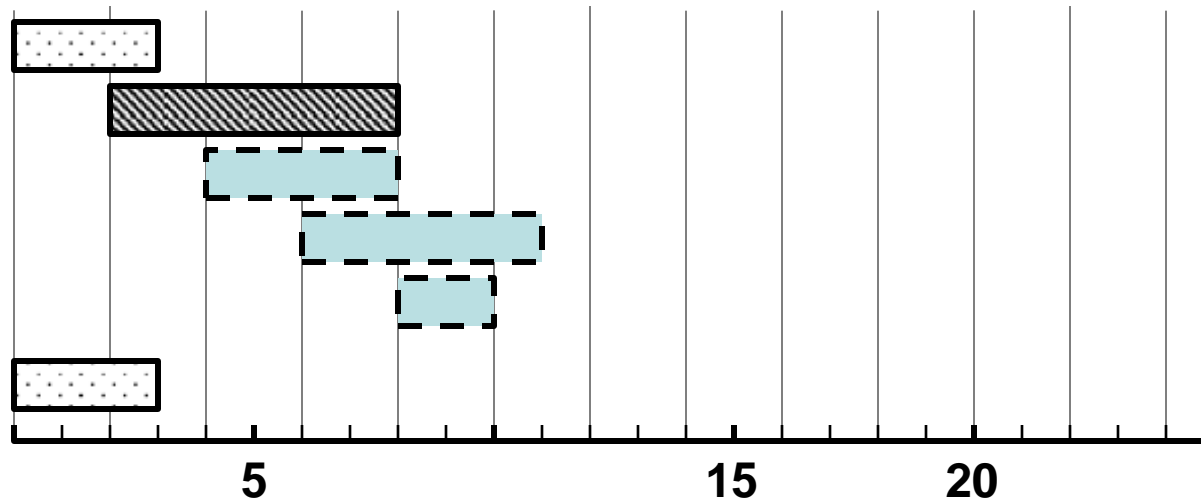
Shortest Job First

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








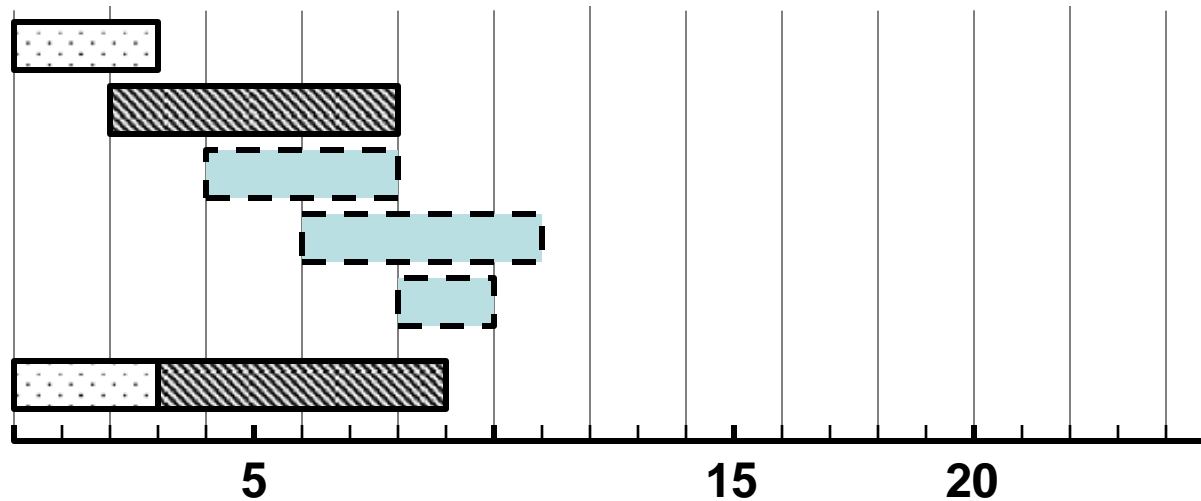
Shortest Job First

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








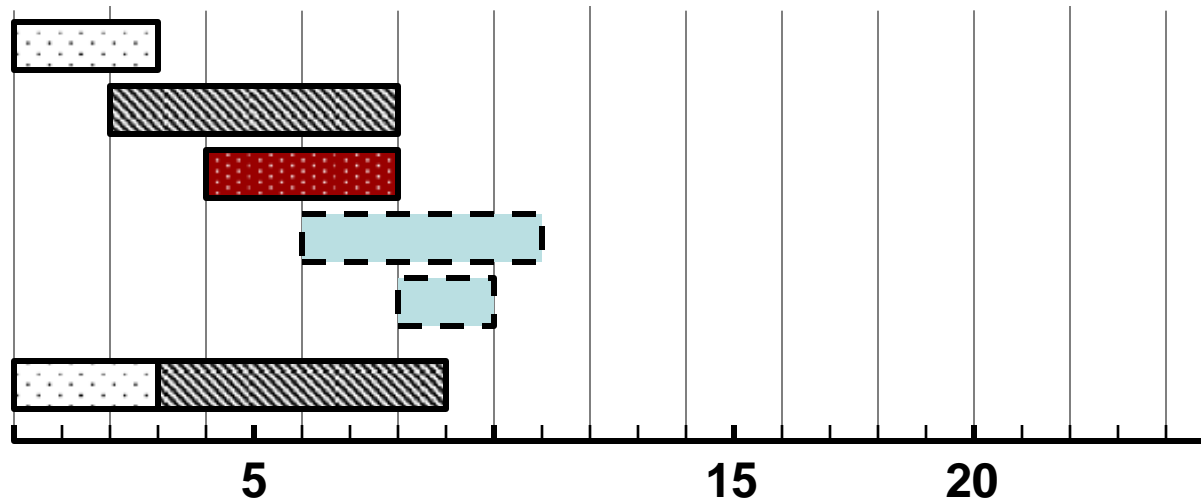
Shortest Job First

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








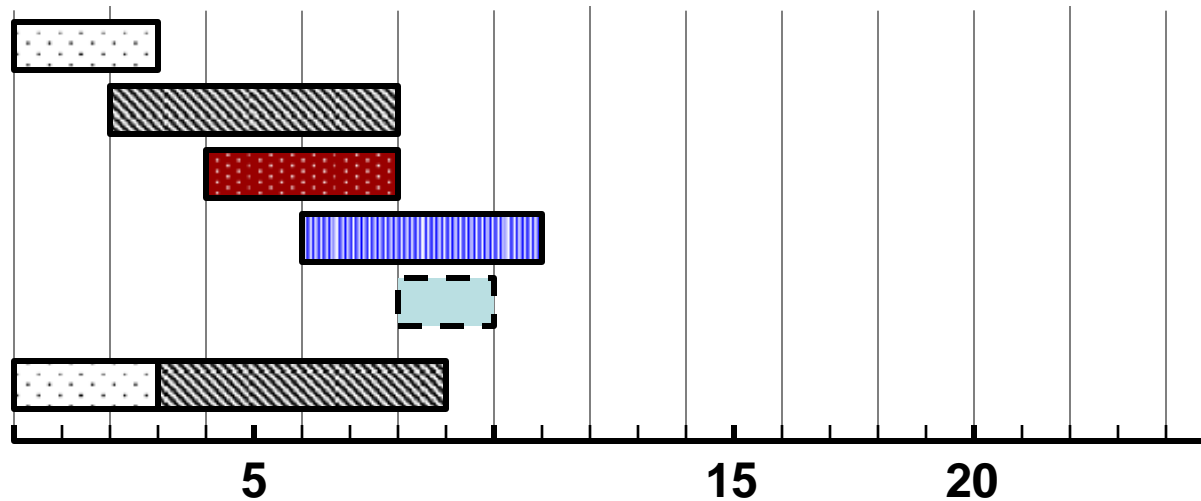
Shortest Job First

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








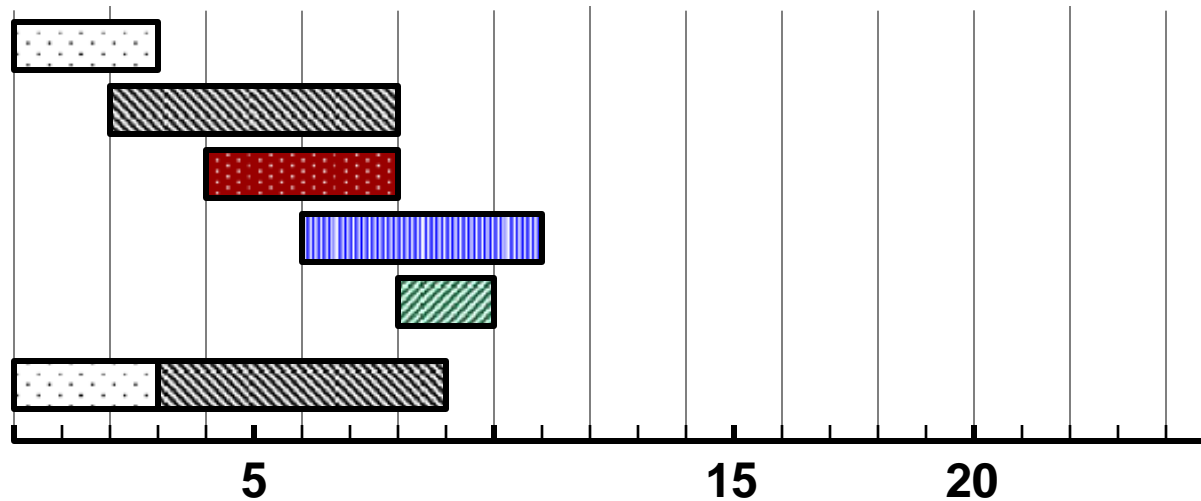
Shortest Job First

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








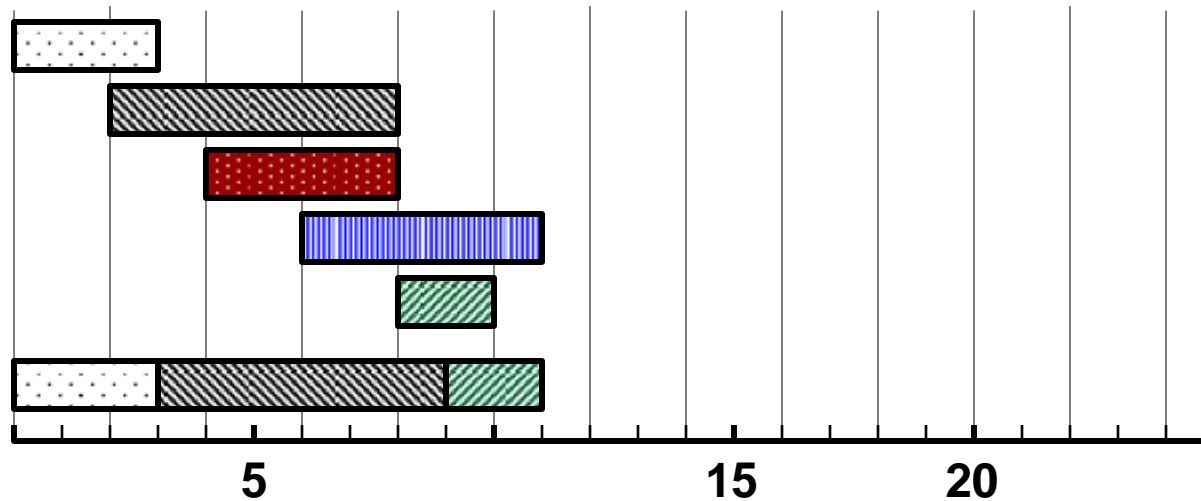
Shortest Job First

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








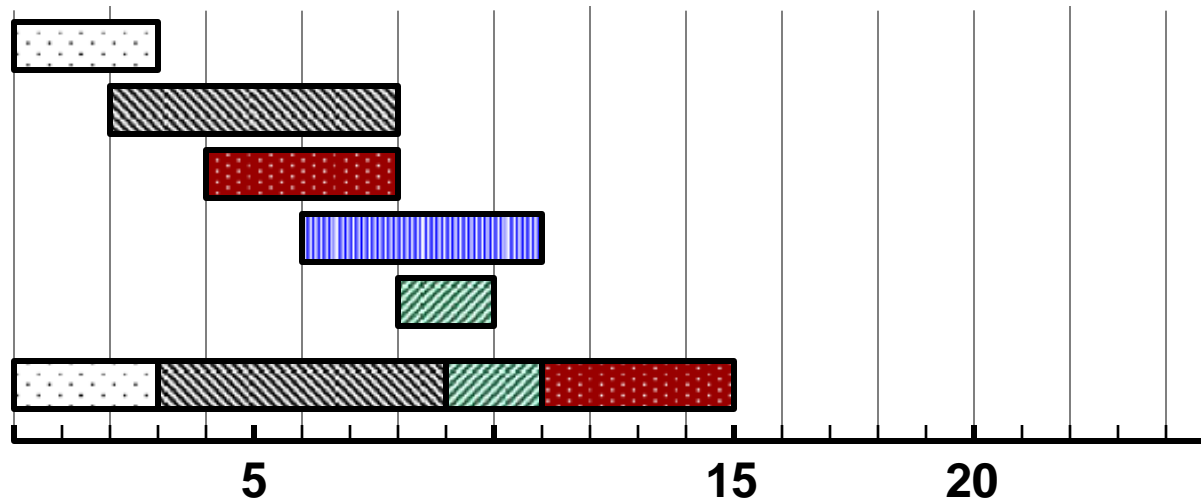
Shortest Job First

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








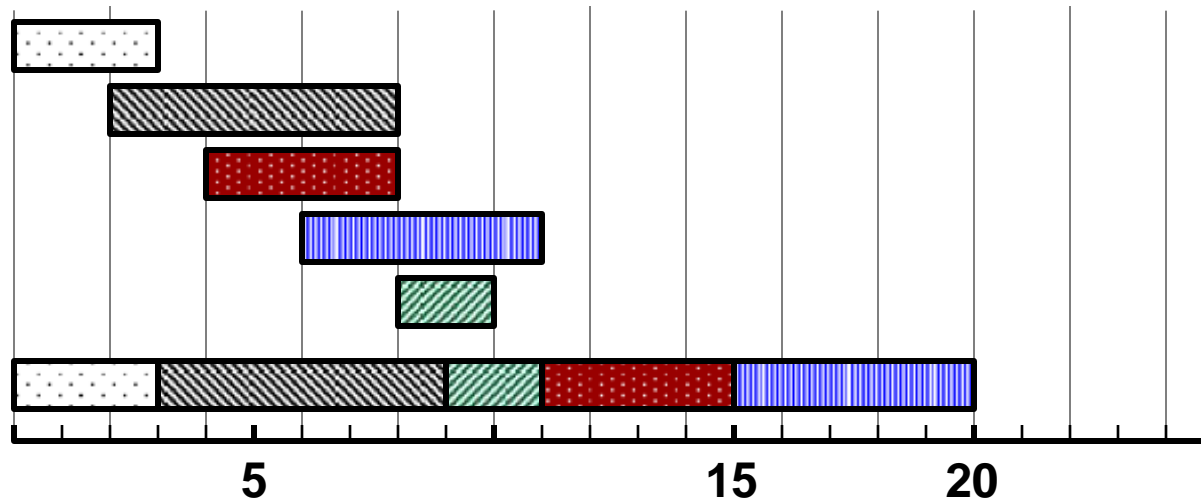
Shortest Job First

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








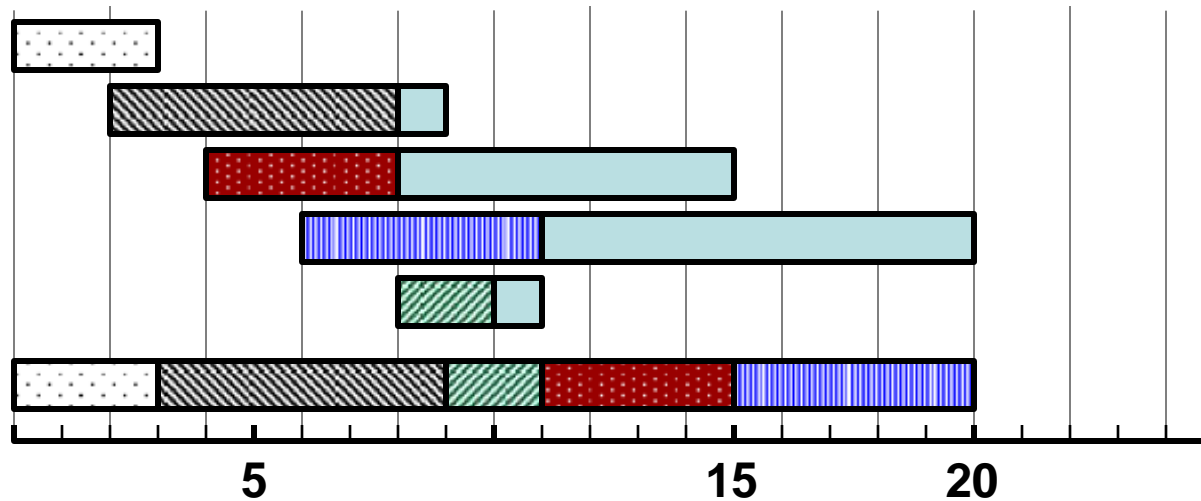
Shortest Job First

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








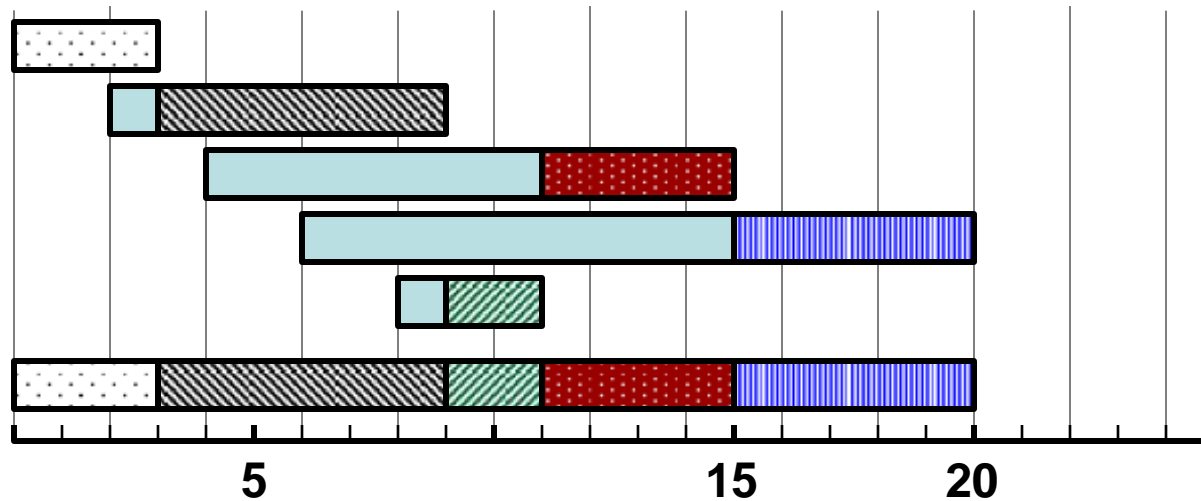
Shortest Job First

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








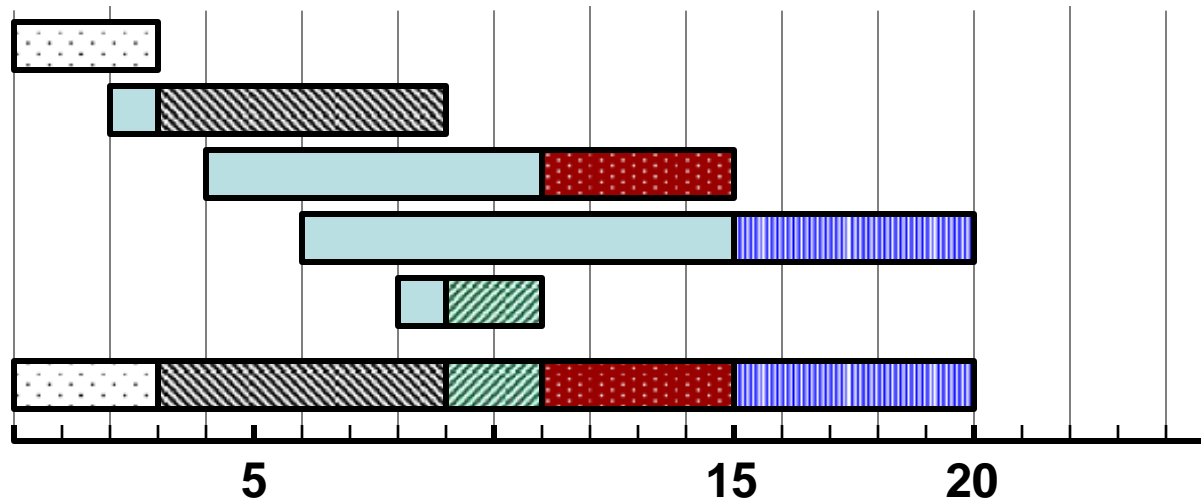
Shortest Job First

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








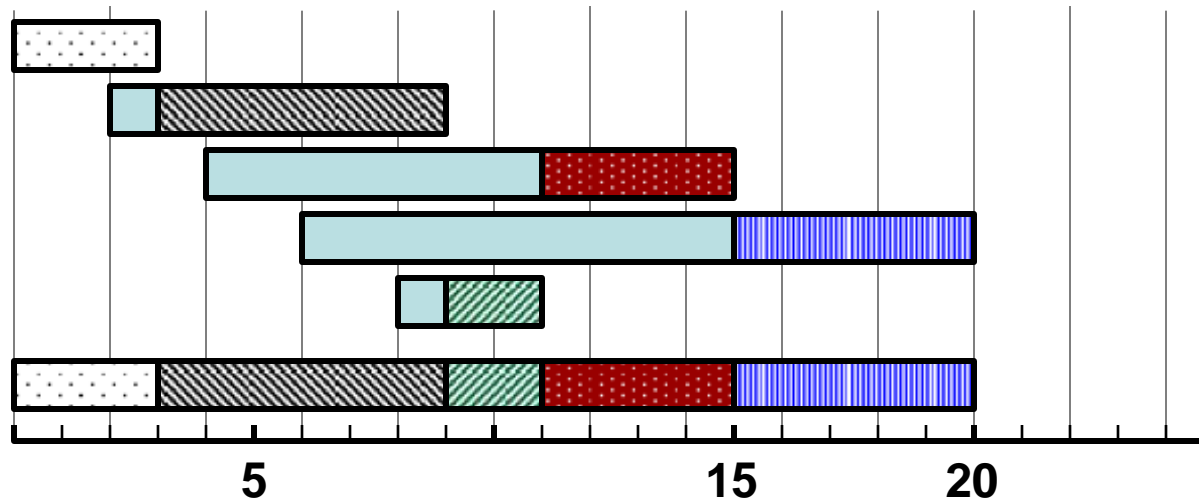
Shortest Job First

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3	0	
 2	2	6	1	
 3	4	4	7	
 4	6	5	9	
 5	8	2	1	



Shortest Job First

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3	0	3
 2	2	6	1	7
 3	4	4	7	11
 4	6	5	9	14
 5	8	2	1	3



اول کوتاه‌ترین کار، بهینه است

- قضیه: در حالت Non-preemptive، اگر الگوریتم SJF را اجرا کنیم، بهینه‌ترین حالت از نظر Delay و Turnaround Time را خواهیم داشت.
- اثبات؟

اول کوتاه‌ترین کار، بهینه است

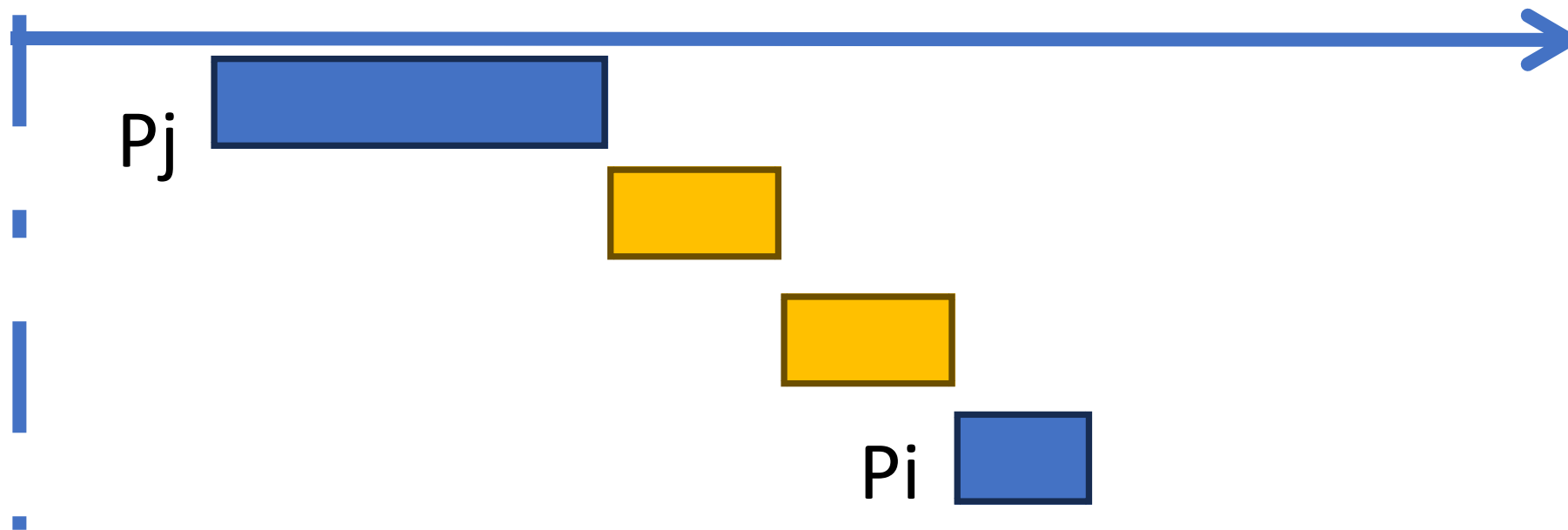
- برهان خلف
- فرض کنید برای پردازش‌های P_1, \dots, P_n ، با زمان مصرف پردازنده‌ی B_1, \dots, B_n ، روش اجرای X بهتر باشد.
- این X ، بیشترین شباهت پیشوندی بین اجرای بهینه به SJF داشته باشد ولی SJF نباشد...
- این روش اجرا در یک جا، پردازش‌ای که کوتاه‌ترین نیست را انتخاب کند.
- مثلاً P_j پیش از P_i که هر دو آماده هستند انتخاب شود، ولی $B_i < B_j$

اول کوتاه‌ترین کار، بهینه است

■ مثلاً P_j پیش از P_i که هر دو آماده هستند انتخاب شود،
ولی $B_i < B_j$

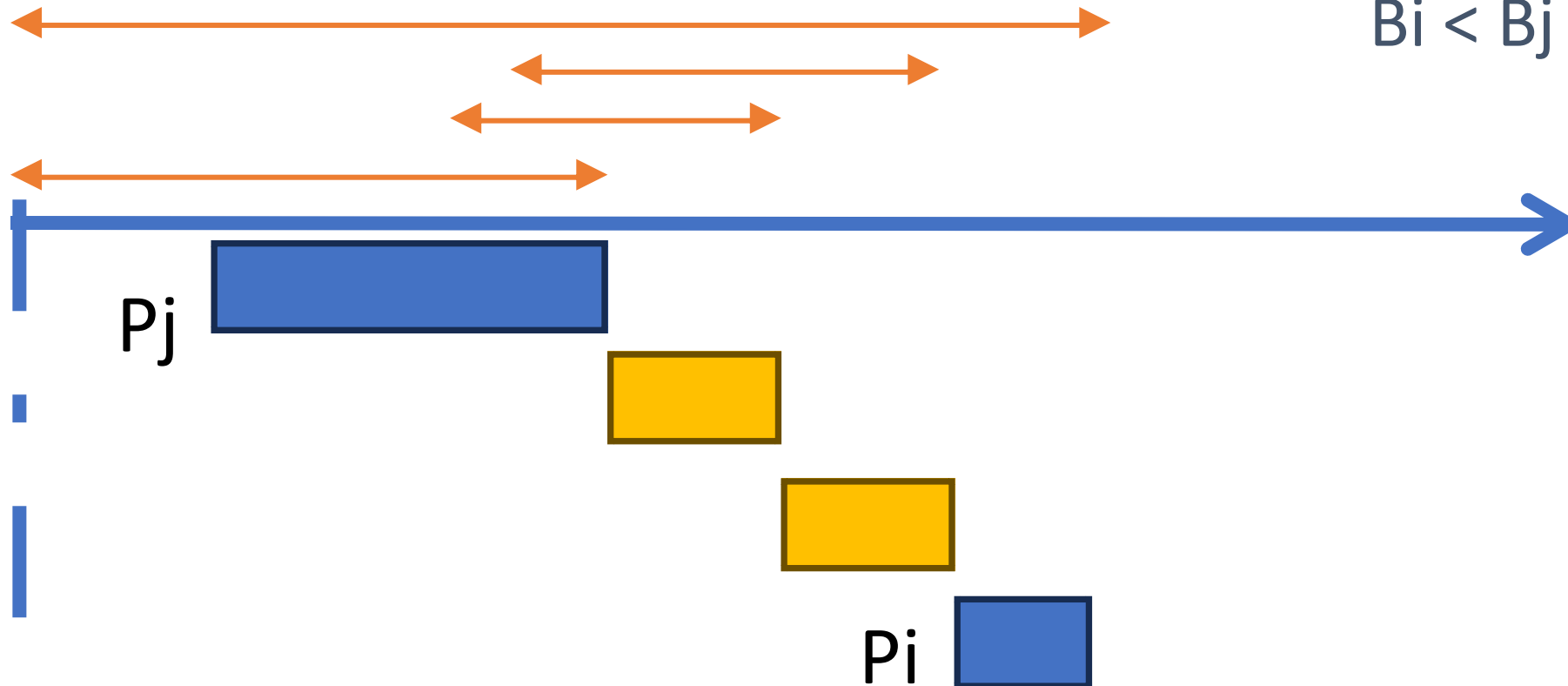
اول کوتاه‌ترین کار، بهینه است

■ مثلاً P_j پیش از P_i که هر دو آماده هستند انتخاب شود،
ولی $B_i < B_j$



اول کوتاه‌ترین کار، بهینه است

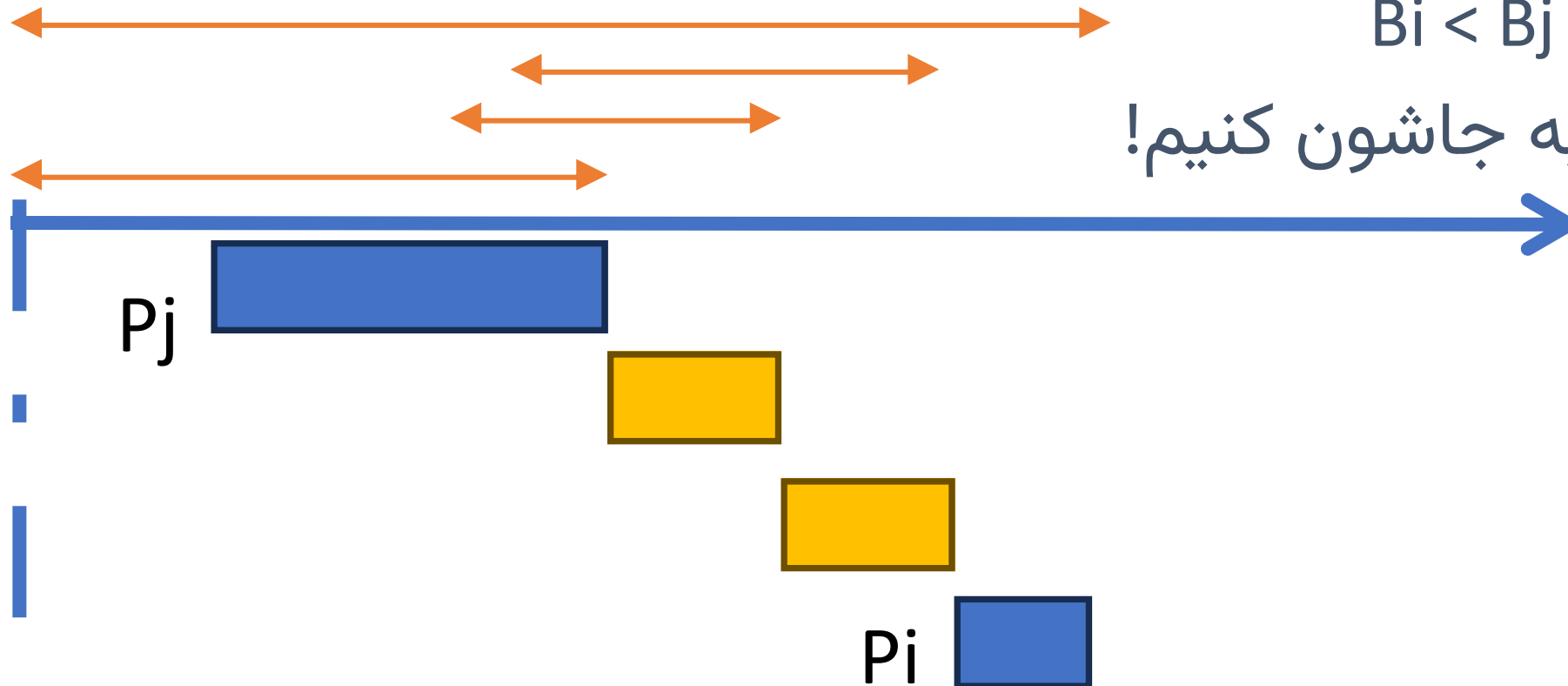
■ مثلاً P_j پیش از P_i که هر دو آماده هستند انتخاب شود، ولی $B_i < B_j$



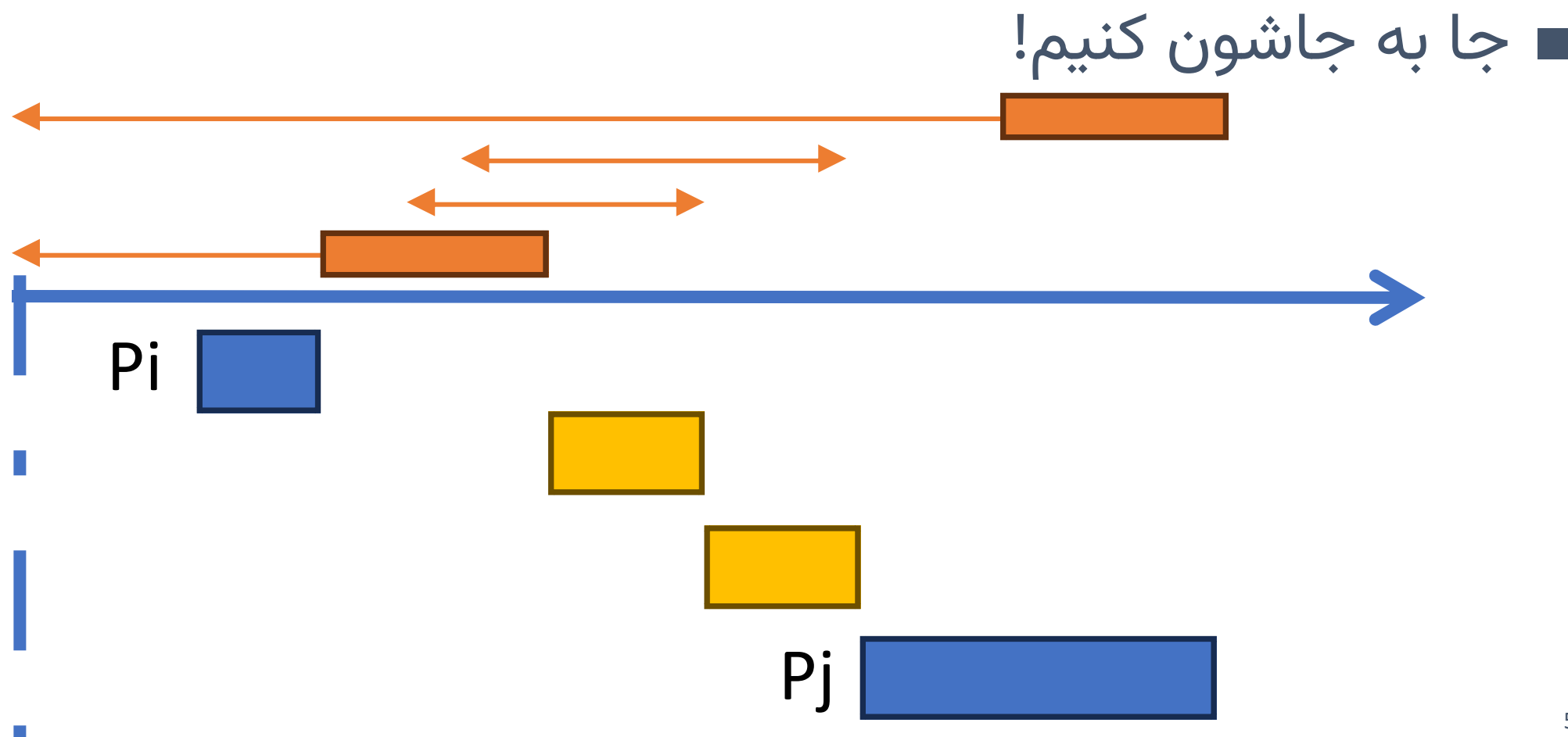
اول کوتاه‌ترین کار، بهینه است

■ مثلاً P_j پیش از P_i که هر دو آماده هستند انتخاب شود، ولی $B_i < B_j$

■ جا به جاشون کنیم!



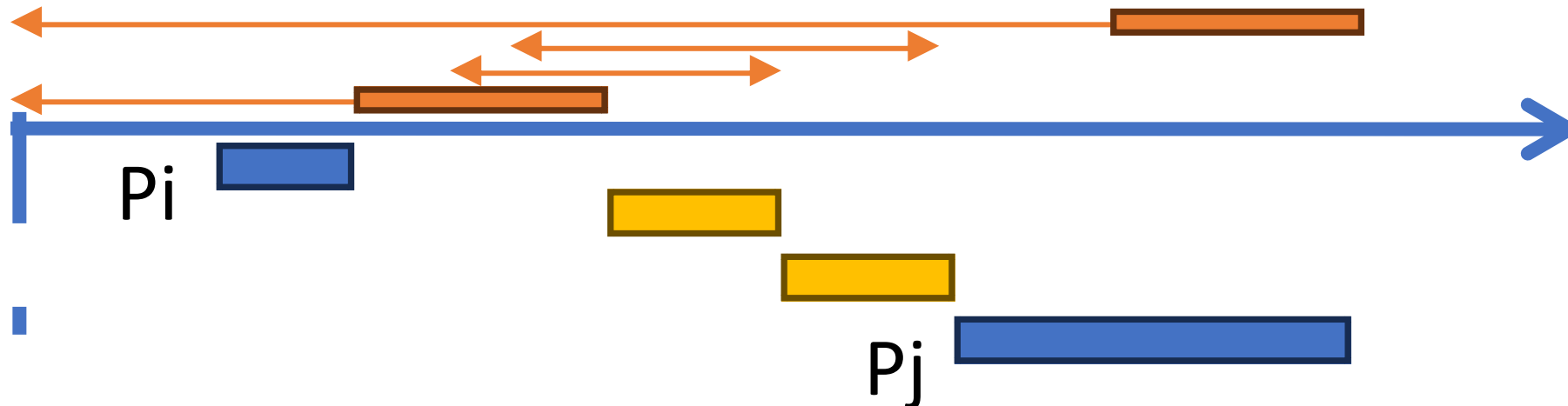
اول کوتاه‌ترین کار، بهینه است



اول کوتاه‌ترین کار، بهینه است

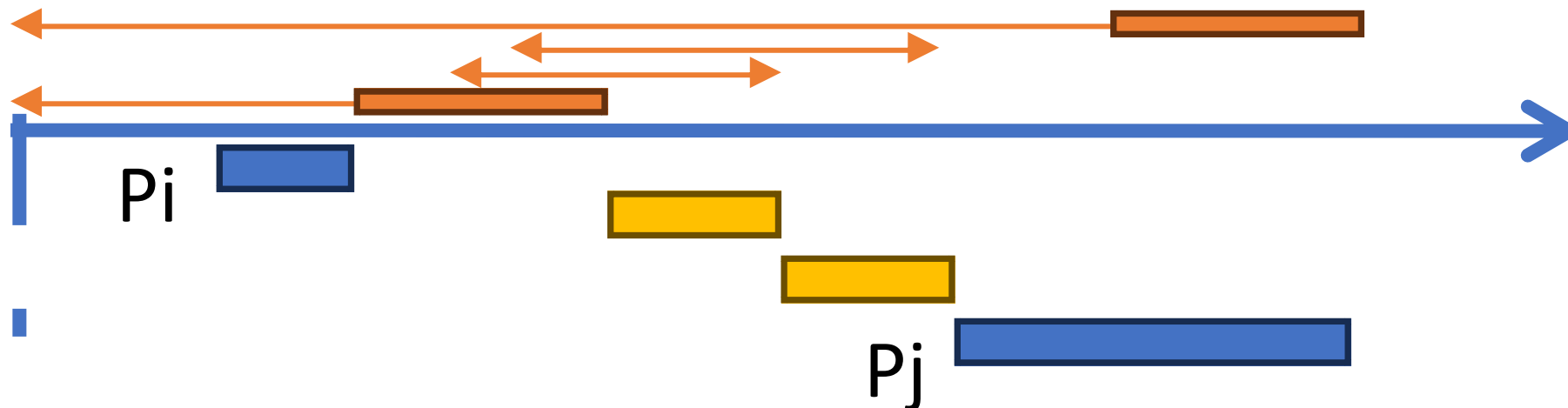
■ جا به جاشون کنیم!

■ حتی اگه بقیه پردازها نتونن زودتر اجرا بشند، turn around time کمتر مساوی هست.



اول کوتاه‌ترین کار، بهینه است

- جا به جاشون کنیم!
- حتی اگه بقیه پردازها نتونن زودتر اجرا بشند، میانگین turn around time کمتر مساوی هست.
- البته این الگوریتم هنوز SJF نیستا








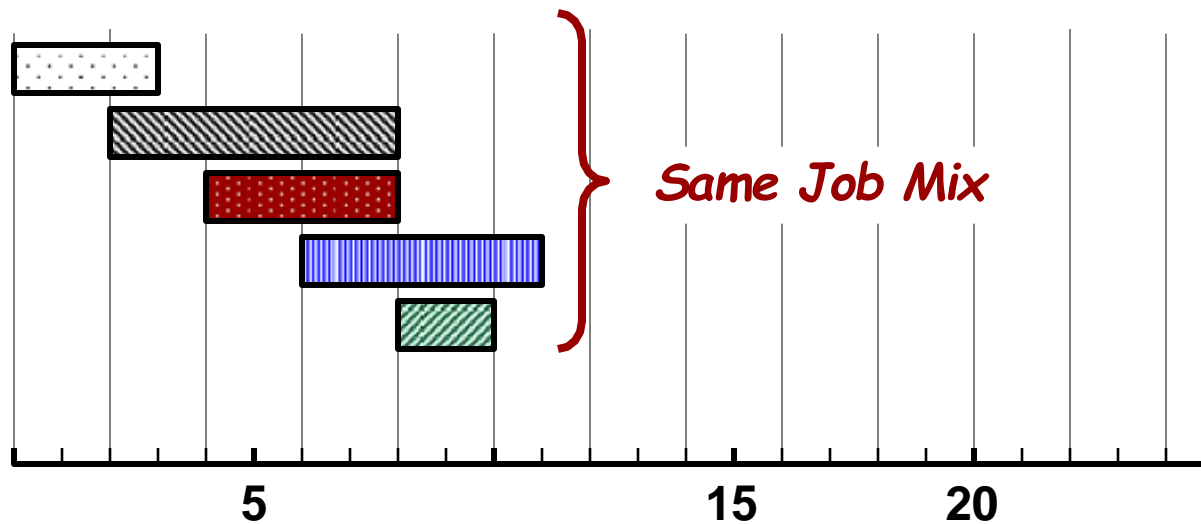
اول کوتاه‌ترین کار، بهینه است

- جا به جاشون کنیم!
- حتی اگه بقیه پردازها نتونن زودتر اجرا بشند، میانگین turn around time کمتر مساوی هست.
- البته این الگوریتم لزوما SJF نیست!
- ولی از نظر پیشوندی نزدیک‌تر شد به SJF و با نزدیک‌ترینی که فرض کرده بودیم در تناقض است!






Shortest Remaining Time

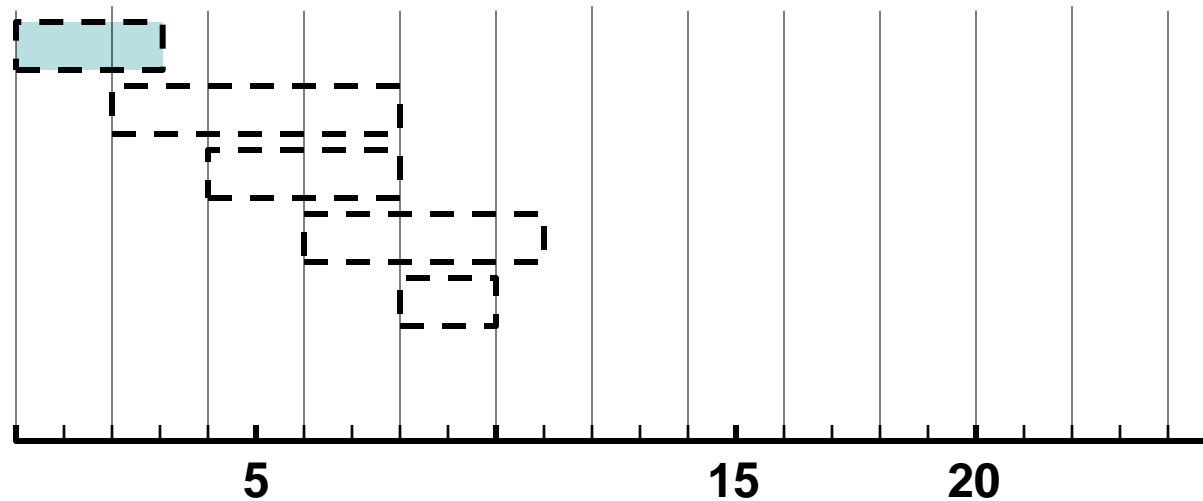
Preemptive version of SJF

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








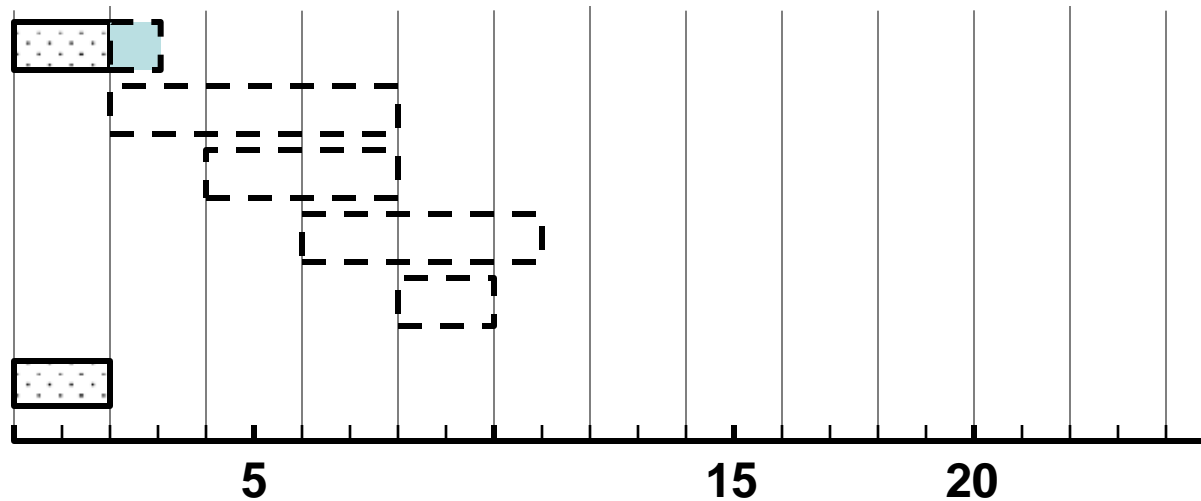
Shortest Remaining Time

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








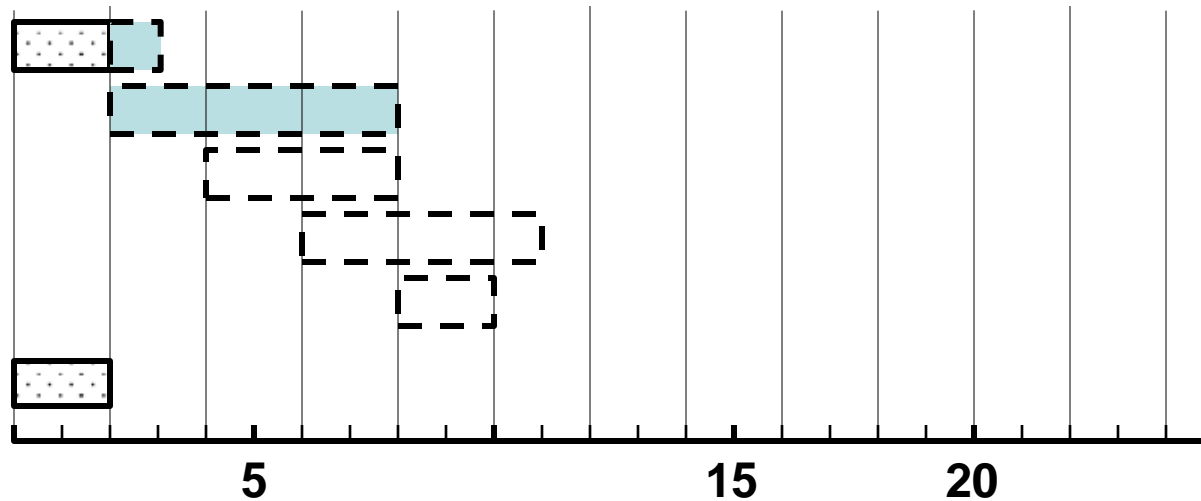
Shortest Remaining Time

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








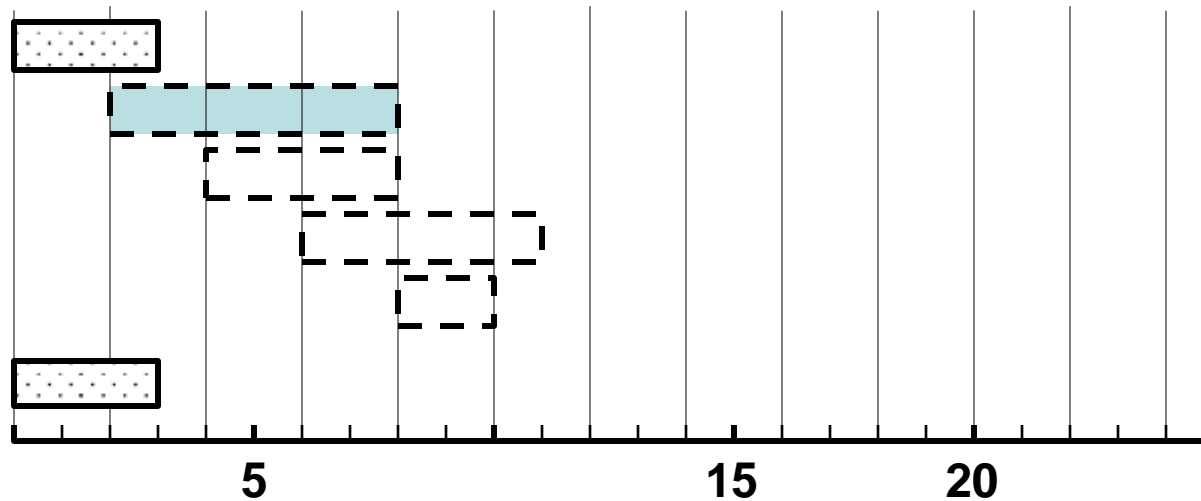
Shortest Remaining Time

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








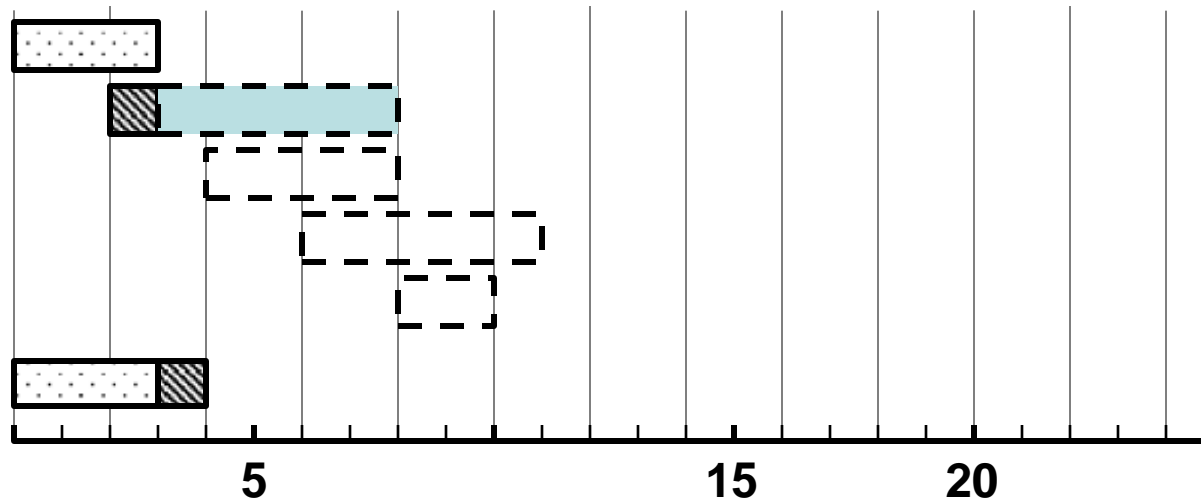
Shortest Remaining Time

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








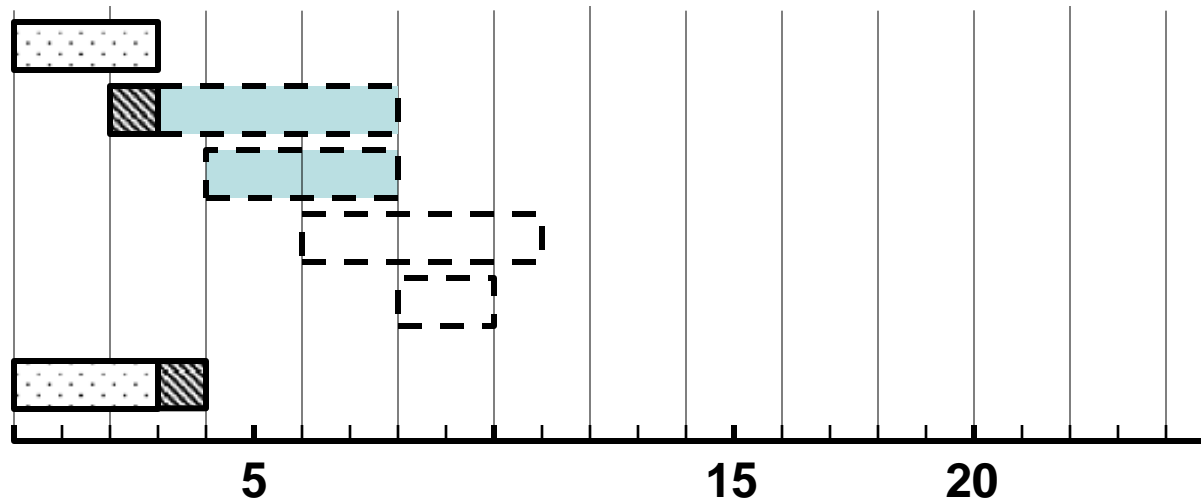
Shortest Remaining Time

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








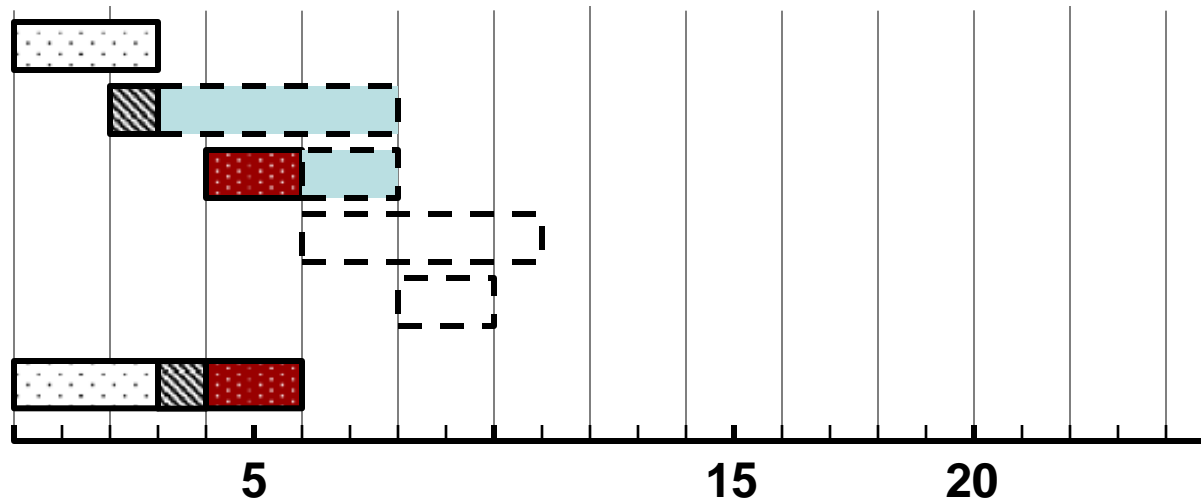
Shortest Remaining Time

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








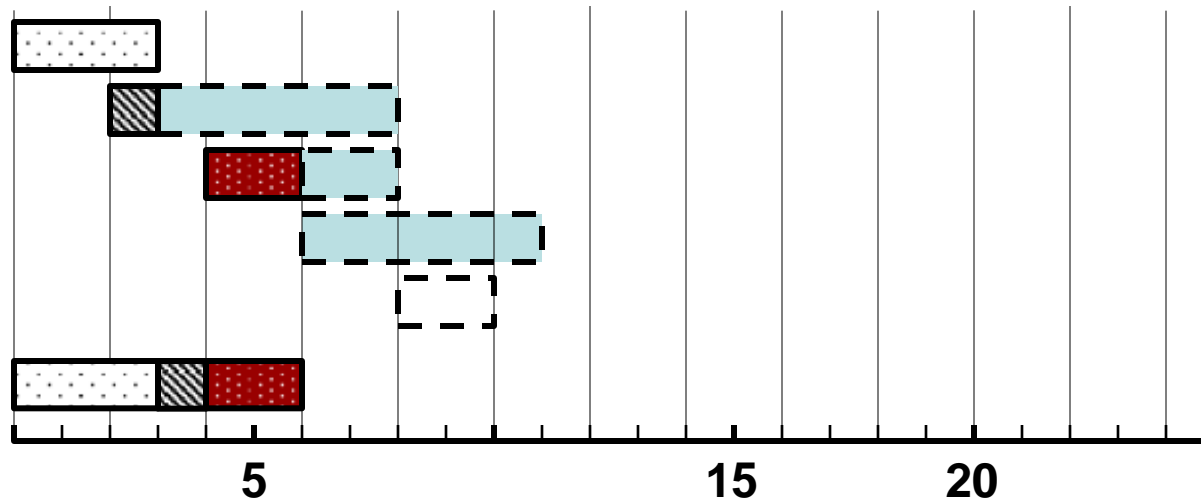
Shortest Remaining Time

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








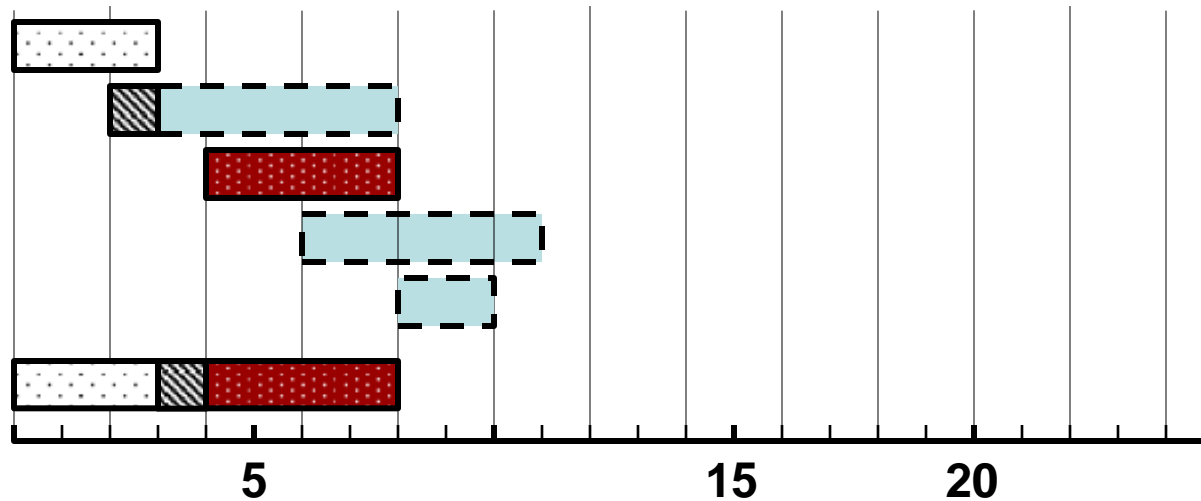
Shortest Remaining Time

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








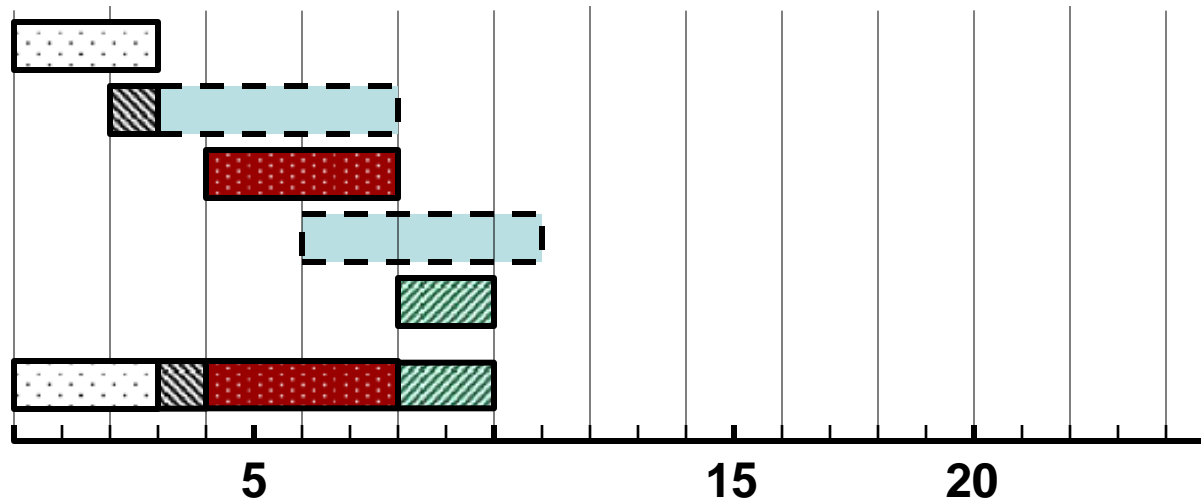
Shortest Remaining Time

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








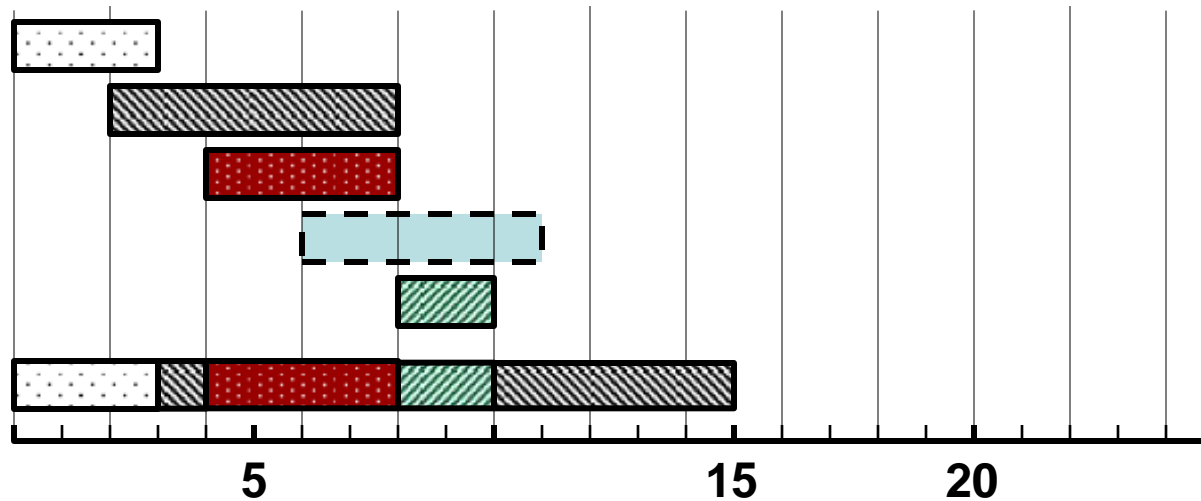
Shortest Remaining Time

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








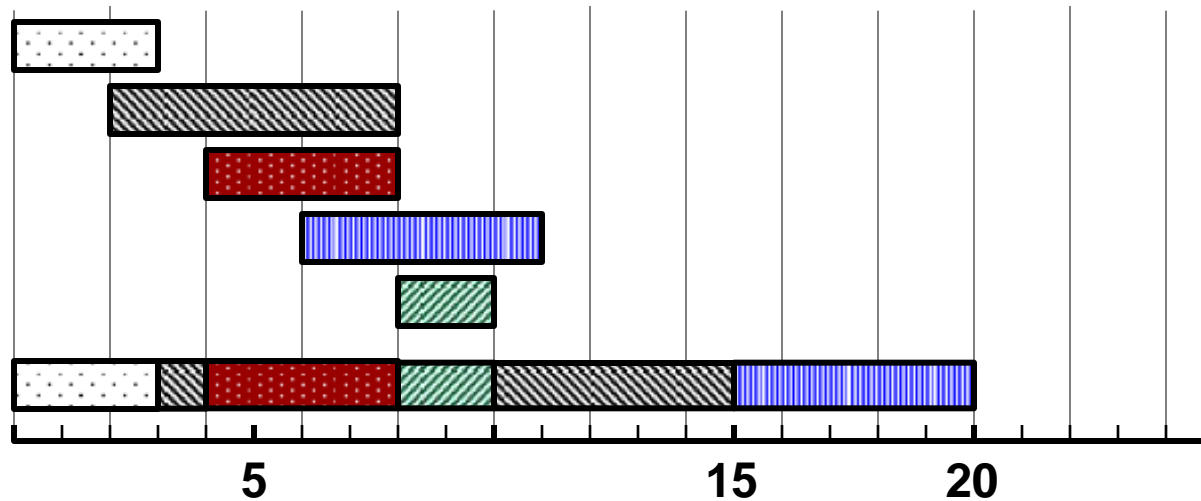
Shortest Remaining Time

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








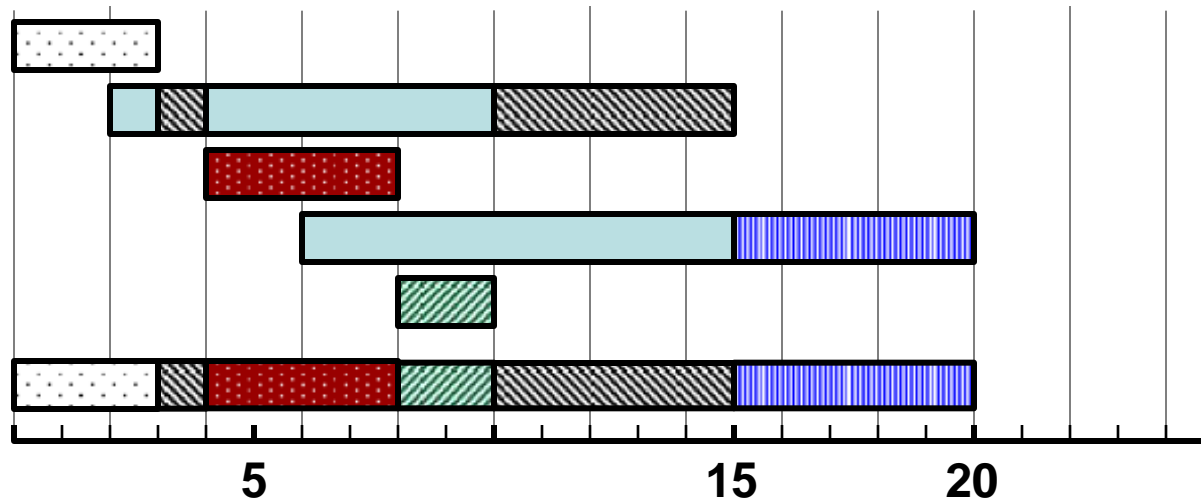
Shortest Remaining Time

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








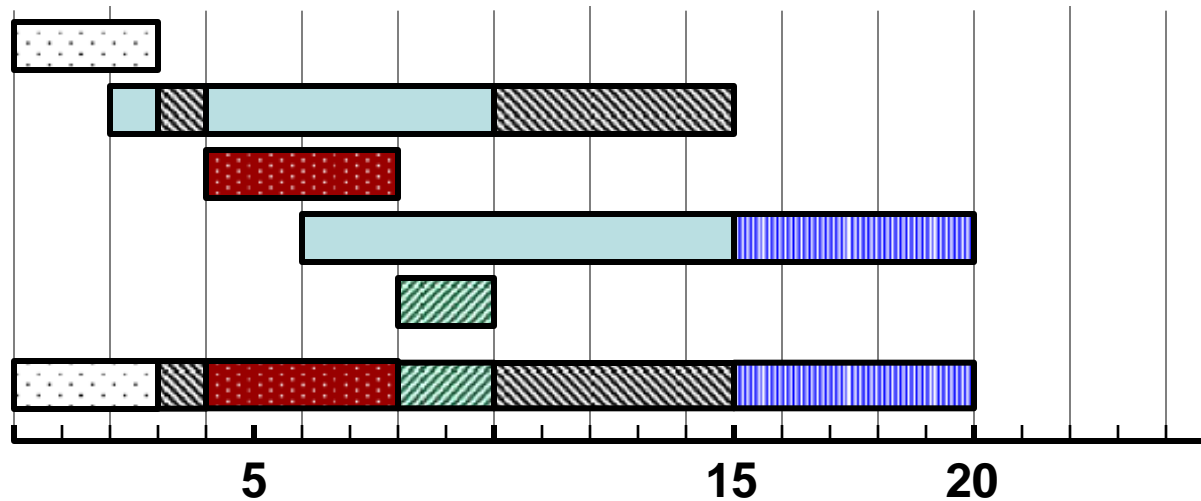
Shortest Remaining Time

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3		
 2	2	6		
 3	4	4		
 4	6	5		
 5	8	2		








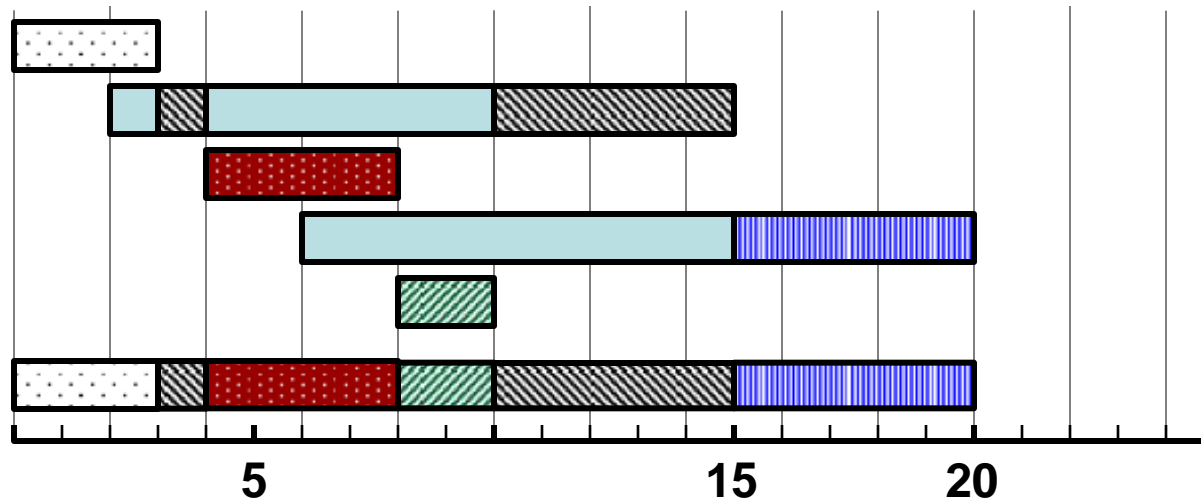
Shortest Remaining Time

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3	0	
 2	2	6	7	
 3	4	4	0	
 4	6	5	9	
 5	8	2	0	



Shortest Remaining Time

Process	Arrival Time	Processing Time	Delay	Turnaround Time
 1	0	3	0	3
 2	2	6	7	13
 3	4	4	0	4
 4	6	5	9	14
 5	8	2	0	2



اول کم مانده ترین کار، بهینه است

■ قضیه: در حالت Preemptive ، اگر الگوریتم SRF را اجرا کنیم، بهینه ترین حالت از نظر Delay و Turnaround Time را خواهیم داشت.

■ اثباتی مشابه...






■ اما، در عمل چطور بدونیم چقدر طول خواهد کشید یک پردازش؟
☹️

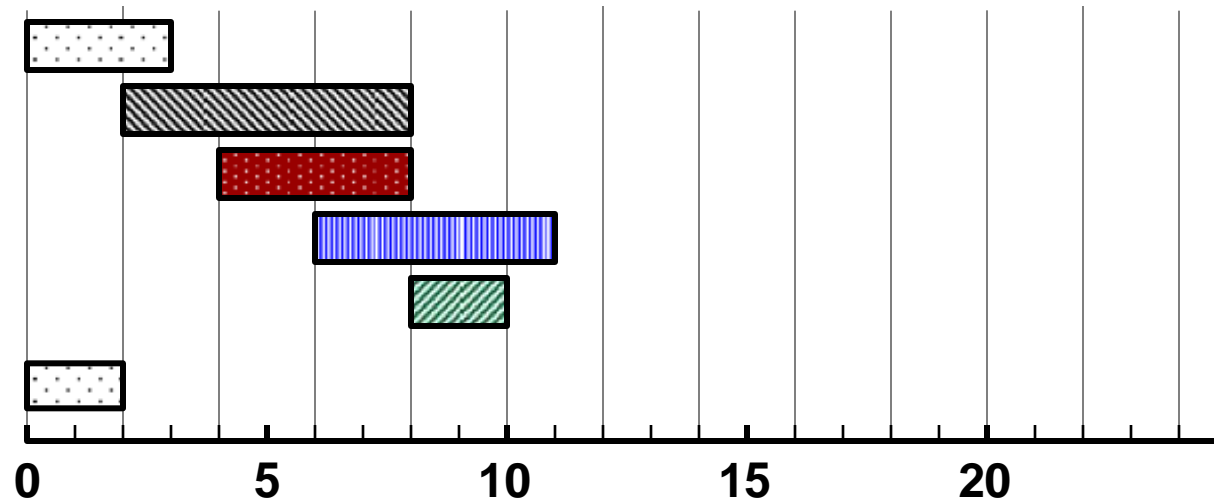
■ در عمل نمی توانیم... پس تلاش کنیم الگوریتم های دیگر بدیم
☹️

Round-Robin Scheduling






- Goal: Enable interactivity
 - Limit the amount of CPU that a process can have at one time.
- Time quantum
 - Amount of time the OS gives a process before intervention
 - The “time slice”
 - *Typically: 10 to 100ms*

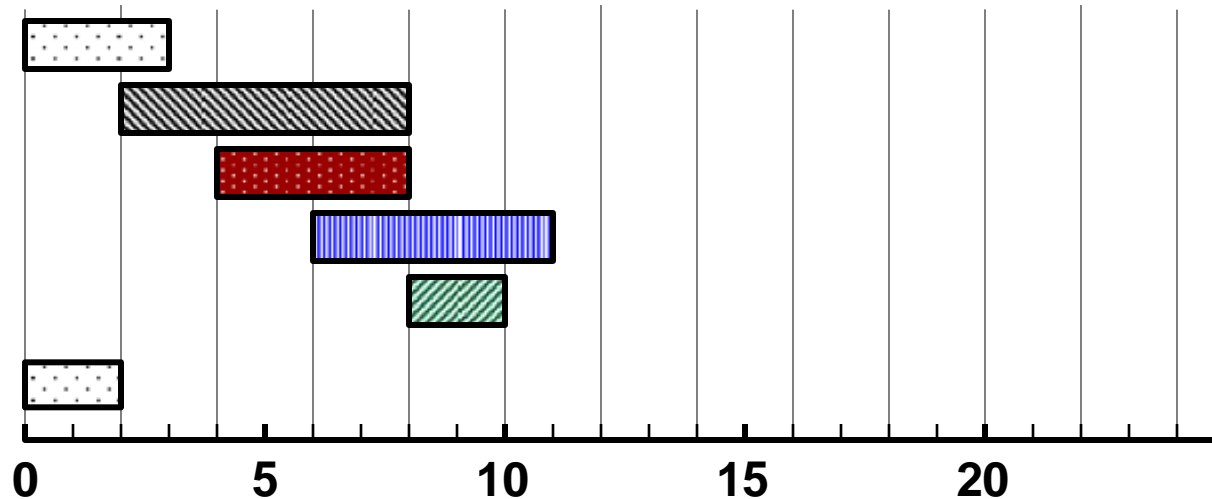
Round-Robin Scheduling

Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

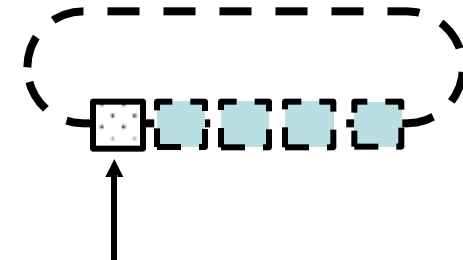


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

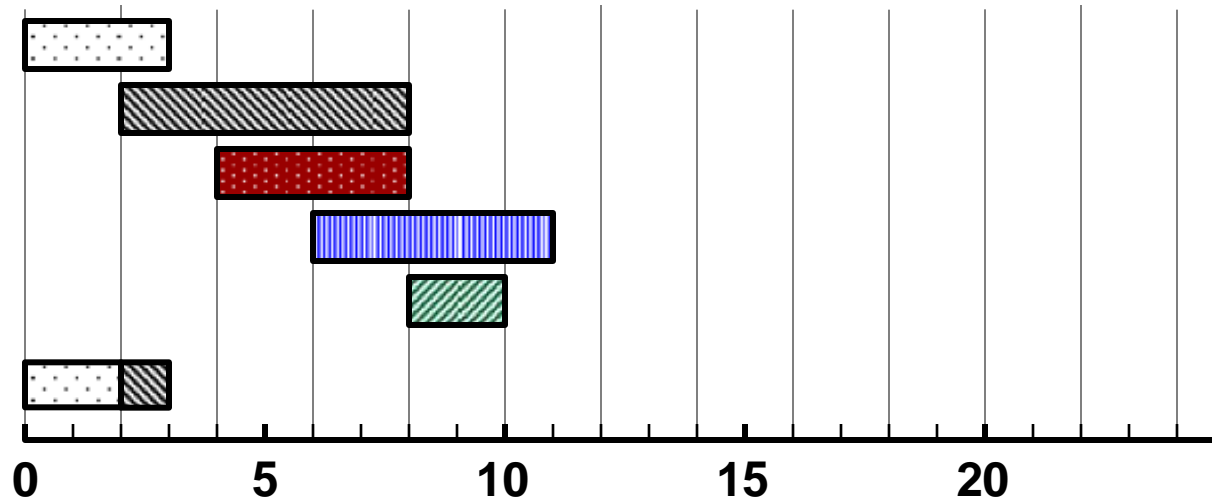


Ready List:

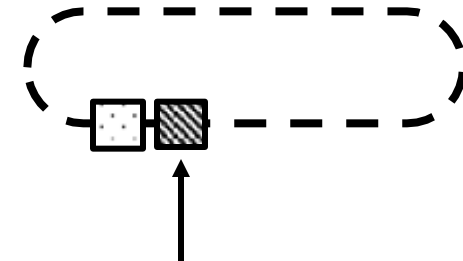


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

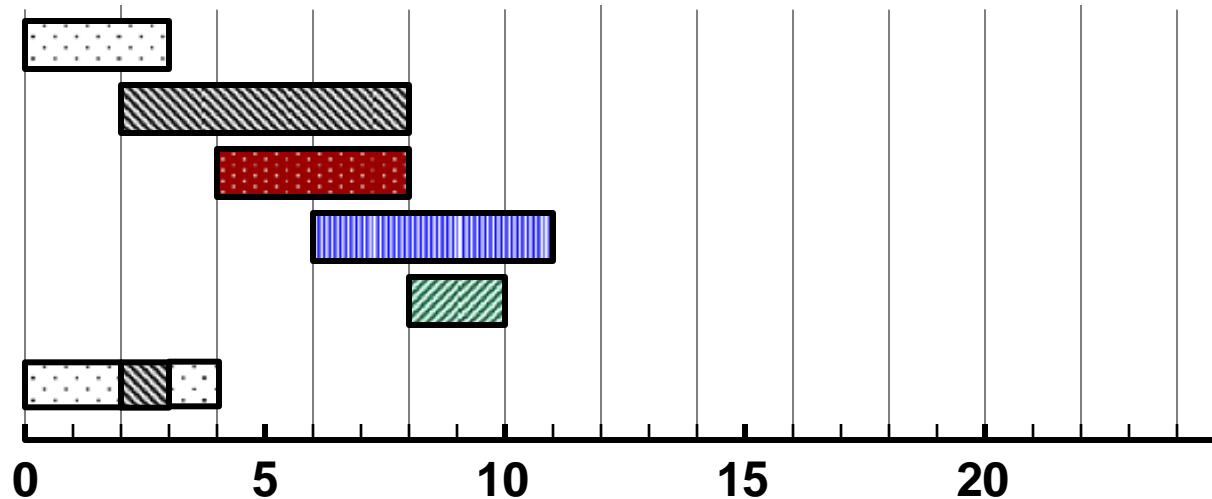


Ready List:

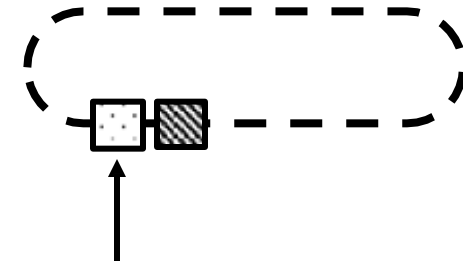


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

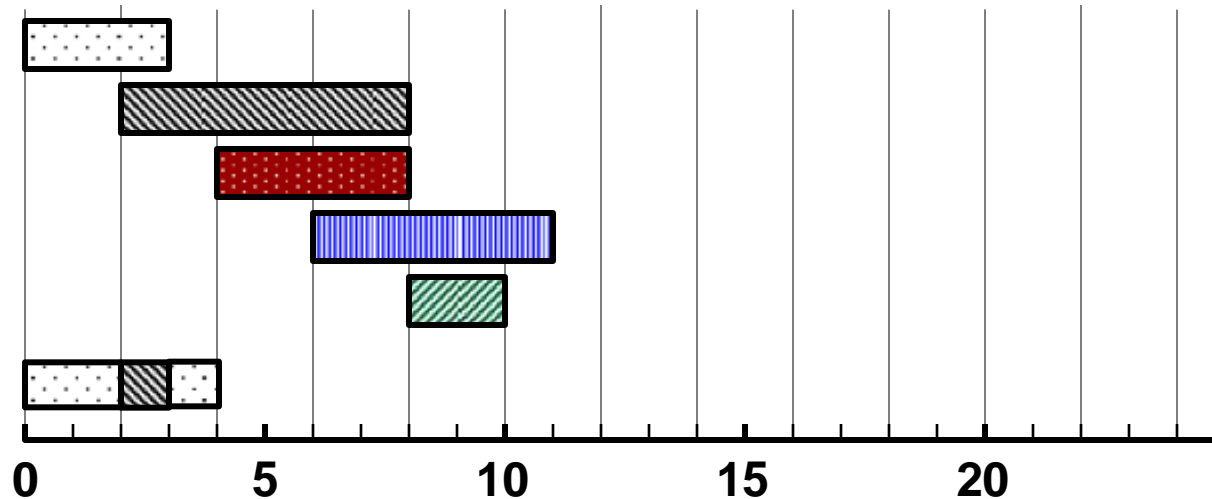


Ready List:

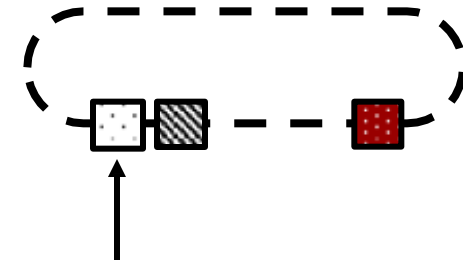


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

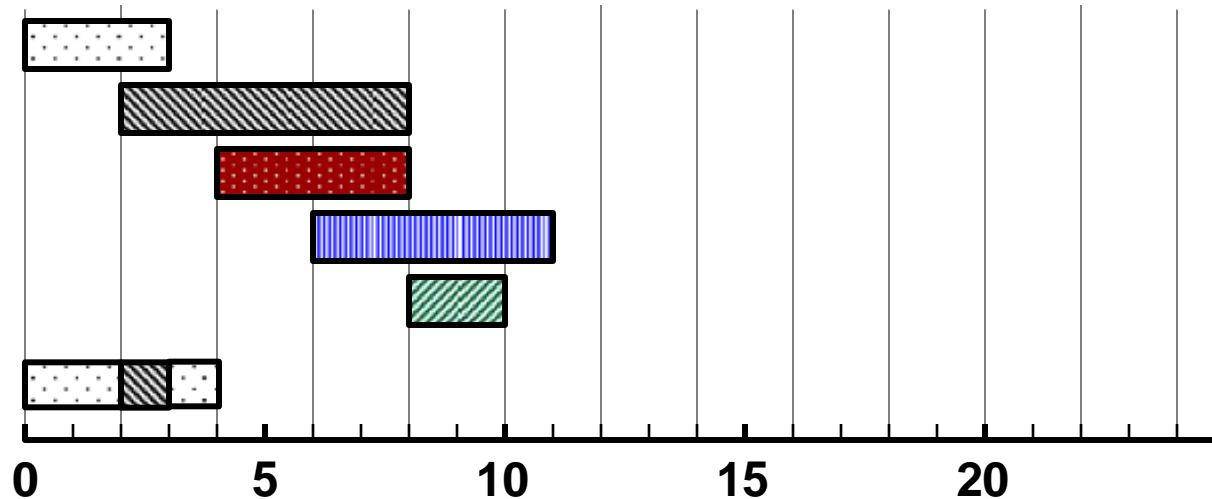


Ready List:

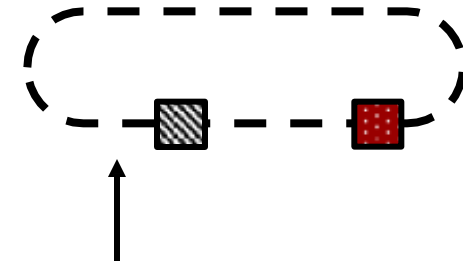


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

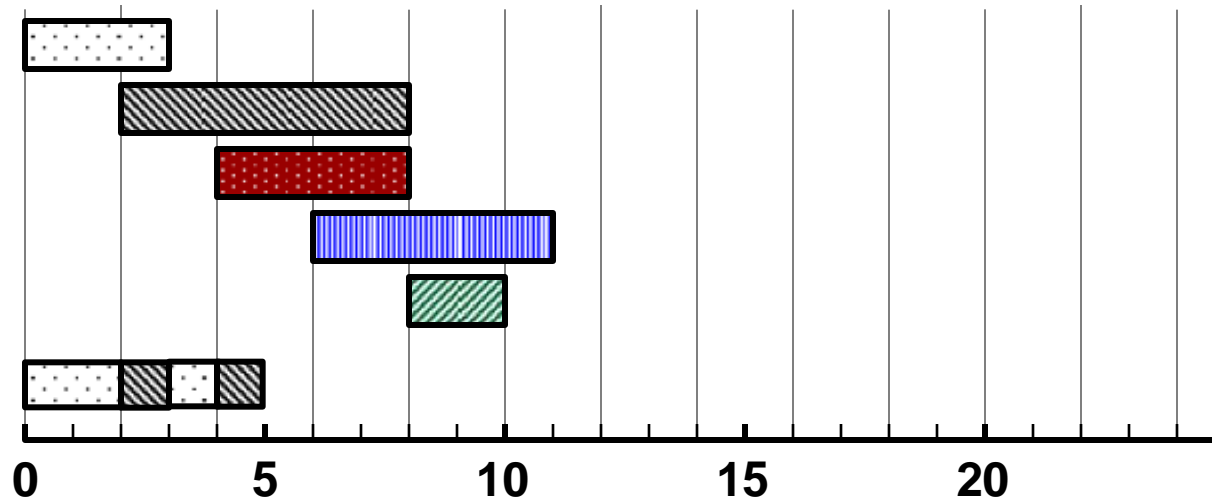


Ready List:

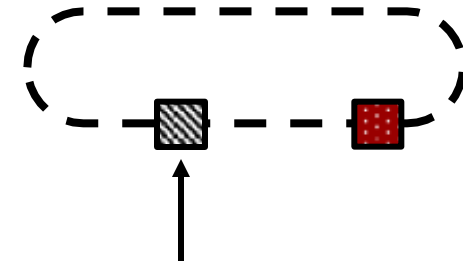


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

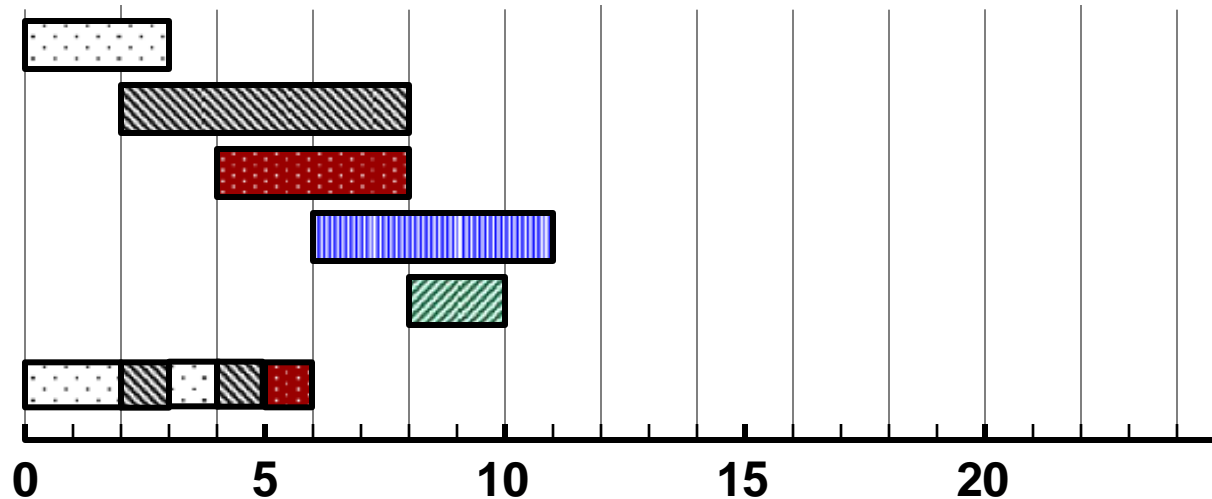


Ready List:

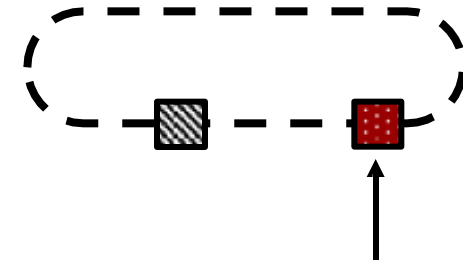


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

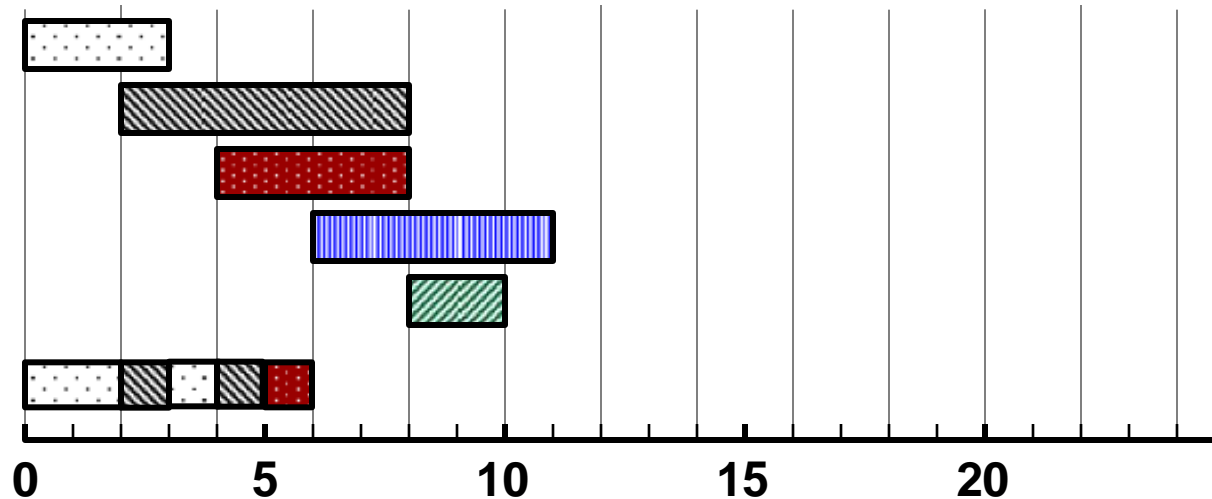


Ready List:

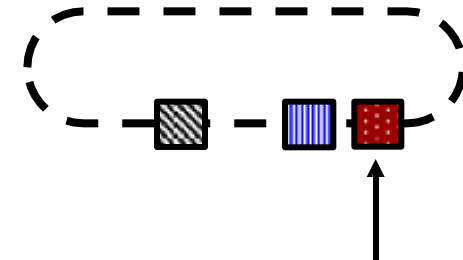


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

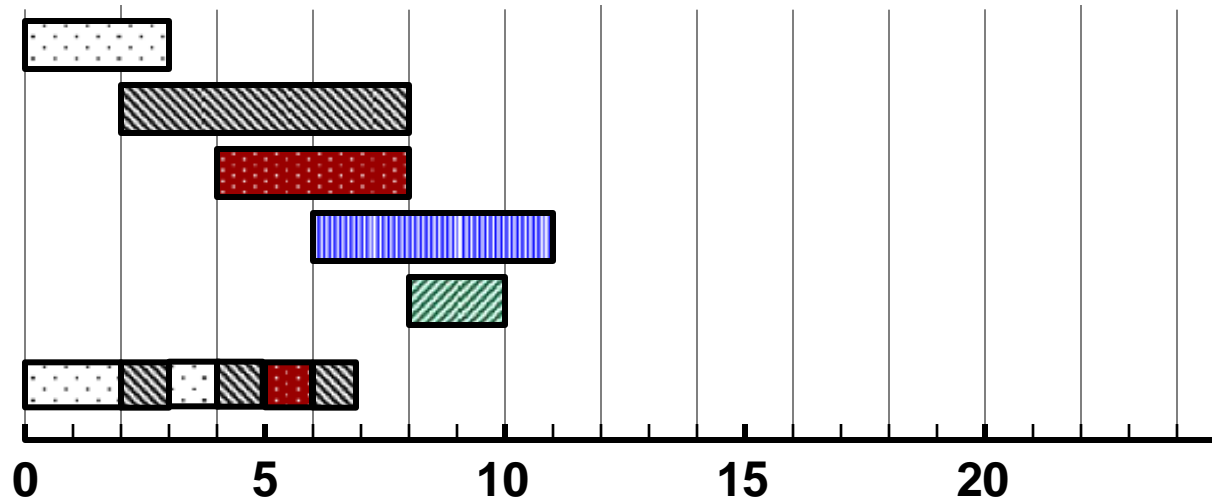


Ready List:

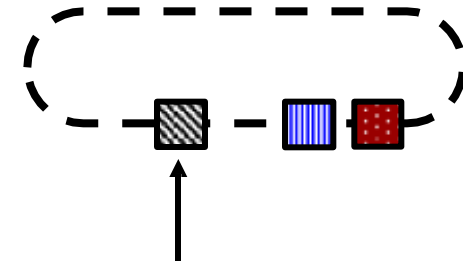


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

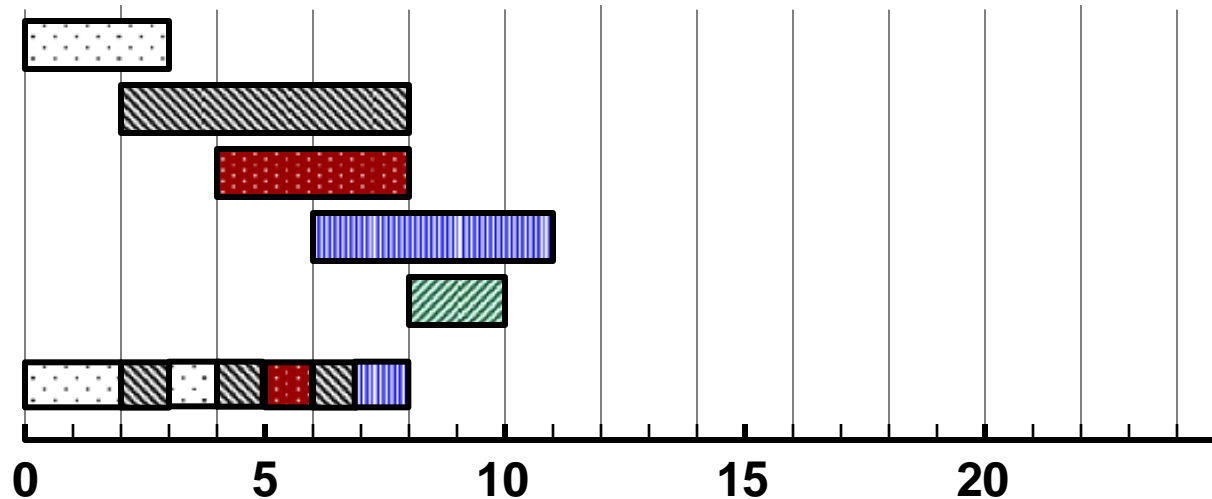


Ready List:

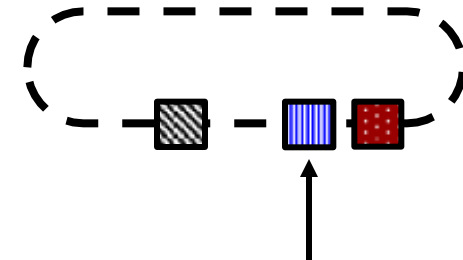


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

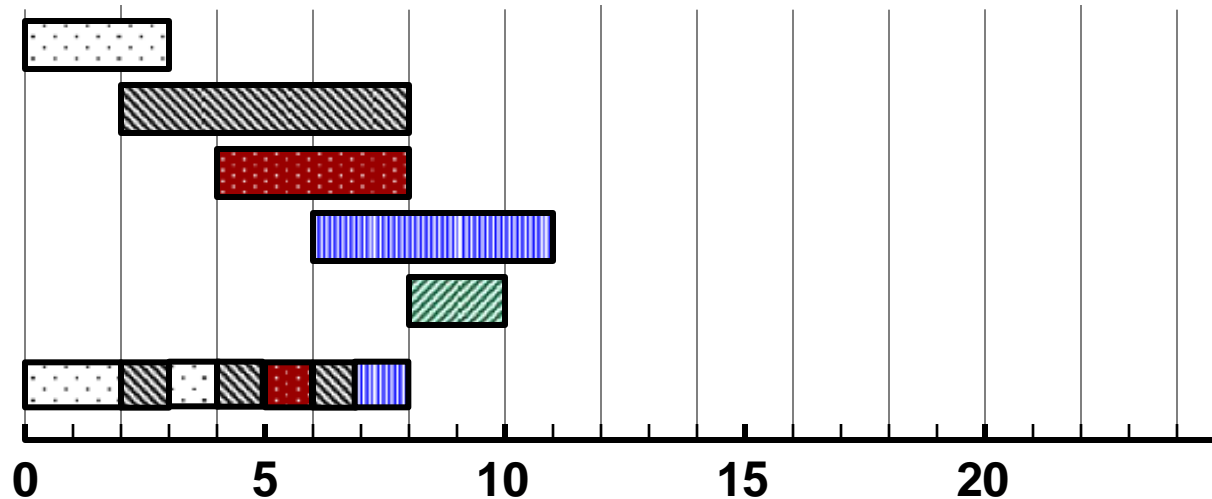


Ready List:

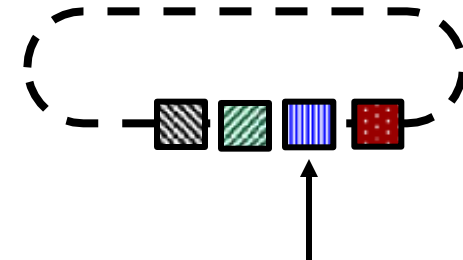


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

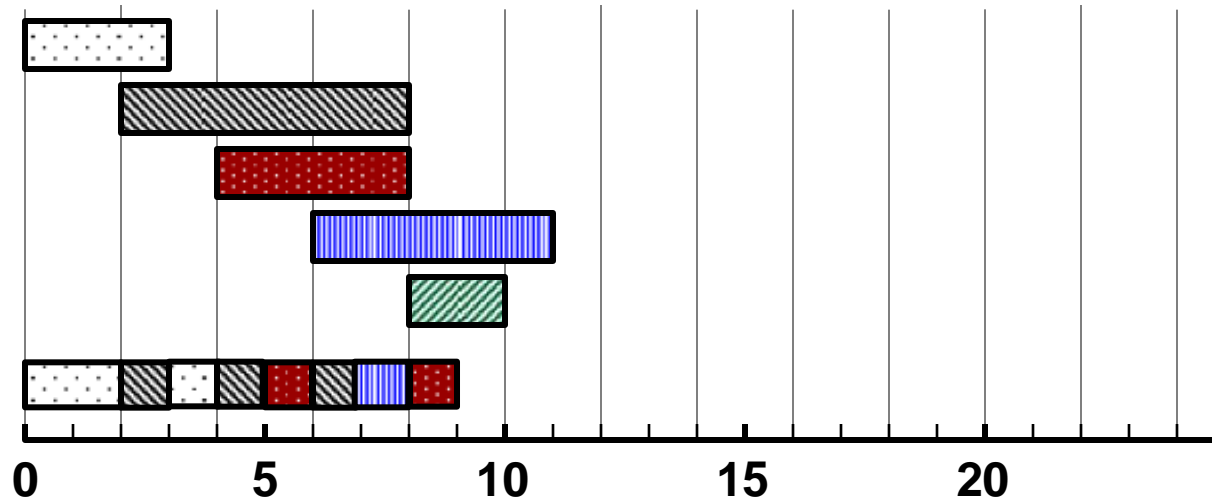


Ready List:

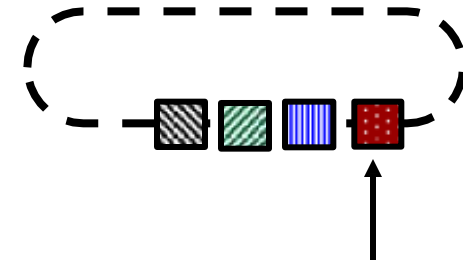


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

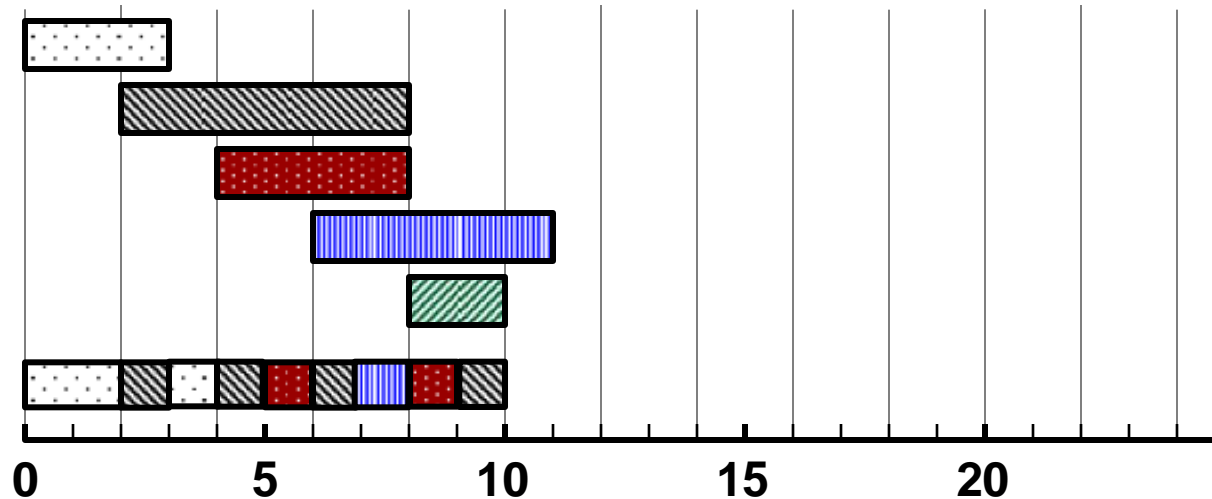


Ready List:

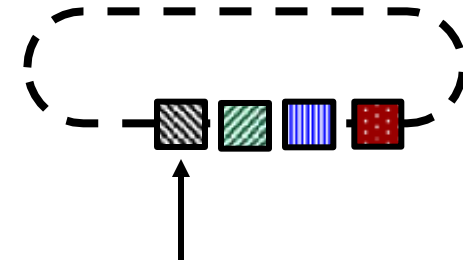


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

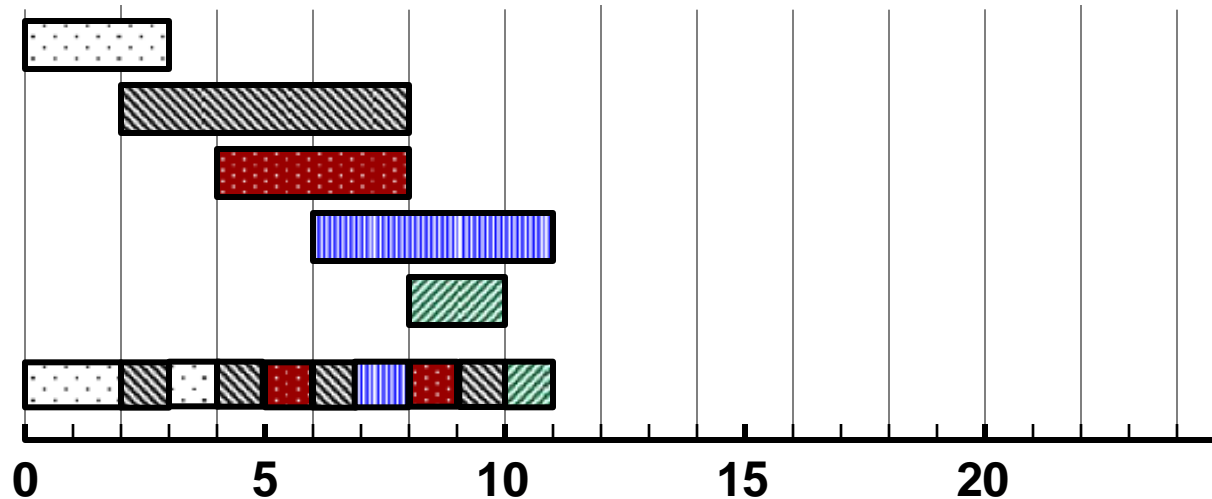


Ready List:

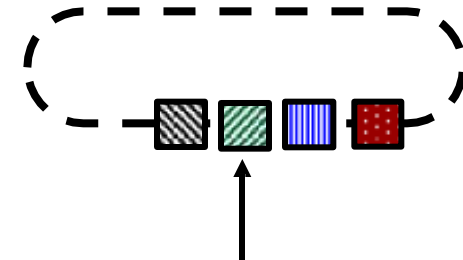


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

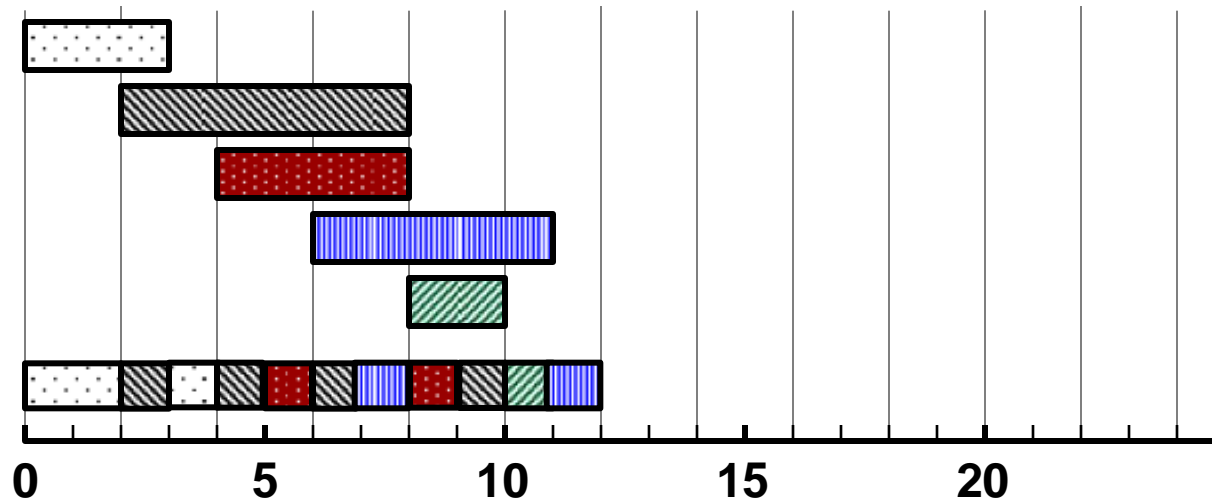


Ready List:

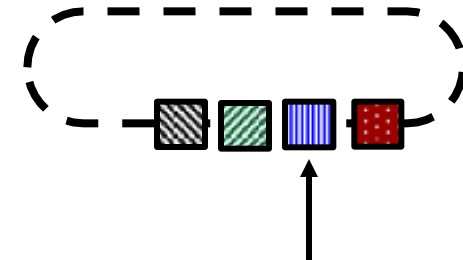


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

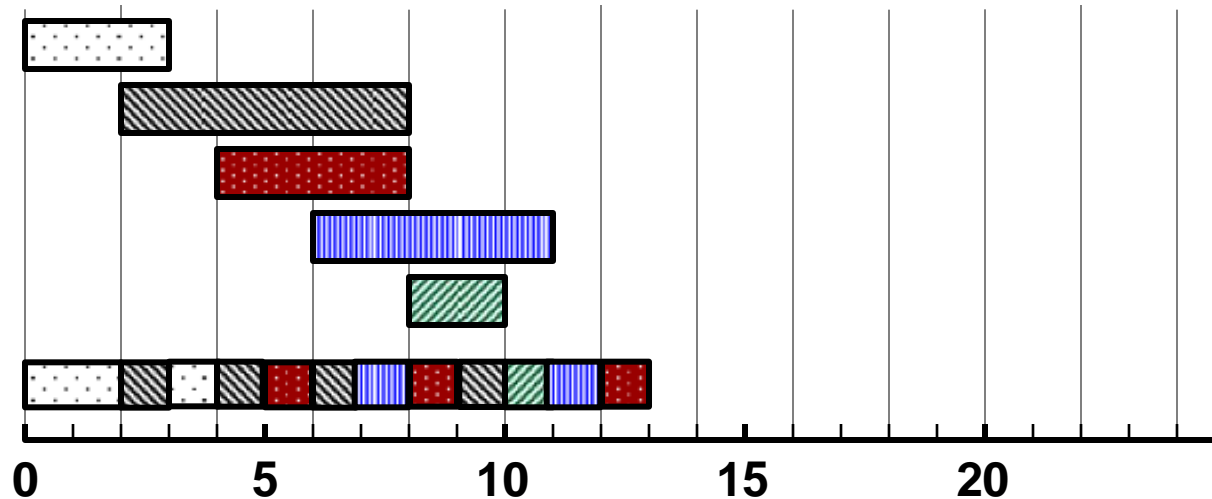


Ready List:

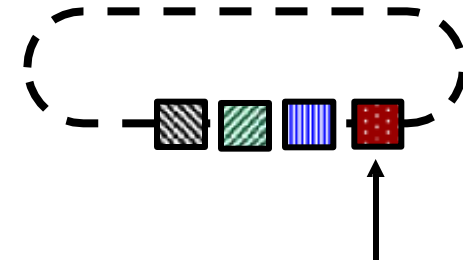


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

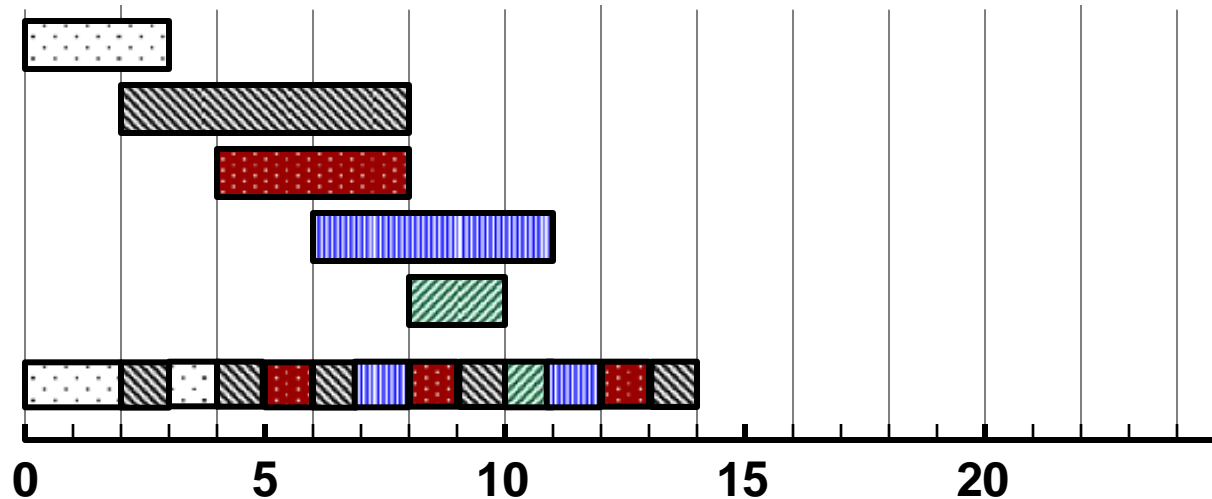


Ready List:

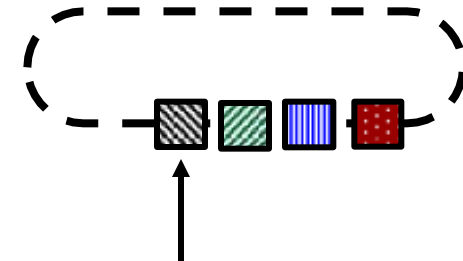


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

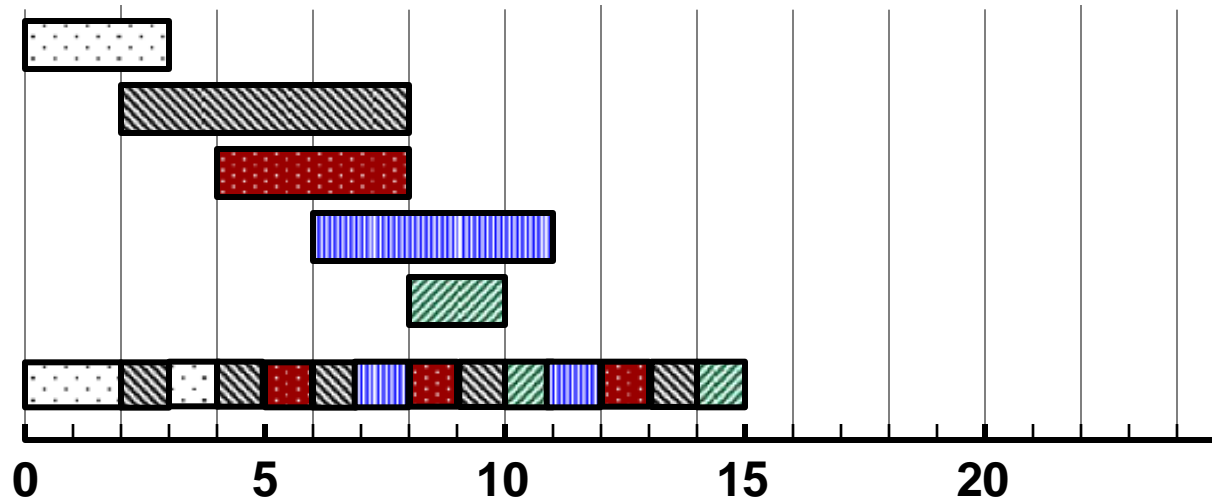


Ready List:

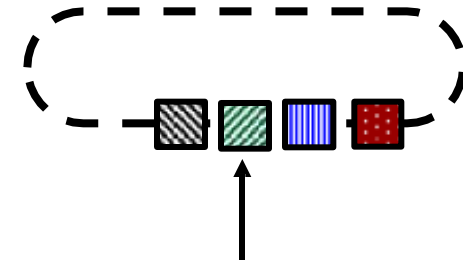


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

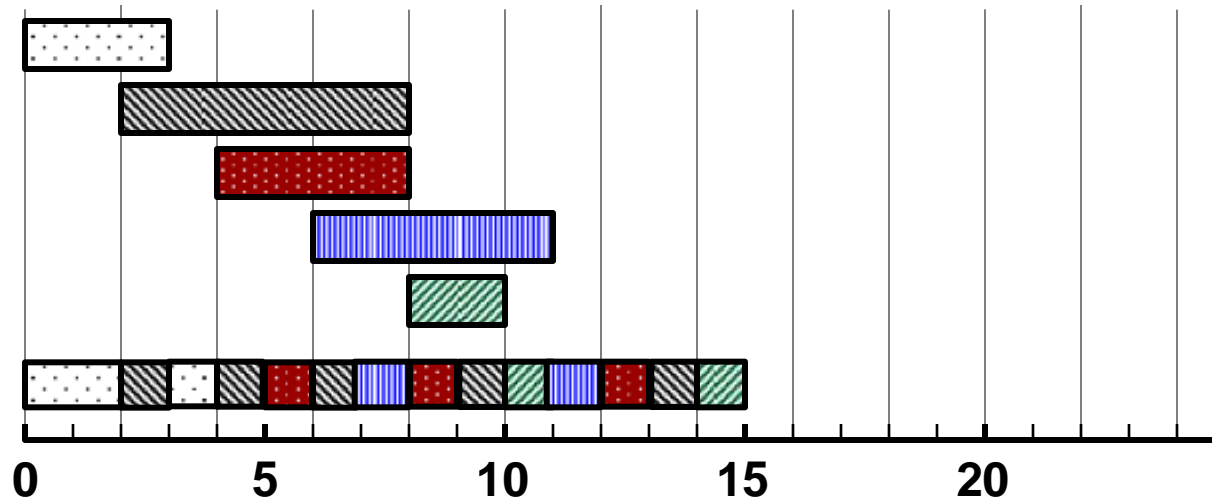


Ready List:

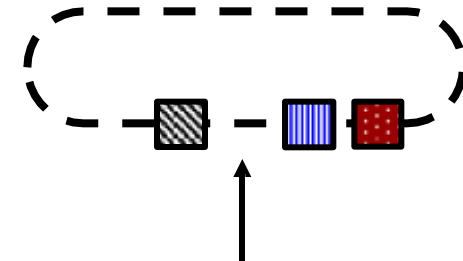


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

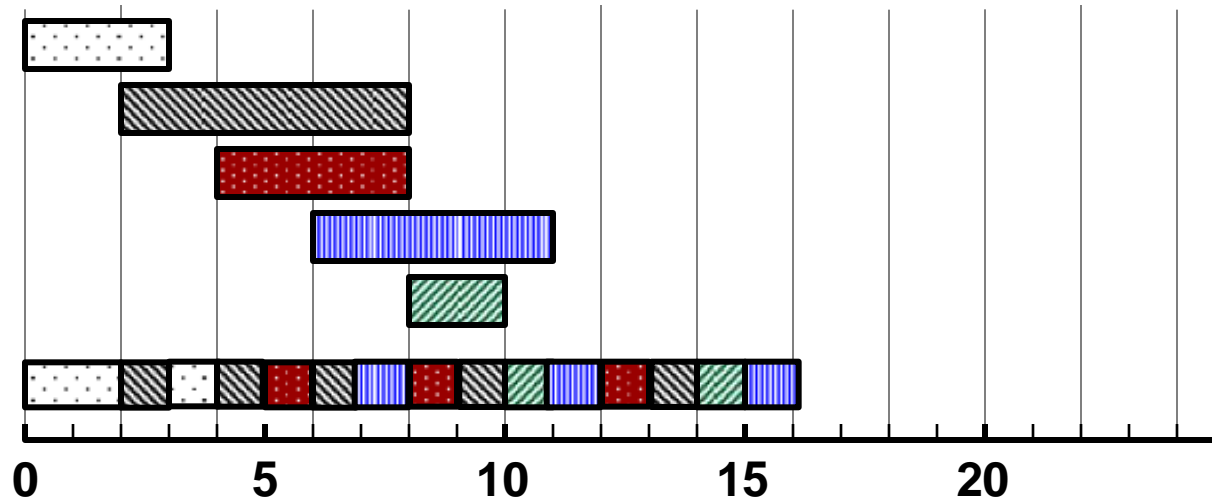


Ready List:

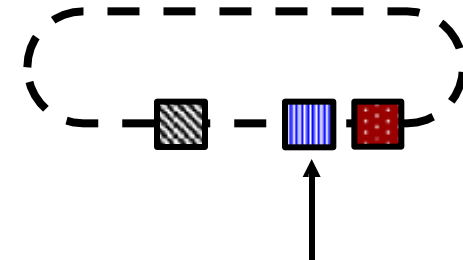


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

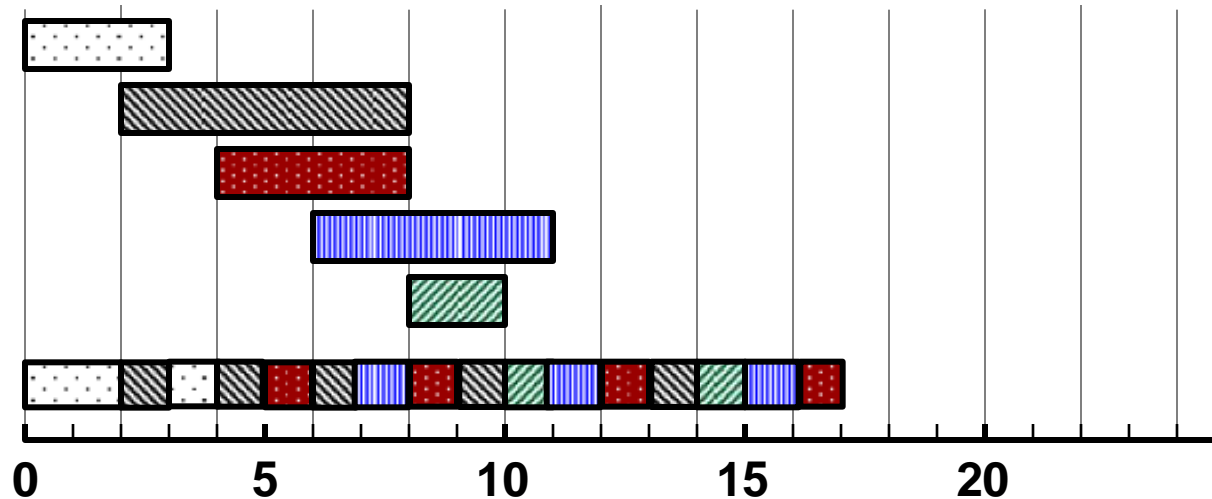


Ready List:

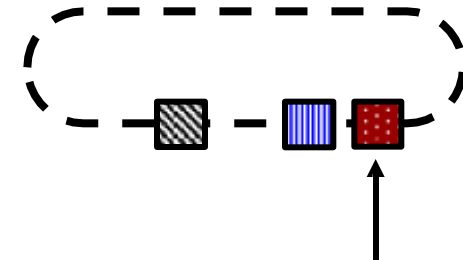


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

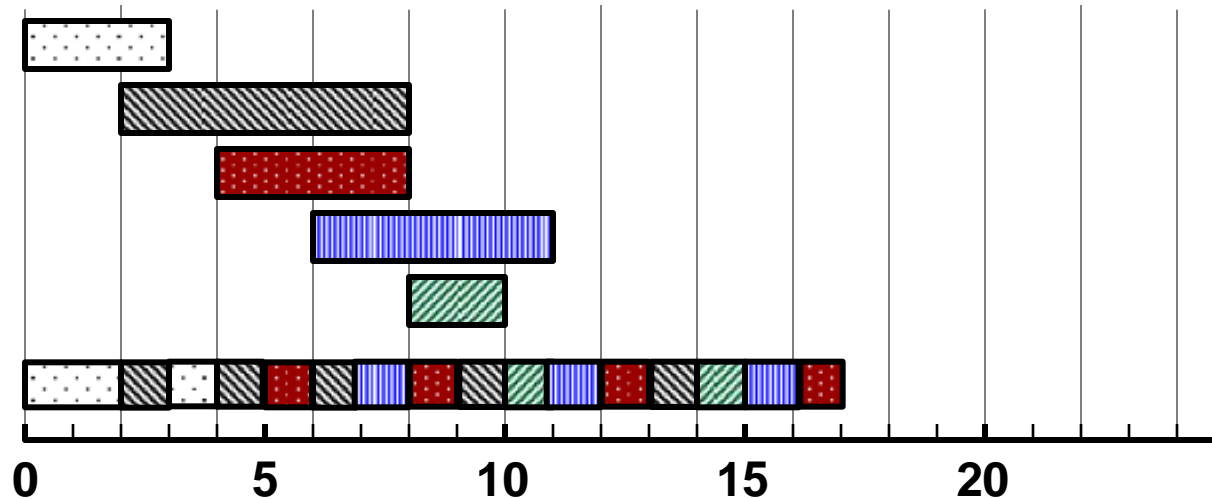


Ready List:

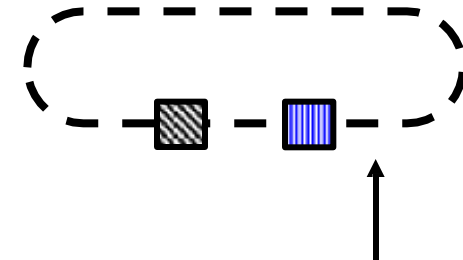


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

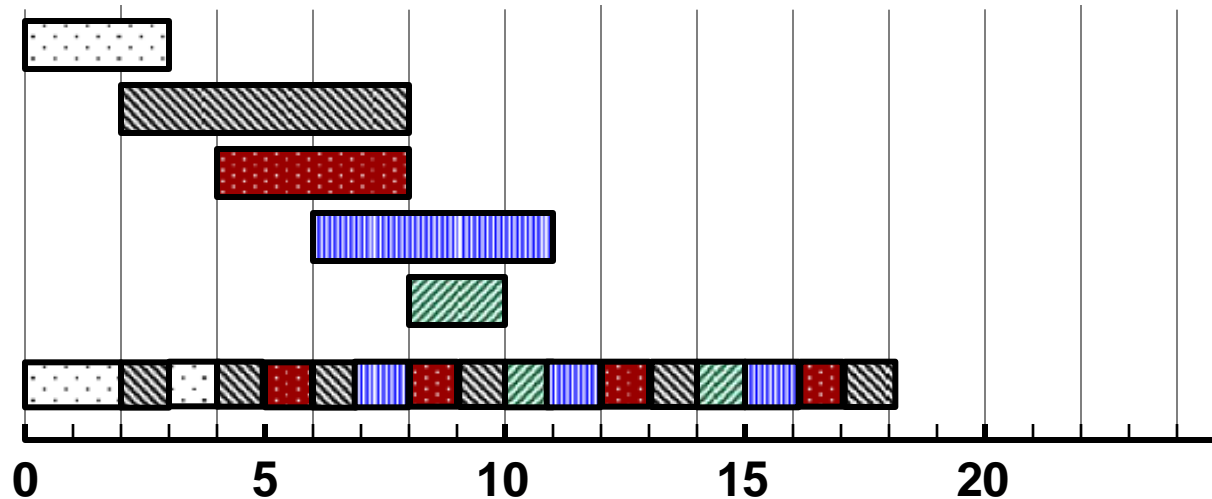


Ready List:

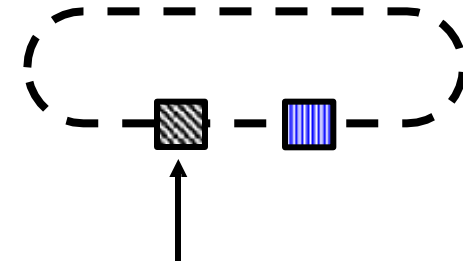


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

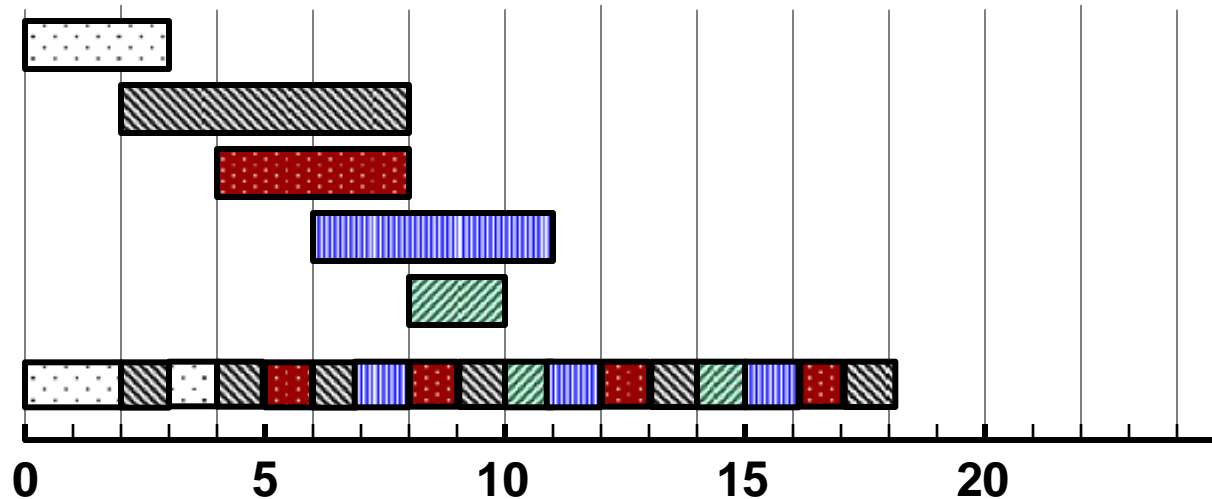


Ready List:

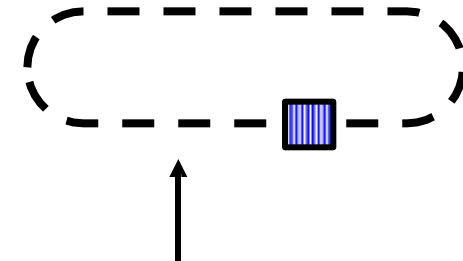


Round-Robin Scheduling






Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2

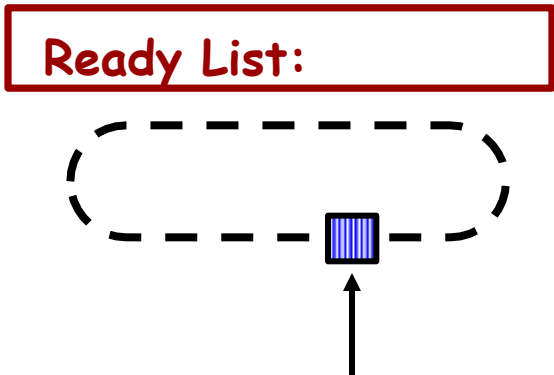
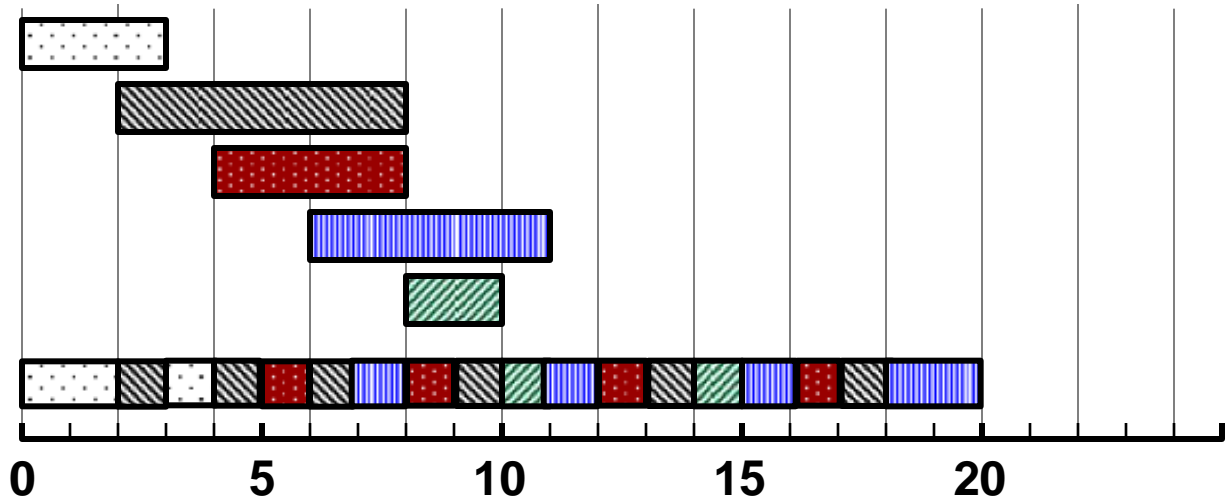


Ready List:








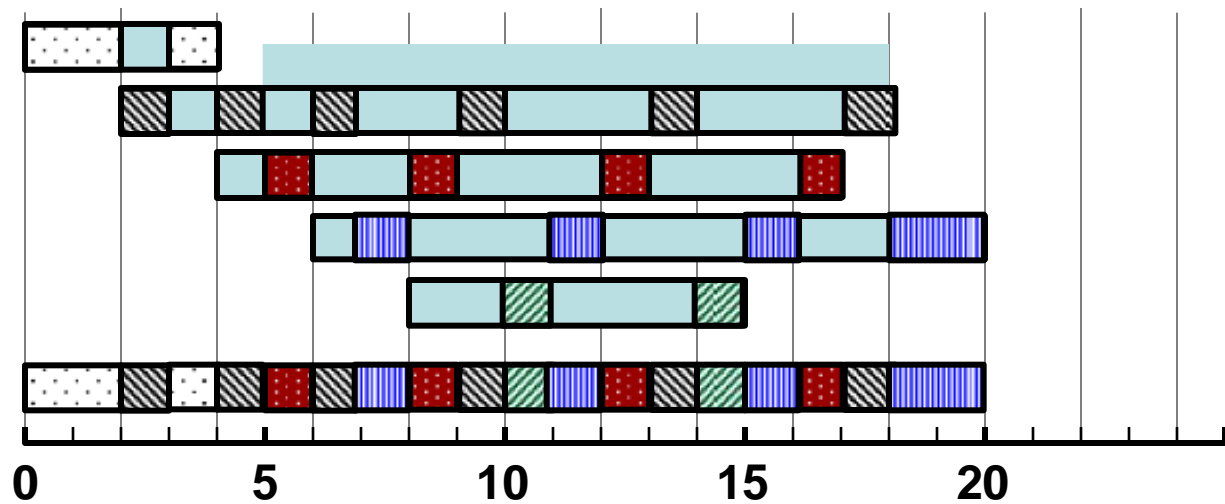
Round-Robin Scheduling

Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2








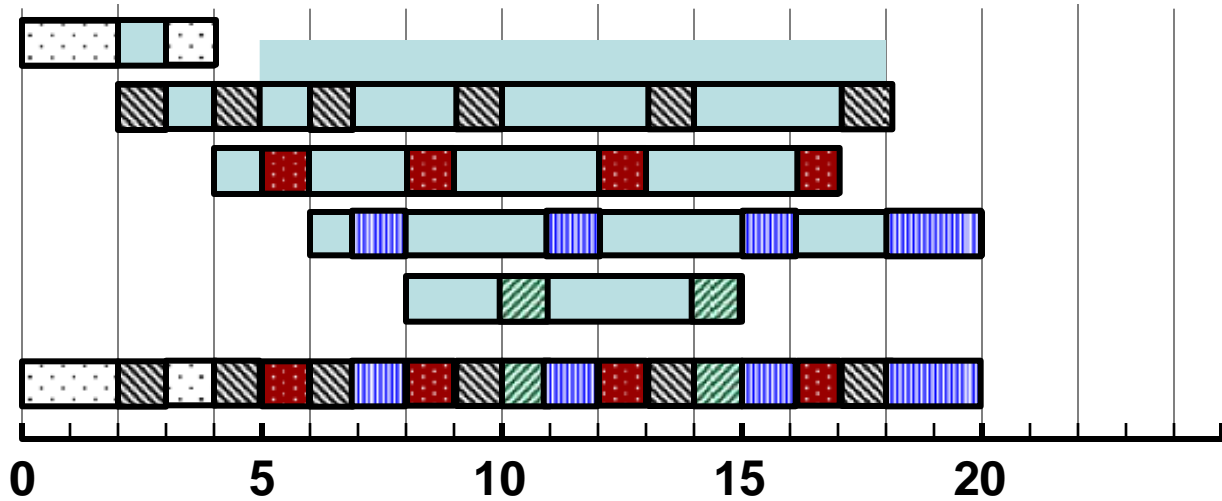
Round-Robin Scheduling

Process	Arrival Time	Processing Time
 1	0	3
 2	2	6
 3	4	4
 4	6	5
 5	8	2



Round-Robin Scheduling

		Arrival	Processing	Turnaround	
Process		Time	Time	Delay	Time
	1	0	3	1	4
	2	2	6	10	16
	3	4	4	9	13
	4	6	5	9	14
	5	8	2	5	7



Round-Robin Scheduling

- Effectiveness of round-robin depends on
 - The number of jobs, and
 - The size of the time quantum.
- Large # of jobs means that the time between scheduling of a single job increases
 - Slow responses
- Larger time quantum means that the time between the scheduling of a single job also increases
 - Slow responses
- Smaller time quantum means higher processing rates but
- also more overhead!

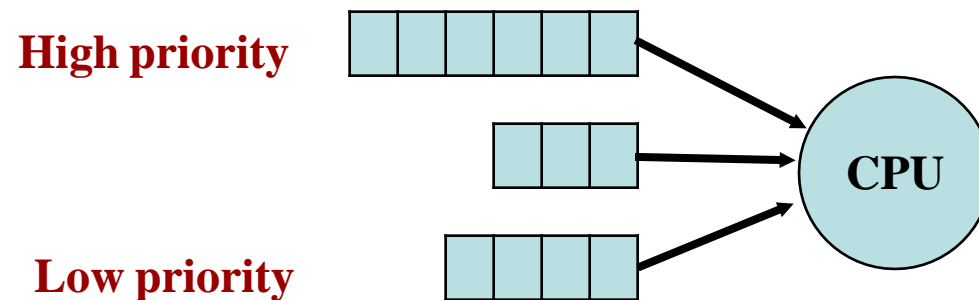
GENERAL PURPOSE SCHEDULERS

Priority Scheduling

- Assign a priority (number) to each process
Schedule processes based on their priority Higher priority processes get more CPU time
- Managing priorities
- Can use “nice” to reduce your priority
- Can periodically adjust a process’ priority
 - *Prevents starvation of a lower priority process (aging)*
 - *Can improve performance of I/O-bound processes by basing priority on fraction of last quantum used*

Multi-Level Queue Scheduling

- Multiple queues, each with its own priority
- Equivalently: each priority level has its own ready queue Within each queue...Round-robin scheduling
- Simplist Approach: A Process' s priority is fixed



Multi-Level Feedback Scheduling

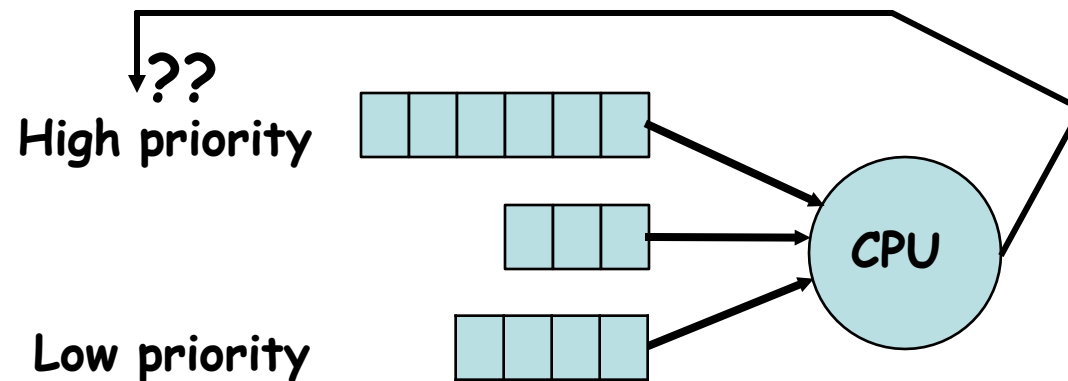
- Problem: Fixed priorities are too restrictive
 - Processes exhibit varying ratios of CPU to I/O times
- Dynamic Priorities
 - Priorities are altered over time, as process behavior changes!
- When do you change the priority of a process?

Multi-Level Feedback Scheduling

- **Solution:** Let the amount of CPU used be an indication of how a process is to be handled
 - Expired time quantum → more processing needed
 - Unexpired time quantum → less processing needed
- Adjusting quantum and frequency vs. adjusting priority?

Multi-Level Feedback Scheduling

- N priority levels, round-robin scheduling within a level
- *Quanta* increase as priority decreases
- Jobs are demoted to lower priorities if they do not complete within the current quantum



Multi-Level Feedback Scheduling

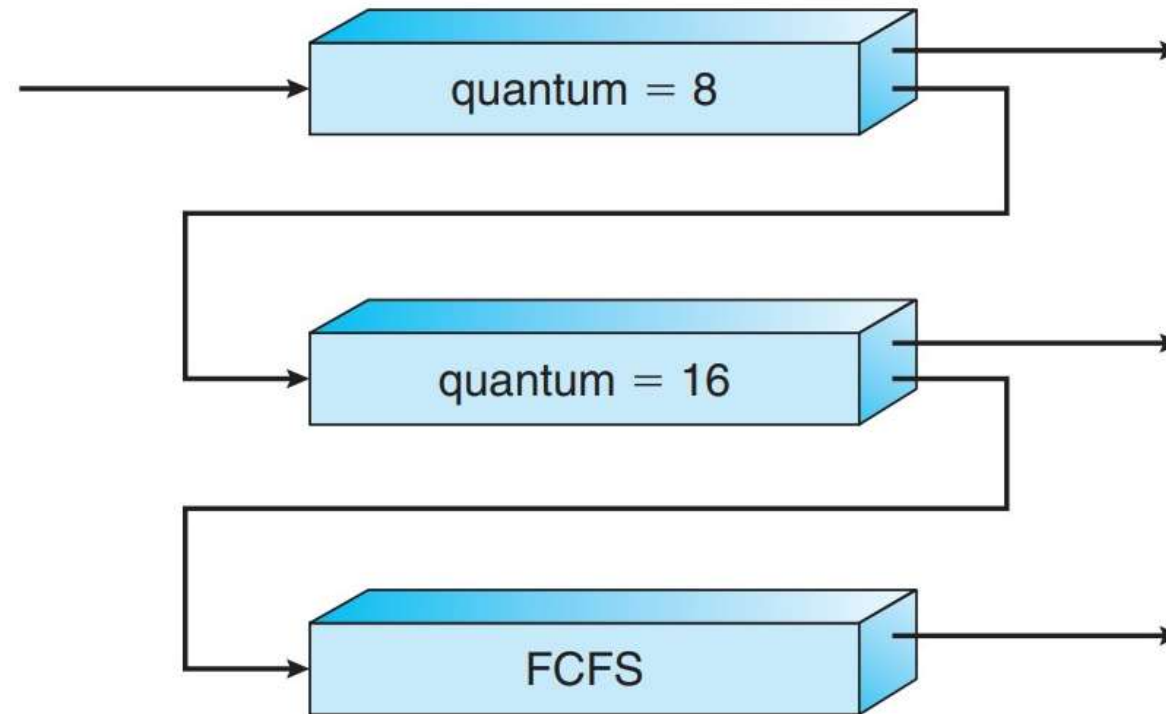


Figure 5.9 Multilevel feedback queues.

Lottery Scheduling

- A kind of proportional share scheduling Scheduler gives each thread some lottery tickets To select the next process to run...
 - The scheduler randomly selects a lottery number
 - The winning process gets to run

Example

Thread A gets 50 tickets

Thread B gets 15 tickets

Thread C gets 35 tickets

There are 100 tickets outstanding

Lottery Scheduling

- A kind of proportional share scheduling Scheduler gives each thread some lottery tickets. To select the next process to run...
 - The scheduler randomly selects a lottery number
 - The winning process gets to run

Example

Thread A gets 50 tickets → 50% of CPU

Thread B gets 15 tickets → 15% of CPU

Thread C gets 35 tickets → 35% of CPU

There are 100 tickets outstanding

A Brief Look at Real-Time Systems

- Typically real-time systems involve several steps that are not in traditional systems
- **Admission control**
 - All processes must ask for resources ahead of time
 - If sufficient resources exist, the job is “admitted” into the system.
- **Resource allocation**
 - Upon admission, the appropriate resources need to be reserved for the task.
- **Resource enforcement**
 - Carry out the resource allocations properly

A Brief Look at Real-Time Systems

- Assume processes are relatively periodic
 - Fixed amount of work per period (e.g. sensor
- systems or multimedia data)
 - Allows for specification of requirements

Real-Time Scheduling

- Two main types of schedulers...
- **Rate-Monotonic Schedulers**
 - Assign a *fixed, unchanging* priority to each process based on its requirements specification
 - No dynamic adjustment of priorities
- **Earliest-Deadline-First Schedulers**
 - Assign dynamic priorities based upon deadlines

Rate Monotonic Schedulers

For preemptable, periodic processes (tasks)

Assigns a fixed priority to each task

T = The period of the task

C = The amount of processing per task period

Process P1



$T = 1$ second

$C = 1/2$ second / period

In RMS scheduling, the question to answer is...

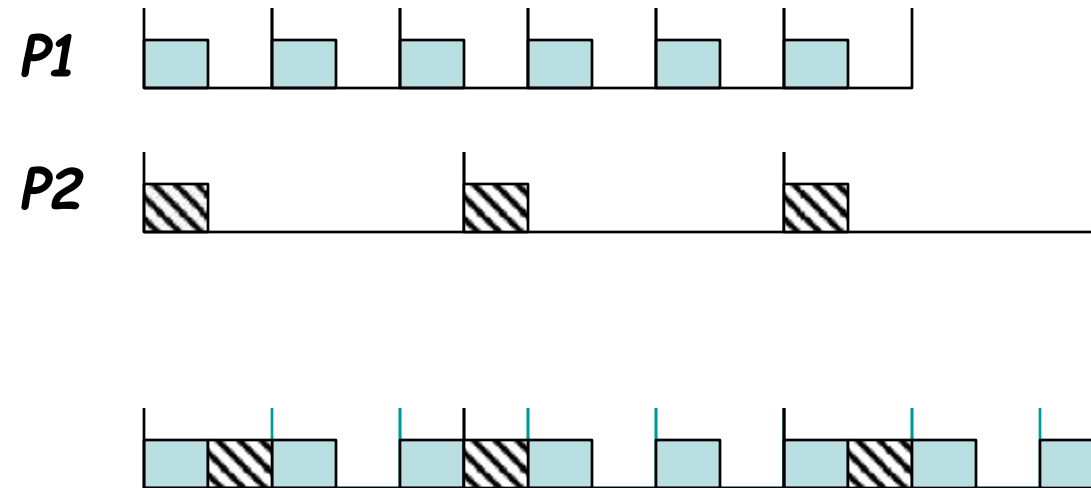
- *What priority should be assigned to a given task?*

Rate Monotonic Schedulers

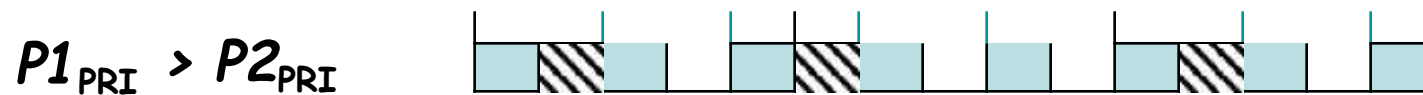
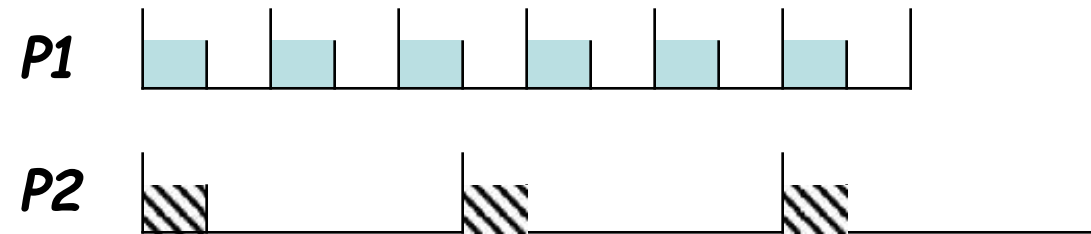
■ Assumptions:

- Processes complete (yield) within their period
- Independent processes
- Same CPU requirements per burst
- Other non-periodic processes have no deadlines
- Instantaneous preemption with no overhead

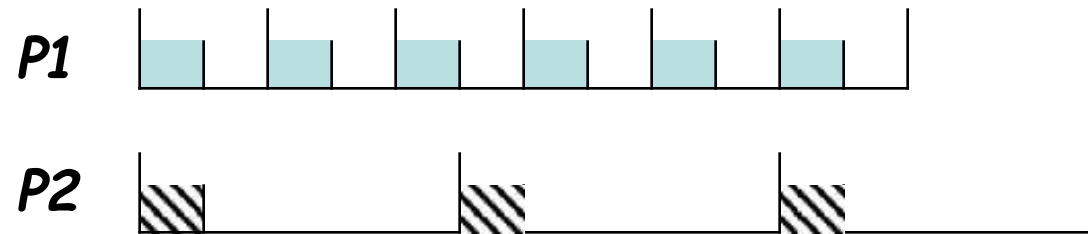
Rate Monotonic Schedulers



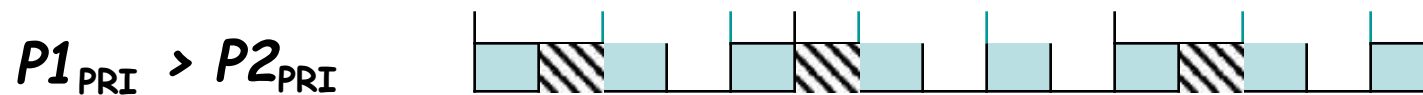
Rate Monotonic Schedulers



Rate Monotonic Schedulers



Which is best?



Earliest Deadline First

*When processes do not have periodic execution
or constant CPU requirements...*

When processes have deadline specifications...

Unlike RMS, EDF uses dynamic priorities (based upon earliest deadline first)

(+) 100% processor utilization

(-) Need to keep track of deadlines

Admission Control

Just check to see if 100% processor utilization.

Sum the C_i/T_i 's and see if less than or equal to 1

What about overhead?