CS 3502
Operating Systems

Scheduling

Kun Suo

Computer Science, Kennesaw State University

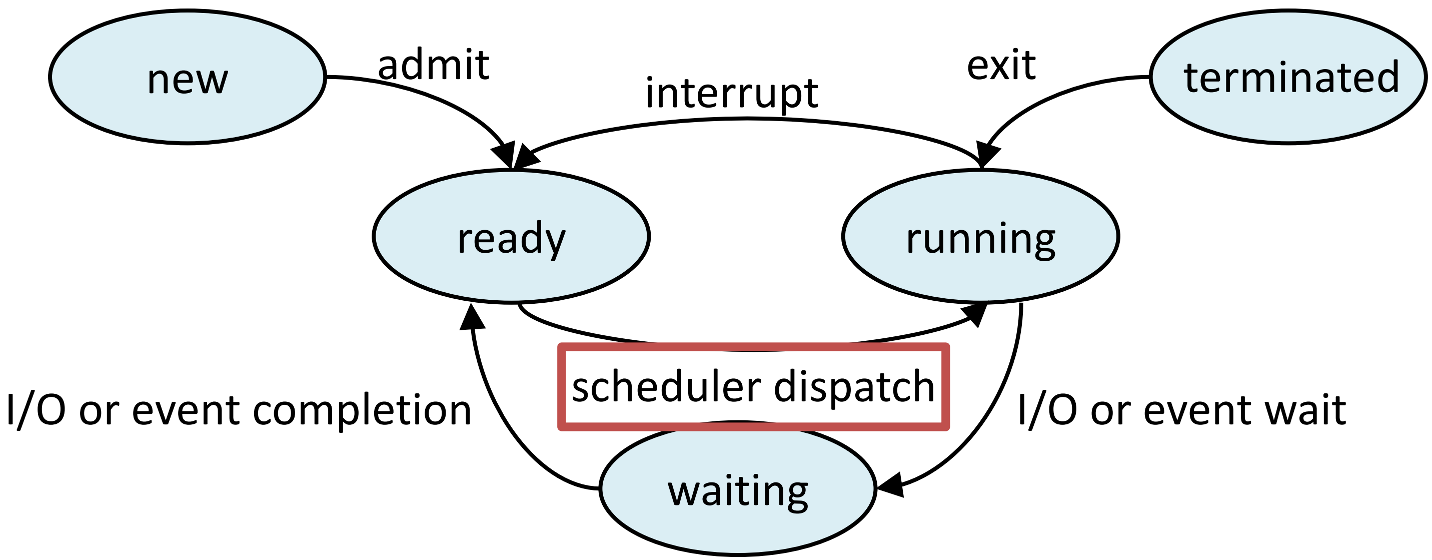
https://kevinsuo.github.io/

Outline

- Introduction to CPU scheduling
  - What is CPU scheduling
  - Why we need CPU scheduling
  - When scheduling happens
- Scheduling policies
  - FCFS, SJF, RR, Priority
  - Scheduling on multiple CPUs
  - Scheduling in Linux

What is CPU scheduling?

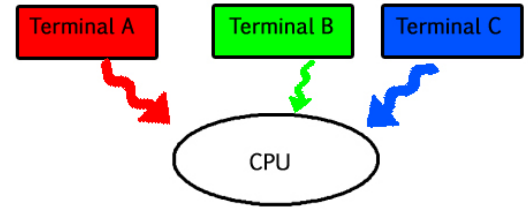
- The five-state process model



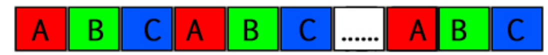
CPU scheduling

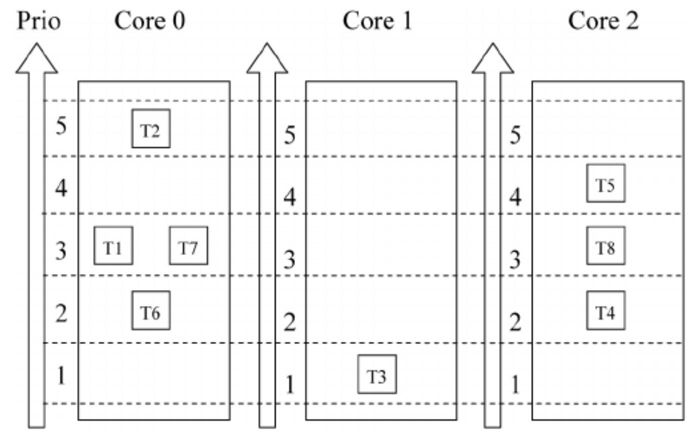
Selects from among the processes/threads that are ready to execute, and allocates the CPU to it

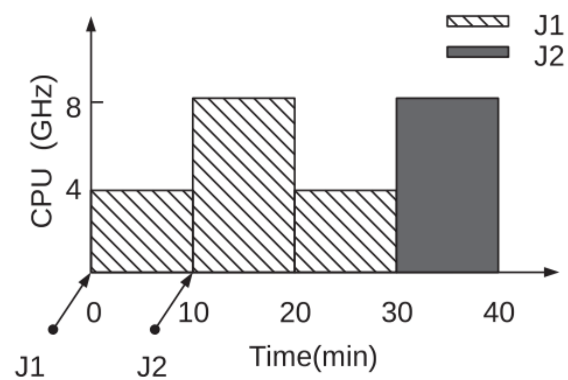
Why CPU scheduling?



- In support of multiprogramming
  - uniprocessor systems
    - Time-sharing processor
  - multiprocessor systems
    - Efficiently distributing tasks
  - Real-time systems
    - Reliably guaranteeing deadlines







Why CPU scheduling?

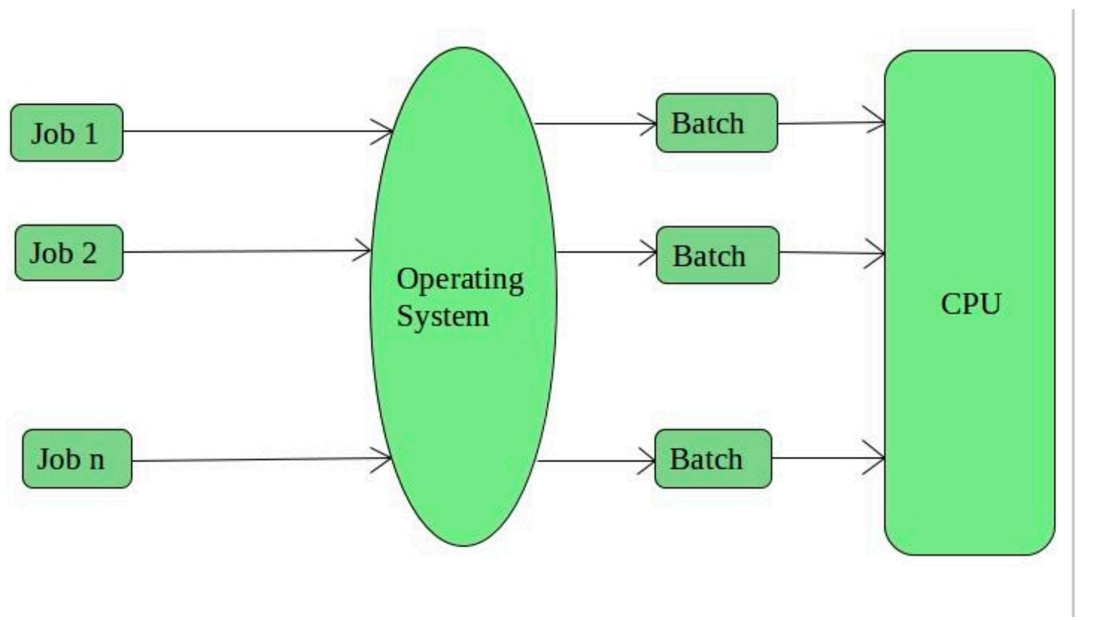
- It is (maybe) the most important part in a OS
  - Why some OS seems to be faster than others?
  - Why I do not see performance improvement when upgrading to a 16-core computer?

Why CPU scheduling? – Different Goals

- All systems
  - Fairness - giving each process a fair share of the CPU
  - Policy enforcement - seeing that stated policy is carried out
  - Balance - keeping all parts of the system busy

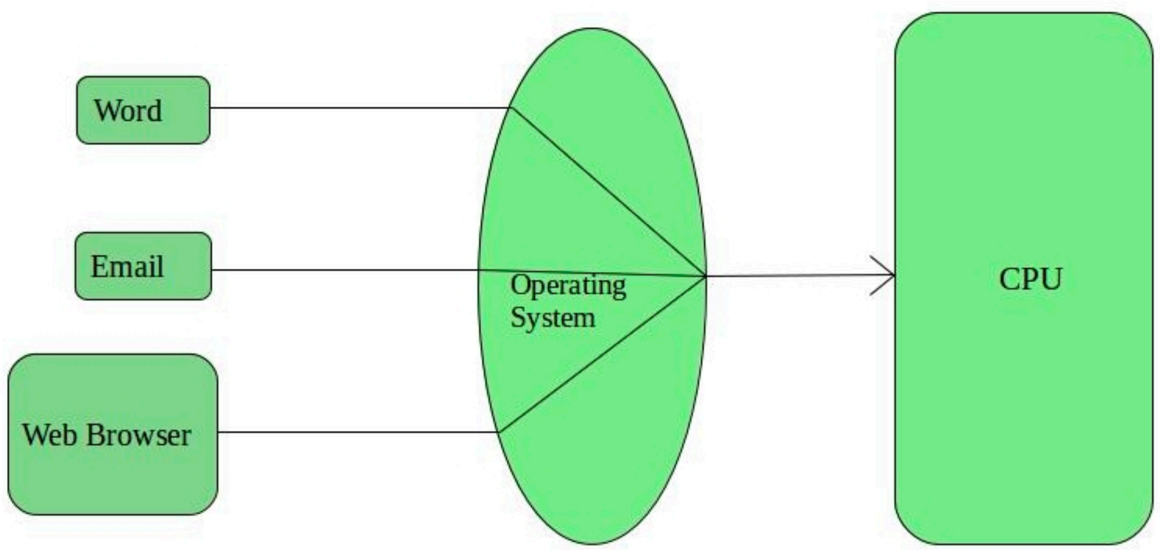
Why CPU scheduling? – Different Goals

- Batch systems
  - Throughput - maximize jobs per hour
  - Turnaround time - minimize time between submission and termination
  - CPU utilization - keep the CPU busy all the time



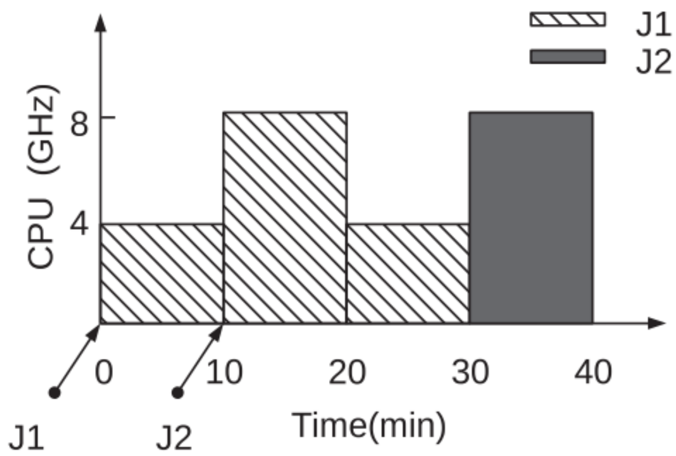
Why CPU scheduling? – Different Goals

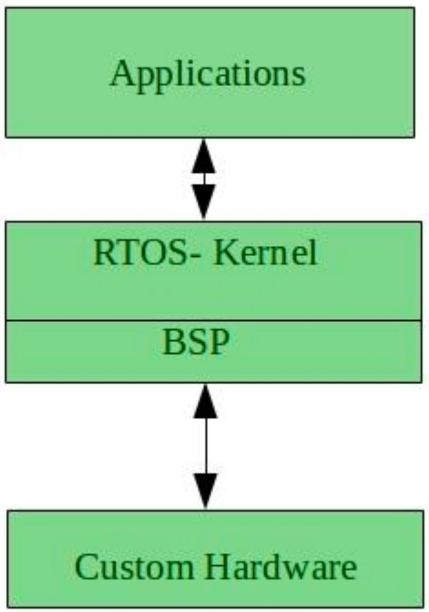
- Interactive systems
  - Response time - respond to requests quickly
  - Proportionality - meet users' expectations



Why CPU scheduling? – Different Goals

- Real-time systems
  - Meeting deadlines - avoid losing data
  - Predictability - avoid quality degradation in multimedia systems





Why CPU scheduling? – Different Goals

- All systems
  - Fairness - giving each process a fair share of the CPU
  - Policy enforcement - seeing that stated policy is carried out
  - Balance - keeping all parts of the system busy
- Batch systems
  - Throughput - maximize jobs per hour
  - Turnaround time - minimize time between submission and termination
  - CPU utilization - keep the CPU busy all the time
- Interactive systems
  - Response time - respond to requests quickly
  - Proportionality - meet users' expectations
- Real-time systems
  - Meeting deadlines - avoid losing data
  - Predictability - avoid quality degradation in multimedia systems

Scheduling Goals: A Different Point of View

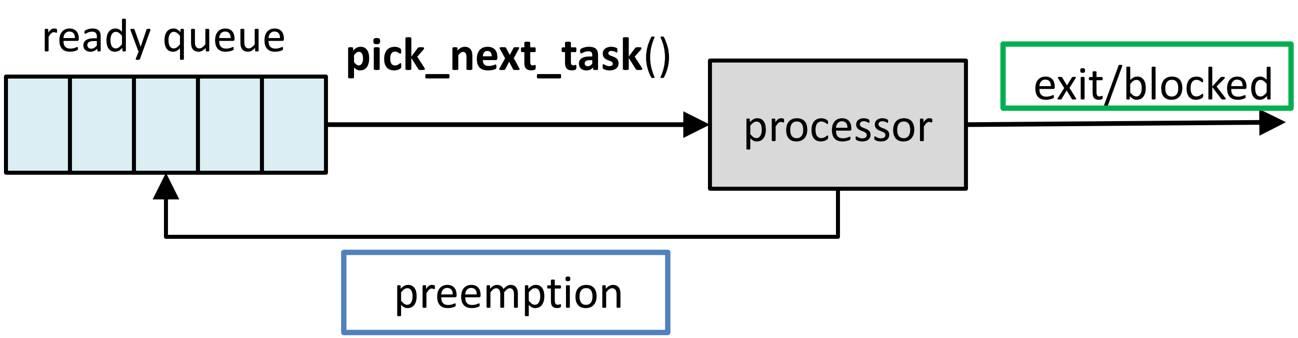
- User oriented → minimize
  - Response time (wait time): the time that the first response is received (interactivity)
  - Turnaround time: the time that the task finishes
  - Predictability: variations in different runs
- System oriented → maximize
  - Throughput: # of tasks that finish per time unit
  - Utilization: the percentage of time the CPU is busy
  - Fairness: avoid starvation

When scheduling happens?

- CPU scheduling may take place at
  - Clock interrupts
  - I/O interrupts
  - I/O completion
  - Termination

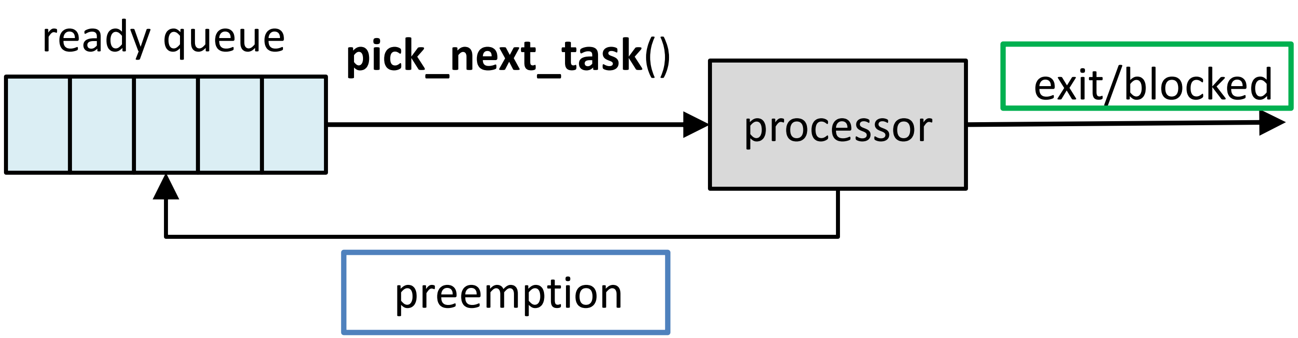
preemptive

non-preemptive

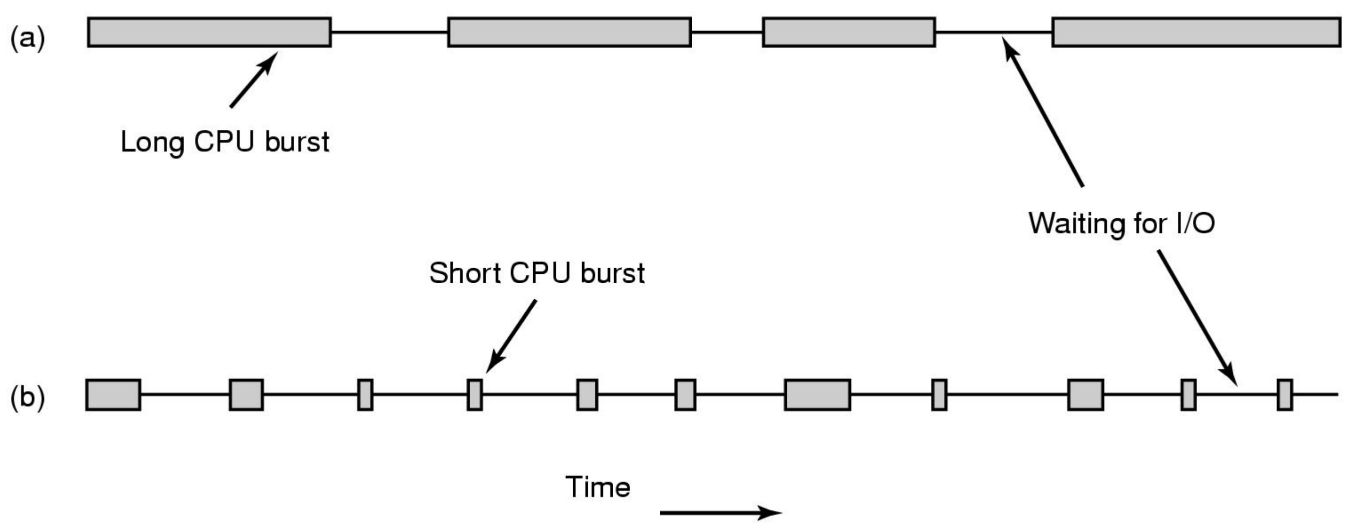


When scheduling happens?

- Non-preemptive
  - Scheduling only when current process terminates or gives up control
- Preemptive
  - Processes can be forced to give up control



CPU-bound tasks vs. I/O-bound tasks




```c
#include <stdio.h>

int main () {
    int a = 1;

    while ( a < 1000000 )
    {
        a++;
    }

    return 0;
}
```

a CPU-bound/CPU-intensive process

```c
#include <stdio.h>

int main()
{
    char name[20];
    printf("Enter name: ");

    scanf("%s", name);
    printf("Your name is %s.", name);

    return 0;
}
```

a I/O-bound/I/O-intensive process

I/O is when a process enters the blocked state waiting for an external device to complete its work

Outline

- Introduction to CPU scheduling
  - What is CPU scheduling
  - Why we need CPU scheduling
  - When scheduling happens
- Scheduling policies
  - FCFS, SJF, RR, Priority
  - Scheduling on multiple CPUs
  - Scheduling in Linux

Scheduling Policies

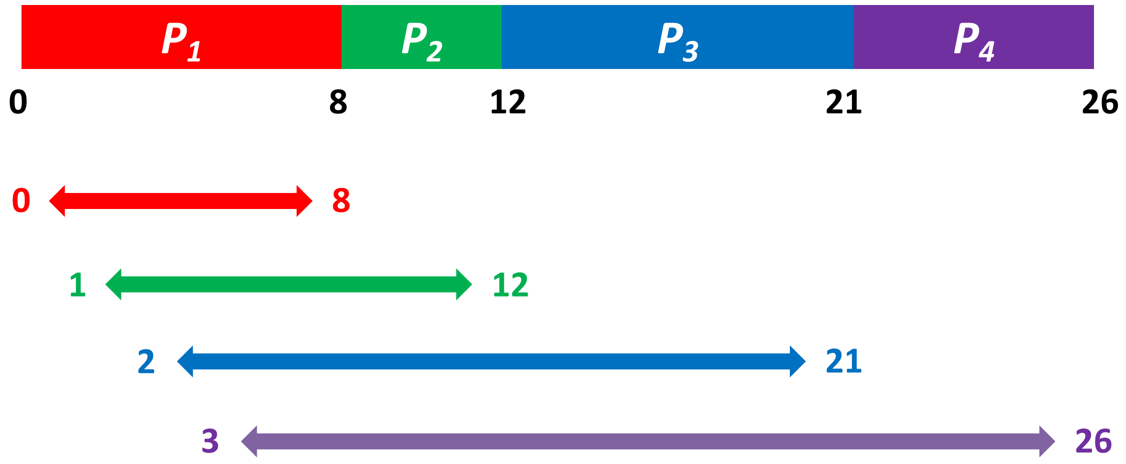
- Batch Systems
  - First-Come First-Serve (FCFS)
  - Shortest Job First
  - Shortest Remaining Time Next
- Interactive Systems
  - Round-Robin
  - Priority Scheduling
  - Multiple Queues
  - Shortest Process Next
  - Guaranteed Scheduling
  - Lottery Scheduling
- Real-time Systems
  - Rate Monotonic Scheduling
  - Earliest Deadline First Scheduling

Not exist best scheduling.
It depends on your goals.

Determine the next
ready task to run

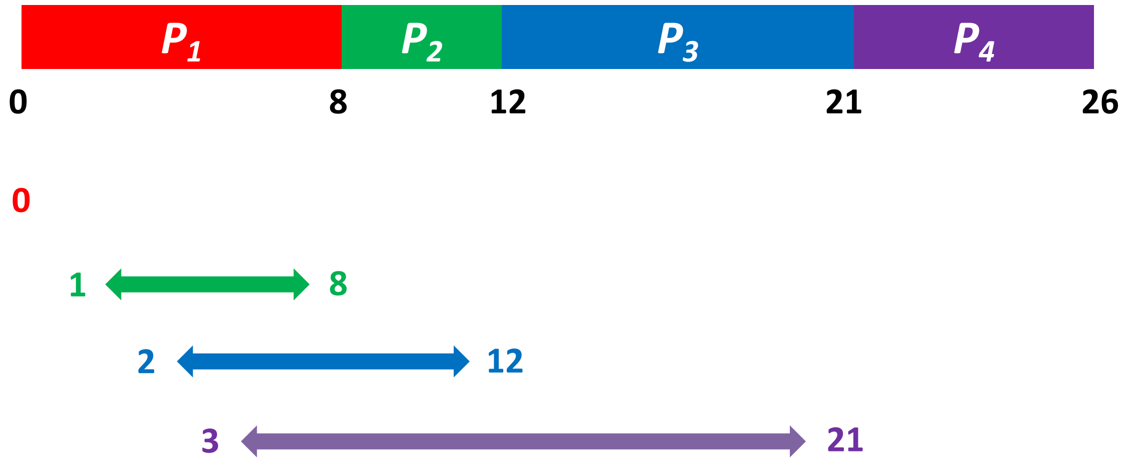
Turnaround time = End time – Arrival time

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



Response time = Start time – Arrival time

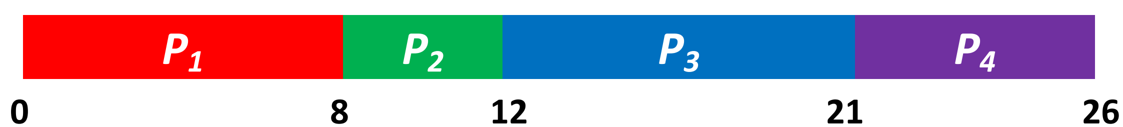
Process	Arrival Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5



First-Come, First-Serve (FCFS)

- CPU schedules the task that arrived earliest, non-preemptive

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

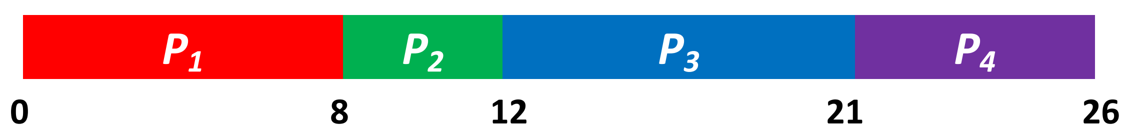


Average turnaround time = ((8-0)+(12-1)+(21-2)+(26-3)) / 4 = 15.25

First-Come, First-Serve (FCFS)

- CPU schedules the task that arrived earliest, non-preemptive

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



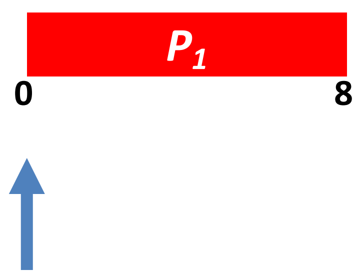
Average response time = (0+(8-1)+(12-2)+(21-3)) /4 = 8.75

Shortest Job First (SJF)

- CPU schedules the task with the shortest remaining time

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

non-preemptive

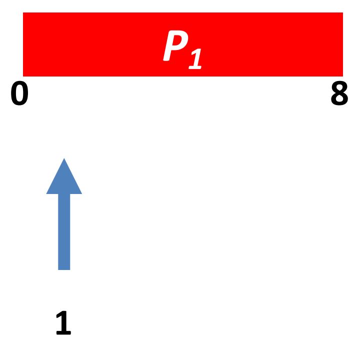


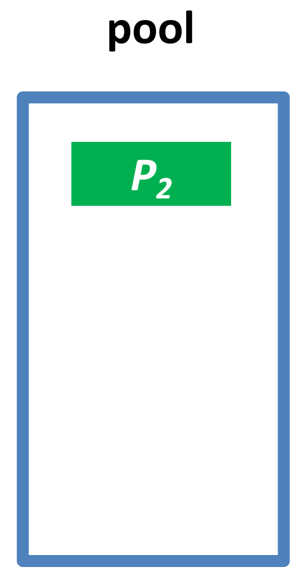
Shortest Job First (SJF)

- CPU schedules the task with the shortest remaining time

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

non-preemptive





Shortest Job First (SJF)

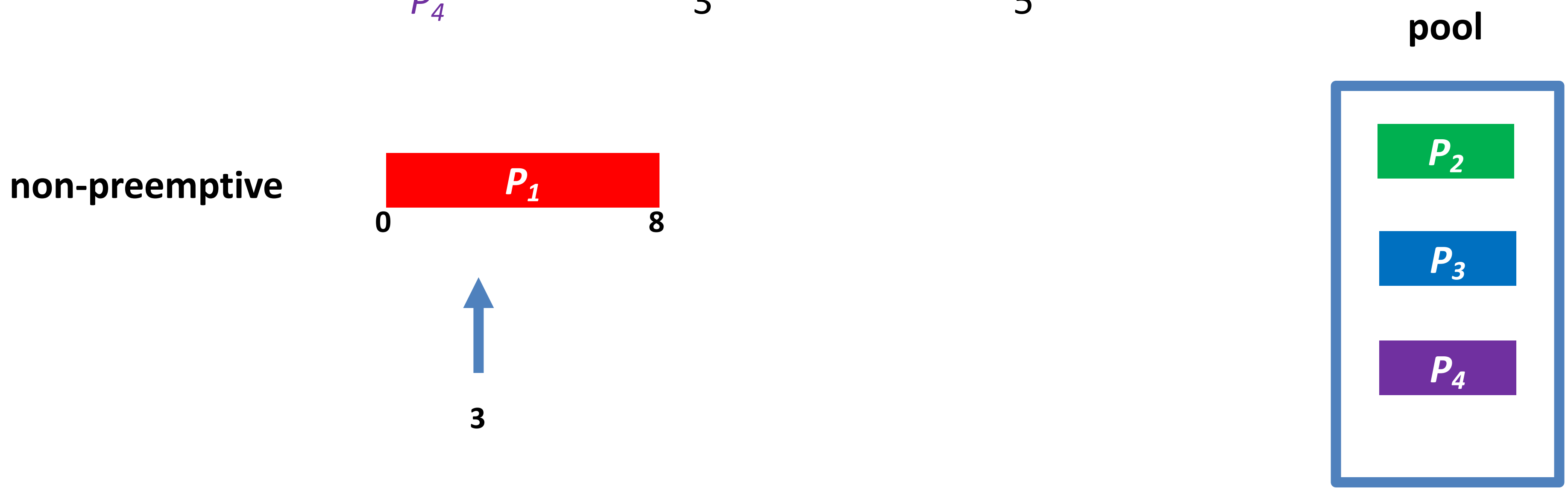
- CPU schedules the task with the shortest remaining time

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

Shortest Job First (SJF)

- CPU schedules the task with the shortest remaining time

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

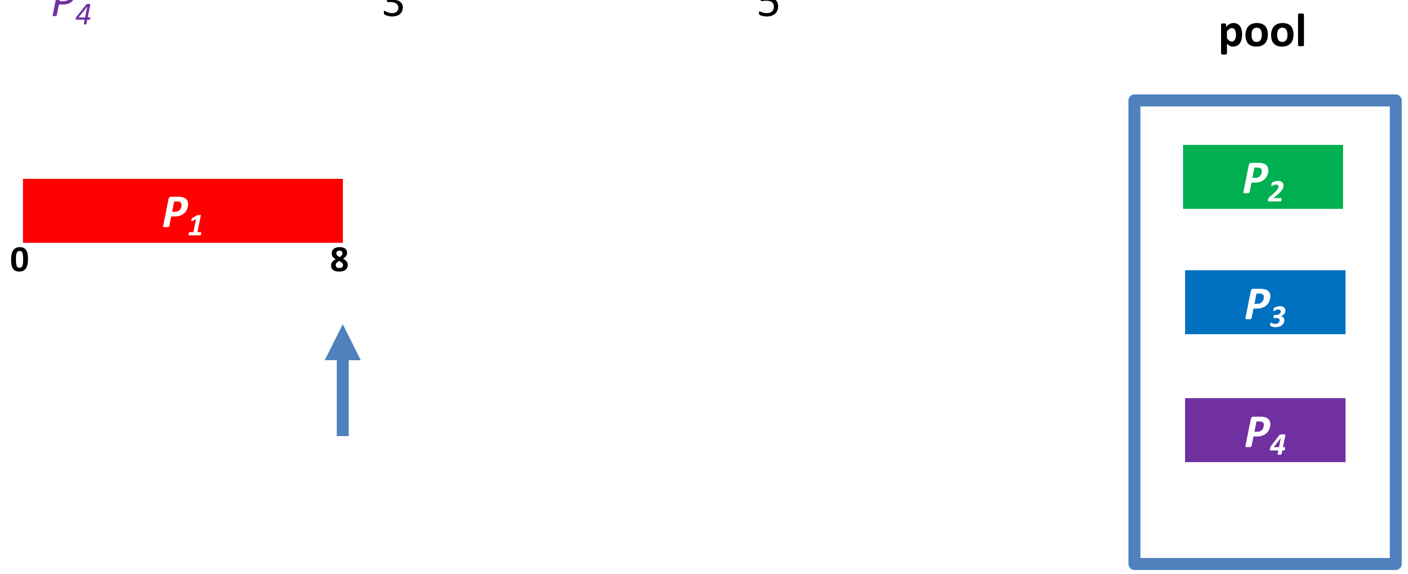


Shortest Job First (SJF)

- CPU schedules the task with the shortest remaining time

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

non-preemptive

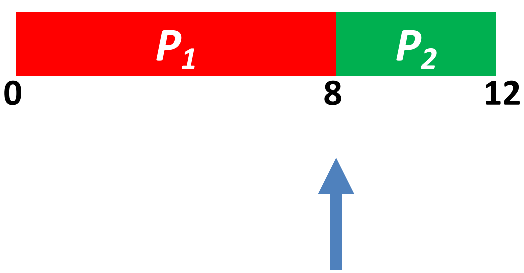


Shortest Job First (SJF)

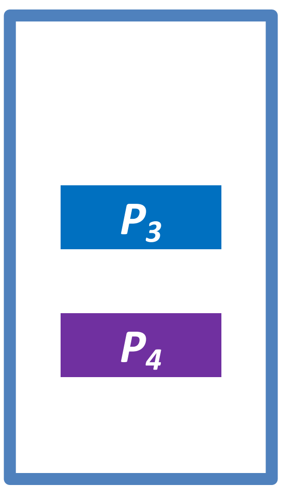
- CPU schedules the task with the shortest remaining time

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

non-preemptive



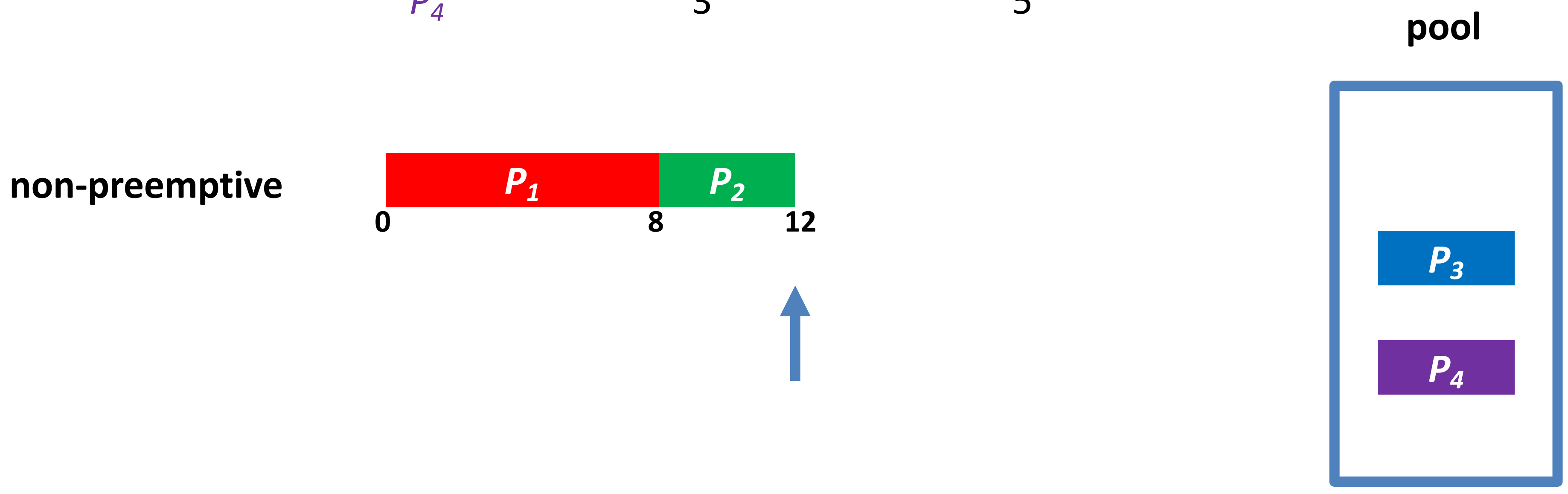
pool



Shortest Job First (SJF)

- CPU schedules the task with the shortest remaining time

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

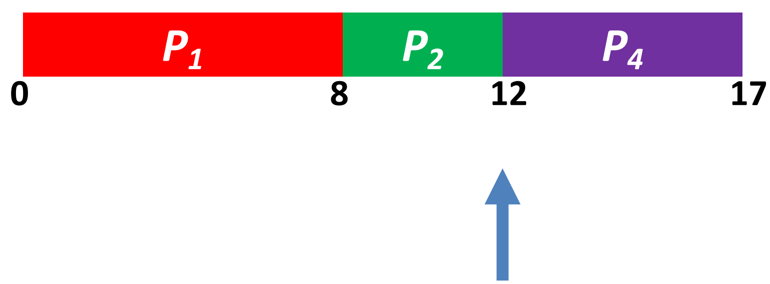


Shortest Job First (SJF)

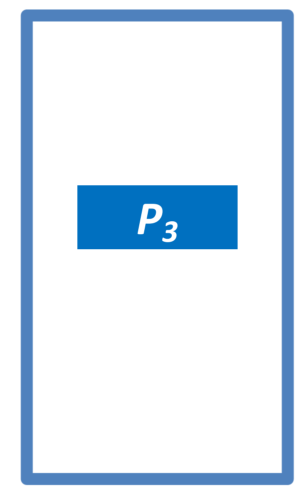
- CPU schedules the task with the shortest remaining time

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

non-preemptive



pool

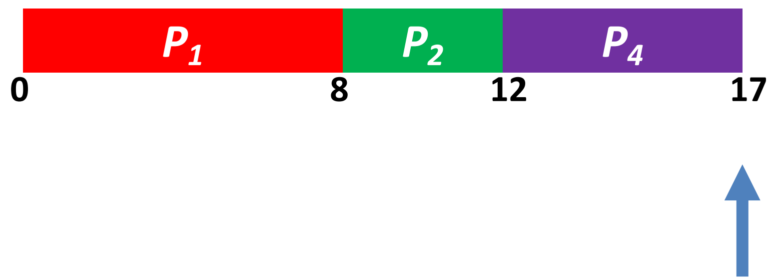


Shortest Job First (SJF)

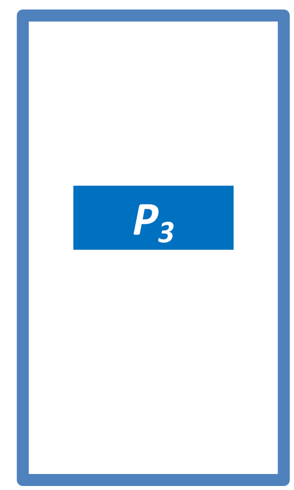
- CPU schedules the task with the shortest remaining time

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

non-preemptive



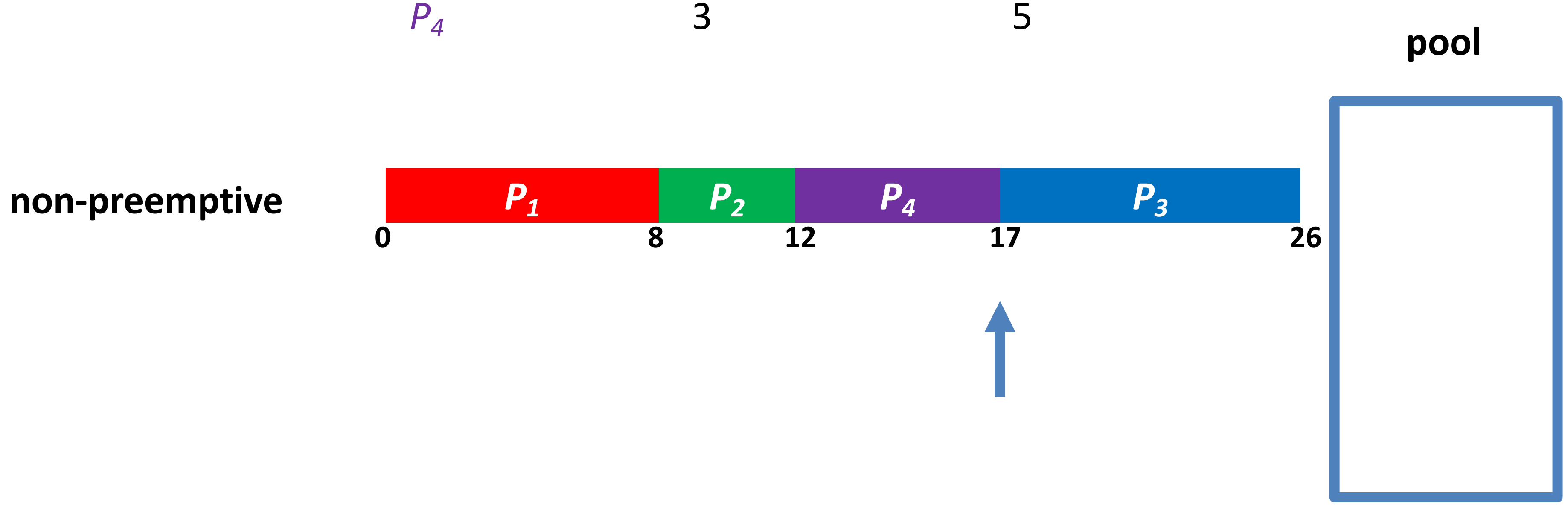
pool



Shortest Job First (SJF)

- CPU schedules the task with the shortest remaining time

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



Shortest Job First (SJF)

- CPU schedules the task with the shortest remaining time

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

non-preemptive

Average turnaround time = ((8-0)+(12-1)+(26-2)+(17-3)) / 4 = 14.25

Shortest Job First (SJF)

- CPU schedules the task with the shortest remaining time

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

non-preemptive

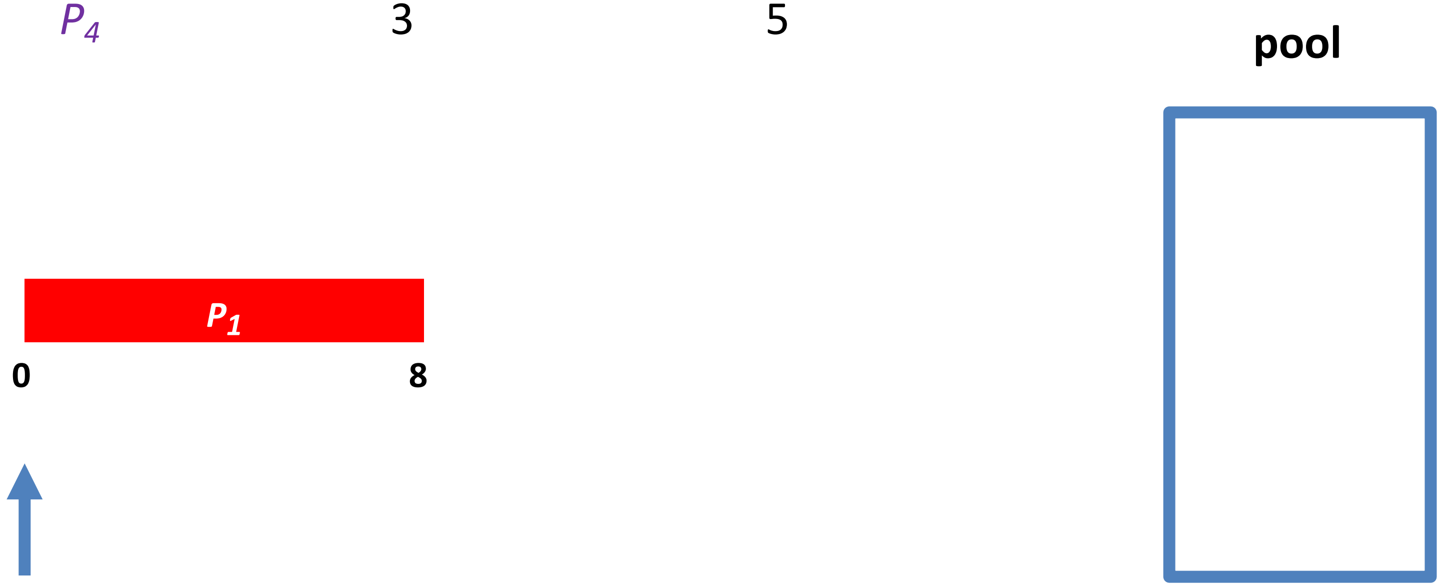
Average response time = (0+(8-1)+(17-2)+(12-3)) /4 = 7.75

Shortest Job First (SJF) with preemption

- CPU schedules the task with the shortest remaining time

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

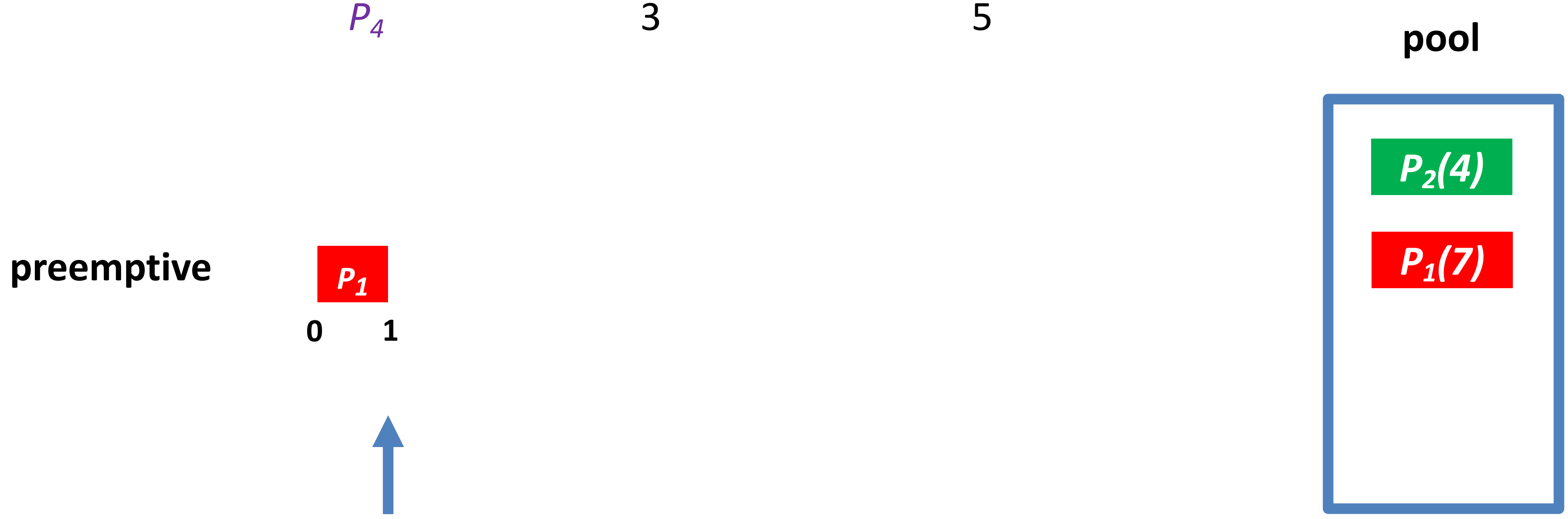
preemptive



Shortest Job First (SJF) with preemption

- CPU schedules the task with the shortest remaining time

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

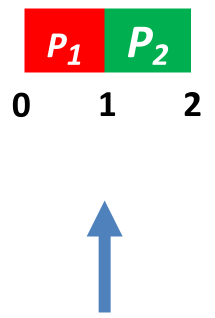


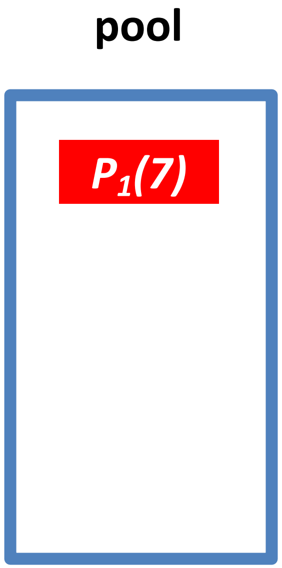
Shortest Job First (SJF) with preemption

- CPU schedules the task with the shortest remaining time

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

preemptive



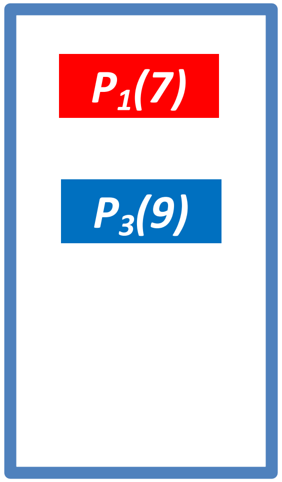


Shortest Job First (SJF) with preemption

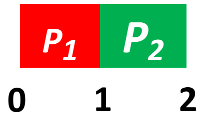
- CPU schedules the task with the shortest remaining time

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

pool



preemptive



P2 remain =3

Shortest Job First (SJF) with preemption

- CPU schedules the task with the shortest remaining time

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

pool

$P_1(7)$

$P_3(9)$

$P_4(5)$

preemptive

P_1	P_2

0 **1** **3**

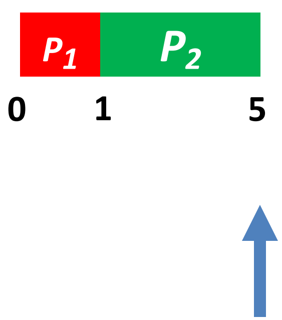
P2 remain =2

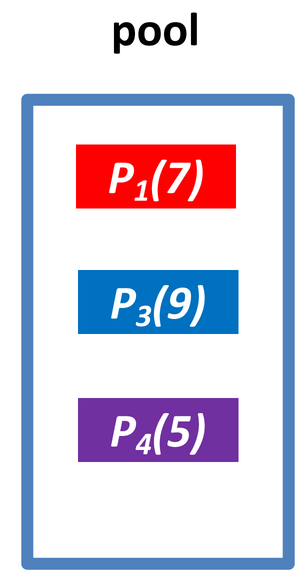
Shortest Job First (SJF) with preemption

- CPU schedules the task with the shortest remaining time

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

preemptive



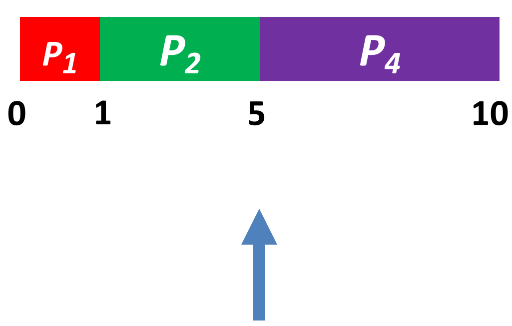


Shortest Job First (SJF) with preemption

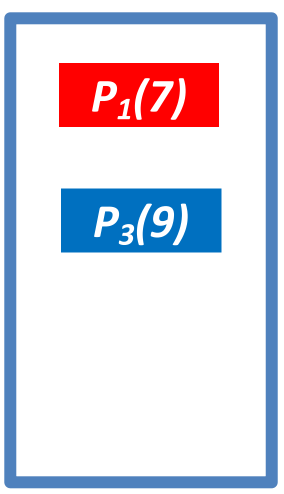
- CPU schedules the task with the shortest remaining time

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

preemptive



pool

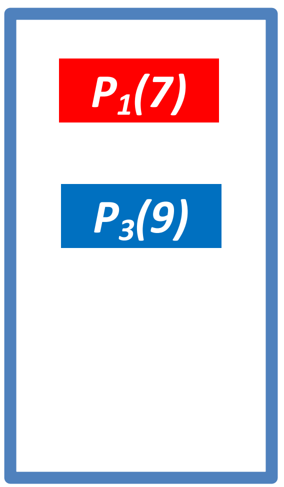


Shortest Job First (SJF) with preemption

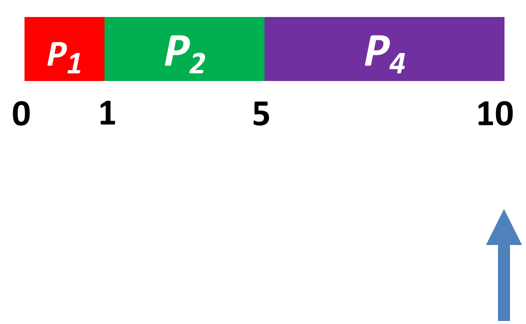
- CPU schedules the task with the shortest remaining time

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

pool



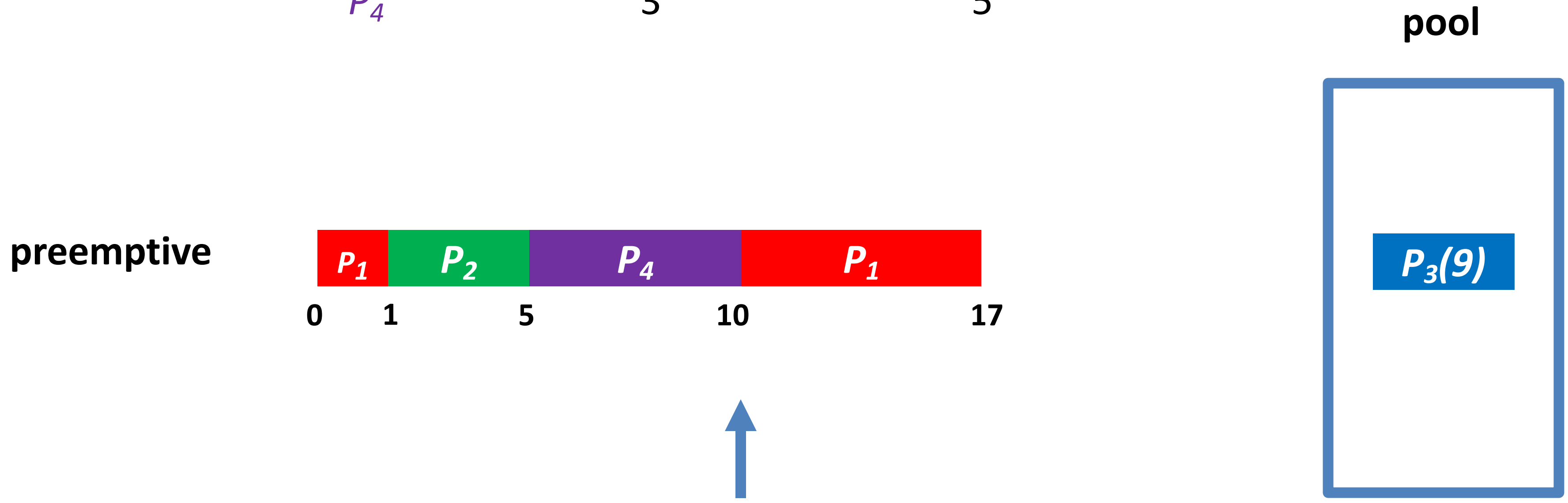
preemptive



Shortest Job First (SJF) with preemption

- CPU schedules the task with the shortest remaining time

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



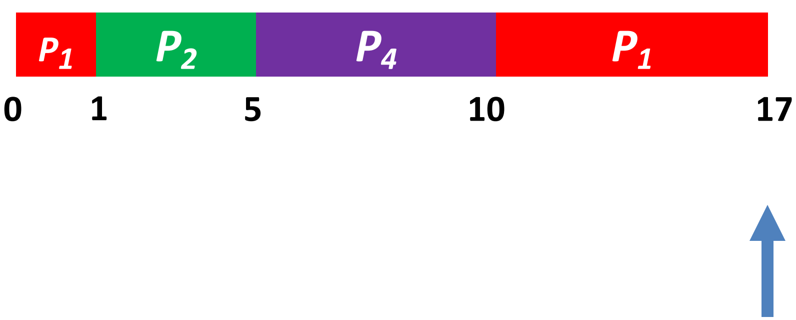
Shortest Job First (SJF) with preemption

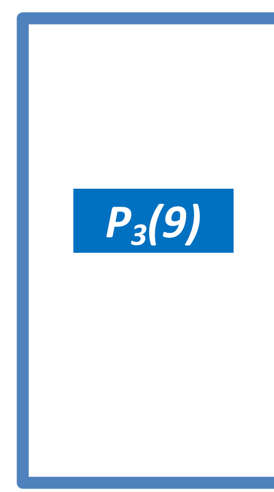
- CPU schedules the task with the shortest remaining time

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

pool

preemptive

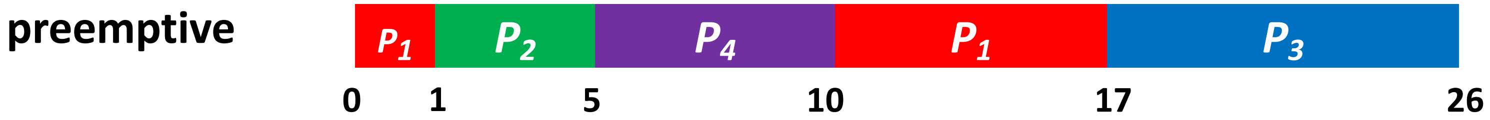




Shortest Job First (SJF) with preemption

- CPU schedules the task with the shortest remaining time

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



Average turnaround time = ((17-0)+(5-1)+(10-3)+(26-2)) / 4 = 13

Shortest Job First (SJF) with preemption

- CPU schedules the task with the shortest remaining time

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

preemptive

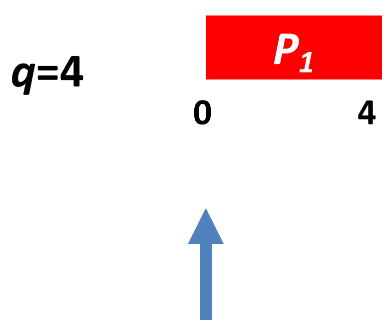
Average response time = (0+(1-1)+(5-3)+(17-2)) /4 = 4.25

Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

pool

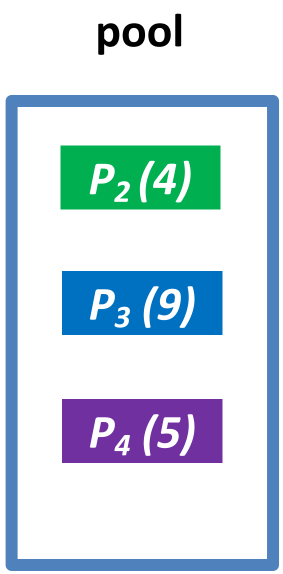
Process	Arrival Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

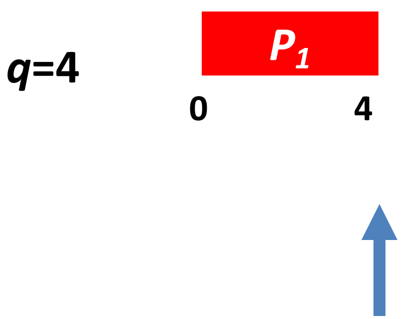


Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

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

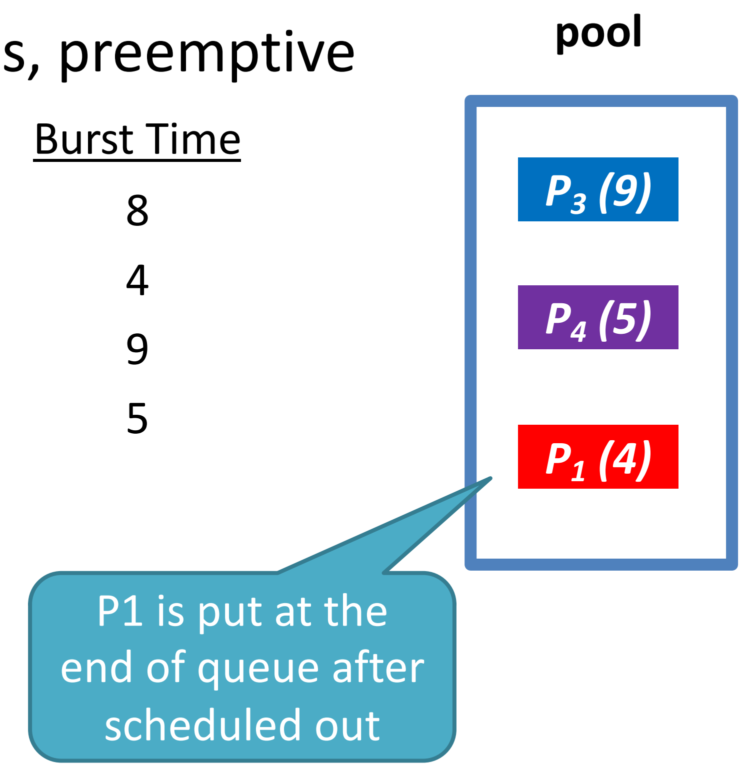


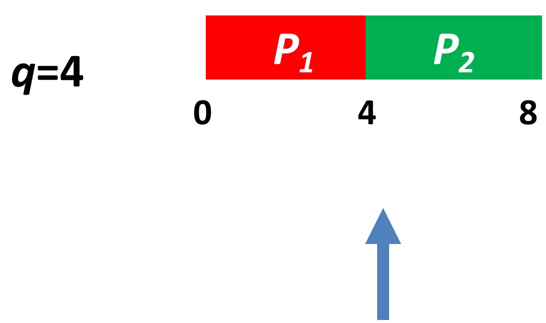


Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

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

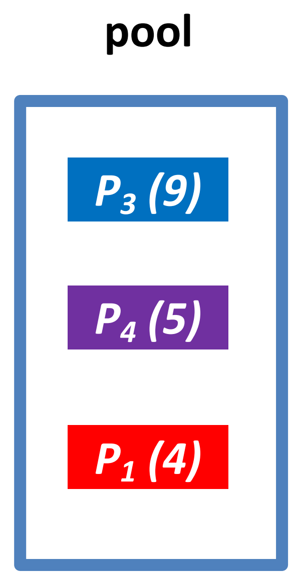


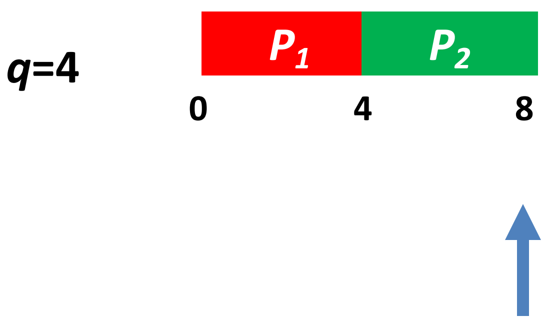


Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

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



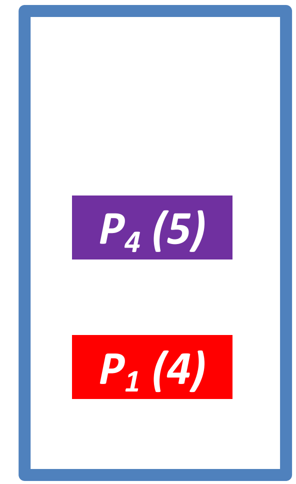


Round Robin (RR)

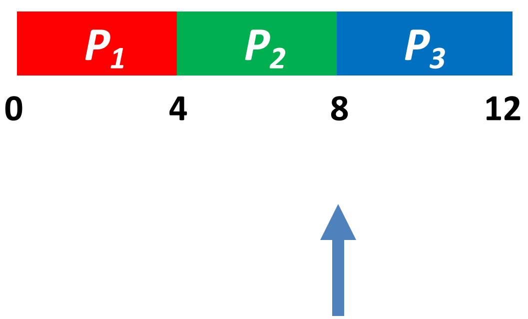
- Like FCFS, but with limited time slices, preemptive

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

pool



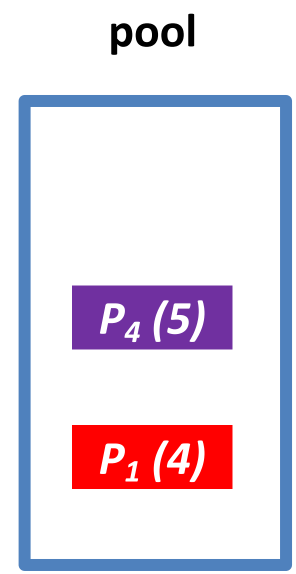
q=4



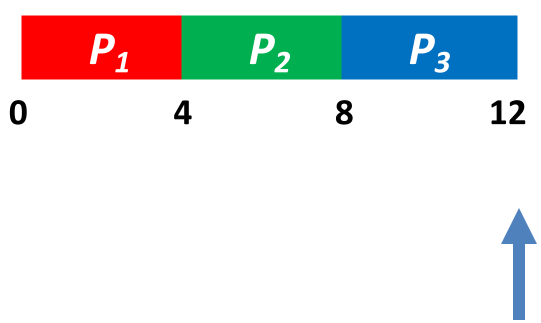
Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

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



q=4

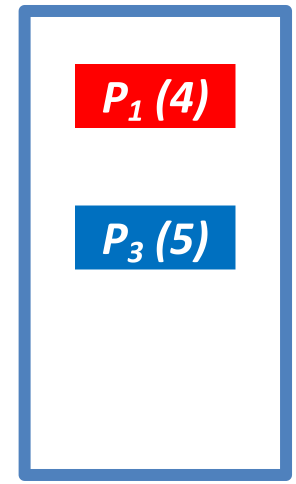


Round Robin (RR)

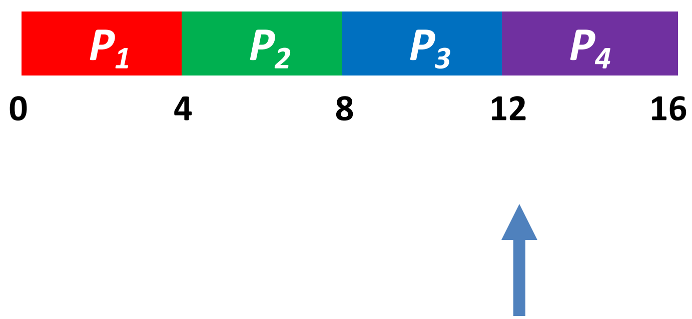
- Like FCFS, but with limited time slices, preemptive

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

pool



q=4

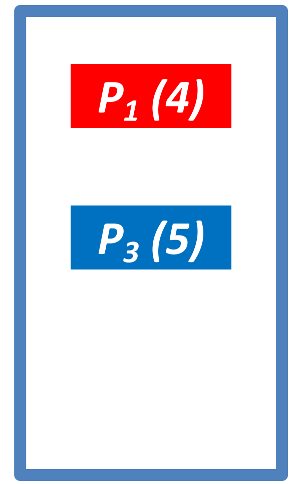


Round Robin (RR)

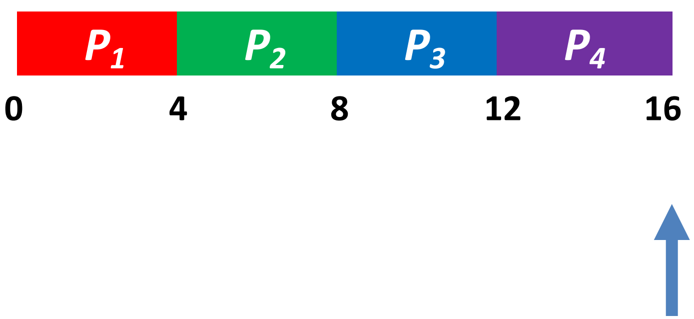
- Like FCFS, but with limited time slices, preemptive

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

pool



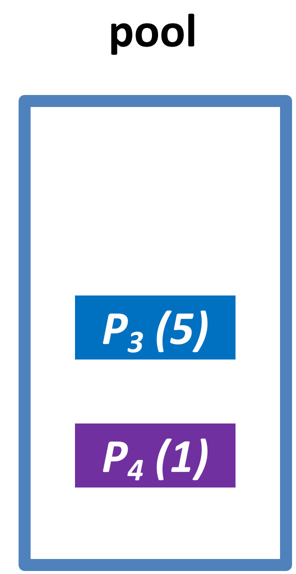
q=4

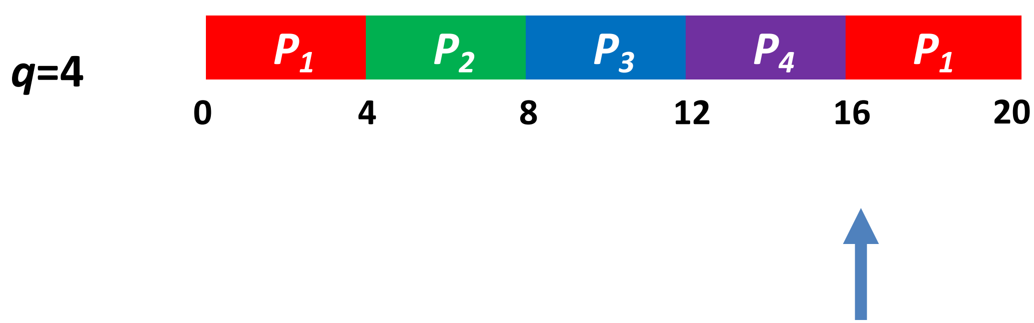


Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

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

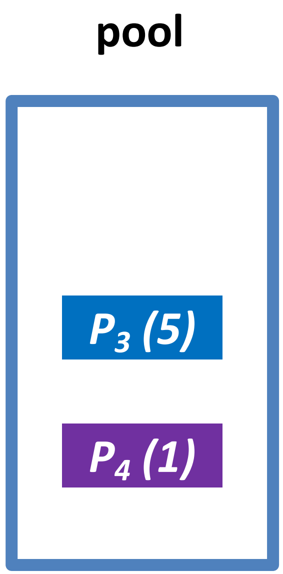


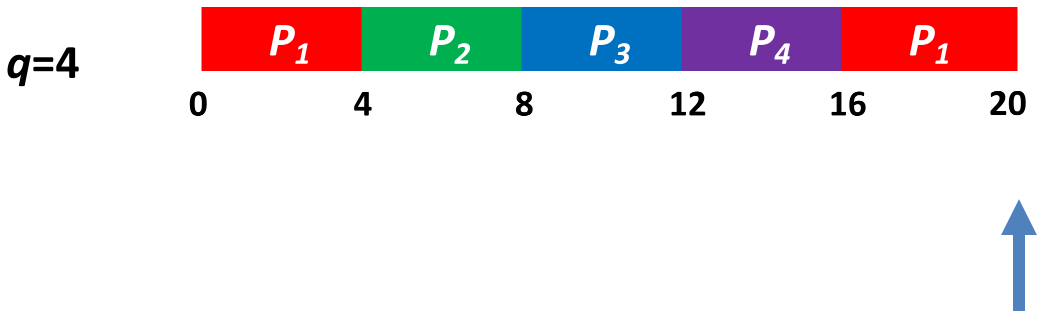


Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

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



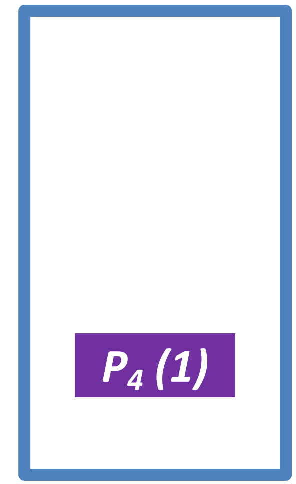


Round Robin (RR)

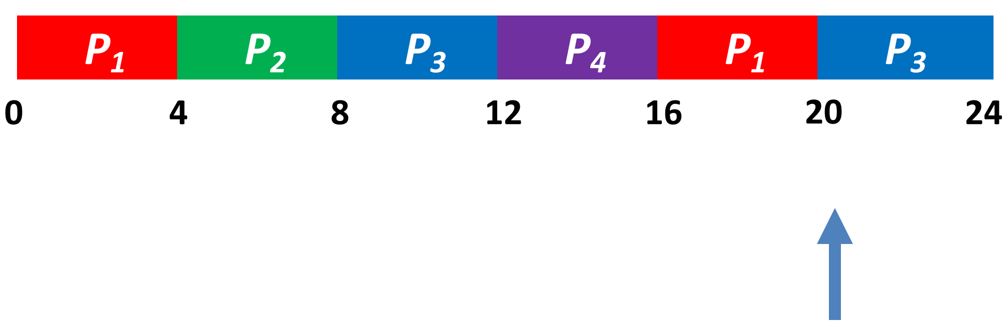
- Like FCFS, but with limited time slices, preemptive

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

pool



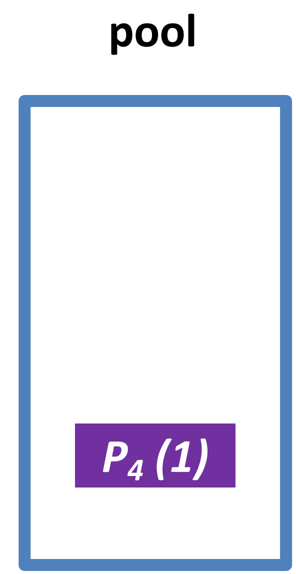
q=4

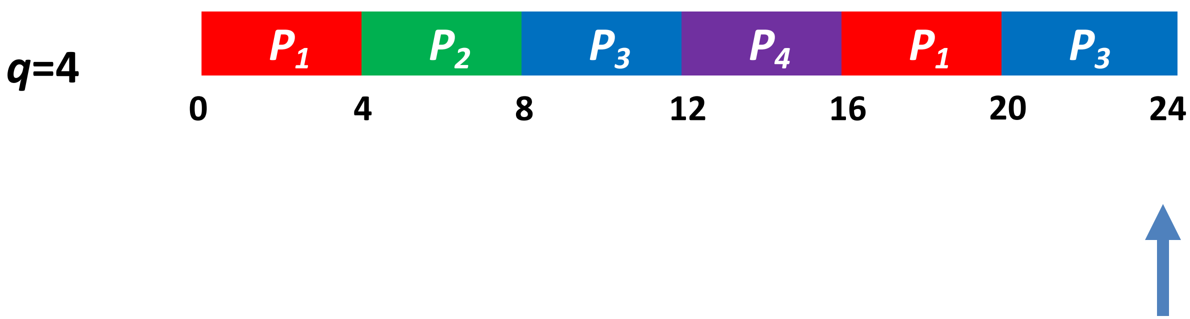


Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

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

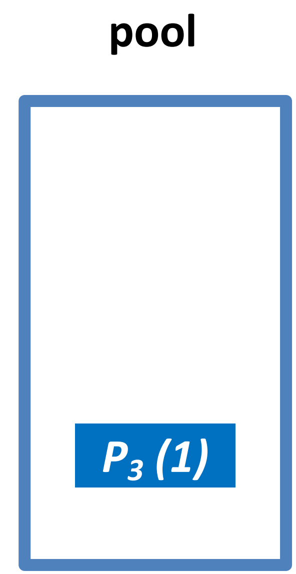




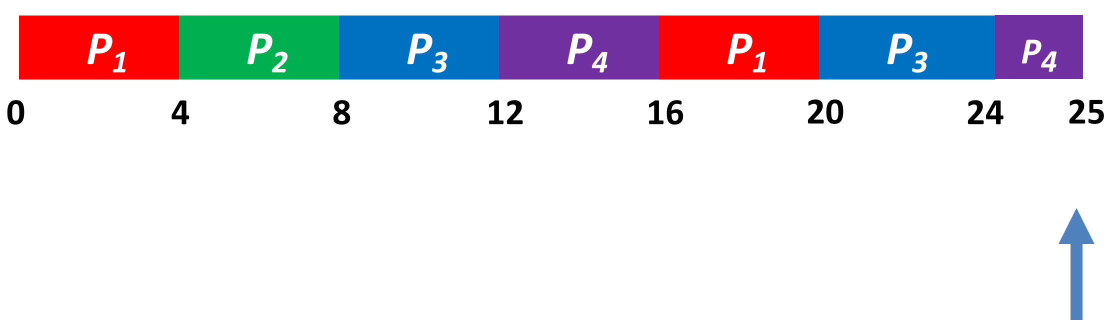
Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

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



***q*=4**



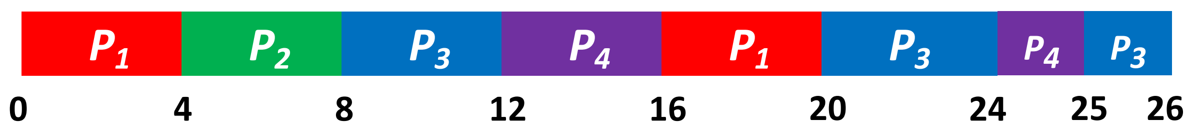
Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

pool

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

q=4



Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

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

***q*=4**

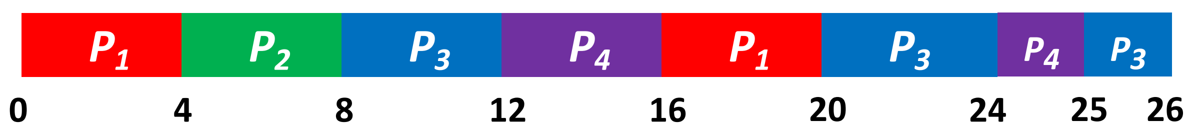
Average turnaround time = ((20-0)+(8-1)+(26-2)+(25-3)) / 4 = 18.25

Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

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

***q*=4**



Average response time = (0+(4-1)+(8-2)+(12-3)) /4 = 4.5

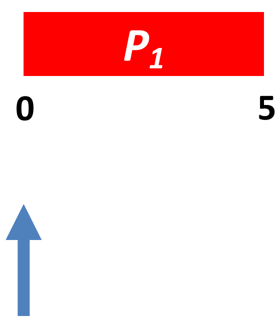
Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

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

pool

q=5

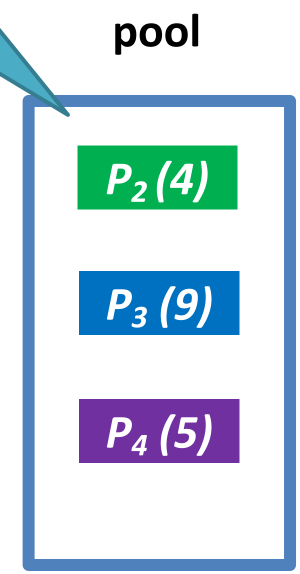


Round Robin (RR)

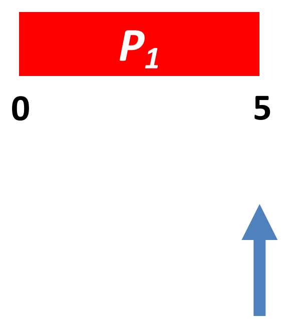
Select the task at the beginning

- Like FCFS, but with limited time slices, preemptive

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



q=5

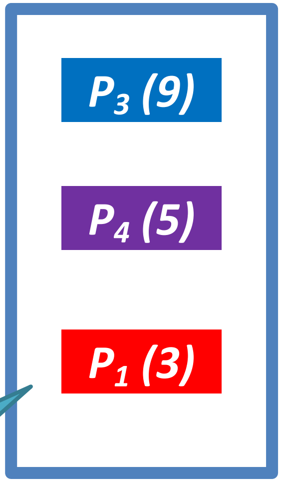


Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

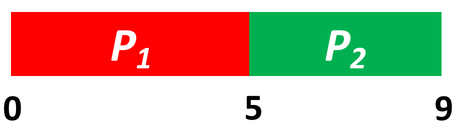
Process	Arrival Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

pool



P1 is put at the end of queue after scheduled out

q=5

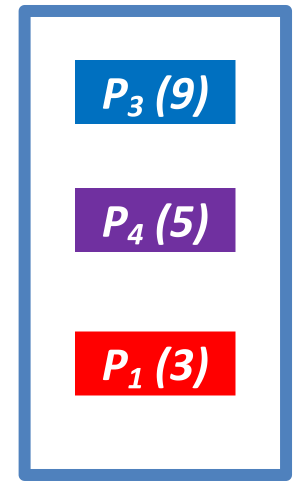


Round Robin (RR)

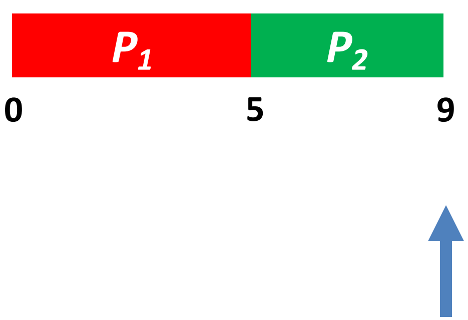
- Like FCFS, but with limited time slices, preemptive

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

pool



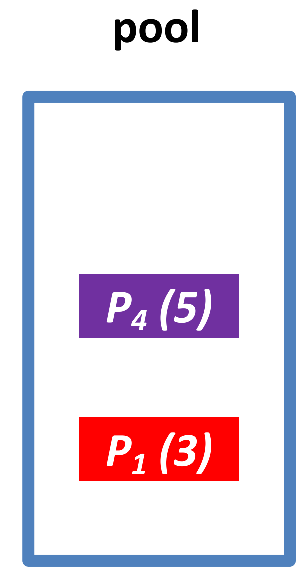
***q*=5**

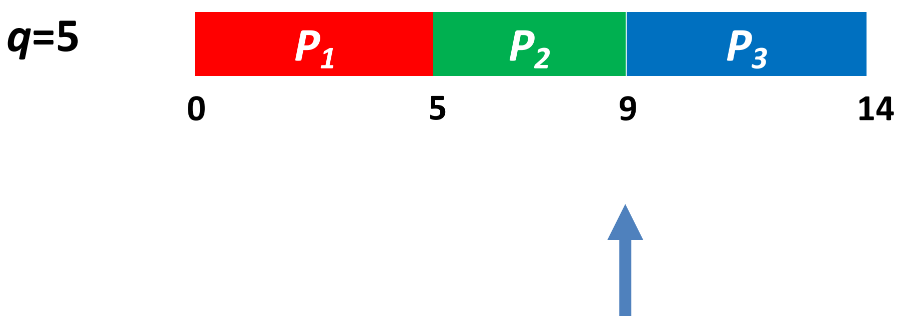


Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

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

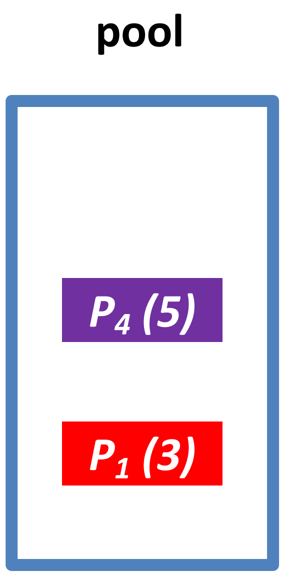


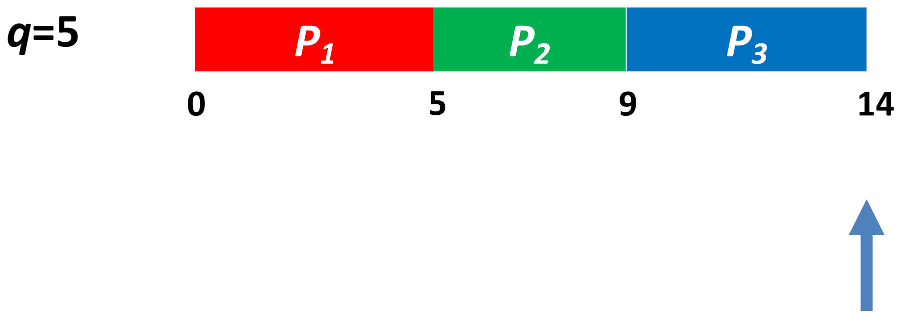


Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

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



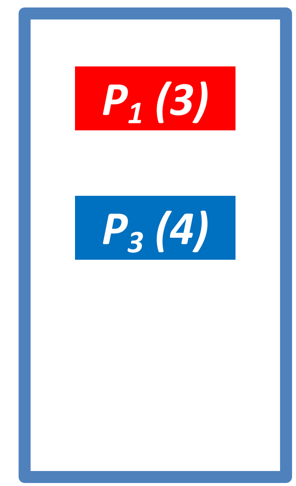


Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

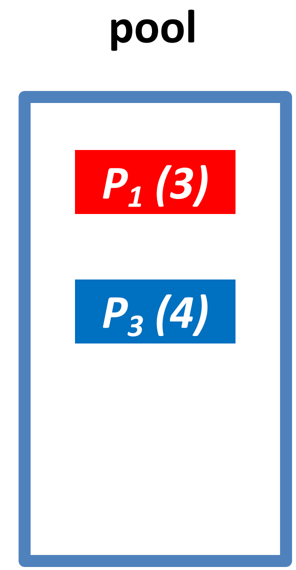
Process	Arrival Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

pool



Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive



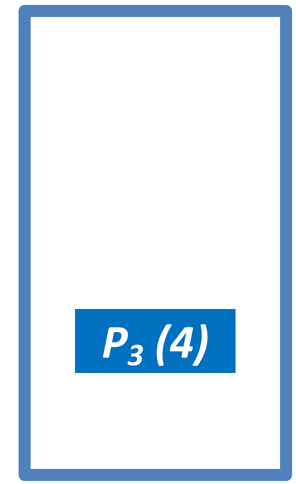
Process	Arrival Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

Round Robin (RR)

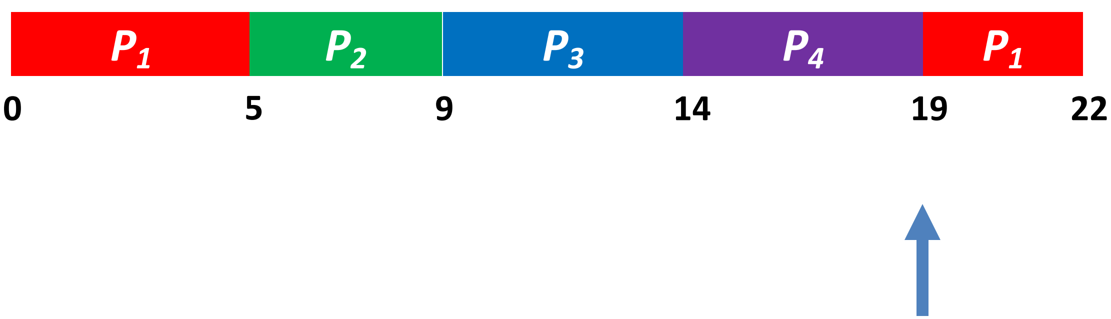
- Like FCFS, but with limited time slices, preemptive

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

pool



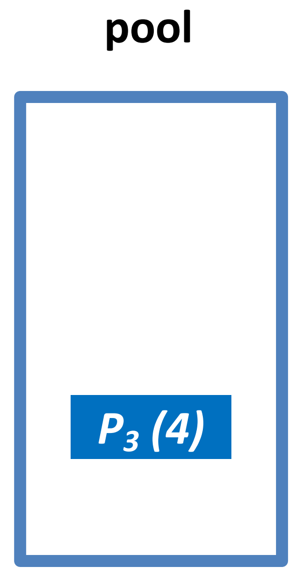
q=5

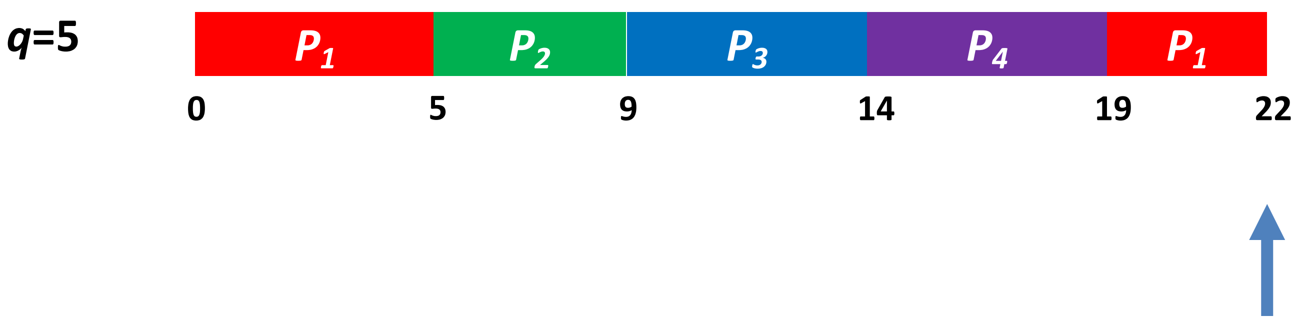


Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

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





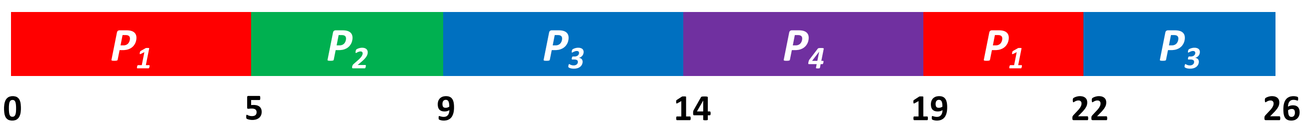
Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

pool

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

***q*=5**

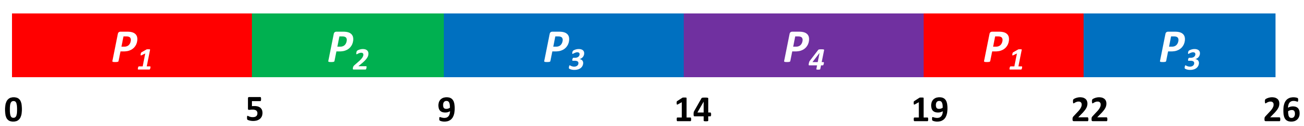


Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

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

***q*=5**



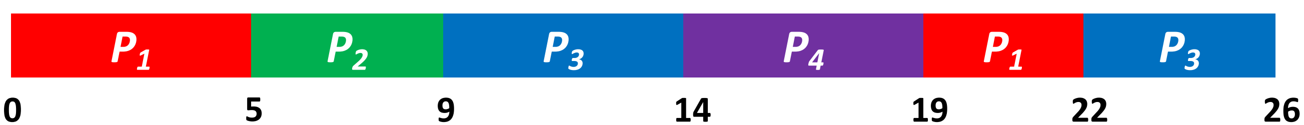
Average turnaround time = ((22-0)+(9-1)+(26-2)+(19-2)) / 4 = 17.5

Round Robin (RR)

- Like FCFS, but with limited time slices, preemptive

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

***q*=5**



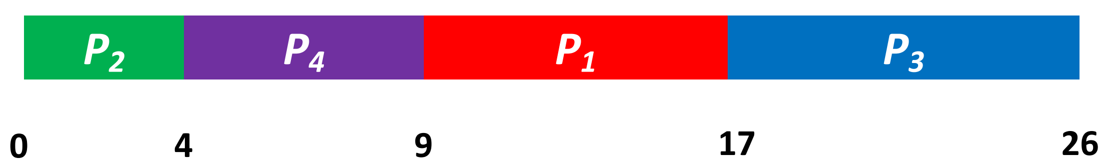
Average response time = (0+(5-1)+(9-2)+(14-3)) /4 = 5.5

Priority Scheduling

- CPU schedules the highest priority (smaller value) first, FCFS within the same priority

Process	Priority	Burst Time
P_1	3	8
P_2	1	4
P_3	4	9
P_4	2	5

Suppose arriving time is same

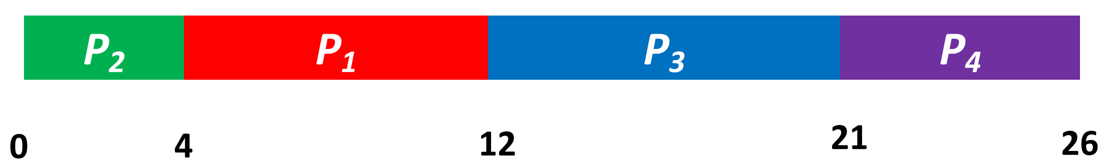


Priority Scheduling

- CPU schedules the highest priority (smaller value) first, FCFS within the same priority

Process	Priority	Burst Time
P_1	2	8
P_2	1	4
P_3	4	9
P_4	4	5

Suppose arriving time is same



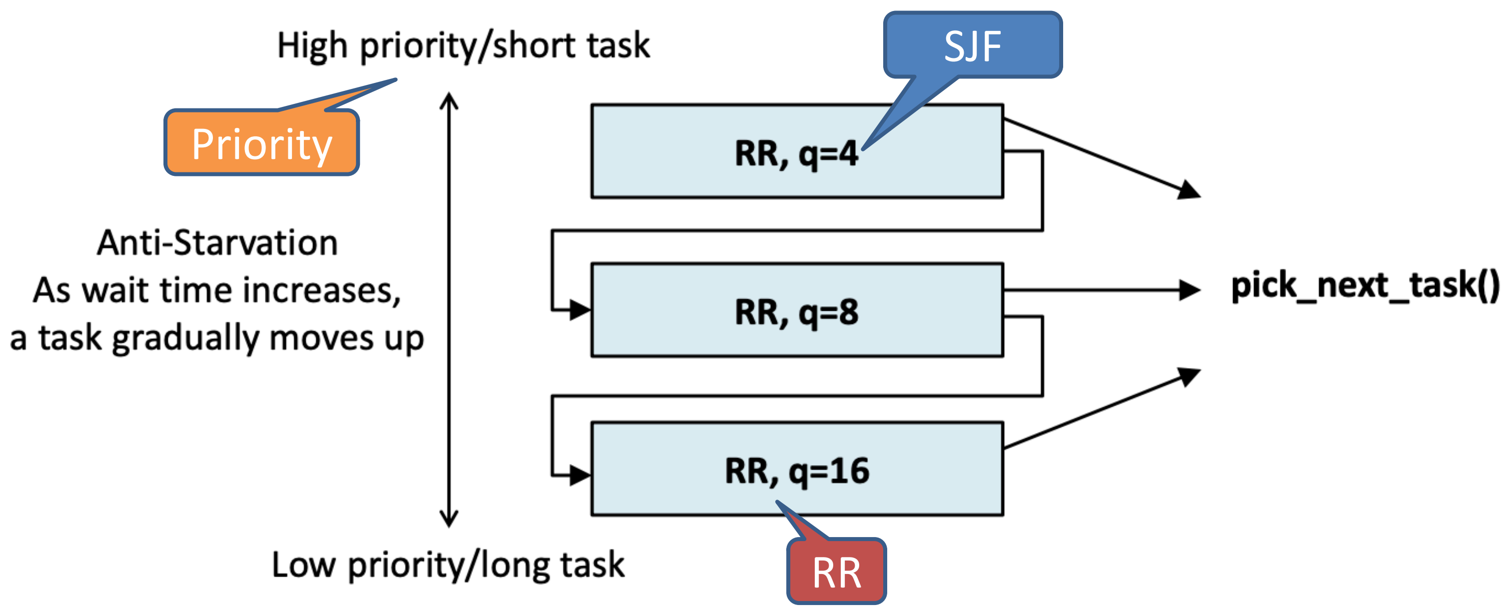
Comparison

	Turnaround time	Response time
FCFS	15.25	8.75
SJF-preemptive	**13**	**4.25**
RR (q=5)	17.5	5.5
Priority scheduling	N/A	N/A

	Throughput	Response time	Starvation
FCFS	TBD	TBD	No
SJF-preemptive	High	Good	Yes
RR	Can be low	Good	No
Priority scheduling	Can be high	Can be good	Can remove

Multilevel Feedback Queue

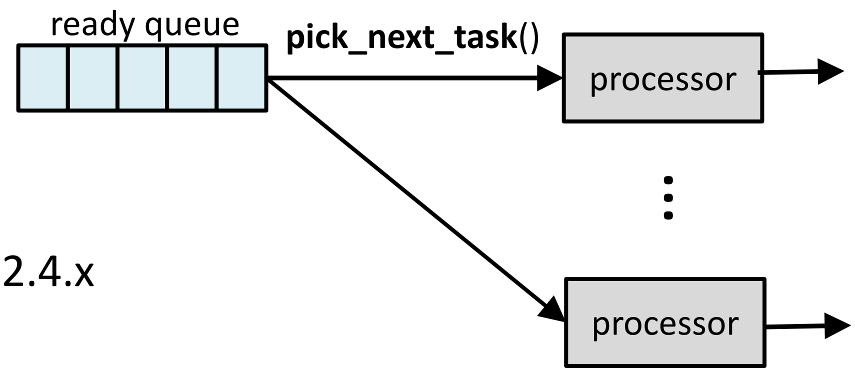
	Throughput	Response time	Starvation
FCFS	TBD	TBD	No
SJF-preemptive	High	Good	Yes
RR	Can be low	Good	No
Priority scheduling	Can be high	Can be good	Can remove



Windows XP, Mac OS X, Linux 2.6.22 and before

Challenges on Emerging Hardware and Applications

- Multi-processor → Single queue



Multiprocessor = more powerful processor

pick_next_task() will be the bottleneck

Linux 2.4.x

Pros:

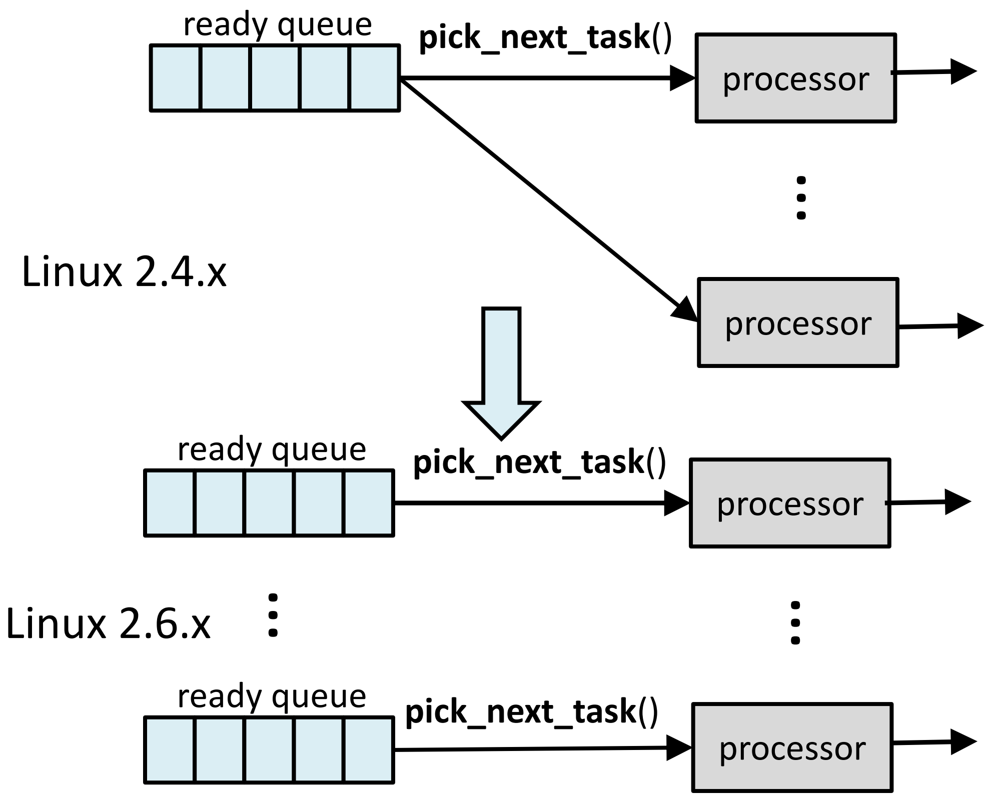
1. Easy to implement
2. Perfect load balancing

Cons:

1. Scalability issues due to centralized synchronization
2. High overhead and low efficiency
3. Hard to maintain cache hotness due to global scheduling

Challenges on Emerging Hardware and Applications

- Multi-processor → Many queues



Multiprocessor = more powerful processor

pick_next_task() will be the bottleneck

Pros:

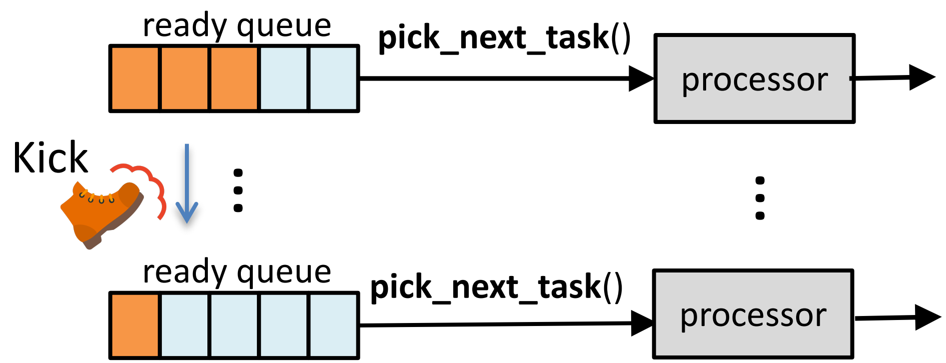
1. Scalable to many CPUs
2. Easy to maintain cache hotness

Cons:

Self-scheduling, could have load imbalance

Overcome Load Imbalance

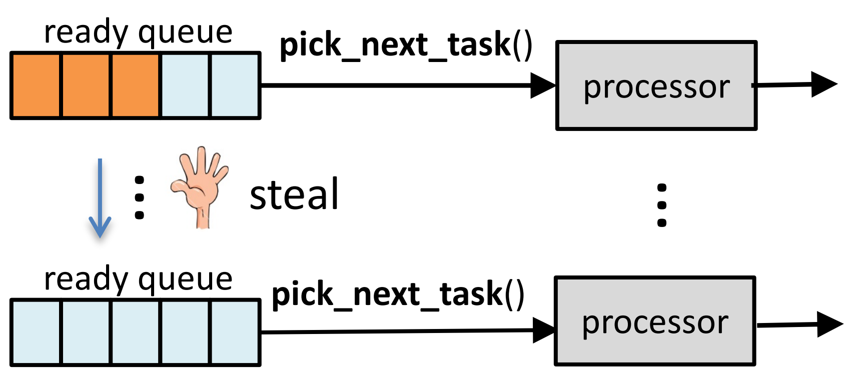
- Push model



Every a while, a kernel thread checks load imbalance and move threads

Made by OS

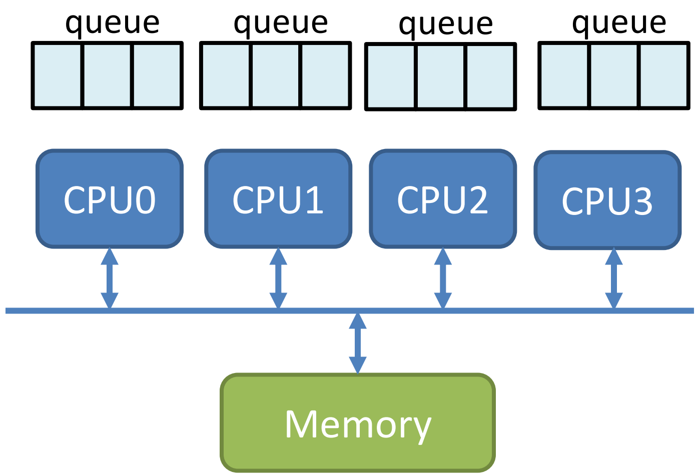
- Pull model



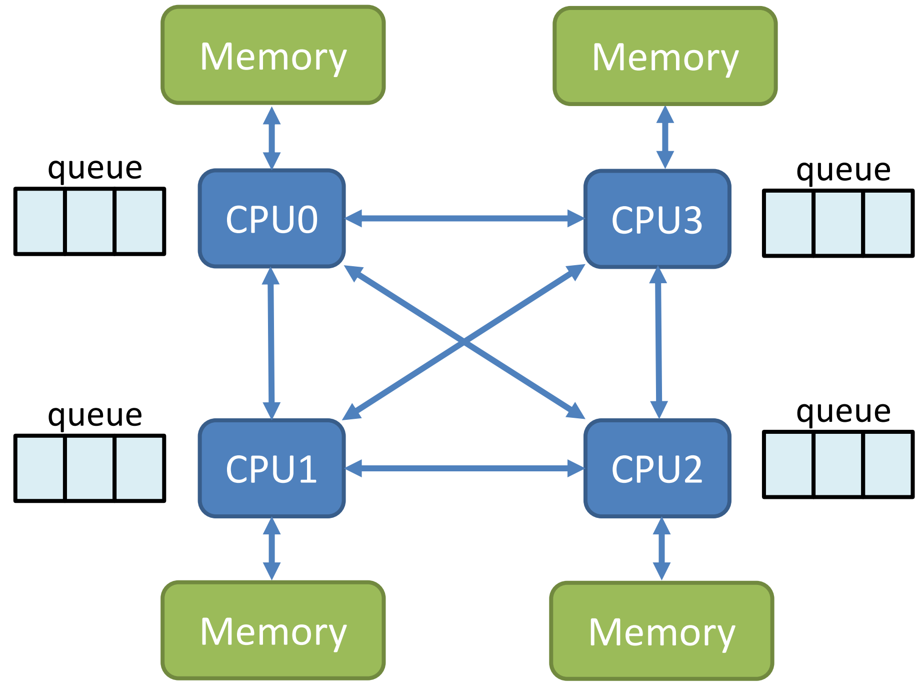
Whenever a queue becomes empty, steal a thread from non-empty queues

Made by local queue. Both are widely used

Load balance on SMP vs. NUMA



symmetric multiprocessing (SMP):
The distance to memory is the same



Non-uniform memory access (NUMA):
The distance to memory is different

Multi processor/core scheduling

The scheduling policy not only considers the fairness, throughput, etc., but also needs to consider the **hardware architecture** (e.g., locality)

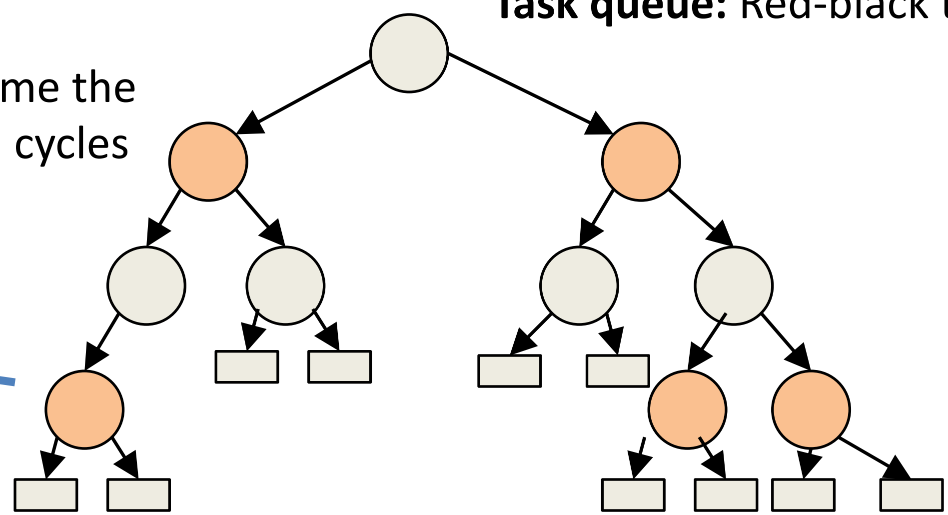
Scheduling in Linux

- Linux Completely Fair Scheduler (CFS)
  - CFS = RR + SJF + Priority + smart data structure + hardware consideration
  - One ready queue for one processor
  - Red-black tree based ready queue

Print it in project 1

vruntime: how much time the task consumes the CPU cycles

Task queue: Red-black tree



pick_next_task()
Select the leftmost task which has the least vruntime

Conclusion

- Introduction to CPU scheduling
  - What is CPU scheduling
  - Why we need CPU scheduling
  - When scheduling happens
- Scheduling policies
  - FCFS, SJF, RR, Priority
  - Scheduling on multiple CPUs
  - Scheduling in Linux