Operating Systems

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Lecture 6a

Dr Ronald Grau

School of Engineering and Informatics

Spring term 2018

Previously

Process Synchronisation

- Critical Section problem
- Synchronisation primitives:
  - Mutex, Semaphore, Monitor, Condition Variable, . . .
  - Transactional Memory
  - Message Passing

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Recap: Synchronisation primitives

Which synchronisation primitive would you use?

1. **Public transport rules**
   No more than the max. amount of people allowed enter a bus.
2. **Commissioning warehouse**
   Always put the same number of items in every box.
3. **Paying the parking fine**
   Customers get served individually in the order of their arrival.
4. **Occupied!**
   Only a single person can enter the bathroom at a time.
5. **Taking everyone onboard**
   A ship will only depart once all crew are back from land leave.
6. **Putting the cart before the horse**
   Before you can peel an egg you need to boil it first.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Today

Deadlocks

- Deadlock conditions
- Methods of handling deadlocks

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Recap: Resources

Related operations

- Request: The process requests the resource. If the request cannot be granted immediately (for example, if the resource is being used by another process), then the requesting process must wait until it can acquire the resource.
- Use: The process can operate on the resource (for example, if the resource is a printer, the process can print on the printer).
- Release: The process releases the resource.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Synchronisation and Priority Scheduling

Scenario

- Low-priority process **L** acquires resource **R**
- **L** is preempted by long-running medium-priority process **M**
- High-priority process **H** wants to acquire **R**, but is blocked
- **H** has to wait until **M** finishes before **L** can release **R**

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Problem: Priority inversion

- Higher priority process has to wait because lower priority process holds the resource it wants to access

Solution: Priority inheritance protocol

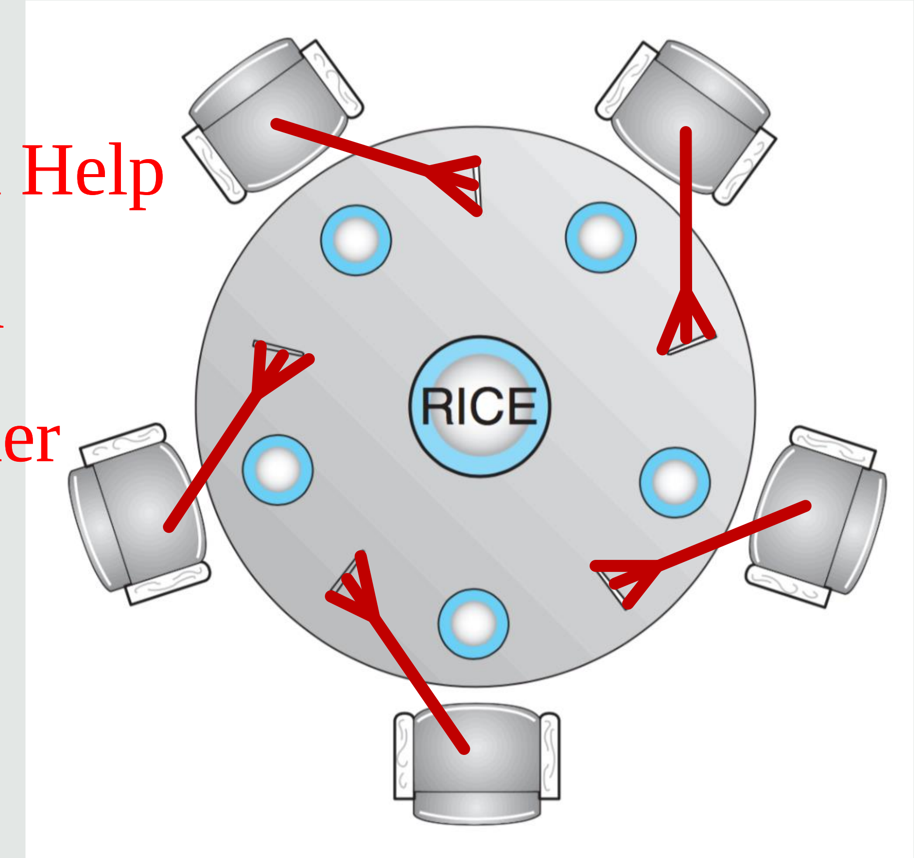
- Temporarily raise the priority of L to the maximum priority of all processes accessing the same resource
- Avoids priority inversion

Dining Philosophers Problem (Dijkstra 1965)

Rules

- The philosophers can either think or eat
- Each philosopher needs two chopsticks to eat
- They can pick up chopsticks from the left and the right as they become available

- How can we make sure they keep thinking and eating in turn, and no-one starves?



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Deadlocks

A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Necessary and sufficient conditions:

- Mutual exclusion: Each resource is either currently assigned to exactly one process or is available.
- Hold-and-wait: Processes currently holding resources that were granted earlier can request new resources.
- No-preemption: Resources previously granted cannot be forcibly taken away from a process. They must be explicitly released by the process holding them.
- Circular wait: There must be a circular list of two or more processes, each of which is waiting for a resource held by the next member of the chain.

How do deal with deadlocks?

Unfortunately, there is no general solution.

- Deadlock prevention: Assignment Project Exam Help
  - Design the system so that at least one of the four conditions never arises.

https://powcoder.com

- Deadlock avoidance:

Add WeChat powcoder

  - Do not approve resource requests that might lead to a deadlock (Safety).
- Deadlock detection:
  - Always approve resource requests if resources are available.
  - Periodically check whether there is a deadlock.
  - If there is a deadlock, perform deadlock recovery.

Deadlock Prevention – Mutual Exclusion

System designed such that one of the four conditions never occurs

- Can we build a system with shared resource access not requiring mutual exclusion?

  No, it is always necessary to prevent race conditions

- General precautions:
  - Make any data resources read-only where there is no need for modification
  - Acquire only those resources that are really needed

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Deadlock Prevention – H&W and NP

- Hold-and-wait
  - Acquire all resources at once, i.e. block until all resources are available

Assignment Project Exam Help

    + Process can execute once all required resources are held

https://powcoder.com

    - Long delays
    - Bad resource utilisation

Add WeChat powcoder

    - Advance knowledge needed about the resources required

- No-preemption

  (a) Process releases resources when it fails to acquire a resource, and re-tries later

  (b) Resource request forces another process to release a resource

  Only possible if the state of the resource can be saved and restored

Deadlock Prevention – Circular wait

- Enforce linear ordering of requests to resource types R_i
  e.g. O(disk drive) = 5, O(printer) = 12
- Process has to acquire all resources of type R_i at once

Assignment Project Exam Help

- After having acquired R_i it can only request resources of type R_j where $O(R_j) > O(R_i)$

Does this prevent deadlocks? Yes

https://powcoder.com

Add WeChat powcoder

- Assuming circular wait condition holds: Each process P_i of processes $P_0 \ldots P_{n-1}$ waits for resource R_i held by process $R_{(i+1) \bmod n}$
  - Our ordering ensures that for each **i** we have $O(R_i) < O(R_{(i+1) \bmod n})$
  - This would mean $O(R_0) < \ldots < O(R_{n-1}) < O(R_0)$
  - Impossible → circular wait condition does not hold

  However, there can be inefficiencies if ordering is poorly chosen

Deadlock Avoidance

Do not approve resource requests that might lead to a deadlock

- When do we know that the system is in a state that might lead to a deadlock?

Assignment Project Exam Help

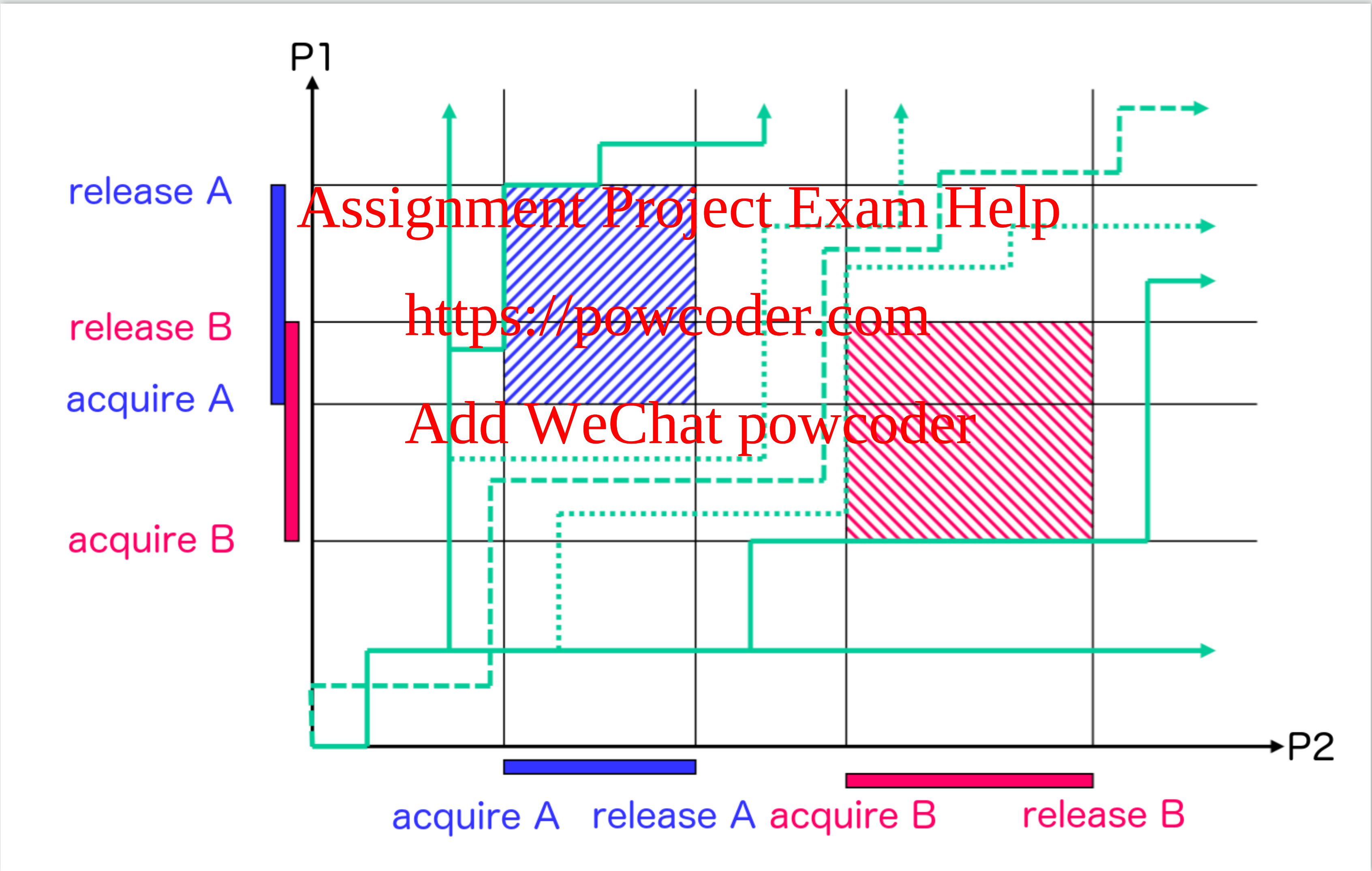
https://powcoder.com

- Deadlock example with two processes:

Add WeChat powcoder

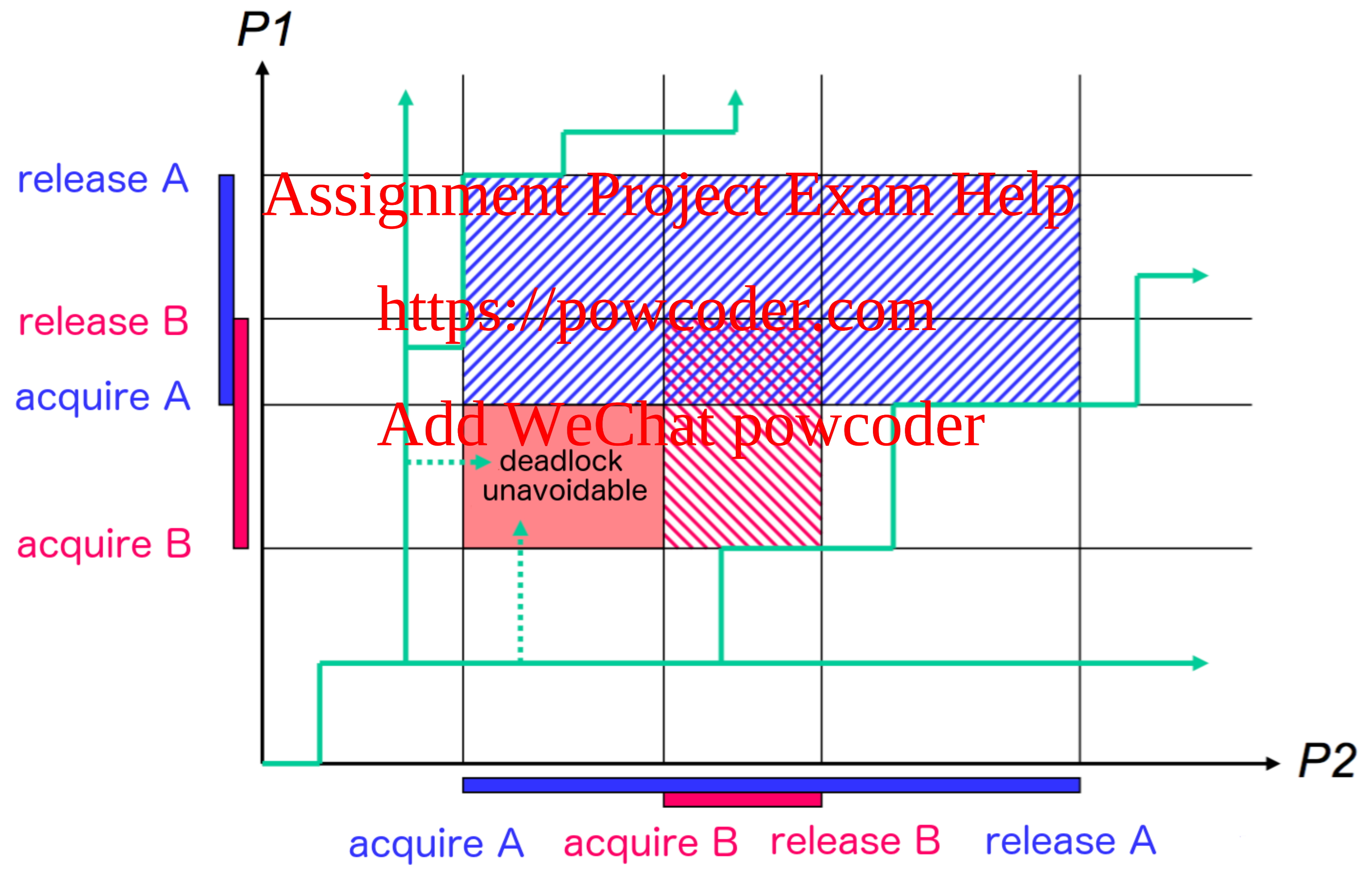
P1	P2
…	…
acquire B	acquire A
…	…
acquire A	release A
…	…
release B	acquire B
…	…
release A	release B
…	…

Resource Trajectories



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Resource Trajectories

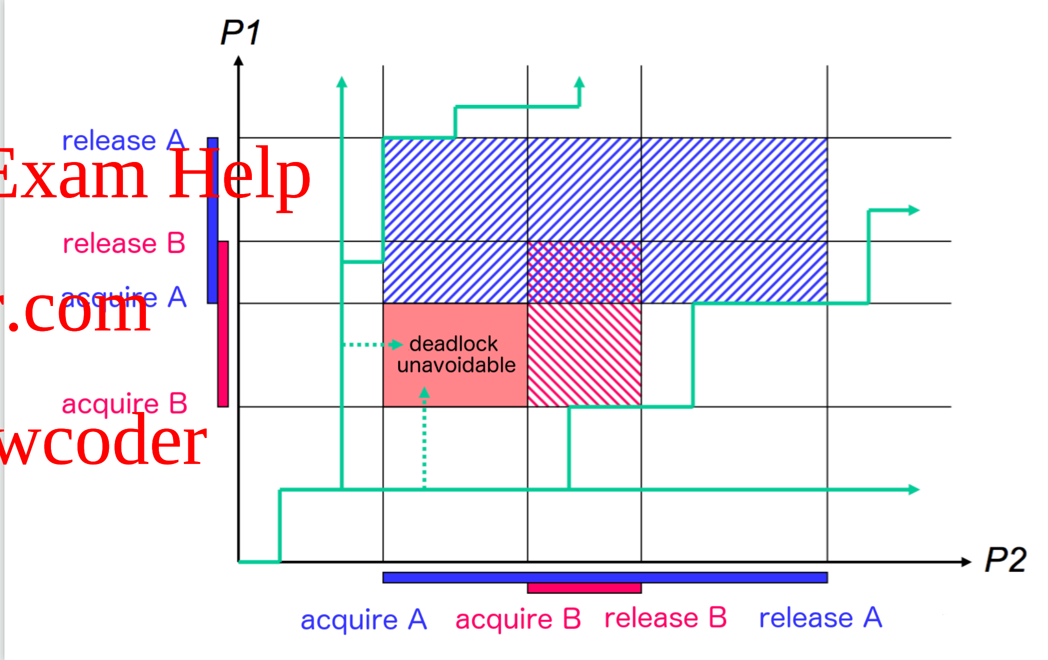


Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Safe and unsafe system states

- Deadlock or unavoidable-deadlock state
  - The system is deadlocked or it will be deadlocked on all trajectories.
- Unreachable state
  - No trajectory can reach such a state, e.g. because of mutual exclusion.
- Safe state:
  - All possible trajectories starting in a safe state are guaranteed to be deadlock-free
- Unsafe state:
  - Existing trajectories starting in unsafe state that lead to deadlock
  - Depends on scheduler

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder



Banker's Algorithm

An algorithm for avoiding deadlocks

- There are `Available[j]` instances of each resource type R_j
- Each process P_i declares the maximum of instances `Maximum[i][j]` of each resource type R_j required
- `Allocation[i][j]` is the number of instances of resource type R_j held by process P_i

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Available		
R_0	R_1	R_2
3	3	2

Maximum			
	R_0	R_1	R_2
P_0	7	5	3
P_1	3	2	2
P_2	9	0	2
P_3	2	2	2
P_4	4	3	3

Allocation			
	R_0	R_1	R_2
P_0	0	1	0
P_1	2	0	0
P_2	3	0	2
P_3	2	1	1
P_4	0	0	2

Banker's Algorithm

An algorithm for avoiding deadlocks

- `Need[i][j] = Maximum[i][j] - Allocation[i][j]`

Assignment Project Exam Help

Need			
	R_0	R_1	R_2
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

=

https://powcoder.com

Maximum			
	R_0	R_1	R_2
P_0	7	5	3
P_1	3	2	2
P_2	9	0	2
P_3	2	2	2
P_4	4	3	3

Add WeChat powcoder

-

Allocation			
	R_0	R_1	R_2
P_0	0	1	0
P_1	2	0	0
P_2	3	0	2
P_3	2	1	1
P_4	0	0	2

Banker's Algorithm – Checking Safety

1. `Work := Available`
   `Finish[i] := false` for `i = 0 … n-1`
2. Find an index i such that `Finish[i] = false ^ 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.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Component-wise ordering on vectors:

`X <= Y` if for all i : `X[i] <= Y[i]`

Banker's Algorithm

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Allocation			
	R_0	R_1	R_2
P_0	0	1	0
P_1	2	0	0
P_2	3	0	2
P_3	2	1	1
P_4	0	0	2

Need			
	R_0	R_1	R_2
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Finish	
P_0	false
P_1	false
P_2	false
P_3	false
P_4	false

Work		
R_0	R_1	R_2
3	3	2

Is there are process that can finish with the resources available (Work)?

Banker's Algorithm

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Allocation			
	R_0	R_1	R_2
P_0	0	1	0
P_1	2	0	0
P_2	3	0	2
P_3	2	1	1
P_4	0	0	2

Need			
	R_0	R_1	R_2
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Finish	
P_0	false
P_1	false
P_2	false
P_3	false
P_4	false

Work		
R_0	R_1	R_2
3	3	2

Yes, P_1

Banker's Algorithm

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Allocation			
	R_0	R_1	R_2
P_0	0	1	0
P_1	0	0	0
P_2	3	0	2
P_3	2	1	1
P_4	0	0	2

Need			
	R_0	R_1	R_2
P_0	7	4	3
P_1	0	0	0
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Finish	
P_0	false
P_1	true
P_2	false
P_3	false
P_4	false

Work		
R_0	R_1	R_2
5	3	2

Finish the process and then release all resources previously allocated to **P_1** , i.e. add previous allocation (2,0,0) to *Work* and set Finish = true

Banker's Algorithm

Allocation			
	R_0	R_1	R_2
P_0	0	1	0
P_1	0	0	0
P_2	3	0	2
P_3	2	1	1
P_4	0	0	2

Need			
	R_0	R_1	R_2
P_0	7	4	3
P_1	0	0	0
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Finish	
P_0	false
P_1	true
P_2	false
P_3	false
P_4	false

Work		
R_0	R_1	R_2
5	3	2

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Banker's Algorithm

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Allocation			
	R_0	R_1	R_2
P_0	0	1	0
P_1	0	0	0
P_2	3	0	2
P_3	0	0	0
P_4	0	0	2

Need			
	R_0	R_1	R_2
P_0	7	4	3
P_1	0	0	0
P_2	6	0	0
P_3	0	0	0
P_4	4	3	1

Finish	
P_0	false
P_1	true
P_2	false
P_3	true
P_4	false

Work		
R_0	R_1	R_2
7	4	3

Banker's Algorithm

Allocation			
	R_0	R_1	R_2
P_0	0	1	0
P_1	0	0	0
P_2	3	0	2
P_3	0	0	0
P_4	0	0	2

Need			
	R_0	R_1	R_2
P_0	7	4	3
P_1	0	0	0
P_2	6	0	0
P_3	0	0	0
P_4	4	3	1

Finish	
P_0	false
P_1	true
P_2	false
P_3	true
P_4	false

Work		
R_0	R_1	R_2
7	4	3

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Banker's Algorithm

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Allocation			
	R_0	R_1	R_2
P_0	0	0	0
P_1	0	0	0
P_2	3	0	2
P_3	0	0	0
P_4	0	0	2

Need			
	R_0	R_1	R_2
P_0	0	0	0
P_1	0	0	0
P_2	6	0	0
P_3	0	0	0
P_4	4	3	1

Finish	
P_0	true
P_1	true
P_2	false
P_3	true
P_4	false

Work		
R_0	R_1	R_2
7	5	3

Banker's Algorithm

Allocation

	R_0	R_1	R_2
P_0	0	0	0
P_1	0	0	0
P_2	3	0	2
P_3	0	0	0
P_4	0	0	2

Need

	R_0	R_1	R_2
P_0	0	0	0
P_1	0	0	0
P_2	6	0	0
P_3	0	0	0
P_4	4	3	1

Finish

P_0	true
P_1	true
P_2	false
P_3	true
P_4	false

Work

R_0	R_1	R_2
7	5	3

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Banker's Algorithm

- Safe!

Allocation			
	R_0	R_1	R_2
P_0	0	0	0
P_1	0	0	0
P_2	0	0	0
P_3	0	0	0
P_4	0	0	0

Need			
	R_0	R_1	R_2
P_0	0	0	0
P_1	0	0	0
P_2	0	0	0
P_3	0	0	0
P_4	0	0	0

Finish	
P_0	true
P_1	true
P_2	true
P_3	true
P_4	true

Work		
R_0	R_1	R_2
10	5	7

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Banker's Algorithm – Checking requests

`Request[i]` for process P_i

1. If `Request[i] <= Need[i]`, go to step 2.
   Otherwise, error (maximum exceeded).
2. If `Request[i] <= Available`, go to step 3.
   Otherwise, P_i must wait (resources not available).
3. Pretend to fulll request:

```
Available := Available - Request[i]
Allocation[i] := Allocation[i] + Request[i]
Need[i] := Need[i] - Request[i]
```

   Approve request if resulting state is safe.
   Otherwise, restore state and P_i has to wait.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Banker's Algorithm – Checking requests

Allocation			
	R_0	R_1	R_2
P_0	0	1	0
P_1	2	0	0
P_2	3	0	2
P_3	2	1	1
P_4	0	0	2

Need			
	R_0	R_1	R_2
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Available		
R_0	R_1	R_2
3	3	2

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- Assume request `(1 0 2)` by P_1

Banker's Algorithm – Checking requests

Allocation			
	R_0	R_1	R_2
P_0	0	1	0
P_1	2	0	0
P_2	3	0	2
P_3	2	1	1
P_4	0	0	2

Need			
	R_0	R_1	R_2
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Available		
R_0	R_1	R_2
3	3	2

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

- Assume request `(1 0 2)` by P_1

Request can be served

Allocation			
	R_0	R_1	R_2
P_0	0	1	0
P_1	3	0	2
P_2	3	0	2
P_3	2	1	1
P_4	0	0	2

Need			
	R_0	R_1	R_2
P_0	7	4	3
P_1	0	2	0
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Available		
R_0	R_1	R_2
2	3	0

Banker's Algorithm – Checking requests

Allocation			
	R_0	R_1	R_2
P_0	0	1	0
P_1	3	0	2
P_2	3	0	2
P_3	2	1	1
P_4	0	0	2

Need			
	R_0	R_1	R_2
P_0	7	4	3
P_1	0	2	0
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Available		
R_0	R_1	R_2
2	3	0

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

- Assume request `(3 3 0)` by P_4

  Request cannot be granted (insufficient resources)

- Assume request `(0 2 0)` by P_0

  Request cannot be granted (would result in unsafe state)

Summary

Deadlocks

- Deadlock conditions:Assignment Project Exam Help
  - Mutual exclusion
  - Hold-and-wait
  - No preemption
  - Circular wait

https://powcoder.com

Add WeChat powcoder

- Methods for handling deadlocks
  - Deadlock prevention
  - Deadlock avoidance
  - Deadlock detection and recovery (next lecture)

Read

- Tanenbaum & Bos., Modern Operating Systems
  - Chapter 6
- Silberschatz et al., Operating System Concepts
  - Chapter 7

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Next Lecture

- Introduction
- Operating System Architectures
- Processes
- Threads - Programming
- Process Scheduling - Evaluation
- Process Synchronisation
- **Deadlocks (continued)**
- Memory Management
- File Systems
- Input / Output
- Security and Virtualisation

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder