CHAPTER 3
DEADLOCKS
Lecture 13

Chapter Contents

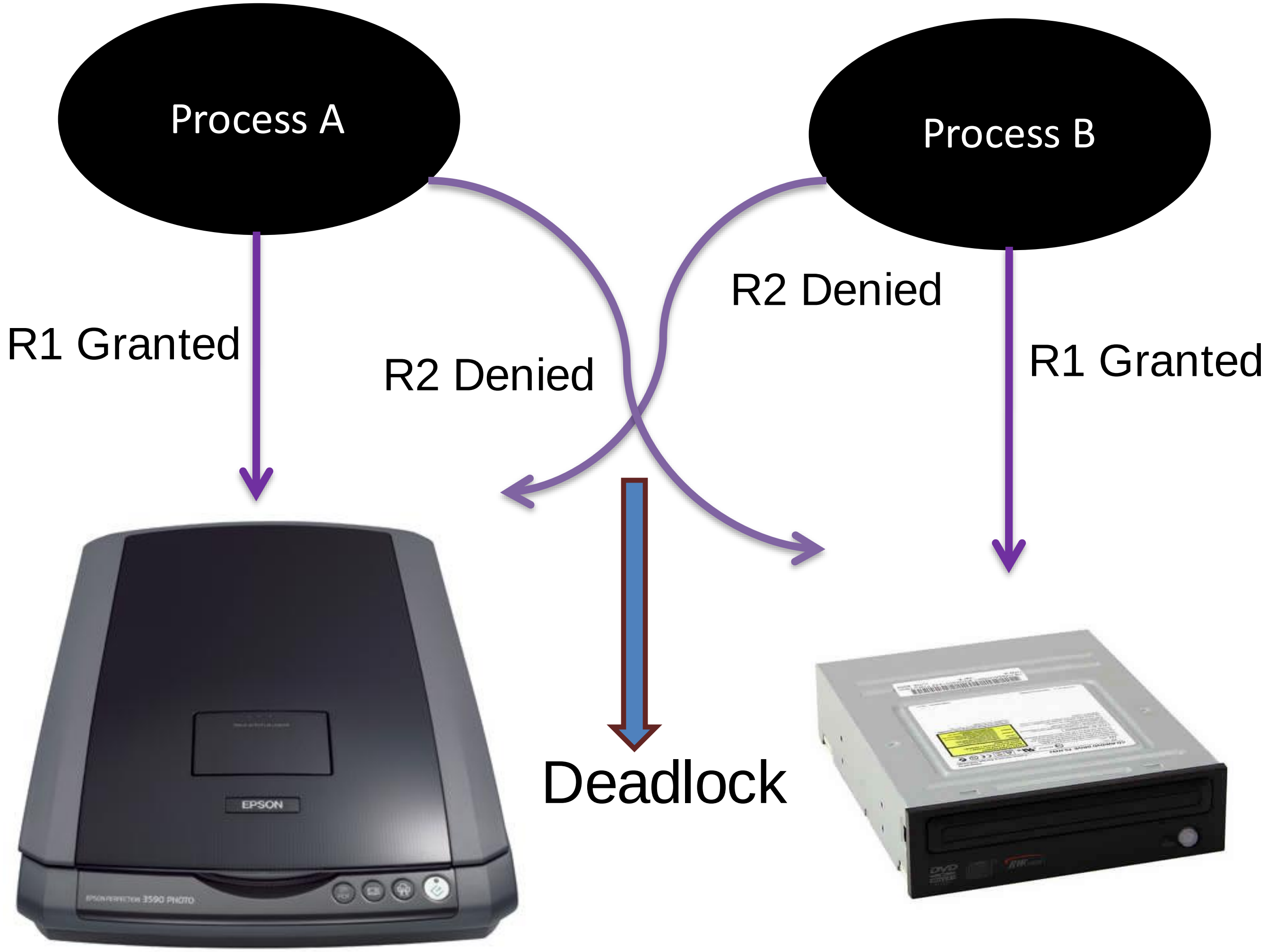
Deadlock-Example

Two or more busses are interacting, they get themselves into a stalemate situation they cannot get out off.

Deadlock-Example (processes & Devices)

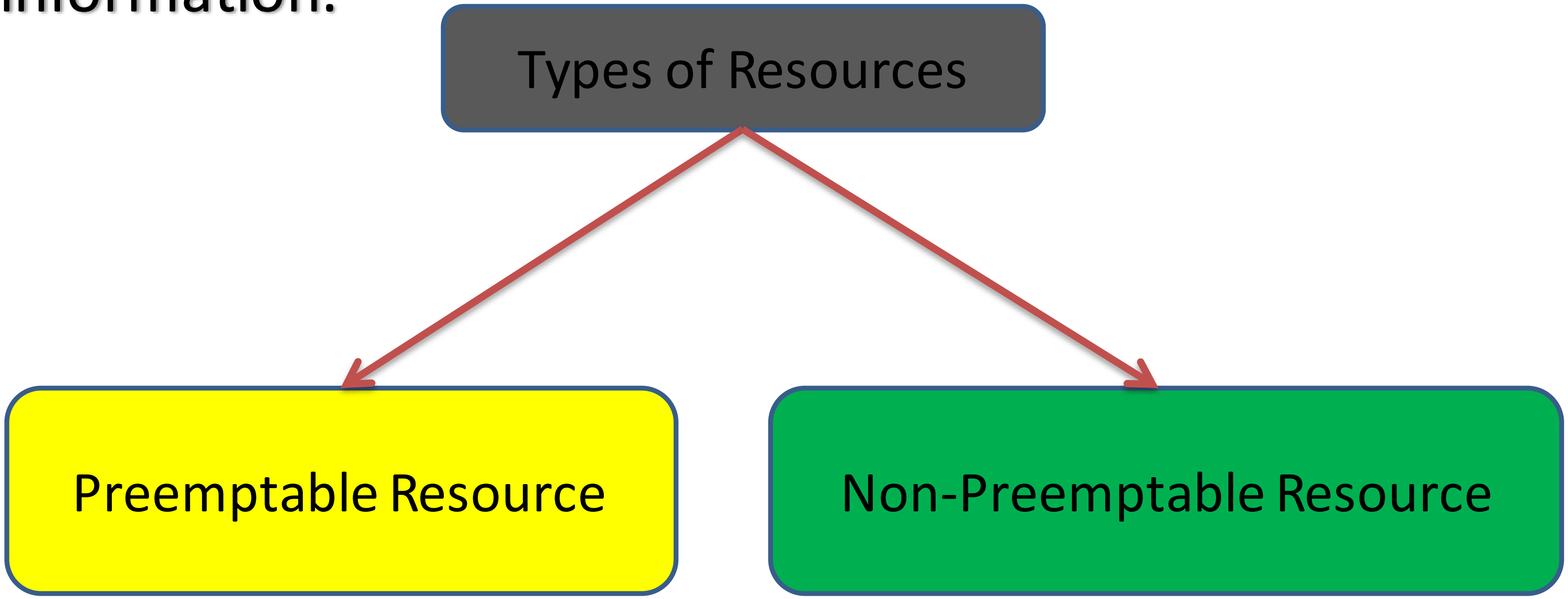
Two processes each want to write a scanned document on the CD. Process A request permission to use the scanner and is granted it. Process B is request the CD writer and is also granted it. Now A asks for the CD writer, but the request is denied until B releases it. Unfortunately, instead of releasing CD writer, B asks for the scanner. At this stage both processes are blocked. This situation is called **DEADLOCK.**

Deadlock-Example (Processes & Devices)



Deadlock-Resources

- A resource is anything that can be used by only a single process at any instant of time.
- A resource can be the hardware device or a piece of information.



Deadlock-Types of Resources

Preemptable Resource

"It is one that can be taken away from the process owing it."

Main memory is an example of preemptable resource.

Non-Preemptable Resource

"It is one that cannot be taken away from the process owing it."

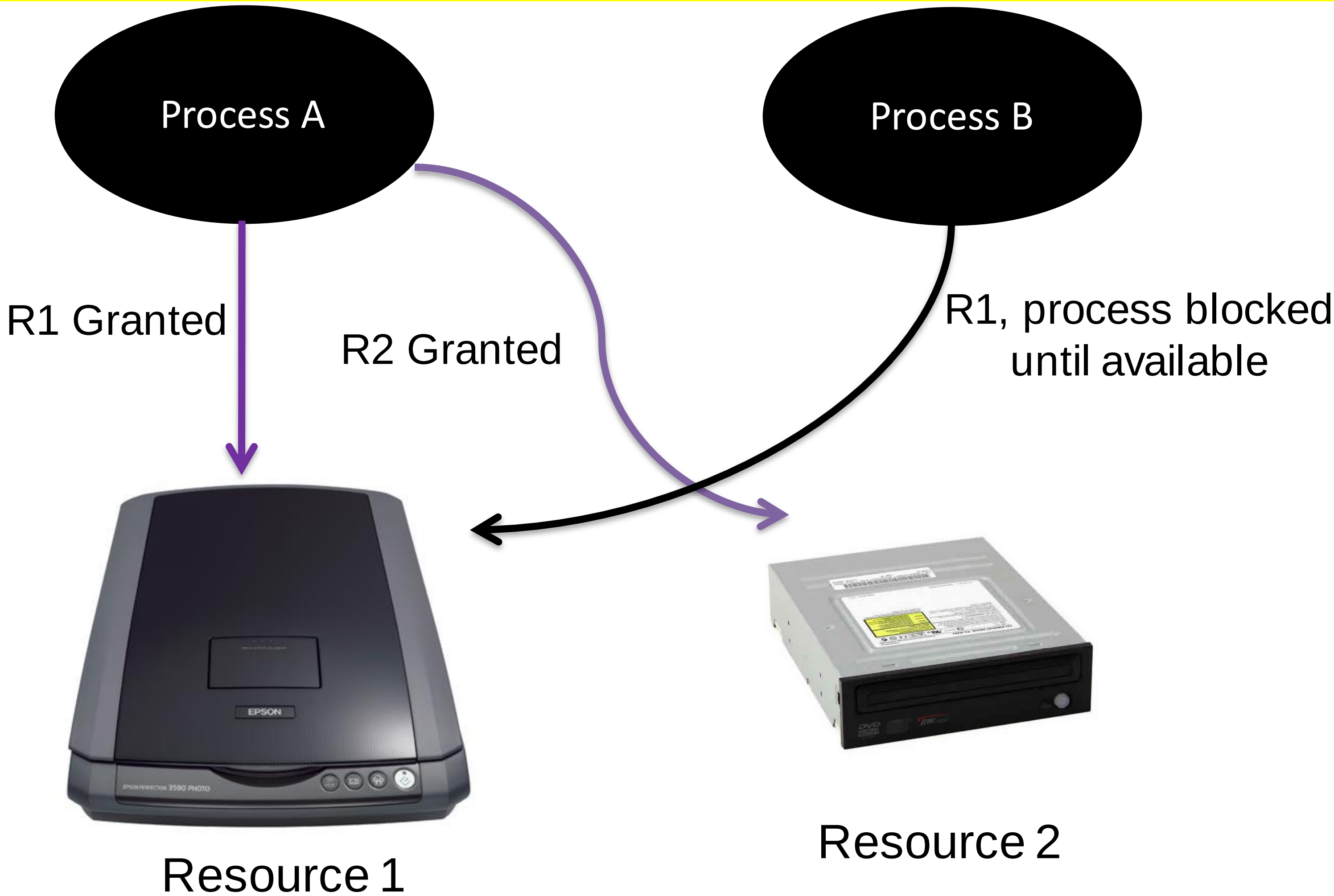
CD-ROM, Tape Drives, Printers etc are the examples of non-preemptable resources.

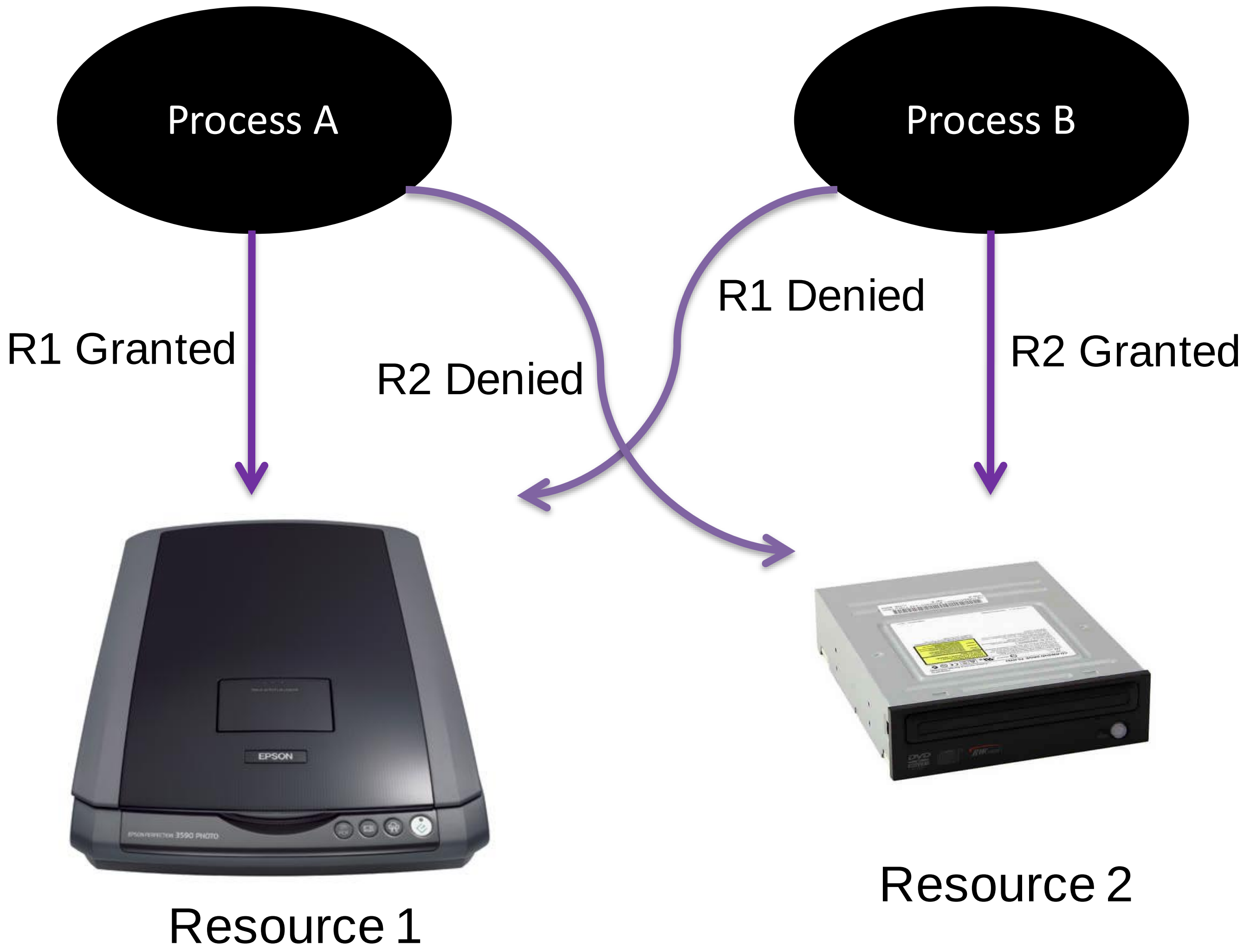
Deadlock-Types of Resources

Sequence of events required to use a resource

1- Request the resource
2- Use the resource
3- Release the resource

Deadlock-Resource Acquisition




Process A
Process B
R1 Granted
R2 Denied
R1 Denied
R2 Granted
EPSON
Resource 1
Resource 2

Definition

“Two or more processes are interacting, they get themselves into a stalemate situation they cannot get out off.”

No action can be taken

Introduction to Deadlocks-Conditions for Deadlock

1- Mutual Exclusion Condition

“Only one process at a time can use the resource.”

If other process requests that resource, the requesting process must be blocked until the resource has been released.

2- Hold and Wait Condition

“Processes currently holding resources can request new resources while waiting for that resource which is held by other process.”

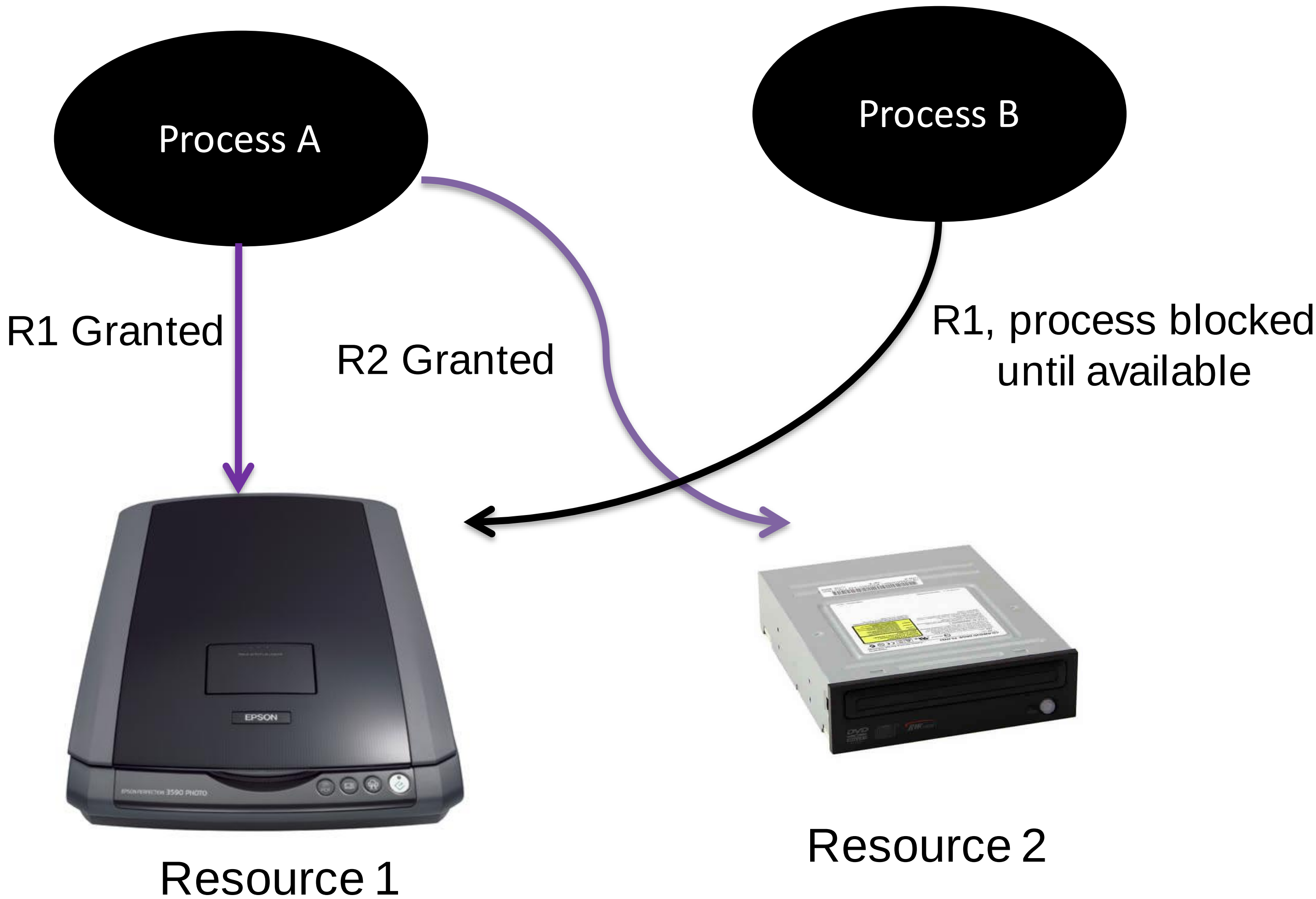
3- Non-Preemptive Condition

“Resources already allocated to a process cannot be taken away.”

4- Circular Wait Condition

“There must be a circular chain of two or more processes, each is waiting for the resource held by the next process.”

Which condition of deadlock is this?



Which condition of deadlock is this?

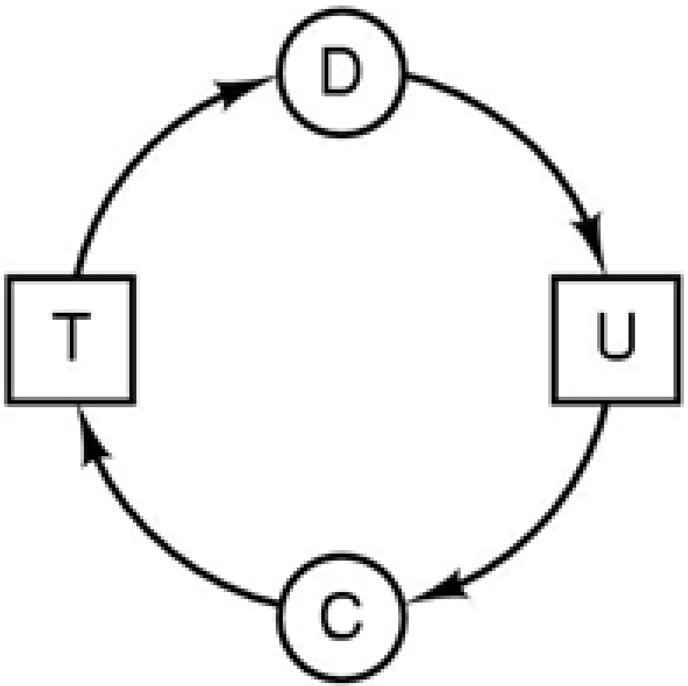
Process A

Process B

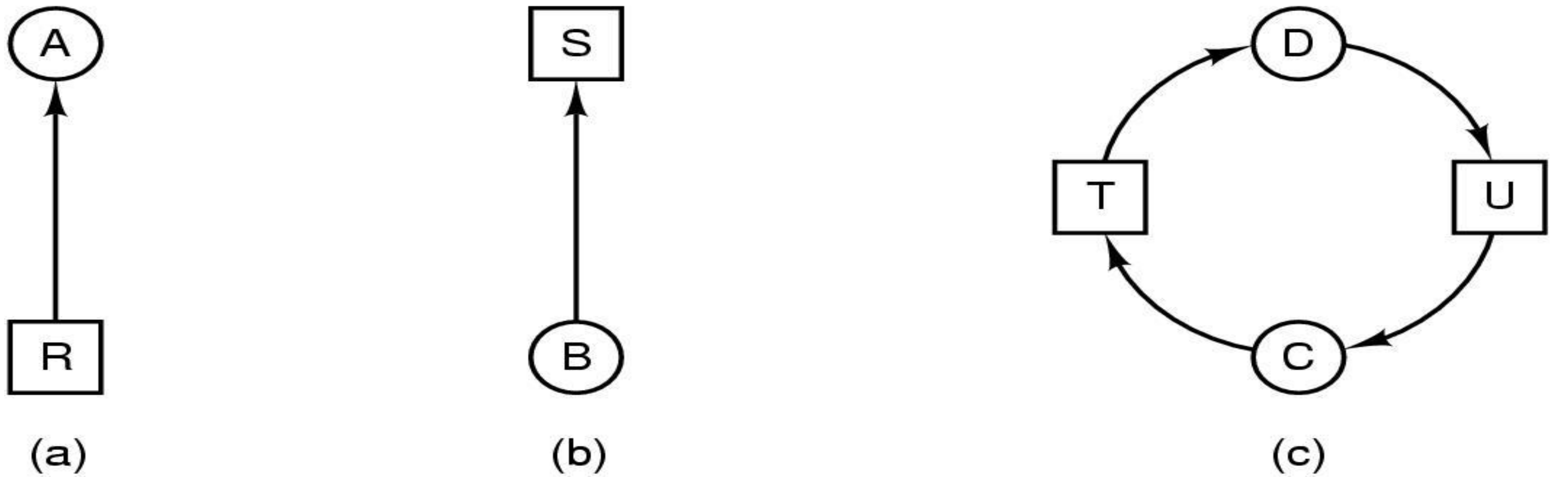
Resource cannot taken away

Resource cannot taken away

Which condition of deadlock is this?



Resource Allocation Graph

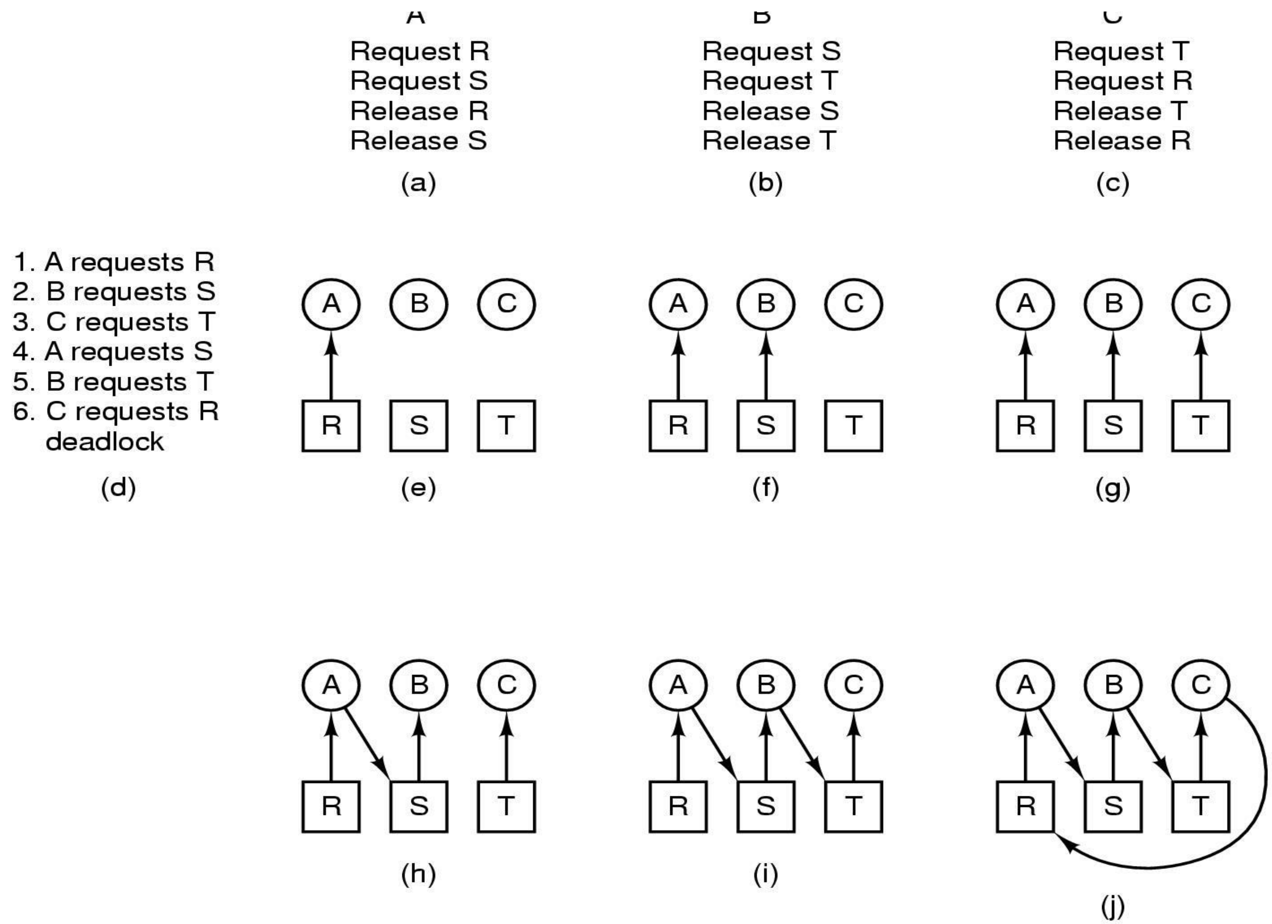


Resource R assigned to process A
Process B is requesting/waiting for resource S
Process C and D are in deadlock over resources T and U

Introduction to Deadlocks-Deadlock Modeling

Class Work

1-Draw a circular chain of picture (j).

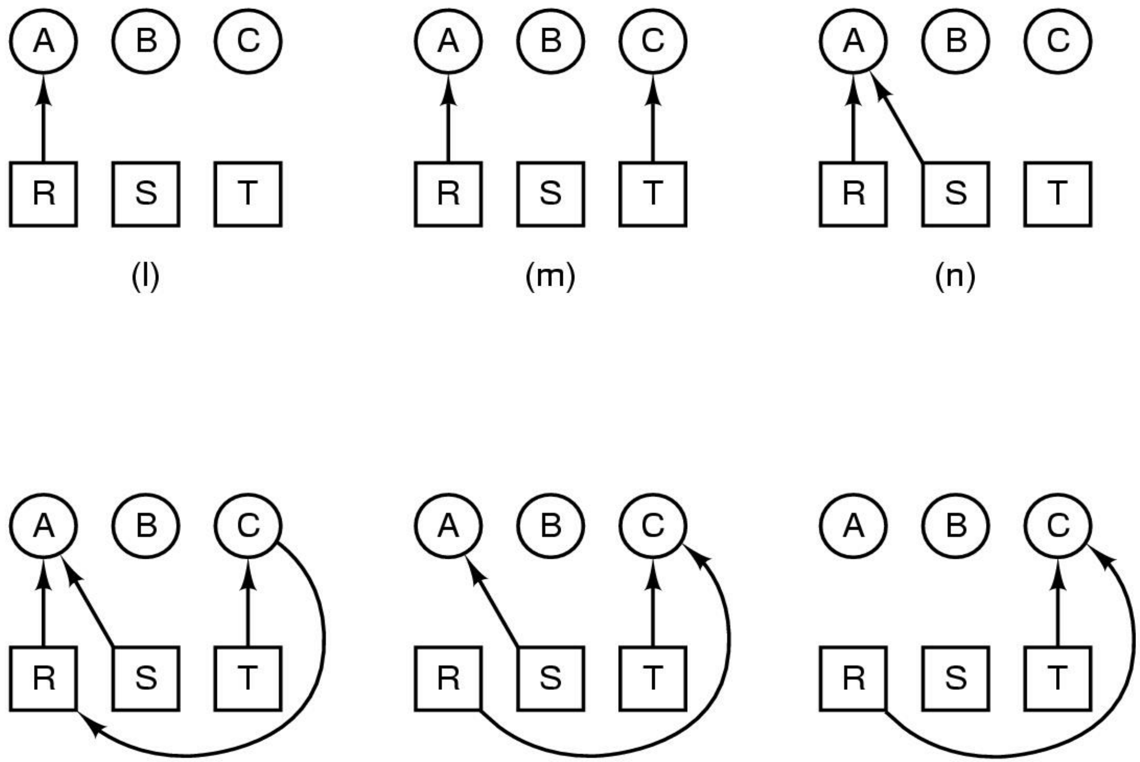


Introduction to Deadlocks-Deadlock Modeling

1. A requests R
2. C requests T
3. A requests S
4. C requests R
5. A releases R
6. A releases S

no deadlock

(k)



(l) (m) (n)

A visual (mathematical) way to determine if a deadlock has, or may occur.

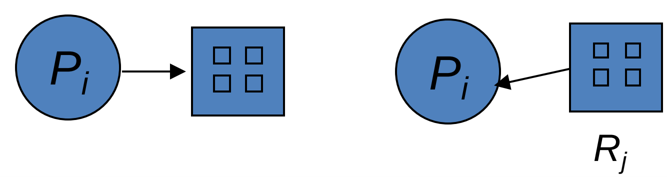
G = (V, E) The graph contains nodes and edges.

V Nodes consist of processes = { P1, P2, P3, ...} and resource types { R1, R2, ...}

E Edges are (Pi, Rj) or (Ri, Pj)

An arrow from the **process** to **resource** indicates the process is **requesting** the resource. An arrow from **resource** to **process** shows an instance of the resource has been **allocated** to the process.

Process is a circle, resource type is square; dots represent number of instances of resource in type. Request points to square, assignment comes from dot.



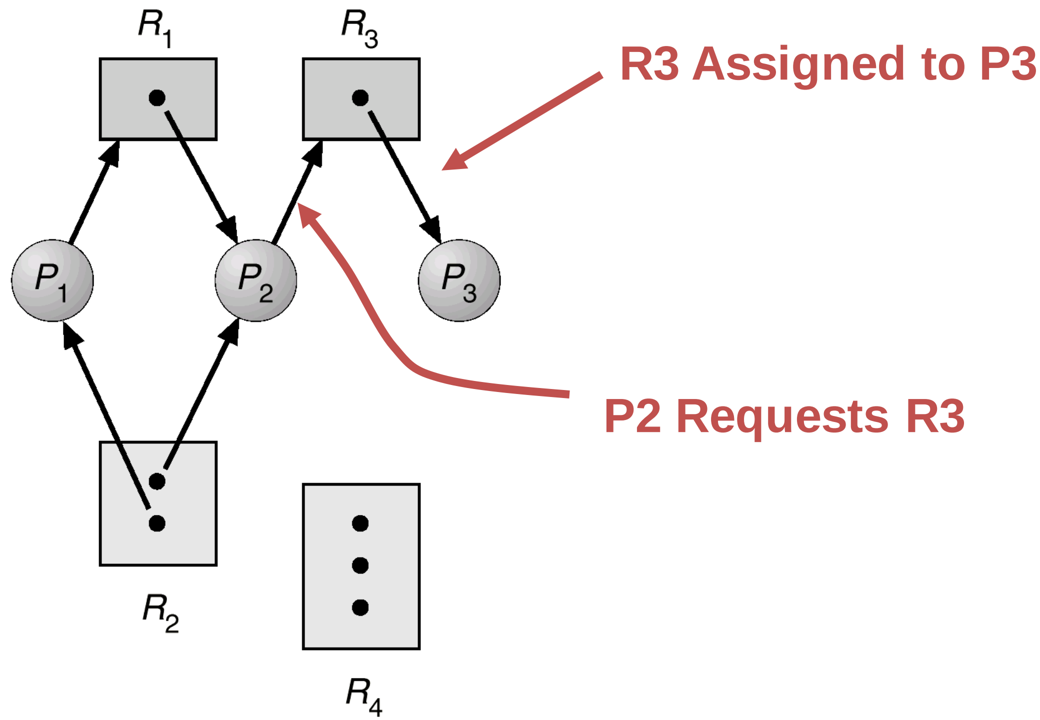
If the graph contains no cycles, then no process is deadlocked.

If there is a cycle, then:

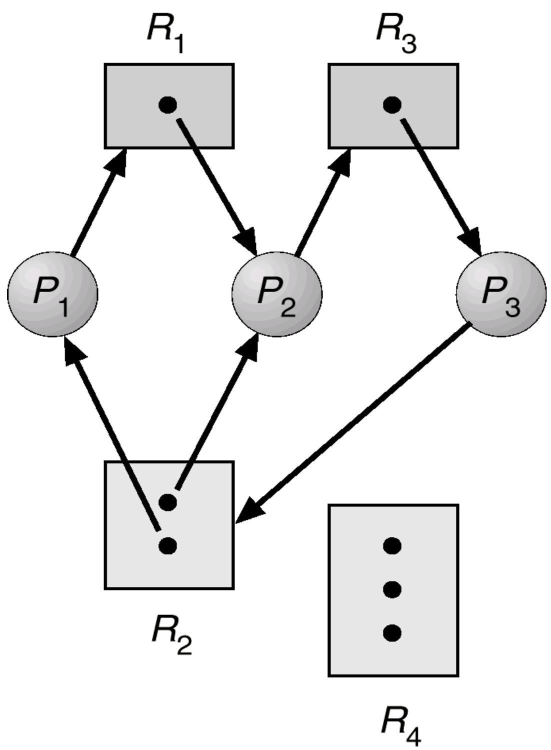
a) If resource types have multiple instances, then deadlock MAY exist.

b) If each resource type has 1 instance, then deadlock has occurred.

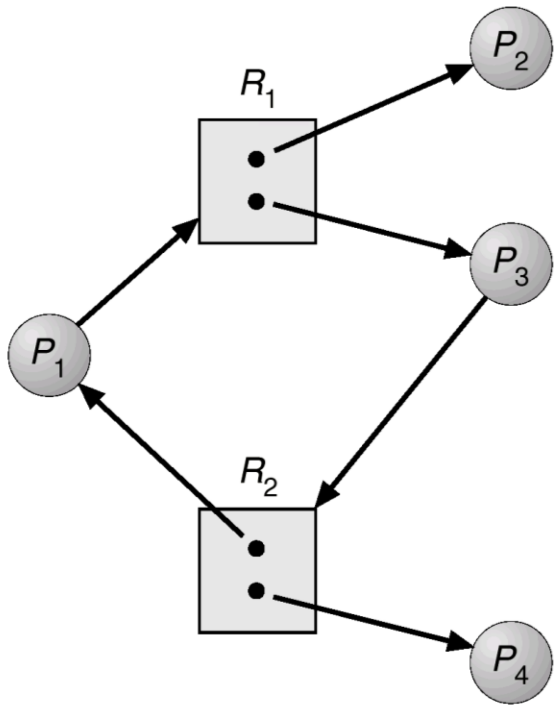
Resource allocation graph



Resource allocation graph
with a deadlock.



Resource allocation graph
with a cycle but no deadlock.



Deadlock Avoidance Algorithm
Safe and Unsafe States

At any instant of time, there is a current state consisting of E, A, C and R.

$$E = \begin{array}{cccc} \text{Tape drives} & \text{Plotters} & \text{Scanners} & \text{CD Roms} \\ (\,4 & 2 & 3 & 1\,) \end{array} \qquad A = \begin{array}{cccc} \text{Tape drives} & \text{Plotters} & \text{Scanners} & \text{CD Roms} \\ (\,2 & 1 & 0 & 0\,) \end{array}$$

Current allocation matrix

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

Request matrix

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

Deadlock Avoidance Algorithm
Safe and Unsafe States

	Has	Max
A	3	9
B	2	4
C	2	7

Free: 3
(a)

	Has	Max
A	3	9
B	4	4
C	2	7

Free: 1
(b)

	Has	Max
A	3	9
B	0	–
C	2	7

Free: 5
(c)

	Has	Max
A	3	9
B	0	–
C	7	7

Free: 0
(d)

	Has	Max
A	3	9
B	0	–
C	0	–

Free: 7
(e)

A state is said to be **Safe** if it is not deadlocked and there is some scheduling order in which every process can run to completion.

OR

Safe state is that in which system can guarantee that all processes will finish.

Deadlock Avoidance Algorithm
Safe and Unsafe States

	Has	Max
A	3	9
B	2	4
C	2	7

Free: 3
(a)

	Has	Max
A	4	9
B	2	4
C	2	7

Free: 2
(b)

	Has	Max
A	4	9
B	4	4
C	2	7

Free: 0
(c)

	Has	Max
A	4	9
B	—	—
C	2	7

Free: 4
(d)

A state is said to be **Un-safe** if it is deadlocked and there is some scheduling order in which every process cannot run to completion.

OR

Un-safe state is that in which system cannot guarantee that all processes will finish.

Deadlock-Avoidance

- **Banker's Algorithm for a Single Resource**

  Banker's algorithm considers each request as it occurs, and see if granting it leads to a safe state, if it does, the request is granted otherwise denied.

Banker’s Algorithm for a Single Resource

- **Banker’s algorithm is used to check which state is safe and which one is un-safe.**
- This state is safe because with 2 resources left the process can delay any requests except C’s, thus letting C finish and release all four of its resources. With four resources in hand, the process can let either D or B have the necessary resources and so on.

	Has	Max
A	0	6
B	0	5
C	0	4
D	0	7

Free: 10

	Has	Max
A	1	6
B	1	5
C	2	4
D	4	7

Free: 2

	Has	Max
A	1	6
B	2	5
C	2	4
D	4	7

Free: 1

Banker's Algorithm for Multiple Resources

Process	Tape drives	Plotters	Scanners	CD ROMs
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

Resources assigned

Process	Tape drives	Plotters	Scanners	CD ROMs
A	1	1	0	0
B	0	1	1	2
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0

Resources still needed

E = (6342)
P = (5322)
A = (1020)

P=Possessed resources
E=Existing resources (total resources)
A=Available resources
Need[i][j] = Max[i][j] - Allocation [i][j]

Banker’s Algorithm for Multiple Resources

ALGORITHM:

1. If (Need[i] ≤ Available)

   Then Available= Available + Allocation[i];

2. *if no such row exists, we are deadlocked because no process can acquire the resources it needs to run to completion.*

- *If there's more than one such row, just pick one.*

Repeat 1 and 2 until all processes are either virtually terminated (safe state), or a deadlock is detected (unsafe state).

CLASS EXERCISE-1
Find safe sequence?

Total=[10,5,7]

5 processes P_0 through P_4

3 resource types *A* (10 units), *B* (5 units), and *C* (7 units).

	Allocation	***Max***	***Available***
	A B C	***A B C***	***A B C***
P_0	**0 1 0**	**7 5 3**	**3 3 2**
P_1	**2 0 0**	**3 2 2**	
P_2	**3 0 2**	**9 0 2**	
P_3	**2 1 1**	**2 2 2**	
P_4	**0 0 2**	**4 3 3**	

SOLUTION EXERCISE-1

Need = Max – Allocation

	Need
	A B C
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

P_1, P_3, P_4, P_2, P_0 satisfies safety criteria.

If (Need[i] ≤ Available)
Then Available= Available + Allocation[i];

WORKING: Available

Available=	(3,3,2)	
	+(2,0,0)	Allocation[p1]
Now available=	(5,3,2)	
	+(2,1,1)	Allocation[p3]
Now available=	(7,4,3)	
	+(0,0,2)	Allocation[p4]
Now available=	(7,4,5)	
	+(3,0,2)	Allocation[p2]
Now available=	(10,4,7)	
	+(0,1,0)	Allocation[p0]
Now available=	(10,5,7)	

As available== TOTAL --> STOP with safe sequence

Class Exercise: 2

1- Total Matrix?
2- MAX matrix for each process
3- If any safe sequence exist? Find.

Allocation

	A	B	C	D
P_0	2	1	2	2
P_1	4	0	2	1
P_2	1	3	2	1
P_3	1	1	1	0
P_4	2	0	2	1

Request

	A	B	C	D
P_0	2	1	4	4
P_1	2	1	2	2
P_2	1	2	2	2
P_3	2	0	0	1
P_4	1	1	1	1

	A	B	C	D
Available: (Work)	3	0	1	1

Sample Question-1 with sol

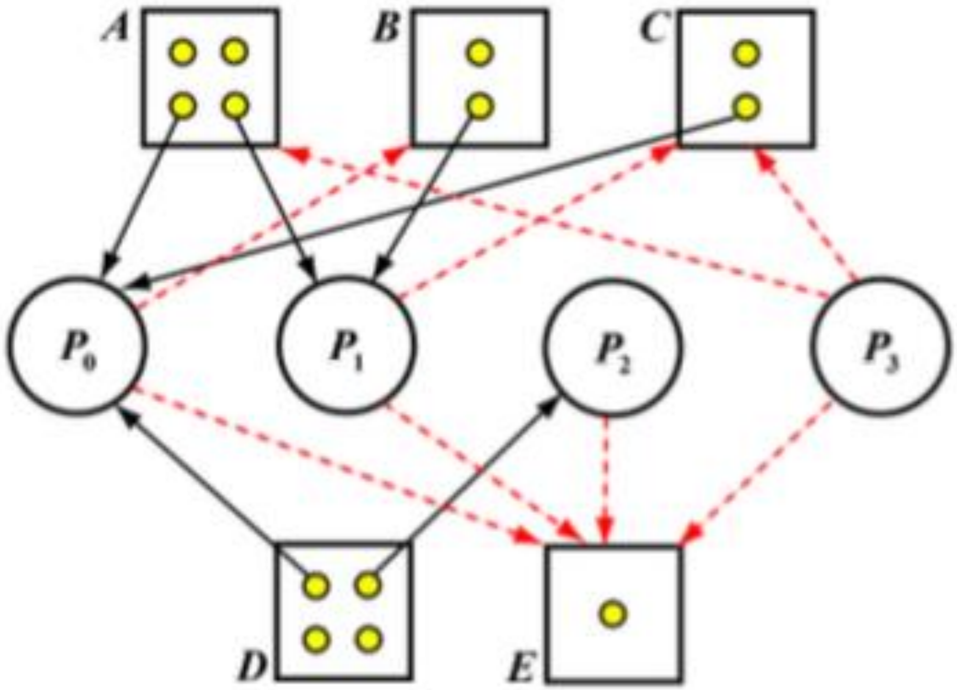
Consider the following snapshot of a system in which five resources A, B, C, D and E are available. The system contains a total of 2 instances of A, 1 of resource B, 1 of resource C, 2 resource D and 1 of resource E.

	Allocation					Request					Available				
	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
P_0	1	0	1	1	0	0	1	0	0	1	2	1	1	2	1
P_1	1	1	0	0	0	0	0	1	0	1					
P_2	0	0	0	1	0	0	0	0	0	1					
P_3	0	0	0	0	0	1	0	1	0	1					

Do the following problems:

- Convert this matrix representation to a resource allocation graph.
- Use the deadlock detection algorithm to determine whether the system contains a deadlock. Which processes are involved in the deadlock?
- While you are use the deadlock detection algorithm, add and remove directed edges of the re- source allocation graph.

1.Answer: The corresponding resource allocation graph is shown below:

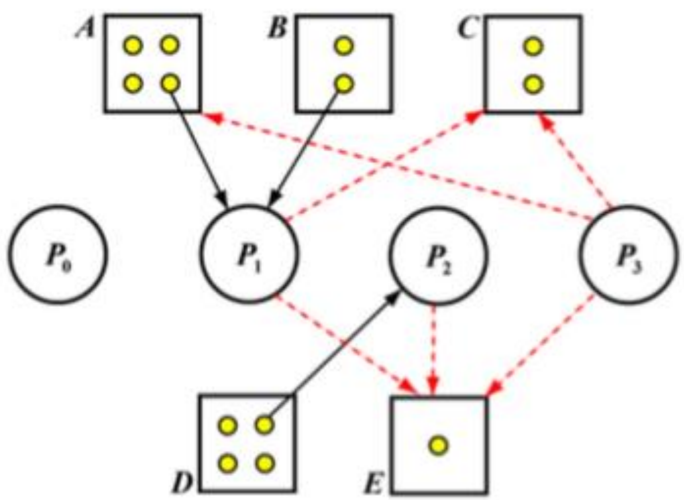


1.Note that red dashed-lined arrows are used to indicate unsatisfied “requests”

Currently, because P0′s Request=[0,1,0,0,1]≤Available=[2,1,1,2,1],we run P0 and reclaim its Allocation, and the new Allocation = Old Allocation + Available = [1, 0, 1, 1, 0] + [2, 1, 1, 2, 1] = [3, 1, 2, 3, 1]. The new matrix representation becomes:

	Allocation					Request					Available				
	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
P_0											3	1	2	3	1
P_1	1	1	0	0	0	0	0	1	0	1					
P_2	0	0	0	1	0	0	0	0	0	1					
P_3	0	0	0	0	0	1	0	1	0	1					

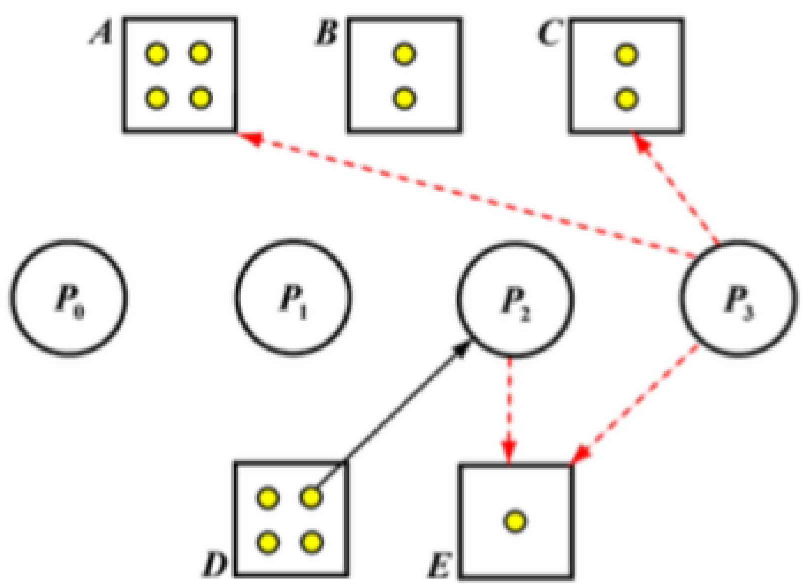
For the resource allocation graph, because all resources allocated to P0 are returned and P0 has no new request (so far), the new graph is obtained by removing all request edges and allocation edges as shown below:



Now, we have P1′s Request = [0,0,1,0,1] ≤ Available = [3,1,2,3,1]. We can run P1, and reclaim its Allocation=[1,1,0,0,0]. The new Available=[3,1,2,3,1]+P′s Allocation=[1,1,0,0,0]= [4, 2, 2, 3, 1]. As a result, the new matrix representation is:

	Allocation					*Request*					*Available*				
	A	*B*	*C*	*D*	*E*	*A*	*B*	*C*	*D*	*E*	*A*	*B*	*C*	*D*	*E*
P_0											4	2	2	3	1
P_1															
P_2	0	0	0	1	0	0	0	0	0	1					
P_3	0	0	0	0	0	1	0	1	0	1					

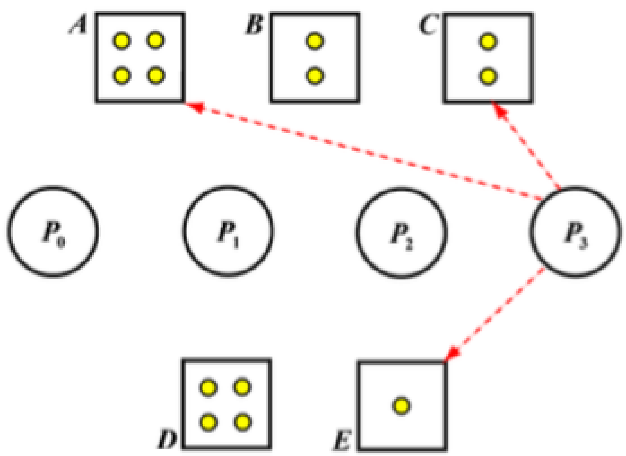
By removing all edges of P1 yields a new resource allocation graph:



Next, we run P2 and the new matrix representation and resource allocation graph are:

	Allocation					Request					Available				
	A	*B*	*C*	*D*	*E*	*A*	*B*	*C*	*D*	*E*	*A*	*B*	*C*	*D*	*E*
P_0											4	2	2	4	1
P_1															
P_2															
P_3	0	0	0	0	0	1	0	1	0	1					

Removing all edges from and to P1 yields a new resource allocation graph:

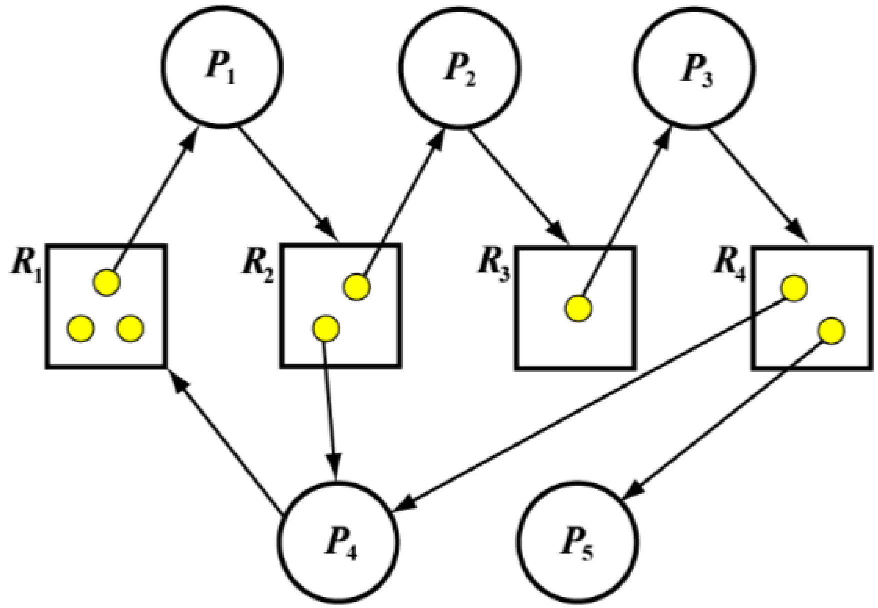


Finally, P3 can run and return all of its resources. Because all involved processes are finished, this system does not have a deadlock.

Safe Sequence= <P0,P1,P2,P3>

Sample Question-2 with sol

Consider the following resource allocation graph.



Do the following problems:

(a) Convert it to the matrix representation (i.e., Allocation, request and Available).

(b) Do a step-by-step execution of the deadlock detection algorithm. For each step, add and remove the directed edges, and redraw the resource allocation graph.

(c) Is there a deadlock? If there is a deadlock, which processes are involved?

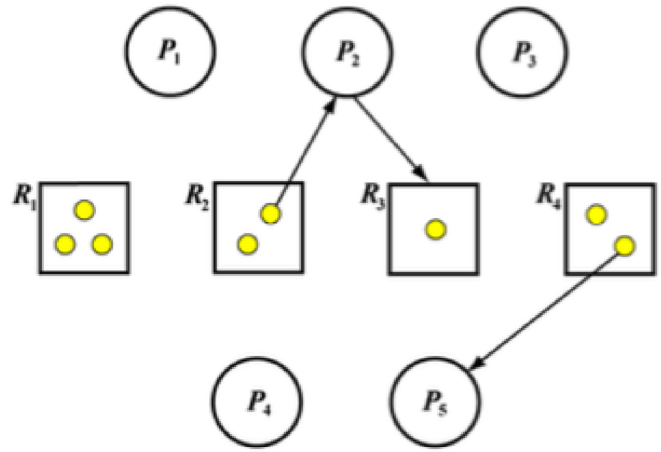
Answer: The matrix representation of the given resource allocation graph is shown below:

	Allocation				*Request*				*Available*			
	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_1	1	0	0	0	0	1	0	0	2	0	0	0
P_2	0	1	0	0	0	0	1	0				
P_3	0	0	1	0	0	0	0	1				
P_4	0	1	0	1	1	0	0	0				
P_5	0	0	0	1	0	0	0	0				

Because P4′sRequest=[1,0,0,0]≤Available=[2,0,0,0],P4 runs and returns is Allocation=[0,1,0,1] making the new Available = [2, 0, 0, 0] + [0, 1, 0, 1] = [2, 1, 0, 1]. The matrix representation becomes:

	Allocation				*Request*				*Available*			
	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_1									3	1	0	1
P_2	0	1	0	0	0	0	1	0				
P_3	0	0	1	0	0	0	0	1				
P_4												
P_5	0	0	0	1	0	0	0	0				

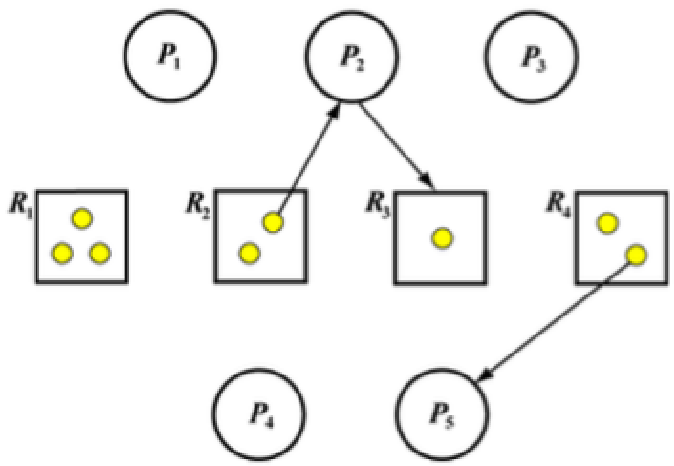
Here is the corresponding resource allocation graph:



The next process is P3 because P3'sRequest=[0,0,0,1]≤Available=[3,1,0,1].After P3 finishes its work, its Allocation = [0, 0, 1, 0] is returned to Available = [3, 1, 0, 1] + [0, 0, 1, 0] = [3, 1, 1, 1]:

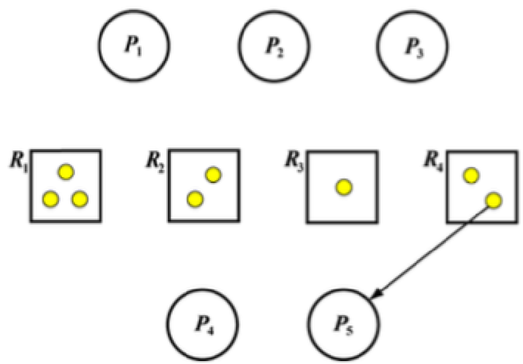
	Allocation				*Request*				*Available*			
	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_1									3	1	1	1
P_2	0	1	0	0	0	0	1	0				
P_3												
P_4												
P_5	0	0	0	1	0	0	0	0				

The resource allocation graph is shown below:



Now we can run P2 and the yields (i.e., matrix representation and resource allocation graph) are:

	Allocation				*Request*				*Available*			
	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_1									3	2	1	1
P_2												
P_3												
P_4												
P_5	0	0	0	1	0	0	0	0				

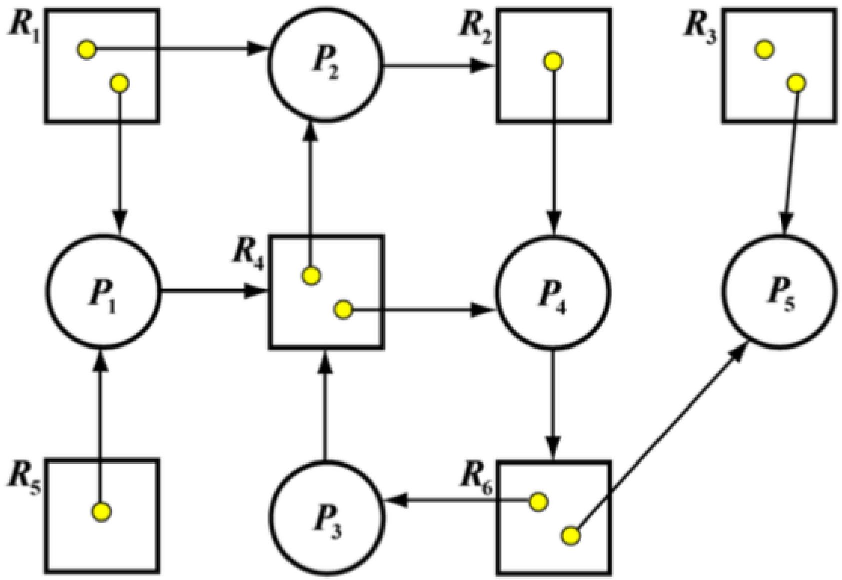


Finally, we can run P5 and all processes are done! **Note** that there are cycles P1 → R2 → P4 → R1 → P1 and P1 → R2 → P2 → R3 → P3 → R4 → P4 → R1 → P1. However, there is no deadlock.

SAFE SEQUENCE<P4,P1,P3,P2,P5>

EXERCISE QUESTION:

Consider the following resource allocation graph.



Do the following problems:

(a) Convert it to the matrix representation(i.e., Allocation, Request and Available).

(b) Do a step-by-step execution of the deadlock detection algorithm. For each step, add and remove the directed edges, and redraw the resource allocation graph.

(c) Is there a deadlock? If there is a deadlock, which processes are involved?

EXERCISE QUESTION:

Consider the following snapshot of a system in which four resources A, B, C and D area available. The system contains a total of 6 instances of A, 4 of resource B, 4 of resource C, 2 resource D.

	Allocation				*Max*				*Need*				*Available*			
	A	*B*	*C*	*D*	*A*	*B*	*C*	*D*	*A*	*B*	*C*	*D*	*A*	*B*	*C*	*D*
P_0	2	0	1	1	3	2	1	1					6	4	4	2
P_1	1	1	0	0	1	2	0	2								
P_2	1	0	1	0	3	2	1	0								
P_3	0	1	0	1	2	1	0	1								

- Convert this matrix representation to a resource allocation graph.

Do the following problems using the banker's algorithm:

• Compute what each process might still request and fill this in under the column Need. • Is the system in a safe state? Why or why not?

• Is the system deadlocked? Why or why not?

• If a request from P3 arrives for (2, 1, 0, 0), can the request be granted immediately?