Parul® University
Vadodara, Gujarat

NAAC GRADE A++

Information and Communication Technology

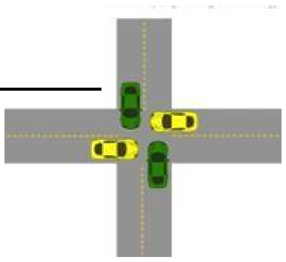
Deadlock

Study Guide

Assistant Prof. Sumersing Patil
CSE-AI&DS, PIET
Parul University

UNIT-4: DEADLOCKS

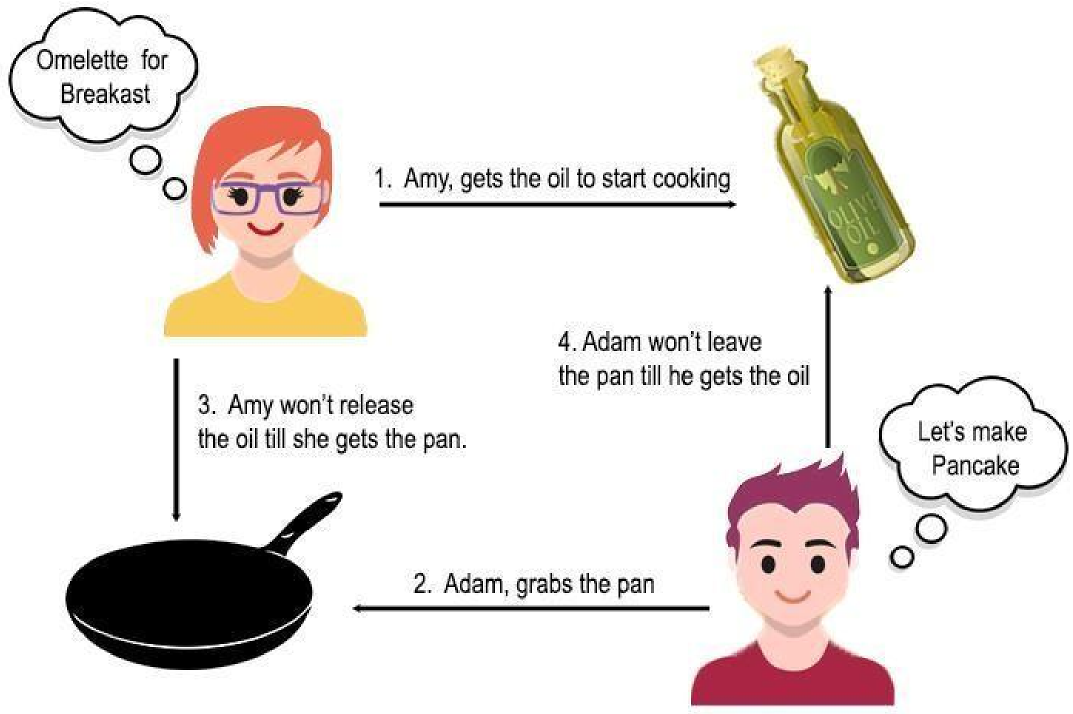
4	**UNIT-4**: **DEADLOCKS**: Definition, Necessary and sufficient conditions for Deadlock, Deadlock Prevention, Deadlock Avoidance: Banker's algorithm, Deadlock detection and Recovery.	10%	5

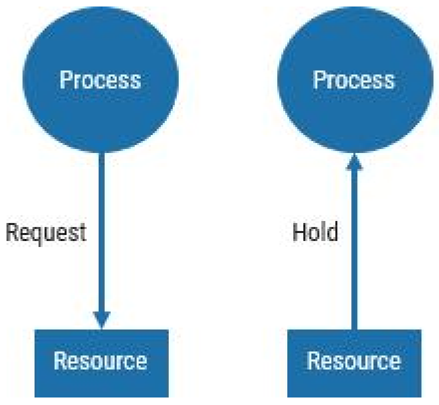


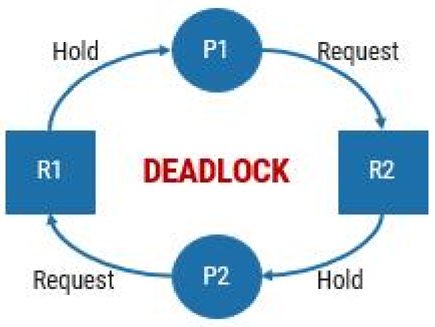
- Basic concepts of
- Deadlock
- Deadlock ignorance
    - Ostrich algorithm
- Deadlock detection and recovery
- Deadlock avoidance
    - Banker's algorithm
- Deadlock prevention

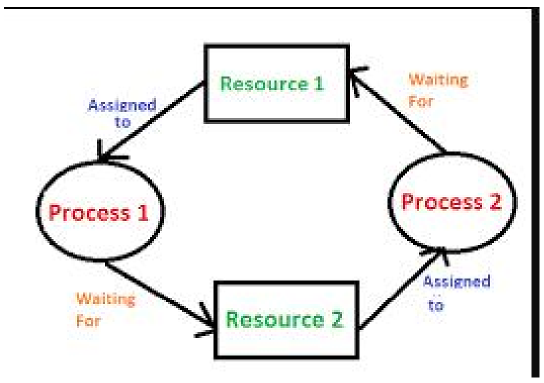
What is Deadlock?

- 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**.
- Deadlocks are a **set of blocked processes each holding a resource and waiting to acquire a resource held by another process**.
- All processes keep waiting for each other to complete and none get executed.
- Resources are blocked by the process.
- Necessary conditions are mutual exclusion, hold and wait, no preemption, circular wait.
- Also known as circular wait.
- It can be prevented by avoiding the necessary conditions for deadlock.


Omelette for Breakast
1. Amy, gets the oil to start cooking
4. Adam won't leave the pan till he gets the oil
3. Amy won't release the oil till she gets the pan.
Let's make Pancake
2. Adam, grabs the pan


Process
Request
Resource
Process
Hold
Resource


Hold
P1
Request
R1
DEADLOCK
R2
Request
P2
Hold


Resource 1
Assigned to
Waiting For
Process 1
Process 2
Waiting For
Resource 2
Assigned to

Preemptable and non-preemptable resource

- Preemptable:- Preemptive resources are those **which can be taken away from a process without causing any ill effects** to the process.
    - Example:-Memory.
- Non-preemptable:- Non-pre-emptive resources are those **which cannot be taken away from the process without causing any ill effects** to the process.
    - Example:-CD-ROM (CD recorder), Printer.

Conditions that lead to deadlock (Deadlock characteristics)

1. Mutual exclusion
    - **Each resource is either** currently **assigned to exactly one process** or **is available**.
    - Only **one process at a time can use a resource.**
2. Hold and wait
    - Process currently holding resources granted earlier can **request more resources**.
3. No preemption
    - Previously granted resources **cannot be forcibly taken away** from process.
4. Circular wait
    - There must be a **circular chain of 2 or more processes**. Each process is waiting for resource that is held by next member of the chain.

- **All four of these conditions must be present for a deadlock to occur.**

Strategies for dealing with deadlock

1. Just **ignore** the problem
2. **Detection and recovery**.
    - Let deadlocks occur, detect them and take action.
3. Dynamic **avoidance** by careful resource allocation.
4. **Prevention**, by structurally negating (killing) one of the four required conditions.

Deadlock ignorance (Ostrich Algorithm)

- When **storm approaches**, an **ostrich puts his head in the sand (ground)** and **pretend (imagine) that there is no problem at all**.
- **Ignore**the **deadlock** and **pretend** that **deadlock never occur**.
- Reasonable if
    - deadlocks occur very rarely
    - difficult to detect
    - cost of prevention is high
- **UNIX** and **Windows** takes this approach

Deadlock detection and recovery

Deadlock detection for single resource (RAG - Resource Allocation Graph)

Deadlock detection for multiple resources

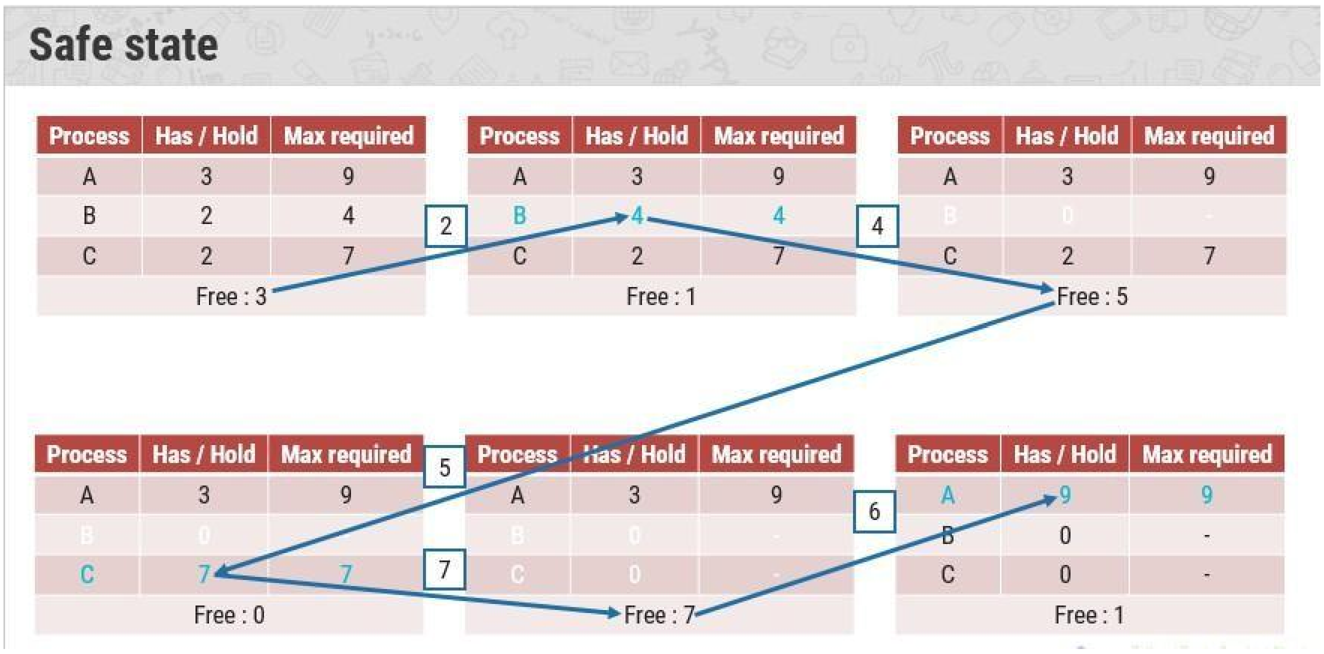
Deadlock avoidance (Banker's algorithm)

- Deadlock can be avoided by allocating resources carefully.
- Carefully analyze each resource request to see if it can be safely granted.
- Need an algorithm that can always avoid deadlock by making right choice all the time (Banker's algorithm).
- Banker's algorithm for single resource
- Banker's algorithm for multiple resource

Process	Has / Hold	Max required
A	3	9
B	2	4
C	2	7
Free : 3		

- 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 even if all of them suddenly request their maximum number of resources immediately.
- Total resources are 10
- 7 resources already allocated
- So there are 3 still free
- A need 6 resources more to complete it.
- B need 2 resources more to complete it.
- C need 5 resources more to complete it.

Safe state



Process	Has / Hold	Max required
A	3	9
B	2	4
C	2	7
	Free : 3	

Process	Has / Hold	Max required
A	3	9
B	4	4
C	2	7
	Free : 1	

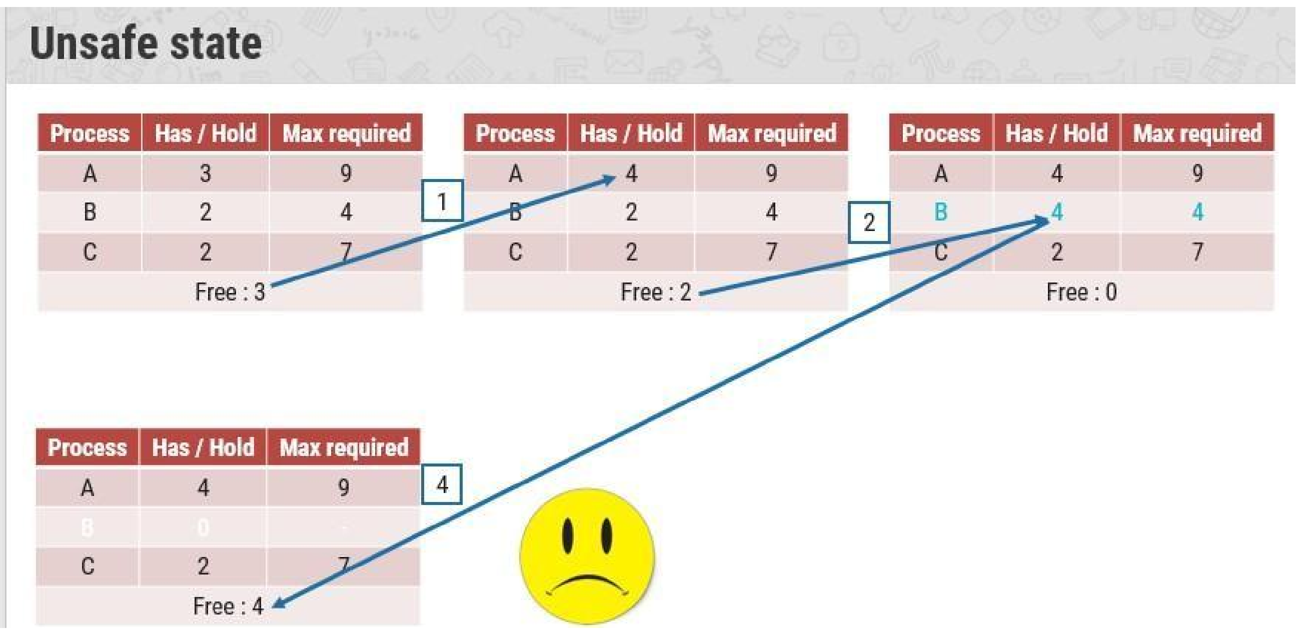
Process	Has / Hold	Max required
A	3	9
B	0	-
C	2	7
	Free : 5	

Process	Has / Hold	Max required
A	3	9
B	0	
C	7	7
	Free : 0	

Process	Has / Hold	Max required
A	3	9
B	0	-
C	0	-
	Free : 7	

Process	Has / Hold	Max required
A	9	9
B	0	-
C	0	-
	Free : 1	

Unsafe state



Process	Has / Hold	Max required
A	3	9
B	2	4
C	2	7
	Free : 3	

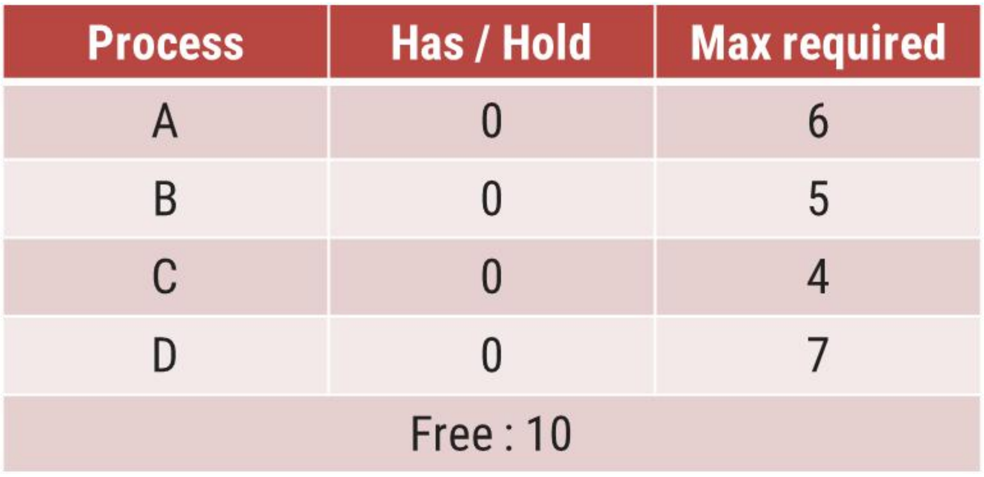
Process	Has / Hold	Max required
A	4	9
B	2	4
C	2	7
	Free : 2	

Process	Has / Hold	Max required
A	4	9
B	4	4
C	2	7
	Free : 0	

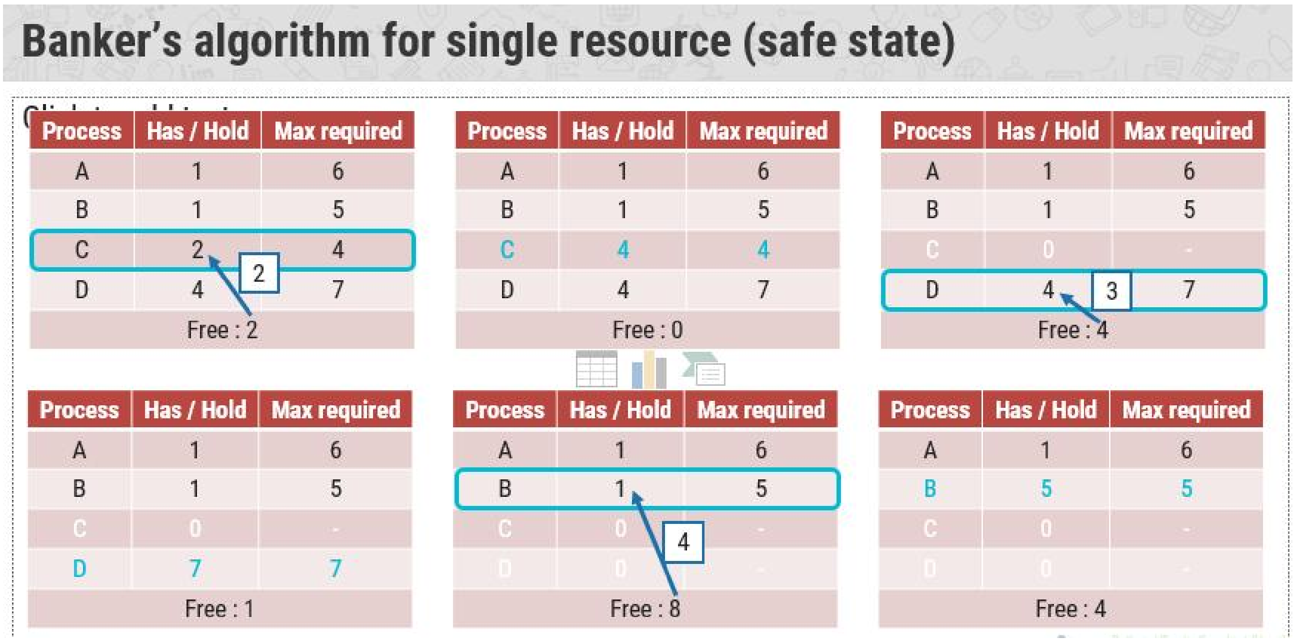
Process	Has / Hold	Max required
A	4	9
B	0	-
C	2	7
	Free : 4	

Banker’s algorithm for single resource

- What the algorithm does is check to see if granting the request leads to an unsafe state. If it does, the request is denied.
- If granting the request leads to a safe state, it is carried out.
- If we have situation as per figure
  - then it is safe state
  - because with 10 free units
  - one by one all customers can be served.



Process	Has / Hold	Max required
A	0	6
B	0	5
C	0	4
D	0	7
Free : 10		



Banker's algorithm for single resource (safe state)

Process	Has / Hold	Max required
A	1	6
B	1	5
C	2	4
D	4	7
Free : 2		

2

Process	Has / Hold	Max required
A	1	6
B	1	5
C	4	4
D	4	7
Free : 0		

Process	Has / Hold	Max required
A	1	6
B	1	5
C	0	-
D	4	7
Free : 4		

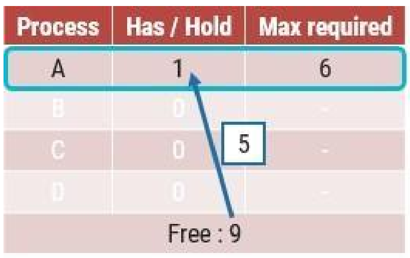
3

Process	Has / Hold	Max required
A	1	6
B	1	5
C	0	-
D	7	7
Free : 1		

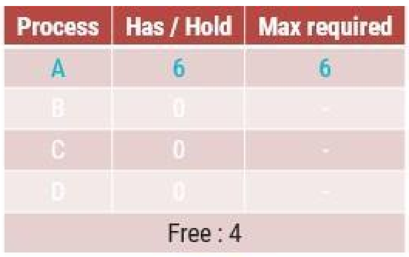
Process	Has / Hold	Max required
A	1	6
B	1	5
C	0	-
D	0	-
Free : 8		

4

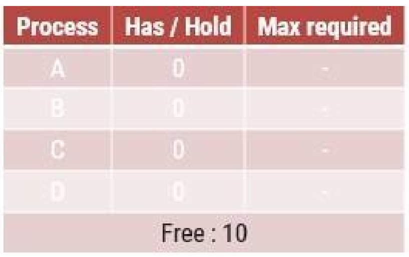
Process	Has / Hold	Max required
A	1	6
B	5	5
C	0	-
D	0	-
Free : 4		



Process	Has / Hold	Max required
A	1	6
B	0	-
C	0	-
D	0	-
Free : 9		



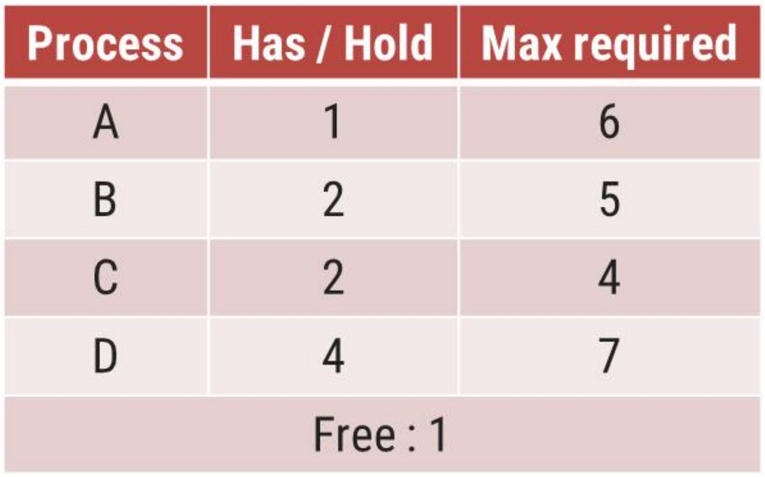
Process	Has / Hold	Max required
A	6	6
B	0	-
C	0	-
D	0	-
Free : 4		



Process	Has / Hold	Max required
A	0	-
B	0	-
C	0	-
D	0	-
Free : 10		

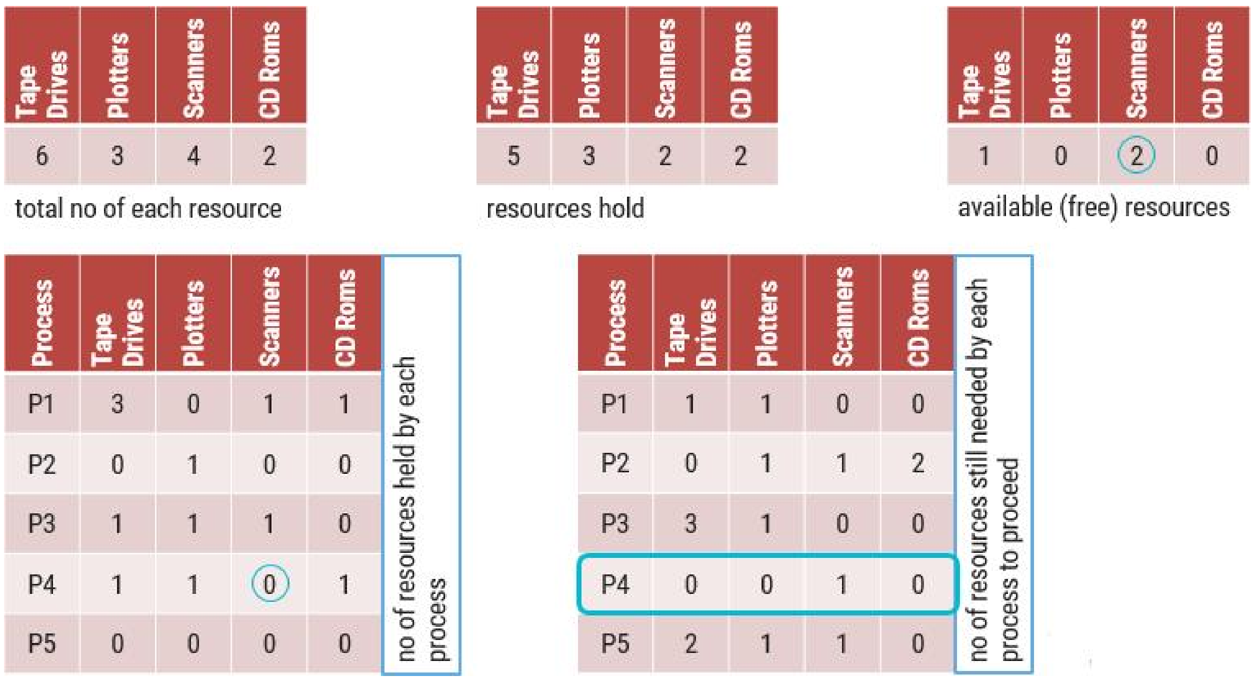
- The order of execution is C, D, B, A. So if we can find proper order of execution then there is no deadlock.

Banker's algorithm for single resource (unsafe state)



Process	Has / Hold	Max required
A	1	6
B	2	5
C	2	4
D	4	7
Free : 1		

Banker's algorithm for multiple resource



Tape Drives	Plotters	Scanners	CD Roms
6	3	4	2

total no of each resource

Tape Drives	Plotters	Scanners	CD Roms
5	3	2	2

resources hold

Tape Drives	Plotters	Scanners	CD Roms
1	0	2	0

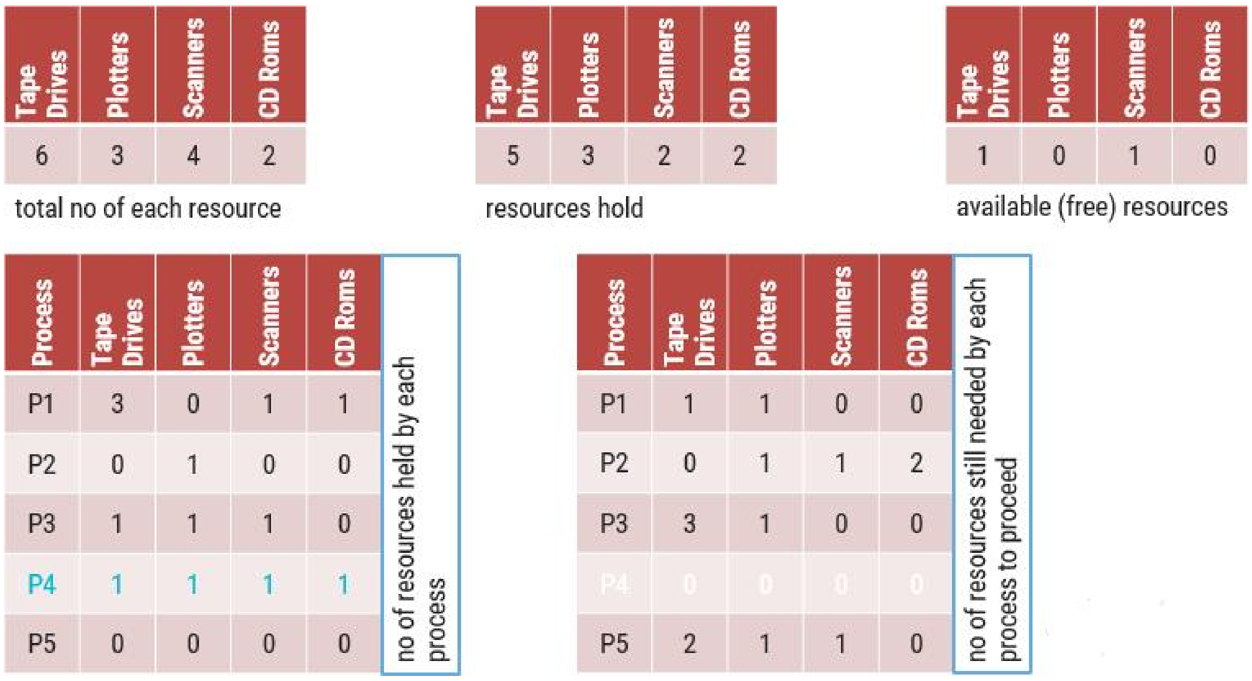
available (free) resources

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	3	0	1	1
P2	0	1	0	0
P3	1	1	1	0
P4	1	1	0	1
P5	0	0	0	0

no of resources held by each process

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	1	1	0	0
P2	0	1	1	2
P3	3	1	0	0
P4	0	0	1	0
P5	2	1	1	0

no of resources still needed by each process to proceed



Tape Drives	Plotters	Scanners	CD Roms
6	3	4	2

total no of each resource

Tape Drives	Plotters	Scanners	CD Roms
5	3	2	2

resources hold

Tape Drives	Plotters	Scanners	CD Roms
1	0	1	0

available (free) resources

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	3	0	1	1
P2	0	1	0	0
P3	1	1	1	0
P4	1	1	1	1
P5	0	0	0	0

no of resources held by each process

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	1	1	0	0
P2	0	1	1	2
P3	3	1	0	0
P4	0	0	0	0
P5	2	1	1	0

no of resources still needed by each process to proceed

P4,1,2,3,5

Deadlock recovery

1. Recovery through pre-emption

2. Recovery through rollback

3. Recovery through killing processes

Deadlock avoidance (Banker's algorithm)

- **Safe and unsafe states**

Banker's algorithm for single resource

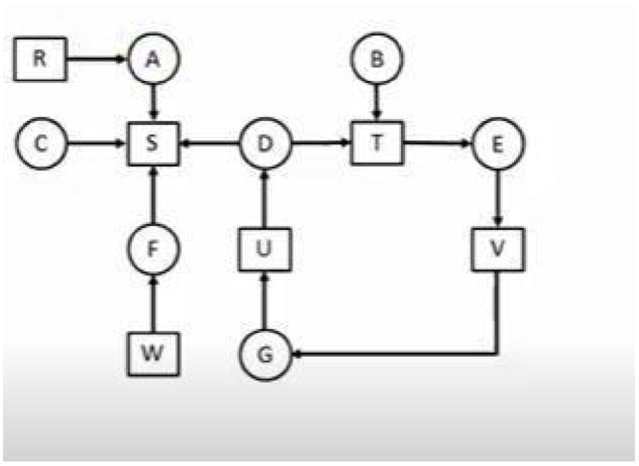
- **Banker's algorithm for single resource (safe state)**
- **Banker's algorithm for single resource (unsafe state)**

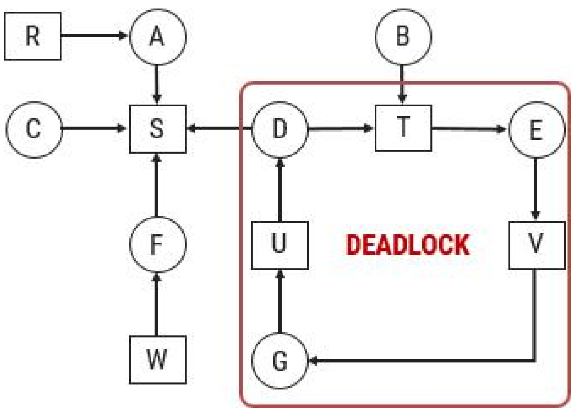
Banker's algorithm for multiple resource

Deadlock prevention

- Deadlock can be prevented by **attacking the one of the four conditions** that leads to deadlock.
    - Attacking the **Mutual Exclusion Condition**
    - Attacking the **Hold and Wait Condition**
    - Attacking the **No Preemption Condition**

Deadlock detection for single resource (RAG - Resource Allocation Graph)


$$\begin{tikzcd}
R \arrow[r] & A \arrow[d] & & B \arrow[d] & \\
C \arrow[r] & S & D \arrow[l] \arrow[r] & T \arrow[r] & E \arrow[d] \\
& F \arrow[u] & U \arrow[u] & & V \arrow[dll] \\
& W \arrow[u] & G \arrow[u] & &
\end{tikzcd}$$



- We are starting from node D.

- Empty list L = ()
- Add current node so Empty list = (D).
- From this node there is one outgoing arc to T so add T to list.
- So list become L = (D, T).
- Continue this step….so we get list as below

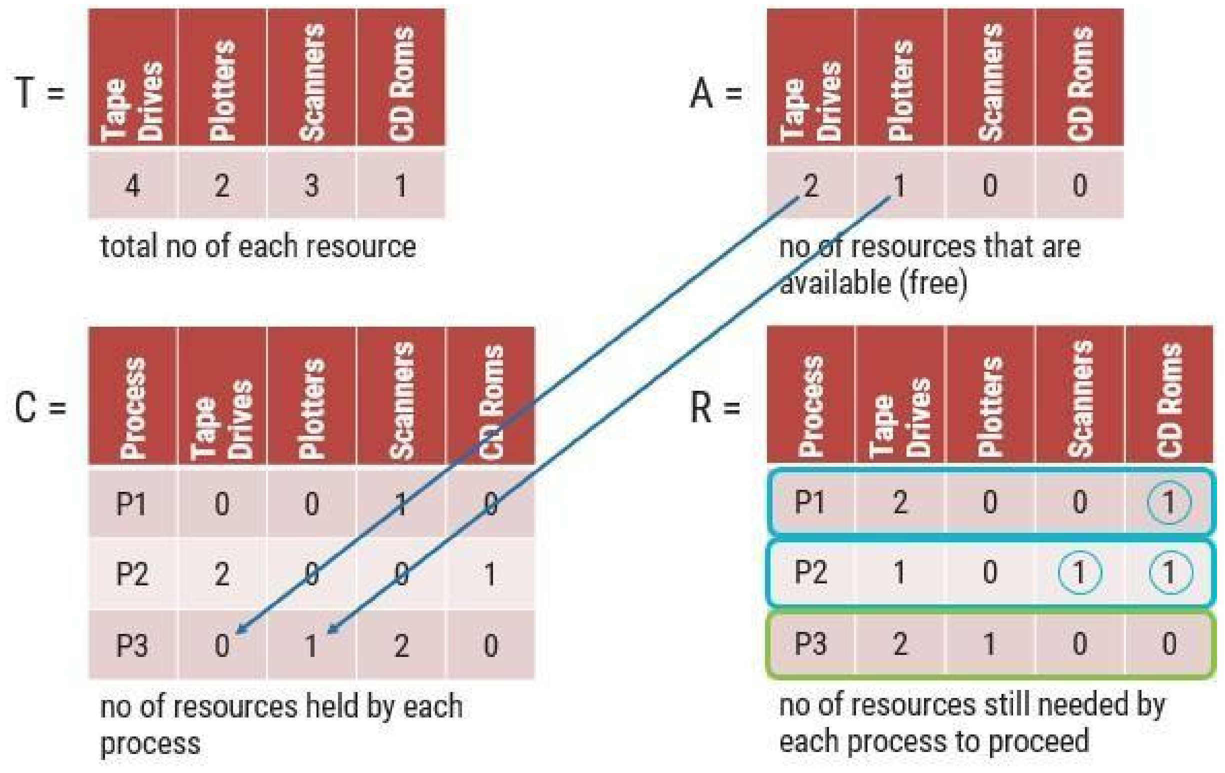
  L = (D, T, E)................. L = (**D**, T, E, V, G, U, **D**)
- In the above step in list the node **D appears twice**, **so deadlock**.

- Algorithm for detecting deadlock for single resource

↳For each node, N in the graph, perform the following five steps with N as the starting node.

1) Initialize L to the empty list, designate all arcs as unmarked.
2) Add current node to end of L, check to see if node now appears in L two times. If it does, graph contains a cycle (listed in L), algorithm terminates.
3) From given node, see if any unmarked outgoing arcs. If so, go to step 4; if not, go to step 5.
4) Pick an unmarked outgoing arc at random and mark it. Then follow it to the new current node and go to step 2.
5) If this is initial node, graph does not contain any cycles, algorithm terminates. Otherwise, dead end. Remove it, go back to previous node, make that one current node, go to step 2.

Deadlock detection for multiple resources



T =

Tape Drives	Plotters	Scanners	CD Roms
4	2	3	1

total no of each resource

A =

Tape Drives	Plotters	Scanners	CD Roms
2	1	0	0

no of resources that are available (free)

C =

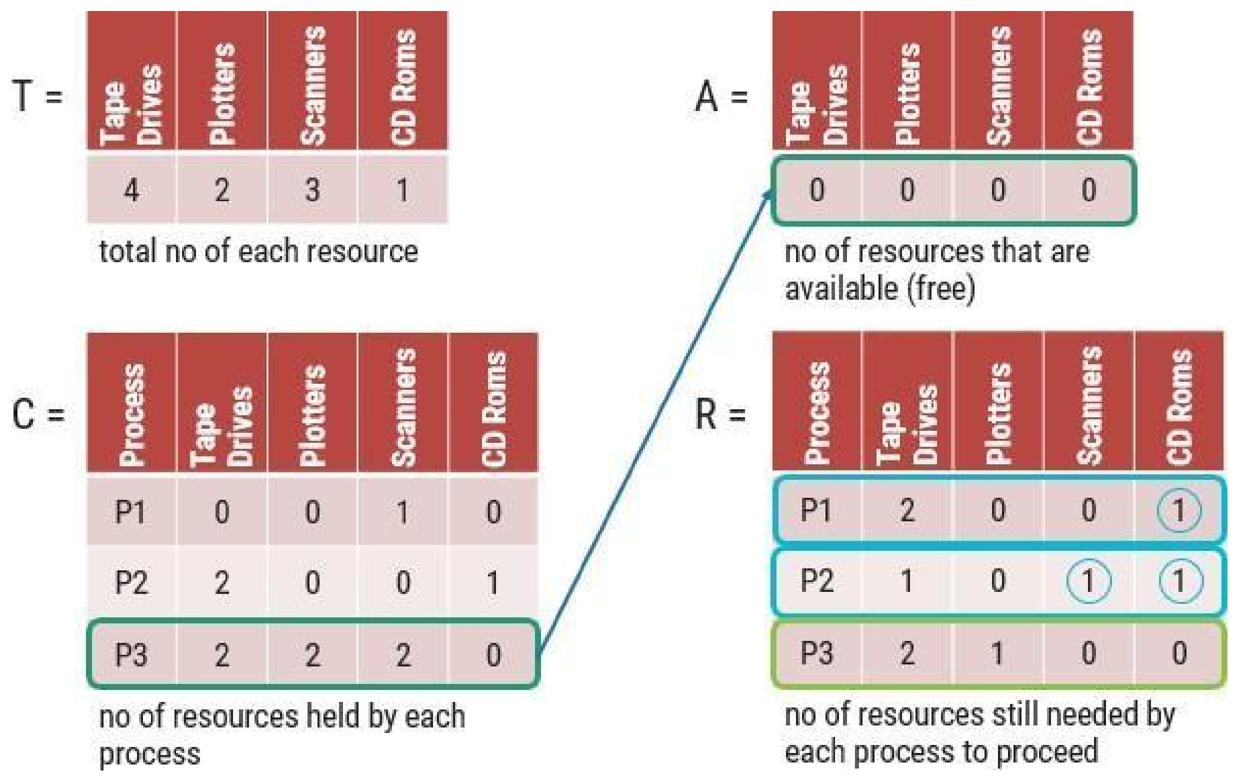
Process	Tape Drives	Plotters	Scanners	CD Roms
P1	0	0	1	0
P2	2	0	0	1
P3	0	1	2	0

no of resources held by each process

R =

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	2	0	0	1
P2	1	0	1	1
P3	2	1	0	0

no of resources still needed by each process to proceed



T =

Tape Drives	Plotters	Scanners	CD Roms
4	2	3	1

total no of each resource

A =

Tape Drives	Plotters	Scanners	CD Roms
0	0	0	0

no of resources that are available (free)

C =

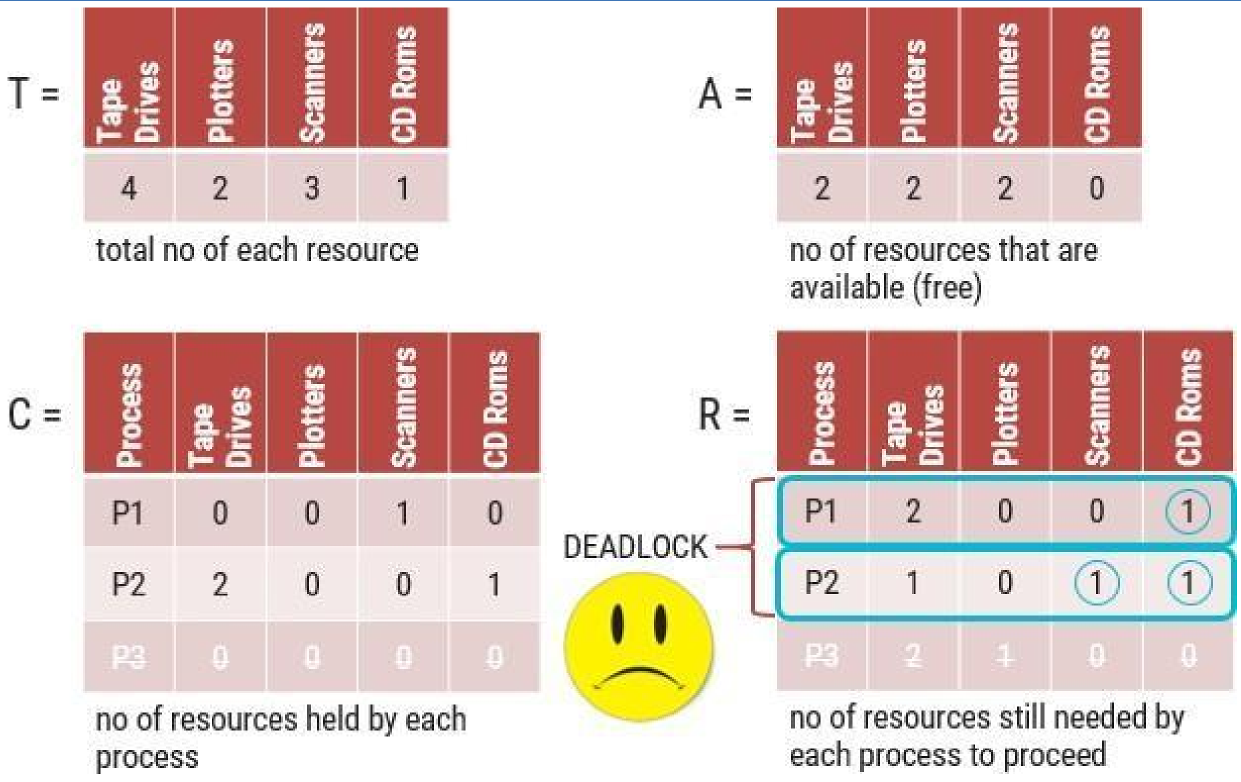
Process	Tape Drives	Plotters	Scanners	CD Roms
P1	0	0	1	0
P2	2	0	0	1
P3	2	2	2	0

no of resources held by each process

R =

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	2	0	0	1
P2	1	0	1	1
P3	2	1	0	0

no of resources still needed by each process to proceed



T =

Tape Drives	Plotters	Scanners	CD Roms
4	2	3	1

total no of each resource

A =

Tape Drives	Plotters	Scanners	CD Roms
2	2	2	0

no of resources that are available (free)

C =

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	0	0	1	0
P2	2	0	0	1
~~P3~~	~~0~~	~~0~~	~~0~~	~~0~~

no of resources held by each process

R =

Process	Tape Drives	Plotters	Scanners	CD Roms
P1	2	0	0	①
P2	1	0	①	①
~~P3~~	~~2~~	~~1~~	~~0~~	~~0~~

no of resources still needed by each process to proceed