Operating System(OS)

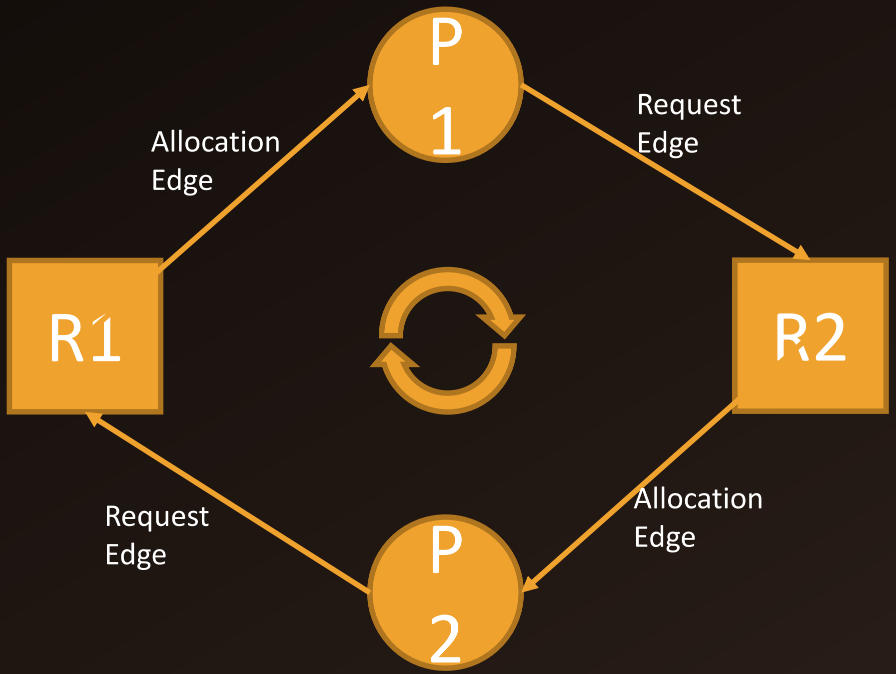
BCA IV SEM OS

Unit - 3 Deadlocks

OPERATING SYSTEM

DEADLOCKS

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some Other Process.



Mutual Exclusion

Hold And Wait

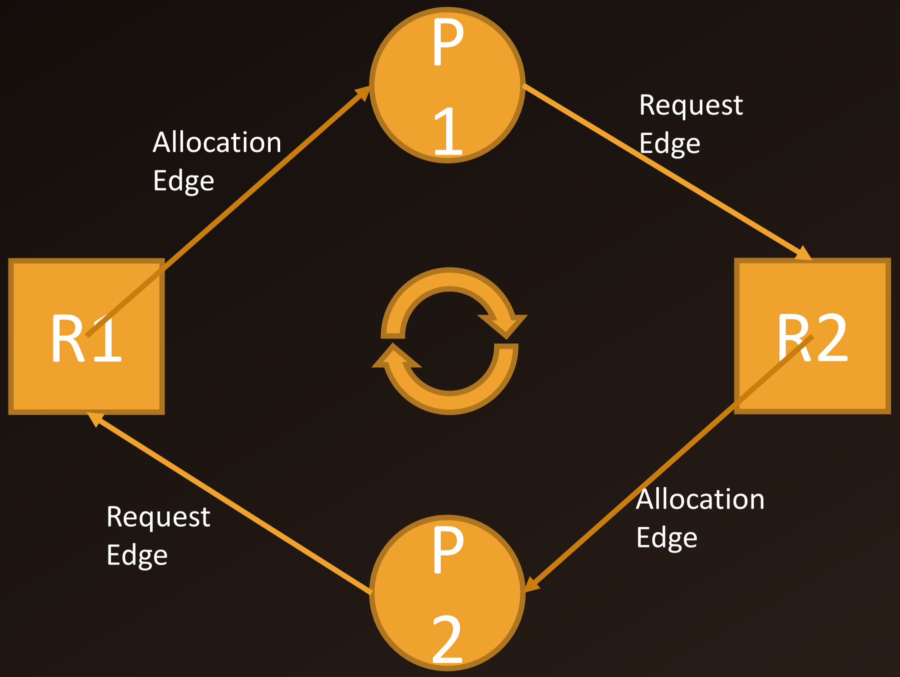
Deadlocks Conditions

No Preemption

Circular Wait

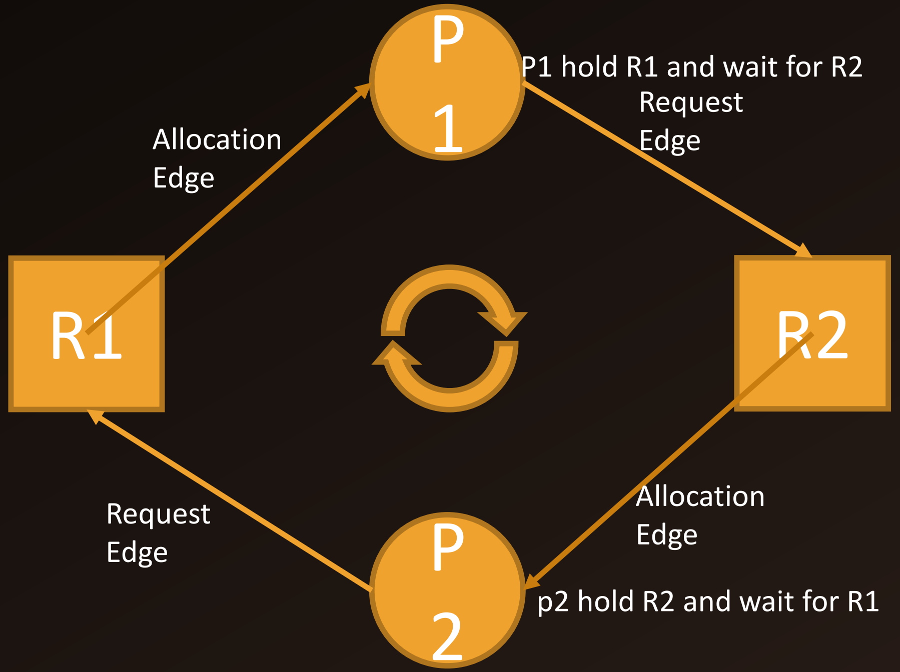
Deadlocks Conditions → **Mutual Exclusion**

- Only one process can use the resources at any given instant of time.
- Non-shareable resource then Deadlock accurse.
- R1,R2 have multiple resource but use only one at a time.



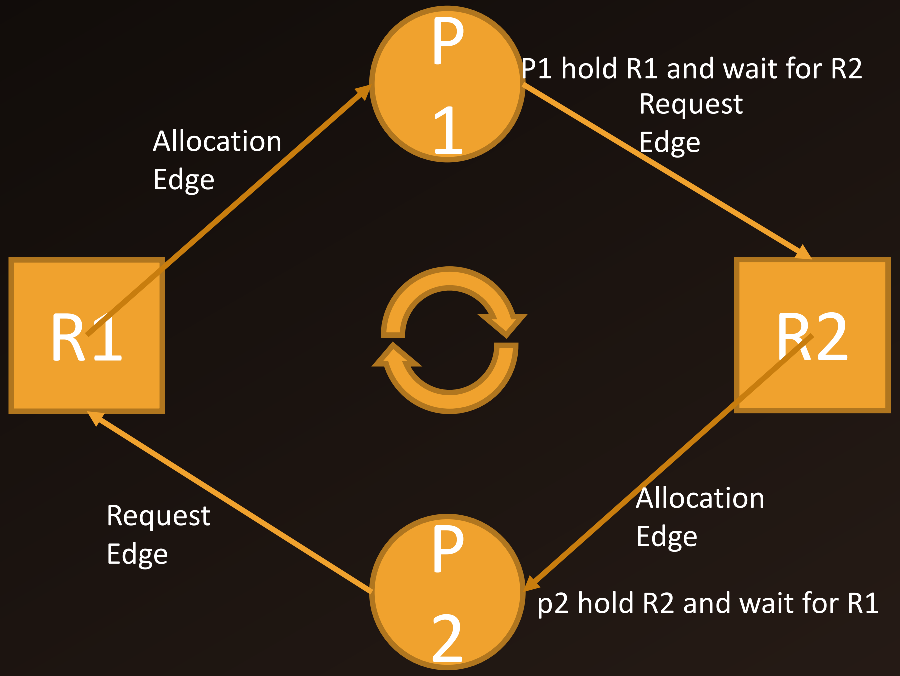
Deadlocks Conditions → **Hold And Wait**

- If one process holding a resource and waiting for another resource that is hold by another process then it leads to a deadlock.



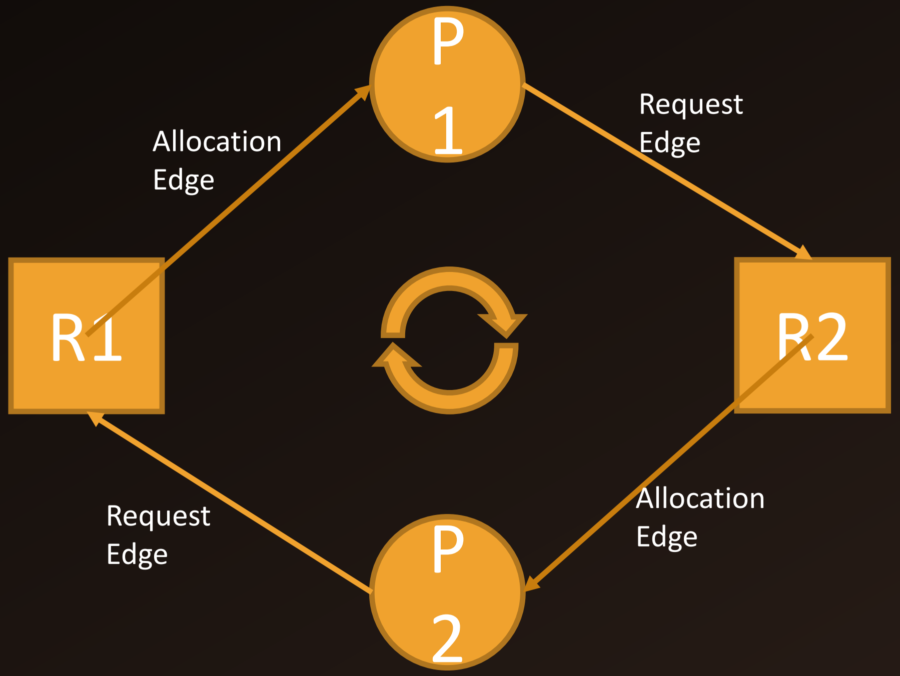
Deadlocks Conditions ➔ **No Pre-emption**

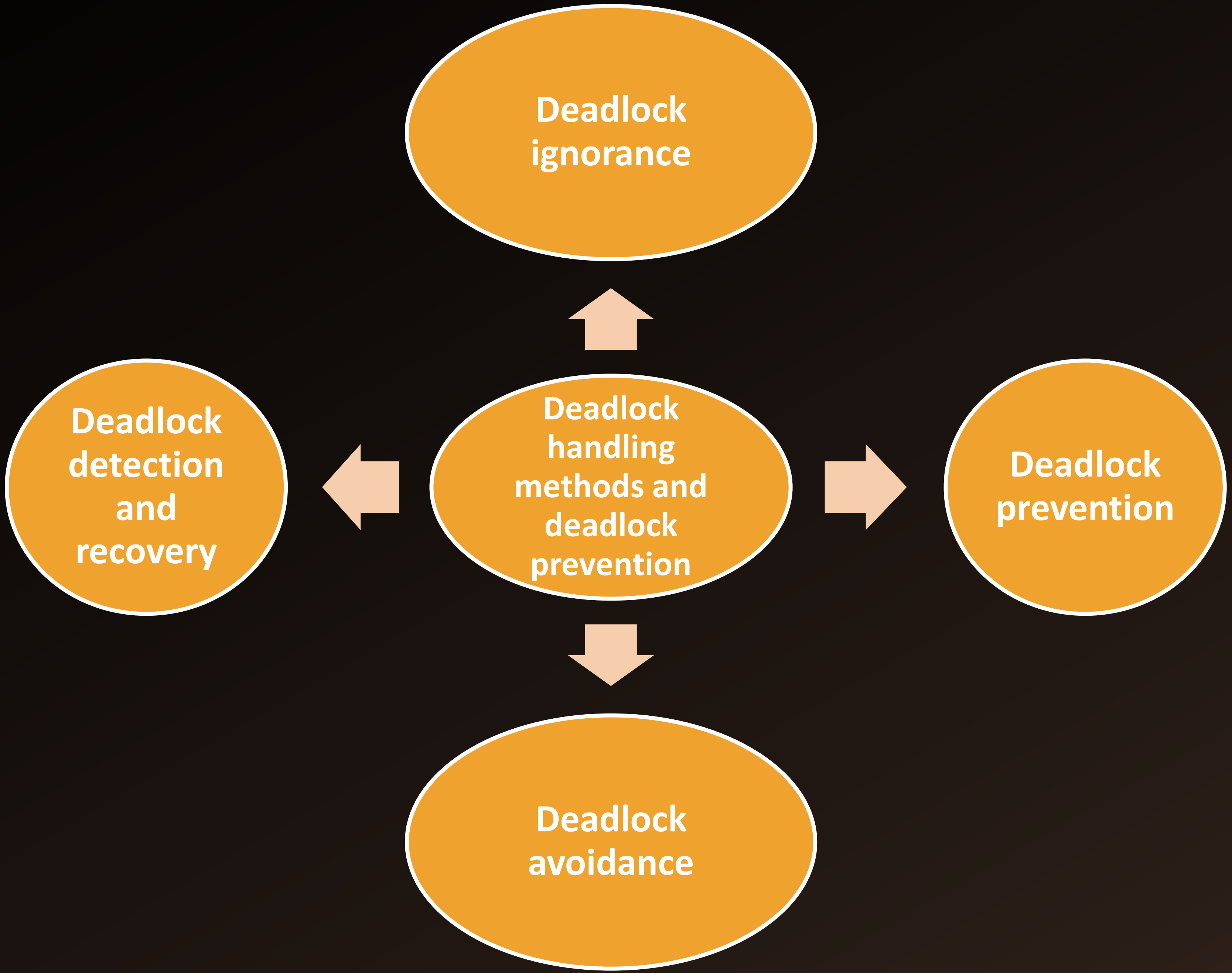
- If we have set no priority for all processes then every process demands that it will execute first and utilize the resource.



Deadlocks Conditions → **Circular Wait**

- If processes are waiting for resource in a circle and resource busy with other processes in circle then deadlock accurse.




Deadlock ignorance
Deadlock detection and recovery
Deadlock handling methods and deadlock prevention
Deadlock prevention
Deadlock avoidance

Deadlock Avoidance (prevention Algorithm)

Mutual exclusion

- Sharable resource Like:- Round Robin(Time slice/time quantum)
- Resource share on Time slice/time quantum based

No Preemption

- If any process priority is high then it's execute first
- If other process use resources and process priority is low then its remove using of resources

Hold and Wait

- Process used resources from R1 and hold process R2 deadlock occurs
- For resolving use condition → No hold or No wait

Circular Wait

- High priority level process no weight for low priority level process

Deadlock Avoidance (Bankers Algorithm)

Banker algorithm is the combination of safety algorithm and the resource request algorithm

Step 1

- Initialise work(availability of recourses) available
- Finish[i]= false, 1,2,3,4,5…..n/n-1

Step 2

- Check the availability
- If Need[i]<=work go to step 3
- Else finish[i]==false if I class not exist go to step 4

Step 3

- Work= work + Allocation(i)
- Finish[i]=True then go to step2

Step 4

- Finish[i] == true for all process system is safe state.

Safety Algorithm

Allocation Recourses
Available Recourses
Maximum Recourses Allocation

Recourses need
=
Maximum Recourses Allocation
-
Allocation Recourses

P1=need=x, work=y
P2=need=x, work=y
If(need<=work):
    P1 Executed
    work=work + a location

else if (need<=word):
    P2 Executed
    work =work + a location

Deadlock Avoidance(Bankers Algorithm)

Ex→ Consider a system that contain five process p1, p2, p3, p4, p5 and three resource type A, B, C. A has 10, B has 5 and c has 7 instance.

Process	Resource Allocation			Max Resource			Available work Resource True : AWR + RA			Resource Need MR - RA			Work RN <= AWR		
	A	B	C	A	B	C	A	B	C	A	B	C	A	B	C
P1	0	1	0	7	5	3	3	3	2	7	4	3	False waiting		
P2	2	0	0	3	2	2	3	3	2	1	2	2	True execute Safety Algorithm →1		
P3	3	0	2	9	0	2	5	3	2	6	0	0	False waiting		
P4	2	1	1	2	1	1	5	3	2	0	0	0	True execute Safety Algorithm →1,2		
P5	0	0	2	4	3	3	7	4	3	4	3	1	True execute Safety Algorithm →1,2,3		

STEP1 → Available work =332 need<= work

Continue to next slide

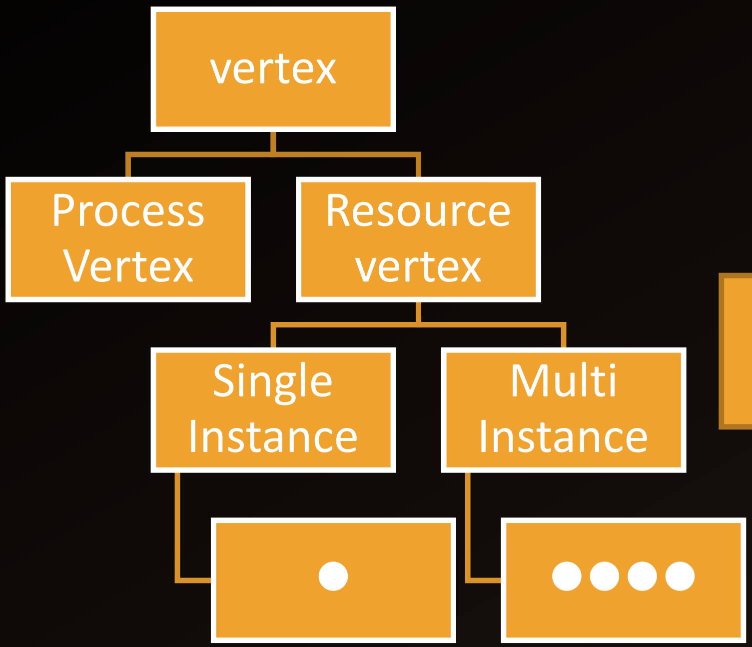
Deadlock Avoidance(Bankers Algorithm)

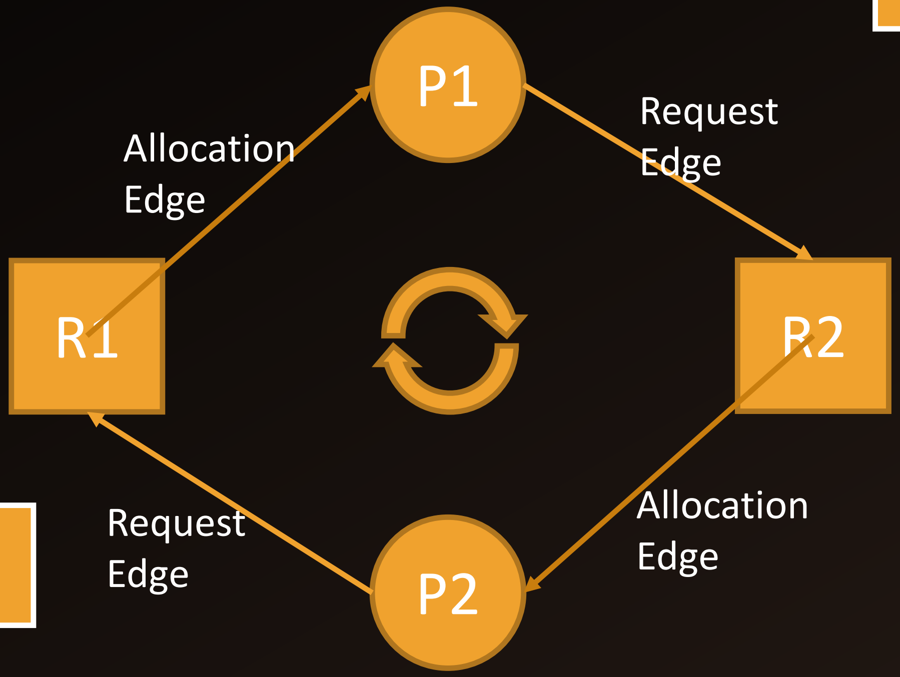
Ex➔ Consider a system that contain five process p1, p2, p3, p4, p5 and three resource type A, B, C. A has 10, B has 5 and c has 7 instance.

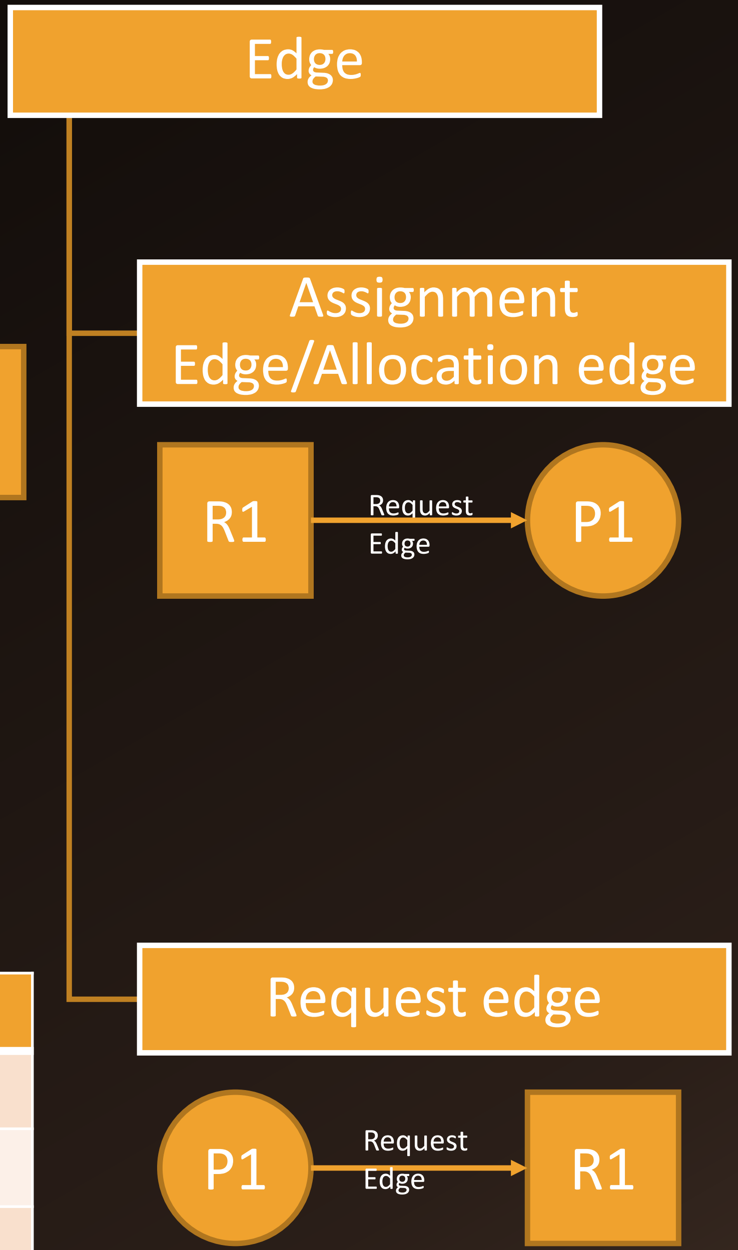
Process	**Resource Allocation**			**Max Resource**			**Available work Resource True : AWR + RA**			**Resource Need MR - RA**			**Work RN <= AWR**		
	A	B	C	A	B	C	A	B	C	A	B	C	A	B	C
P1	0	1	0	7	5	3	7	4	5	7	4	3	True execute Safety Algorithm➔1,2,3,4		
P2	2	0	0	3	2	2				1	2	2	True execute Safety Algorithm➔1		
P3	3	0	2	9	0	2	7	5	5	6	0	0	True execute Safety Algorithm➔1,2,3,4,5		
P4	2	1	1	2	1	1				0	0	0	True execute Safety Algorithm➔1,2		
P5	0	0	2	4	3	3				4	3	1	True execute Safety Algorithm➔1,2,3		
Final safe state is p2,p4,p5,p1,p3 with							A	B	C	Instance.					
							10	5	7	If match and found equal to given than Bankers safety algorithm is prefect.					

STEP1 ➔ Available work =332 need<= work

Resource Allocation Graph (RAG)



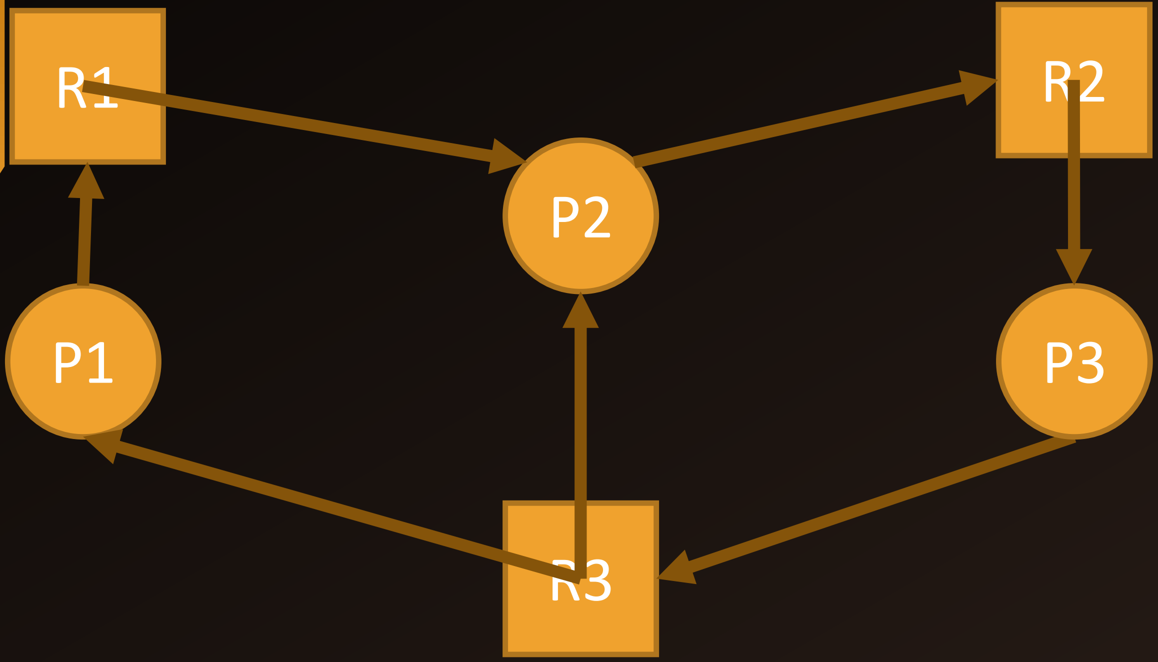




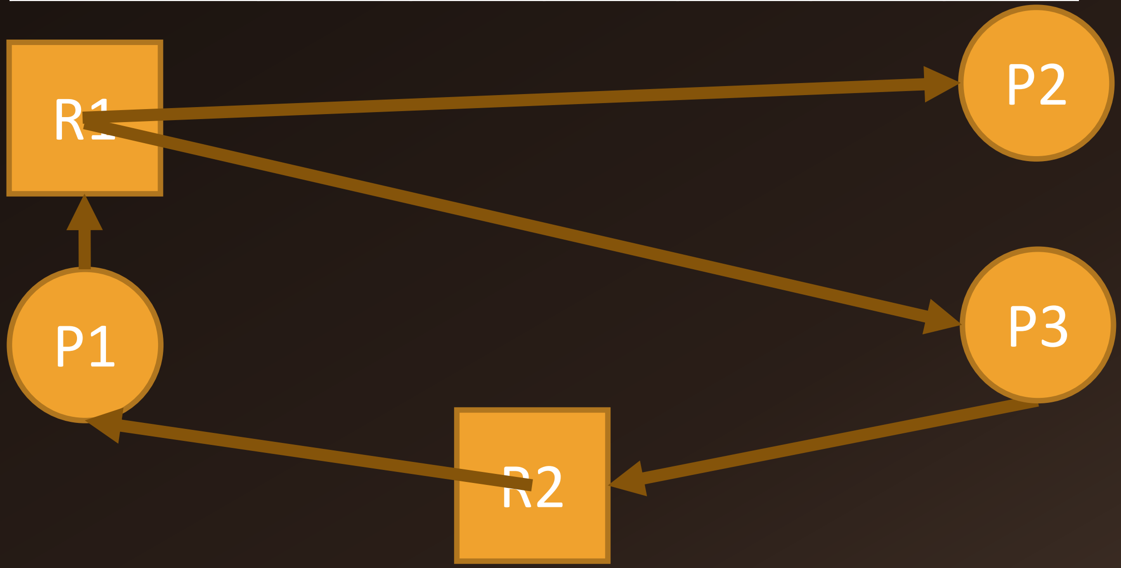
PROCESS	ALLOCATED		REQUEST		AVAILABLE	
	R1	R2	R1	R2	R1	R2
P1	1	0	0	1	0	0
P2	0	1	1	0		
Deadlock appear.						

Resource Allocation Graph (RAG)

PROCESS	ALLOCATED			REQUEST			AVAILABLE		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
P1									
P2							Deadlock appear.		
P3									



PROCESS	ALLOCATED		REQUEST		AVAILABLE	
	R1	R2	R1	R2	R1	R2
P1						
P2						
P3						



Resource Allocation Graph (RAG)

PROCESS	ALLOCATED			REQUEST			AVAILABLE		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
P1	0	0	1	1	0	0	0	0	0
P2	1	0	1	0	1	0	Deadlock appear.		
P3	0	1	0	0	0	1			

PROCESS	ALLOCATED		REQUEST		AVAILABLE	
	R1	R2	R1	R2	R1	R2
P1	0	1	1	0	0	0
					1	0
P2	1	0	0	0	1	0
					0	1
P3	1	0	0	1	1	1
					1	0
					2	1