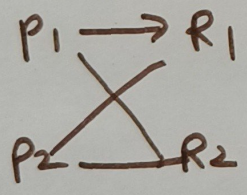
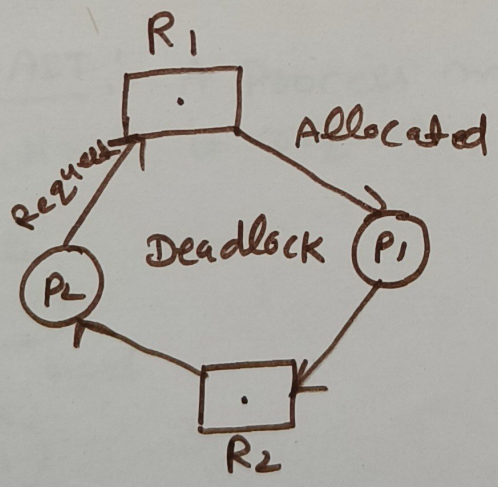
DEADLOCK :- A Deadlock is a Situation where a group of Processes are permanently blocked as a requet of each Process having acquired a subset of the resources needed for its completion and waiting for release of the remaining resource held by others in the same group.

Deadlock is a State Between two or more Resources in which they are completing for each other Resources for completing their tasks. and no one is Ready to leave Resources. that means waiting for that which cannot occur.





DEADLOCK CHARACTERIZATION :-

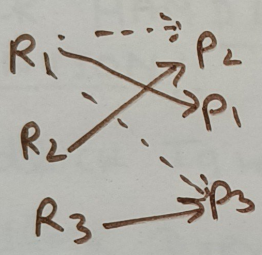
In a deadlock Processes never finish executing and System resources are tied-up Preventing other Jobs from Starting.

NECESSARY CONDITIONS FOR DEADLOCK :-

Deadlock can arise if four conditions hold

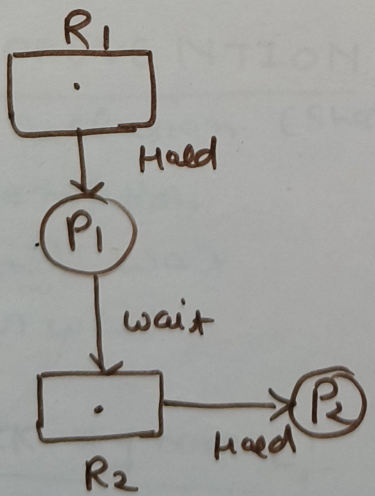
(1) Mutual exclusion
(2) Hold and wait
(3) No-Preemption
(4) circular wait.

(1) MUTUAL EXCLUSION:-
only one Process may use a resource at a time. once a Process has been allocated a Particular resource, it has exclusive use of the resource. No other Process can use a resource while it is allocated to a Process.



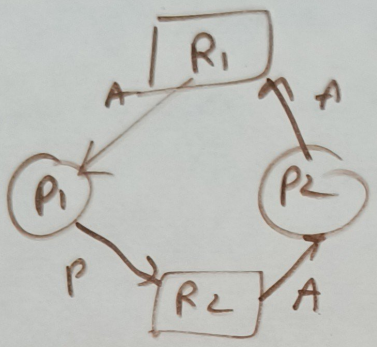
one-to-one

(2) HOLD AND WAIT:- A Process may Hold a resource at the same time it requests another one.



(3) NO PRE-EMPTION:- No Resource can be forcibly removed from a Process Holding it. Resources can be released only one by the explicit action of the Process, rather than by the action of an external authority.

(4) CIRCULAR WAIT :- All the Processes must be waiting for the resource in a cyclic manner.



DEADLOCK HANDLING METHOD AND DEADLOCK PREVENTION :-

⇒ Deadlock Ignorance
⇒ Deadlock Prevention
⇒ Deadlock avoidance (Banker's Algo)
⇒ Deadlock detection and recovery. (RAG)

DEADLOCK PREVENTION :-

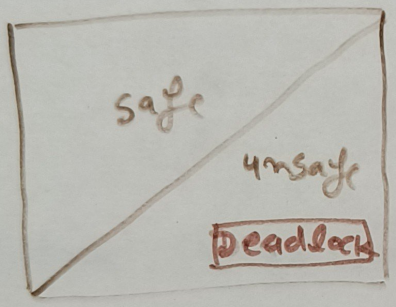
(1) Mutual Exclusion (Shared Resource
(2) No Preemption
(3) Hold and wait
(4) circular wait.

(1) Deadlock Ignorance :- we ignore the Problem as if it does not exist.

(2) Deadlock Prevention :-

DEADLOCK AVOIDANCE :-

(1) **Safe State :-** A State is safe if the system can allocate Resource upto its maximum in some sequence (safe sequence) and still Avoid Deadlock. Safe sequence may not be unique there may be multible safe sequence.



Safety Algorithm :-

(i) work = Available
finish = false

(ii) Find an 'i' such that
Finish [i] = false and
Need $i \leq$ work
if no such i, go to step 4

(iii) Finish [i] = true
work = work + Allocation

(iv) if Finish [i] = true for all i
then the system is safe.

⇒ 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 them suddenly request their maximum number of resource immediately.

Ex

Process	Allocated	maximum
A	3	9
B	2	4
C	2	7

total resources are 10

7 resources already allocated

So there are three resources are free

Free Resource = 10 − 7

= 3

A need 6 resources more to complet it

Need = maximum − Allocation

= 9 − 3

= 6

B need = 4 − 2

= 2 Resources

C need = 7 − 2 = 5 resources more to complete

Process	Allocate	max	
A	3	9	9-3 = 6
B	2	4	4-2=2
C	2	7	7-2=5

Free Resource = 3

Process	Alloc	max	
A	3	9	9-3=6
B	0	-	-
C	2	7	7-2=5

Free Resource = 5

Process	Allocated	max	
A	3	9	9-3=6
B	0	-	
C	0	-	

free resource = 7

Process	Allocated	max
A	0	-
B	0	-
C	0	-

free = 1

Safe Sequence

$< B, C, A >$

BANKER'S ALGORITHM:-

⇒ The Banker's algorithm is the best known of the avoidance strategies. The strategy is modelled after the leading policies employed in banking system. The resource-allocation graph algorithm is suitable to a resource allocation system with single instances of each resources type.

⇒ Banker's Algorithm is suitable for resource allocation system with multiple instances of each resource type. Algorithm makes decisions on granting a resource based on whether or not granting the request will put the system into an unsafe state.

⇒ Several data structure must be maintained to implement the Banker's algorithm. Let n be the number of processes in the system and m be the number of resource types. We need the following data-structure.

(1) Available
(2) max
(3) Allocation
(4) Need.

(1) Available :- A vector of length m indicates the number of available resources of each type. if available [j] = k then there are k instances of resource type R_j available.

(2) Max :- An n x m matrix defines the maximum demand of each Process. if max [i, j] = k then process P_i may request at most k instances of resource type R_j.

(3) Allocation :- An n x m matrix defines the number of resources of each type currently allocated to each Process. if allocation [i, j] = k, then Process P_i is currently allocated k instances of resource type R_j.

(4) Need :- An n x m matrix indicates the remaining resource need of each Process. if need [i, j] = k, then Process P_i may need k more instances of resource type R_j to complete its task. Need[i,j] = Allocation [i, j]

ALGORITHM :-

Bankers Algorithm is the combination of Safety Algorithm and the resource request Algorithm

Step1: Initialize work = available

finish[i] = false for i = 0, 1, 2, 3 ... n-1

Step2:

Check the availability

Need[i] <= work go to Step 3

else finish[i] = = false if i does not exist go to Step-4

Step-3: work = work + Allocation[i]

finish[i] = true then go to Step 2

~~else finish[i] = false if i~~

Step-4: if finish[i] = = true for all

System is safe state

Need = Maximum - Allocation

Example:

Consider a system that contain five Process P_1, P_2, P_3, P_4, P_5, and three types A, B & C

A has 10, B has 5 and C has 7 instance.

Process	Allocation			Maximum			Available			Need		
	A	B	C	A	B	C	A	B	C	A	B	C
P_1	0	1	0	7	5	3	3	3	2	7	4	3
P_2	2	0	0	3	2	2				1	2	2
P_3	3	0	2	9	0	2				6	0	0
P_4	2	1	1	2	1	1				0	0	0
P_5	0	0	2	4	3	3				4	3	1

Step 1:

Available work = 3 3 2

Need(1) $\leq$ work

743 $\leq$ 332 P_1 is not executed.

$Need_2 \leq work$

122 $\leq$ 332 P_2 is executed

work = work + Allocation of P_2

work = 332 + 200

work = 532

$Need_3 \leq work$

600 $\leq$ 532 false P_3 waiting

$Need_4 \leq work$

000 $\leq$ 532 true P_4 executed.

New work = work + Allocation of P_4

= 532 + 211

= 743

$Need_5 \leq work$

$431 \leq 743$ true P_5 executed.

$Need_1 \leq work$

$743 \leq 745$ true.

work = work + Allocation of P_1

work = 745 + 010

= 755

$Need_3 \leq work$

$600 \leq 755$ true P_3 is executed

new work = work + Allocation of P_3

new work = 755 + 302

= 1057

Safe Sequece

$< P_2, P_4, P_5, P_1, P_3 >$ Ans

Ex2

Process	Allocation A B C D	maximum A B C D	Available A B C D	Need A B C D
P_0	0 0 1 2	0 0 1 2	1 5 2 0	0 0 0 0
P_1	1 0 0 0	1 7 5 0	1 5 3 2	0 7 5 0
P_2	1 3 5 4	2 3 5 6	2 8 8 6	1 0 0 2
P_3	0 6 3 2	0 6 5 2	2 14 11 8	0 0 2 0
P_4	0 0 1 4	0 6 5 6	2 14 12 12	0 6 4 2
			3 14 12 12	

Do yourself

$\langle P_0, P_2, P_3, P_4, P_1 \rangle$

RESOURCES ALLOCATION GRAPH!

 ⇒ Process

 ⇒ Resources

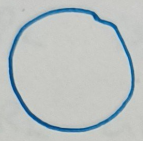
 ⇒ Number of Instances

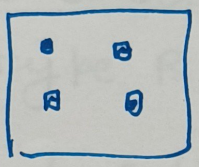
 ⇒ Request and Allocated arrow Like

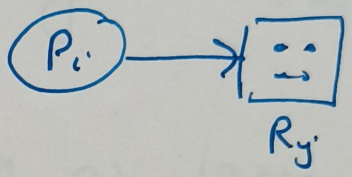
Resource Allocation graph is used to describe the deadlock. It is also called system resource graph.

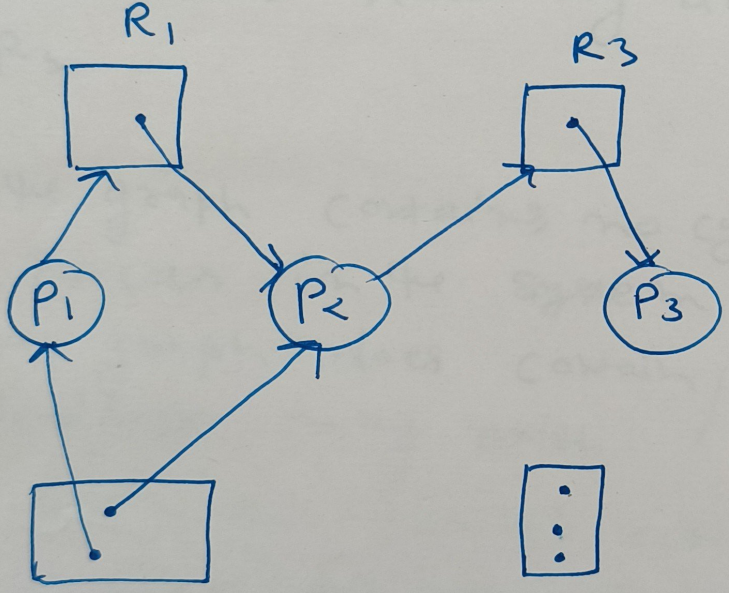
graph consists of a set of vertices (V) and set of edges (E).

All the active Processes in the system denoted by $P = \{P_1, P_2 \cdots P_n\}$ and set consisting of all resource type in the system is denoted by $R = \{R_1, R_2, \cdots R_m\}$.

Process  Resource type with 4 instances







(Resource allocation graph)

in this graph

Resource R_1 = one instance

Resource R_2 = two "

" R_3 = one "

" R_4 = three "

Process States:

Process P_1 is holding an instance of resource type R_2 and is waiting for an instance of Resource type R_1.

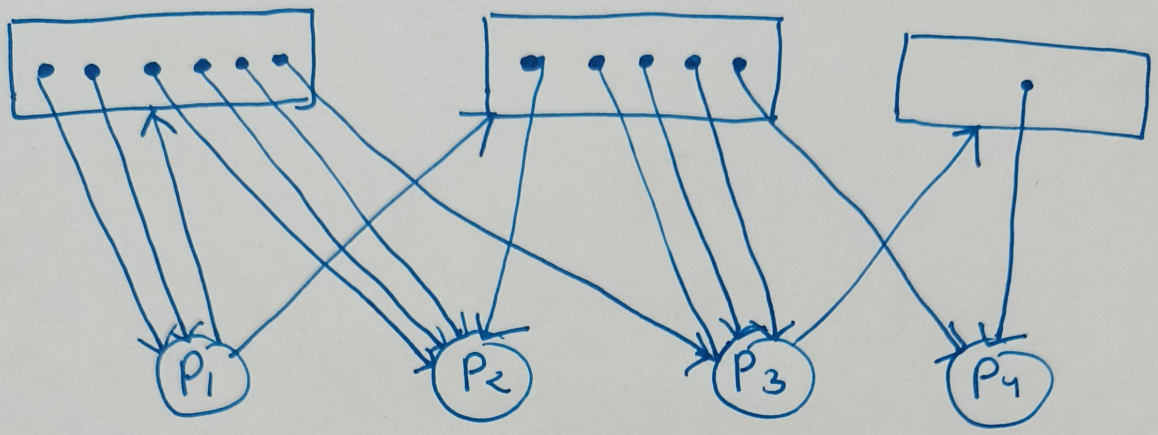
Process P_2 is Holding an instance of R_1 and R_2 and waiting for an instance of Resource type R_3.

Process P_3 is holding an instance of R_3

if the graph contains no cycles, then no Process in the System is deadlock. if the graph does contain a cycle then a deadlock may exist.

Ex: given the process resource usage and availability draw the resource-allocation graph.

Process	current allocation			outstanding requests			Available resource		
	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3
P_1	2	0	0	1	1	0	0	0	0
P_2	3	1	0	0	0	0			
P_3	1	3	0	0	0	1			
P_4	0	1	1	0	1	0			



Ex Consider the following snapshot of a system.

Process	Allocated			maximum			Available		
	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3
P_1	2	2	3	3	6	8	7	7	10
P_2	2	0	3	4	3	3			
P_3	1	2	4	3	4	4			

Answer the following questions using the banker's algorithm.

(1) what is the content of the matrix need?

(2) Is the system in safe state or unsafe state.

D.Y.S

Practice Problem