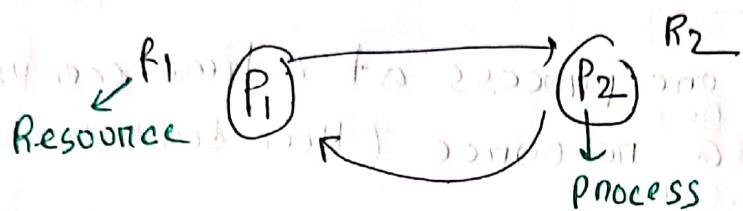


what is deadlock?

→ deadlock is a situation that occurs in OS because when any process enters waiting state, another waiting processes is holding demanded resource



two or more process waiting for each other.

Resource type



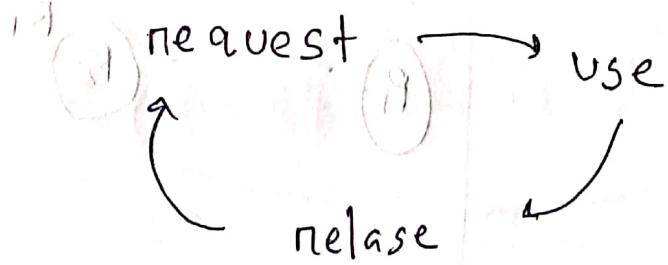
cpu cycle

memory space

I/O device

using semaphores

what process do with resource?



when Deadlocks occurs ?

Deadlock
Detection

→ if there is **four** condition arise

① Mutual exclusion :-

only one process at a time can use a resource. (Hence deadlock can occurs)

(if resource are share with multiple process
deadlock never happens)

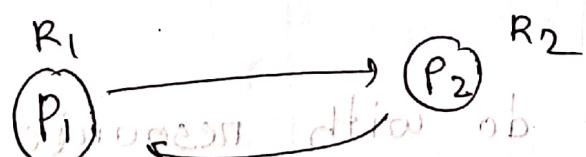
② Hold and wait :-

a process holding at least one resource is waiting to acquire additional resources held by another process

Hence, deadlock occurs

Hold and wait

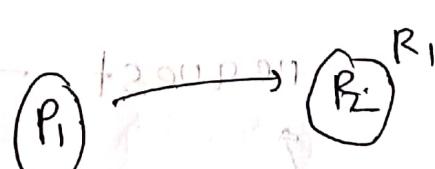
Hence P_1
holding resource
 R_1 and waiting
for resource
 R_2



No deadlock

no Hold
only wait.

Hence P_1 holding no resource but waiting for resource R_2



③ No preemption :-

a resource can be released only by the process holding it, after the process has completed.

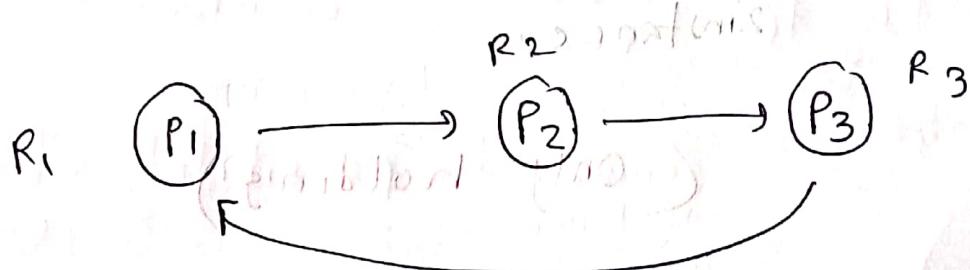
P_1 cannot release R_1 until finish. but P_1 will not finish because it's waiting for R_2 . Hence deadlock can occurs.

if preemptive, no deadlock, e.g. P_1 can release R_1 with finishing its job

④ Circular wait :-

one process is waiting for the resource is held by second process, which also waiting for the resource held by third process. This waiting continue until last process is waiting for a resource held by first process.

is called circular wait



Resource Allocation

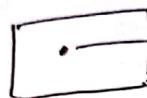
In the Graph

Process node :-



resource node :-

Process, P requesting instance of R

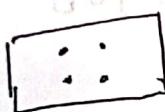


instance of

resource node

resource node

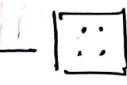
Hence 1



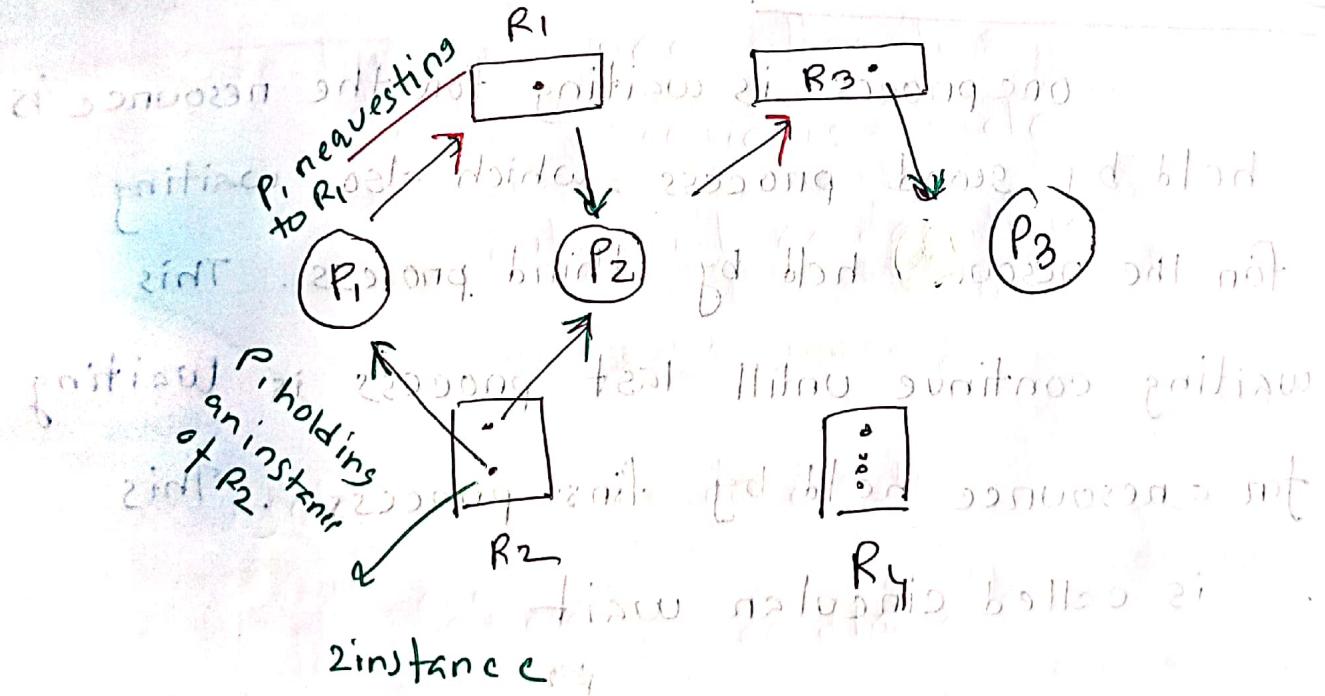
Hence
instance of



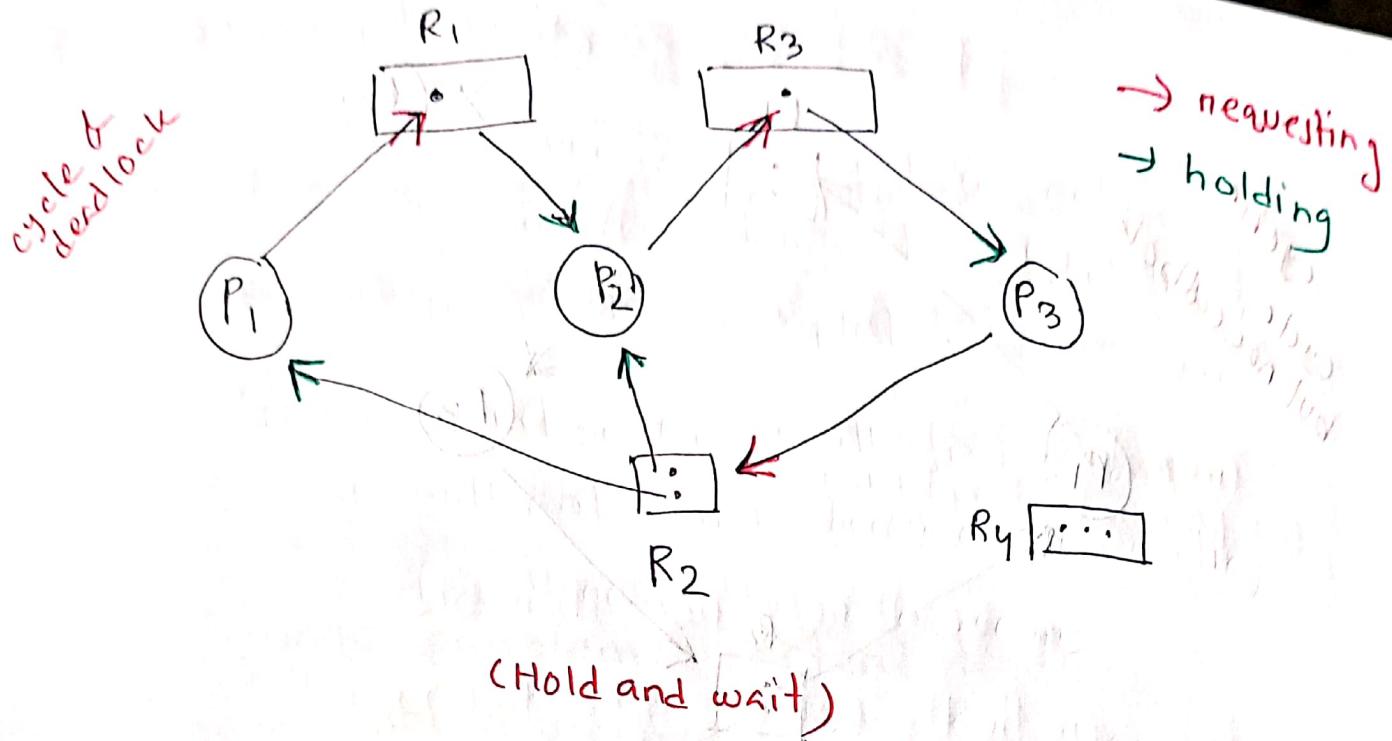
Process, P holding
an instance of R



-> finds available (P)



(Only holding)

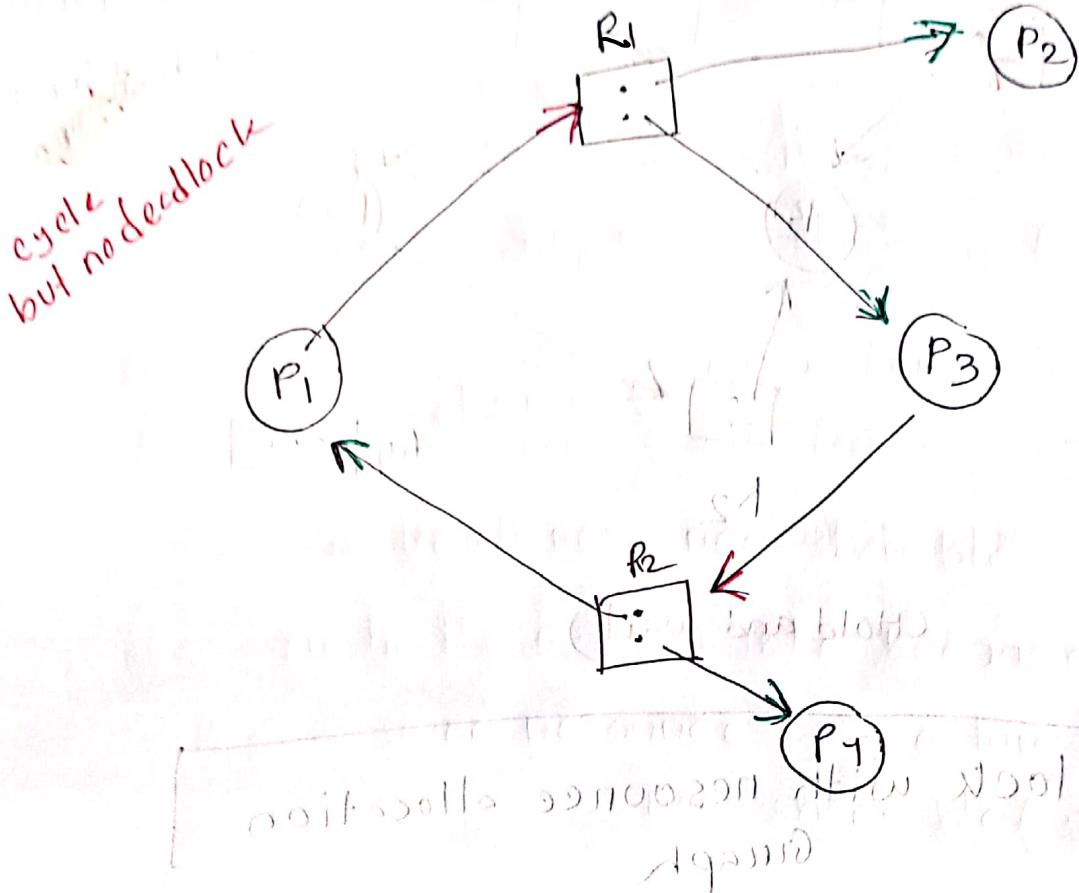


Deadlock with resource allocation
Graph

P₁ waiting for instance 1 of R₂, while holding
an instance of R₂

P₂ waiting for instance of R₃, while holding
an instance of from both R₁, R₂

P₃ waiting for instance of R₂, while
holding an instance of R₃



In this resource allocation

graph, there is a cycle

but still no deadlock

~~no cycle \rightarrow no deadlock~~

~~if cycle $\xrightarrow{\text{one instance}} \text{to deadlock}$~~

~~serveral instances \rightarrow possibility to deadlock (not sure)~~

Methods for handling deadlocks

- ① deadlock prevention (Never enter deadlock)
- ② deadlock avoidance (Don't enter "deadlock")
- ③ allows a system to enter deadlock then recover.
- ④ Ignore problem & pretend deadlock never occurred.

-; Since D block (9)

provided that software team

is programmed to prevent

common faults like block anomalies

as shown in the following diagram

common faults

Temporary

Deadlock

Self-loop

Resource lock

for information, for instance

Deadlock prevention

if we can prevent any of the deadlock detection

then there will be no deadlock.

Practically prevention is not possible

④ preventing

Mutual Exclusion :-

make all resource sharable

practically not possible because

some resources are naturally

non-sharable (write only files)

② Hold & write :-

must guarantee that whenever

a process request a resource, it

does not hold any other resource

low resource utilization

Method-1

require process
to request and
be allocated all its
resources before
it begins execution

↓ method-2

allow process
to request
resource only
when the
process has none
allocated to it

③ No preemption:-

process

→ if a process is holding some resources and requests for another resource, at first, the process has to release all allocated / holding resources, the request requested resource will be allocated to it.

→ Process will be restarted only if can regain its old resources as well as new ones that is requesting.

→ cannot apply to locks & semaphores

④ Circular wait:-

impose a total ordering of all resource types & require that each process request resource in an increasing number.

Deadlock avoidance

- ① Each process declares the maximum number of resources of each type available to it, the number of resources that it may need, and its maximum demand.
- ② Resource allocation state is defined by the numbers of available & allocated resources, the maximum demands of processes and maximum demands of the processes.
- ③ Deadlock avoidance algorithm examines resource allocation state so that there can never be a circular wait condition.

What is safe state?

final

- a state of system is called safe if the system can allocate the resources requested by all the processes without entering into deadlock.
- system $S = (SBSR)$ is in safe state if exists sequence $\langle P_0, P_1, P_2, \dots, P_n \rangle$ of all processes in the system such that for each P_i ,
the resource P_i can still be requested and be satisfied by currently available resources + resources held by P_j with $j < i$.

<u>process</u>	<u>maximum need</u>	<u>current allocation on instance</u>	<u>need</u>
P ₀	10	5	5
P ₁	9	2	7
P ₂	7	2	5

$\therefore \text{available resource} = 12 - (5+2+2) = 3$

now available resource = 3

$\therefore P_1$ will enter first ($P_1 \text{ need}(2) \leq \text{available resource}(3)$)

After P_1 finish, it will release its resource

: Now total available

$$\text{resource} = 3 + 2 = 5$$

$\therefore P_0$ will enter. ($P_0 \text{ need}(5) \leq \text{available resource}(5)$)

\therefore after P_0 finish it will leave its resource

now total available resource = $S_1 + S_2 = 10$

: now P_2 enters ; P_2 needs (7) \leq total available resources (10)

$\therefore P_2$ terminates.

All processes got their desired resource without deadlock.

\therefore state sequence = $\langle P_1, P_0, P_2 \rangle$

system is safe hence

state of system

<u>Process</u>	<u>maximum need</u>	<u>current allocation</u>	<u>need</u>
P_0	10	5	5
P_1	4	2	2
P_2	9	3	6

available resource = $12 - (5+2+3) = 2$

available resource = $12 - (5+2+3) = 2$

now P_1 enters C_2 (P_1 needs \leq available resource)

after P_1 finish; total available resource = $2+2=4$

∴ P_0 finish; total available resource = $2+2=4$

∴ now none P_0 or P_2 can enter

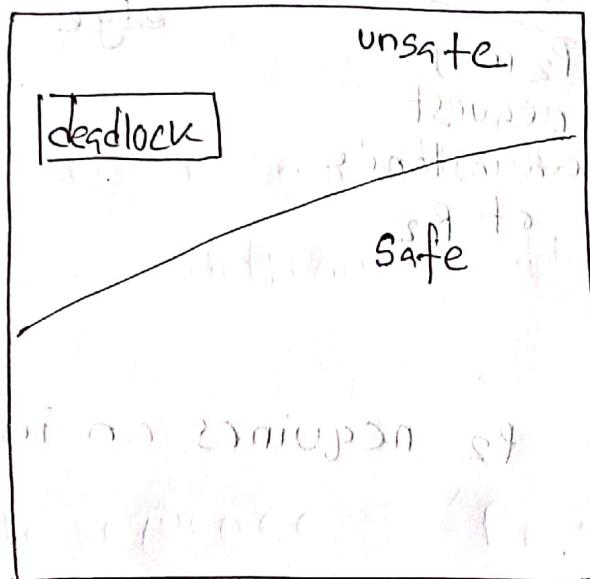
because

P_0/P_2 needs $>$ available resource

Unsafe state

safe state \rightarrow deadlock

unsafe state \rightarrow possibility of deadlock



single

instance

Using
resource
allocated
graph

multiple

instance

Banzen's
Algorithm

Avoidance
Algorithm

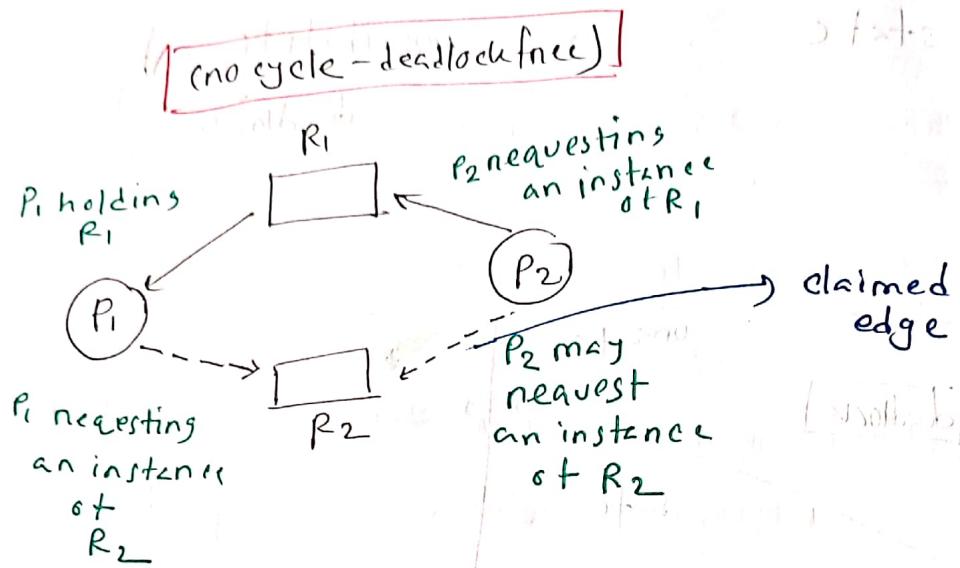
multiple
instances

of form $E(S)$

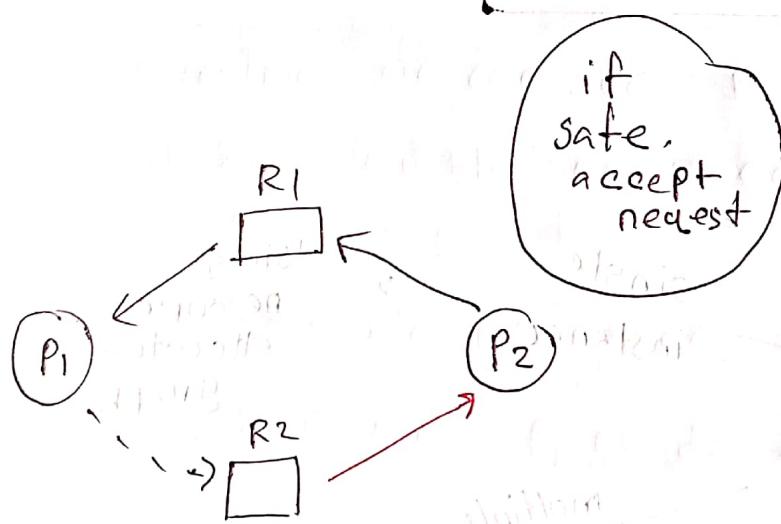
subset limit

to respond to request of tasks

avoiding deadlock using Resource - Allocation graph



what happen if P2 requires an instance of R2.



cycle, deadlock possibility

: unsafe

Deny the request of P2 for R2

① we will temporarily accept the request and give it to P2

② Now check the system is safe or unsafe

③ if unsafe, will deny the request & undo change

Deadlock avoidance with Banker's Algorithm

n = number of processes

m = number of resource type

① Available

② Max

③ Allocation

④ Need = Max - Allocation

(Given all)

Q) 5 process P₀ through P₄

: 3 resource type

A (10 instance) B (5 instance) C (7 instance)

Process	Allocation			(Max - allocation) Need	Available
	A	B	C		
P ₀	3	5	3	7 4 3	3 3 2
P ₁	2	0	0	5 5 3	3 3 2
P ₂	3	0	2	6 0 0	3 3 2
P ₃	1	3	2	5 0 1	3 3 1
P ₄	0	0	2	4 3 3	3 3 1

: Find safe state sequence?

(Multiple ans; but ~~task dependent~~ find ans
for actual/convention ans)

Ans:- safe sequence :-

$(P_1, P_3, P_0, P_2, P_4)$

[3 3 2]

<u>Process</u>	<u>Allocation</u>	<u>Need</u>	<u>Available</u>
	A B C	A B C	A B C
P ₀	2 0 0	1 2 2	5 3 2
P ₁	1 0 1	7 4 3	(1 by applying)
P ₂	3 0 2	6 0 0	10 5 5
P ₃	0 0 2	4 3 1	10 5 7
P ₄	0 0 2	4 3 1	<u>= [(A B C)]</u> resource type in given

Explains,

In Question 3, we can see P₀ is incomplete

- ∴ it has available resources A(3) B(3) C(2)
- ∴ In this available resource, P₁ can complete its job, cause P₁ needs 1 < available resources

1 0 1 8 8 1 2 2 6 0 3 3 2

∴ P₁ enters.

after P₁ finishes job

Computer will start next

\therefore it releases its allocated resources.

$$\text{Available resource} = \begin{array}{r} A \\ 3 \\ + 2 \\ \hline 5 \end{array} \quad \begin{array}{r} B \\ 3 \\ 0 \\ \hline 3 \end{array} \quad \begin{array}{r} C \text{ (initial)} \\ 2 \\ 0 \\ \hline 2 \end{array}$$

(available past)
(P₁ allocation)

P₃ needs < available resources

P₃ enters cause

0 1 1 5 3 2

0 0 0 0 0 0

0 0 0 0 0 0

0 0 0 0 0 0

$$\text{Now available resource} = \begin{array}{r} 5 \\ 3 \\ + 2 \\ \hline 7 \end{array} \quad \begin{array}{r} 3 \\ 1 \\ \hline 4 \end{array} \quad \begin{array}{r} 2 \\ 1 \\ \hline 3 \end{array}$$

(P₃ alloc)

\therefore now P₀ can enter cause now P₀ needs 2 available resources

7 4 3 7 4 3

Now question why P₀. why not P₁ cause we are traversing serially?

Ans:- if take P₄ after P₄ finish available resource

after P₀ finish. available resource

$$\begin{array}{ccc} A & B & C \\ \text{initial} & 7 & 4 & 3 \\ \text{consumes} & +0 & +1 & +0 \\ \hline & 7 & 5 & 3 \end{array}$$

now, P₂ enters cause

P₂ needs Available
resource

$$\begin{array}{cccc} & 0 & 0 & 0 \\ & 0 & 0 & 0 \\ & 7 & 5 & 3 \end{array}$$

now P₂ consumes available resource

$$\begin{array}{ccc} A & B & C \\ \text{initial} & 7 & 5 & 3 \\ \text{consumes} & +3 & 0 & +2 \\ \hline & 10 & 5 & 5 \end{array}$$

now P₂ enters cause only needs available
resource

now P₂ finishes cause available resource

available resource

$$\begin{array}{ccc} A & B & C \\ \text{initial} & 7 & 5 & 3 \\ \text{consumes} & -7 & -5 & -3 \\ \hline & 0 & 0 & 0 \end{array}$$

$$\begin{array}{ccc} 0 & 0 & 0 \\ 10 & 5 & 3 \\ \hline 7 & \end{array}$$

there are multiple ans :-

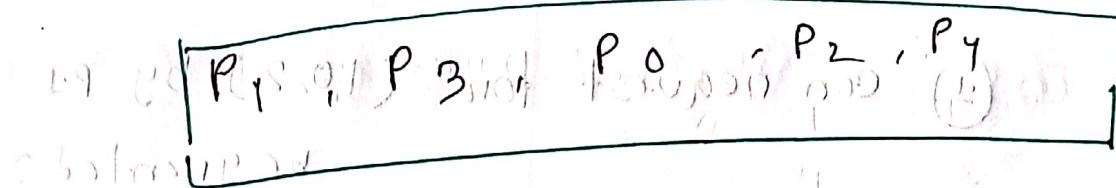
$\langle P_1, P_3, P_4, P_0, P_2 \rangle$ (Gives best ans)

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

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

Actual/convention

why choose this sequence of processes



if P_1, P_3 can both got chance

shortname depends on their needs & available resource

if P_1 we will choose P_1 first
cause it comes first in process list

same P_0, P_4

\therefore if P_0/P_4 needs & available resource

\therefore we choose P_0 first.

Q2 :- In the diagram

process	Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C
P ₀	0	1	0	7	1	3	3	3	2
P ₁	2	0	0	1	2	2	3	3	3
P ₂	3	0	2	6	0	0	3	3	3
P ₃	2	1	1	0	1	1	3	3	3
P ₄	0	0	2	4	3	1	3	3	3

(1) can request for (1,0,2) by P₁ be granted?

(2) can request for (3,3,0) by P₄ be granted?

(3) can request for (0,1,0) by P₀ be granted.

$P_1(1, 0, 2)$

① Ans:- First update this values:

~~Allocation of P_1 :- $\begin{matrix} 2 & 0 & 0 \end{matrix}$ (past)~~

Allocation of P_1 :- $\begin{matrix} +1 & 0 & 2 \\ \hline 3 & 0 & 2 \end{matrix}$ (new)

Need of P_1 :- $\begin{matrix} 1 & 2 & 2 \\ -1 & 0 & 2 \end{matrix}$ newest

available :- $\begin{matrix} 3 & 3 & 2 \\ -1 & 0 & 2 \end{matrix}$ past

Allocation :- $\begin{matrix} 3 & 3 & 2 \\ -2 & 3 & 0 \end{matrix}$ newest

Now the diagrams :-

	<u>Allocation</u>	<u>Need</u>	<u>Available</u>
P_0	$\begin{matrix} 1 & 0 & 0 \end{matrix}$	$\begin{matrix} 1 & 7 & 4 \end{matrix}$	$\begin{matrix} 3 & 2 & 3 \end{matrix}$
P_1	$\begin{matrix} 3 & 0 & 2 \end{matrix}$	$\begin{matrix} 0 & 2 & 0 \end{matrix}$	$\begin{matrix} 0 & 1 & 0 \end{matrix}$
P_2	$\begin{matrix} 3 & 0 & 2 \end{matrix}$	$\begin{matrix} 6 & 0 & 0 \end{matrix}$	$\begin{matrix} 0 & 1 & 0 \end{matrix}$
P_3	$\begin{matrix} 2 & 1 & 1 \end{matrix}$	$\begin{matrix} 8 & 0 & 0 \end{matrix}$	$\begin{matrix} 1 & 1 & 1 \end{matrix}$
P_4	$\begin{matrix} 0 & 0 & 2 \end{matrix}$	$\begin{matrix} 4 & 3 & 1 \end{matrix}$	$\begin{matrix} 0 & 0 & 0 \end{matrix}$

P_0 :- $\begin{matrix} 1 & 0 & 0 \end{matrix}$, P_1 :- $\begin{matrix} 3 & 0 & 2 \end{matrix}$, P_2 :- $\begin{matrix} 3 & 0 & 2 \end{matrix}$, P_3 :- $\begin{matrix} 2 & 1 & 1 \end{matrix}$, P_4 :- $\begin{matrix} 0 & 0 & 2 \end{matrix}$

P_0 :- $\begin{matrix} 1 & 0 & 0 \end{matrix}$, P_1 :- $\begin{matrix} 3 & 0 & 2 \end{matrix}$, P_2 :- $\begin{matrix} 3 & 0 & 2 \end{matrix}$, P_3 :- $\begin{matrix} 2 & 1 & 1 \end{matrix}$, P_4 :- $\begin{matrix} 0 & 0 & 2 \end{matrix}$

P_0 :- $\begin{matrix} 1 & 0 & 0 \end{matrix}$, P_1 :- $\begin{matrix} 3 & 0 & 2 \end{matrix}$, P_2 :- $\begin{matrix} 3 & 0 & 2 \end{matrix}$, P_3 :- $\begin{matrix} 2 & 1 & 1 \end{matrix}$, P_4 :- $\begin{matrix} 0 & 0 & 2 \end{matrix}$

P_0 :- $\begin{matrix} 1 & 0 & 0 \end{matrix}$, P_1 :- $\begin{matrix} 3 & 0 & 2 \end{matrix}$, P_2 :- $\begin{matrix} 3 & 0 & 2 \end{matrix}$, P_3 :- $\begin{matrix} 2 & 1 & 1 \end{matrix}$, P_4 :- $\begin{matrix} 0 & 0 & 2 \end{matrix}$

P_0 :- $\begin{matrix} 1 & 0 & 0 \end{matrix}$, P_1 :- $\begin{matrix} 3 & 0 & 2 \end{matrix}$, P_2 :- $\begin{matrix} 3 & 0 & 2 \end{matrix}$, P_3 :- $\begin{matrix} 2 & 1 & 1 \end{matrix}$, P_4 :- $\begin{matrix} 0 & 0 & 2 \end{matrix}$

P_0 :- $\begin{matrix} 1 & 0 & 0 \end{matrix}$, P_1 :- $\begin{matrix} 3 & 0 & 2 \end{matrix}$, P_2 :- $\begin{matrix} 3 & 0 & 2 \end{matrix}$, P_3 :- $\begin{matrix} 2 & 1 & 1 \end{matrix}$, P_4 :- $\begin{matrix} 0 & 0 & 2 \end{matrix}$

~~now check there any safe sequence exist or not. if exists the request of P_1 will be granted!~~

Yes there is a safe sequence

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

\therefore so the request of P_1 will be granted.

<u>Process</u>	<u>Allocation</u>	<u>Need</u>	<u>Available</u>
	A B C	A B C	5 3 2
P_1	3 0 2	0 1 0	7 4 3
P_3	2 1 1	0 1 1	7 5 3
P_0	0 0 0	0 1 0	7 5 3
P_2	1 3 0	2 0 0	10 5 5
P_4	0 0 2	0 4 3	10 5 7

P₄(3, 3, 0)

Diagram-1

continue
from last
ans table
2, 3, 0

Allocation P₄ :-

$$\begin{array}{r} 0 \ 0 \ 0 \\ + 3 \ 3 \ 0 \\ \hline 3 \ 3 \ 2 \end{array}$$

need P₄ :-

$$\begin{array}{r} 4 \ 3 \ 1 \\ - 3 \ 3 \ 0 \\ \hline 1 \ 0 \ 1 \end{array}$$

available :-

$$\begin{array}{r} 2 \ 3 \ 0 \\ - 3 \ 3 \ 0 \\ \hline 1 \ 0 \ 0 \end{array}$$

∴ available \leq request

∴ so we find the system
is unsafe

∴ we rolled back to old state

P₀ (0, 2, 0)

unsafe
=
continue
from last
table

allocation :-

$$\begin{array}{r} 0 \ 1 \ 0 \\ + 0 \ 2 \ 0 \\ \hline 0 \ 3 \ 0 \end{array}$$

need :-

$$\begin{array}{r} 7 \ 4 \ 3 \\ - 0 \ 2 \ 0 \\ \hline 7 \ 2 \ 3 \end{array}$$

available :-

$$\begin{array}{r} 2 \ 3 \ 0 \\ - 0 \ 2 \ 0 \\ \hline 2 \ 1 \ 0 \end{array}$$

<u>process</u>	<u>Allocation</u>	<u>need</u>	<u>available</u>
P ₀	0 3 0	7 2 3	2 1 0
P ₁	3 0 2	0 2 0	2 1 0
P ₂	3 0 2	6 0 0	2 1 0
P ₃	2 1 1	0 1 1	2 1 0
P ₄	0 0 2	4 3 1	2 1 0

<u>Process</u>	<u>Allocation</u>	<u>need</u>	<u>available</u>
P ₁	3 0 2	4 2 0	2 1 0
P ₃	2 1 1	0 1 1	5 1 2
P ₀	0 3 0	7 2 3	7 2 3
P ₂	3 0 2	6 0 0	10 5 5
P ₄	0 0 2	4 3 1	10 5 7

<u>Process</u>	<u>Alloc</u>	<u>need</u>	<u>available</u>	<u>available</u>
P ₀	0 3 0	7 2 3	—	2 1 0 Not found enough resources
P ₁	3 0 2	0 2 0	—	11
P ₂	3 0 2	6 0 0	—	11
P ₃	2 1 1	0 1 1	—	11
P ₄	0 0 2	4 3 1	—	11 (No safe sequence)

Pretend the allocated resource to P_i by
modifying the state as follows:

$$\text{available} = \text{available} - \text{request}$$

$$\text{allocation} = \text{allocation} + \text{request}$$

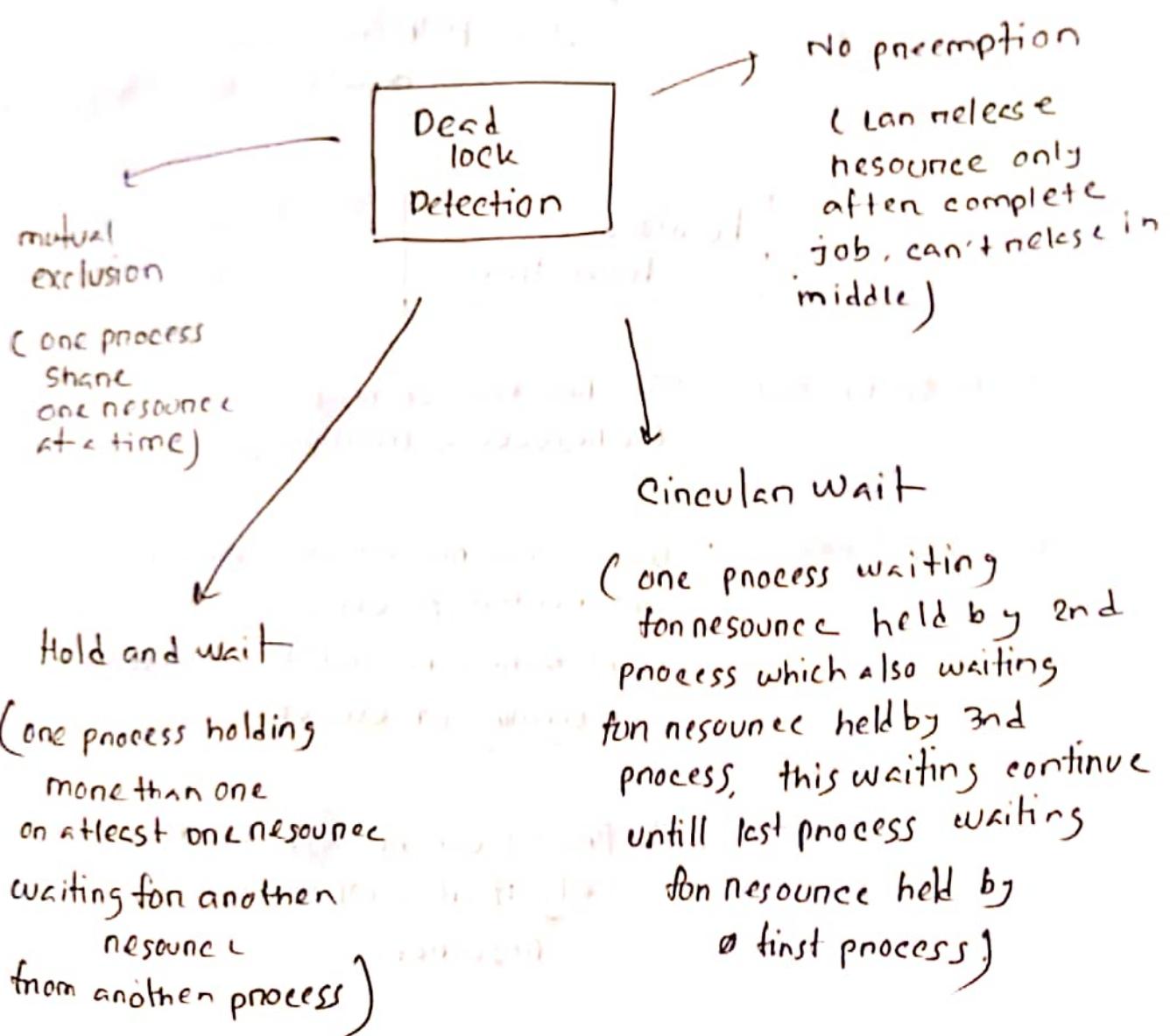
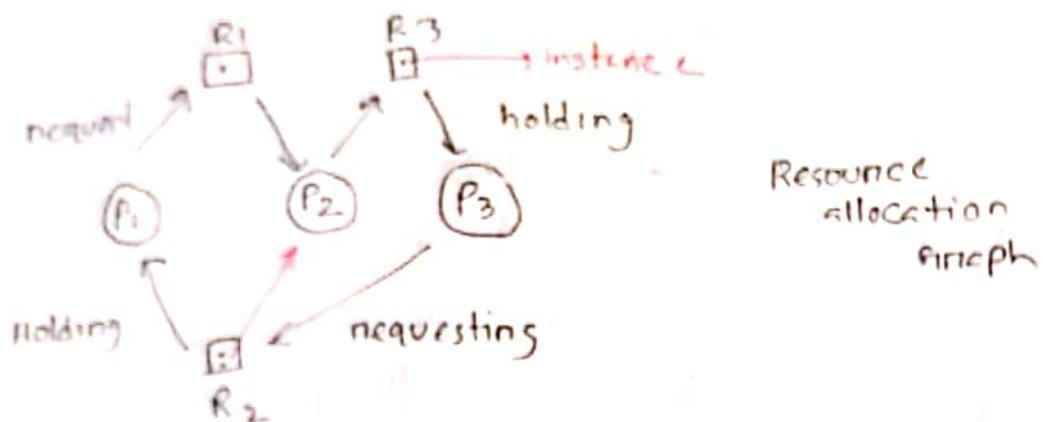
$$\text{Need} = \text{need} - \text{request}$$

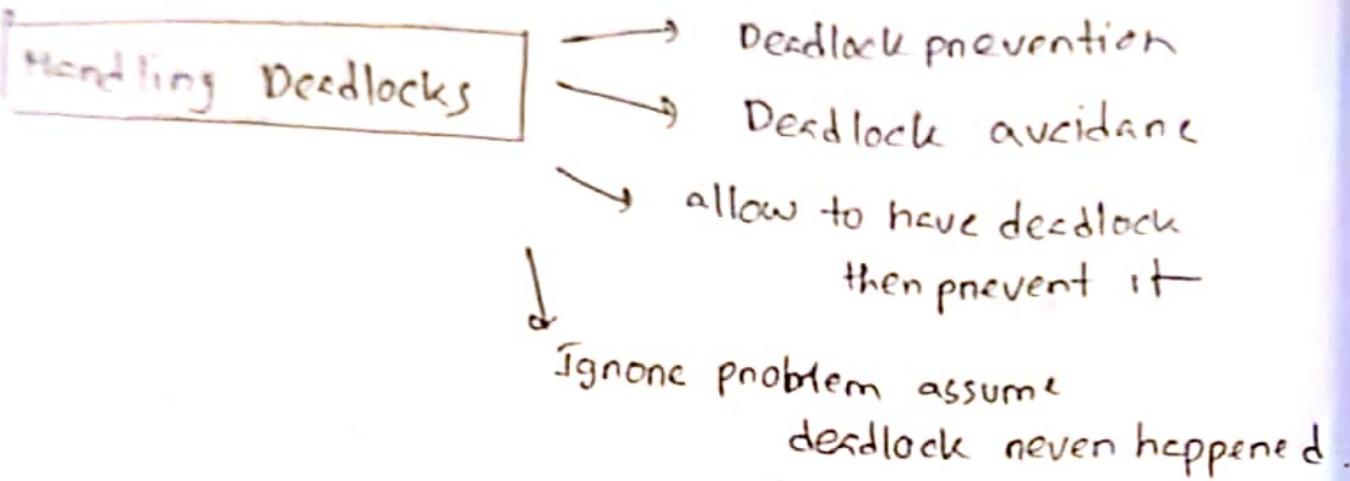
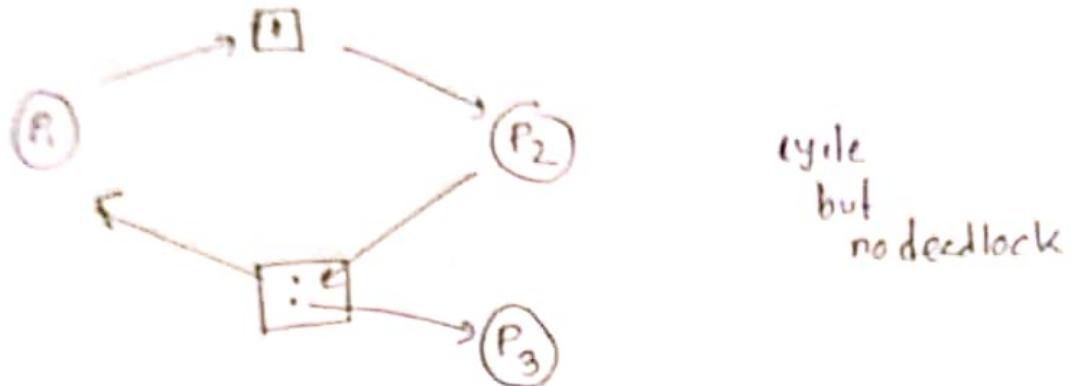
\therefore if safe \rightarrow then resource allocated to P_i ,

\therefore in unsafe $\rightarrow P_i$ must wait, old resource
- allocation state is
restored.

$$\therefore \text{complexity} = O(n^2m)$$

Deadlock is a situation that occurs in OS when any processes enter a waiting state [because] another waiting process holding the demanded resource.





Deadlock Prevention

② (mutual Exclusion) make resource shareable
one resource → multiple process

③ (Hold & wait) ① Process will request what resource one needed for execution at first and all of resource will be allocated at the beginning of execution

(ii) Process can request for resource only if it is not holding any resource.

③ (No preemption) if process request for any resource first it have to release all resource it is holding / allocated

④ (Circular wait) Order resource and process have to request for resource in an increasing number.

A state of system is called safe system if the system can allocate all its resources requested by all the process in system without entering deadlock.

$\text{requested} \leq \text{available}$ resource \rightarrow safe

$\text{request} > \text{available}$ \rightarrow unsafe \rightarrow possibility of deadlock

deadlock avoidance

single instance

Resource allocation graph

multiple instances

Banker's Algorithm

- temporarily accept the request
- check any loop on safe/unsafe
- if unsafe decline the request

$$\text{allocation} = \text{allocation} + \text{request}$$

$$\text{available} = \text{available} - \text{request}$$

$$\text{need} = \text{need} - \text{request}$$

after that if we found a safe sequence

→ accept the request

if unsafe → process

have to wait
and old resource allocation state is recovered.

Pretend the allocated resource to P_i by modifying the state as follows:

$$\text{available} = \text{available} - \text{request}$$

$$\text{allocation} = \text{allocation} + \text{request}$$

$$\text{Need} = \text{need} - \text{request}$$

\therefore if safe \rightarrow then resource allocated to P_i ,

\therefore in unsafe $\rightarrow P_i$ must wait, old resource allocation state is restored.

$\therefore \text{complexity} = O(n^2m)$

deadlock

avoidance

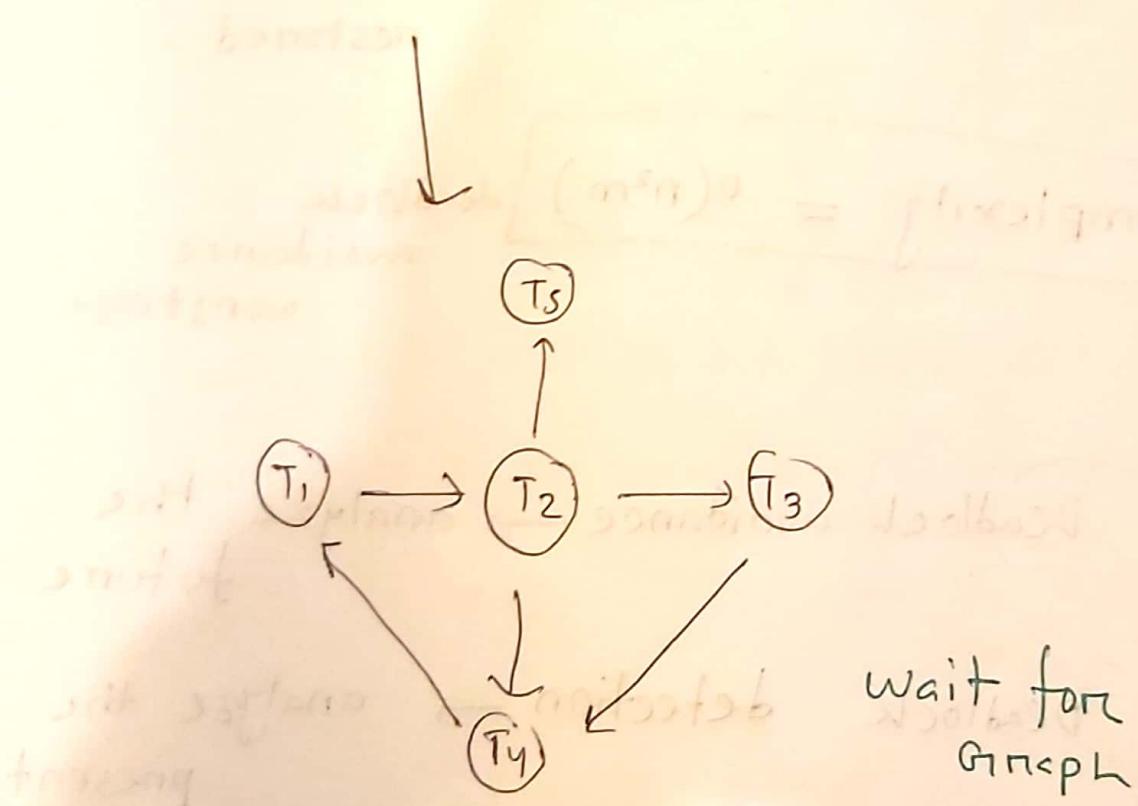
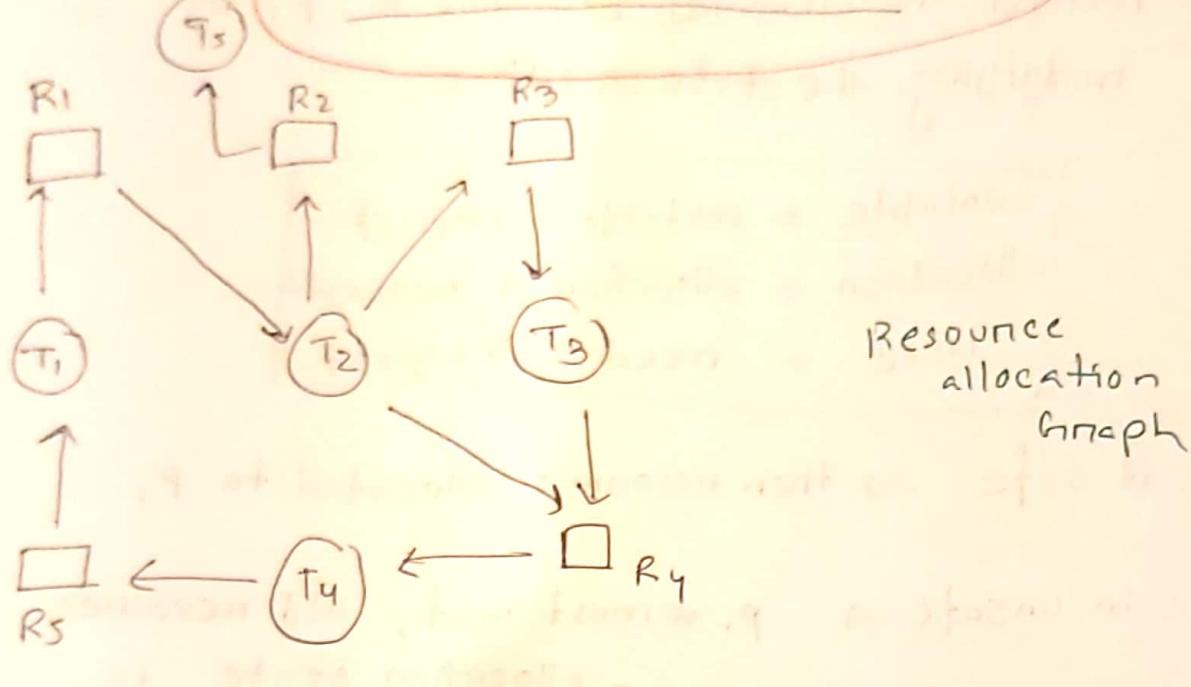
worst case

Deadlock avoidance \rightarrow analyze the future

Deadlock detection \rightarrow analyze the present

avoidance
is better

Deadlock detection



Q Five Processes P₀ to P₄

three resource type :- A(7 instance)

B(2 ")

C(6 ")

Process	Allocation			Request			Available		
	A	B	C	A	B	C	A	B	C
P ₀	0	1	0	0	0	0	0	0	1
P ₁	2	0	0	2	0	2			
P ₂	3	0	3	1	0	0	1	0	0
P ₃	2	1	1	1	0	0	1	0	0
P ₄	0	0	2	0	0	2			

∴ check if there is any deadlock?

Ans:-

<u>Process</u>	<u>available</u>
P ₀	0 0 0
P ₂	3 1 3
P ₁	5 1 3
P ₃	7 2 4
P ₄	7 2 6

∴ Hence available is 0 0 0. But we can see P₀ request for 0 0 0, so P₀ can finish its and after its done its release its allocated resource 0 1 0.

∴ Now available resource

$$\begin{array}{r} \text{A} \quad \text{B} \quad \text{C} \\ 0 \quad 0 \quad 0 \\ +0 \quad +1 \quad +0 \\ \hline 0 \quad 1 \quad 0 \end{array}$$

∴ Now P₂ request \leq available

$$0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0$$

P_2 start executing & then release its resources.

$$\begin{array}{c} \text{A} \quad \text{B} \quad \text{C} \\ \text{now available} \quad 0 \quad 1 \quad 0 \\ \hline 1 \quad 3 \quad 1 \quad 0 \quad + \quad 3 \\ \hline 3 \quad 1 \quad 3 \end{array}$$

\therefore Now P_1 request \leq available
 $(2, 0, 0)$

$\therefore P_1$ executing & then release its allocated resource

$$\begin{array}{c} \text{A} \quad \text{B} \quad \text{C} \\ \text{now available} \quad 3 \quad 1 \quad 3 \\ \hline 2 \quad 0 \quad 1 \quad 0 \\ \hline 5 \quad 1 \quad 3 \end{array}$$

$\therefore P_3$ request \leq available
 $(2, 1, 1)$

$\therefore P_3$ start executing then release its resources.

$$\begin{array}{c} \text{A} \quad \text{B} \quad \text{C} \\ \text{Now available} \quad 5 \quad 1 \quad 3 \\ \hline 2 \quad 1 \quad 1 \\ \hline 7 \quad 2 \quad 4 \end{array}$$

\therefore now P_4 request \leq available
 $0 \quad 0 \quad 2$ $7 \quad 2 \quad 4$

$\therefore P_4$ start executing.

\therefore now available =

A	0	2
7	2	4
0	0	2
<hr/>		
7 2 6		

\therefore so we found a safe sequence

$$= \langle P_0, P_2, P_1, R_3, P_4 \rangle$$

\therefore so there is no deadlock

Q

<u>Process</u>	<u>Allocation</u>			<u>Request</u>	<u>available</u>	
	A	B	C	A	B	C
P ₀	0	1	0	1	0	0
P ₁	2	0	0	0	2	0
P ₂	3	0	3	0	0	1
P ₃	2	1	1	1	0	0
P ₄	0	0	2	0	0	2

∴ check if there is a deadlock?

Ans:

<u>Process</u>	<u>available</u>
allocation	A B C 0 0 0
P ₀	0 1 0
(P ₁)	Not found enough/available resource to start
(P ₂)	"
(P ₃)	"
(P ₄)	"

\therefore Hence available $0, 0, 0$

P_0 request \leq available
 $(0, 0, 0)$

$\therefore P_0$ starts executing & then after finish it releases its resource allocated.

\therefore now available $\begin{array}{ccc} A & B & C \\ 0 & 0 & 0 \\ +0 & +1 & +0 \\ \hline 0 & 1 & 0 \end{array}$

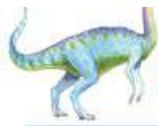
\therefore now P_1, P_2, P_3, P_4 request $>$ available.

so did not find any safe sequence to complete process

\therefore The system is unsafe

So,

\therefore There is a deadlock in the system.



Recovery from Deadlock: Resource Preemption

- **Selecting a victim** – minimize cost
- **Rollback** – return to some safe state, restart process for that state



Recovery from Deadlock: Process Termination

- Abort all deadlocked processes
- Abort one process at a time until the deadlock cycle is eliminated
- In which order should we choose to abort?
 1. Priority of the process
 2. How long process has computed, and how much longer to completion
 3. Resources the process has used
 4. Resources process needs to complete
 5. How many processes will need to be terminated
 6. Is process interactive or batch?