Deadlock

Deadlock concepts :-

→ Two or more processes are said to be in deadlock if and only if they wait for the happening of an event which will never happen.

Consequences :- → Throughput / Efficiency drops

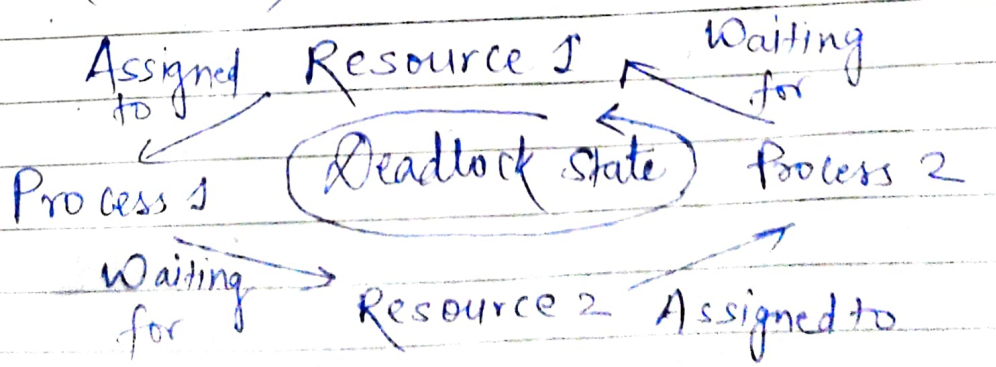
→ Ineffective utilization of resources

→ Deadlock is therefore undesirable

Deadlock V/s **Starvation**

↓ Infinite blocking of Processes (forever)

↓ (Indefinite Blocking)

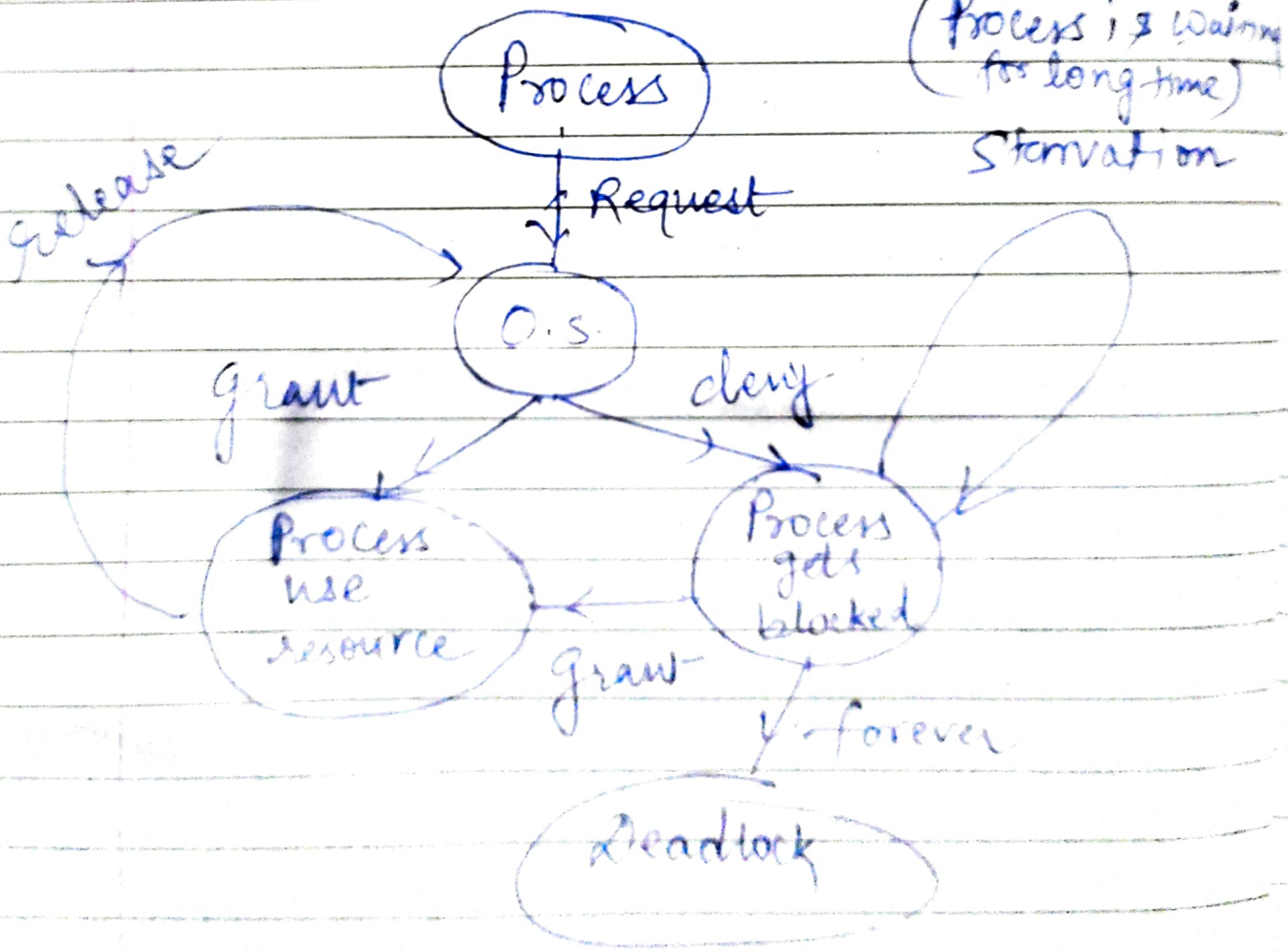


System Model :

→ n : no. of Processes
$\langle P_1 \dots P_n \rangle$

→ 'm' : Resources (H/W / S/W)
$\langle R_1 \dots R_m \rangle$

- Single instance
- Multi instance (copies)

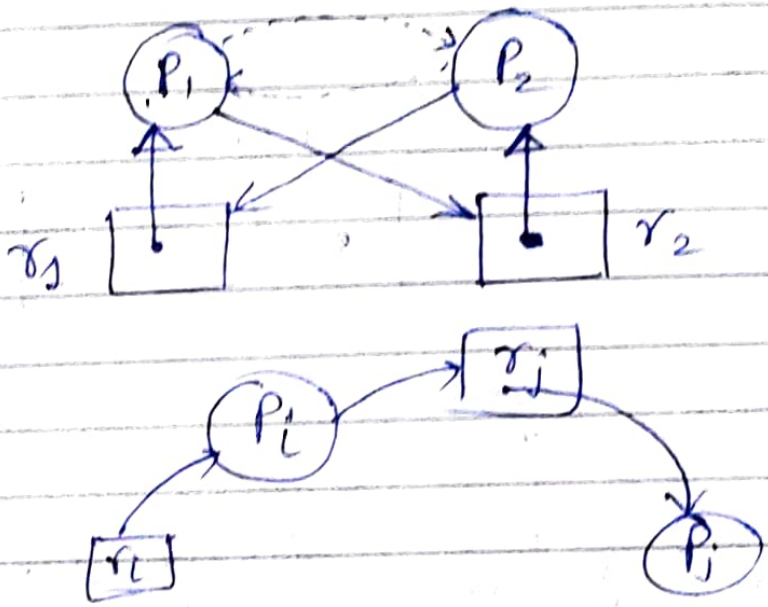


Deadlock Characterization

I. Necessary Condition :

a) Mutual exclusion :
   └ shared resources
   <critical section>

b) Hold & Wait :

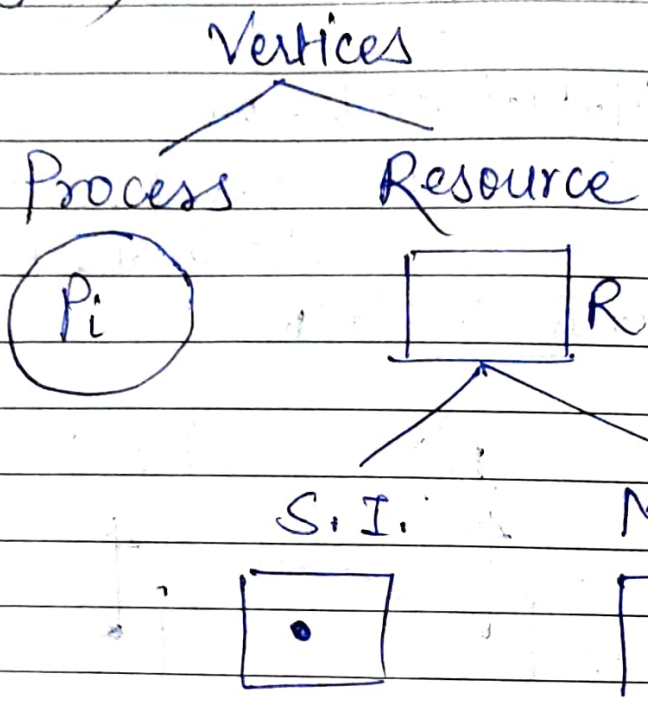


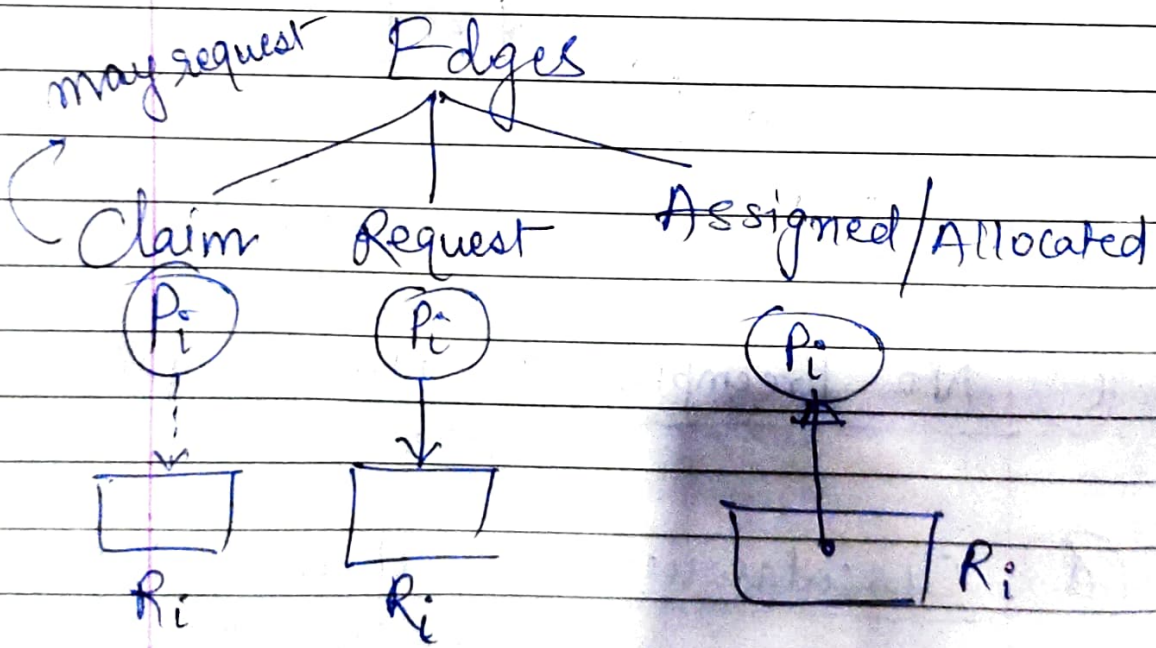
(c) No Preemption :- No Preemption of resources

(d) Circular wait :- whenever there is a cycle then deadlock may or liable to happen

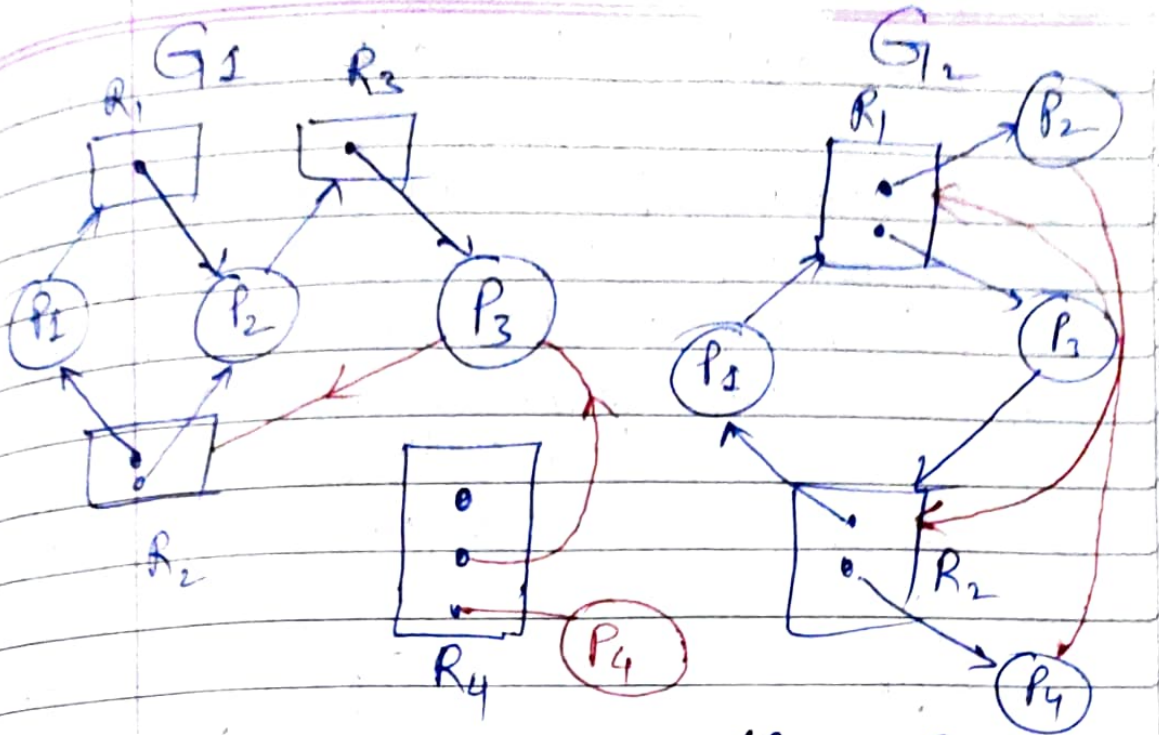
Resource Allocation Graph:-

G(V, E)





R.A.G is multigraph



G1

P_1 & P_2 are blocked

P_3 is not blocked

P_3 is runnable on CPU

Process which is not blocked are runnable

$(P_1 - R_1 - P_2 - R_3 - P_3 - R_2 - P_1)$ Cycle

Deadlock Handling strategies

1. Deadlock Prevention } Type 1
2. Deadlock avoidance }
3. Deadlock detection & Recovery } Type 2
4. Deadlock Ignorance. (No strategy) }

Type 1 :- Deadlock Never occurs.

Type 2 :- Deadlock definitely occurs.

Deadlock avoidance is known as Banker's Algorithm.

[Dijkstra Algo] → Implemented by Edwin Dijecktras

Deadlock Ignorance also called Ostrich Algorithm

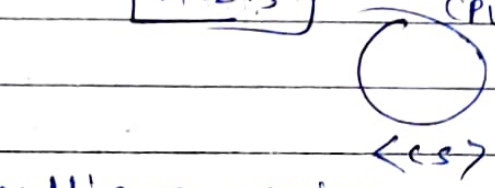
Deadlock Detection & Recovery is also called * (Doctors Algo)

Deadlock Ignorance (Ostrich Algo)
(No strategy)

→ Pretend there is no problem
→ Resonable if
  * Deadlock occurs very rarely
  * Cost of Prevention is high.
→ UNIX and WINDOWS take this approach.
→ It is trade-off between
  * Convinience
  * Correctness

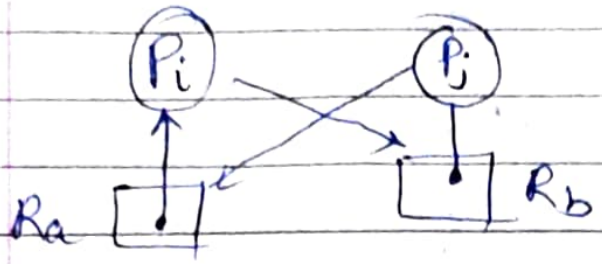
Deadlock Prevention :-
(By dissatisfying/Negating one or more of the necessary condition)

a) Mutual exclusion :- Mutual exclusion Principle is non Compromisable/dissatisfying



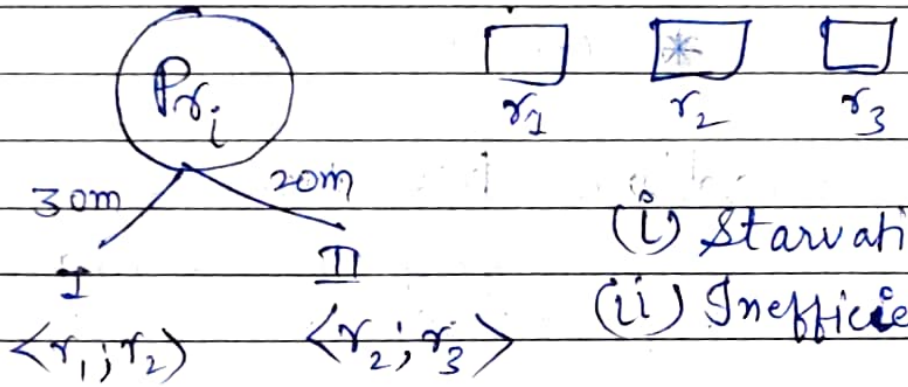
- Since every multiprogramming environment has at least one shared resource therefore Mutual exclusion Non dissatisfiable.

b) Hold and wait :-



!(Hold and wait) = Hold or wait

1) Process must request and be allocated ALL resources prior to its start.

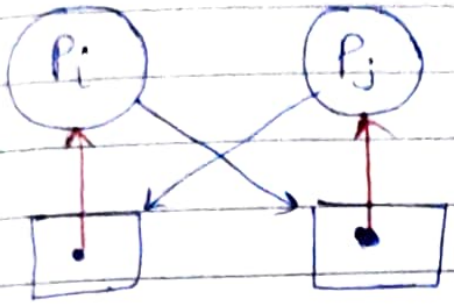


(i) Starvation
(ii) Inefficiency.

2) Process must release all resources before making a fresh/new request;

→ Starvation
→ No efficiency.

3) !(No Preemption) => Preemption of Resources from Processes.



Preemption

forceful (Running Process should not block)

Self. (Selfless) (Let other Process Complete first) (Starvation)

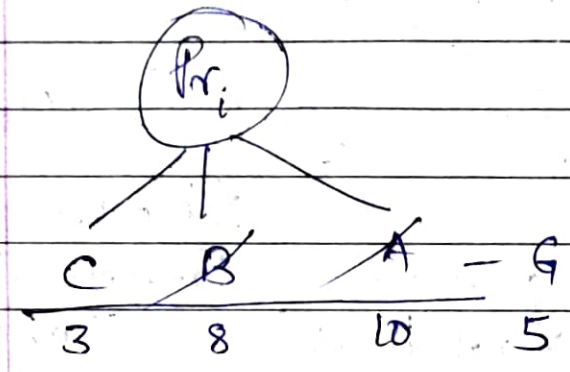
!(No Preemption) = Preemption

4) Circular wait :- is prevented by a total Order ~~return~~ relation among all processes and resources;

I) Assign unique id No's to each Resource;

II) Never allow a Process to request a lower numbered Resource than the last one allocated.

Resources	Res. id
A	10
B	8
C	3
D	4
E	12
P	20
G	5



Resource Precemption (B, & A)

3 - 5 - 8 - 10

< Starvation >

1. If the R.A.G, has resources of only multi-Instance Type, then Cycle is only a Necessary Condition; for deadlock
   Cycle → deadlock
2. If RAG, has resources of both (single and multi-instance) then presence of Cycle is only a necessary Condition; for deadlock

   Cycle ⟶ Deadlock

3. If R.A.G. Resources of only single Instance type, then cycle is a necessary and sufficient Condition for deadlock;

Cycle ⟹ Deadlock

Deadlock Prevention

→ To design a system in such a way that the possibility of deadlock is excluded a priori.

→ Prevention Philosophy: We know what the Preconditions are; so Prevent one or more these from occuring.

→ for Example: Circular wait can be Prevented by linear Ordering of resource Types. If a process holds resources of type R_j. then it can request resource type R_K, $K>1$, but not R_i where $i<=j$. Similarly, any other process holding R_i can request R_j but a process holding R_j can not request R_i.

III Deadlock Avoidance

1) Single instance Resources (Resource allocation graph algorithm)

2) Multi instance Resources (SI+MI) (Bankers Algo)
   (i) Safety algo.
   (ii) Resource Request Algo.

Note:- Both the algorithm are based on Apriori knowledge/information.

1. Resource Allocation Graph Algo.

(Single instance Resources)

System State

- Safe → (No deadlock)
- Unsafe → Danger of Deadlock (warning)

* * Banker's Algo (Multiinstance) Dijkstra (Avoidance):-

(i) Safety Algorithm
(ii) Resource request Algo.
(Apriori Information)
Data Structures (System state)

1. n: no. of Process ($P_1 \dots P_n$)
2. m: no. of resources ($R_1 \dots R_m$)
3. Maximum $[1 \dots n, 1 \dots m]_{n \times m}$

$Max[i,j] = K$

$P_i \xrightarrow[max]{} K(R_j)$

Process P_i demands (apriori) 'K' Copies of $[R_j]$;

4. Allocation $[1 \dots n, 1 \dots m]_{n \times m}$

$Alloc[i,j] = a$

$P_i \xleftarrow[alloc]{} a[R_j] \quad [a \leq K]$

5. Need $[1 \dots n, 1 \dots m]_{n \times m} \Rightarrow [Max - Alloc.]$

$Need[i,j] = b$

$P_i \xrightarrow[Need]{} b(R_j) \quad [b = K - a]$

6) Request [1...n, 1...m]$_{n \times m}$

$P_i \xrightarrow{req} C(R_j)$ @ time 't'.

$Req[i,j] = c$

$[c \leq b]$

7) Total [1 ... m]

Total [j] = 'z'.

There are 'z' copies of R_j in the system

8) Available [1 ... m] $[e \leq z]$

Avail [j] = e

There are e(R_j) free available @ time 't'.

$$Avail = Total - \sum_{i=1}^{n} alloc_i$$

Ex 1) n = 5; m = 1, R = 22

Pid	Max R	Alloc R	Need R	Avail R
P_1	10	5	5	~~22~~ 3
P_2	6	4	4	~~2~~
P_3	12	5	7	
P_4	~~5~~	2	3	
P_5	6	3	3	

Safety Algo: System is said to be "safe" iff the need of all processes can be satisfied with the available Resources in some Order; otherwise it is unsafe.

R = 22 Avail = 3

$\langle P_4, P_5, P_1, P_2, P_3 \rangle \Rightarrow$ Safe Sequence

5 ⟨8⟩ ⟨13⟩ ⟨17⟩ ⟨20⟩ ⟨22⟩

We can have multiple safe sequences.

	Allocation			Max			Available		
	A	B	C	A	B	C	3	3	2
P_0	0	1	0	7	5	3			
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			

$\langle A, B, C \rangle = \underbrace{\langle 10, 5, 7 \rangle}_{\text{Total}}$

~~To ∘ ⟨P1, P3,~~

~~⟨5, 3, 2⟩~~

	Available			Need			
	A	B	C		A	B	C
	3	3	2	P_0	7	4	3
P_1	5	3	2	P_1	1	2	2
P_3	7	4	3	P_2	6	0	0
P_4	7	4	5	P_3	0	1	1
P_0	7	5	5	P_4	4	3	1

$T_0 = \langle P_1; P_3; P_4; P_0; P_2 \rangle$

$P_2 \langle 10, 5, 7 \rangle$

System is safe.

Ques After granting the req_i the resulting state should be safe

(ii) Resource Request Algorithm

Algo Res_Request (P_i, Req_i, $Alloc_i$, $Need_i$, Avail)

{

1. $Req_i \leq Need_i$
2. $Req_i \leq Avail$
3. [Assume to have satisfied the Req_i]
   a. $Avail = Avail - Req_i$
   b. $Need = Need_i - Req_i$
   c) $Alloc_i = Alloc_i + Req_i$
4. Run Safety Algo
5. System is safe.

∴ If System is SAFE then grant the req; else deny the req; & the Block the process P_i.

T_1 : P_1 : ⟶

$Req_1 < 1, 0, 2 >$

Bank (Branch)

C

SBI

S.B. : 20L (Max)

Withdrawen : 2L (Alloc.)

Balance : 18L (Need)

Banker

18L Avail case

t_1 : C ⟶ 16L

9:30 AM Cheque

<Req>

P5

$n = 5$; $m = 1$; $R = 21$

Pid	Max R	Alloc R	Need R	Avail R
P_1	10	5	5	2
P_2	8	4	4	
P_3	12	5	7	
P_4	5	2	3	
P_5	6	3	3	

$\langle A, B, C \rangle = \langle 10, 5, 7 \rangle$

System is not Safe

	Allocation A	B	C	MAX A	B	C	Available A	B	C	Need A	B	C
P_0	0	1	0	7	5	3	3	3	2	7	4	3
P_1	2	0	0	3	2	2	$\langle 5$	3	$2\rangle$	1	2	2 $\langle 0$ 2 $0\rangle$
P_2	3	0	2	9	0	2	$\langle 7$	4	$3\rangle$	6	0	0
P_3	2	1	1	2	2	2	$\langle 7$	4	$5\rangle$	0	1	1
P_4	0	0	2	4	3	3	$\langle 7$	5	$5\rangle$	4	3	1
							$\langle 10$	5	$7\rangle$			

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

P_5

T_1: P_1 :→

$Req_1 \langle 1, 0, 2 \rangle$ →

Available	Allocation	
3 3 2	$\langle 3\ 0\ 2\rangle$	T_2 → P_4 :→ ✗
$\langle 2\ 3\ 0\rangle$		Req_4 $\langle 3\ 3\ 0\rangle$
$\langle 5\ 3\ 2\rangle$	$\langle P_1, P_3, P_4, P_0, P_2 \rangle$	
$\langle 7\ 4\ 3\rangle$		
$\langle 7\ 4\ 5\rangle$		T_3 P_0 :- $\langle 0, 2, 0\rangle$
$\langle 7\ 5\ 5\rangle$		
$\langle 10\ 5\ 7\rangle$		

Process	Allocation				Max				Available				Need			
	A	B	C	D	A	B	C	D	A	B	C	D	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					0	7	5	0
P_2	1	3	5	4	2	3	5	6					1	0	0	2
P_3	0	6	3	2	0	6	5	2					0	0	2	0
P_4	0	0	1	4	0	6	5	6					0	6	4	2

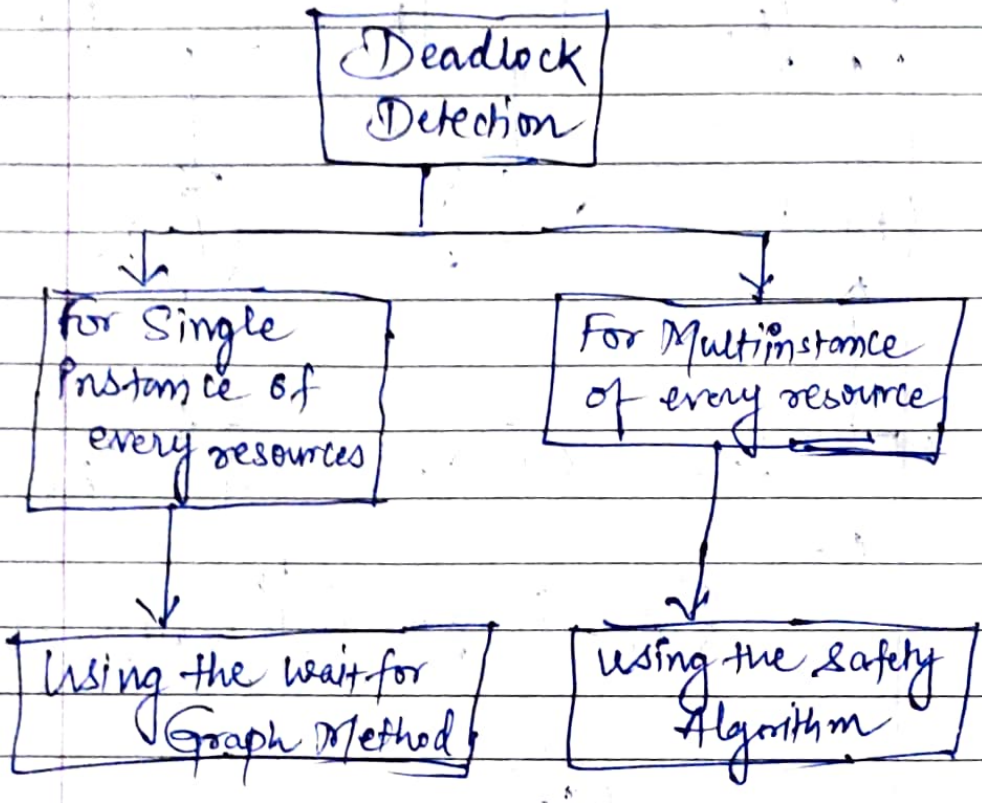
$\langle P_0, P_2, P_3, P_4, P_1 \rangle$

<1 5 3 2>
<2 8 8 6>
<2 14 11 8>
<2 14 12 12>
<3, 14 12 12>

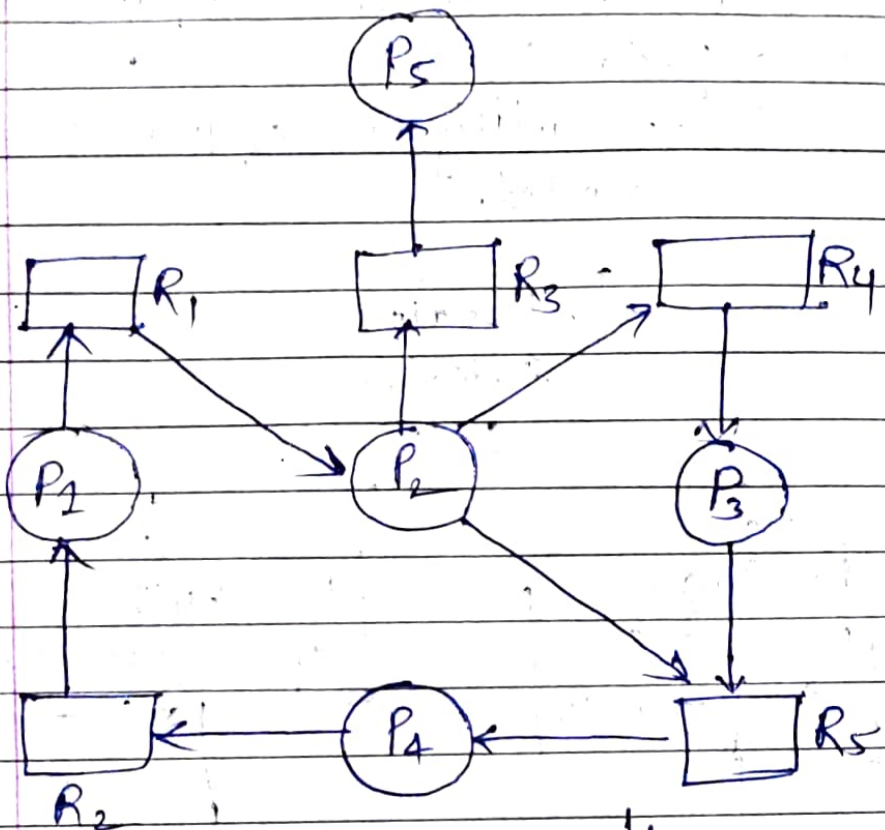
8 (iv) Deadlock Detection & Recovery :-

(When to activate apply Detection Algorithm)

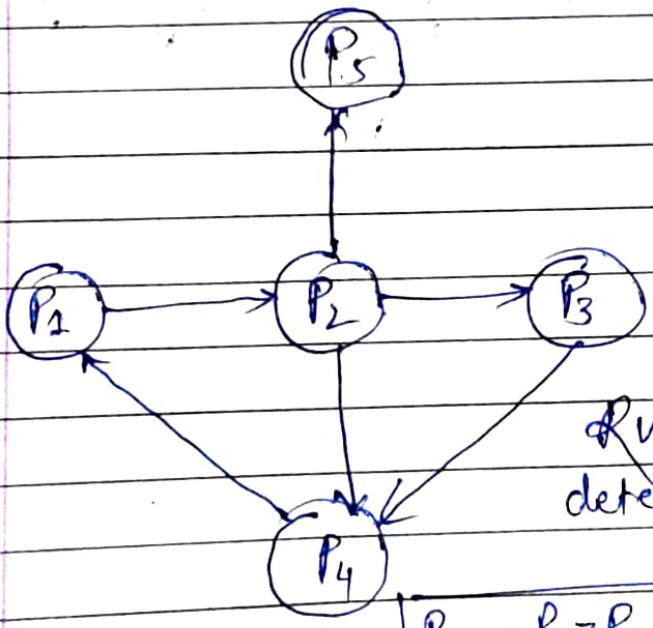
→ Under utilization of processes are blocked.



I. Deadlock detection for Single Instance Resource :-



Ex. Graph G ⟹ wait for graph



Run Cycle detection Algo.

$P_1 - P_2 - P_3 - P_4 - P_1$

Cycle → Deadlock

2) Deadlock detection with multi instance Resources (Safety Algo. Approach)

<ABC> : <7, 2, 6> Total

	Allocation			Request			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	0	0	0	0	0	0
P_1	2	0	0	2	0	2	<0 1 0>		
P_2	3	0	3	0	0	0	<3 1 3>		
P_3	2	1	1	1	0	0	<5 2 4>		
P_4	0	0	2	0	0	2	<5 2 6>		
							<7 2 6>		

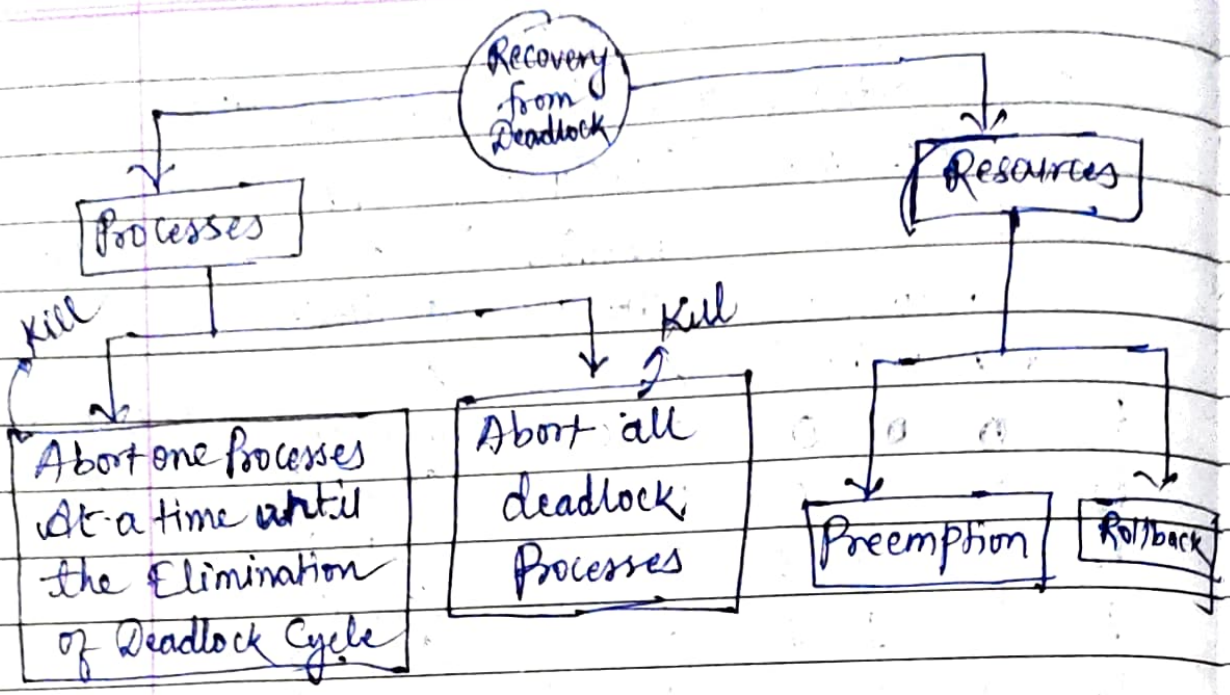
To: <P_0, P_2, P_3, P_4, P_1>

Note :- System is safe if and only if the Request of all processes are satisfiable with available copies in some order otherwise it is deadlock.

T_1 : (P_2) : <0, 0, 1>

Blocked <P_0, deadlock>

P_5



1. Prevention
2. Avoidance } C_1 Proactive
3. Detection & Recovery } C_2 Reactive
4. Ignorance } C_3 Inactive

Conceptual Problems

(1) Consider a System having 'n' processes and a single 'R' having '6' copies; each proces need 2 copies of R to complete.

(a) what is the min(n) to cause deadlock? 6

(b) what is the max(n) for deadlock freedom. 5

$R = 6$, $P_i \rightarrow 2(R)$

P_1	1	P_1	1
P_2	1	P_2	1
P_3	1	P_3	1
P_4	1	P_4	1
P_5	1	P_5	1
P_6	1	P_6	

Blocked

Q. $n = 3$, Process (P_1, P_2, P_3)

$R =$

$P_i \rightarrow 2(R)$ (3)

a) what is max(R) to Cause Deadlock.

b) what is min(R) ensure Deadlock freedom? (4)

$P_1 - 1$

$P_2 - 1 \quad 1$

$P_3 - 1$

P_5

Q. n - Process

$R = 6$ Copies

$P_i \rightarrow 3(R)$ 3

a) what is min(n) to Cause deadlock

b) what is max(n) for deadlock freedom. 2

$P_1 - 2$

$P_2 - 2$ } Deadlock

$P_3 - 2$

(Q4)

	max Demand R	
P_1	10 — 9	$n = 5$
P_2	8 — 7	R
P_3	5 — 4	
P_4	12 — 11	
P_5	6 — 5	36

a) What is Max(R) to cause Deadlock: 36

b) Min(R) for deadlock freedom

37

<1 - - 36>

A system is having 3 user processes each requesting 2 unit of resources 'R'. The minimum number

$3 \times 2 = 6$ ✓

	P_1	P_2	P_3	
"2"	1	1		
"3"	1	1	1	(1)
"4"	1	1	1	

max no. of resources + 1

P_1 P_2 P_3

2 3 4

(1) (2) 3 (6) + 1

P_1 P_2 - - - - P_n

x_1 x_2 - - - - x_n

(x_1-1) (x_2-1) - - - - - (x_n-1)

$$\left(\sum_{i=1}^{n} x_i\right) - n + 1$$

R = 6 ; P_i = 2

P_1^1 P_2^1 P_3^1 P_4^1 P_5^1 | P_6^1

R = 6 P_i = 3

P_1 (2) P_2 (2) | P_3 (2)

R = 100 P_i = 2

(100) Process → Deadlock

(99) ✓

R = 100 P_i = 3

$n \times 2 = 100$

$n = (50)$ ✓

max. no. of process which has no deadlock = (49)

R = 100 P_i = 4

$n \times 3 = 100$

$n = \lceil \frac{100}{3} \rceil = 34$

$33 \times 3 = 99$

$1 \times 1 = 1$

100

R: 100 P_i : 5

$n \times (4) = 100$

What is the minimum no. of Process that has to be present in System because there could be a deadlock

$n \times 4 = 100$

$n = 25$

(Q4)

10P , P_i = 3

R = ?

$10 \times 2 = 20$ (max no of Process which is in deadlock)

10P P_i = 3

R = ?