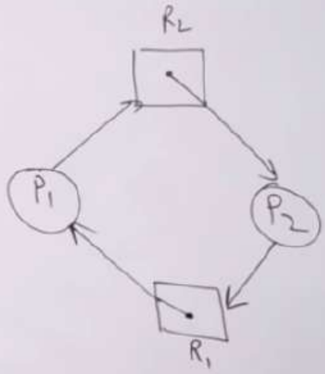
Part 4.1 Deadlock in operating system in hindi basic idea concept definition prevention avoidance

Deadlock

→ In a multiprogramming system, a number of process compete for limitted number of resources and if a resource is not available at that instance then process enters into waiting state

→ if a process unable to change its waiting state indefinitely because the resources requested by it are held by another waiting process. then system is said to be in deadlock.



System model:-

→ Every process will request for the resource

→ if entertained then, process will use the resource

→ Process must release the resource after use

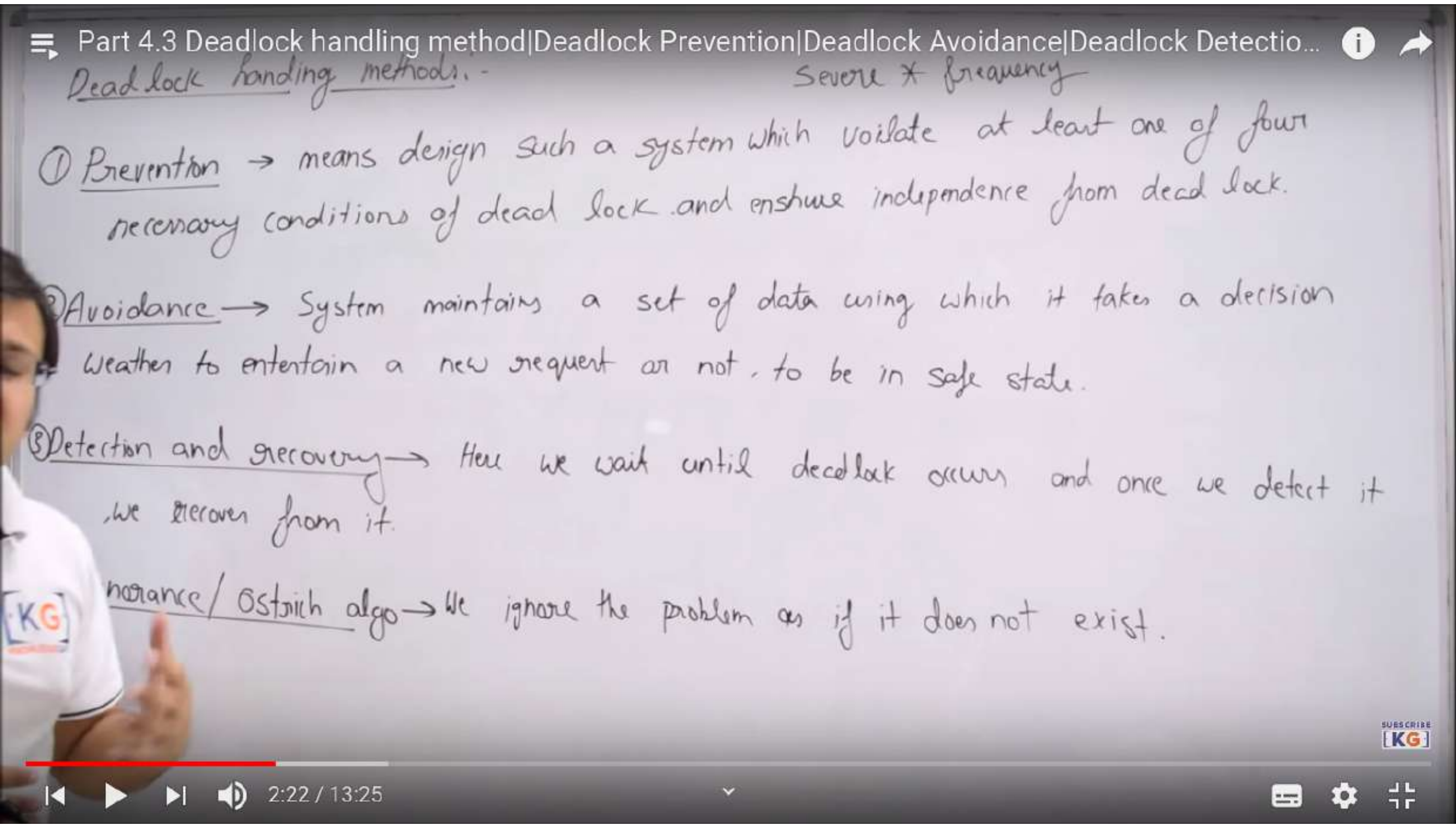
Necessary conditions of Deadlock:-

Mutual exclusion:- At least one resource type in the system which can be used in non-sharable mode i.e. mutual exclusion (one-at a time/one-by one) e.g Printer.

Hold & wait:- A process is currently holding at least one resource and requesting additional resources which are being held by other processes. $P_1 \longrightarrow P_2 \longrightarrow P_3$

No pre-emption:- A resource can not be pre-empted from a process by any other process resource can be released only voluntarily by the process holding it.

Circular wait:- Each process must be waiting for a resource which is being held by another process, which in turn is waiting for the first process to release the resource.


Part 4.3 Deadlock handling method|Deadlock Prevention|Deadlock Avoidance|Deadlock Detectio...
Dead lock handling methods.-
Severe * frequency
① Prevention → means design such a system which voilate at least one of four necessary conditions of dead lock and enshure independence from dead lock.
② Avoidance → System maintains a set of data using which it takes a decision weather to entertain a new request or not, to be in safe state.
③ Detection and recovery → Here we wait until deadlock occurs and once we detect it we recover from it.
norance/ Ostrich algo → We ignore the problem as if it does not exist.
2:22 / 13:25
KG

Part 4.5 Violation of Hold & Wait under deadlock prevention approach operating system

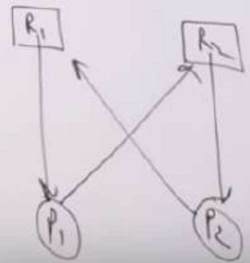
Deadlock prevention :- (Hold and wait)

→ Conservative approach :- Process is allowed to start execution if and only if it has acquired all the resources (Less efficient, not implementable, easy, deadlock independence)

→ Do not hold :- Process will acquire only desired resources, but before making any fresh request it must release all the resources that it currently hold (efficient, implementable)

→ Wait timeout :- We place a maximum time upto which a process can wait. After which process must release all the holding resources & exit.

$R_1 \quad R_4 \; R_5 — R_{10}$

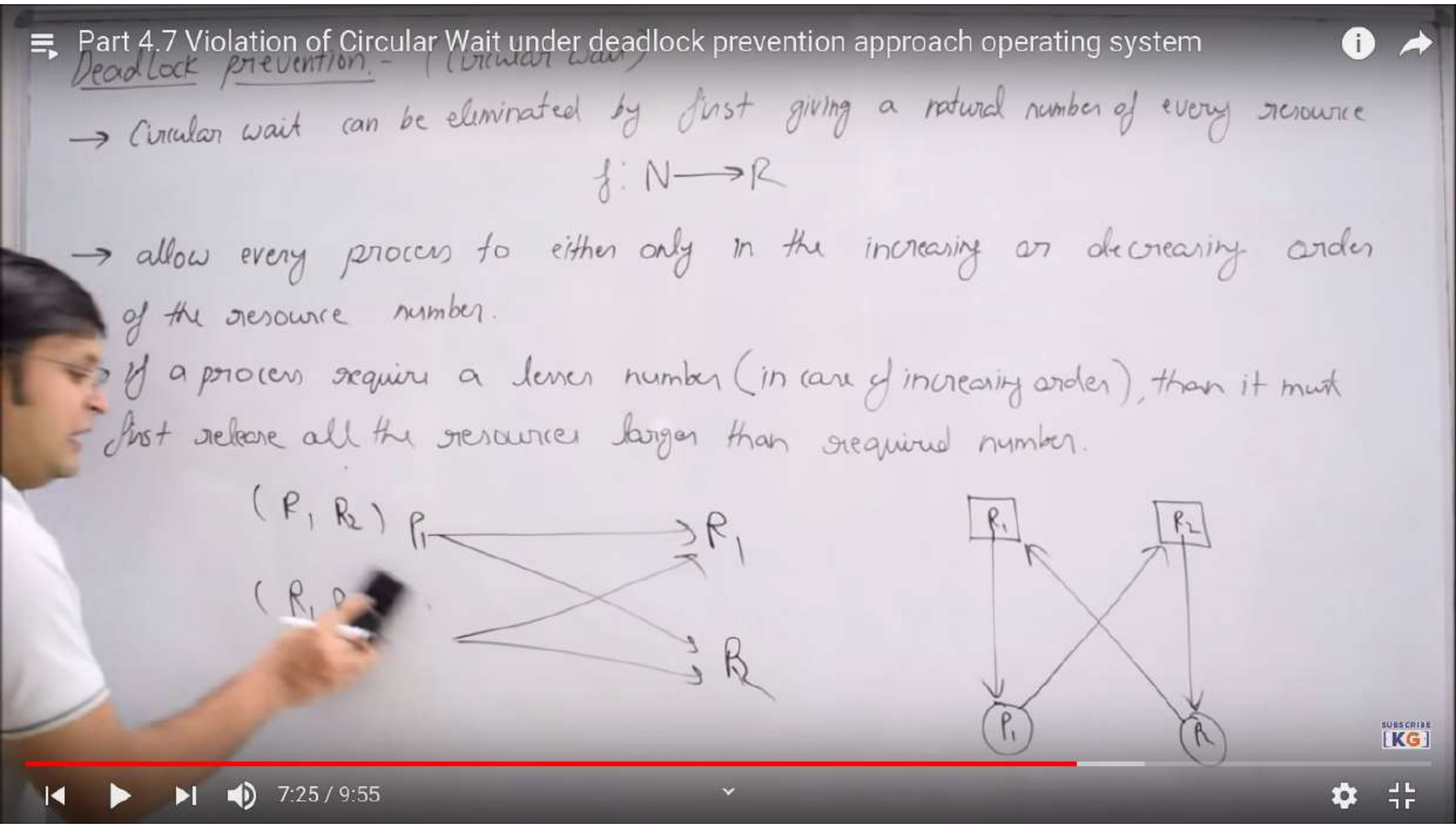


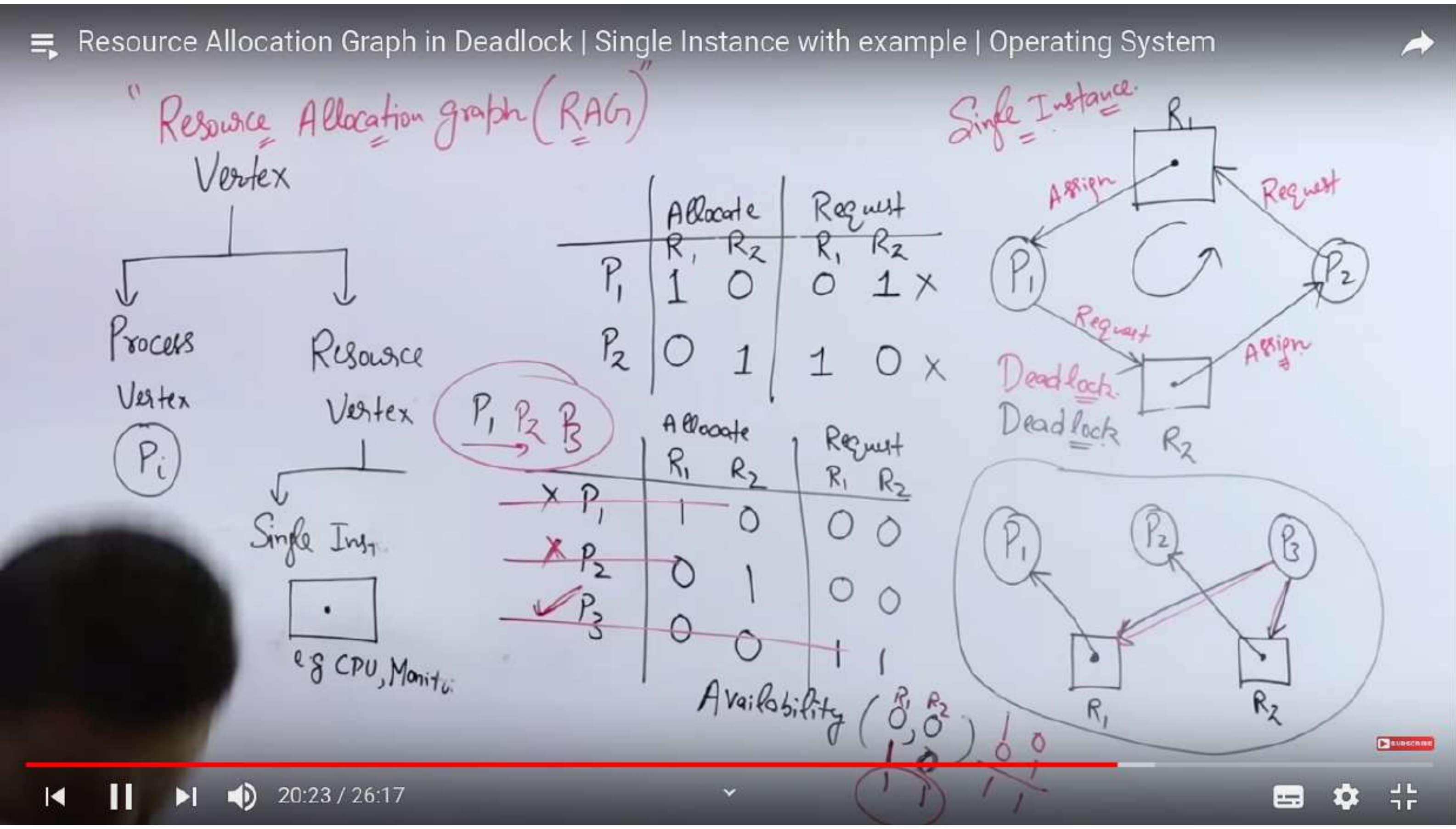
Deadlock prevention:- (No Pre-emption)

Force-full preemption:- We allow a process to forcefully preempt the resource holding by other processes.

→ This method may be used by high priority process or system process.

→ the process which are in waiting state must be selected as a victim insted of process in the running state.


Part 4.7 Violation of Circular Wait under deadlock prevention approach operating system
Deadlock prevention. - (Circular wait)
→ Circular wait can be eliminated by first giving a natural number of every resource
f: N → R
→ allow every process to either only in the increasing or decreasing order of the resource number.
→ If a process require a lesser number (in case of increasing order), then it must first release all the resources larger than required number.
(R1, R2) P1
R1
R2
R1
R2
P1
P2
7:25 / 9:55


Resource Allocation Graph in Deadlock | Single Instance with example | Operating System
"Resource Allocation graph (RAG)"
Vertex
Process Vertex
P_i
Resource Vertex
Single Inst
e.g CPU, Monitor
Allocate
Request
R_1 R_2
P_1 1 0 0 1 X
P_2 0 1 1 0 X
P_1 P_2 P_3
X P_1 1 0 0 0
X P_2 0 1 0 0
✓ P_3 0 0 1 1
Availability (0, 0)
Single Instance
Assign
Request
Deadlock
20:23 / 26:17

"BANKER's Algo"

Total A=10, B=5, C=7

$$\begin{array}{ccc} -7 & -2 & -5 \\ \hline 3 & 3 & 2 \end{array}$$

Deadlock Avoidance.
Deadlock Detection.

Process	Allocation			Max Need			Current Available			Remaining Need = Max - Allocation		
	CPU A	Memory B	Printer C	A	B	C	A	B	C	A	B	C
P_1	0	1	0	7	5	3	3	3	2			
P_2	2	0	0	3	2	2	5	3	2			
P_2	3	0	2	9	0	2	7	4	3			
	2	1	1	4	2	2	7	4	5			
	0	0	2	5	3	3	7	5	5			
	7	2	5									

Safe Sequence.
Unsafe.

$P_2 \rightarrow P_4 \rightarrow P_5 \rightarrow P_1 \rightarrow P_3$