What is Deadlock

Deadlock

PROCESSES WILL WAIT FOR AN EVENT WHICH WILL NEVER OCCUR

Defination

In a multiprogramming environment, several process may compete for a finite number of resources .

A process requests resources , and if the resources are not available at that time , the process enters a waiting state.

Sometimes a waiting process is never again able to change state because the resources it has requested are held by other waiting processes . This situation is called a deadlock.

Example

Consider a system with one printer and DVD drive . Suppose that process *pi* is holding the DVD and process *pj* is holding the printer. if *pi* requests the printer and *pj* requests the DVD drive , a deadlock occurs.

Execution Sequence for a Process

As the process executes it requires resources to be allocated to it.

Under the normal mode of operation, process may utilize a resource only in the following sequence:

- Request
- Use
- Release

Deadlock Characterization

i. Necessary conditions

ii. Resource –Allocation Graph

Necessary Conditions

Mutual exclusion

Hold and Wait

No preemption

Circular wait

Resource-Allocation Graph

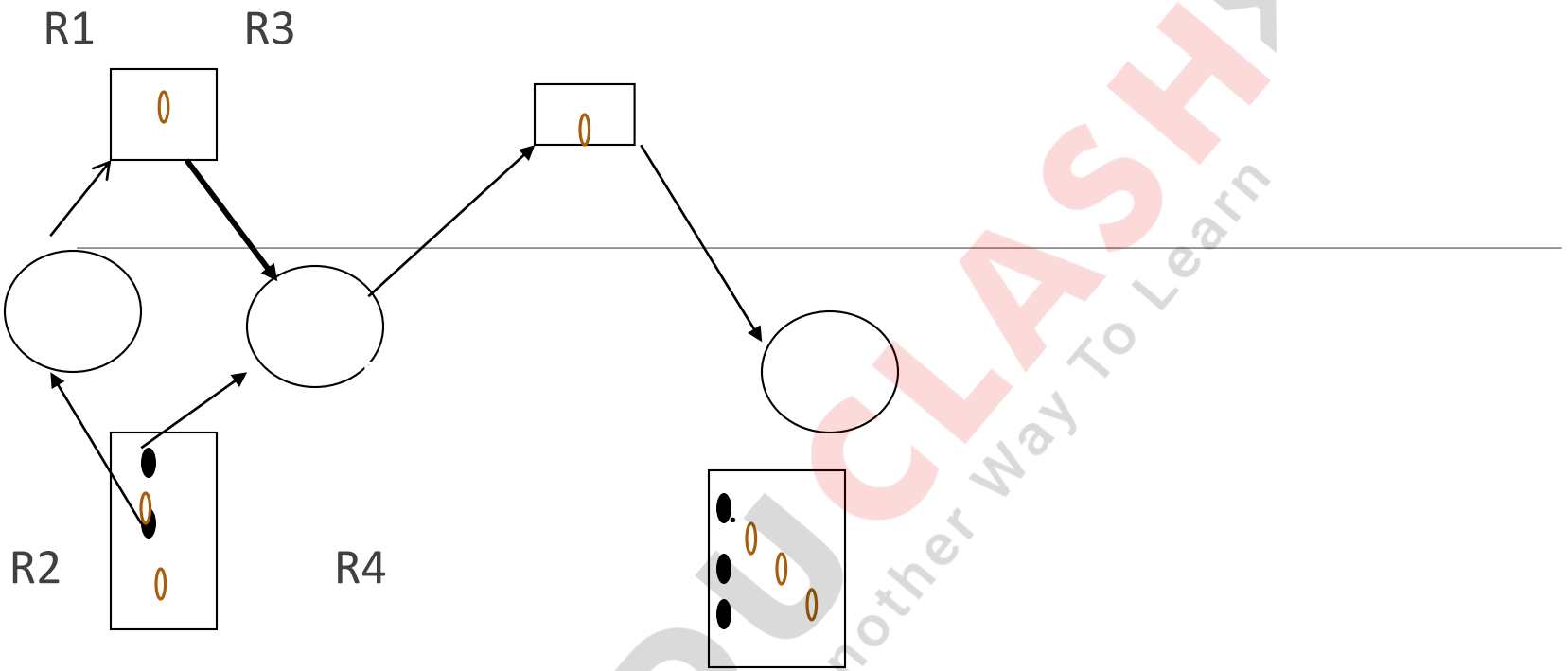
A directed graph which is used to describe deadlock is known as system resource allocation graph.

This graph consists of a set of vertices V and set of edges E.

The set of vertices V is partitioned in to two different types of nodes:

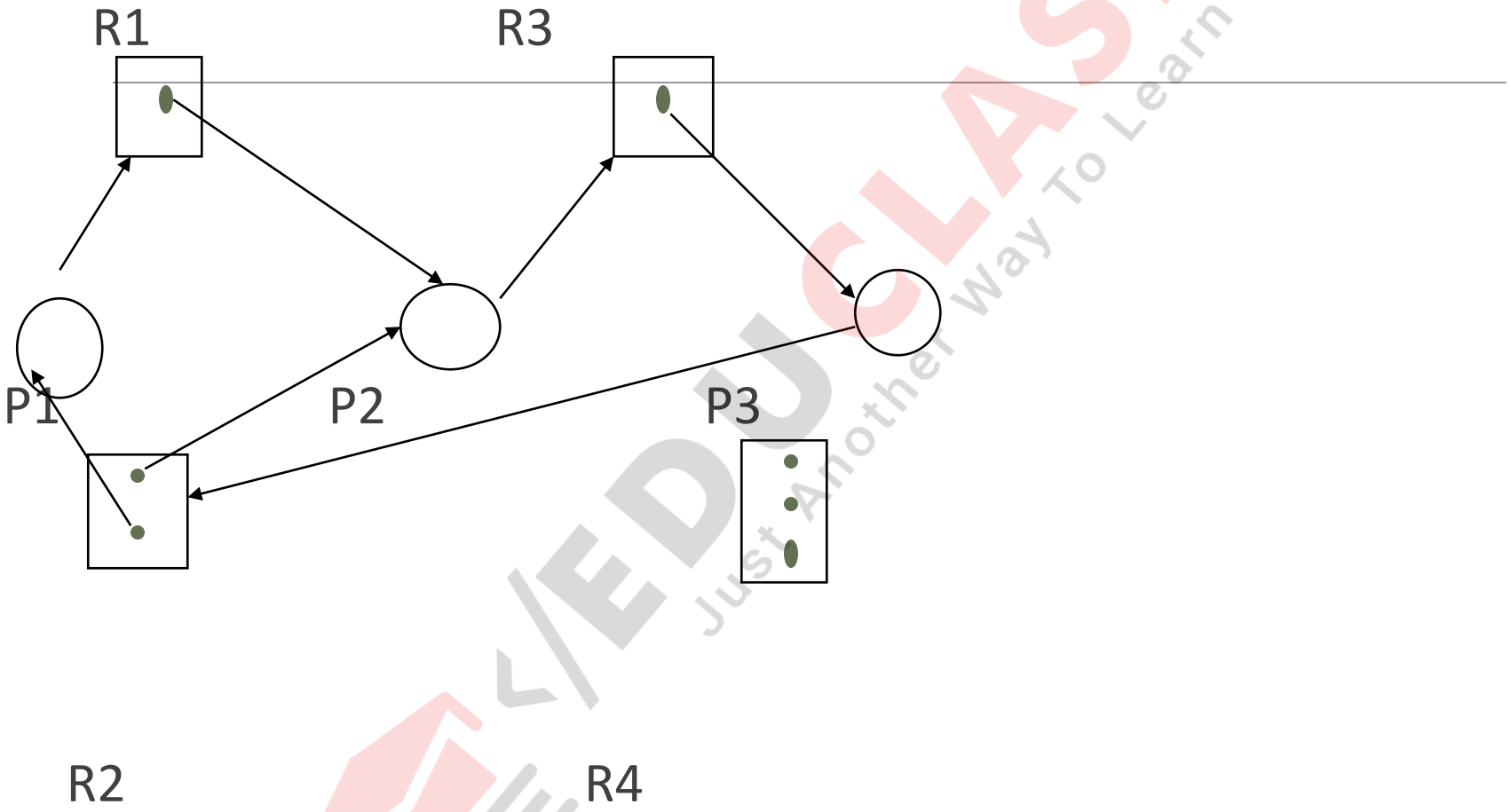
1. P = {P1,P2,……Pn} ,the set consisting of all the active processes in the system.

2. R = {R1,R2,….Rn} ,the set consisting of all resource types in system.

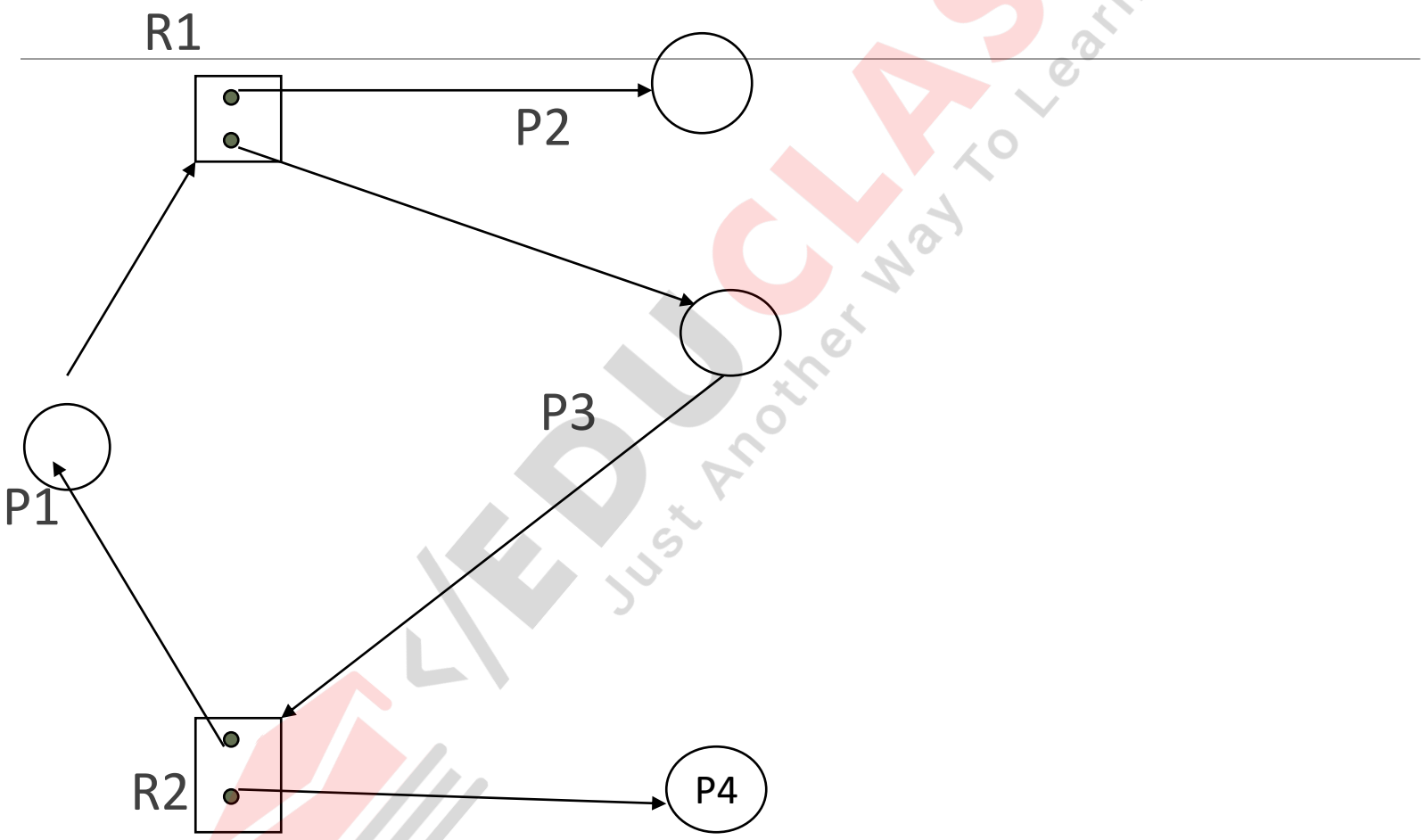


Resource allocation graph

Resource allocation graph with deadlock



Resource allocation graph with a cycle but no deadlock



Methods for handling Deadlock

Can use a protocol to prevent or avoid deadlocks, ensuring that the system will never enter a deadlock state.

Can allow the system to enter a deadlock state, detect it, and recover.

Can ignore the problem altogether and pretend that deadlocks never occur in the system.

Deadlock Prevention

deadlock can prevented by ensuring that at least one of the following conditions can not hold:

- Mutual Exclusion
- Hold and Wait
- No preemption
- Circular Wait

Mutual Exclusion

This condition must hold for non-sharable resources.

Sharable resources in contrast do not require mutually exclusive access

Thus can not be involved in a deadlock.

Eg. Read only files.

Hold & Wait

We must guarantee that, whenever a process requests a resource, it does not hold any other resources.

Protocol:

each process to request & can be allocated all its resources before it begins execution.

Disadvantage: resource utilization may be low.

Resources may be allocated but unused for long time

Protocol:

allows a process to request resources only when it has none.

Disadvantage: starvation is possible.

A process that needs several popular resources may have to wait indefinitely , because at least one of the resources that it needs is always allocated to some other process.

No Preemption

Protocol is to be used:

If a process is holding some resources and requests another that can not be immediately allocated to it.

Then all resources, currently being held are preempted.

No Preemption cont..

The preempted resources are added to the list of resources for which the process is waiting.

Then process will be restarted only when it can regain its old & new resources that it is requesting.

Circular Wait

To impose a total ordering of all resource types and to require that each process requests resources in an increasing order of enumeration.

Protocol:

- Each process can request resources only in an increasing order of enumeration. That is , a process can initially request any number of instances of a resource type.

if several instances of the same resource type are needed,a single request for all of them must be issued.