CSE 330: Operating Systems

Class: 21 **Date:** 11/3

Fall 2016

Note Title

Deadlocks

- happen?
- Semaphore program

$x = 1$
$y = 1$

T_1	T_2
$P(x)$	$P(y)$
$P(y)$	$P(x)$
$\vdots$	

$\rightarrow P(x)$

$\rightarrow P(y)$

$\rightarrow P(y)$

$\rightarrow P(x)$

?

Software solutions $\rightarrow$

Dining Philosophers

→ simple symmetric solution

→ deadlock prone

Deadlocks happen when there is resource allocation (of shared resources)

system model → - Acquire (R)
- use (R)
- Release (R)

4 Necessary Conditions for deadlocks (not sufficient) → all 4 have to be met for deadlocks to happen

① → mutual exclusion
↳ resources are used in exclusive fashion

2) no preemption

→ when a resource is in use by A then it cannot be preempted (or taken away) for use by B → till it is released

3- hold & wait

↳ a process must have the ability to hold resource R_1 and then wait for R_2 (to be available) without releasing R_1

4 – Circular Wait

→ hold & wait can become circular

circular

→ P(x) T_1

P(y)

⋮

P(x) T_2

P(y)

⋮

P(y)

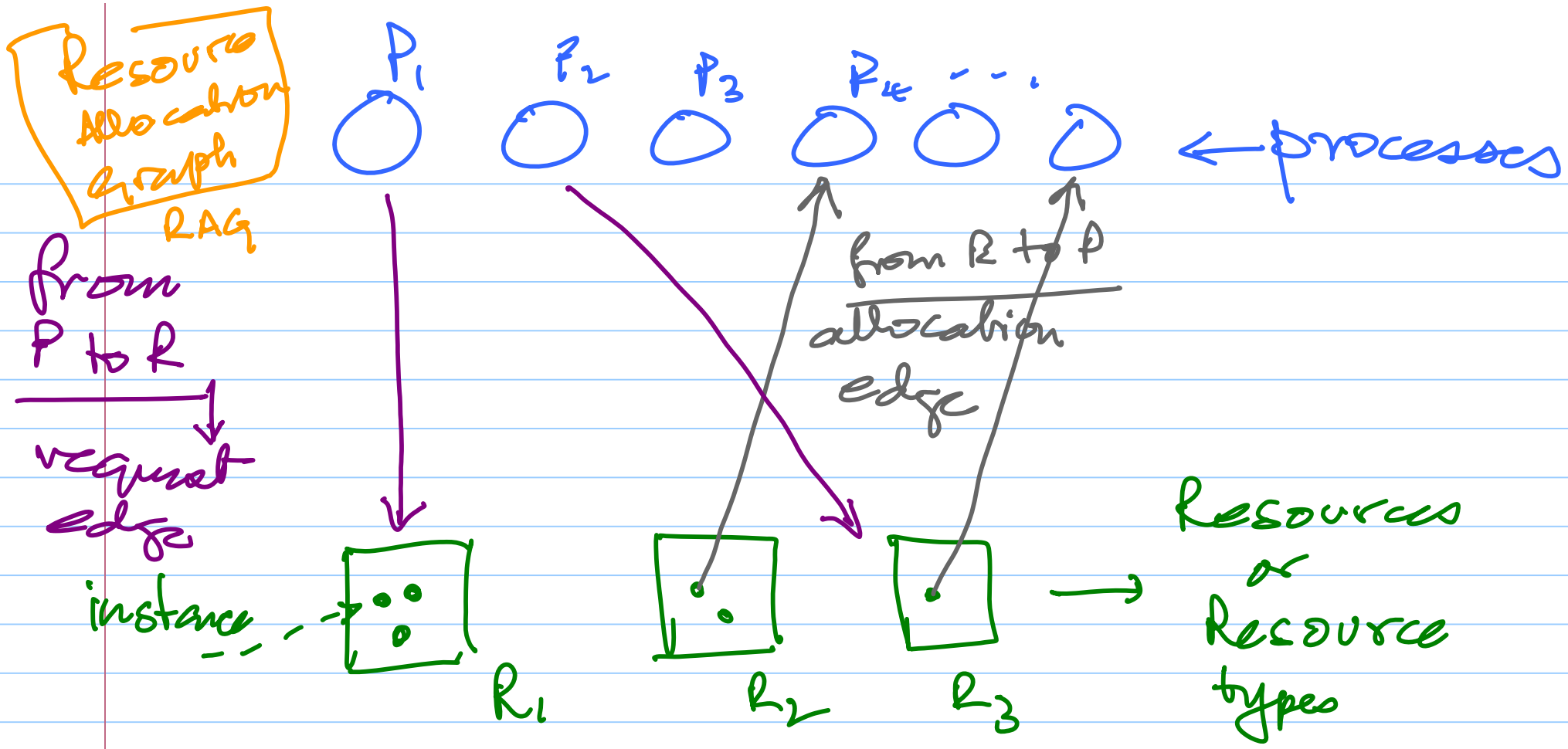
P(x)

Resources

→ case 1 → one instance of each resource

case 2 → multiple instances of each resource

↳ each instance is identical.

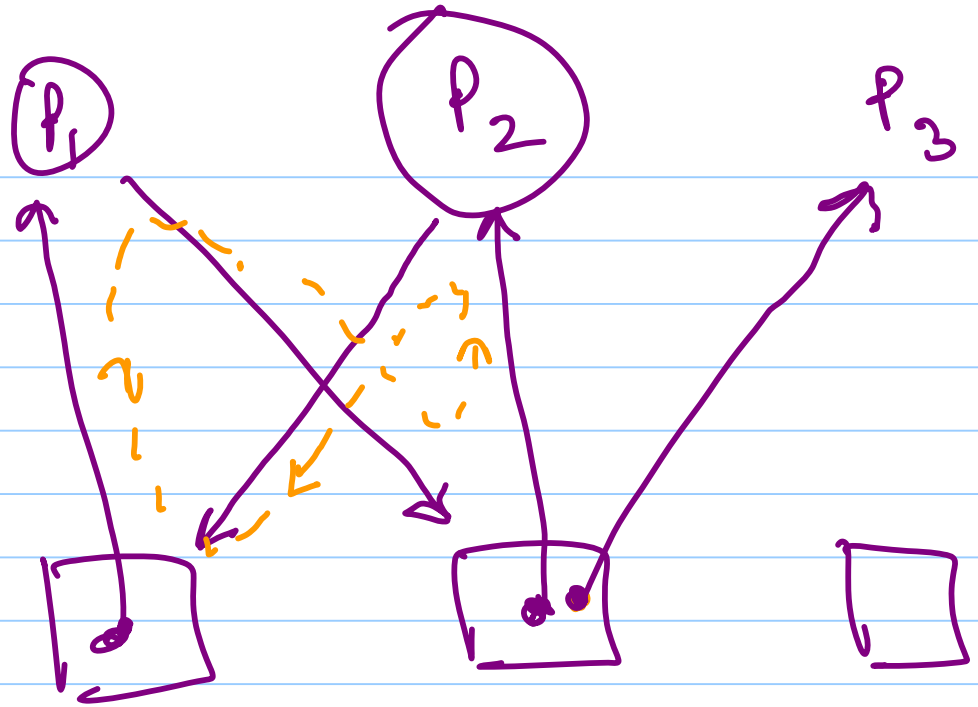

Resource Allocation Graph
RAG
P1
P2
P3
P4
processes
from P to R
request edge
from R to P
allocation edge
instance
R1
R2
R3
Resources or Resource types

Cycles in a RAG

- necessary but not sufficient cond for deadlocks

if cycle → yes deadlock
→ no deadlock] OR

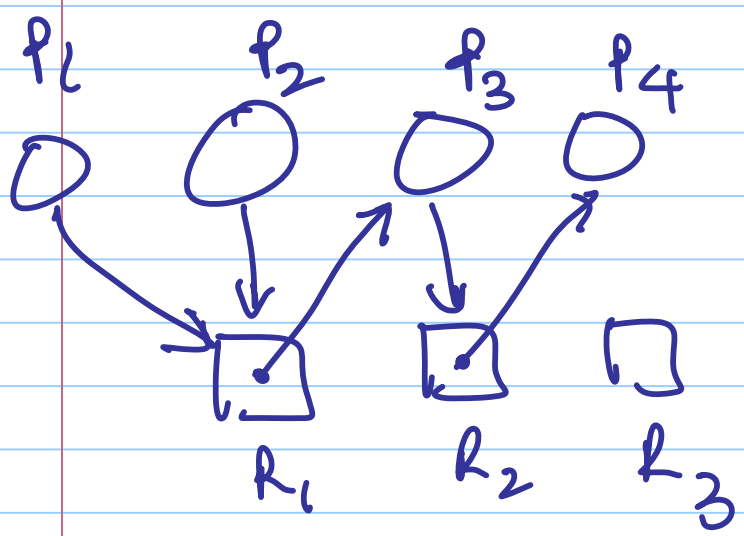
no cycle → no deadlock


P_1
P_2
P_3

A Deadlock is a stable property → once established stays for ever

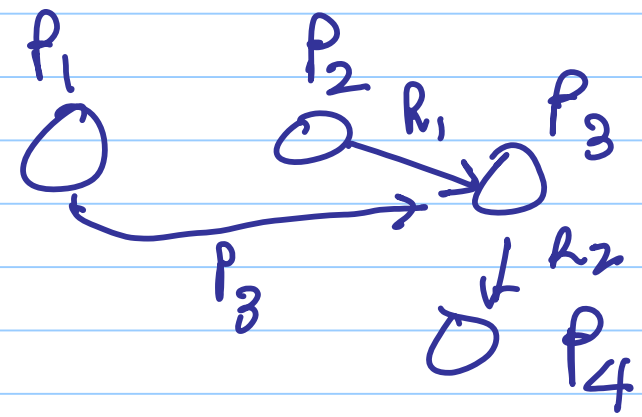
A Starvation (livelock) → unstable property
→ may continue forever OR may not

Special case of RAG → every resource has one ~~int~~ instance



⇒

<u>WFG</u> waits for graph

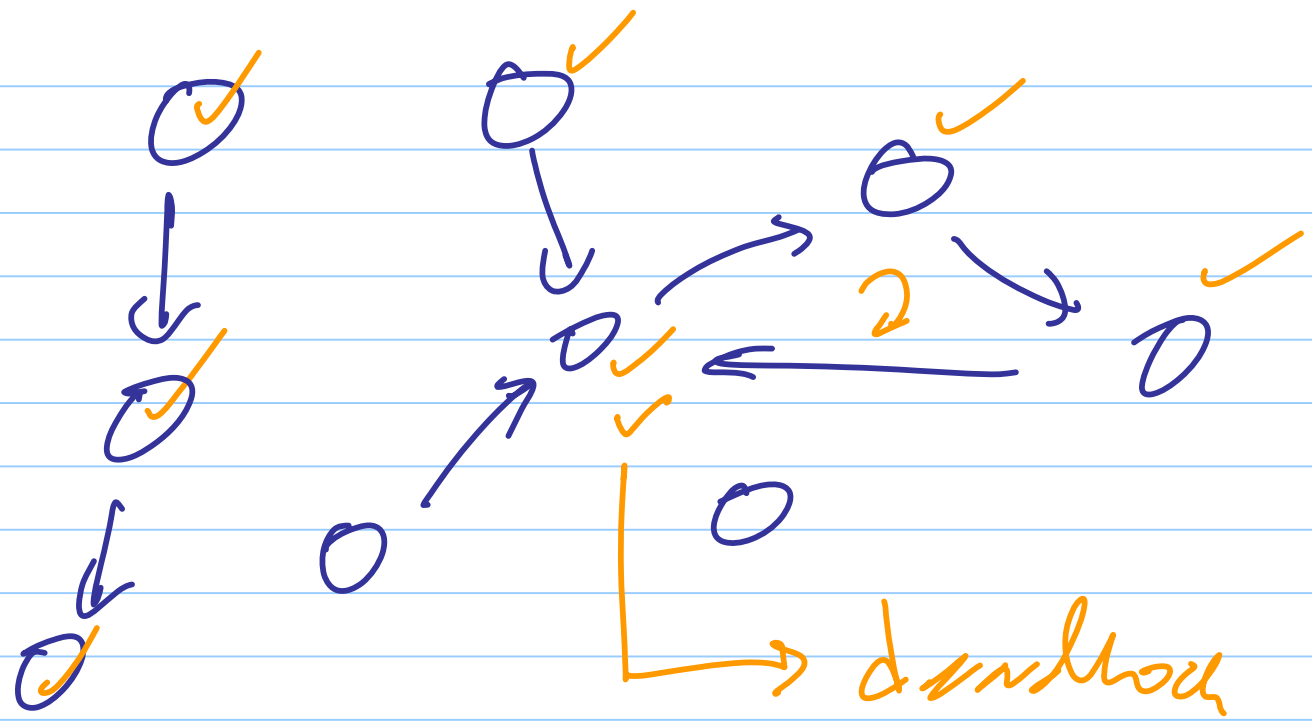


Cycle in a WFG

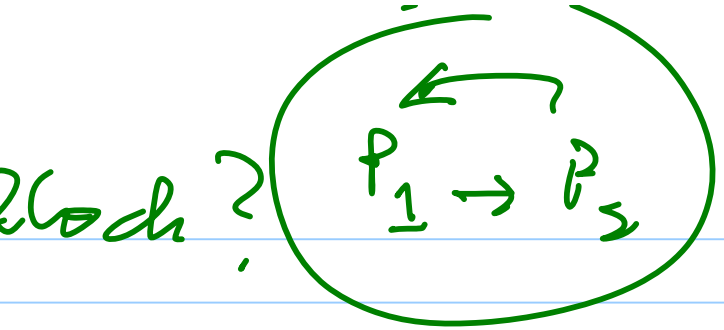
→ Necessary & sufficient cond for deadlock

cycle → deadlock

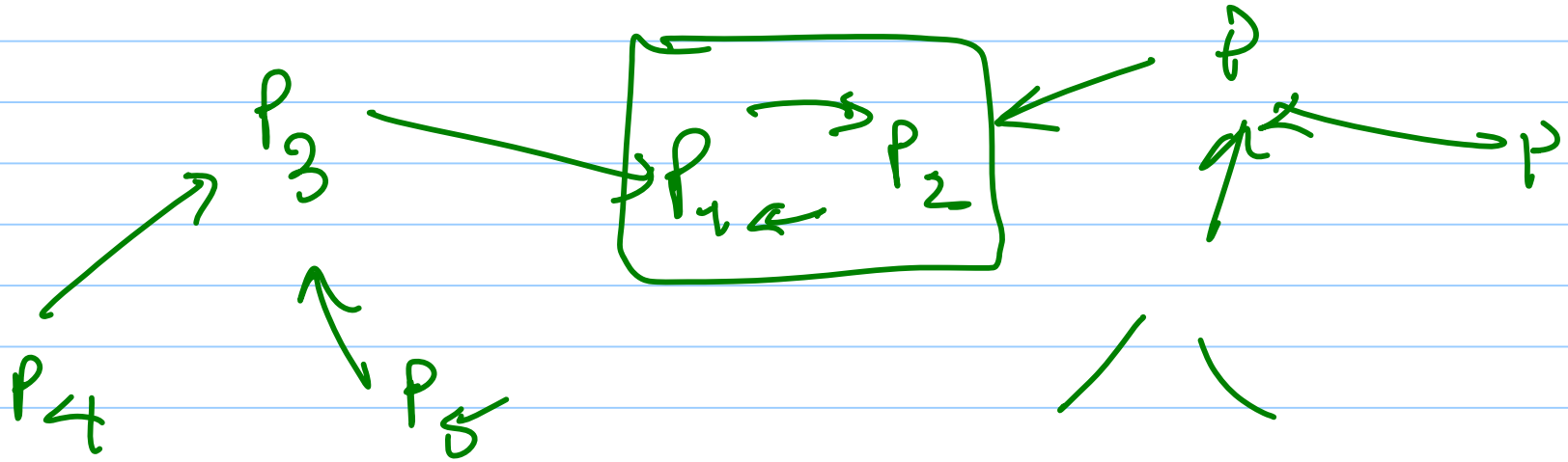
no " → no "


deadlock

- How bad is a deadlock?



100 process running



What to do about deadlocks?

- prevention
- avoidance
- detection (and recovery)

Deadlock prevention

→ make sure deadlocks cannot happen (BEST)

→ deadlock free

→ At least one of the necessary conditions are not met

Cond1 → mutual exclusion

solution → no mutual exclusion

→ all data is read only

↳ hence shared

Cond 2

→ no preemption

⇒ resources can be taken away

↳ cpu, memory → preemptible

↳ printers → spooling

Cond 3

hold & wait

↳ ① 1 resource @ a time

② All resources have to be requested in 1 shot

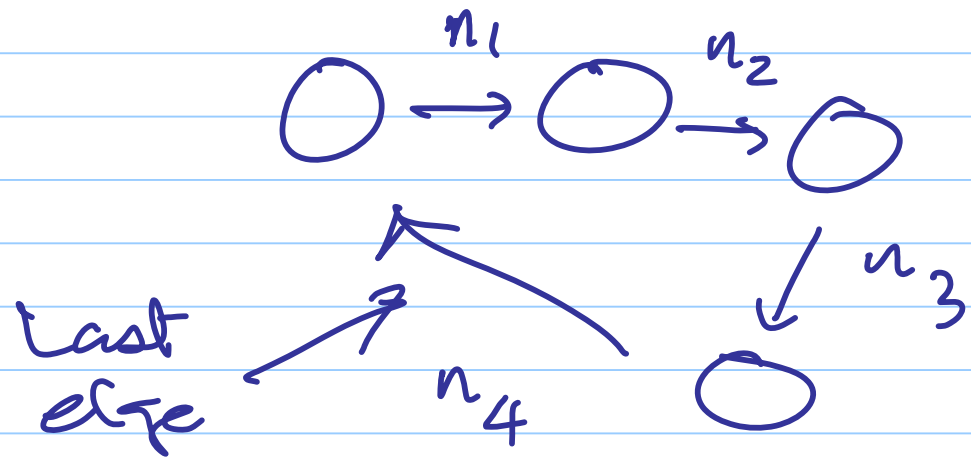
all or nothing

Cond 4 → ordered resource allocation

→ all resources have numbers

if you have R_i you can hold R_i & wait for R_j if $j > i$

WFQ



impossible

$$n_1 < n_2 < n_3 < n_4 < n_1$$