unacademy

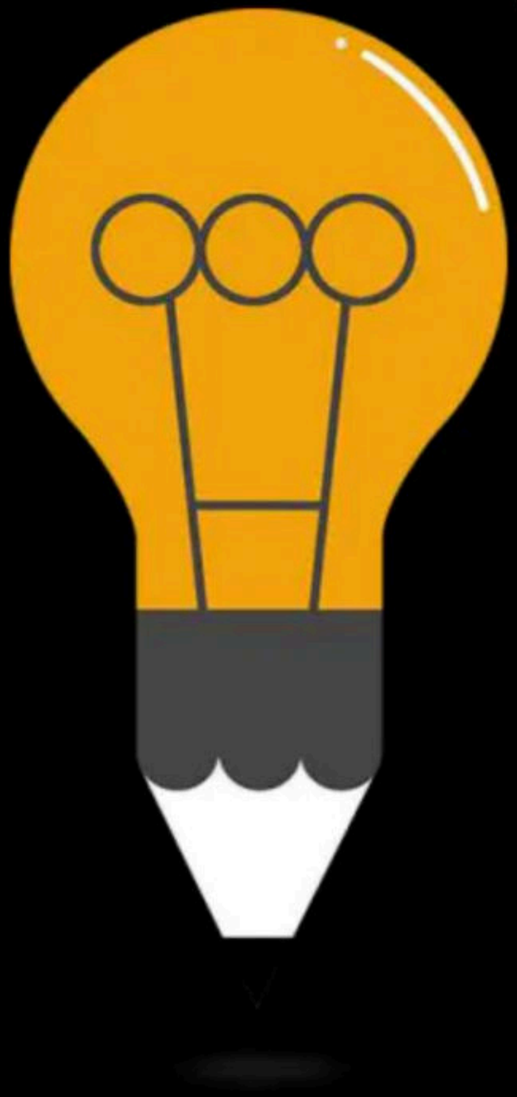
Deadlock Detection & Recovery

Comprehensive Course on Operating System for GATE - 2024/25

Vishvadeep Gothi • Lesson 21 • May 12, 2023

unacademy

easy 5 questions ← Quiz ⇒ semaphore, fork(), system-call, Threads, Deadlock (till yesterday)



Operating System
Deadlock Avoidance & Detection

By: Vishvadeep Gothi

Deadlock

If two or more processes are waiting for such an event which is never going to occur

Necessary Conditions for Deadlock

Deadlock can occur only when all following conditions are satisfied:

1. Mutual Exclusion
2. Hold & Wait
3. No-preemption
4. Circular Wait

Recovery From Deadlock

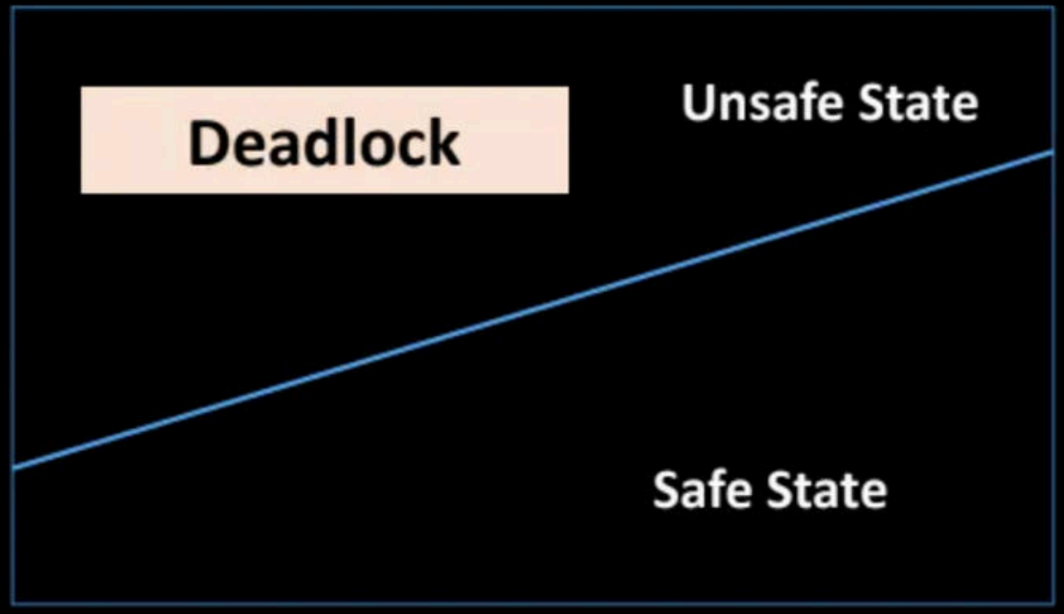
1. Make Sure that deadlock never occur
   - Prevent the system from deadlock or avoid deadlock
2. Allow deadlock, detect and recover
3. Pretend that there is no any deadlock

Deadlock Avoidance

In deadlock avoidance, the OS tries to keep system in safe state

Deadlock Avoidance

In deadlock avoidance, the OS tries to keep system in safe state



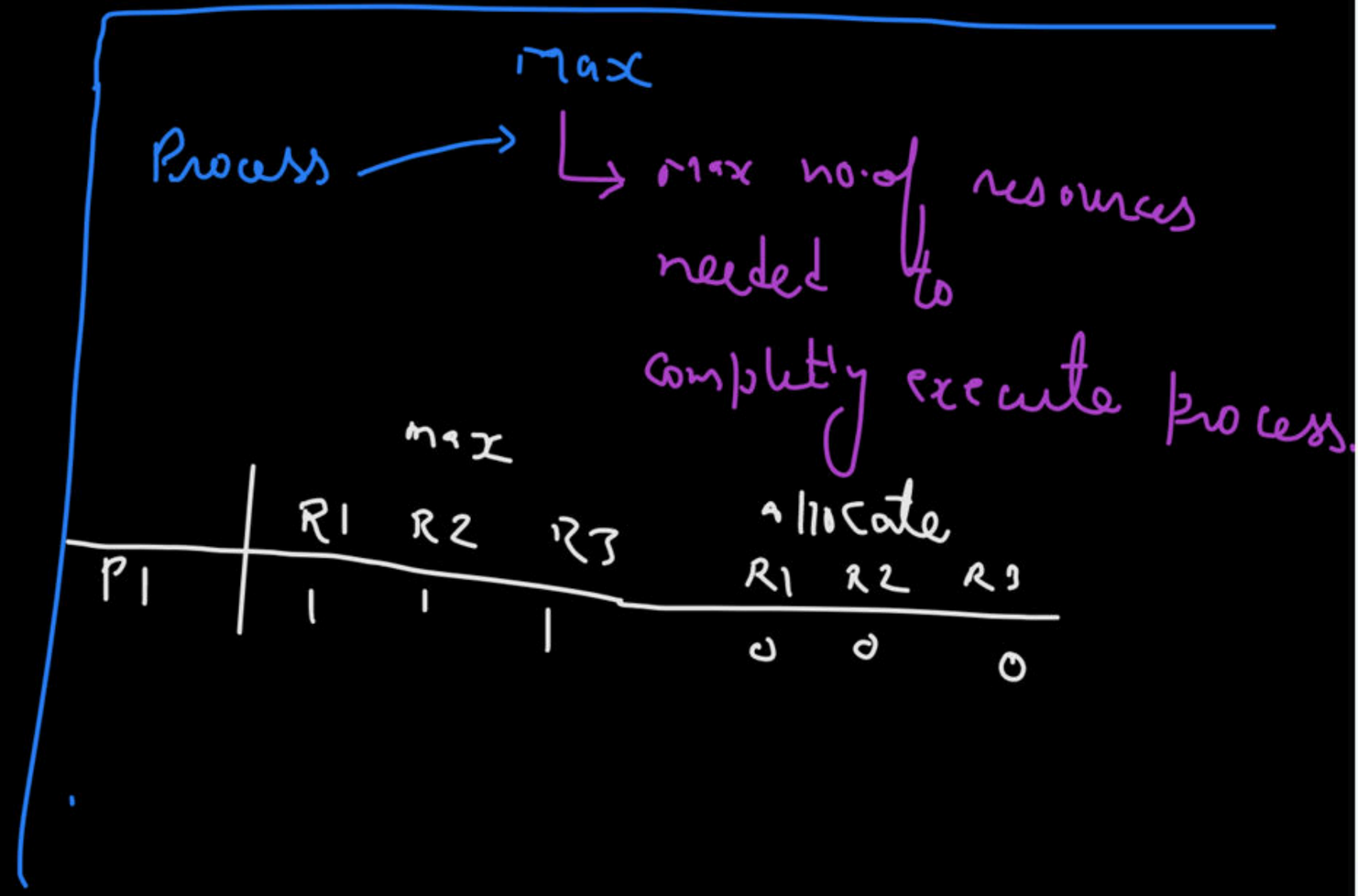
Deadlock Avoidance

In deadlock avoidance, the request for any resource will be granted if the resulting state of the system doesn't cause deadlock in the system.

Banker's Algorithm

The banker's algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety

1. Safety algo
2. Resource Request algo



Banker's Algorithm

Process	Allocation	Max	Available	need = max − Allocation
P1	1	3	~~1~~	2
P2	5	8	P3 ⇒ ~~4~~	3
P3	3	4	P1 ⇒ 5	1
P4	2	7	P2 ⇒ 10	5
			P4 ⇒ 12	

Try to find a process P_i

$need_i \leq available$

For P_3, $need_3 \leq available$

After P_3,

available = available + allocated

all process can finish,

hence system is in "safe state"

Safe sequence ⇒ <P3, P1, P2, P4>

<P3, P1, P4, P2>

<P3, P2, P1, P4>

<P3, P2, P4, P1>

is the sequence

safe sequence

<P1, P2, P3, P4> ?

Banker's Algorithm

Process	Allocation	Max	Available
	A B C	A B C	A B C
P_0	0 1 0	7 5 3	~~3 3 2~~
P_1	2 0 0	3 2 2	5 3 2
P_2	3 0 2	9 0 2	7 4 3
P_3	2 1 1	2 2 2	7 5 3
P_4	0 0 2	4 3 3	10 5 5
			10 5 7

Need

A	B	C
7	4	3
1	2	2
6	0	0
0	1	1
4	3	1

Safe Sequence => $\langle P_1, P_3, P_0, P_2, P_4 \rangle$

At some point of time

$need_i > available$ for all i

$\Downarrow$

system is in unsafe state.

Banker's Algorithm

1. Allocation:
2. Max:
3. Need:
4. Available:

Banker's Algorithm

1. Let Work and Finish be vectors of length 'm' and 'n' respectively.
   Initialize: Work = Available
   Finish[i] = false; for i=1, 2, 3, 4….n
2. Find an i such that both
   (a) Finish[i] = false
   (b) $Need_i$ <= Work
   if no such i exists goto step (4)
3. Work = Work + Allocation[i]
   Finish[i] = true
   goto step (2)
4. if Finish [i] = true for all i
   then the system is in a safe state

Question

Total resource

A	B	C	D
3	14	12	12

available

1	5	2	0

Process	Allocation A B C D	Max A B C D
P1	0 0 1 2	0 0 1 2
P2	1 0 0 0	1 7 5 0
P3	1 3 5 4	2 3 5 6
P4	0 6 3 2	0 6 5 4
P5	0 0 1 4	0 6 5 6

	Available A B C D	Need A B C D
	~~1 5 2 0~~	0 0 0 0
P1	~~1 5 3 2~~	0 7 5 0
P3	~~2 8 8 6~~	1 0 0 2
P2	~~3 8 8 6~~	0 0 2 2
P4	~~3 14 11 8~~	0 6 4 2
P5	3 14 12 12	

safe state =>

safe seq. => <P1, P3, P2, P4, P5>

Banker's Algorithm

Process	Allocation	Max	Available
	A B C	A B C	A B C
P_0	0 1 0	7 5 3	~~3 3 2~~ 2 3 0
P_1	~~3 2 0 0 2~~	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	

Need

A	B	C
7	4	3
~~1 2 2~~		0 2 0
6	0	0
0	1	1
4	3	1

$\langle P_1, P_3, P_2, P_4, P_5 \rangle$

safe sequence

Question

? granted

What will happen if process P1 requests one additional instance of resource type A and two instances of resource type C?

A B C

$Request_{P1} = \langle 1, 0, 2 \rangle$

① if $Request_i \leq need_i$ then next steps
o/w invalid request.

② if $Request_i \leq available$ then next step
not enough resource

③ $Allocation_i = Allocation_i + Request_i$

$Need_i = Need - Request_i$

$Available = Available - Request_i$

4. Run safety algo. ; if system is safe then request is granted

o/w request rejected.

1. $\langle 1, 0, 2 \rangle \leq \langle 1, 2, 2 \rangle$ valid request

2. $\langle 1, 0, 2 \rangle \leq \langle 3, 3, 2 \rangle$ resources available

safe state

Request granted

Banker's Algorithm

$Req_0 = \langle 1, 0, 2 \rangle$

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

need

A	B	C
~~7~~	~~4~~	~~3~~
6	4	1
1	2	2
6	0	0
0	1	1
4	3	1

no any process having $need_i \leq available$

Hence, not safe $\Rightarrow$ Req. of P_0 rejected

Question

What will happen if process P0 requests one additional instance of resource type A and two instances of resource type C?

$P_0 \quad \langle 1, 0, 2 \rangle$

Question

Homework

What will happen if process P3 requests one additional instance of resource type B?

original table

Resource Request Algorithm

1. *If $Request_i$ <= $Need_i$*
   Goto step (2) ; otherwise, raise an error condition, since the process has exceeded its maximum claim.
2. *If $Request_i$ <= Available*
   Goto step (3); otherwise, P_i must wait, since the resources are not available.
3. *Have the system pretend to have allocated the requested resources to process P_i by modifying the state as follows:*
   Available = Available – Requesti
   $Allocation_i$ = $Allocation_i$ + $Request_i$
   $Need_i$ = $Need_i$– $Request_i$

Question

Homework

Req 1. What will happen if process P1 requests one additional instance of resource type A and two instances of resource type C? Req. P1 <1, 0, 2>

Req. 2. What will happen if process P3 requests one additional instance of resource type B?

Options :-

1. only req1 is granted individually
2. only req2 is granted individually
3. Both rejected
4. Both granted one after another
5. Both individually granted but not together!

Deadlock Detection

1. When all resources have single instance
2. When resources have multiple instances

Deadlock Detection

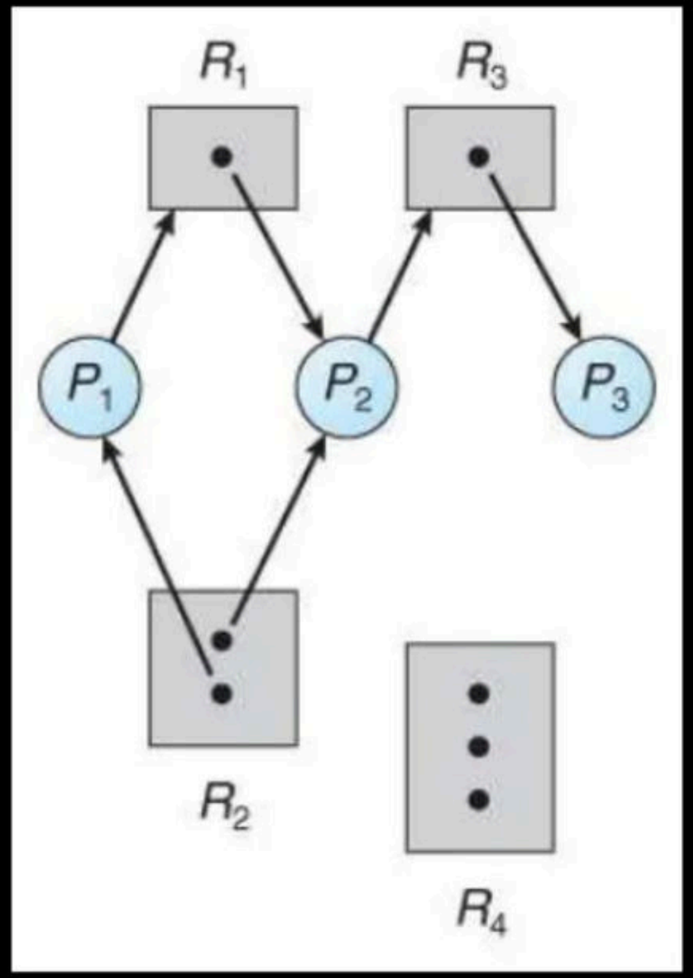
When all resources have single instance:

Deadlock detection is done using wait-for-graph

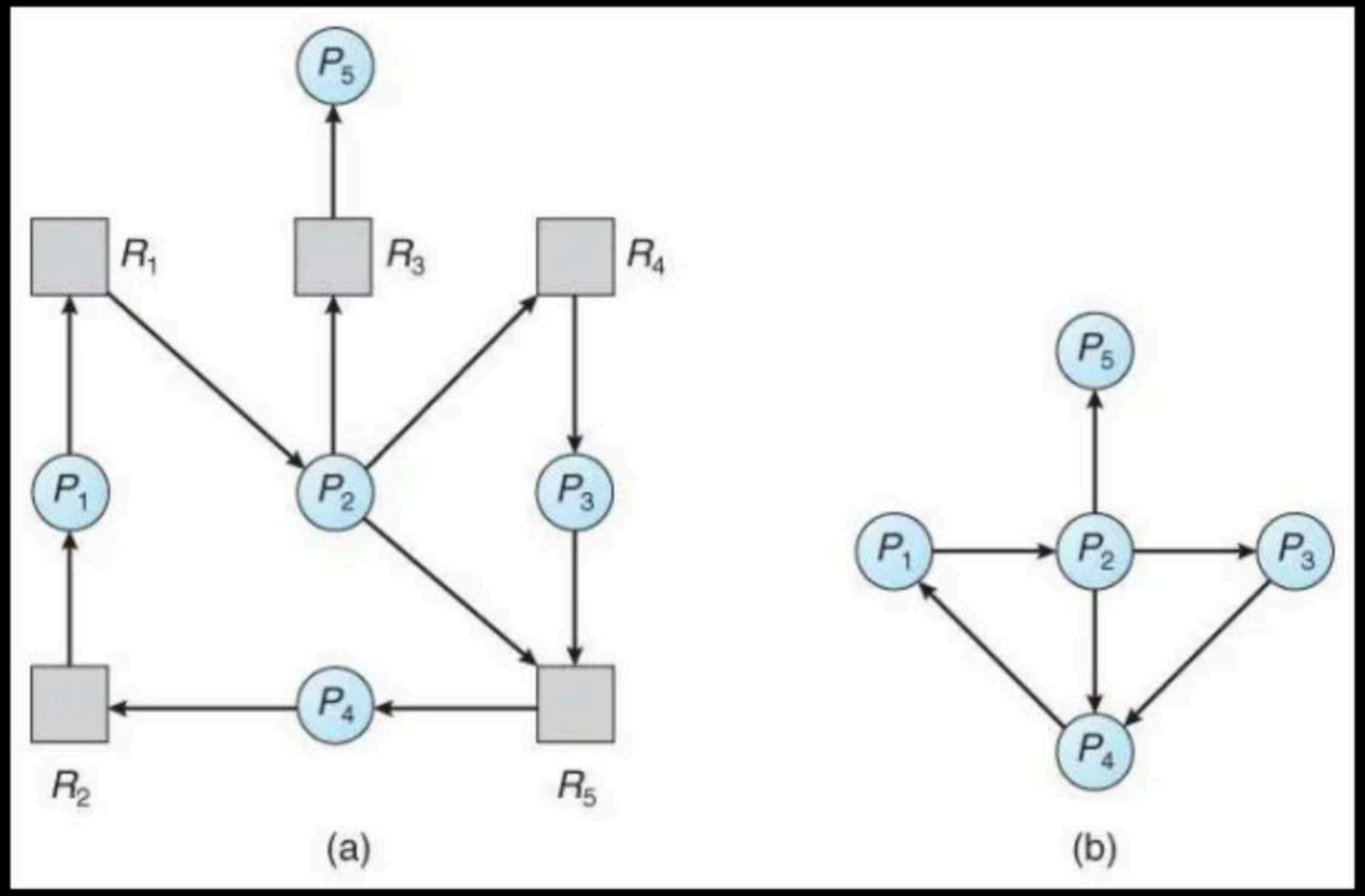
Wait For Graph

It is created from resource allocation graph

Wait For Graph



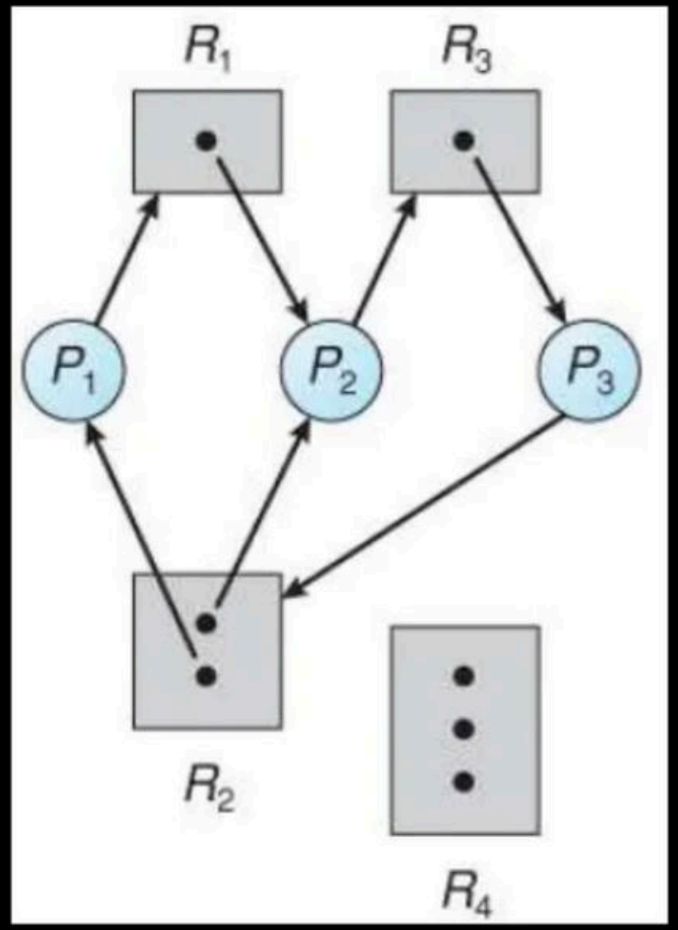
Wait For Graph



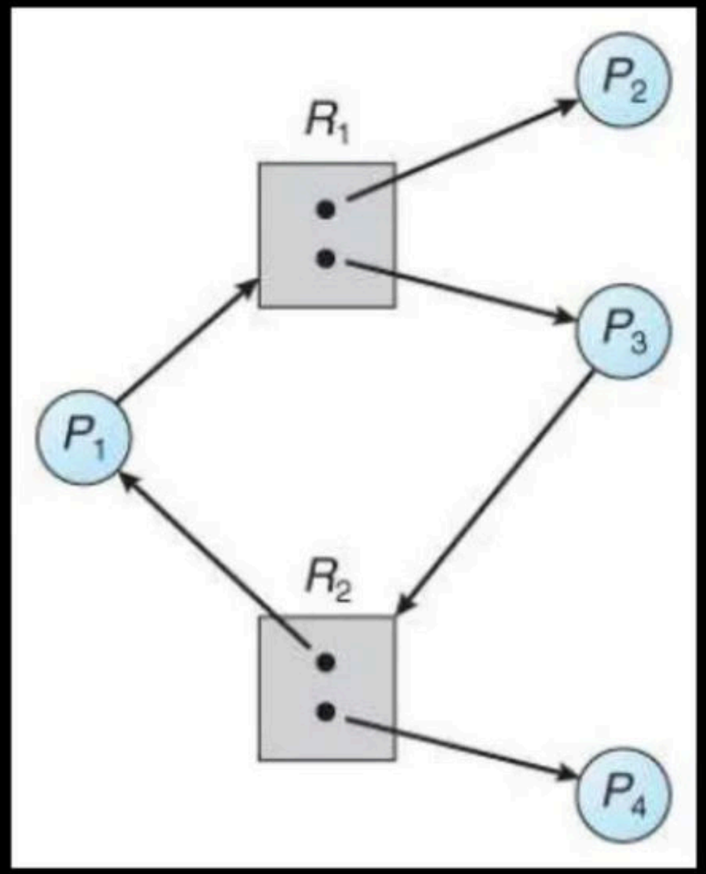
Wait For Graph

If a resource category contains more than one instance, then the presence of a cycle in the resource-allocation graph indicates the possibility of a deadlock, but does not guarantee one.

Wait For Graph: Example



Wait For Graph: Example



Deadlock Detection

When resources have multiple instance:

Deadlock detection is done using a specific algorithm

Deadlock Detection Algorithm

	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	
P_2	3 0 3	0 0 0	
P_3	2 1 1	1 0 0	
P_4	0 0 2	0 0 2	

Deadlock Detection Algorithm

1. Let *Work* and *Finish* be vectors of length m and n respectively.
   Initialize *Work= Available*. For *i=0, 1, …., n-1*, if $Request_i$ = 0, then *Finish[i]* = true; otherwise, *Finish[i]*= false.
2. Find an index i such that both
   a) *Finish[i] == false*
   b) $Request_i$ <= *Work*
   If no such *i* exists go to step 4.
3. *Work= Work+ Allocation*$_i$
   Finish[i]= true
   Go to Step 2.
4. If *Finish[i]== false* for some i, 0<=i<n,
   then the system is in a deadlocked state.
   Moreover, if *Finish[i]==false* the process P_i is deadlocked.

Detection-Algorithm Usage

When should the deadlock detection be done? Frequently, or infrequently?

Detection-Algorithm Usage

1. Do deadlock detection after every resource allocation
2. Do deadlock detection only when there is some clue

Recovery From Deadlock

There are three basic approaches to recovery from deadlock:

1. Inform the system operator and allow him/her to take manual intervention
2. Terminate one or more processes involved in the deadlock
3. Preempt resources.

Process Termination

1. Terminate all processes involved in the deadlock
2. Terminate processes one by one until the deadlock is broken

Process Termination

Many factors that can go into deciding which processes to terminate next:

1. Process priorities.
2. How long the process has been running, and how close it is to finishing.
3. How many and what type of resources is the process holding
4. How many more resources does the process need to complete
5. How many processes will need to be terminated
6. Whether the process is interactive or batch

Resource Preemption

Important issues to be addressed when preempting resources to relieve deadlock:

1. Selecting a victim
2. Rollback
3. Starvation

unacademy

Happy Learning.!

OS CPU sch.

PYQ => 2:30pm