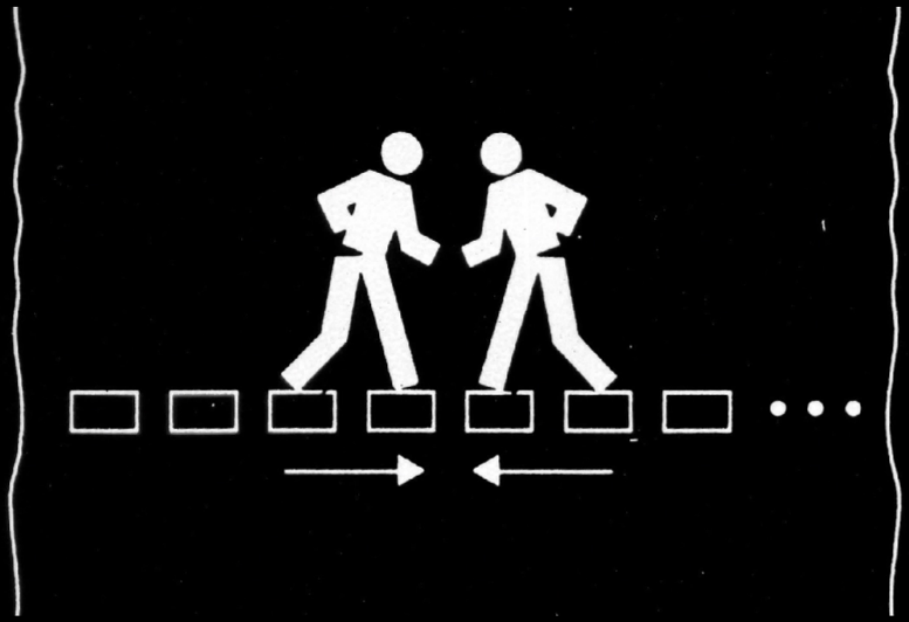
Lecture: 34

Deadlock

Deadlock



The Deadlock Problem

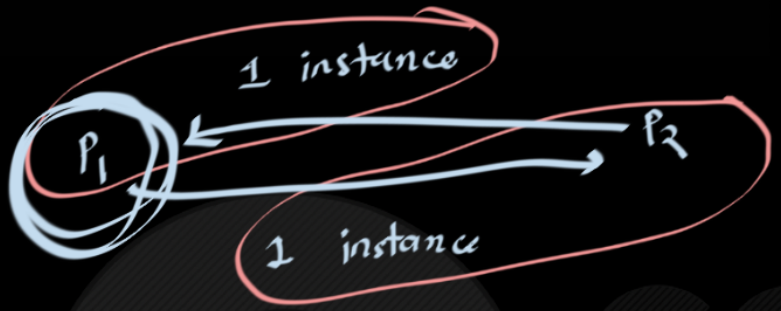
A set of blocked processes each holding a resource and waiting to acquire a resource held by another process in the set .

Example:

A system has 2 disk drives, P1 and P2 each hold one disk drive and each needs another one

Example:

A system has 2 disk drives, P1 and P2 each hold one disk drive and each needs another one



P1 saying: to complete my execution i want 1 more instance

P2 saying: to complete my execution i want 1 more instance

lack of paisa
money
Resources

this is the one fundamental reason but we can't solve it.

we will see other ways.

Deadlock example in semaphores:

Semaphores A and B, initialized to 1

```
P1                    P2
wait (A);             wait(B);
wait (B);             wait(A);
```


IMPERIAL COLLEGE OF
Explain Deadlock, and we'll hire you.
Hire me, and I'll explain it to you.
let's discuss the salary.

EXAMPLES: (memes)

- "It takes money to make money".
- You can't get a job without experience; you can't get experience without a job.

Four Conditions for Deadlock

1. Mutual exclusion condition (Resources are Not Shareable)
2. Hold and wait condition
3. No preemption condition
4. Circular wait condition

Deadlock

Four Conditions for Deadlock

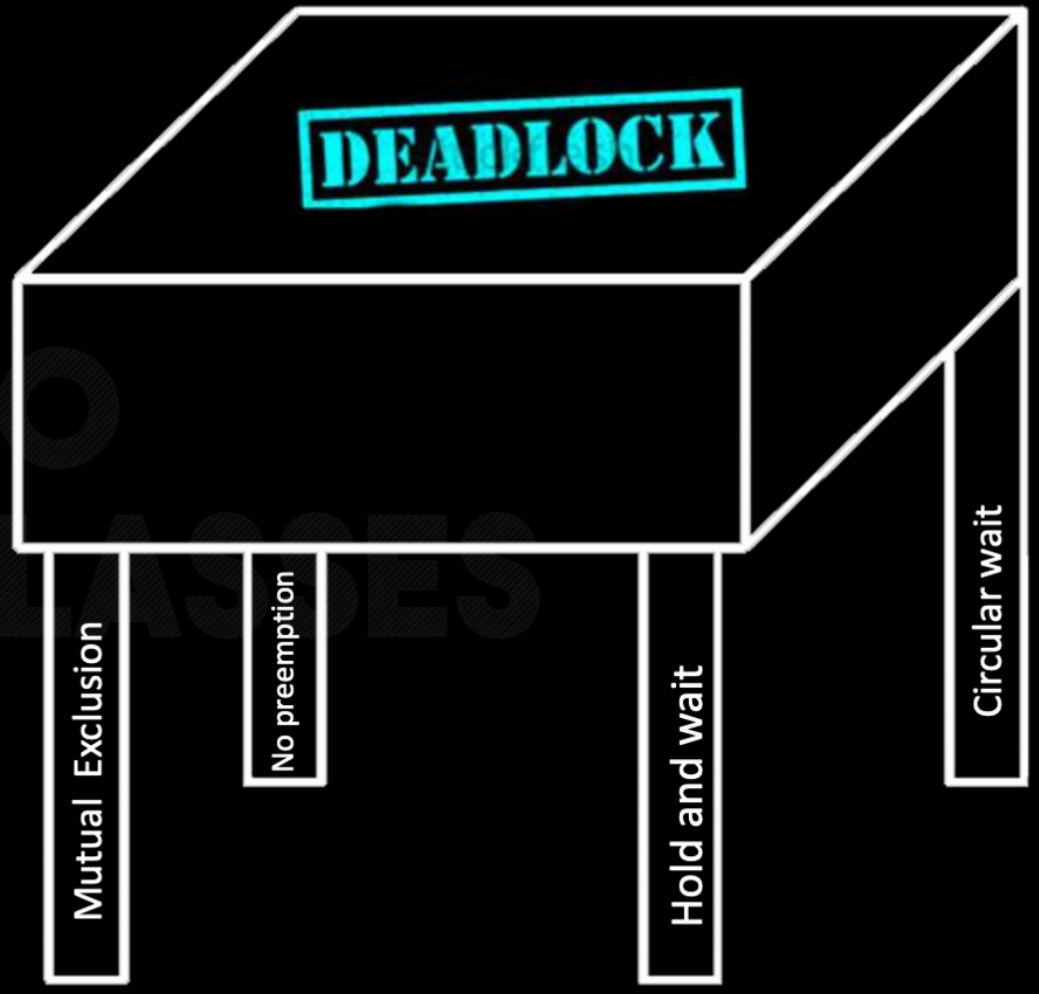
1. Mutual exclusion condition: Resources can not be shared
2. Hold and wait condition: Process holding resources and waiting for some other resources
3. No preemption condition: Previously granted resources cannot forcibly taken away. (Note the this is preemption of resources not processes)
4. Circular wait condition: Processes waiting for resources in circle.

4. **Circular wait:** there exists a set $\{P_0, P_1, \ldots, P_n\}$ of waiting processes such that
   - P_0 is waiting for a resource that is held by P_1,
   - P_1 is waiting for a resource that is held by P_2,
   - …,
   - P_{n-1} is waiting for a resource that is held by P_n, and
   - P_n is waiting for a resource that is held by P_0.

These 4 conditions are like four legs of the table

If any leg break, you will not have deadlock.

These conditions are **Necessary conditions** which means having all of them doesn't mean deadlock i.e. these are not sufficient condition.



GO CLASSES

Deadlock $\Rightarrow$ All four cond^n Must be there

4 cond^{n}s are satisfied $\Rightarrow$ Deadlock may or may not

Having these four conditions ⇒ Deadlock may or may not occur

NOT having any of the conditions ⇒ Deadlock will **never** occur

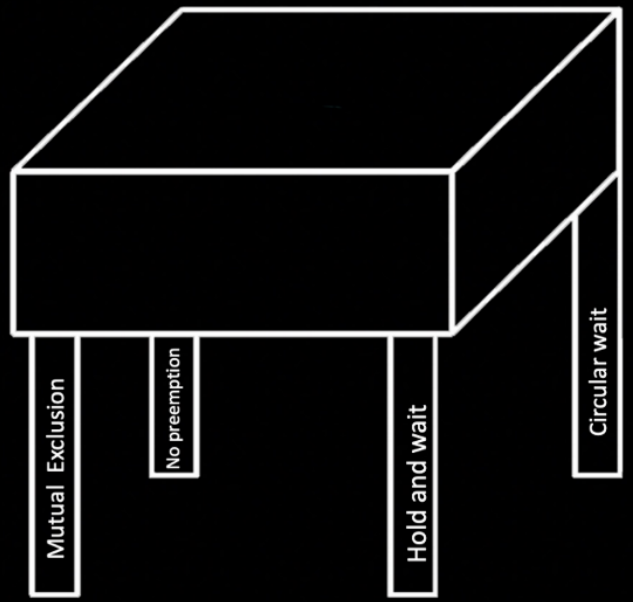


Figure: Four conditions satisfied but no deadlock

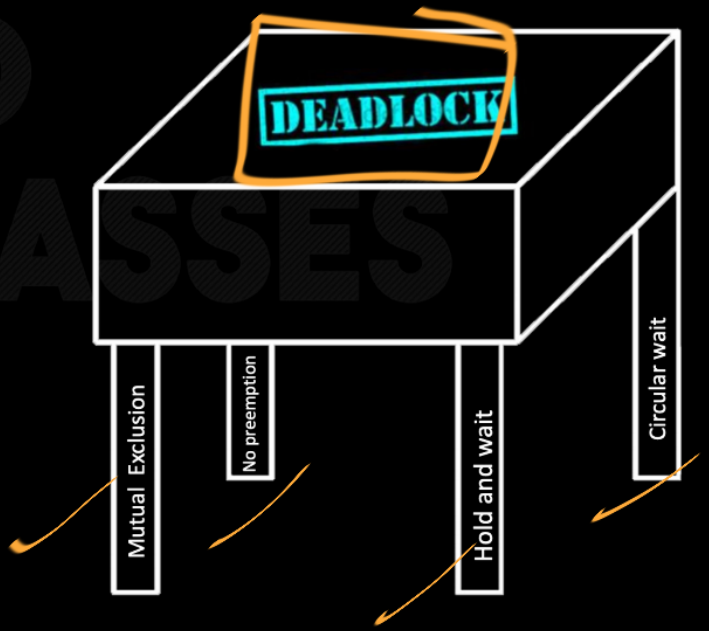


Figure: Four conditions satisfied and deadlock

True/False

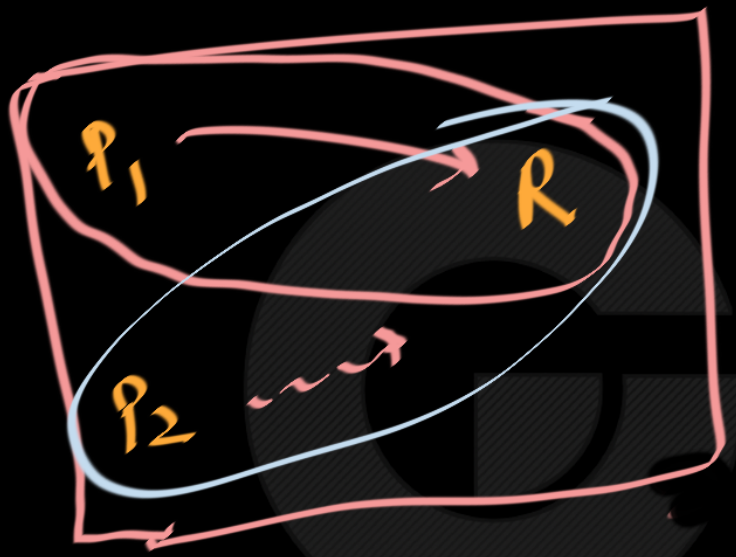
Question from Galvin

Deadlock cannot occur among processes that need at most one (non-shareable) resource each.

↓

every process just need 1 resource

P_1 R_1 P_2 ⇒ Deadlock can not occur


P_1
R
P_2

True/False

Question from Galvin

Deadlock cannot occur among processes that need at most one (non-shareable) resource each.

↓

every process just need 1 resource

⇒ Deadlock can not occur

Hold and wait will never be there ⇒ No deadlock

True/False

Question from Galvin

Deadlock cannot occur among processes that need at most one (non-shareable) resource each.

Answer: True

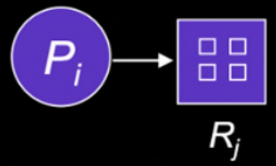
We will never have Hold and wait condition here.

Resource-Allocation Graph

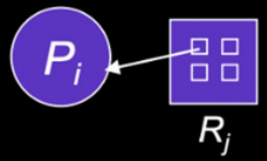
- Process

- Resource Type with 4 instances

- P_i requests instance of R_j



- P_i is holding an instance of R_j



GO CLASSES

Question:

Consider the following information about resources in a system:

- There are two classes of allocatable resource labelled R1 and R2.
- There are two instances of each resource.
- There are four processes labelled P1 through P4.
- There are some resource instances already allocated to processes, as follows:
  - one instance of R1 held by P2, another held by P3
  - one instance of R2 held by P1, another held by P4
- Some processes have requested additional resources, as follows:
  - P1 wants one instance of R1
  - P3 wants one instance of R2

a. (5 marks)

Draw the resource allocation graph for this system. Use the style of diagram from the lecture notes.

GO CLASSES

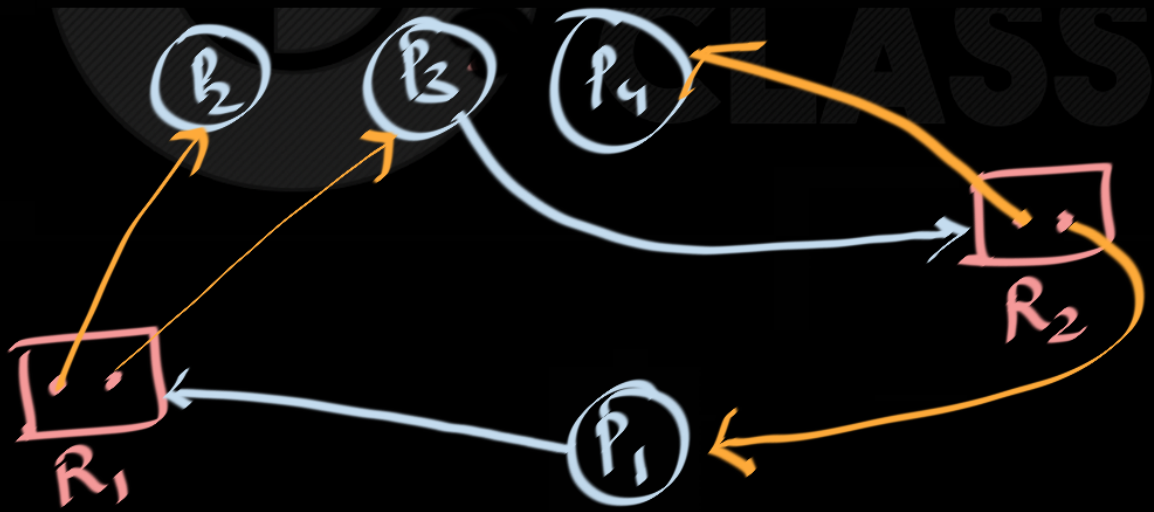
Question:

Consider the following information about resources in a system:

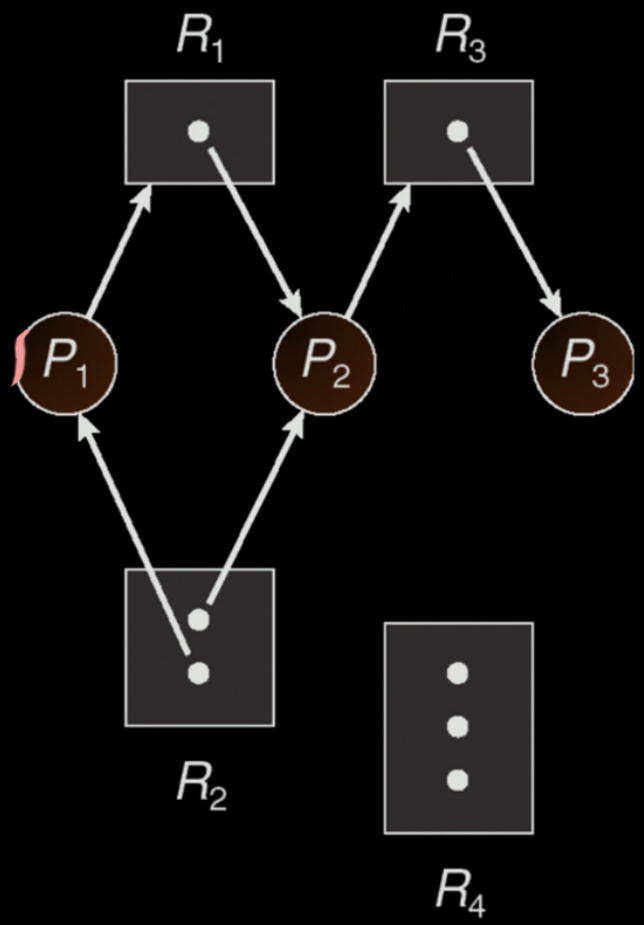
- There are two classes of allocatable resource labelled R1 and R2.
- There are two instances of each resource.
- There are four processes labelled P1 through P4.
- There are some resource instances already allocated to processes, as follows:
  - one instance of R1 held by P2, another held by P3
  - one instance of R2 held by P1, another held by P4
- Some processes have requested additional resources, as follows:
  - P1 wants one instance of R1
  - P3 wants one instance of R2

a. (5 marks)

Draw the resource allocation graph for this system. Use the style of diagram from the lecture notes.

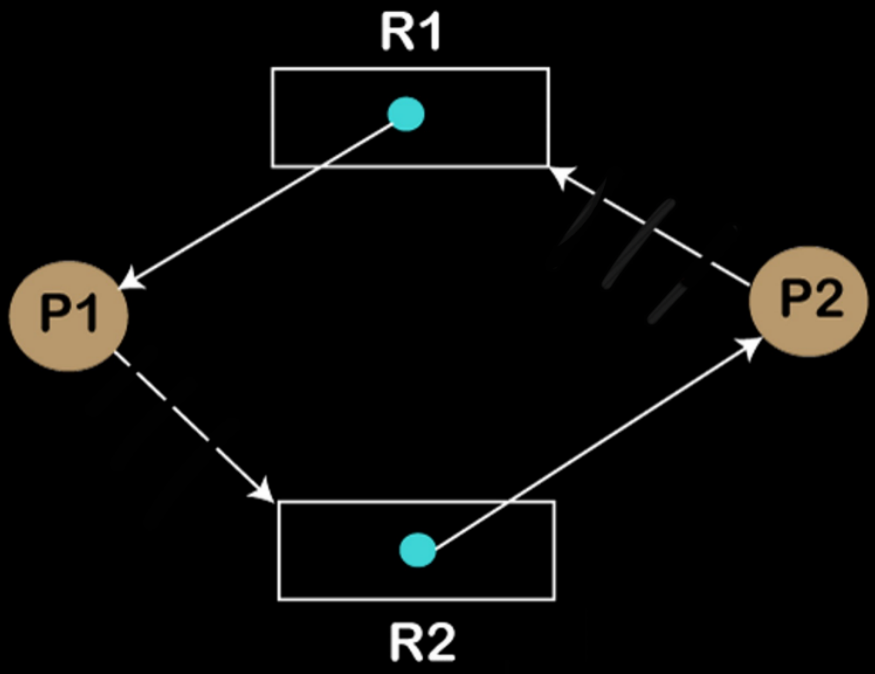


Example of a Resource Allocation Graph



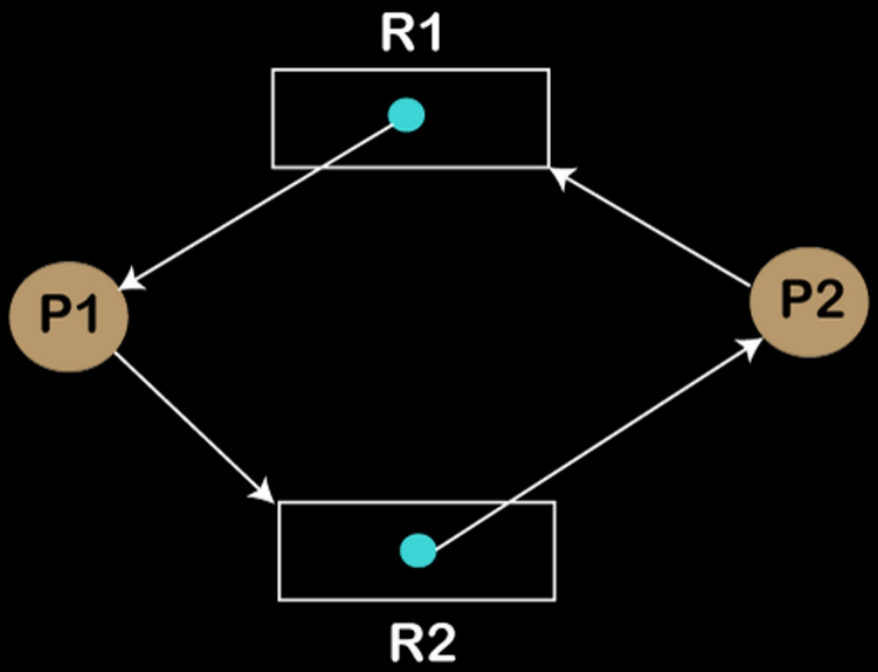
Question:

Does the corresponding system has deadlock ?



"Deadlock detection algorithm"

formal way of checking

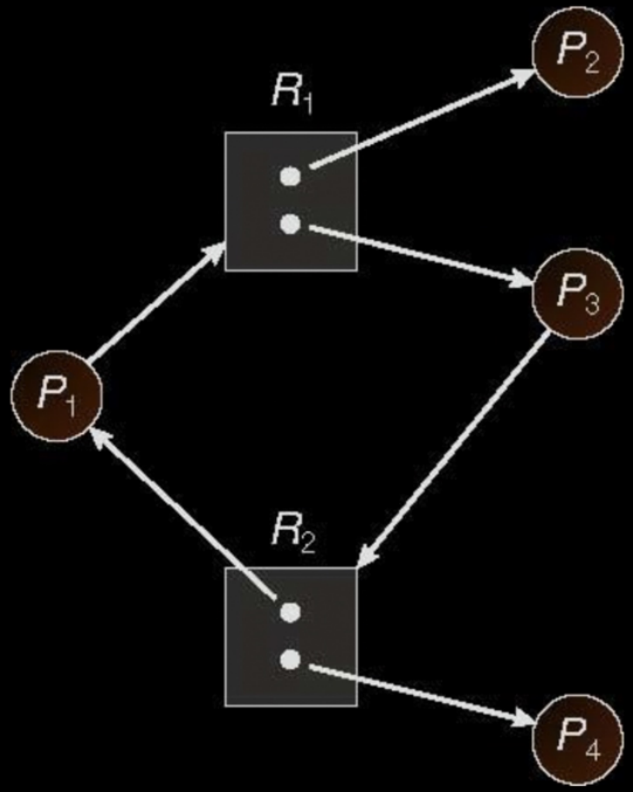


Answer:

Yes, since it has a cycle and each resource is having single instance.

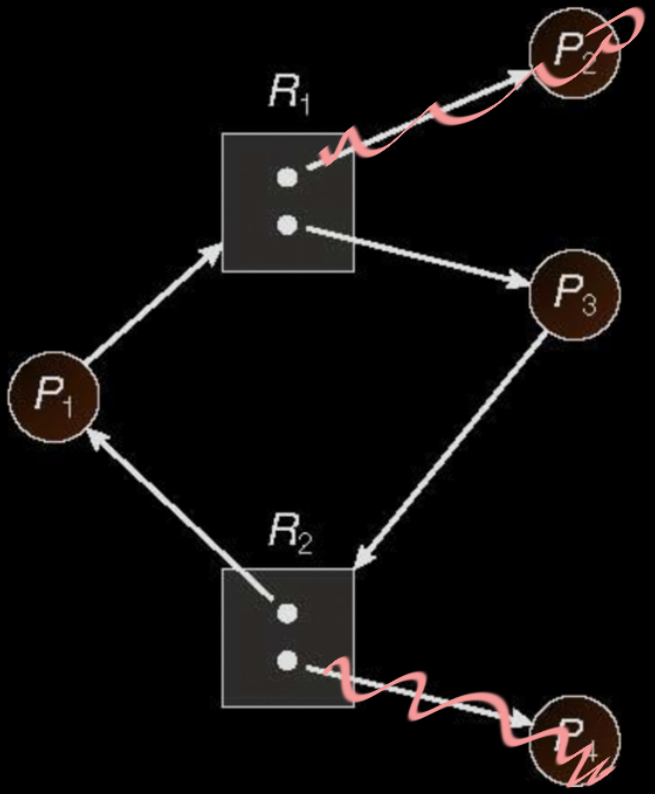
Question:

Does the corresponding system has deadlock ?



Question:

Does the corresponding system has deadlock ?

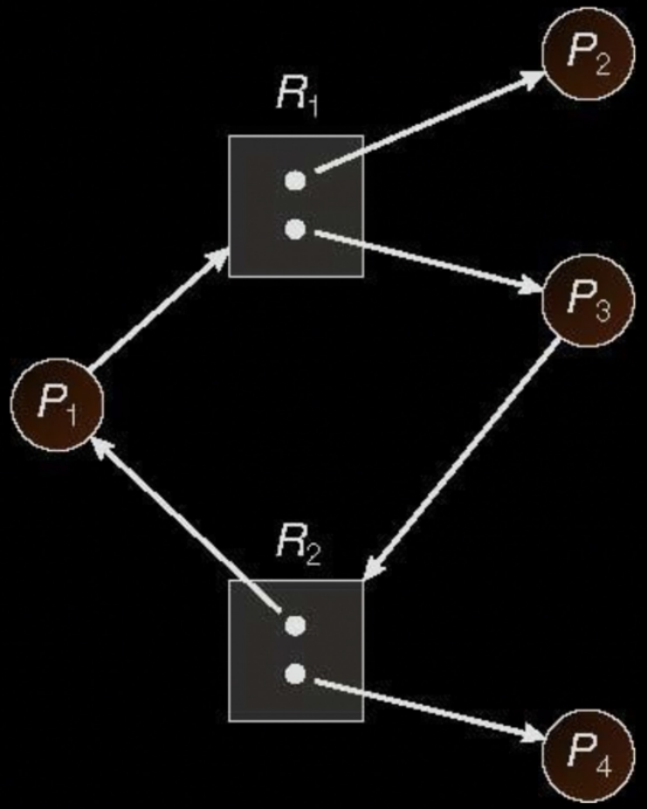


P_4 P_2 P_1 P_3

P_2 P_4 P_1 P_3

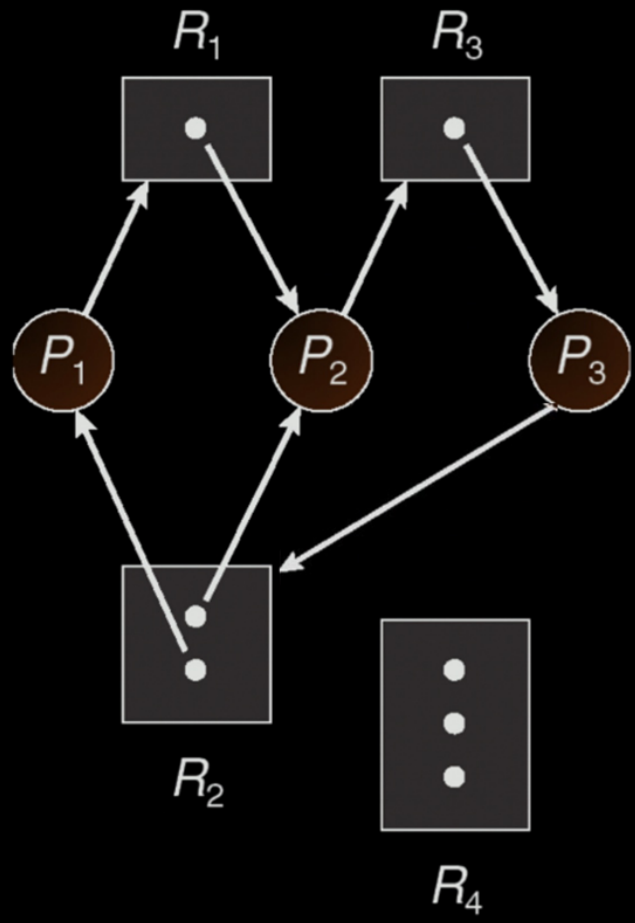
P_2 P_4 P_3 P_1

Graph With A Cycle But No Deadlock



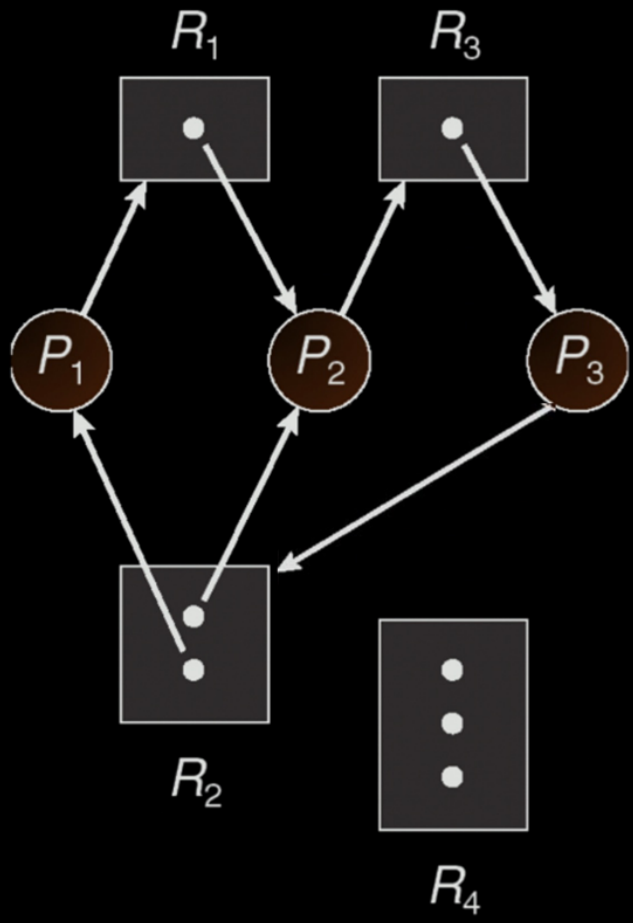
Question:

Does the corresponding system has deadlock ?



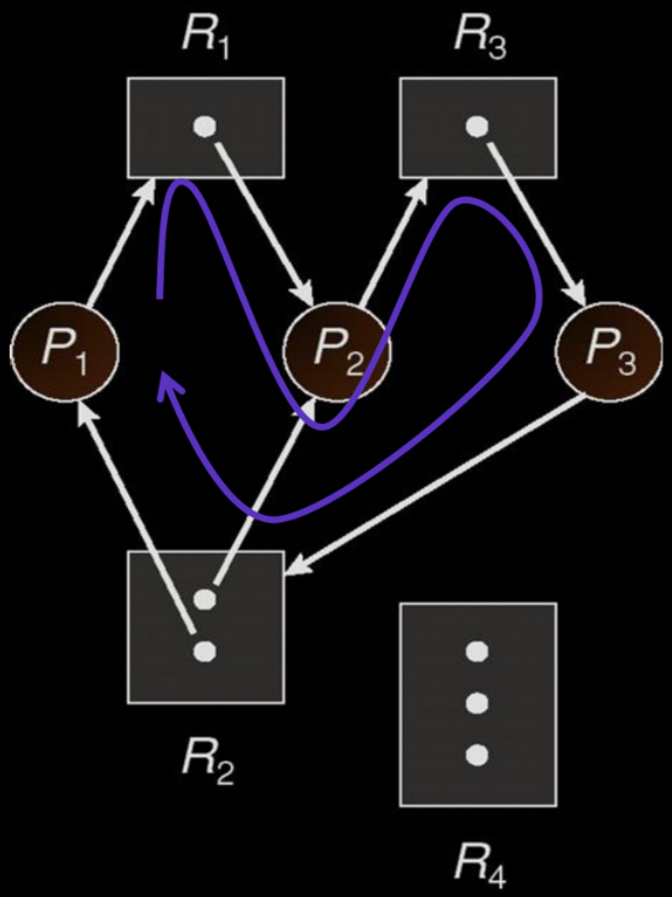
Question:

Does the corresponding system has deadlock ?

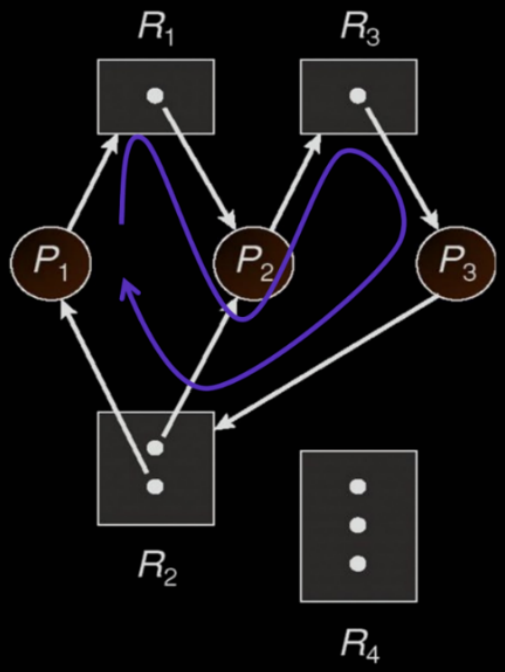


YES

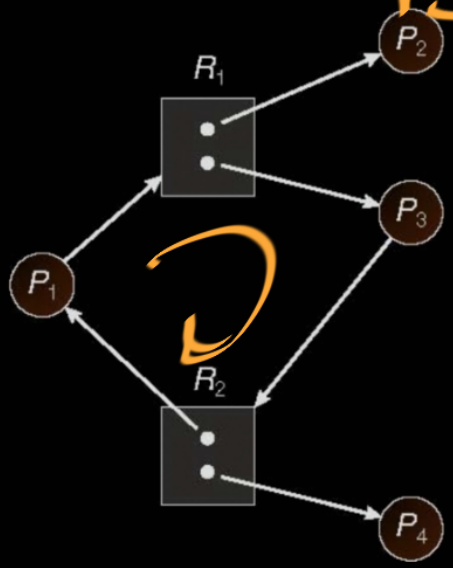
Resource Allocation Graph With A Deadlock

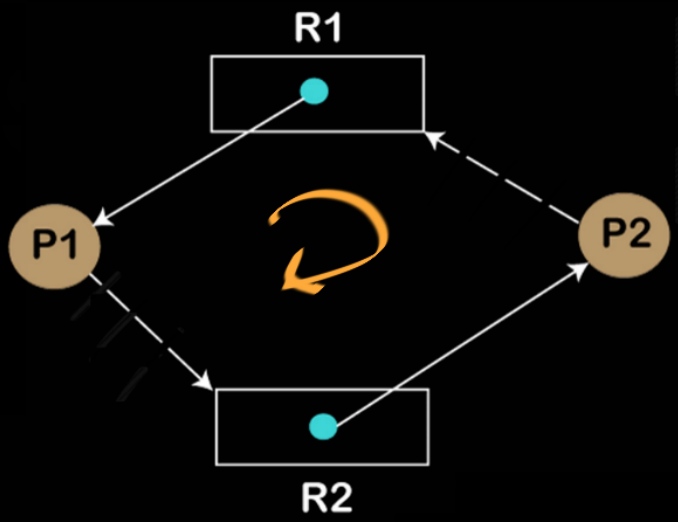


Resource Allocation Graph With A Deadlock



Graph With A Cycle But No Deadlock





Deadlock ✓

Basic Facts

- If graph contains no cycles ⇒ no deadlock (No circular wait)
- If graph contains a cycle ⇒
  - if only one instance per resource type, then deadlock

Basic Facts

- If graph contains no cycles ⇒ no deadlock (No circular wait)
- If graph contains a cycle ⇒
  - if only one instance per resource type, then deadlock
  - if several instances per resource type, possibility of deadlock

Basic Facts

- If graph contains no cycles ⇒ no deadlock
- If graph contains a cycle ⇒
  - if only one instance per resource type, then deadlock
  - if several instances per resource type, possibility of deadlock

If there is one instance per resource type then we can check deadlock using resource allocation graph but if there are multiple instance per resource type then we need deadlock detection method (later in the lectures)

Question:

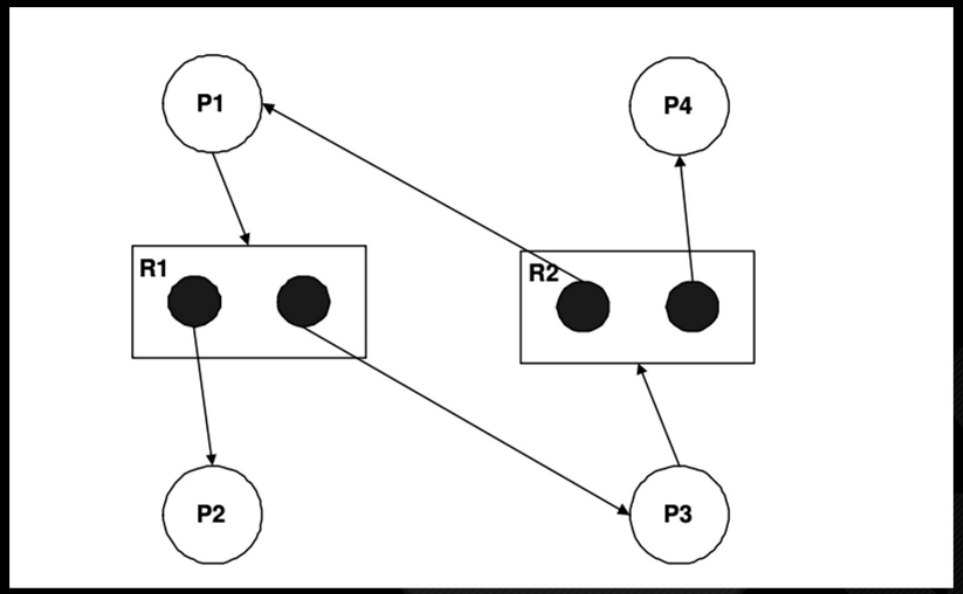
Consider the following information about resources in a system:

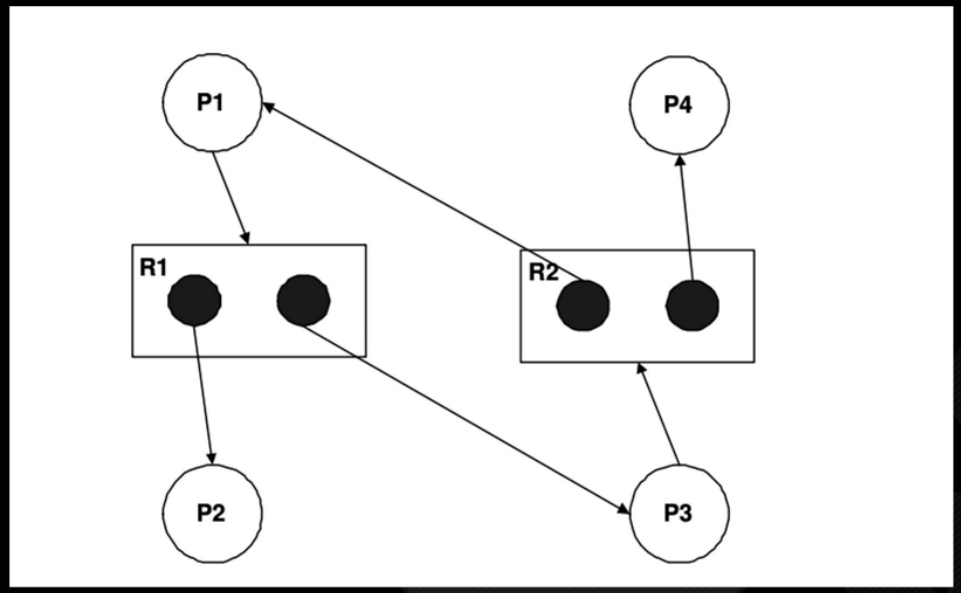
- There are two classes of allocatable resource labelled R1 and R2.
- There are two instances of each resource.
- There are four processes labelled P1 through P4.
- There are some resource instances already allocated to processes, as follows:
  - one instance of R1 held by P2, another held by P3
  - one instance of R2 held by P1, another held by P4
- Some processes have requested additional resources, as follows:
  - P1 wants one instance of R1
  - P3 wants one instance of R2

a. (5 marks)
Draw the resource allocation graph for this system. Use the style of diagram from the lecture notes.

b. (4 marks)
Is this system deadlocked? If so, state which processes are involved. If not, give an execution sequence that eventually ends, showing resource acquisition and release at each step.

https://student.cs.uwaterloo.ca/~cs350/common/old-exams/S06midSol.pdf

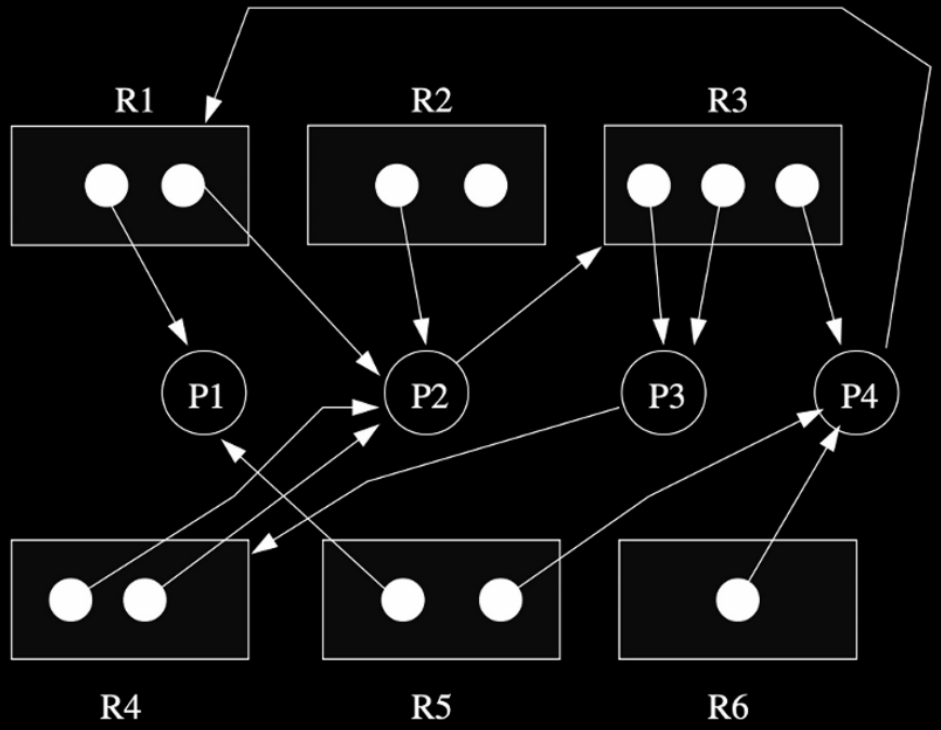

P1
P4
R1
R2
P2
P3



Not deadlocked (even though a cycle exists in the graph). One possible execution sequence: P2, P1, P4, P3.

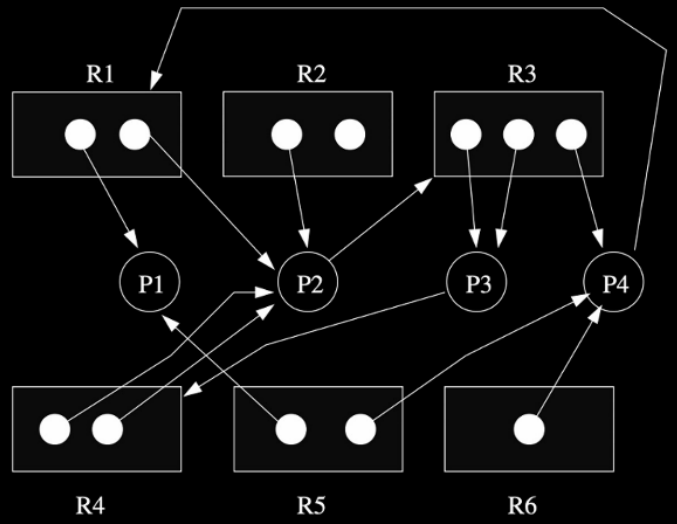
Question:

Consider the following resource allocation graph, which is like the ones shown in the course notes and discussed in class. Is there a deadlock in this system? Answer "yes" or "no". If your answer is yes, indicate which processes are involved in the deadlock. If your answer is "no", indicate at least one sequence in which the processes could run to completion, e.g., "P1, then P2, then P3, then P4".



https://student.cs.uwaterloo.ca/~cs350/common/old-exams/S07mid1sol.pdf

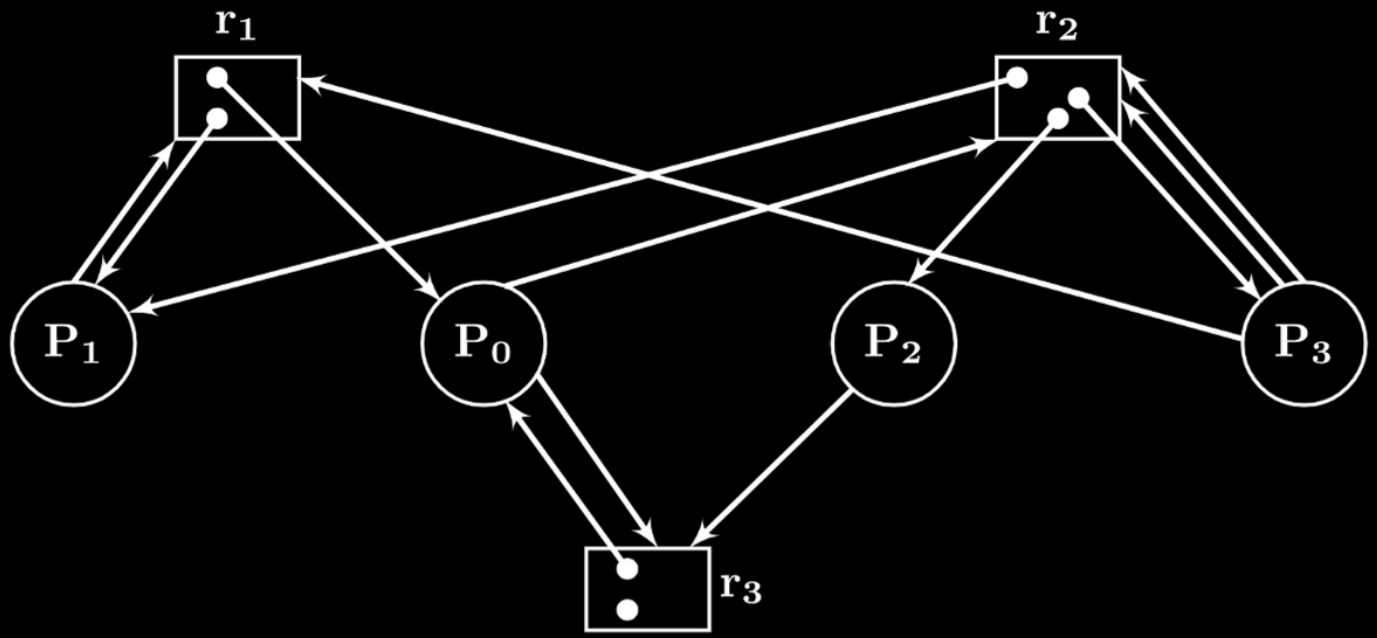
Consider the following resource allocation graph, which is like the ones shown in the course notes and discussed in class. Is there a deadlock in this system? Answer "yes" or "no". If your answer is yes, indicate which processes are involved in the deadlock. If your answer is "no", indicate at least one sequence in which the processes could run to completion, e.g., "P1, then P2, then P3, then P4".



There is no deadlock. Processes can run to completion in the order P1,P4,P2,P3.

GATE CSE 1994

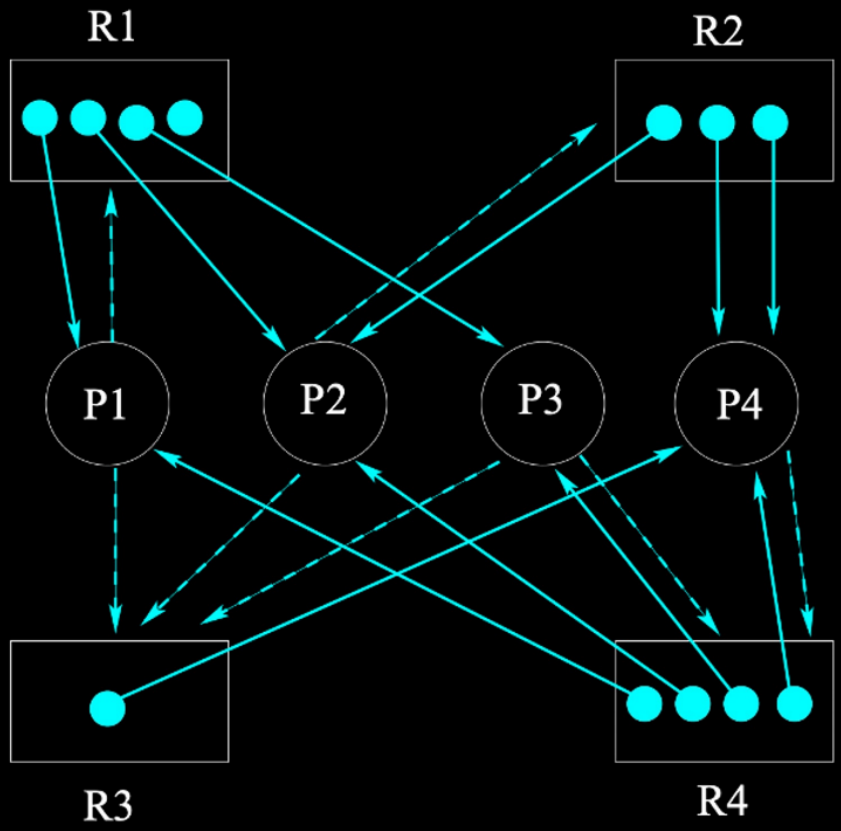
Is given system in deadlock ?



H.W.

Question

Is given system in deadlock ?



https://student.cs.uwaterloo.ca/~cs350/common/old-exams/W10-midterm-sol.pdf

The system IS DEADLOCKED.

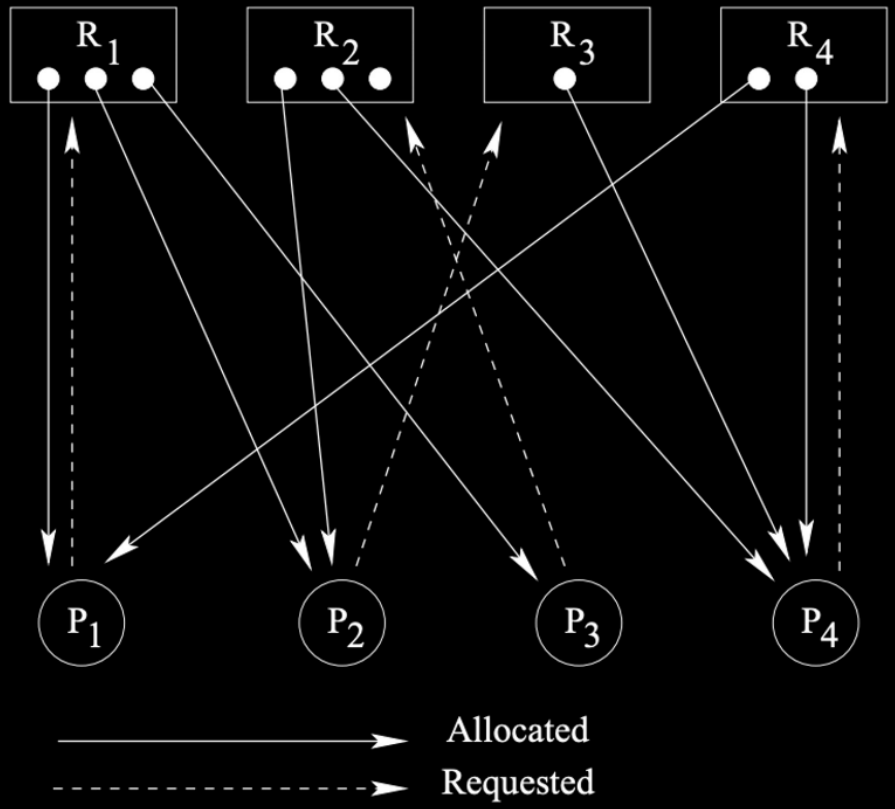
The only resource available is one R_1 and

Each of P_1, P_2, P_3 and P_4 need an R_3 or an R_4 and there aren't any of those available so there is a deadlock.

Or the only unallocated resource is 1 R_1 and giving it to ANY of the P_i's doesn't help any of them to finish, so there is deadlock.

Question

Is given system in deadlock ?



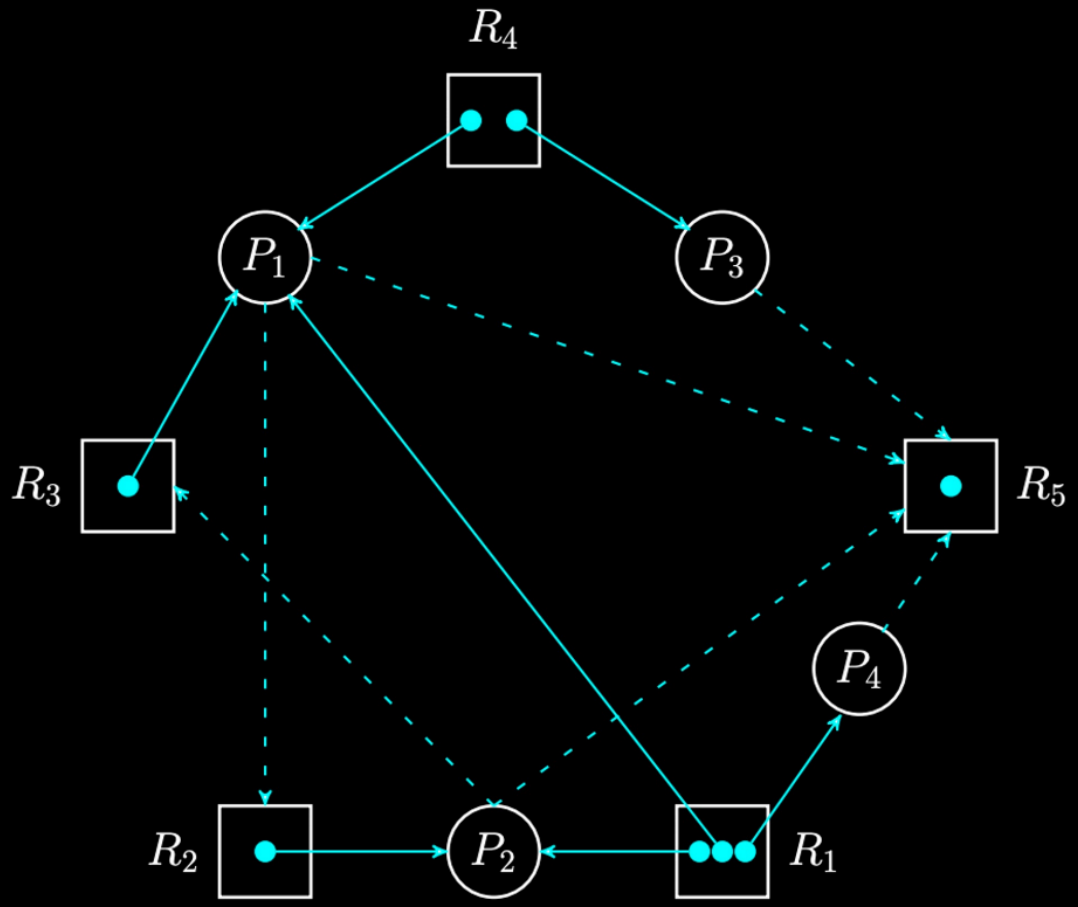
https://student.cs.uwaterloo.ca/~cs350/common/old-exams/F07-midterm-sol.pdf

The system is NOT deadlocked.

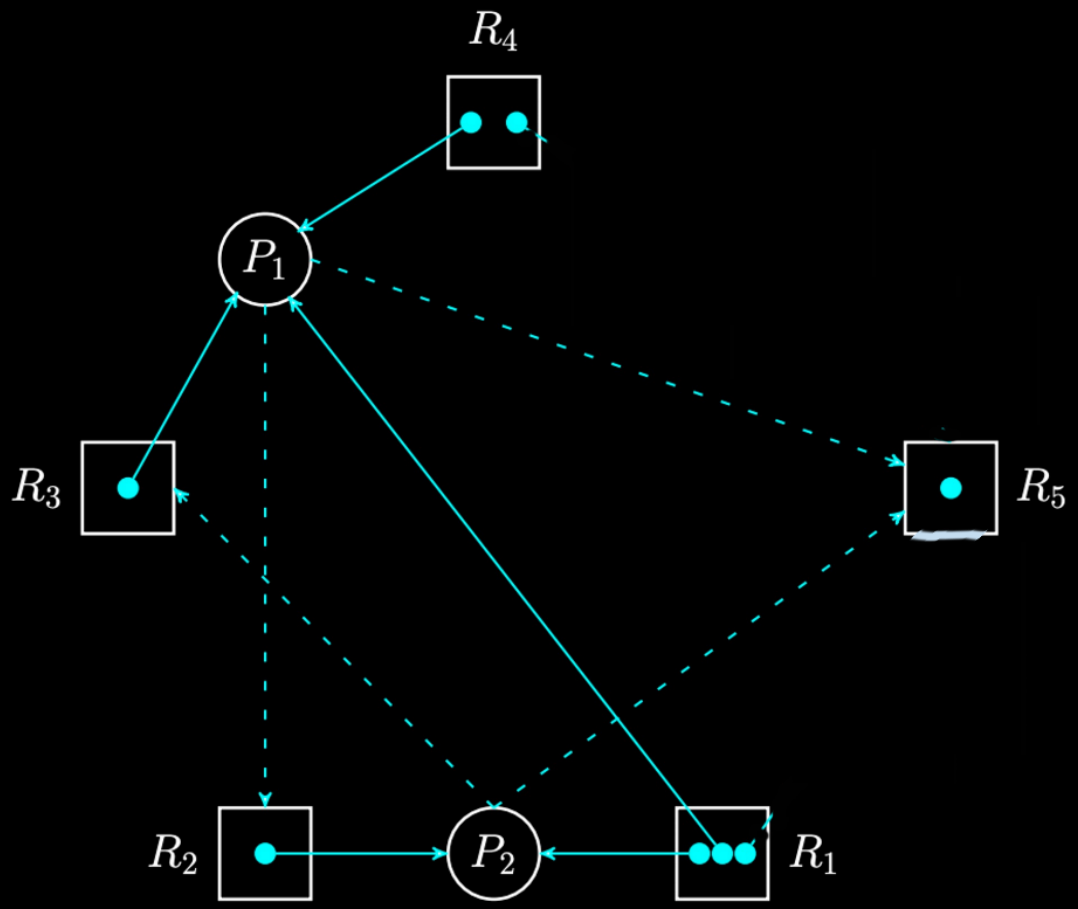
P_3 needs and R_2 and one is unallocated so assign it to P_3 then,
it (P_3) has all the resourse it needs so it finishes and free $1xR_1$,
P_1 gets R_1 and then it can finish and free $2xR_1 and 1xR_4$
P_4 gets R_4 and then it can finish and free $2xR_4 and 1xR_3$
P_2 gets R_3 and then it can finish and free $1xR_1 and 1xR_2 and 1xR_3$

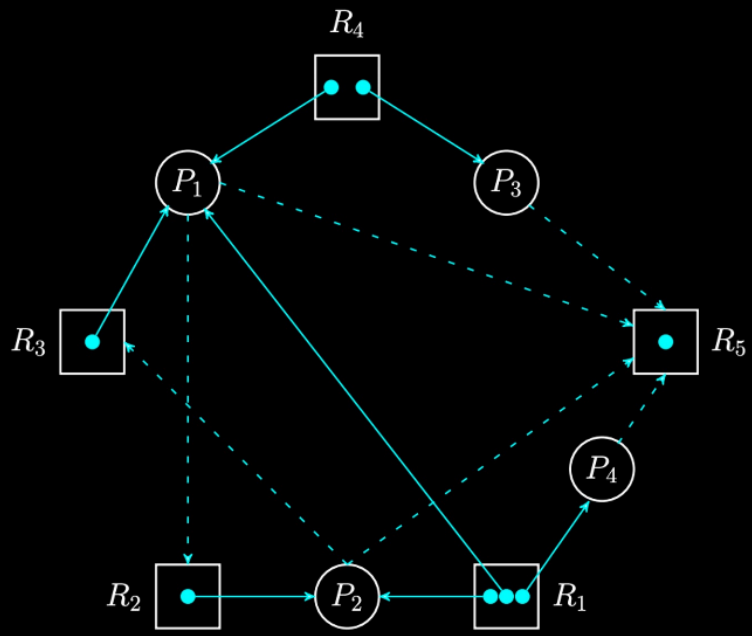
Question

Is given system in deadlock ?



https://student.cs.uwaterloo.ca/~cs350/common/old-exams/S11-midterm-sol.pdf


R_4
P_1
R_3
R_5
R_2
P_2
R_1



So the system is deadlocked. P_1 is allocated R_3 and it needs R_2 to finish and P_2 is allocated R_2 and it needs R_3 to finish.

GATE CSE 2018 | Question: 24

30

Consider a system with 3 processes that share 4 instances of the same resource type. Each process can request a maximum of K instances. Resources can be requested and releases only one at a time. The largest value of K that will always avoid deadlock is ___

gatecse-2018 | operating-system | deadlock-prevention-avoidance-detection | easy | numerical-answers

$3(K-1)+1 \leq 4$

$K-1 \leq 1 \Rightarrow K \leq 2$

https://gateoverflow.in/204098/gate-cse-2018-question-24

74

Number of processes $= 3$
Number of Resources $= 4$

Let's distribute each process one less than maximum demand $(K-1)$ resources. i.e.
$3(K-1)$

Provide an additional resource to any of three processes for deadlock avoidance.

Best answer

Total resources $= 3(K-1)+1 = 3K-2$

Now, this $3K-2$ should be less than or equal to the number of resources we have right now.

$3K-2 \leq 4$

$\implies 3K \leq 6$

$\implies K \leq 2$

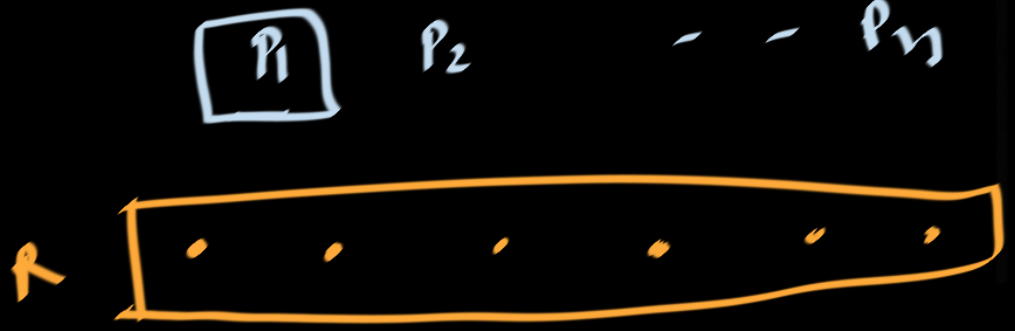
So, largest value of $K = 2$

GATE CSE 1992 | Question: 02-xi

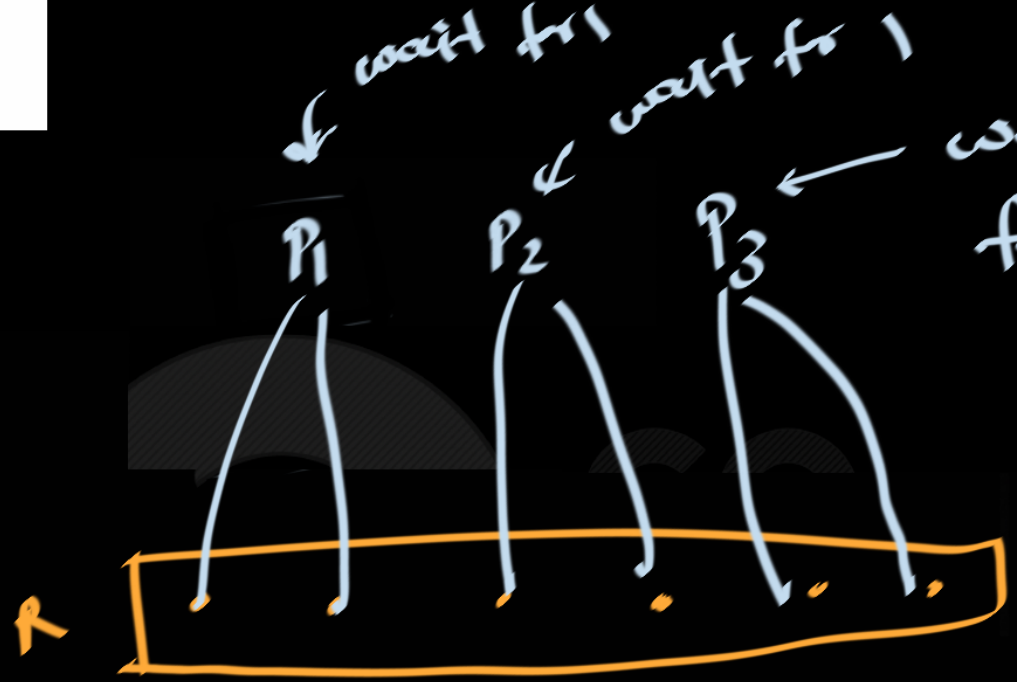
21

A computer system has 6 tape devices, with n processes competing for them. Each process may need 3 tape drives. The maximum value of n for which the system is guaranteed to be deadlock-free is:

A. 2
B. 3
C. 4
D. 1



https://gateoverflow.in/568/gate-cse-1992-question-02-xi



deadlock can not occur

$2n+1$ = total resources

6

n: number of process

$2n+1 = 6$

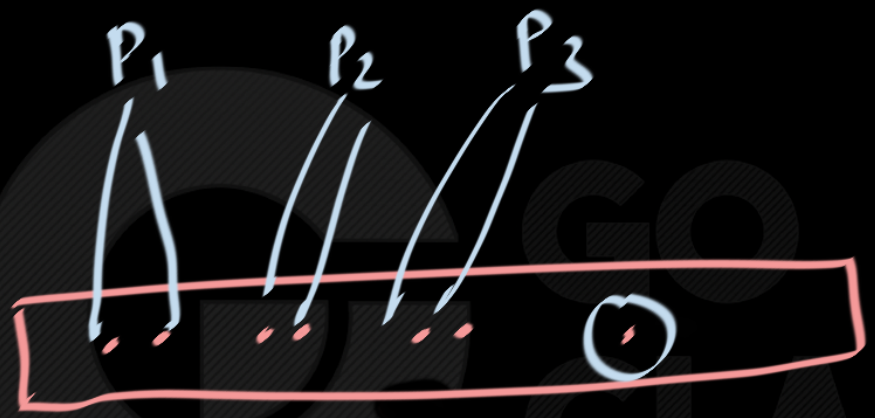
resources we eat up

$\Rightarrow n = 2$

$n = \left\lfloor \frac{5}{2} \right\rfloor$

$= 2.5$

$= 2$



total No. of resources $> \left(\sum_{i=1}^{n} (\max_i - 1) \right) + 1$

GO CLASSES

	P_1	P_2	P_3
max need	5	4	2

How many minimum resources need to make sure there is No deadlock?

$4+3+1 = 8$

$8+1 = 9$ ← minimum one need

GO CLASSES

	P_1	P_2	P_3
max need	5	4	2

How many minimum resources need to make sure there is No deadlock? $4+3+1 = 8$

$8+1 = 9$ ← minimum one need

with 11 resources ⇒ no deadlock

with 8 resources ⇒ deadlock possible.

with 2 resources ⇒ deadlock Always

with 3 " ⇒ deadlock possible

11

Best answer

Allocate max-1 resources to all processes and add one more resource to any process (Pigeon hole principle) so that this particular process can be completed (resources can be freed) and there is no deadlock.

Max resources required is 3.

$\therefore (3-1) * n + 1 = 6$

$n = \left\lfloor \frac{5}{2} \right\rfloor = 2$

Correct Answer: A

GATE CSE 1993 | Question: 7.9, UGCNET-Dec2012-III: 41

22

Consider a system having m resources of the same type. These resources are shared by 3 processes A, B, and C which have peak demands of 3, 4, and 6 respectively. For what value of m deadlock will not occur?

A. 7
B. 9
C. 10
D. 13
E. 15

A B C
3 4 6

3 + 3 + 5 = 10

(10)

https://gateoverflow.in/2297/gate-cse-1993-question-7-9-ugcnet-dec2012-iii-41

13 and 15.

45

Best answer

Consider the worst scenario: all processes require one more instance of the resource. So, $P1$ would have got 2, $P2-3$ and $P3-5$. Now, if one more resource is available at least one of the processes could be finished and all resources allotted to it will be free which will lead to other processes also getting freed. So, $2+3+5=10$ would be the maximum value of m so that a deadlock can occur.

Arjun

GATE CSE 1997 | Question: 6.7

23

An operating system contains 3 user processes each requiring 2 units of resource R. The minimum number of units of R such that no deadlocks will ever arise is

A. 3
B. 5
C. 4
D. 6

H.W.

https://gateoverflow.in/2263/gate-cse-1997-question-6-7

18

Best answer

If we have X number of resources where X is sum of $r_i - 1$ where r_i is the resource requirement of process i, we might have a deadlock. But if we have one more resource, then as per Pigeonhole principle, one of the process must complete and this can eventually lead to all processes completing and thus no deadlock.

Here, $n = 3$ and $r_i = 2$ for all i. So, in order to avoid deadlock **minimum** no. of resources required

$$= \sum_{i=1}^{3}(2-1) + 1 = 3 + 1 = 4.$$

PS: Note the **minimum** word, any higher number will also cause no deadlock.

Correct Answer: C

GATE CSE 1998 | Question: 1.32

25

A computer has six tape drives, with n processes competing for them. Each process may need two drives. What is the maximum value of n for the system to be deadlock free?

A. 6
B. 5
C. 4
D. 3

$n+1 = 6$

$n = 5$

https://gateoverflow.in/1669/gate-cse-1998-question-1-32

29

Best answer

Each process needs 2 drives

Consider this scenario

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

This is scenario when a deadlock would happen, as each of the process is waiting for 1 more process to run to completion. And there are no more Resources available as max 6 reached. If we could have provided one more R to any of the process, any of the process could have executed to completion, then released its resources, which further when assigned to other and then other would have broken the deadlock situation.

In case of processes, if there are less than 6 processes, then no deadlock occurs.

Consider the maximum case of 5 processes.

P_1	P_2	P_3	P_4	P_5
1	1	1	1	1

In this case system has 6 resources max,and hence we still have 1 more R left which can be given to any of the processes, which in turn runs to completion, releases its resources and in turn others can run to completion too.

Answer (B).

GATE CSE 2005 | Question: 71

39

Suppose n processes, $P_1, \ldots P_n$ share m identical resource units, which can be reserved and released one at a time. The maximum resource requirement of process P_i is s_i, where $s_i > 0$. Which one of the following is a sufficient condition for ensuring that deadlock does not occur?

A. $\forall i,\ s_i, < m$

B. $\forall i,\ s_i < n$

C. $\sum_{i=1}^{n} s_i < (m+n)$

D. $\sum_{i=1}^{n} s_i < (m \times n)$

GATE CSE 2005 | Question: 71

39

Suppose n processes, $P_1, \ldots P_n$ share m identical resource units, which can be reserved and released one at a time. The maximum resource requirement of process P_i is s_i, where $s_i > 0$. Which one of the following is a sufficient condition for ensuring that deadlock does not occur?

A. $\forall i,\ s_i, < m$

B. $\forall i,\ s_i < n$

C. $\sum_{i=1}^{n} s_i < (m+n)$

D. $\sum_{i=1}^{n} s_i < (m \times n)$

$$\sum_{i=1}^{n} (s_i - 1) + 1 \leq m$$

$$\sum_{i=1}^{n} s_i - n + 1 \leq m$$

113

Best answer

To ensure deadlock never happens allocate resources to each process in following manner:
Worst Case Allocation (maximum resources in use without any completion) will be $(\max \text{requirement} - 1)$ allocations for each process. i.e., $s_i - 1$ for each i

Now, if $\sum_{i=1}^{n} (s_i - 1) \leq m$ dead lock can occur if m resources are split equally among the n processes and all of them will be requiring one more resource instance for completion.

Now, if we add just one more resource, one of the process can complete, and that will release the resources and this will eventually result in the completion of all the processes and deadlock can be avoided. i.e., to avoid deadlock

$$\sum_{i=1}^{n} (s_i - 1) + 1 \leq m$$

$$\implies \sum_{i=1}^{n} s_i - n + 1 \leq m$$

$$\implies \sum_{i=1}^{n} s_i < (m + n).$$

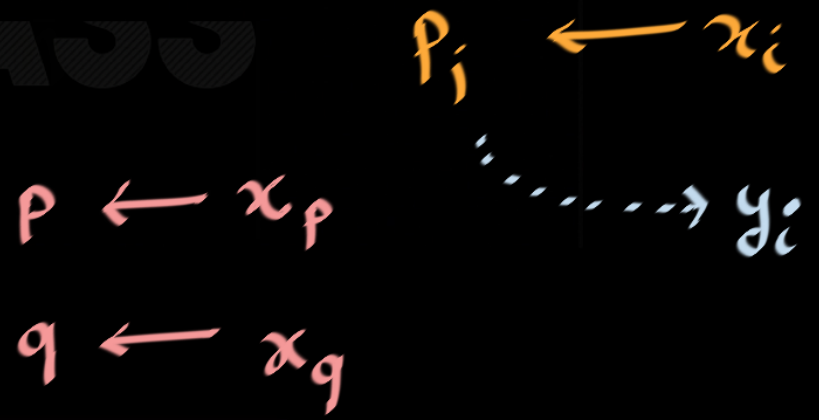
Correct Answer: C

GATE CSE 2006 | Question: 66

51

Consider the following snapshot of a system running n processes. Process i is holding x_i instances of a resource $R, 1 \leq i \leq n$. Currently, all instances of R are occupied. Further, for all i, process i has placed a request for an additional y_i instances while holding the x_i instances it already has. There are exactly two processes p and q and such that $y_p = y_q = 0$. Which one of the following can serve as a necessary condition to guarantee that the system is not approaching a deadlock?

A. $\min(x_p, x_q) < \max_{k \neq p,q} y_k$
B. $x_p + x_q \geq \min_{k \neq p,q} y_k$
C. $\max(x_p, x_q) > 1$
D. $\min(x_p, x_q) > 1$



$x_p + x_q$

$p_i \leftarrow x_i$

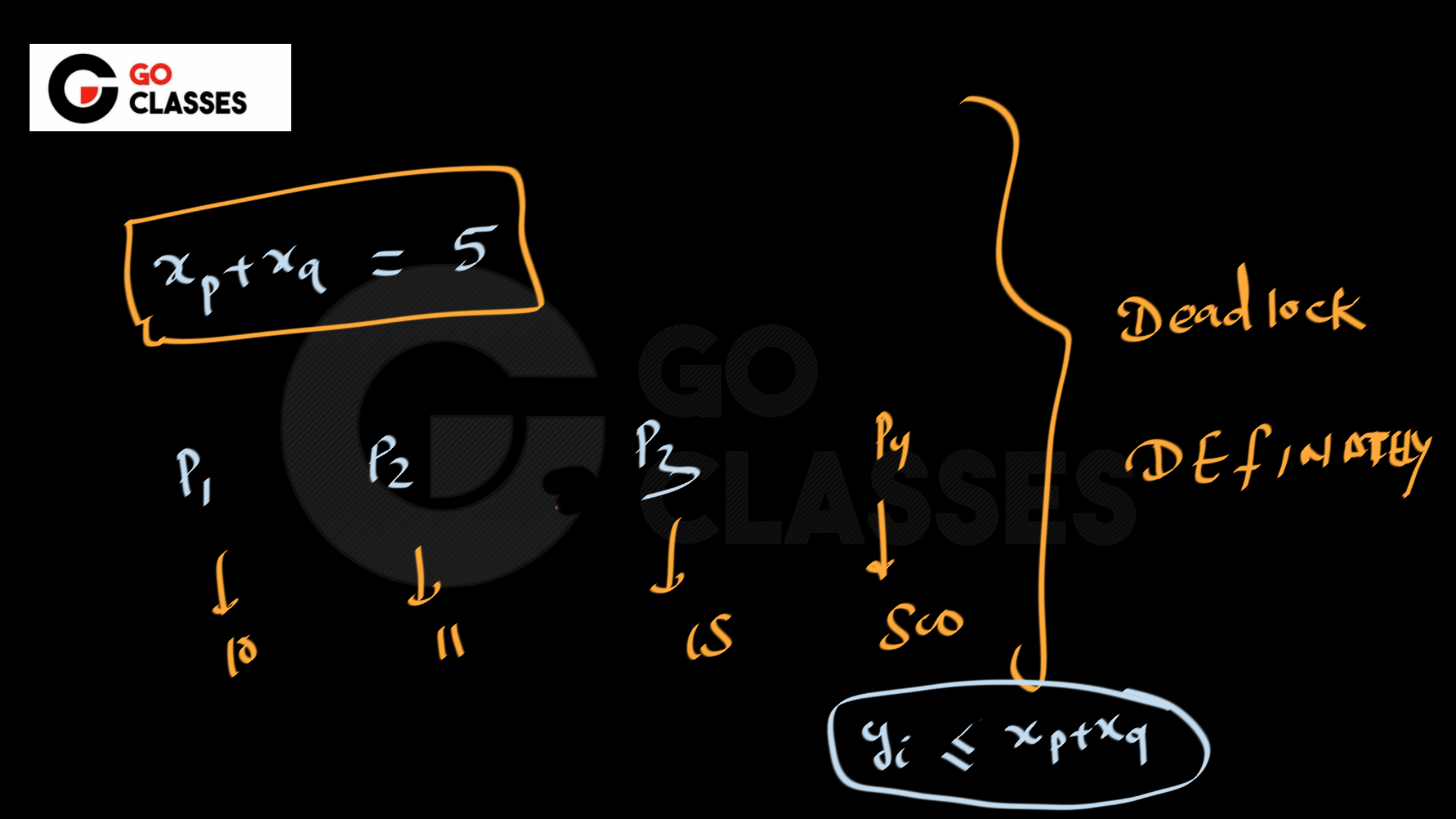
$\dashrightarrow y_i$

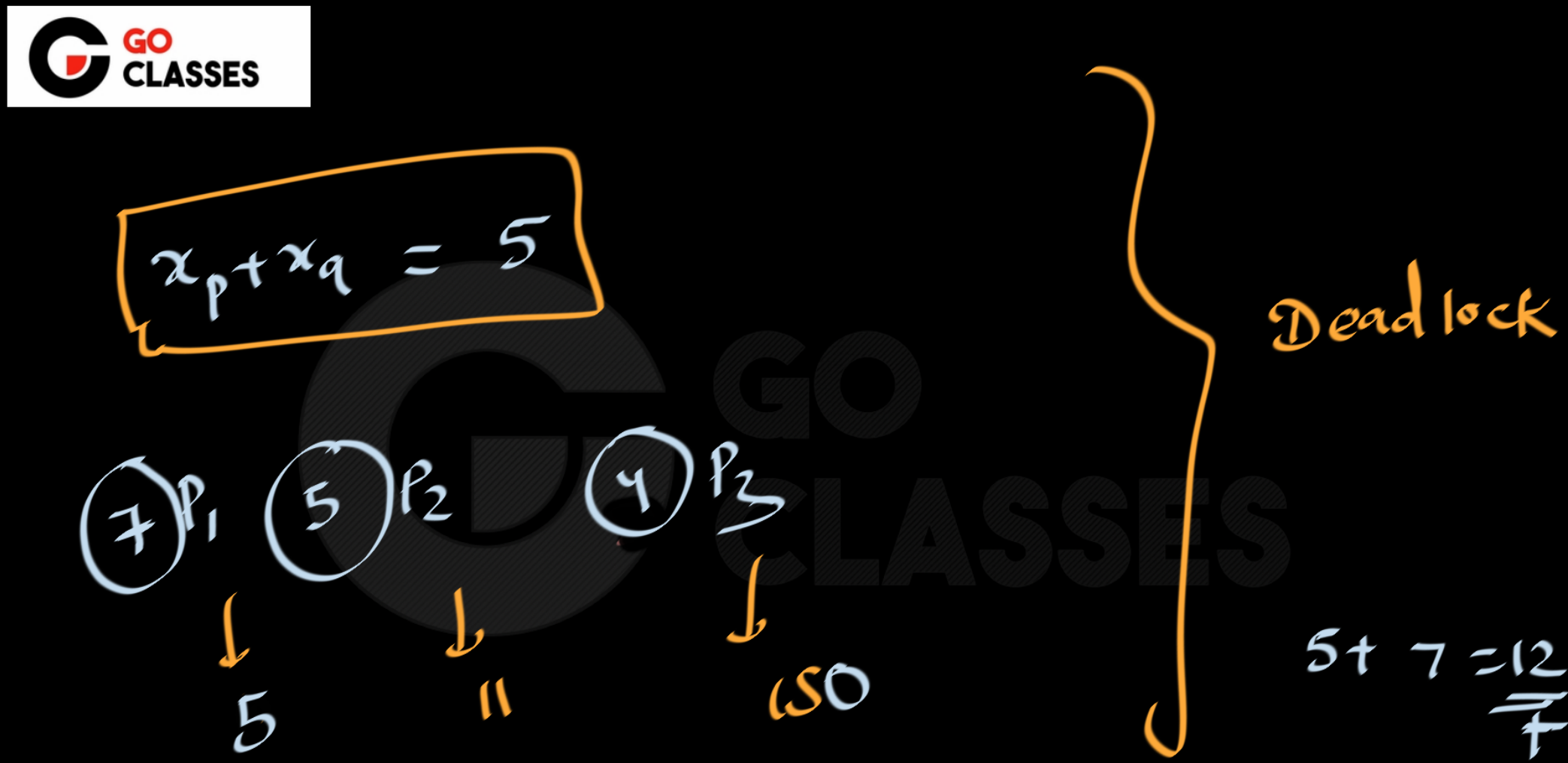
p_s ✓ $y_s \le x_p + x_q$

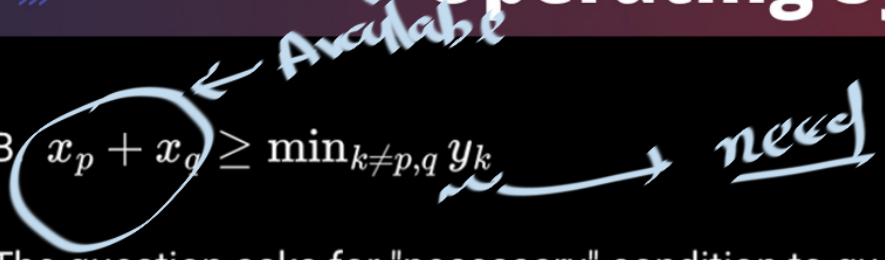
p, q, s

↓

✓ $x_p + x_q + x_s$


$x_p + x_q = 5$
P_1
P_2
P_3
P_4
10
11
15
Sc
$y_i \leq x_p + x_q$
Deadlock
DEFINATELY


$x_p + x_q = 5$
7 P1
5 P2
4 P3
5
11
150
Deadlock
5+7=12
=7
5
=17



70

Best answer

B $x_p + x_q \geq \min_{k \neq p,q} y_k$

The question asks for "necessary" condition to guarantee no deadlock. i.e., without satisfying this condition "deadlock" MUST be there.

Both the processes p and q have no additional requirements and can be finished releasing x_p + x_q resources. Using this we can finish one more process only if condition B is satisfied.

PS: Condition B just ensures that the system can proceed from the current state. It does not guarantee that there won't be a deadlock before all processes are finished.

answered Jun 9, 2015 • edited Dec 14, 2020 by **gatecse**

comment Follow share this

Arjun

GATE CSE 2014 Set 3 | Question: 31

20

A system contains three programs and each requires three tape units for its operation. The minimum number of tape units which the system must have such that deadlocks never arise is _____.

gatecse-2014-set3 operating-system resource-allocation numerical-answers easy

https://gateoverflow.in/2065/gate-cse-2014-set-3-question-31

27

Up to, 6 resources, there can be a case that all process have 2 each and dead lock can occur. With 7 resources, at least one process's need is satisfied and hence it must go ahead and finish and release all 3 resources it held. So, no dead lock is possible.

Best answer

answered Oct 21, 2014 • selected Jun 26, 2018 by **Arjun**

comment Follow share this

Arjun

GATE CSE 2015 Set 2 | Question: 23

39

A system has 6 identical resources and N processes competing for them. Each process can request at most 2 requests. Which one of the following values of N could lead to a deadlock?

A. 1
B. 2
C. 3
D. 4

https://gateoverflow.in/8114/gate-cse-2015-set-2-question-23

39

$3 \times 2 = 6$
$4 \times 2 = 8$

I guess a question can't get easier than this- (D) choice. (Also, we can simply take the greatest value among choice for this question)

Best answer

[There are 6 resources and all of them must be in use for deadlock. If the system has no other resource dependence, $N = 4$ cannot lead to a deadlock. But if $N = 4$, the system can be in deadlock in presence of other dependencies.

Why $N = 3$ cannot cause deadlock? It can cause deadlock, only if the system is already in deadlock and so the deadlock is independent of the considered resource. Till $N = 3$, all requests for considered resource will always be satisfied and hence there won't be a waiting and hence no deadlock with respect to the considered resource.]

answered Feb 12, 2015 • edited Jul 9, 2018 by **kenzou**

comment Follow share this


Arjun

GO CLASSES

Relationship b/w

number of Processes

and

number of Resources

for deadlock to not occur:

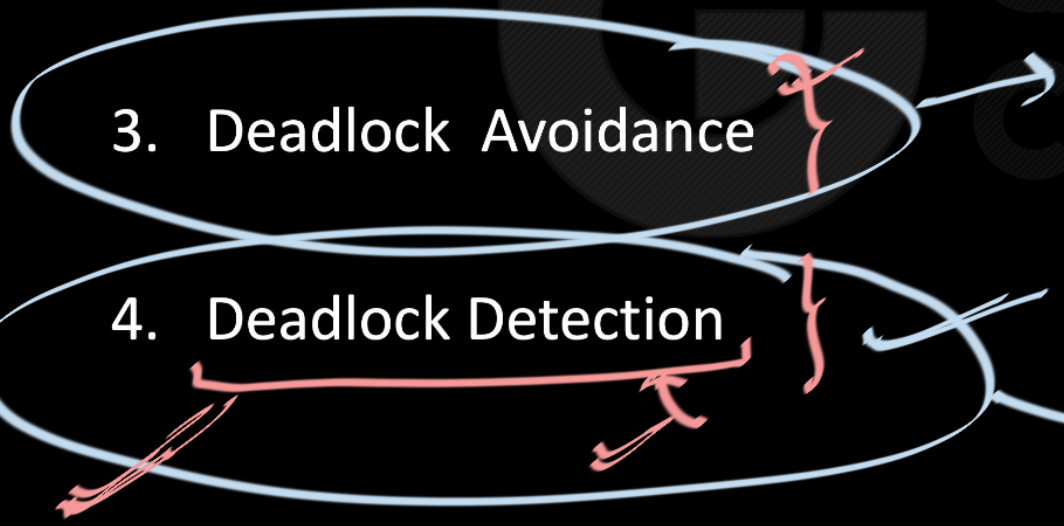
S_i : max resources any ith process need

$$\left[\left[\sum_{i=1}^{n}(S_i - 1)\right] + 1\right] \leq \text{total resources}$$

← Cond^n to avoid to deadlock

Strategies for dealing with Deadlocks

1. Just ignore the problem altogether
2. Deadlock Prevention
3. Deadlock Avoidance
4. Deadlock Detection



we somehow make sure that deadlock will never occur

we ALLOW deadlock, we detect it, then we can kill some process

Strategies for dealing with Deadlocks -1

1. Just ignore the problem altogether

- Ignore the problem and pretend that deadlocks never occur in the system;
  - Restart the system “manually” if the system “seems” to be deadlocked or stops functioning.
  - Note that the system may be “frozen” temporarily!
  - used by most operating systems, including UNIX

Ignore Deadlocks: ← **Most Operating systems do this!!**

The Ostrich Algorithm:

Put your head in the sand and pretend there is no problem at all

Strategies for dealing with Deadlocks - 2

2. Deadlock Prevention

Prevention by negating one of the four necessary conditions for deadlock.

→ Mutual exclusion of resources : Sorry can't do anything
→ Hold and wait
→ No preemption of resources : Sorry
→ circular wait

{ Attacking the Mutual Exclusion Condition

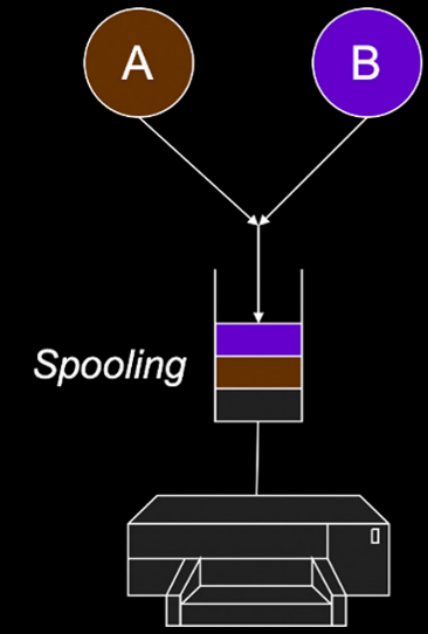
↓

Not possible to attack by nature

Resources are not Shareable

Attacking the Mutual Exclusion Condition

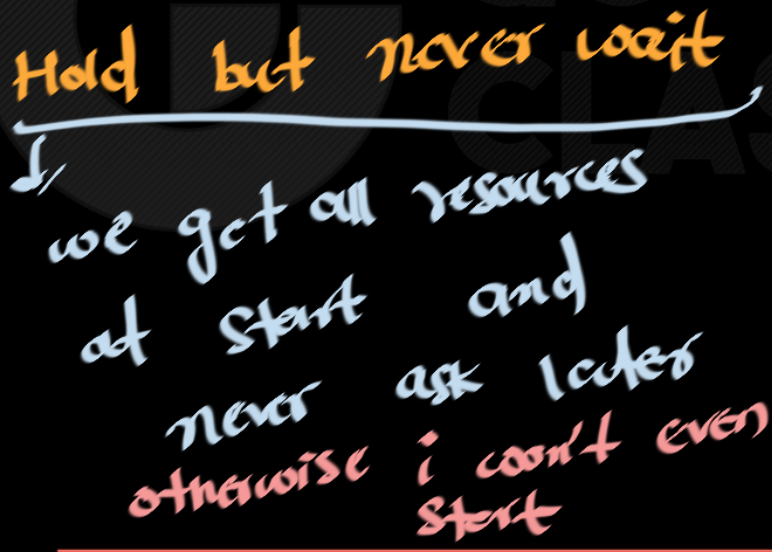
- Share the resources but most of the time resources are not sharable
  - Some devices (such as printer) can be spooled
    - only the printer daemon uses printer resource
    - thus deadlock for printer eliminated
  - Not all devices can be spooled



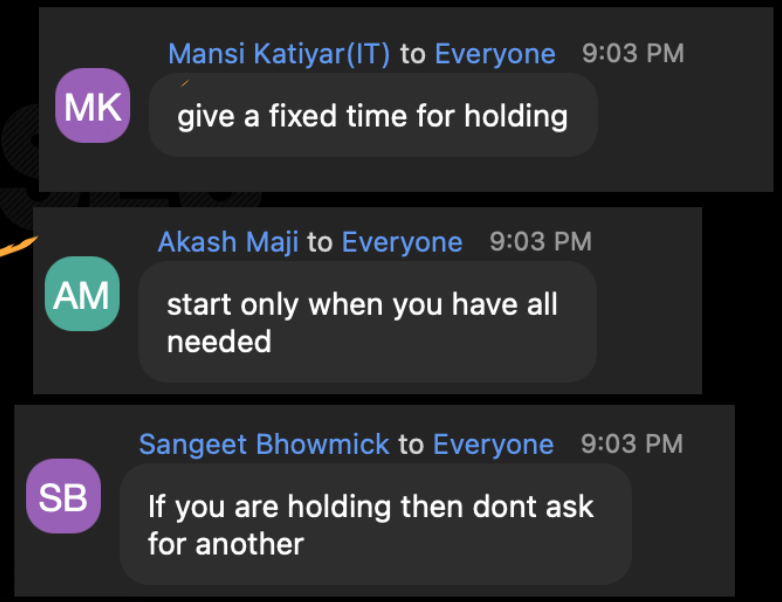
Attacking the Hold and Wait Condition

Process should either hold or wait but not together.

- Solution1:



- Solution2:



GO CLASSES

Problem with Soln 1

P_1 need 3 resources → DVP, file, printer

P_1 → Copy some data from DVP to file and print it later

P_1 need DVP + file

later need file + printer

Soln suggest us to get DVP, file, and printer before start of execution but it will lead to low resource utilization

Attacking the Hold and Wait Condition

Hold + wait

Process should either hold or wait but not together.

- Solution1:

Hold ✓ wait ✗

Hold but never wait

we get all resources at start and never ask later otherwise i can't even start

- Solution2:

✓ you can wait but whenever you want to wait, don't hold.

Mansi Katiyar(IT) to Everyone 9:03 PM
MK
give a fixed time for holding

Akash Maji to Everyone 9:03 PM
AM
start only when you have all needed

Sangeet Bhowmick to Everyone 9:03 PM
SB
If you are holding then dont ask for another


Dvd + file
Copy
wait
Printer

GO CLASSES

~~Hold and wait~~

Soln 1 : Hold but never wait

Soln 2 : if you want to wait then you release whatever you already hold on.

Attacking the Hold and Wait Condition

Process should either hold or wait but not together.

- Solution1:
  If ALL resources are available then acquire or just wait for ALL.

- Solution2:
  If a process is holding some resource and trying to acquire some other resource then it must release all the resources first

2. Attacking the Hold and Wait Condition

Problems in Soln

- Require processes to request resources before starting
  - a process never has to wait for what it needs
- Problems
  - may not know required resources at start of run
  - also ties up resources other processes could be using

3. Attacking the No Preemption Condition

- This is not a viable option
- Consider a process given the printer
  - halfway through its job
  - now forcibly take away printer
  - !!??

Sorry can't do anything

4. Attacking the Circular Wait Condition

Impose a total ordering of all resource types and require each process to request resources in increasing order

Resource No

3 { Printer

5 { Speaker

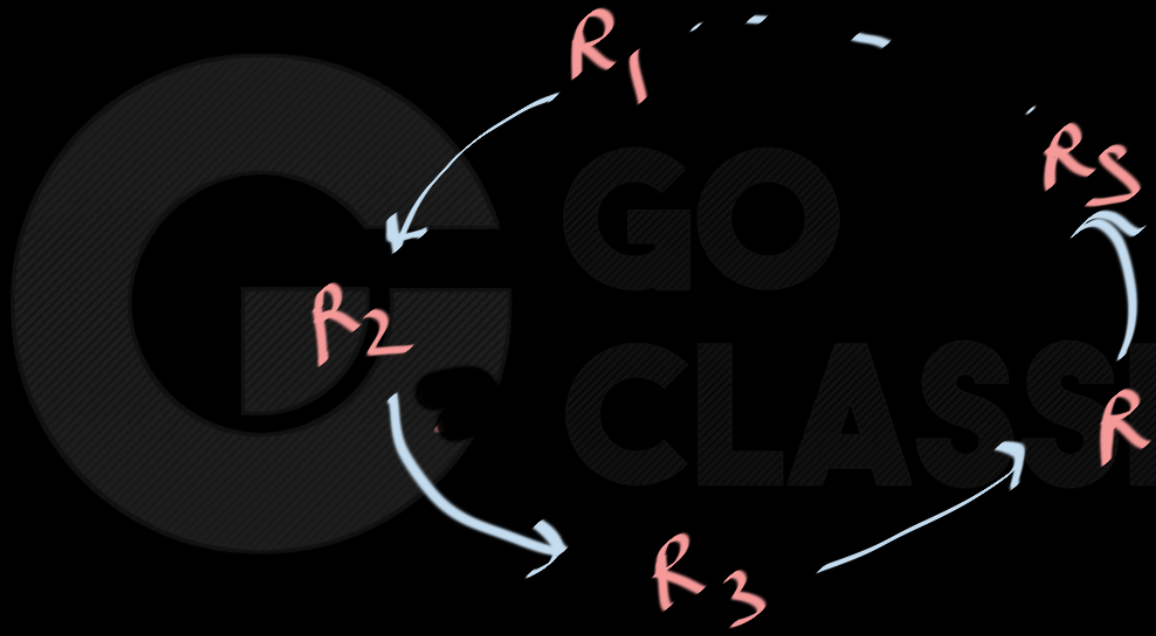
Attacking the Circular Wait Condition

Impose a total ordering of all resource types and require each process to request resources in increasing order

Resource Numbers

- tape drive = R1
- disk drive = R2
- printer = R10

We can now consider the following protocol to prevent deadlocks: 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 —say, R_i. After that, the process can request instances of resource type R_j if and only if $j > i$.


R1
R2
R3
R4
R5

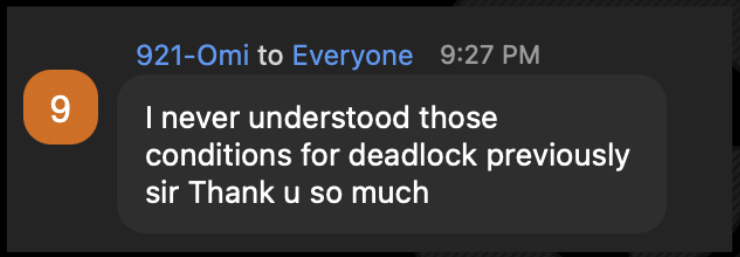
Deadlock Prevention Summary

Condition	Negation of condition	Approach to prevent condition	Comments about Practicality of approach
Mutual Exclusion	No Mutual Exclusion	Spool Everything	Only possible for few resources like Printers. Not possible for all resources. SORRY
Hold and wait	No Hold and wait	Request all resources initially	Require process to request and be allocated all its resources before it begins execution, or allow process to request resources only when the process has none. Low resource utilization.
No preemption	Preemption is allowed	Take resources away	This is very inefficient since we may need to restart process every time we take away resource from it. SORRY
Circular wait	No Circular wait	Order Resources Numerically	**Yes, it is practical and easy to apply.**

Strategies for dealing with Deadlocks -3

3. Deadlock Avoidance

Banker's algorithm


921-Omi to Everyone 9:27 PM
9
I never understood those conditions for deadlock previously sir Thank u so much

GO CLASSES

1. Just ignore the problem altogether
2. Deadlock Prevention
3. Deadlock Avoidance
4. Deadlock Detection

→ we don't allow one of the cond^n

→ i avoid deadlock, before any activity (allocating resource), i 100% make sure that after allocating resource, system will Not go to deadlock. (here we can have those 4 neccessary cond^n's)

↳ (Banker's algorithm)

GO CLASSES

1. Just ignore the problem altogether
2. Deadlock Prevention → i avoid deadlock, before any activity (allocating resource), i 100% make sure that after allocating
3. Deadlock Avoidance
4. Deadlock Detection → we don't care about anything, we have one algorithm, if you want us, we can run that and check if your system is in deadlock, if it is in deadlock then you can kill process, restart System, ----