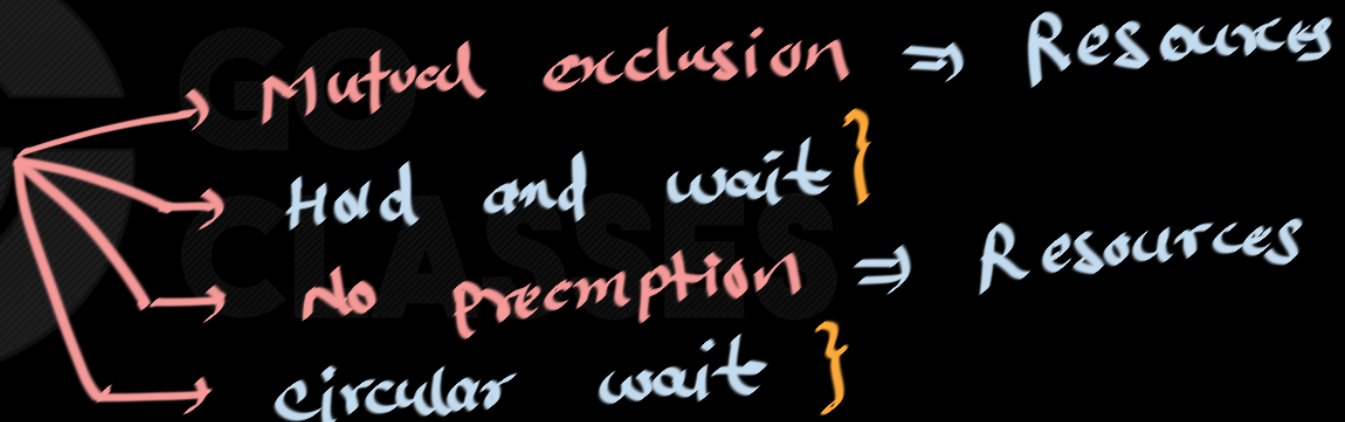
Lecture: 35

Deadlock Avoidance (Banker's Algorithm)
Detection & Recovery

Strategies for dealing with Deadlocks

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



"Sorry"

GO CLASSES

Resource (2) + 2 < Hold and wait, Circular wait

4 Conds exist → circular wait

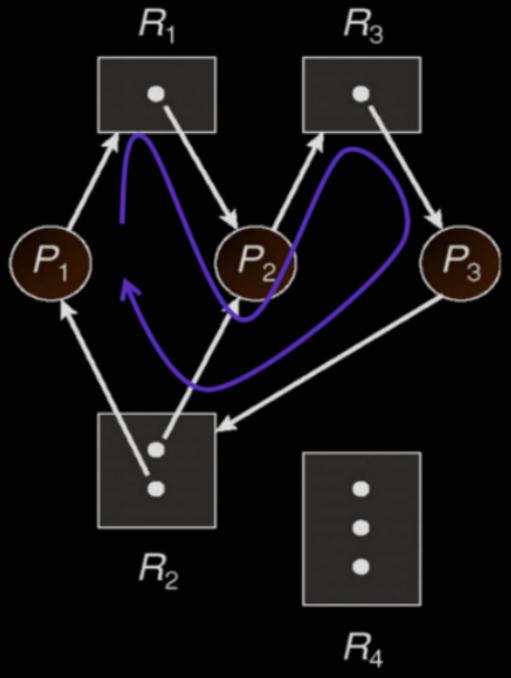
Even though 4 Conds exist

Still Deadlock may not exist

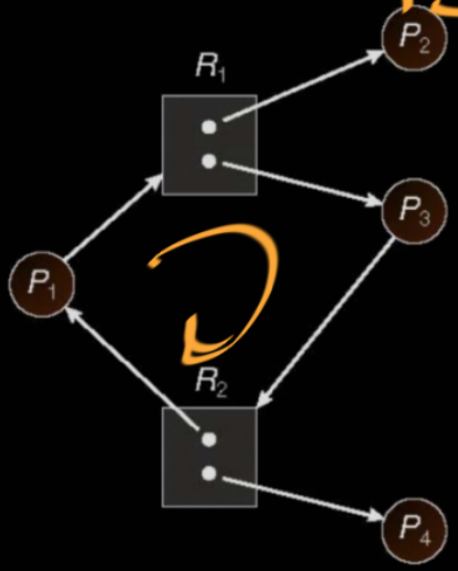
↳ Reason: there could be a cycle (circular wait)

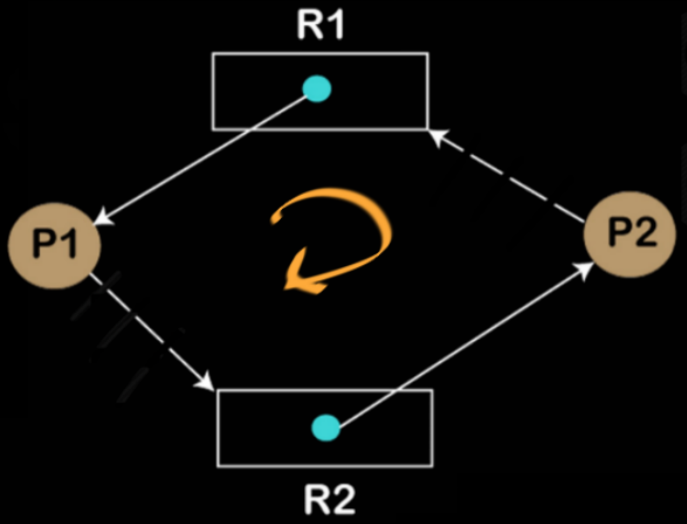
with multiple instance Resource allocation graph

Resource Allocation Graph With A Deadlock



Graph With A Cycle But No Deadlock





Deadlock

Strategies for deadlocks

In general, four strategies are used for dealing with deadlocks:

- **Ignore**: stick your head in the sand and pretend there is no problem at all.
- **Prevent**: design a system in such a way that the possibility of deadlock is excluded *a priori* (e.g., compile-time/statically, by design)
- **Avoid**: make a decision dynamically checking whether the request will, if granted, potentially lead to a deadlock or not (e.g., run-time/dynamically, before it happens)
- **Detect**: let the deadlock occur and detect when it happens, and take some action to recover after the fact (e.g., run-time/dynamically, after it happens)

Deadlock Avoidance ; Says that i will even allow 4 conditions to exist

i will just check whether system is in "safe state" once we allocate requests.

Strategies for dealing with Deadlocks -3

3. Deadlock Avoidance

Requires that the system has some additional ***a priori*** information available

- Simplest and most useful model requires that each process declare the ***maximum number*** of resources of each type that it may need
- The deadlock-avoidance algorithm dynamically examines the resource-allocation state to ensure that there can never be a circular-wait condition

Banker's Algorithm (Dijkstra, 1965)

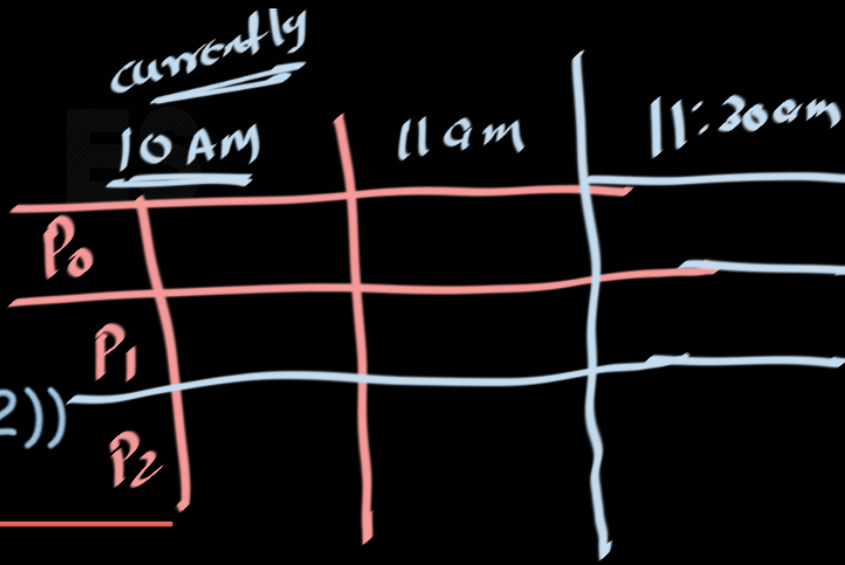
total tape drives = 12

max No. of resources a process need in future (in future)

May or may not Simultaneously

	current usage	could ask for
p0	5	5
p1	2	2
p2	2	7

3 drives remain (12 - (5+2+2))



total tape drives = 12

	current usage	could ask for
p0	5	5
p1	2	2
p2	2	7

P_1, P_0, P_2

Available = 12 - 9 = 3

Available after P_1 = 3 + 2 = 5

Available after P_0 = 5 + 5 = 10

total tape drives = 12

	current usage	could ask for
p0	5	5
p1	2	2
p2	2	7

System

P_1, P_0, P_2 ⇒ in this sequence i can fill requests

if we get requests in this order, we will definately grant.

GO CLASSES

Safe sequence: there is a sequence of Processes in which if we fulfill the requests then we will never face deadlock.

Allocated

need = max - Allocated

total tape drives = 12

	current usage	could ask for
p0	5	5
p1	2	2
p2	2	7

Max
10
4
9

Available

3 drives remain (12 - (5+2+2))

- current state is safe because a safe sequence exists: [p1, p0, p2]
  - p1 can complete with remaining resources 3
  - p0 can complete with remaining+p1 5
  - p2 can complete with remaining+p1+p0 7

total tape drives = 12

	current usage	could ask for
p0	5	5
p1	2	2
p2	2	7

Allocated

need

Available

3 drives remain (12 - (5+2+2))

- current state is safe because a safe sequence exists: [p1, p0, p2]
  - p1 can complete with remaining resources 3
  - p0 can complete with remaining+p1 5
  - p2 can complete with remaining+p1+p0 7

Algo to check safe seq. exist or not.

n: Number of process

work = Available

finish[1...n] = false

find Process i s.t.

need[i] ≤ work && finish[i] is false

work = work + Allocated[i]

finish[i] = true

```
work = avail ;       // accommodate all resources
finish[ 1 ... n ] = false;   // none finished yet

// find a process that can complete its work now
while (find i such that finish[i] == false  and need[i] <= work)
{
  work = work + alloc[i]
  finish[i] = true;
}

if (finish[i] == true for all i)
   return true;
else
   return false;
}
```

loop

Safety check algorithm

GO CLASSES

Allocated
nxm

Tape drive
Printer

	current usage		could ask for	
p0	5	2	5	2
p1	2	3	2	5
p2	2	6	7	0

Need
nxm

Printer

tapedrive

Avalible = | 3 | 6 |

n: Number of processes

m: number of Resources

Data Structures for the Banker's Algorithm

```
int n;              // # threads
int m;              // # resources
int avail[m]        // # of available resources of each type
int max[n,m]        // # of each resource that each thread may want
int alloc[n,m]      //# of each resource that each thread is using
int need[n,m]       // # of resources that each thread might still request
```

Need [i,j] = Max [i,j] − alloc [i,j] (they will give either Max or need)

Data Structures for the Banker's Algorithm

Let n = number of processes, and m = number of resources types.

- **Available**: Vector of length m. If available [j] = k, there are k instances of resource type R_j available
- **Max**: n x m matrix. If $Max[i,j] = k$, then process P_i may request at most k instances of resource type R_j
- **Allocation**: n x m matrix. If Allocation[i,j] = k then P_i is currently allocated k instances of R_j
- **Need**: n x m matrix. If $Need[i,j] = k$, then P_i may need k more instances of R_j to complete its task

$$Need[i,j] = Max[i,j] - Allocation[i,j]$$

GATE CSE 2017 Set 2 | Question: 33

27

A system shares 9 tape drives. The current allocation and maximum requirement of tape drives for that processes are shown below:

Process	Current Allocation	Maximum Requirement
P1	3	7
P2	1	6
P3	3	5

Need

Which of the following best describes current state of the system?

A. Safe, Deadlocked
B. Safe, Not Deadlocked
C. Not Safe, Deadlocked
D. Not Safe, Not Deadlocked

https://gateoverflow.in/118375/gate-cse-2017-set-2-question-33

GATE CSE 2017 Set 2 | Question: 33

27

A system shares 9 tape drives. The current allocation and maximum requirement of tape drives for that processes are shown below:

Process	Current Allocation	Maximum Requirement
P1	3	7
P2	1	6
P3	3	5

Which of the following best describes current state of the system?

A. Safe, Deadlocked
B. Safe, Not Deadlocked
C. Not Safe, Deadlocked
D. Not Safe, Not Deadlocked

Need
4
5
2

Available
2+3=5

P3 P1 P2

Available = 9 − 7 = 2

P3 P2 P1

https://gateoverflow.in/118375/gate-cse-2017-set-2-question-33

46

Process	Current Allocation	Max Requirement	Need
P1	3	7	4
P2	1	6	5
P3	3	5	2

Best answer

Given there are total 9 tape drives,

So, according to the above table we can see we have currently allocated (7 tape drive), so **currently Available tape drives** $= 2$

So, $P3$ can use it and after using it will release it 3 resources **New Available** $= 5$

then $P1$ can use it and will release it 3 resources so **New Available** $= 8$

and lastly $P2$ so, all the process are in **SAFE STATE** and there will be **NO DEADLOCK**

Safe Sequence will be $\mathbf{P3 \rightarrow P2 \rightarrow P1}$ or $\mathbf{P3 \rightarrow P1 \rightarrow P2.}$

Answer will be (B) only.

Question

Pattern : 1

Consider following state of the system -

total tape drives = 12

	current usage	need could ask for
p0	5	5
p1	2	2
p2	2	7

Available = 3

5 → P1 → 16

P1 P0 P2

↓

is this a Safe state ? ✓ Safe state

Question

Pattern : 2 (Banker's algorithm)

Consider following state of the system -

total tape drives = 12

	current usage	could ask for
p0	5	5
p1	2	2
p2	2	7

P_1 P_0 P_2 is Safe Sequence

Now suppose p2 request one drive and we assign it to p2 then would resulting state be in safe state ?

Consider following state of the system -

total tape drives = 12

	current usage	could ask for
p0	5	5
p1	2	2
p2	2	7

Assign 1 tape drive to P2

	current usage	could ask for
p0	5	5
p1	2	2
p2	3	6

Now suppose p2 request one drive and we assign it to p2 then would resulting state be in safe state ?

Available = 12 - 10 = 2

Unsafe state

total tape drives = 12

	current usage	could ask for	max need
p0	5	5	10
p1	2	2	4
p2	2	7	9

Assigning one resource to P2

total tape drives = 12

	current usage	could ask for	max need
p0	5	5	10
p1	2	2	4
p2	3	6	9

total tape drives = 12

	current usage	could ask for	max need
p0	5	5	10
p1	2	2	4
p2	2	7	9

This resulting state is unsafe hence System will not assign a resource to p2. Instead it will wait for p0 or p1 to ask for some resources.

Once other process finishes and give away their resources then system will assign a resource to p2

total tape drives = 12

	current usage	could ask for	max need
p0	5	5	10
p1	2	2	4
p2	3	6	9

1) Given State, we can check State is Safe or not.

2) Given state is safe, and we have a request from some process, then we can check should we grant the request or not

→ Pretend we have granted (change table)

→ check if changed state is safe

→ if safe then grant the req. otherwise don't

GO CLASSES

Banker's algorithm

Given state is safe, and we have a request from some process, then we can check should we grant the request or not

→ Pretend we have granted (change table)

→ check if changed state is safe

→ if safe then grant the req. otherwise don't

Banker's Algorithm (Dijkstra, 1965)

Consider a banker and few customers with credit cards.

Each customer has some maximum limit on credit card.

A customer can ask banker for maximum amount as per his/her credit limit.

Banker gives money to customer, and customer returns money (pays bill) at month end.

GO CLASSES

Bank should not be waiting for the customers to pay back the loans in order to give some more loans to some other Customers.

(OS)

Bank should not be waiting for the customers (Resources) to pay back the loans in order to give some more loans to some other customers. (Process)

Bankers Algorithm

- Modelled on a Banker with Customers
  - The banker has a limited amount of money to loan customers
    - Limited number of resources
  - Each customer can borrow money up to the customer's credit limit
    - Maximum number of resources required
- Basic Idea
  - Keep the bank in a *safe* state
    - So all customers are happy even if they all request to borrow up to their credit limit at the same time.
  - Customers wishing to borrow such that the bank would enter an unsafe state must wait until somebody else repays their loan such that the the transaction becomes safe.

Banker's Algorithm

Pretend

If some process requests for some resources, tentatively assume that we satisfy the request

Now, check if this would leave us in a safe state:

if yes, grant the request,

if no, then leave the state as is and cause process to wait.

Banker's Algorithm

- So...
  - A process pre-declares its worst-case needs
  - Then it asks for what it "really" needs, a little at a time
  - The algorithm decides when to grant requests
- It delays a request unless:
  - It can find a sequence of processes...
  - .... such that it could grant their outstanding need...
  - ... so they would terminate...
  - ... letting it collect their resources...
  - ... and in this way it can execute everything to completion!

Question

A restaurant would like to serve four dinner parties, P1 through P4. The restaurant has a total of 8 plates and 12 bowls. Assume that each group of diners will stop eating and wait for the waiter to bring a requested item (plate or bowl) to the table when it is required. Assume that the diners don't mind waiting. The maximum request and current allocation tables are shown as follows:

Maximum Request	Plates	Bowls
P1	7	7
P2	6	10
P3	1	2
P4	2	4

Current Allocation	Plates	Bowls
P1	2	3
P2	3	5
P3	0	1
P4	1	2

a. Determine the Need Matrix for plates and bowls.

b. Is the system in a safe state?

https://people.eecs.berkeley.edu/~kubitron/courses/cs162-F08/exams/sp04mt1-solutions.pdf

a.

Need	Plates	Bowls
P1	*5*	*4*
P2	*3*	*5*
P3	*1*	*1*
P4	*1*	*2*

Sol'n

b. *The vector of available resources is A = (2, 1).*
First, serve P3, and A will be (2, 2).
Then, serve P4, and A will be (3, 4).
There are not enough resources available to serve P1 or P2. Therefore, the original resource allocation state is ***unsafe****. The restaurant cannot feed all four parties successfully.*

If you did not say that the allocation is unsafe, we deducted one point.
If you said "yes" or safe, we deducted 4 points.
If you said that there was deadlock or starvation, we deducted 3 points (saying you can't finish ever, cost2 points).
If your solution did not include the Banker's Algorithm or you were not specific in your explanation, we deducted 2 points

Question from Galvin

8.9 Consider the following snapshot of a system:

Determine whether or not the following state is unsafe.
If the state is safe, illustrate the order in which the threads may complete.
Otherwise, illustrate why the state is unsafe.

	Allocation	*Max*
	A B C D	*A B C D*
T_0	3 0 1 4	5 1 1 7
T_1	2 2 1 0	3 2 1 1
T_2	3 1 2 1	3 3 2 1
T_3	0 5 1 0	4 6 1 2
T_4	4 2 1 2	6 3 2 5

b. $Available = (0, 3, 0, 1)$

Answer:

b. Not safe. Processes P_2, P_1, and P_3 are able to finish, but no remaining processes can finish.

GATE CSE 1996 | Question: 22

22

A computer system uses the Banker's Algorithm to deal with deadlocks. Its current state is shown in the table below, where $P0$, $P1$, $P2$ are processes, and $R0$, $R1$, $R2$ are resources types.

	Maximum Need			Current Allocation			Available		
	R0	R1	R2	R0	R1	R2	R0	R1	R2
P0	4	1	2	1	0	2	2	2	0
P1	1	5	1	0	3	1			
P2	1	2	3	1	0	2			

A. Show that the system can be in this state

B. What will the system do on a request by process $P0$ for one unit of resource type $R1$?

gate1996 | operating-system | resource-allocation | normal | descriptive

https://gateoverflow.in/2774/gate-cse-1996-question-22

GATE CSE 1996 | Question: 22

22

A computer system uses the Banker's Algorithm to deal with deadlocks. Its current state is shown in the table below, where $P0$, $P1$, $P2$ are processes, and $R0$, $R1$, $R2$ are resources types.

	Maximum Need			Current Allocation			Available		
	R0	R1	R2	R0	R1	R2	R0	R1	R2
P0	4	1	2	1	1	2	2	2	0
P1	1	5	1	0	3	1			
P2	1	2	3	1	0	2			

A. Show that the system can be in this state

B. What will the system do on a request by process $P0$ for one unit of resource type $R1$?

gate1996 | operating-system | resource-allocation | normal | descriptive

https://gateoverflow.in/2774/gate-cse-1996-question-22

26

Best answer

Allocation

	R0	R1	R2
P0	1	0	2
P1	0	3	1
P2	1	0	2

MAX NEED

	R0	R1	R2
P0	4	1	2
P1	1	5	1
P2	1	2	3

Future Need

	R0	R1	R2
P0	3	1	0
P1	1	2	0
P2	0	2	1

Available $= (2 \quad 2 \quad 0)$

$P1(1 \quad 2 \quad 0)$'s needs can be met. P1 executes and completes releases its allocated resources.

$A = (2 \quad 2 \quad 0) + (0 \quad 3 \quad 1) = (2 \quad 5 \quad 1)$

Further $P2(0 \quad 2 \quad 1)$ s needs can be met.

$A = (2 \quad 5 \quad 1) + (1 \quad 0 \quad 2) = (3 \quad 5 \quad 3)$

next $P0$ s needs can be met.

Thus safe sequence exists $P1P2P0$.

Next Request $P0(010)$

Allocation

	R0	R1	R2
P0	1	0+1=1	2
P1	0	3	1
P2	1	0	2

MAX NEED

	R0	R1	R2
P0	4	1	2
P1	1	5	1
P2	1	2	3

Future Need

	R0	R1	R2
P0	3	0	0
P1	1	2	0
P2	0	2	1

Available $= (2 \quad 2-1=1 \quad 0)$

Here, also not a single request need by any process can be made.

a. System is in safe state.
b. Since request of $P0$ can not be met,system would delay the request and wait till resources are available.

Question

Banker's Algorithm: Example

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	3	3	2
P1	2	0	0	3	2	2			
P2	3	0	2	9	0	2			
P3	2	1	1	2	2	2			
P4	0	0	2	4	3	3			

Is this system in safe state ?

Question

Available

A	B	C
3	3	2

	Allocation			Max			Need		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	7	4	3
P1	2	0	0	3	2	2	1	2	2
P2	3	0	2	9	0	2	6	0	0
P3	2	1	1	2	2	2	0	1	1
P4	0	0	2	4	3	3	4	3	1

	3	3	2
P_1	2	0	0
	5	3	2
P_3	2	1	1
	7	4	3

	7	4	3
P_0	0	1	0
	7	5	3

P_1 P_3 P_0 P_2 P_4 ← Safe seq.

this is a safe state:

safe sequence [P1, P3, P4, P2, P0]

Question

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	3	3	2
P1	2	0	0	3	2	2			
P2	3	0	2	9	0	2			
P3	2	1	1	2	2	2			
P4	0	0	2	4	3	3			

safe sequence [P1, P3, P4, P2, P0]

Suppose P1 requests (1,0,2), will Banker's algorithm allow this request ?

Now suppose that P1 requests (1,0,2)
add it to P1's allocation
subtract it from Available

We get the following state: -

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	3	3	2
P1	2	0	0	3	2	2			
P2	3	0	2	9	0	2			
P3	2	1	1	2	2	2			
P4	0	0	2	4	3	3			

Assign (1,0,2) to P_1

⇒

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	2	3	0
P1	3	0	2	3	2	2			
P2	3	0	2	9	0	2			
P3	2	1	1	2	2	2			
P4	0	0	2	4	3	3			

This is still safe: safe seq [P1, P3, P4, P0, P2].

Question

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	2	3	0
P1	3	0	2	3	2	2			
P2	3	0	2	9	0	2			
P3	2	1	1	2	2	2			
P4	0	0	2	4	3	3			

safe seq [P1, P3, P4, P0, P2].

Suppose P4 requests (3,3,0), will Banker's algorithm allow this request ?

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	2	3	0
P1	3	0	2	3	2	2			
P2	3	0	2	9	0	2			
P3	2	1	1	2	2	2			
P4	0	0	2	4	3	3			

P4 requests (3,3,0)

- not enough available resources: has to wait

Question

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	2	3	0
P1	3	0	2	3	2	2			
P2	3	0	2	9	0	2			
P3	2	1	1	2	2	2			
P4	0	0	2	4	3	3			

safe seq [P1, P3, P4, P0, P2].

Suppose P0 requests (0,2,0), will Banker's algorithm allow this request ?

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	2	3	0
P1	3	0	2	3	2	2			
P2	3	0	2	9	0	2			
P3	2	1	1	2	2	2			
P4	0	0	2	4	3	3			

P0 requests (0,2,0)

- there are enough resources, but... This is unsafe state (why?)

So P0 has to wait

GO CLASSES

Example : Consider following state

Where Number of tape drives = 12

	current usage	could ask for
p0	5	5
p1	2	2
p2	3	6

Available = 12 − 10 = 2

is this a safe state?

if not then what does unsafe state mean?

GO CLASSES

Example : Consider following state

Where Number of tape drives = 12

	current usage	could ask for
p0	5	5
p1	2	2
p2	3	6

4 available

P_1 , P_0 , P_2

Available = $12 - 10 = 2$

(MSQ)

n = 10

(made by me)

Question

Let $< P_1, P_2, \ldots, P_n >$ be the ONLY safe state of a system. Then which of the following is/are ALWAYS true ?

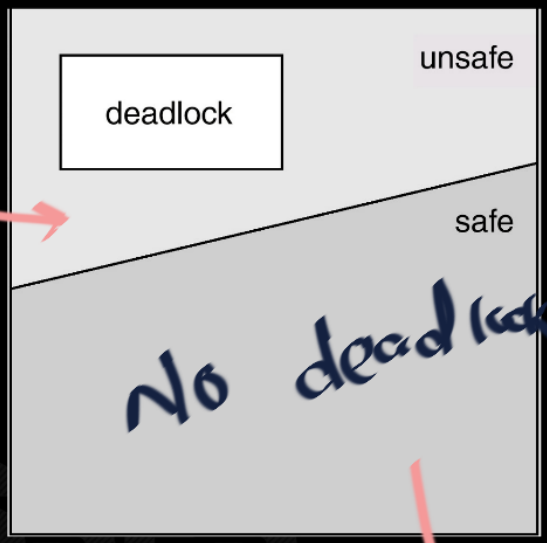
A. If P_2 is the first process to request for some resources then P_2 always have to wait. → NO

B. P_1's requests will always be satisfied without even checking if resulting state (after processing P_1's requests) is in safe state or not. ⇒ YES

C. Processes requests will always be satisfied in the same order as safe state sequence and we can not satisfy any out of order process request. ⇒ false

D. We can never satisfy any request coming from P_n without satisfying request from P_1. ⇒ false

Basic Facts



the processes may ask very less demand in future

- If a system is in safe state ⇒ no deadlocks.
- If a system is in unsafe state ⇒ possibility of deadlock.
- Avoidance ⇒ ensure that a system will never enter an unsafe state.

whatever the processes ask we can satisfy in some order.

Question

Consider the following snapshot of a system:

	C (Current allocation matrix)					R (Request matrix)					A (Available resource vector)			
	A	B	C	D		A	B	C	D		A	B	C	D
P_1	1	0	0	0		3	2	2	1		0	3	1	4
P_2	7	1	2	1		7	1	3	1					
P_3	2	1	1	0		3	1	4	0					
P_4	0	0	2	0		4	2	2	1					

Answer the following questions using Banker's algorithm:

a. Determine if the system is currently in a safe state using Banker's algorithm.

b. If a request from process P1 arrives for (0, 0, 1, 0), can the requested be granted immediately?

https://www2.cs.sfu.ca/CourseCentral/300/zonghuag/SampleCMPT300MidtermAnswers.pdf

GATE CSE 2007 | Question: 57

23

A single processor system has three resource types X, Y and Z, which are shared by three processes. There are 5 units of each resource type. Consider the following scenario, where the column **alloc** denotes the number of units of each resource type allocated to each process, and the column **request** denotes the number of units of each resource type requested by a process in order to complete execution. Which of these processes will finish **LAST**?

need

	alloc			request		
	X	Y	Z	X	Y	Z
P0	1	2	1	1	0	3
P1	2	0	1	0	1	2
P2	2	2	1	1	2	0

Available =

A. $P0$
B. $P1$
C. $P2$
D. None of the above, since the system is in a deadlock

gatecse-2007 operating-system resource-allocation normal

GATE CSE 2007 | Question: 57

23

A single processor system has three resource types X, Y and Z, which are shared by three processes. There are 5 units of each resource type. Consider the following scenario, where the column **alloc** denotes the number of units of each resource type allocated to each process, and the column **request** denotes the number of units of each resource type requested by a process in order to complete execution. Which of these processes will finish **LAST**?

	alloc			request		
	X	Y	Z	X	Y	Z
P0	1	2	1	1	0	3
P1	2	0	1	0	1	2
P2	2	2	1	1	2	0

A. $P0$
B. $P1$
C. $P2$
D. None of the above, since the system is in a deadlock

gatecse-2007 operating-system resource-allocation normal

need

Available = 0 1 2

P1 2 0 1

2 1 3

P0 1 2 1

3 3 4

27

Best answer

The answer is (**C**).

Available Resources

X	Y	Z
0	1	2

Now, $P1$ will execute first, As it meets the needs. After completion, The available resources are updated.

Updated Available Resources

X	Y	Z
2	1	3

Now $P0$ will complete the execution, as it meets the needs.

After completion of $P0$ the table is updated and then $P2$ completes the execution.

Thus $P2$ completes the execution in the last.

GATE CSE 2014 Set 1 | Question: 31

41

An operating system uses the *Banker's algorithm* for deadlock avoidance when managing the allocation of three resource types X, Y, and Z to three processes $P0$, $P1$, and $P2$. The table given below presents the current system state. Here, the *Allocation matrix* shows the current number of resources of each type allocated to each process and the *Max matrix* shows the maximum number of resources of each type required by each process during its execution.

	Allocation			Max		
	X	Y	Z	X	Y	Z
P0	0	0	1	8	4	3
P1	3	2	0	6	2	0
P2	2	1	1	3	3	3

There are 3 units of type X, 2 units of type Y and 2 units of type Z still available. The system is currently in a **safe** state. Consider the following independent requests for additional resources in the current state:

REQ1: $P0$ requests 0 units of X, 0 units of Y and 2 units of Z

REQ2: $P1$ requests 2 units of X, 0 units of Y and 0 units of Z

Which one of the following is **TRUE**?

A. Only REQ1 can be permitted.
B. Only REQ2 can be permitted.
C. Both REQ1 and REQ2 can be permitted.
D. Neither REQ1 nor REQ2 can be permitted.

https://gateoverflow.in/1800/gate-cse-2014-set-1-question-31

39

Best answer

Option (B)

Request 1 if permitted does not lead to a safe state.

After allowing Req 1,

	Allocated			Max			Requirement		
P0	0	0	3	8	4	3	8	4	0
P1	3	2	0	6	2	0	3	0	0
P2	2	1	1	3	3	3	1	2	2

$\text{Available} : X = 3, Y = 2, Z = 0$

Now we can satisfy $P1's$ requirement completely. So Available becomes :
$X = 6, Y = 4, Z = 0.$

Since, Z is not available now, neither $P0's$ nor $P2's$ requirement can be satisfied. So. it is an unsafe state.

GATE CSE 2008 | Question: 65

44

Which of the following is NOT true of deadlock prevention and deadlock avoidance schemes?

A. In deadlock prevention, the request for resources is always granted if the resulting state is safe

B. In deadlock avoidance, the request for resources is always granted if the resulting state is safe

C. Deadlock avoidance is less restrictive than deadlock prevention

D. Deadlock avoidance requires knowledge of resource requirements *apriori*..

https://gateoverflow.in/488/gate-cse-2008-question-65

GATE CSE 2008 | Question: 65

44

Which of the following is NOT true of deadlock prevention and deadlock avoidance schemes?

A. In deadlock prevention, the request for resources is always granted if the resulting state is safe

B. In deadlock avoidance, the request for resources is always granted if the resulting state is safe

C. Deadlock avoidance is less restrictive than deadlock prevention

D. Deadlock avoidance requires knowledge of resource requirements *apriori*..

⇒ false

⇒ true

true

⇒ true

https://gateoverflow.in/488/gate-cse-2008-question-65

65

Option **A** is answer.
The main difference between deadlock prevention and deadlock avoidance lies in the definition of deadlock itself.
We know Necessary_conditions for deadlock to happen. Means deadlock can be there only if these conditions are satisfied. But it does not mean that if these conditions are satisfied then deadlock will always be there.

The above definition is the main idea and fine line on that both strategies differ.

Deadlock Prevention says: Let's prevents one of the conditions.
Deadlock Avoidance says: Let me allow all conditions to hold simultaneously (all conditions holds simultaneously doesn't guarantee deadlock) but I will also check any chance of deadlock (i.e system in the safe state if I allocate requested resource.), It allows the four conditions but makes judicious decisions so that the deadlock point is not potentially reached.

Clearly, Deadlock prevention is more restrictive. It decreases throughput also. At the same time we can not use Deadlock avoidance practically as it demands to know uses of all resources in advance.

The analogy is like this:
(Now take an example of protection from viral through the water.)
Deadlock Prevention: Don't use water
Deadlock Avoidance: Use water but first filter out.

answered Dec 17, 2016

edit hide comment Unfollow Pip Box

Delete with Reason Wrong Useful

share this

Sachin Mittal 1

iv) If the Banker's algorithm will not approve a resource request, and the resource request is processed, then system necessarily will enter deadlock.

TRUE FALSE

Why?

FALSE. *Banker's algorithm only guarantees that a system is in a safe state. If a system is in an unsafe state, deadlock may or may not occur. The correct answer was worth 1 points and the justification was worth an additional 2 points.*

https://hkn.eecs.berkeley.edu/examfiles/cs162_fa16_mt2_sol.pdf

ii) (4 points) When using the Banker's algorithm for resource allocation, if the system is in an unsafe state, will it always eventually deadlock? Briefly (1-2 sentences) state why or why not.

No, because processes may not request their total possible resources, and may release some resources before acquiring others. Full credit was given for saying the Banker's algorithm does not allow a system to enter an unsafe state.

https://inst.eecs.berkeley.edu/~cs162/fa19/static/exams/sp13-mt1-solutions.pdf

Avoidance – Tradeoff

- Allowing only safe states is more flexible than Prevention
- But rejecting ***all*** unsafe states reduces efficiency
  - System could enter unsafe state and then return to safety
- Hmm...

Deadlock avoidance is less restrictive than deadlock prevention

http://www.cs.cmu.edu/~410-f06/lectures/L14_Deadlock.pdf

GATE CSE 2022 | Question: 16

9

Which of the following statements is/are TRUE with respect to deadlocks?

A. Circular wait is a necessary condition for the formation of deadlock.
B. In a system where each resource has more than one instance, a cycle in its wait-for graph indicates the presence of a deadlock.
C. If the current allocation of resources to processes leads the system to unsafe state, then deadlock will necessarily occur.
D. In the resource-allocation graph of a system, if every edge is an assignment edge, then the system is not in deadlock state.

gatecse-2022 | operating-system | resource-allocation | multiple-selects | 1-mark

https://gateoverflow.in/371920/gate-cse-2022-question-16

GATE CSE 2022 | Question: 16

9

Which of the following statements is/are TRUE with respect to deadlocks?

A. Circular wait is a necessary condition for the formation of deadlock. ✓
B. In a system where each resource has more than one instance, a cycle in its wait-for graph indicates the presence of a deadlock. ✗
C. If the current allocation of resources to processes leads the system to unsafe state, then deadlock will necessarily occur. ✗
D. In the resource-allocation graph of a system, if every edge is an assignment edge, then the system is not in deadlock state. ✓

(No hold & wait)

gatecse-2022 | operating-system | resource-allocation | multiple-selects | 1-mark

https://gateoverflow.in/371920/gate-cse-2022-question-16

6

Best answer

Option A, D

A. Circular Wait is one of the four necessary conditions for deadlock to happen. **True**

B. Not necessarily, if every resource had only a single instance then, a cycle would've been necessary and sufficient for a deadlock to occur. A Cycle in a multi-instance resource is necessary but isn't sufficient and in this case, each resource is made up of more than one instance, a simple contradiction. **False**

C. An Unsafe state is where no method of allocation of resources can prevent deadlock from happening. Whereas in a safe state, there exists a method of allocation in which all processes are complete. **False**

D. Resource Allocation graph accommodates future resource requirements in the form of request edges. If there are no request edges, then resource requirements for all the processes are satisfied. **True**

Deadlock Detection and Recovery

Strategies for dealing with Deadlocks

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

more flexible

more flexible

GO CLASSES

Deadlock Detection and Recovery

i will alloc EVERY request whatsoever

and will check for deadlock periodically,

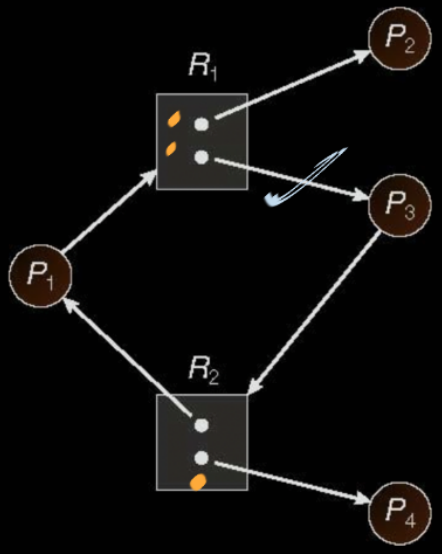
if it is deadlock then will recover from it

↳ Restart, kill the prog
preempt the resources

GO CLASSES

Detection

Available = [R_1 R_2]



allocated

	R_1	R_2
P_1		
P_2		
P_3		
P_4		

Request

	R_1	R_2
P_1	1	0
P_2		

Avoidance

Available

allocated

need

GO CLASSES

Available = 0 1 2

	alloc		
	X	Y	Z
P0	1	2	1
P1	2	0	1
P2	2	2	1

instead of need

	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	

Q: is this system in deadlock or not?

instead of need

	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	

10 : 00 : 00 : 00

Q: is this system in deadlock or not?

Available 0 0 0

0 1 0

0 1 0

3 0 3

3 1 3

Recovery from Deadlock

- Recovery through preemption
  - take a resource from some other process
  - depends on nature of the resource
- Recovery through rollback
  - checkpoint a process periodically
  - use this saved state
  - restart the process if it is found deadlocked

Recovery from Deadlock

- Recovery through killing processes
  - crudest but simplest way to break a deadlock
  - kill one of the processes in the deadlock cycle
  - the other processes get its resources
  - choose process that can be rerun from the beginning