



RAMA UNIVERSITY

www.ramauniversity.ac.in

FACULTY OF ENGINEERING & TECHNOLOGY

BCA-307 Operating System

Lecturer-16

Manisha Verma

Assistant Professor

Computer Science & Engineering

Deadlocks

- **Deadlock Detection**
- **Recovery from Deadlock**
- **Combined Approach to Deadlock Handling**



safe state algorithm.

Let *Work* and *Finish* be vectors of length m and n , respectively. Initialize:

$Work = Available$

$Finish[i] = false$ for $i = 0, 1, \dots, n-1$

2. Find an i such that both:

(a) $Finish[i] = false$

(b) $Need_i \leq Work$

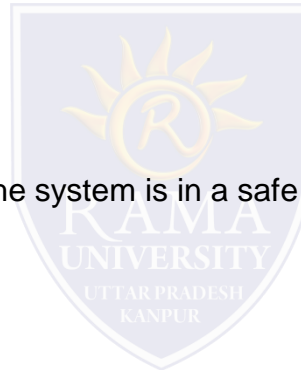
If no such i exists, go to step 4

3. $Work = Work + Allocation_i$

$Finish[i] = true$

go to step 2

4. If $Finish[i] == true$ for all i , then the system is in a safe state algorithm.



Resource-Request Algorithm for Process P_i

$Request_i$ = request vector for process P_i . If $Request_i[j] = k$ then process P_i wants k instances of resource type R_j

1. If $Request_i \leq Need_i$ go to step 2. Otherwise, raise error condition, since process has exceeded its maximum claim

2. If $Request_i \leq Available$, go to step

3. Otherwise P_i must wait, since resources are not available

4. Pretend to allocate requested resources to P_i by modifying the state as follows:

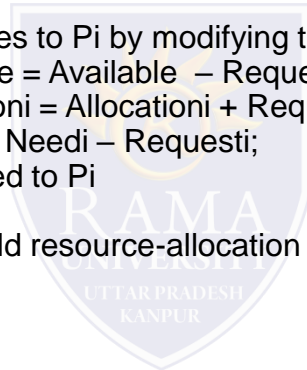
$Available = Available - Request_i$;

$Allocation_i = Allocation_i + Request_i$;

$Need_i = Need_i - Request_i$;

If safe \Rightarrow the resources are allocated to P_i

If unsafe \Rightarrow P_i must wait, and the old resource-allocation state is restored



Example of Banker's Algorithm

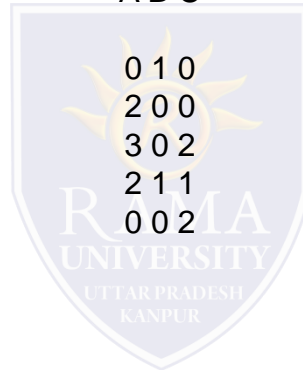
5 processes P0 through P4;

3 resource types:

A (10 instances), B (5 instances), and C (7 instances)

Snapshot at time T0:

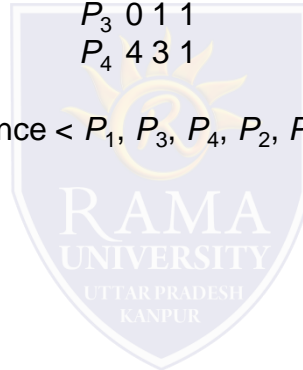
	Available	Allocation			Max		
		A	B	C	A	B	C
C							
P0		0	1	0	7	5	3
P1		2	0	0	3	2	2
P2		3	0	2	9	0	2
P3		2	1	1	2	2	2
P4		0	0	2	4	3	3



The content of the matrix **Need** is defined to be **Max – Allocation**

	<u>Need</u>		
	A	B	C
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

The system is in a safe state since the sequence $\langle P_1, P_3, P_4, P_2, P_0 \rangle$ satisfies safety criteria



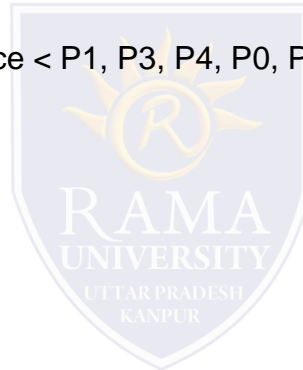
Check that Request \leq Available (that is, $(1,0,2) \leq (3,3,2) \Rightarrow$ true

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

Executing safety algorithm shows that sequence $\langle P1, P3, P4, P0, P2 \rangle$ satisfies safety requirement

Can request for $(3,3,0)$ by P4 be granted?

Can request for $(0,2,0)$ by P0 be granted?



Deadlock Detection

Allow system to enter deadlock state

Detection algorithm

Recovery scheme

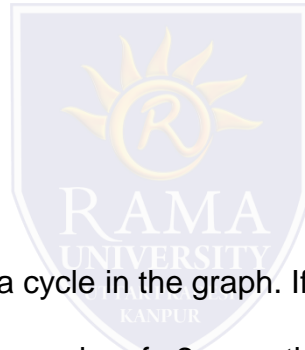
Maintain wait-for graph

Nodes are processes

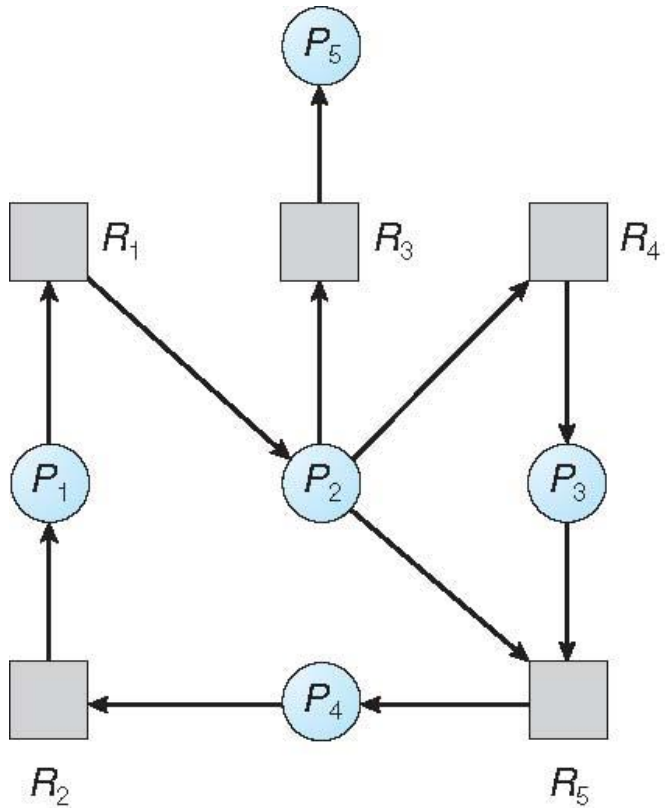
$P_i \rightarrow P_j$ if P_i is waiting for P_j

Periodically invoke an algorithm that searches for a cycle in the graph. If there is a cycle, there exists a deadlock

An algorithm to detect a cycle in a graph requires an order of n^2 operations, where n is the number of vertices in the graph

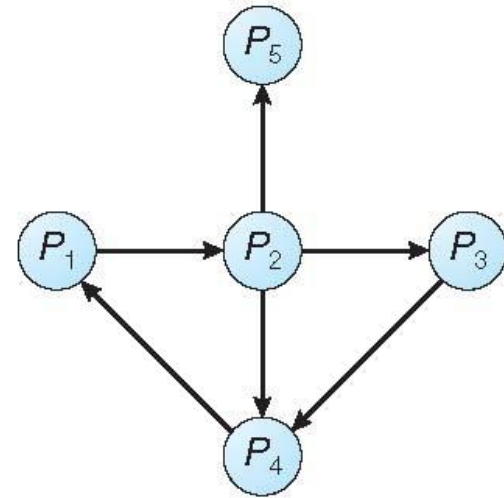


Resource-Allocation Graph and Wait-for Graph



(a)

Resource-Allocation Graph



(b)

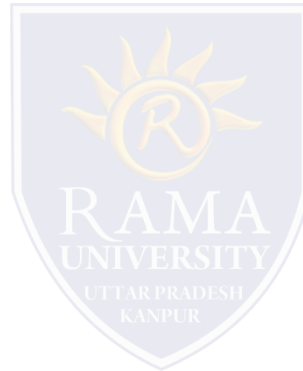
wait-for graph

Resource Type

Available: A vector of length m indicates the number of available resources of each type

Allocation: An $n \times m$ matrix defines the number of resources of each type currently allocated to each process

Request: An $n \times m$ matrix indicates the current request of each process. If $\text{Request}[i][j] = k$, then process P_i is requesting k more instances of resource type R_j .



Detection Algorithm

1. Let Work and Finish be vectors of length m and n, respectively Initialize:

(a) Work = Available

(b) For $i = 1, 2, \dots, n$, if $Allocation_i \neq 0$, then
Finish[i] = false; otherwise, Finish[i] = true

2. Find an index i such that both:

(a) Finish[i] == false

(b) Request_i ≤ Work

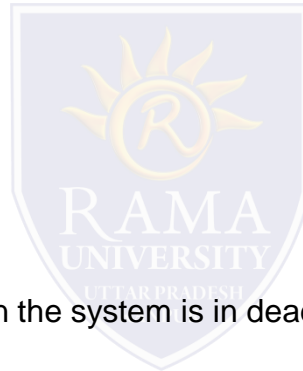
If no such i exists, go to step 4

3. Work = Work + Allocation_i

Finish[i] = true

go to step 2

4. If Finish[i] == false, for some i, $1 \leq i \leq n$, then the system is in deadlock state. Moreover, if Finish[i] == false, then P_i is deadlocked



Example of Detection Algorithm

Example of Detection Algorithm

Five processes P_0 through P_4 ; three resource types
A (7 instances), B (2 instances), and C (6 instances)

Snapshot at time T_0 :

		<u>Allocation</u>	<u>Request</u>	<u>Available</u>
		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

Sequence $\langle P_0, P_2, P_3, P_1, P_4 \rangle$ will result in $Finish[i] = true$ for all i

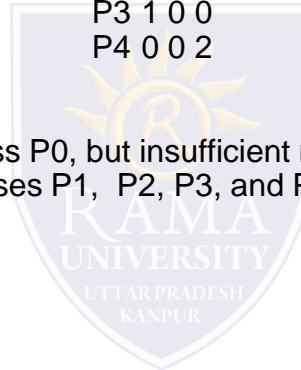
Example (Cont.)

P2 requests an additional instance of type C

	Request		
	A	B	C
P0	0	0	0
P1	2	0	2
P2	0	0	1
P3	1	0	0
P4	0	0	2

State of system?

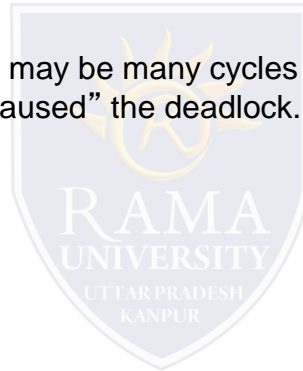
Can reclaim resources held by process P0, but insufficient resources to fulfill other processes; requests
Deadlock exists, consisting of processes P1, P2, P3, and P4



Detection-Algorithm Usage

- When, and how often, to invoke depends on:
 - How often a deadlock is likely to occur?
 - How many processes will need to be rolled back?
 - one for each disjoint cycle

If detection algorithm is invoked arbitrarily, there may be many cycles in the resource graph and so we would not be able to tell which of the many deadlocked processes “caused” the deadlock.



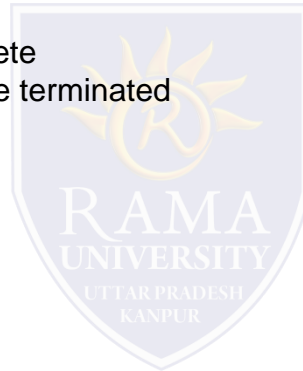
Recovery from Deadlock: Process Termination

Abort all deadlocked processes

Abort one process at a time until the deadlock cycle is eliminated

In which order should we choose to abort?

1. Priority of the process
2. How long process has computed, and how much longer to completion
3. Resources the process has used
4. Resources process needs to complete
5. How many processes will need to be terminated
6. Is process interactive or batch?

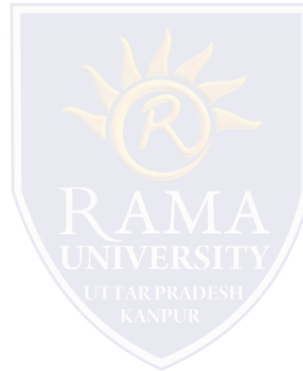


Recovery from Deadlock

Selecting a victim – minimize cost

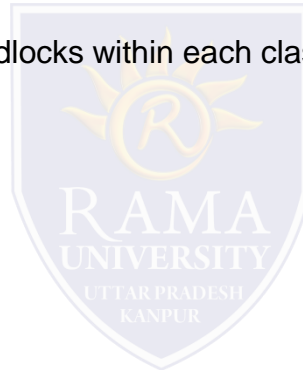
Rollback – return to some safe state, restart process for that state

Starvation – same process may always be picked as victim, include number of rollback in cost factor



Combined Approach to Deadlock Handling

- Combine the three basic approaches
 - prevention
 - avoidance
 - detection
- allowing the use of the optimal approach for each of resources in the system.
- Partition resources into hierarchically ordered classes.
- Use most appropriate technique for handling deadlocks within each class.



Which one of the following is a visual (mathematical) way to determine the deadlock occurrence?

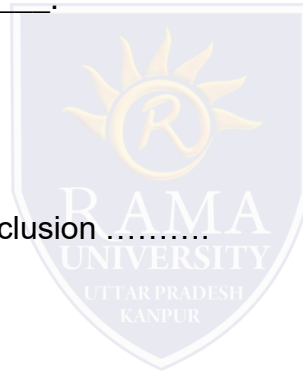
- A. resource allocation graph
- B. starvation graph
- C. inversion graph
- D. none of the mentioned

The request and release of resources are _____.

- A. command line statements
- B. interrupts
- C. system calls
- D. special programs

For non sharable resources like a printer, mutual exclusion

- A. must exist
- B. must not exist
- C. may exist
- D. None of these



For sharable resources, mutual exclusion :

- A. is required
- B. is not required
- C. None of these

Deadlock handling approach.....

- A.Prevention
- B.Avoidance
- C.Detection
- D.All of these

