Complex

Experiential Learning (site visits)

Forum Theater

Jigsaw Discussion

Inquiry Learning

Role Playing

Active Review Sessions (Games or Simulations)

Interactive Lecture

Hands-on Technology

Case Studies

Brainstorming

Groups Evaluations

Peer Review

Informal Groups

Triad Groups

Large Group Discussion

Think-Pair-Share

Writing (Minute Paper)

Self-assessment

Pause for reflection

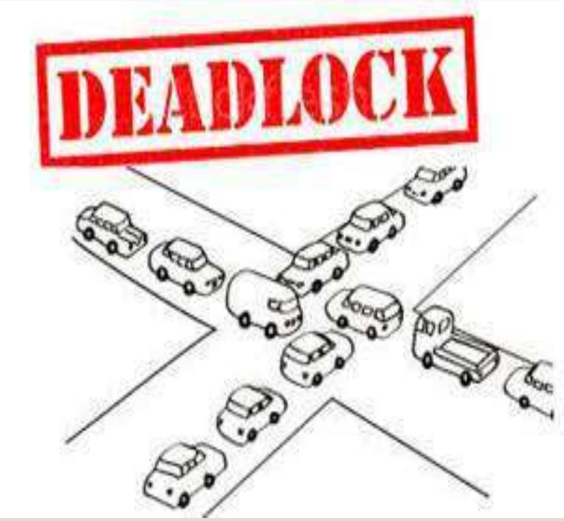
Simple

CO3

Session - 6

COURSE NAME: OPERATING SYSTEMS

COURSE CODE: 23CS2104R/A



Deadlock: Detection and Recovery

AIM OF THE SESSION

To familiarize students with the basic concept of Deadlocks.

INSTRUCTIONAL OBJECTIVES

This Session is designed to:

1. Demonstrate what is meant by Banker's Algorithm.
2. Demonstrate problems on Banker's Algorithm.
3. Describe the Methods for Deadlock Detection and Recovery .

LEARNING OUTCOMES

At the end of this session, you should be able to:

1. Defines what Deadlock is.
2. Defines Wait-for Graph .
3. Summarize the Concept of Deadlock.

EXAMPLE OF BANKER'S ALGORITHM FOR AVOIDANCE

- 5 processes P_0 through P_4;

  3 resource types:

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

- Snapshot at time T_0:

	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	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

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

	Need
	$A\ B\ C$
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 $< P_1, P_3, P_4, P_0, P_2>$ satisfies safety criteria m=3, n=5 — Step 1 of Safety Algo

Work = Available

Work =			
	3	3	2

	0	1	2	3	4
Finish =	false	false	false	false	false

For i = 0 — Step 2

$Need_0$ = 7, 4, 3

7,4,3 3,3,2

Finish [0] is false and $Need_0$ > Work

So P_0 must wait — But Need ≤ Work

For i = 1 — Step 2

$Need_1$ = 1, 2, 2

1,2,2 3,3,2

Finish [1] is false and $Need_1$ < Work

So P_1 must be kept in safe sequence

Step 3

3, 3, 2 2, 0, 0

Work = Work + $Allocation_1$

	A	B	C
Work =	5	3	2

	0	1	2	3	4
Finish =	false	true	false	false	false

For i = 2 — Step 2

$Need_2$ = 6 , 0, 0

6, 0, 0 5,3, 2

Finish [2] is false and $Need_2$ > Work

So P_2 must wait

For i=3 — Step 2

$Need_3$ = 0, 1, 1

0, 1, 1 5, 3, 2

Finish [3] = false and $Need_3$ < Work

So P_3 must be kept in safe sequence

Step 3

5, 3, 2 2, 1, 1

Work = Work + $Allocation_3$

	A	B	C
Work =	7	4	3

	0	1	2	3	4
Finish =	false	true	false	true	false

For i = 4 — Step 2

$Need_4$= 4, 3, 1

4, 3, 1 7, 4, 3

Finish [4] = false and $Need_4$ < Work

So P_4 must be kept in safe sequence

Step 3

7, 4, 3 0, 0, 2

Work = Work + $Allocation_4$

	A	B	C
Work =	7	4	5

	0	1	2	3	4
Finish =	false	true	false	true	true

For i = 0 — Step 2

$Need_0$ = 7, 4, 3

7, 4, 3 7, 4, 5

Finish [0] is false and Need < Work

So P_0 must be kept in safe sequence

Step 3

7, 4, 5 0, 1 , 0

Work = Work + $Allocation_0$

	A	B	C
Work =	7	5	5

	0	1	2	3	4
Finish =	true	true	false	true	true

For i = 2 — Step 2

$Need_2$ = 6 , 0, 0

6, 0, 0 7, 5, 5

Finish [2] is false and $Need_2$ < Work

So P_2 must be kept in safe sequence

Step 3

7, 5, 5 3, 0, 2

Work = Work + $Allocation_2$

	A	B	C
Work =	10	5	7

	0	1	2	3	4
Finish =	true	true	true	true	true

Step 4

Finish [i] = true for $0 \le i \le n$

Hence the system is in Safe state

The safe sequence is P_1, P_3 , P_4 , P_0, P_2

CONSIDER P_1 REQUEST (1,0,2)

- 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*
P_0	0 1 0	7 4 3	2 3 0
P_1	3 0 2	0 2 0	
P_2	3 0 2	6 0 0	
P_3	2 1 1	0 1 1	
P_4	0 0 2	4 3 1	

- Executing safety algorithm shows that sequence $< P_1, P_3, P_4, P_0, P_2>$ satisfies safety requirement
- Can request for (3,3,0) by P_4 be granted?
- Can request for (0,2,0) by P_0 be granted?

$m=3, n=5$ Step 1 of Safety Algo

Work = Available

Work =	2	3	0

	0	1	2	3	4
Finish =	false	false	false	false	false

For $i = 0$ Step 2

$Need_0 = 7, 4, 3$

Finish [0] is false and $\overset{7,4,3}{Need_0} > \overset{2,3,0}{Work}$

So P_0 must wait

But Need ≤ Work

For $i = 1$ Step 2

$Need_1 = 0, 2, 0$

Finish [1] is false and $\overset{0,2,0}{Need_1} < \overset{2,3,0}{Work}$

So P_1 must be kept in safe sequence

Step 3

$Work = \overset{2,3,0}{Work} + \overset{3,0,2}{Allocation_1}$

	A	B	C
Work =	5	3	2

	0	1	2	3	4
Finish =	false	true	false	false	false

For $i = 2$ Step 2

$Need_2 = 6, 0, 0$

Finish [2] is false and $\overset{6,0,0}{Need_2} > \overset{5,3,2}{Work}$

So P_2 must wait

For $i=3$ Step 2

$Need_3 = 0, 1, 1$

Finish [3] = false and $\overset{0,1,1}{Need_3} < \overset{5,3,2}{Work}$

So P_3 must be kept in safe sequence

Step 3

$Work = \overset{5,3,2}{Work} + \overset{2,1,1}{Allocation_3}$

	A	B	C
Work =	7	4	3

	0	1	2	3	4
Finish =	false	true	false	true	false

For $i = 4$ Step 2

$Need_4 = 4, 3, 1$

Finish [4] = false and $\overset{4,3,1}{Need_4} < \overset{7,4,3}{Work}$

So P_4 must be kept in safe sequence

Step 3

$Work = \overset{7,4,3}{Work} + \overset{0,0,2}{Allocation_4}$

	A	B	C
Work =	7	4	5

	0	1	2	3	4
Finish =	false	true	false	true	true

For $i = 0$ Step 2

$Need_0 = 7, 4, 3$

Finish [0] is false and $\overset{7,4,3}{Need} < \overset{7,4,5}{Work}$

So P_0 must be kept in safe sequence

Step

$Work = \overset{7,4,5}{Work} + \overset{0,1,0}{Allocation_0}$

	A	B	C
Work =	7	5	5

	0	1	2	3	4
Finish =	true	true	false	true	true

For $i = 2$ Step

$Need_2 = 6, 0, 0$

Finish [2] is false and $\overset{6,0,0}{Need_2} < \overset{7,5,5}{Work}$

So P_2 must be kept in safe sequence

Step

$Work = \overset{7,5,5}{Work} + \overset{3,0,2}{Allocation_2}$

	A	B	C
Work =	10	5	7

	0	1	2	3	4
Finish =	true	true	true	true	true

Step

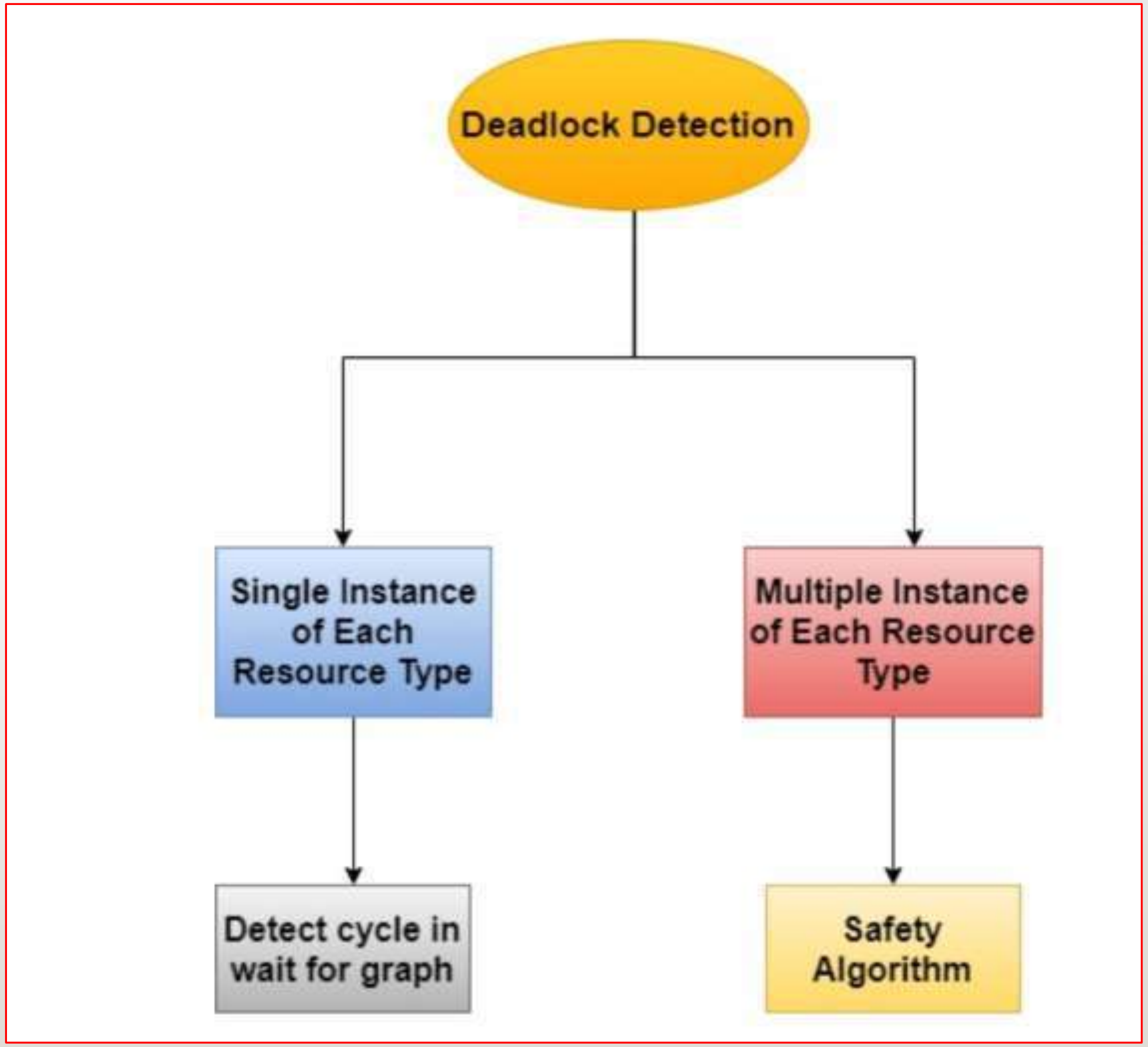
Finish [i] = true for $0 \leq i \leq n$

Hence the system is in Safe state

The safe sequence is P_1, P_3, P_4, P_0, P

DEADLOCK DETECTION

- Allow system to enter deadlock state
- Detection algorithm
- Recovery scheme



SINGLE INSTANCE OF EACH RESOURCE TYPE

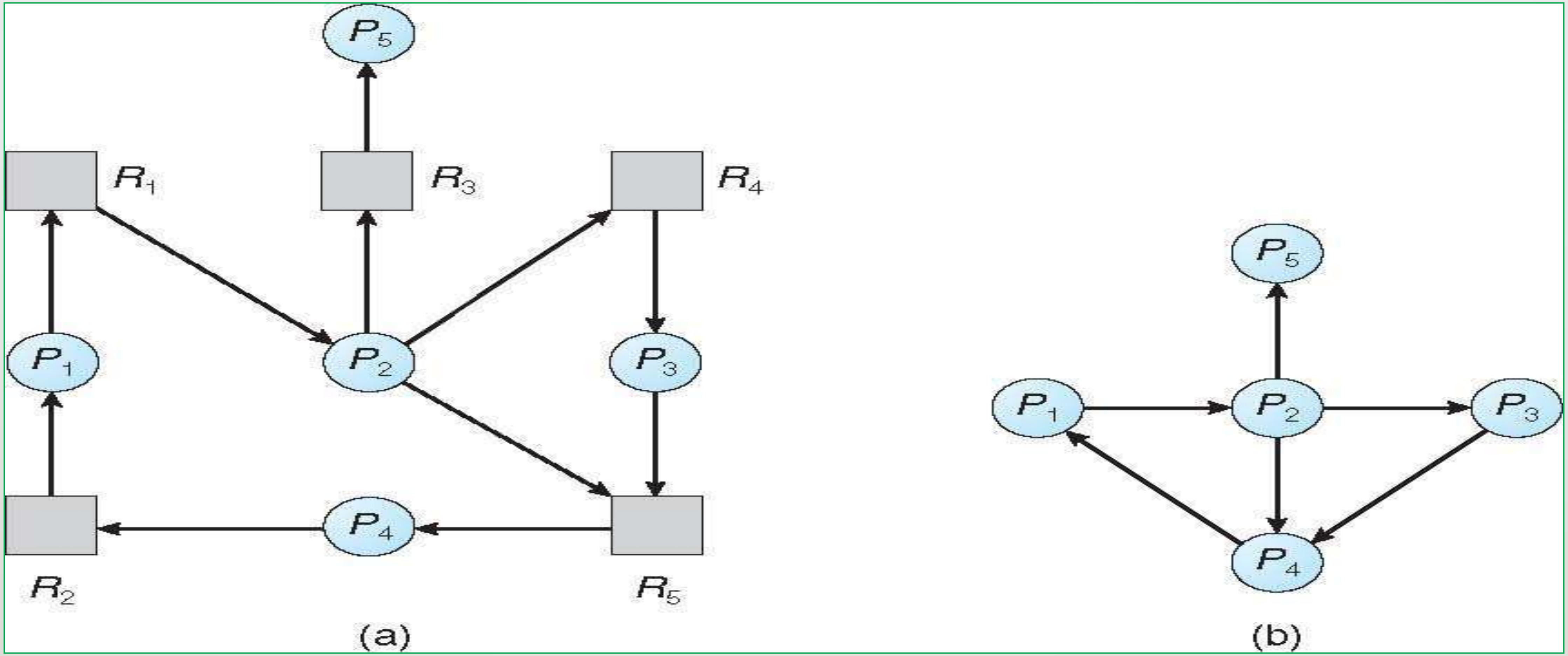
Maintain **wait-for** graph

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



Resource-Allocation Graph

Corresponding wait-for graph

SEVERAL INSTANCES OF A RESOURCE TYPE

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

Allocation: An ***n* x *m*** matrix defines the number of resources of each type currently allocated to each process

Request: An ***n* x *m*** matrix indicates the current request of each process. If ***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, …, *n***, if $\boldsymbol{Allocation_i} \neq \mathbf{0}$, then ***Finish*[i] = *false***; otherwise, ***Finish*[i] = *true***

2. Find an index ***i*** such that both:

   (a) ***Finish*[*i*] == *false***

   (b) $\boldsymbol{Request_i} \leq \boldsymbol{Work}$

   If no such ***i*** exists, go to step 4

3. $\boldsymbol{Work = Work + Allocation_i}$

   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 i deadlocked.

Algorithm requires an order of O(m x n^2) operations to detect whether the system is in deadlocked state

EXAMPLE OF DETECTION ALGORITHM

- Five processes P_0 through P_4;
  Three Resources A,B,C : A has 7 , *B has* 2 , and *C* has 6 instances
- Snapshot at time T_0:

	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	

- Sequence $<P_0, P_2, P_3, P_1, P_4>$ OR <P0,P2,P3,P4,P1> will result in ***Finish[i] = true*** for all ***i.***

- P_2 requests an additional instance of type C

	Request
	A B C
P_0	0 0 0
P_1	2 0 2
P_2	0 0 1
P_3	1 0 0
P_4	0 0 2

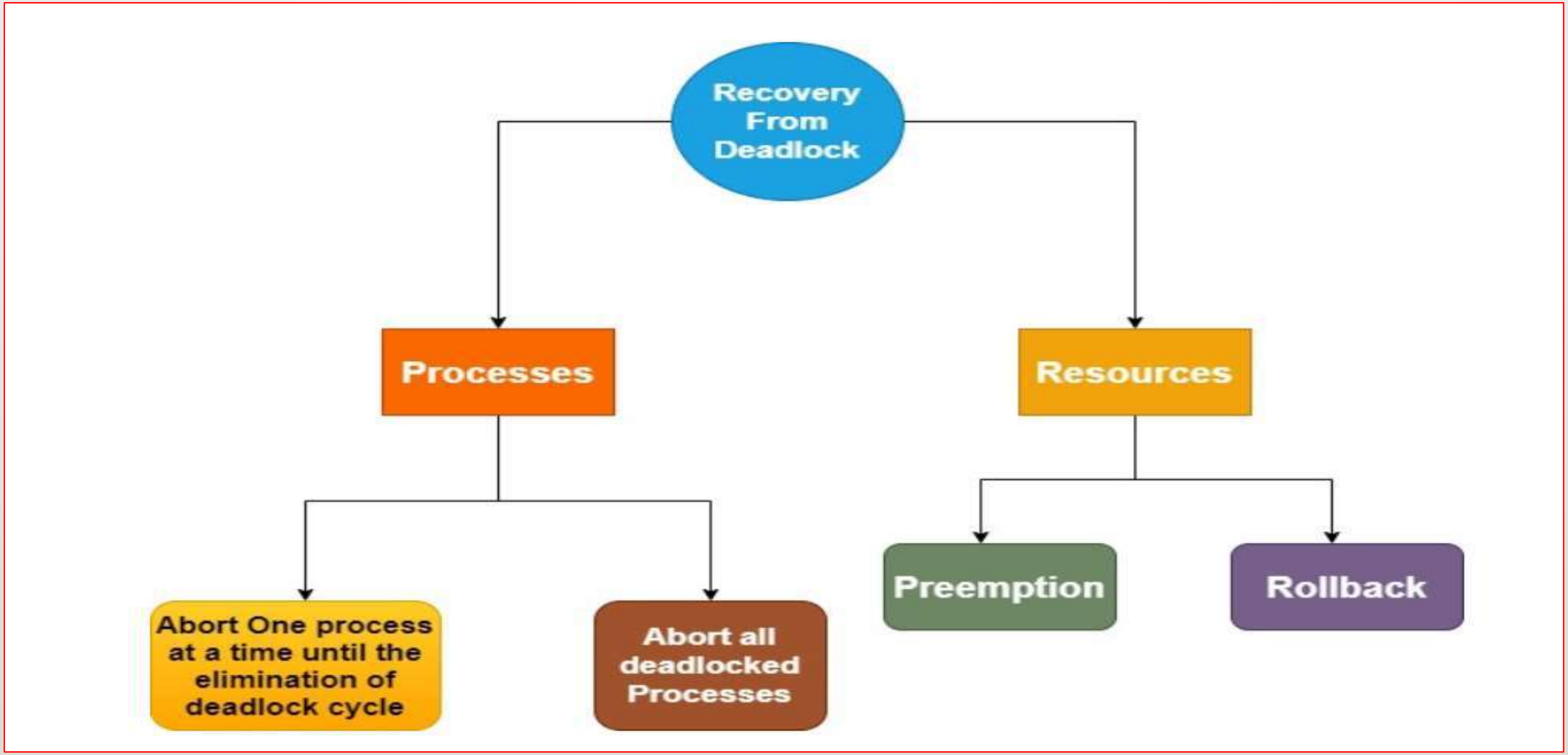
- State of the system?
  - Can reclaim resources held by process P_0, but insufficient resources to fulfill other processes; requests
  - Deadlock exists, consisting of processes P_1, P_2, P_3, and P_4

RECOVERY FROM DEADLOCK

When a detection algorithm determines that a deadlock exists then there are several available alternatives. There one possibility and that is to inform the operator about the deadlock and let him deal with this problem manually.

- Another possibility is to let the system recover from the deadlock automatically. These are two options that are mainly used to break the deadlock.

RECOVERY FROM DEADLOCK



RECOVERY FROM DEADLOCK

Approaches To Breaking a Deadlock

1.Process Termination

- To eliminate the deadlock, we can simply kill one or more processes. For this, we use two methods:

I. **Abort all the Deadlocked Processes:** Aborting all the processes will certainly break the deadlock but at a great expense. The deadlocked processes may have been computed for a long time,.

II. **Abort one process at a time until the deadlock is eliminated:** Abort one deadlocked process at a time, until the deadlock cycle is eliminated from the system.

RECOVERY FROM DEADLOCK: RESOURCE PREEMPTION

Selecting a victim – minimize cost

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

Starvation – The same process may always be picked as a victim, including the number of rollbacks in the cost factor.

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 the process interactive or batch?

THANK YOU

Team – Operating System