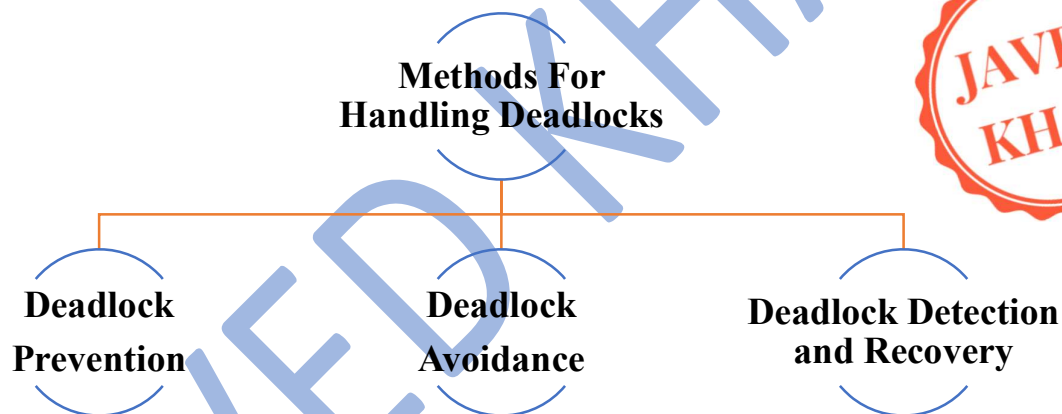




In a multiprogramming system, several processes may compete for a finite number of resources which may result into occurring of a situation in which two or more processes may attempt to access a resource which is already locked by another process and therefore cannot be shared.

This situation in which two or more processes are waiting indefinitely because the resources they have requested for are being held by one another is termed as **Deadlock**.

➤ To handle this situation following **three** methods are used:



One solution is to avoid deadlock by only granting resources if granting them cannot result in a deadlock situation later. However, this works only if the system knows what requests for resources a process will be making in the future, and this is an unrealistic assumption.

The text describes the **banker's algorithm** but then points out that it is essentially impossible to implement because of this assumption.

The **banker's algorithm** is a **resource allocation** and **deadlock avoidance algorithm** developed by **Edsger Wybe Dijkstra** that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes a **safe state check** to test for possible activities, before deciding whether allocation should be allowed to continue. It helps the operating system to successfully share the resources between all the processes.


❖ **Banker's algorithm** is named so because it is used in banking system to check whether loan can be sanctioned to a person or not.

When a new process is created in a computer system, the process must provide all types of information to the operating system like upcoming processes, requests for their resources, counting them, and delays. Based on these criteria, the operating system decides which process sequence should be executed or waited so that no deadlock occurs in a system.

✓ Therefore, it is also known as **deadlock avoidance** or **deadlock detection algorithm** in the operating system.

Example1:

Finding out using given set of five processes P_0, P_1, P_2, P_3, P_4 and **three** resources of type R_1, R_2, R_3 whether the deadlock occur or not. Resource type R_1 has **10** instances, R_2 has **5** instances and R_3 has **7** instances.



Process	Allocation			Maximum Need		
	R_1	R_2	R_3	R_1	R_2	R_3
P_0	0	1	0	7	5	3
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

Solution:

Given that, instances of resource $R_1 = 10, R_2 = 5, R_3 = 7$.

Following calculations are observed:

Process	Allocation			Maximum Need			Current Availability			Remaining Need		
	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3
P_0	0	1	0	7	5	3	3	3	2	7	4	3
P_1	2	0	0	3	2	2	5	3	2	1	2	2
P_2	3	0	2	9	0	2	7	4	3	6	0	0
P_3	2	1	1	2	2	2	7	4	5	0	1	1
P_4	0	0	2	4	3	3	7	5	5	4	3	1
	7	2	5				10	5	7			

Here,

Remaining Need = Maximum Need – Current Availability

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

Here, the processes have been executed successfully.

Hence, **NO DEADLOCK** occurs!



Example2:

Finding out using given set of five processes P_0, P_1, P_2, P_3, P_4 and **three** resources of type R_1, R_2, R_3 whether the deadlock occur or not. Resource type R_1 has 7 instances, R_2 has 3 instances and R_3 has 7 instances.



Process	Allocation			Maximum Need		
	R_1	R_2	R_3	R_1	R_2	R_3
P_0	1	0	1	4	2	3
P_1	1	0	2	9	0	2
P_2	2	0	1	3	2	2
P_3	1	0	0	2	2	1
P_4	1	1	2	7	5	3

Solution:

Given that, instances of resource $R_1 = 7, R_2 = 3, R_3 = 7$.

Following calculations are observed:

Process	Allocation			Maximum Need			Current Availability			Remaining Need		
	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3
✓ P_0	1	0	1	4	2	3	1	2	1	3	2	2
P_1	1	0	2	9	0	2	3	2	2	8	0	0
✓ P_2	2	0	1	3	2	2	4	2	2	1	2	1
✓ P_3	1	0	0	2	2	1	5	2	9	1	2	1
P_4	1	1	2	7	5	3				6	4	1

6 1 6

Here,

$$\text{Remaining Need} = \text{Maximum Need} - \text{Current Availability}$$

Here, all the processes have not been executed successfully and the system is not in **safe state** as the number of resources available are insufficient in order to fulfil the requirement of processes P_1 and P_4 .

Hence, **DEADLOCK** occurs!

+Advantages+

1. It contains various resources that meet the requirements of each process.
2. Each process should provide information to the operating system for upcoming resource requests, the number of resources, and how long the resources will be held.
3. It helps the operating system manage and control process requests for each type of resource in the computer system.

-Disadvantages-

1. It requires a fixed number of processes, and no additional processes can be started in the system while executing the process.
2. The algorithm does no longer allows the processes to exchange its maximum needs while processing its tasks.
3. Each process has to know and state their maximum resource requirement in advance for the system.
4. The number of resource requests can be granted in a finite time, but the time limit for allocating the resources is one year.

Important Points To Be Noted –

- ❖ A **safe state** is one in which the system can allocate resources to each process up to the maximum in some order and still avoid deadlock.
- ❖ A system is in a **safe state** if there is a **safe sequence**.
- ❖ **Allocation** indicates the resources which are allocated to the processes.
- ❖ **Maximum Need** means the maximum requirement of resources to the processes.
- ❖ **Current Availability** is the resources (i.e., number of resources) which are currently available.
- ❖ **Banker's algorithm** requires that whenever a new process is created, it should specify the maximum number of instances of each resource type that it exactly needs.

