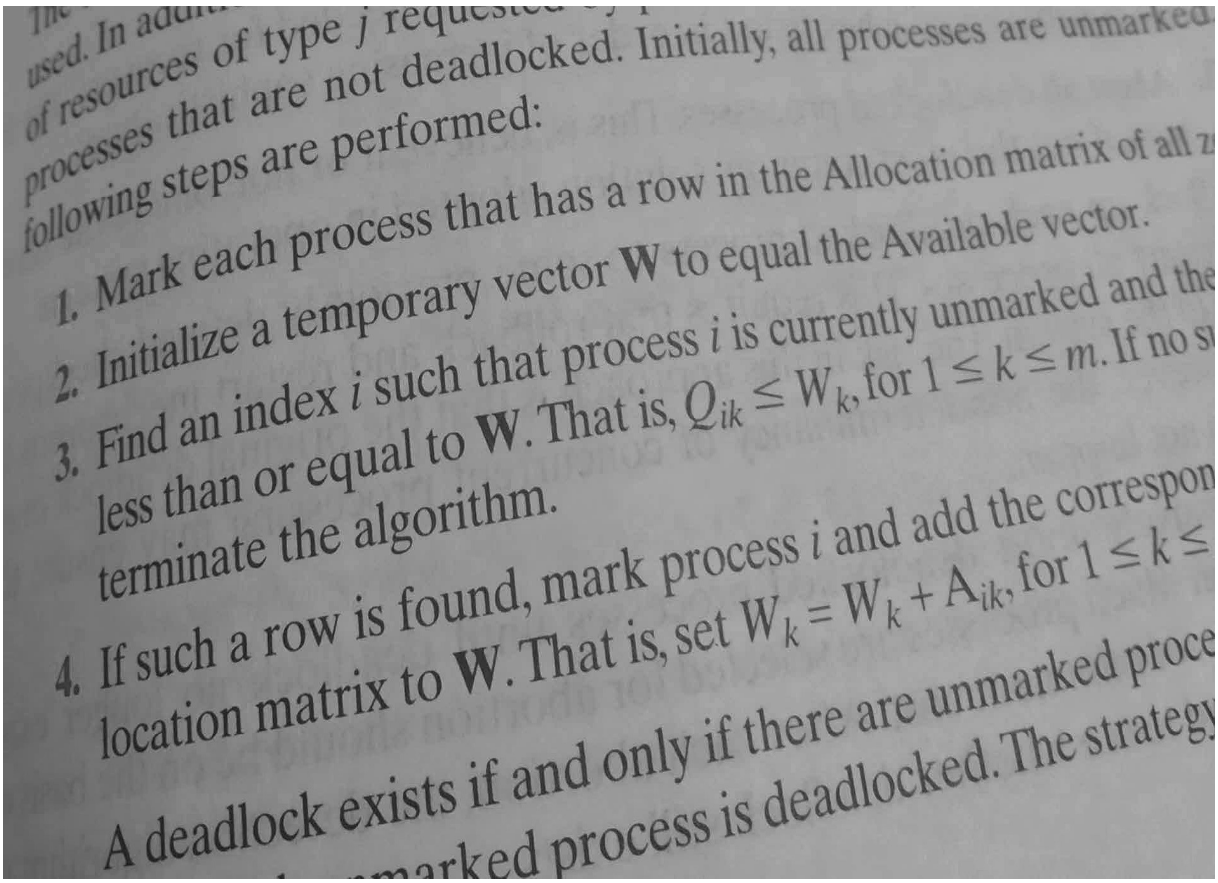
Deadlock Detection Algorithm



used. In addi...
of resources of type j request...
processes that are not deadlocked. Initially, all processes are unmarked.
following steps are performed:

1. Mark each process that has a row in the Allocation matrix of all z...
2. Initialize a temporary vector **W** to equal the Available vector.
3. Find an index i such that process i is currently unmarked and the... less than or equal to **W**. That is, $Q_{ik} \le W_k$, for $1 \le k \le m$. If no su... terminate the algorithm.
4. If such a row is found, mark process i and add the correspon... location matrix to **W**. That is, set $W_k = W_k + A_{ik}$, for $1 \le k \le$ ...

A deadlock exists if and only if there are unmarked proce...
...marked process is deadlocked. The strategy...

	R1	R2	R3	R4	R5
P1	0	1	0	0	1
P2	0	0	1	0	1
P3	0	0	0	0	1
P4	1	0	1	0	1

Request matrix Q

	R1	R2	R3	R4	R5
P1	1	0	1	1	0
P2	1	1	0	0	0
P3	0	0	0	1	0
P4	0	0	0	0	0

Allocation matrix A

R1	R2	R3	R4	R5
2	1	1	2	1

Resource vector

R1	R2	R3	R4	R5
0	0	0	0	1

Available vector

Figure 6.10 **Example for Deadlock Detection**

find a process whose resource requests can be satisfied with the available resources, and then assume that those resources are granted and that the process runs to completion and releases all of its resources. The algorithm then looks for another process to satisfy. Note that this algorithm does not guarantee to prevent deadlock; that will depend on the order in which future requests are granted. All that it does is determine if deadlock currently exists.

We can use Figure 6.10 to illustrate the deadlock detection algorithm. The algorithm proceeds as follows:

1. Mark P4, because P4 has no allocated resources.
2. Set **W** = (0 0 0 0 1).
3. The request of process P3 is less than or equal to **W**, so mark P3 and set **W** = **W** + (0 0 0 1 0) = (0 0 0 1 1).
4. No other unmarked process has a row in **Q** that is less than or equal to **W**. Therefore, terminate the algorithm.

The algorithm concludes with P1 and P2 unmarked, indicating that these processes are deadlocked.

Deadlock Detection

Req: Q

	R1	2	3	4	5
1	0	1	0	0	1
2	0	0	1	0	1
3	0	0	0	0	1
4	1	0	1	0	1

Req.

A

1	0	1	1	0	1
1	1	0	0	0	2
0	0	0	1	0	3 *
0	0	0	0	0	4 *
2	1	1	2	0	
2	1	1	2	1	

R: [2 | 1 | 1 | 2 | 1]

V: [0 | 0 | 0 | 0 | 1]

✓ all zero row in A mark (P_4 mark)

✓ Let W = current V = (00001)

(3) i find P_i in Q that is unmarked in A

and $\underline{\underline{P_i(Q) \le W}}$. ∴ P_3 marked

and set W = 00001 +

00010 → P_3' All

00011.

1,2 Remain Unmarked → . Deadlock

∴ Deadlocked on 1,2.

Recovery From Deadlock

- 3 approaches to recovery from deadlock: Inform the system operator, and allow him/her to take manual intervention.
- Terminate one or more processes involved in the deadlock
- Preempt resources.
- **Process Termination - Two basic approaches:**
- Terminate all processes involved in the deadlock.
- Terminate processes one by one until the deadlock is broken.

conservative, but requires deadlock detection after each step.

2^{nd} Solution Details

- Process priorities.
- time serviced already and remaining time
- How many and what type of resources is the process holding. (Are they easy to preempt and restore?)
- Count of resources additionally required for completion
- No of processes to be terminated
- Interactive or batch process.
- non-restorable changes to any resource.

Resource Pre-emption

Selecting a victim - Deciding which resources to preempt from which processes – as discussed earlier

Rollback - ideal rollback a preempted process to a safe state prior to the point at which that resource was originally allocated to the process.

Starvation - to avoid starvation ; One option would be to use a priority system, and increase the priority of a process based on number of prempts