Resource = $\mathbf{R} = (R_1, R_2, \dots, R_m)$	Total amount of each resource in the system
Available = $\mathbf{V} = (V_1, V_2, \dots, V_m)$	Total amount of each resource not allocated to any process
Claim = $\mathbf{C} = \begin{pmatrix} C_{11} & C_{12} & \dots & C_{1m} \\ C_{21} & C_{22} & \dots & C_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ C_{n1} & C_{n2} & \dots & C_{nm} \end{pmatrix}$	C_{ij} = requirement of process i for resource j
Allocation = $\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1m} \\ A_{21} & A_{22} & \dots & A_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ A_{n1} & A_{n2} & \dots & A_{nm} \end{pmatrix}$	$A_{ij} = \text{current allocation to process } i \text{ of resource } j$

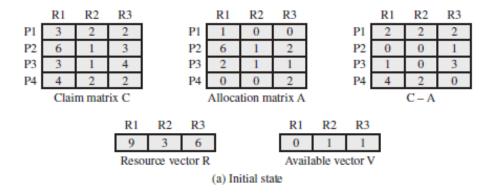
6.3 / DEADLOCK AVOIDANCE 305

```
struct state {
    int resource[m];
    int available[m];
    int claim[n][m];
    int alloc[n][m];
}
```

(a) Global data structures

(b) Resource allocation algorithm

Example:



Deadlock detection algorithm

Example:

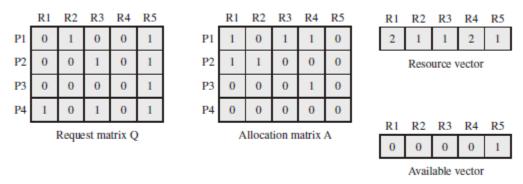


Figure 6.10 Example for Deadlock Detection

- 1. Mark each process that has a row in the Allocation matrix of all zeros. A process that has no allocated resources cannot participate in a deadlock.
- 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 ith row of Q is less than or equal to W. That is, $Q_{ik} \leq W_{ik}$, for $1 \leq k \leq m$. If no such row is found, terminate the algorithm.
- 4. If such a row is found, mark process i and add the corresponding row of the allocation matrix to W. That is, set $W_k = W_k + A_{ik}$, for $1 \le k \le m$. Return to step 3.