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


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

(a) Global data structures

```
if (alloc [i,*] + request [*] > claim [i,*])
      <error>;                    /* total request > claim*/
else if (request [*] > available [*])
     <suspend process>;
else  {                                 /* simulate alloc */
     <define newstate by:
     alloc [i,*] = alloc [i,*] + request [*];
     available [*] = available [*] - request [*]>;
}
if  (safe (newstate))
      <carry out allocation>;
else  {
      <restore original state>;
      <suspend process>;
}
```

(b) Resource allocation algorithm

```
boolean safe (state S) {
    int currentavail[m];
    process rest[<number of processes>];
    currentavail = available;
    rest = {all processes};
    possible = true;
    while (possible) {
       <find a process Pk in rest such that
       claim [k,*] - alloc [k,*]<= currentavail;
       if (found) {         /* simulate execution of Pk */
          currentavail = currentavail + alloc [k,*];
          rest = rest - {Pk};
       }
       else  possible = false;
    }
    return (rest == null);
}
```

(c) Test for safety algorithm (banker's algorithm)

VideoNote **Figure 6.9 Deadlock Avoidance Logic**

Example:

Claim matrix C

	R1	R2	R3
P1	3	2	2
P2	6	1	3
P3	3	1	4
P4	4	2	2

Allocation matrix A

	R1	R2	R3
P1	1	0	0
P2	6	1	2
P3	2	1	1
P4	0	0	2

C – A

	R1	R2	R3
P1	2	2	2
P2	0	0	1
P3	1	0	3
P4	4	2	0

Resource vector R

R1	R2	R3
9	3	6

Available vector V

R1	R2	R3
0	1	1

(a) Initial state

Deadlock detection algorithm

Example:

Request matrix Q

	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

Allocation matrix A

	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

Resource vector

R1	R2	R3	R4	R5
2	1	1	2	1

Available vector

R1	R2	R3	R4	R5
0	0	0	0	1

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 *i*th 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 $1 \leq k \leq m$. Return to step 3.**