Exp 5: Simulate Bankers Algorithm for Deadlock Avoidance

Suppose n is the number of processes, and m is the number of each type of resource used in a computer system.

1. **Available**: It is an array of length 'm' that defines each type of resource available in the system. When Available[j] = K, means that 'K' instances of Resources type R[j] are available in the system.
2. **Max:** It is a [n x m] matrix that indicates each process P[i] can store the maximum number of resources R[j] (each type) in a system.
3. **Allocation:** It is a matrix of m x n orders that indicates the type of resources currently allocated to each process in the system. When Allocation [i, j] = K, it means that process P[i] is currently allocated K instances of Resources type R[j] in the system.
4. **Need:** It is an M x N matrix sequence representing the number of remaining resources for each process. When the Need[i] [j] = k, then process P[i] may require K more instances of resources type Rj to complete the assigned work.
   Nedd[i][j] = Max[i][j] - Allocation[i][j].
5. **Finish**: It is the vector of the order **m**. It includes a Boolean value (true/false) indicating whether the process has been allocated to the requested resources, and all resources have been released after finishing its task.

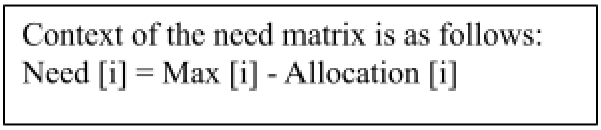
Safety Algorithm

It is a safety algorithm used to check whether or not a system is in a safe state or follows the safe sequence in a banker's algorithm:

1. There are two vectors **Wok** and **Finish** of length m and n in a safety algorithm.
   Initialize: Work = Available
   Finish[i] = false; for I = 0, 1, 2, 3, 4… n - 1.

2. Check the availability status for each type of resources [i], such as:

   Need[i] <= Work (Available)
   Finish[i] == false
   If the i does not exist, go to step 4.

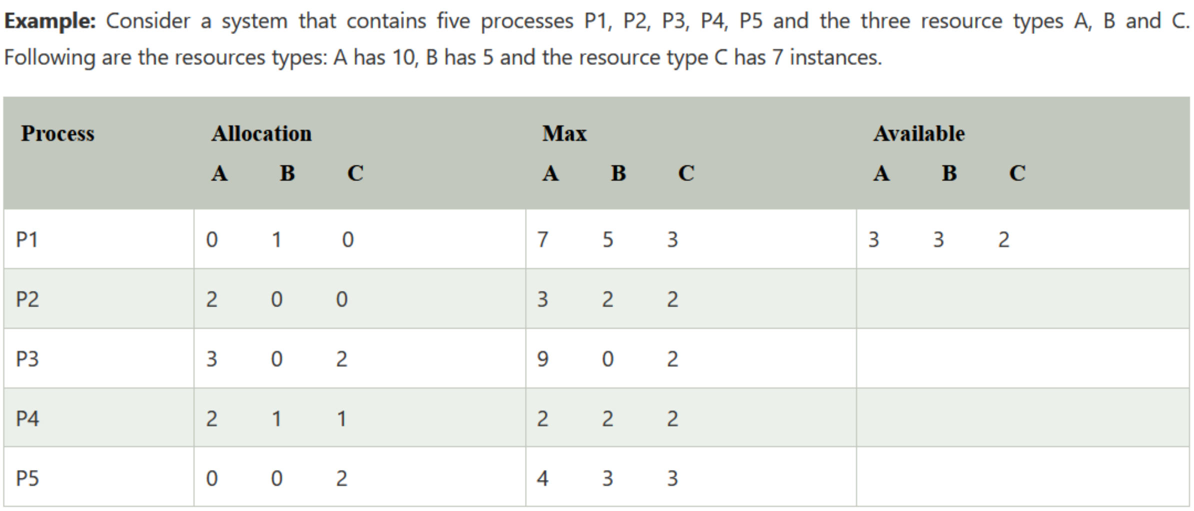

Context of the need matrix is as follows:
Need [i] = Max [i] - Allocation [i]

3. Work = Work +Allocation(i) // to get new resource allocation

   Finish[i] = true

   Go to step 2 to check the status of resource availability for the next process.

4. If Finish[i] == true; it means that the system is safe for all processes.

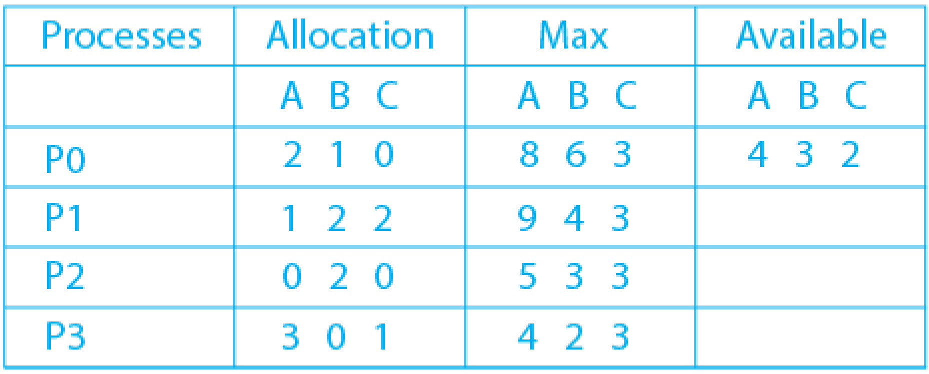


Example: Consider a system that contains five processes P1, P2, P3, P4, P5 and the three resource types A, B and C. Following are the resources types: A has 10, B has 5 and the resource type C has 7 instances.

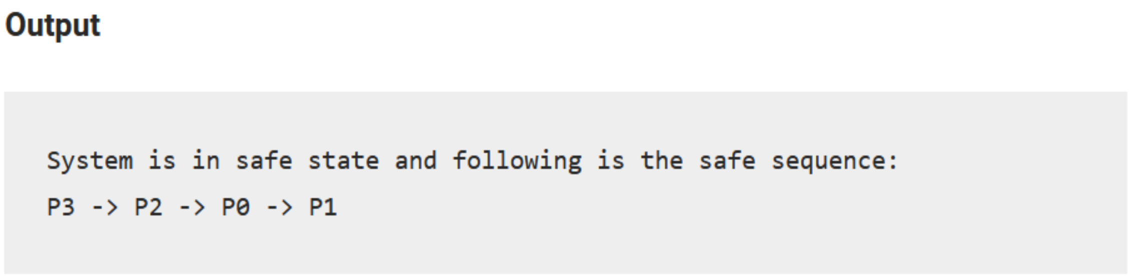
Process	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P1	0	1	0	7	5	3	3	3	2
P2	2	0	0	3	2	2			
P3	3	0	2	9	0	2			
P4	2	1	1	2	2	2			
P5	0	0	2	4	3	3			

Hence, we execute the banker's algorithm to find the safe state and the safe sequence like P2, P4, P5, P1 and P3.

Example:



Processes	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P0	2	1	0	8	6	3	4	3	2
P1	1	2	2	9	4	3			
P2	0	2	0	5	3	3			
P3	3	0	1	4	2	3			



Output

```
System is in safe state and following is the safe sequence:
P3 -> P2 -> P0 -> P1
```