BANKER’S ALGORITHM

<u>Define deadlock and implement methods for its avoidance, detection and identify goals of protection.</u>

Banker’s Algorithm

• Banker’s Algorithm is used to determine whether a process’s request for allocation of resources be safely granted immediately.

or

• The grant of request be deferred to a later stage.

• For the banker’s algorithm to operate, each process has to a priori specify its maximum requirement of resources.

• A process is admitted for execution only if its maximum requirement of resources is within the system capacity of resources.

•The Banker’s algorithm is an example of resource allocation policy that avoids deadlock.

Example:- Consider the following table of a system:

Process	Allocated				Max				Work Available				Need (Max – Allocation)			
	R1	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
P1	0	0	1	2	0	0	1	2	2	1	0	0				
P2	2	0	0	0	2	7	5	0								
P3	0	0	3	4	6	6	5	0								
P4	2	3	5	4	4	3	5	6								
P5	0	3	3	2	0	6	5	2								

1. Compute NEED Matrix.
2. Is the system in safe state? Justify.

Solution:- Consider the following table of the system:

Process	Allocated				Max				Work Available				Need (Max – Allocation)			
	R1	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
P1	0	0	1	2	0	0	1	2	2	1	0	0				
P2	2	0	0	0	2	7	5	0								
P3	0	0	3	4	6	6	5	0								
P4	2	3	5	4	4	3	5	6								
P5	0	3	3	2	0	6	5	2								

1. **Compute NEED Matrix = ?**

Need [i] = Max[i] - Allocated[i],

Therefore,

Need Matrix

NEED MATRIX	R1	R2	R3	R4
P1	0	0	0	0
P2	0	7	5	0
P3	6	6	2	4
P4	2	0	0	2
P5	0	3	2	0

Final Table

Process	Allocated				Max				Work Available				Need (Max – Allocation)			
	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
P1	0	0	1	2	0	0	1	2	2	1	0	0	0	0	0	0
P2	2	0	0	0	2	7	5	0					0	7	5	0
P3	0	0	3	4	6	6	5	0					6	6	2	4
P4	2	3	5	4	4	3	5	6					2	0	0	2
P5	0	3	3	2	0	6	5	2					0	3	2	0

2. **Is the system is Safe State?**

<u>By applying the Banker's Algorithm:</u>

Let **Avail** = Available; i.e . Avail = {2,1,0,0}

Iteration 1. Check all processes from P1 to P5.

<u>For P1:→</u>

if (**P1 Need <= Work Available**)→TRUE

{0,0,0,0} <= {2,1,0,0} →TRUE

then calculate

Work Available= Work Available + Allocated [P1]

= {2,1,0,0} + = {0,0,1,2}

Work Available = **{2,1,1,2}**

Final Table with new Work Available

Process	Allocated				Max				Work Available				Need (Max – Allocation)			
	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
P1	0	0	1	2	0	0	1	2	2	1	0	0	0	0	0	0
P2	2	0	0	0	2	7	5	0	**2**	**1**	**1**	**2**	0	7	5	0
P3	0	0	3	4	6	6	5	0					6	6	2	4
P4	2	3	5	4	4	3	5	6					2	0	0	2
P5	0	3	3	2	0	6	5	2					0	3	2	0

2. **Is the system is Safe State?**

<u>**By applying the Banker's Algorithm:**</u>

Iteration 1.

<u>For P2:→</u>

if (**P2 Need <= Work Available**)→FALSE

{0,7,5,0} <= {2,1,1,2} →FALSE

//then Check for next process.

2. **Is the system is Safe State?**

<u>By applying the Banker's Algorithm:</u>

Iteration 1.

<u>For P3:→</u>

if (**P3 Need <= Work Available**)→FALSE

{6,6,2,4} <= {2,1,1,2} →FALSE

//then Check for next process.

2. **Is the system is Safe State?**

By applying the Banker's Algorithm:

Iteration 1.

For P4:→

if (**P4 Need <= Work Available**)→TRUE

{2,0,0,2} <= {2,1,1,2} →TRUE

then calculate

Work Available = Work Available + Allocated [P4]

= {2,1,1,2} + = {2,3,5,4}

Work Available = **{4,4,6,6}**

Final Table with new Work Available

Process	Allocated				Max				Work Available				Need (Max – Allocation)			
	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
P1	0	0	1	2	0	0	1	2	2	1	0	0	0	0	0	0
P2	2	0	0	0	2	7	5	0	2	1	1	2	0	7	5	0
P3	0	0	3	4	6	6	5	0	**4**	**4**	**6**	**6**	6	6	2	4
P4	2	3	5	4	4	3	5	6					2	0	0	2
P5	0	3	3	2	0	6	5	2					0	3	2	0

2. **Is the system is Safe State?**

<u>By applying the Banker's Algorithm:</u>

Iteration 1.

<u>For P5:→</u>

if (**P5 Need <= Work Available**)→TRUE

{0,3,2,0} <= {4,4,6,6} →TRUE

then calculate

Work Available = Work Available + Allocated [P5]

= {4,4,6,6} + = {0,3,3,2}

Work Available = **{4,7,9,8}**

Final Table with new Work Available

Process	Allocated				Max				Work Available				Need (Max – Allocation)			
	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
P1	0	0	1	2	0	0	1	2	2	1	0	0	0	0	0	0
P2	2	0	0	0	2	7	5	0	2	1	1	2	0	7	5	0
P3	0	0	3	4	6	6	5	0	4	4	6	6	6	6	2	4
P4	2	3	5	4	4	3	5	6	**4**	**7**	**9**	**8**	2	0	0	2
P5	0	3	3	2	0	6	5	2					0	3	2	0

2. **Is the system is Safe State?**

By applying the Banker's Algorithm:

Iteration 2. Check only process P2 to P3.

For P2:→

if (**P2 Need <= Work Available**)→TRUE

{0,7,5,0} <= {4,7,9,8} →TRUE

then calculate

Work Available = Work Available + Allocated [P2]

= {4,7,9,8} + = {2,0,0,0}

Work Available = **{6,7,9,8}**

Final Table with new Work Available

Process	Allocated				Max				Work Available				Need (Max – Allocation)			
	R1	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
P1	0	0	1	2	0	0	1	2	2	1	0	0	0	0	0	0
P2	2	0	0	0	2	7	5	0	2	1	1	2	0	7	5	0
P3	0	0	3	4	6	6	5	0	4	4	6	6	6	6	2	4
P4	2	3	5	4	4	3	5	6	4	7	9	8	2	0	0	2
P5	0	3	3	2	0	6	5	2	**6**	**7**	**9**	**8**	0	3	2	0

2. **Is the system is Safe State?**

By applying the Banker's Algorithm:

Iteration 2. Check only process P2 to P3.

For P3:→

if (**P3 Need <= Work Available**)→TRUE

{0,3,2,0} <= {6,7,9,8} →TRUE

then calculate

Work Available = Work Available + Allocated [P3]

= {6,7,9,8} + = {0,0,3,4}

Work Available = **{6,7,12,12} =System Capacity**

Final Table with new Work Available

Process	Allocated				Max				Work Available				Need (Max – Allocation)			
	R1	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
P1	0	0	1	2	0	0	1	2	2	1	0	0	0	0	0	0
P2	2	0	0	0	2	7	5	0	2	1	1	2	0	7	5	0
P3	0	0	3	4	6	6	5	0	4	4	6	6	6	6	2	4
P4	2	3	5	4	4	3	5	6	4	7	9	8	2	0	0	2
P5	0	3	3	2	0	6	5	2	6	7	9	8	0	3	2	0
									7	**7**	**12**	**12**				

Since, all the processes got TRUE marked, no further iterations are required.

Therefore, **Safe Sequence = P1, P4, P5, P2 , P3**

Therefore, the System is in the Safe State.