Banker’s Algorithm

Deadlock avoidance algorithm

	Allocation			Max Need			Available			Remaining Need		
	CPU	RAM	Port	CPU	RAM	Port	CPU	RAM	Port	CPU	RAM	Port
P0	0	1	0	7	5	3						
P1	2	0	0	3	2	2						
P2	3	0	2	9	0	2						
P3	2	1	1	4	2	2						
P4	0	0	2	5	3	3						

T.A.R: *7* *2* *5*

The system has total 10 CPUs, 5 Rams, 7 Ports

T.A.R -> Total number of allocated resources of each type.

For example: At current time, all 5 processes have allocated 7 CPU out of total 10 CPU, 2 RAM out of 5 RAM, 5 Ports out of 7 ports

Allocation: At current time, how many number of instances of each resources are already being allocated by processes.

Max Need: A process must have 'Max Need' number of instances of each resources at the same time to being executed. For example: P3 requires total 4 CPUs, 2 RAMs and 2 Ports at the same time to complete its execution.

Remaining Need = Max Need – Allocation

Remaining Need means how many number of instances of each resources are required by a process to be executed. For example, at current time, P3 has allocated 2 CPUs and need total of 4 CPUs, has allocated 1 RAM and need total 2 RAMs, has allocated 1 port and need total 2 Ports at the same time to being executed.

Available: This column contains the total number of available or free resources at current time.

------------------------------- Algorithms ------------------------------------

We will start this algorithm from P0 and then we will check P1, then P2, then P3, then P4. After that we will start again from incompleted processes.

Step 1:

At current time P0 has allocated 0 CPU, 1 RAM, 0 ports. P0 needs total of 7 CPU, 5 RAM, 3 Ports. We have used total of 7 CPU, 2 RAM, 5 Ports till current time. The system has (10-7) = 3 CPUs, (5-2) = 3 RAMs and (7-5) = 2 Ports free which can be allocated to any requested processes.

For P0 to be executed, it needs 7 CPU, 4 RAM and 3 Ports more at current time. But at current time, free and available numbers are respectively: 3, 3, 2. So, we process P0 cannot allocate resources.

	Allocation			Max Need			Available			Remaining Need		
	CPU	RAM	Port	CPU	RAM	Port	CPU	RAM	Port	CPU	RAM	Port
P0	0	1	0	7	5	3	3	3	2	7	4	3 ✗
P1	2	0	0	3	2	2				1	2	2
P2	3	0	2	9	0	2				6	0	0
P3	2	1	1	4	2	2				2	1	1
P4	0	0	2	5	3	3				5	3	1

T.A.R: *7* *2* *5*

Step 2:

Now we go for P1:

P1 needs 1, 2, 2 resources more and at current time, free and available numbers are respectively: 3, 3, 2. So, process P1 will allocate resources and will execute completely. After this P1 will release all of its allocated resources. So the number of total available/free resources will increase. Now the available number of resources are: (3+2) = 5 , (3+0) = 3 , (2+0) = 2.

	Allocation			Max Need			Available			Remaining Need		
	CPU	RAM	Port	CPU	RAM	Port	CPU	RAM	Port	CPU	RAM	Port
P0	0	1	0	7	5	3	3 +2	3 0	2 0	7	4	3
P1	2	0	0	3	2	2	5	3	2	~~1~~	~~2~~	~~2~~ ✓
P2	3	0	2	9	0	2				6	0	0
P3	2	1	1	4	2	2				2	1	1
P4	0	0	2	5	3	3				5	3	1

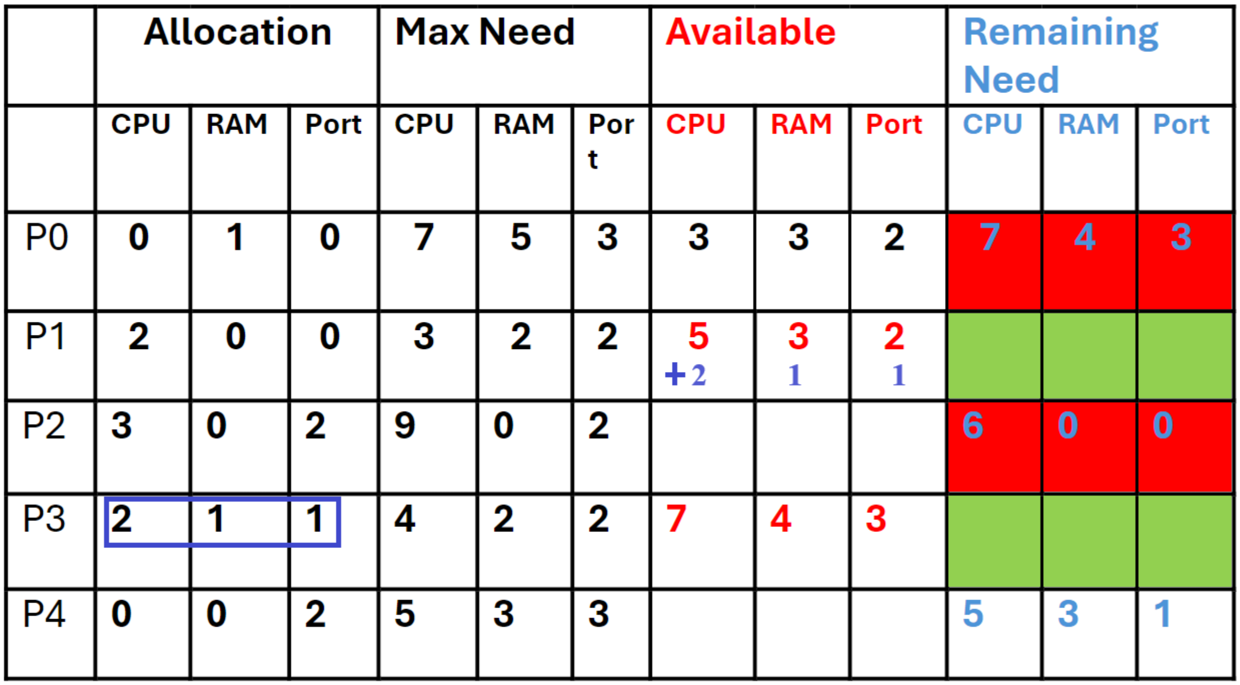
So first process which will complete its execution: P1

Step 3:

For P2:

We cannot allocate as 6 > 5.

For P3: P3 can allocated all three processes as per its need.



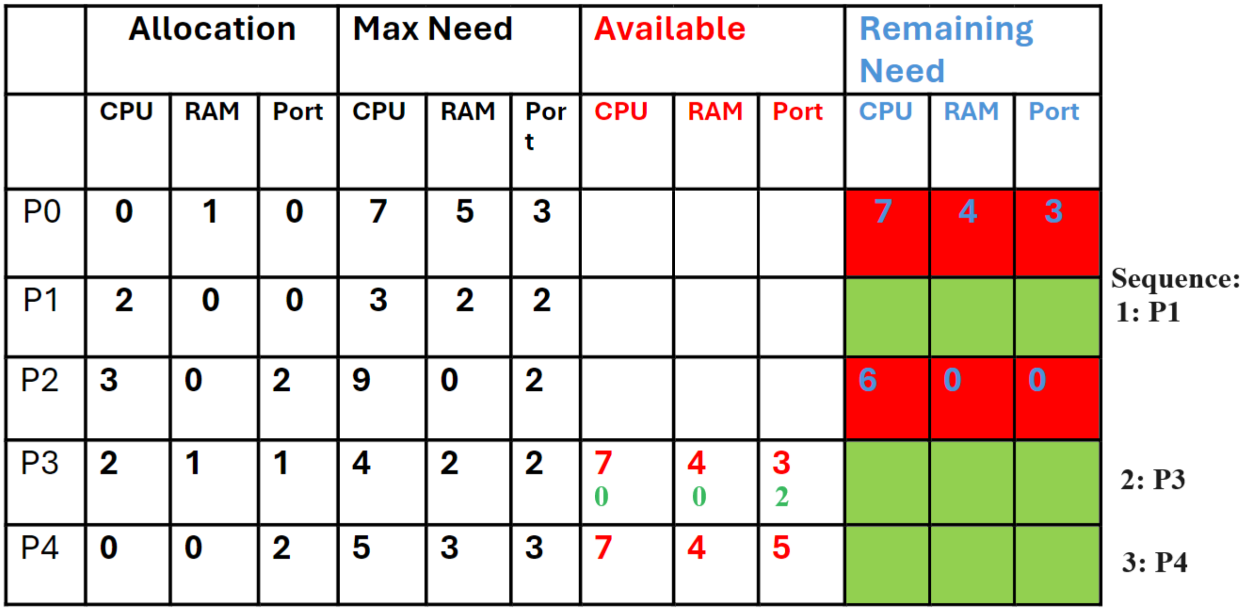
	Allocation			Max Need			Available			Remaining Need		
	CPU	RAM	Port	CPU	RAM	Port	CPU	RAM	Port	CPU	RAM	Port
P0	0	1	0	7	5	3	3	3	2	7	4	3
P1	2	0	0	3	2	2	5 +2	3 1	2 1	🟩	🟩	🟩
P2	3	0	2	9	0	2				6	0	0
P3	2	1	1	4	2	2	7	4	3	🟩	🟩	🟩
P4	0	0	2	5	3	3				5	3	1

2nd process which will complete its execution: P3

Step 4:

For P4:

P4 will be executed after P3 as total available resources are 7,4,3 and it needs 5,3,1. So it will execute and release its allocated resources.



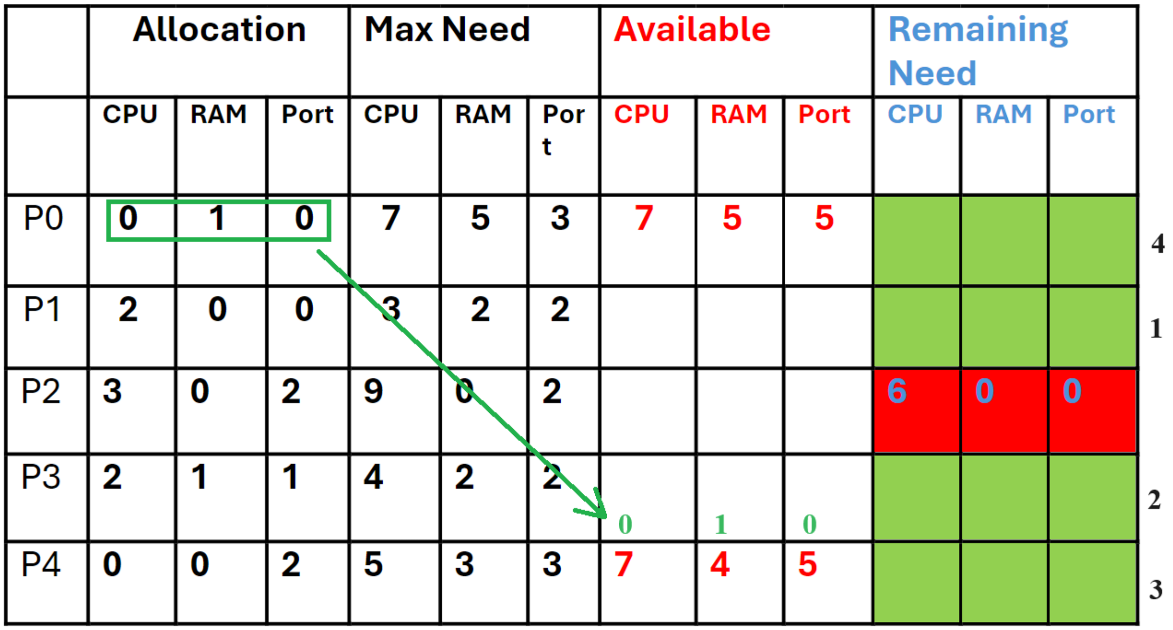
	Allocation			Max Need			Available			Remaining Need			
	CPU	RAM	Port	CPU	RAM	Port	CPU	RAM	Port	CPU	RAM	Port	
P0	0	1	0	7	5	3				7	4	3	
P1	2	0	0	3	2	2				🟩	🟩	🟩	Sequence: 1: P1
P2	3	0	2	9	0	2				6	0	0	
P3	2	1	1	4	2	2	7 0	4 0	3 2	🟩	🟩	🟩	2: P3
P4	0	0	2	5	3	3	7	4	5	🟩	🟩	🟩	3: P4

3rd process which will complete its execution: P4

Step 5:

Now our available resources are 7,4,5 and P0 needs 7,4,3/ So, P0 will complete its execution and release its allocated resources.

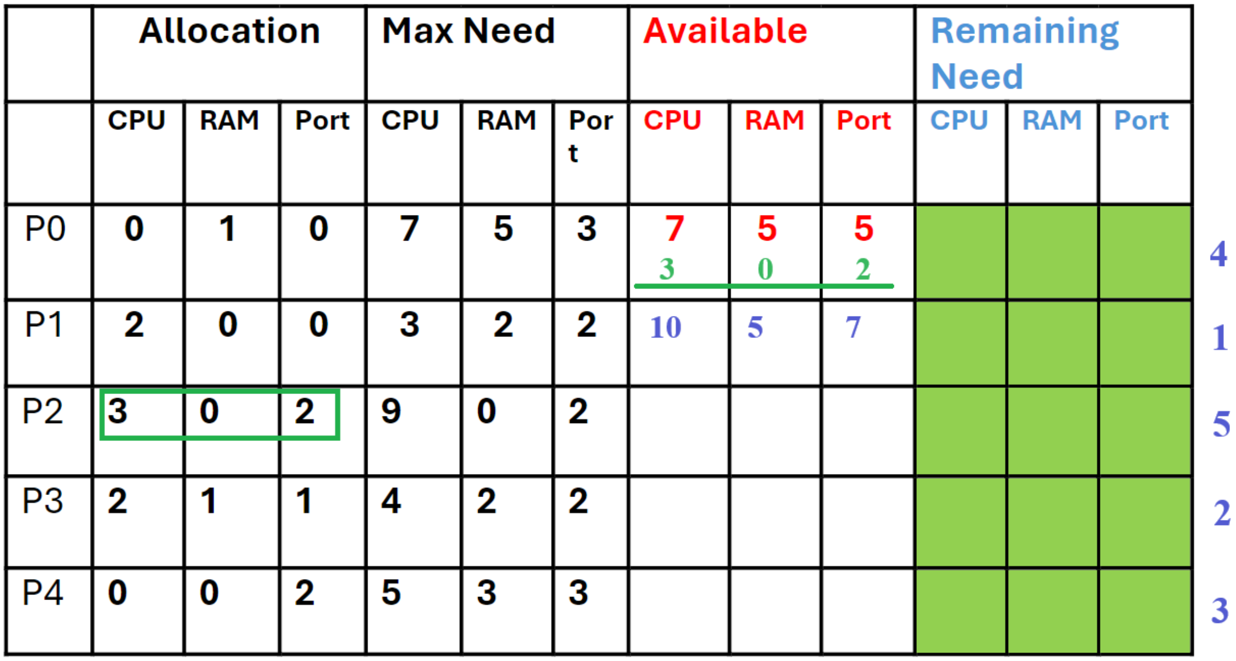
4th process which will complete its execution: P1



	Allocation			Max Need			Available			Remaining Need			
	CPU	RAM	Port	CPU	RAM	Port	CPU	RAM	Port	CPU	RAM	Port	
P0	0	1	0	7	5	3	7	5	5	🟩	🟩	🟩	4
P1	2	0	0	3	2	2				🟩	🟩	🟩	1
P2	3	0	2	9	0	2				6	0	0	
P3	2	1	1	4	2	2	0	1	0	🟩	🟩	🟩	2
P4	0	0	2	5	3	3	7	4	5	🟩	🟩	🟩	3

Step 6:

Finally P2 needs 6,0,0 more resources and available 7,5,5. So it will also complete its execution.



	Allocation			Max Need			Available			Remaining Need			
	CPU	RAM	Port	CPU	RAM	Port	CPU	RAM	Port	CPU	RAM	Port	
P0	0	1	0	7	5	3	7 3	5 0	5 2	🟩	🟩	🟩	4
P1	2	0	0	3	2	2	10	5	7	🟩	🟩	🟩	1
P2	3	0	2	9	0	2				🟩	🟩	🟩	5
P3	2	1	1	4	2	2				🟩	🟩	🟩	2
P4	0	0	2	5	3	3				🟩	🟩	🟩	3

So the safe sequence : P1-> P3 -> P4 -> P0 -> P2

If all processes complete its execution, no deadlock