Consider the following snapshot of a system

	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	1	5	2	0
P_1	1	0	0	0	1	7	5	0				
P_2	1	3	5	4	2	3	5	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				

With reference to Bankers algorithm

i) What is the content of the matrix need?

ii) Is the system in a safe state?

iii) If a request from process P1 arrives for (0, 4, 2, 0), can the request be granted immediately?

Need matrix is calculated by subtracting Allocation Matrix from the Max matrix

	Need(Max-Allocation)			
	A	B	C	D
P_0	0	0	0	0
P_1	0	7	5	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	6	4	2

To check if system is in a safe state

- The Available matrix is [1520][1520].
- A process after it has finished execution is supposed to free up all the resources it hold.
- We need to find a safety sequence such that it satisfies the criteria need Need≤AvailableNeed≤Available.

- Since $Need(P0) \leq Available$Need(P0)≤Available , we select $P0.[Available]=[Available]+[Allocation(P0)]$P0.[Available]=[Available]+[Allocation(P0)]

$Available=[1520]+[0012]=[1532]$Available=[1520]+[0012]=[1532]

- Need(P2) ≤ Available →→Available=[1 5 3 2]+[1 3 5 4]=[2 8 8 6]
- Need(P3) ≤ Available→→ Available=[2 8 8 6]+[0 6 3 2]=[2 14 11 8]
- Need(P4) ≤ Available→→Available=[2 14 11 8]+[0 0 1 4]=[2 14 12 12]
- Need(P1) ≤ Available→→Available=[2 14 12 12]+[1 0 0 0]=[3 14 12 12]
- Safe Sequence is <p0,p2,p3,p4,p1>

A request from process P1 arrives for (0,4,2,0)

- System receives a request for P1 for $Req(P1)[0420]$Req(P1)[0420]
- First we check if Req(P1) is less than $Need(P1)$ Need(P1) $\rightarrow [0420]<[0750] is true$ [0420]<[0750]istrue
- Now we check if Req(P1) is less than $Available$ Available $\rightarrow [0420]<[1520] is true$ [0420]<[1520]istrue .
- So we update the values as:
  - $Available=Available-Request=[1520]-[0420]=[1100]$Available=Available−Request=[1520]−[0420]=[1100]
  - $Allocation=allocation(P1)+Request=[1000]+[0420]=[1420]$Allocation=allocation(P1)+Request=[1000]+[0420]=[1420]
  - $Need=Need(P1)-Request=[0750]-[0420]=[0330]$Need=Need(P1)−Request=[0750]−[0420]=[0330]

	Allocation				Max				Need				Available			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	0	0	0	0	**1**	**1**	**0**	**0**
P_1	**1**	**4**	**2**	**0**	**1**	**7**	**5**	**0**	**0**	**3**	**3**	**0**				
P_2	1	3	5	4	2	3	5	6	1	0	0	2				
P_3	0	6	3	2	0	6	5	2	0	0	2	0				
P_4	0	0	1	4	0	6	5	6	0	6	4	2				

- This is the modified table
- On verifying, we see that the safe sequence still remains the same .The system continues to remain in a safe state.
- **Example:**
  Considering a system with five processes P_0 through P_4 and three resources of type A, B, C. Resource type A has 10 instances, B has 5 instances and type C has 7 instances. Suppose at time t_0 following snapshot of the system has been taken:

Process	Allocation	Max	Available
	A B C	A B C	A B C
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

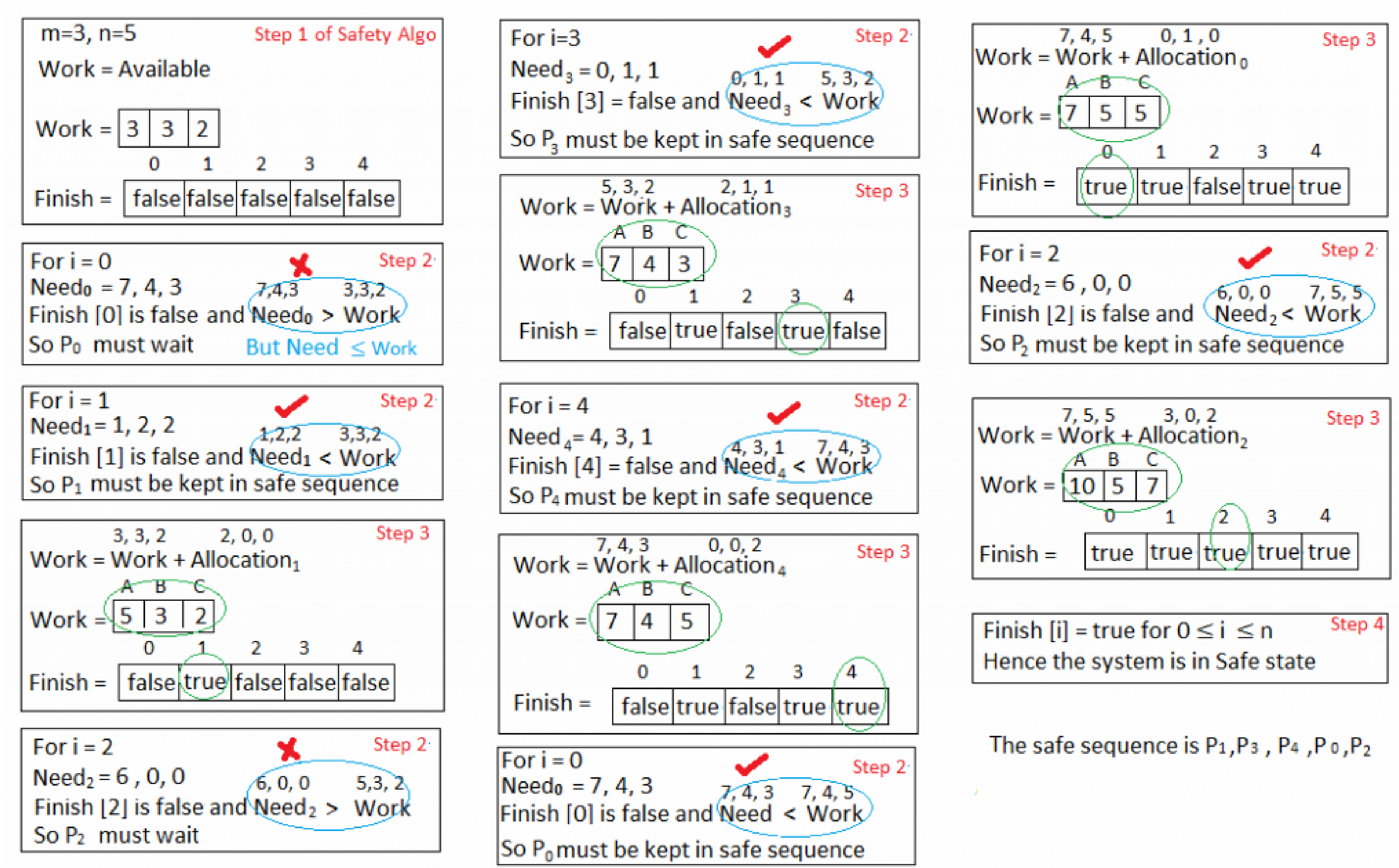
- **Question1. What will be the content of the Need matrix?**
  Need [i, j] = Max [i, j] – Allocation [i, j]

So, the content of Need Matrix is:

Process	Need		
	A	B	C
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

- **Question2. Is the system in a safe state? If Yes, then what is the safe sequence?**
  Applying the Safety algorithm on the given system,

•



- **Question3. What will happen if process P_1 requests one additional instance of resource type A and two instances of resource type C?**

$$\begin{array}{l} \quad\quad\quad\quad\;\; A\;\; B\;\; C \\ Request_1 = 1, 0, 2 \end{array}$$

To decide whether the request is granted we use Resource Request algorithm

Step 1

$$\overset{1,\,0,\,2}{Request_1} < \overset{1,\,2,\,2}{Need_1}$$

Step 2

$$\overset{1,\,0,\,2}{Request_1} < \overset{3,\,3,\,2}{Available}$$

Step 3

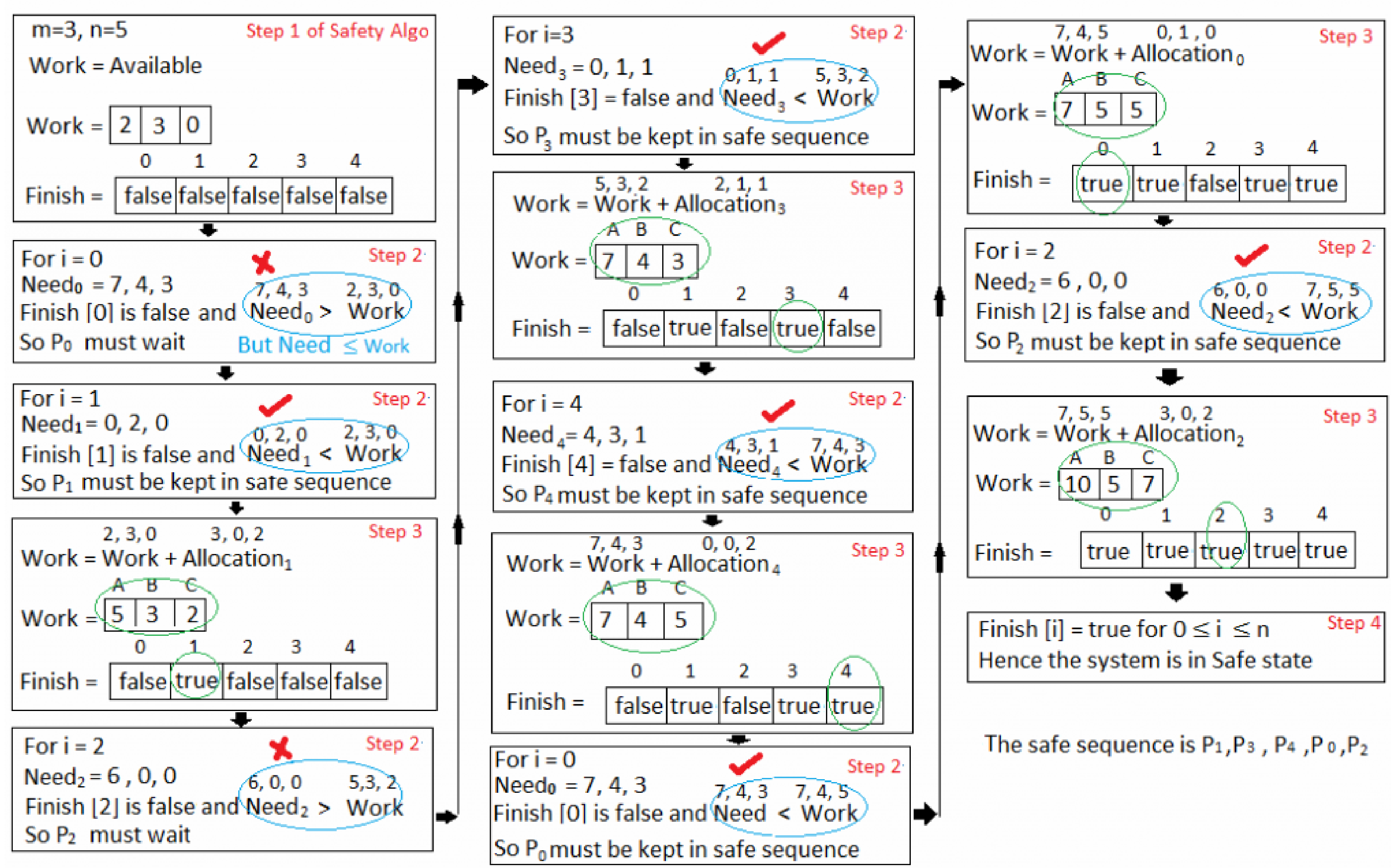
Available = Available – $Request_1$

$Allocation_1$ = $Allocation_1$ + $Request_1$

$Need_1$ = $Need_1$ - $Request_1$

Process	Allocation	Need	Available
	A B C	A B C	A B C
P_0	0 1 0	7 4 3	2 3 0
P_1	3 0 2	0 2 0	
P_2	3 0 2	6 0 0	
P_3	2 1 1	0 1 1	
P_4	0 0 2	4 3 1	

- We must determine whether this new system state is safe. To do so, we again execute Safety algorithm on the above data structures.

- 

- Hence the new system state is safe, so we can immediately grant the request for process **P_1**.

Bankers algorithm problems

Prob. Consider the following snapshot of a system-

Process	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P0	0	0	1	2	0	0	1	2	1	5	2	0
P1	1	0	0	0	1	7	5	0				
P2	1	3	5	4	2	3	5	6				
P3	0	6	3	2	0	6	5	2				
P4	0	0	1	4	0	6	5	6				

Answer the following questions using the Banker's algorithm-

(i) What is the content of the matrix need?
(ii) Is the system in a safe state?

(iii) If a request from process P1 arrives for (0,4,2,0), can the request be granted immediately?

Ans.

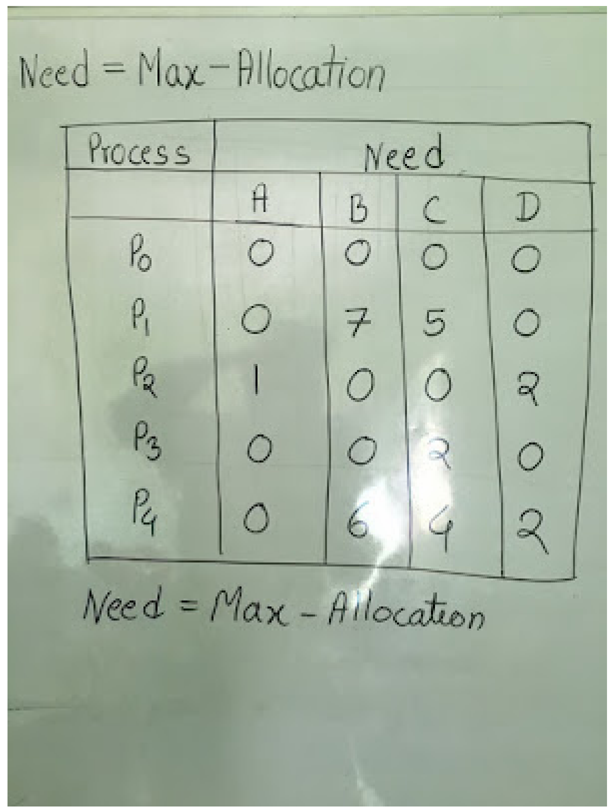
Banker's algorithm: Need Calculation

Steps to calculate need:

Step 1: in row of process P0, use formula
Need=Max – Allocation

Step 2: Follow step 1 above for all other processes i.e. P1, P2, P3, P4, P5.

Result given below.



Need = Max-Allocation

Process	Need			
	A	B	C	D
P_0	0	0	0	0
P_1	0	7	5	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	6	4	2

Need = Max - Allocation

Bankers algorithm: Need calculation

Steps to calculate Safe state:

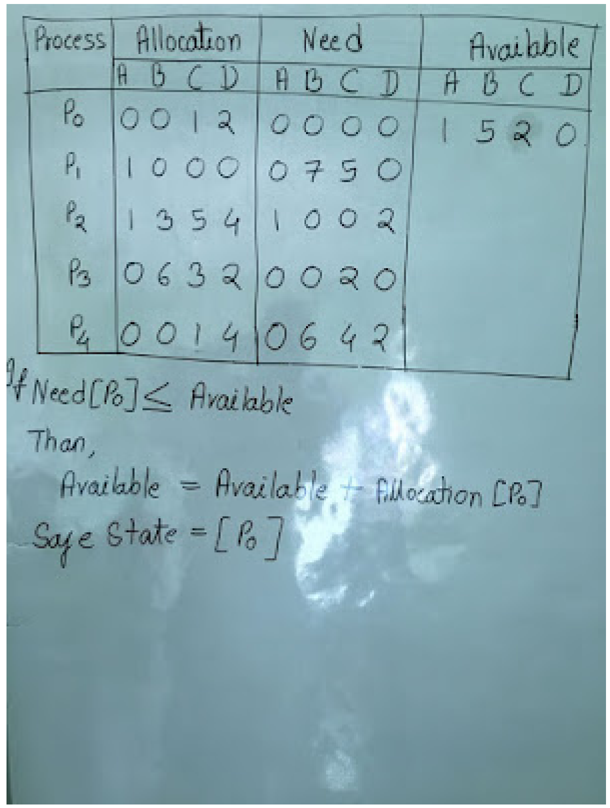


Image 1

Step 1: Find the process which have Need lesser than Available.

If **need** of process is lesser than **available,** add its **allocation** to the **available** and remove that process from the table.

Here as shown below,

Need [Process P0] is less than Available
Than,
New **Available = Available + Allocation [Process P0]** (Available value is updated in Image 2)

So ,

Safe state = [P0].

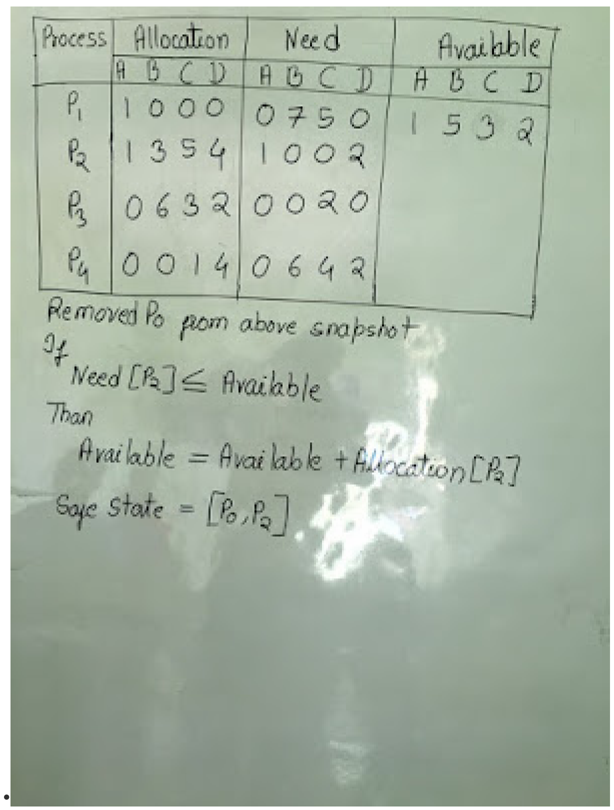


Image 2

Step 2: In STEP 1, We find the process P0 which have Need lesser than Available.
We removed Process P0 from table, and updated the Available. (See Image 2).

Now we repeat Step 1 as explain above.

Here as shown below,

Need [Process P2] is less than Available
Than,
New **Available = Available + Allocation [Process P2]**(Available value is updated in Image 3)
So ,

Safe state = [P0, P2].

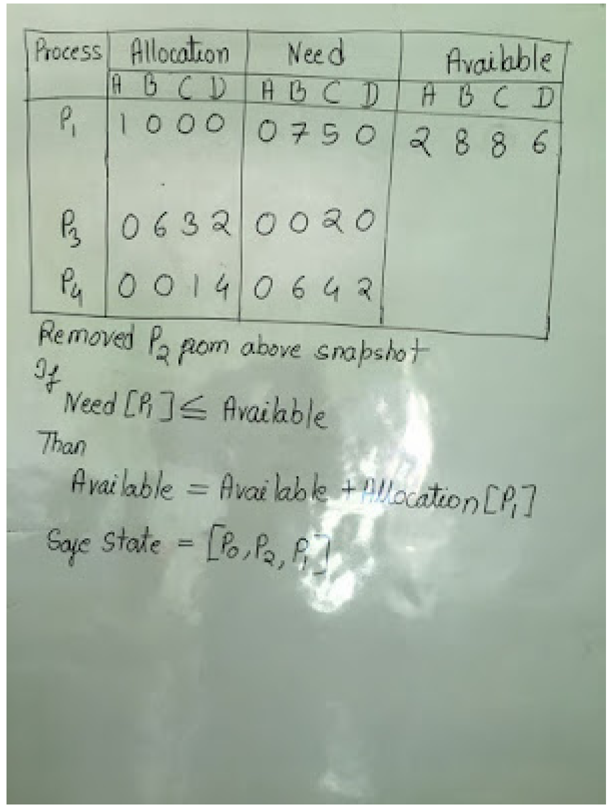


Image 3

Step 3: In STEP 2, we find the process P2 which have Need lesser than Available.
We removed Process P2 from table, and updated the Available.(See Image 3).

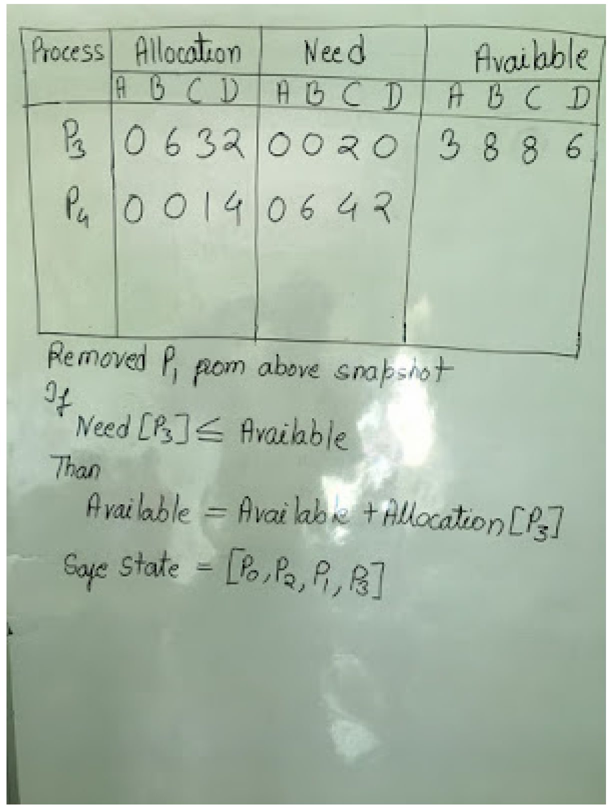
Now we repeat Step 1 as explain above.

Here as shown below,

Need [Process P1] is less than Available
Than,

New **Available = Available + Allocation [Process P1]**(Available value is updated in Image 4)
So ,

Safe state = [P0, P2, P1].



Process	Allocation A B C D	Need A B C D	Available A B C D
P_3	0 6 3 2	0 0 2 0	3 8 8 6
P_4	0 0 1 4	0 6 4 2	

Removed P_1 from above snapshot
If
Need [P_3] $\leq$ Available
Than
Available = Available + Allocation [P_3]
Safe State = [P_0, P_2, P_1, P_3]

Image 4

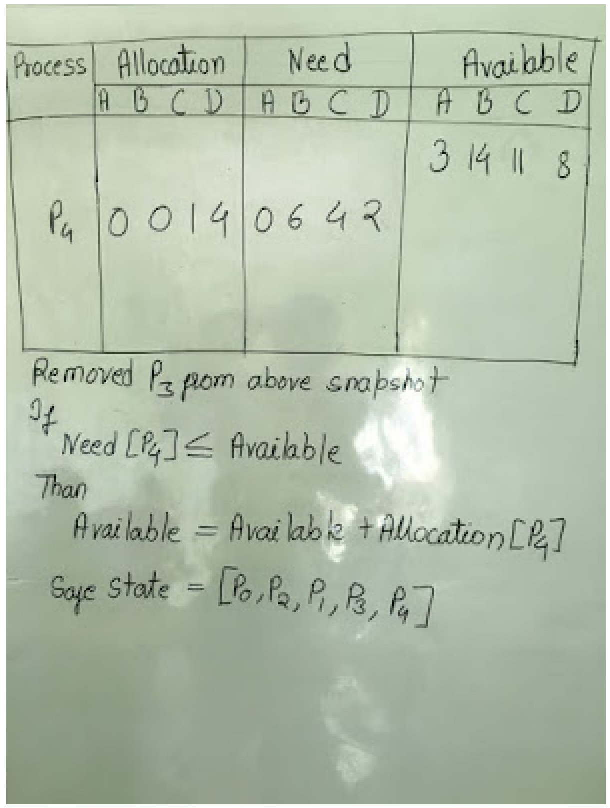
Step 4: In STEP 3, we find the process P1 which have Need lesser than Available.
We removed Process P1 from table, and updated the Available.(See Image 4).

Now we repeat Step 1 as explain above.

Here as shown below,

Need [Process P3] is less than Available
Than,
New **Available = Available + Allocation [Process P3]**(Available value is updated in Image 5)
So ,

Safe state = [P0, P2, P1, P3].



Process	Allocation				Need				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
									3	14	11	8
P_4	0	0	1	4	0	6	4	2				

Removed P_3 from above snapshot
If
Need $[P_4] \leq$ Available
Than
Available = Available + Allocation $[P_4]$
Safe state = $[P_0, P_2, P_1, P_3, P_4]$

Image 5

Step 5: In STEP 4, we find the process P3 which have Need lesser than Available. We removed Process P3 from table, and updated the Available.(See Image 5).

Now we repeat Step 1 as explain above.

Here as shown below,

Need [Process P4] is less than Available

So ,

Safe state = [P0, P2, P1, P3, P4].

As all the processes comes under safe state, so system is in a safe state.