Operating System| Banker’s Algorithm

The banker’s algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes an “s-state” check to test for possible activities, before deciding whether allocation should be allowed to continue.

Following **Data structures** are used to implement the Banker’s Algorithm:

Let **‘n’** be the number of processes in the system and **‘m’** be the number of resources types.

Available :

- It is a 1-d array of size **‘m’** indicating the number of available resources of each type.
- Available[j] = k means there are **‘k’** instances of resource type $\mathbf{R_j}$

Max :

- It is a 2-d array of size ‘**n*m’** that defines the maximum demand of each process in a system.
- Max[i, j] = k means process $\mathbf{P_i}$ may request at most **‘k’** instances of resource type $\mathbf{R_{j.}}$

Allocation :

- It is a 2-d array of size **‘n*m’** that defines the number of resources of each type currently allocated to each process.
- Allocation[i, j] = k means process $\mathbf{P_i}$ is currently allocated **‘k’** instances of resource type $\mathbf{R_j}$

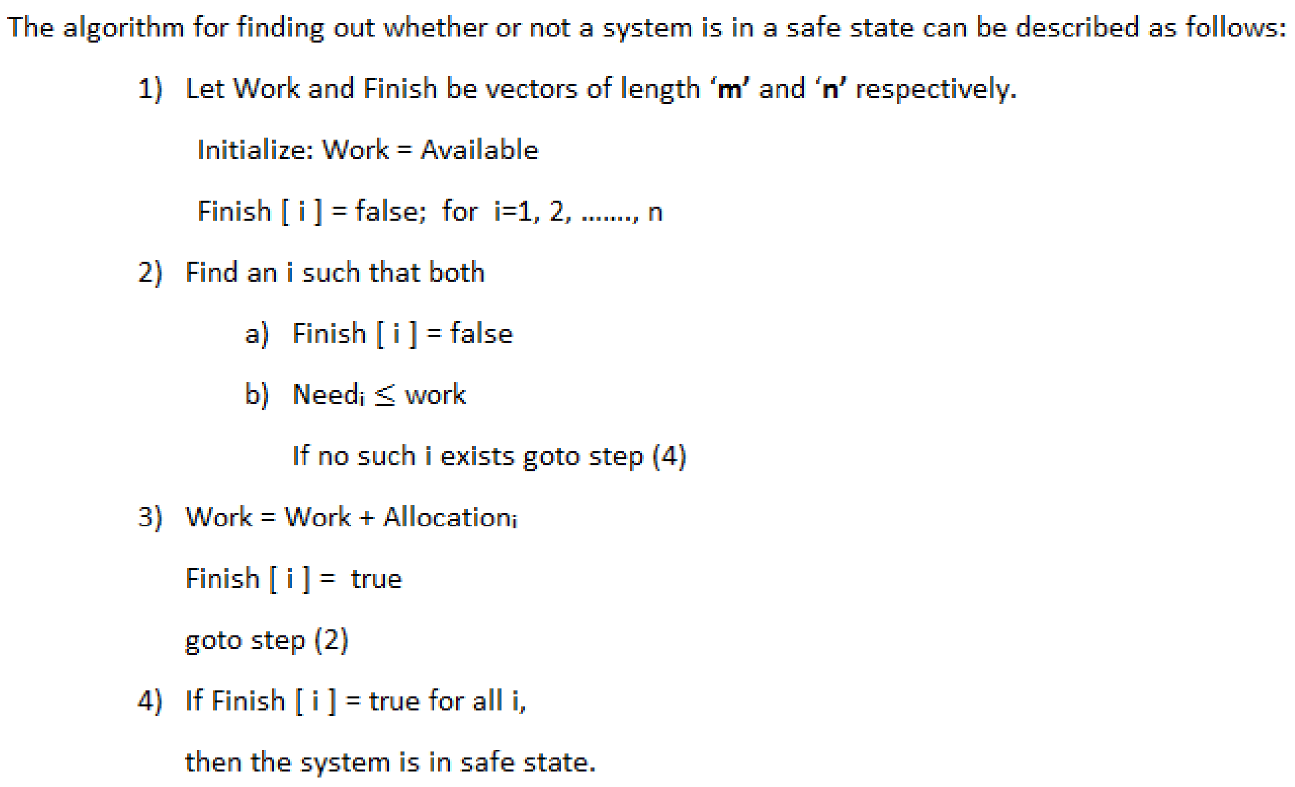
Need :

- It is a 2-d array of size **‘n*m’** that indicates the remaining resource need of each process.
- Need [i, j] = k means process $\mathbf{P_i}$ currently allocated **‘k’** instances of resource type $\mathbf{R_j}$
- Need [i, j] = Max [i, j] – Allocation [i, j]

Allocation_i specifies the resources currently allocated to process P_i and Need_i specifies the additional resources that process P_i may still request to complete its task.

Banker’s algorithm consist of Safety algorithm and Resource request algorithm

Safety Algorithm

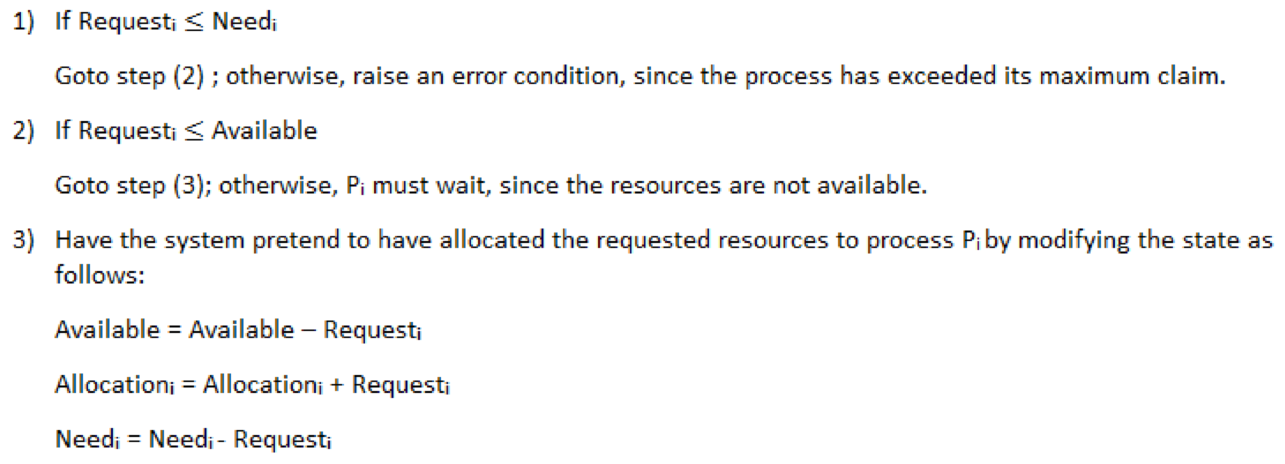


The algorithm for finding out whether or not a system is in a safe state can be described as follows:

```
1) Let Work and Finish be vectors of length 'm' and 'n' respectively.
    Initialize: Work = Available
    Finish [ i ] = false;  for  i=1, 2, ......., n
2) Find an i such that both
    a) Finish [ i ] = false
    b) $Need_i \leq work$
       If no such i exists goto step (4)
3) $Work = Work + Allocation_i$
   Finish [ i ] =  true
   goto step (2)
4) If Finish [ i ] = true for all i,
   then the system is in safe state.
```

Resource-Request Algorithm

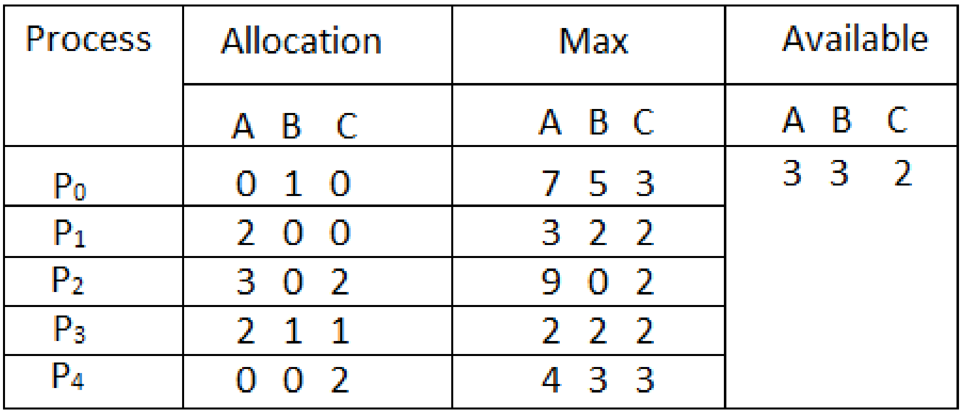
Let $Request_i$ be the request array for process P_i. $Request_i$ [j] = k means process P_i wants k instances of resource type R_j. When a request for resources is made by process P_i, the following actions are taken:



```
1) If $Request_i \leq Need_i$
   Goto step (2) ; otherwise, raise an error condition, since the process has exceeded its maximum claim.
2) If $Request_i \leq Available$
   Goto step (3); otherwise, $P_i$ must wait, since the resources are not available.
3) Have the system pretend to have allocated the requested resources to process $P_i$ by modifying the state as
   follows:
   $Available = Available – Request_i$
   $Allocation_i = Allocation_i + Request_i$
   $Need_i = Need_i - Request_i$
```

Example:

Considering a system with five processes P_0 through P_4 and three resources types 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:

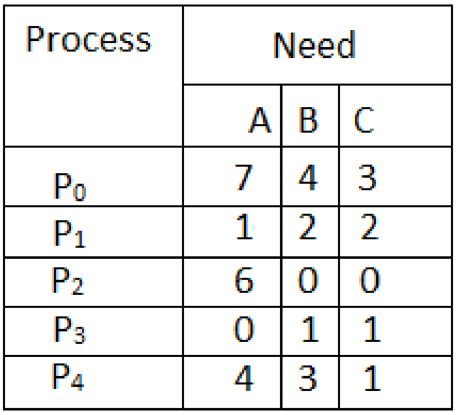


Process	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	3	3	2
P_1	2	0	0	3	2	2			
P_2	3	0	2	9	0	2			
P_3	2	1	1	2	2	2			
P_4	0	0	2	4	3	3			

Question1. What will be the content of the Need matrix?

Need [i, j] = Max [i, j] – Allocation [i, j]

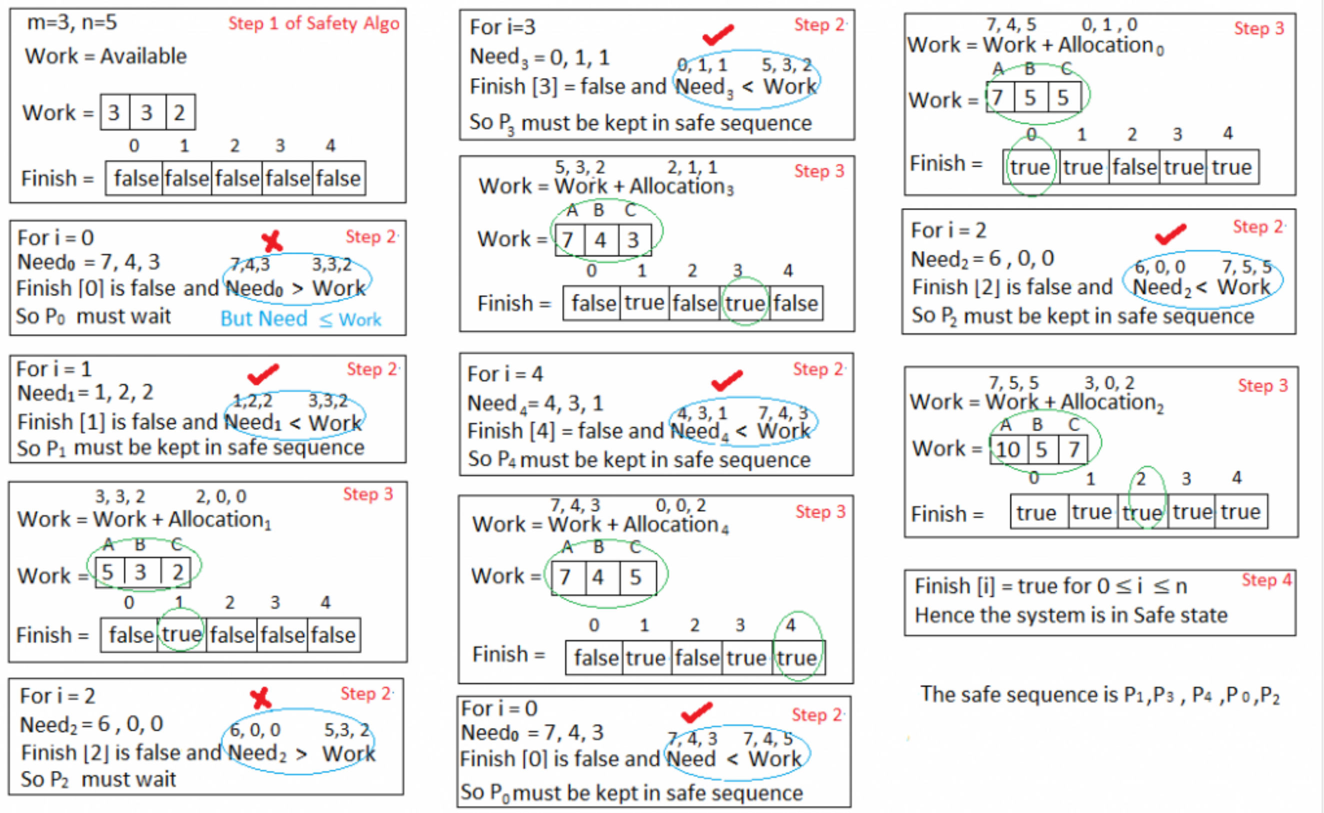
So, the content of Need Matrix is:



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 safe state? If Yes, then what is the safe sequence?

Applying the Safety algorithm on the given system,



m=3, n=5 Step 1 of Safety Algo

Work = Available

Work = 3 | 3 | 2

Finish =

0	1	2	3	4
false	false	false	false	false

For i = 0 ✗ Step 2

$Need_0$ = 7, 4, 3

Finish [0] is false and $\overset{7,4,3}{Need_0} > \overset{3,3,2}{Work}$

So P_0 must wait But Need ≤ Work

For i = 1 ✓ Step 2

$Need_1$ = 1, 2, 2

Finish [1] is false and $\overset{1,2,2}{Need_1} < \overset{3,3,2}{Work}$

So P_1 must be kept in safe sequence

Step 3

Work = $\overset{3,3,2}{Work} + \overset{2,0,0}{Allocation_1}$

Work = 5 | 3 | 2 (A B C)

Finish =

0	1	2	3	4
false	true	false	false	false

For i = 2 ✗ Step 2

$Need_2$ = 6 , 0, 0

Finish [2] is false and $\overset{6,0,0}{Need_2} > \overset{5,3,2}{Work}$

So P_2 must wait

For i=3 ✓ Step 2

$Need_3$ = 0, 1, 1

Finish [3] = false and $\overset{0,1,1}{Need_3} < \overset{5,3,2}{Work}$

So P_3 must be kept in safe sequence

Step 3

Work = $\overset{5,3,2}{Work} + \overset{2,1,1}{Allocation_3}$

Work = 7 | 4 | 3 (A B C)

Finish =

0	1	2	3	4
false	true	false	true	false

For i = 4 ✓ Step 2

$Need_4$ = 4, 3, 1

Finish [4] = false and $\overset{4,3,1}{Need_4} < \overset{7,4,3}{Work}$

So P_4 must be kept in safe sequence

Step 3

Work = $\overset{7,4,3}{Work} + \overset{0,0,2}{Allocation_4}$

Work = 7 | 4 | 5 (A B C)

Finish =

0	1	2	3	4
false	true	false	true	true

For i = 0 ✓ Step 2

$Need_0$ = 7, 4, 3

Finish [0] is false and $\overset{7,4,3}{Need} < \overset{7,4,5}{Work}$

So P_0 must be kept in safe sequence

Step 3

Work = $\overset{7,4,5}{Work} + \overset{0,1,0}{Allocation_0}$

Work = 7 | 5 | 5 (A B C)

Finish =

0	1	2	3	4
true	true	false	true	true

For i = 2 ✓ Step 2

$Need_2$ = 6 , 0, 0

Finish [2] is false and $\overset{6,0,0}{Need_2} < \overset{7,5,5}{Work}$

So P_2 must be kept in safe sequence

Step 3

Work = $\overset{7,5,5}{Work} + \overset{3,0,2}{Allocation_2}$

Work = 10 | 5 | 7 (A B C)

Finish =

0	1	2	3	4
true	true	true	true	true

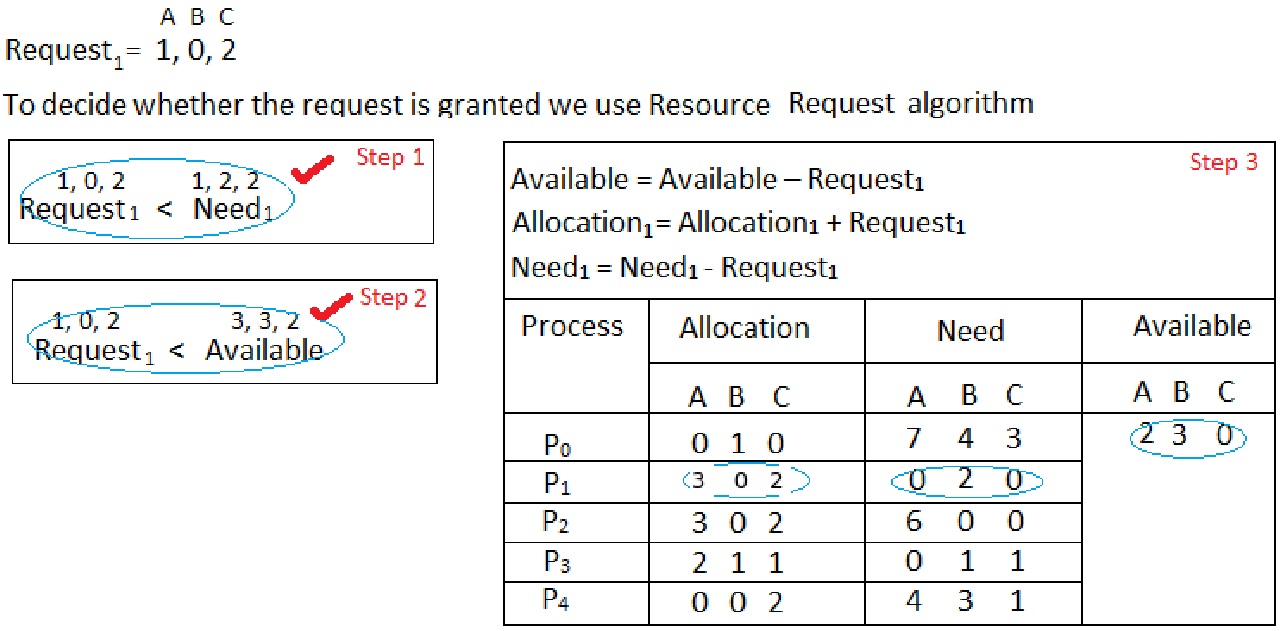
Step 4

Finish [i] = true for 0 ≤ i ≤ n

Hence the system is in Safe state

The safe sequence is P_1, P_3, P_4, P_0, P_2

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



$Request_1 = \overset{A\ B\ C}{1, 0, 2}$

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

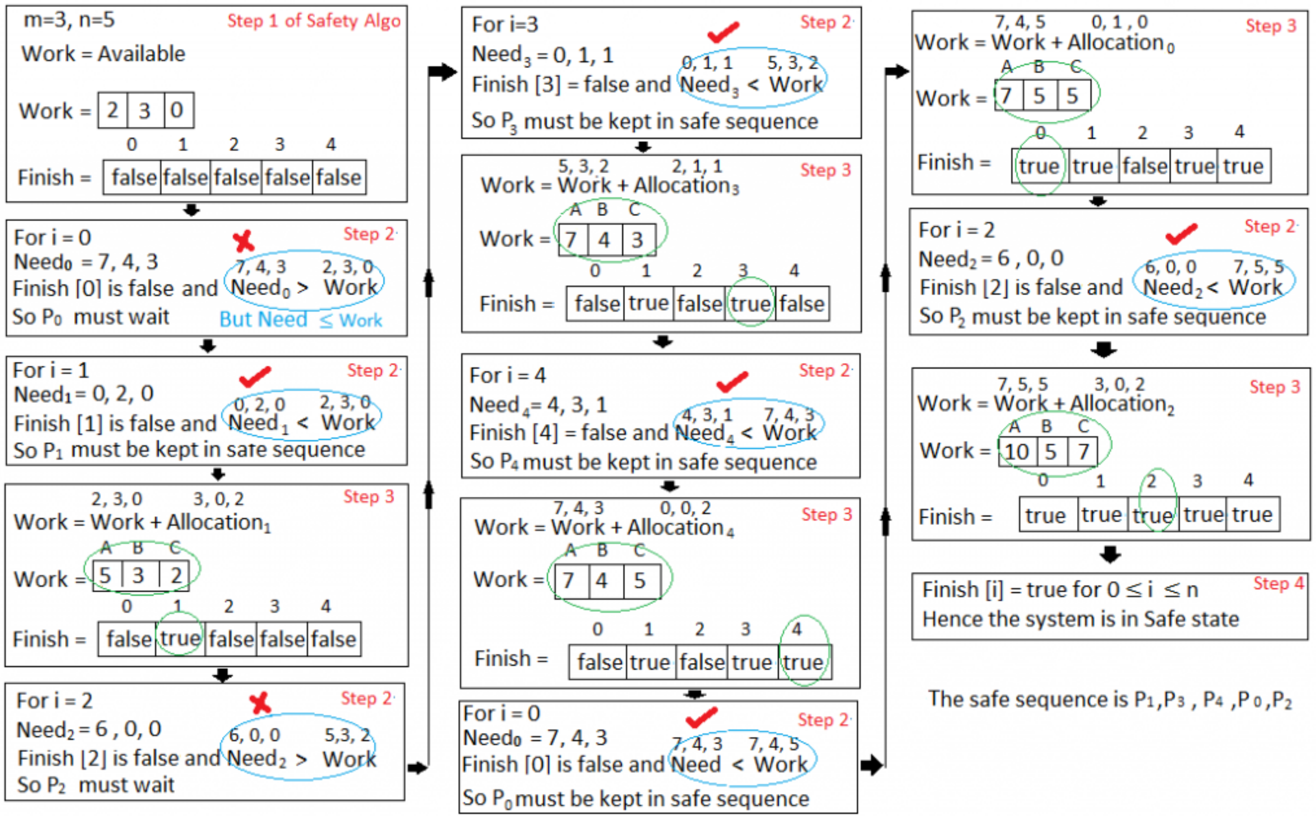
Available = Available – $Request_1$

$Allocation_1 = Allocation_1 + Request_1$

$Need_1 = Need_1 - Request_1$

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.



m=3, n=5
Step 1 of Safety Algo
Work = Available
Work = 2 3 0
0 1 2 3 4
Finish = false false false false false
For i = 0
Step 2
Need₀ = 7, 4, 3
7, 4, 3 2, 3, 0
Finish [0] is false and Need₀ > Work
So P₀ must wait
But Need ≤ Work
For i = 1
Step 2
Need₁ = 0, 2, 0
0, 2, 0 2, 3, 0
Finish [1] is false and Need₁ < Work
So P₁ must be kept in safe sequence
2, 3, 0 3, 0, 2
Step 3
Work = Work + Allocation₁
A B C
Work = 5 3 2
0 1 2 3 4
Finish = false true false false false
For i = 2
Step 2
Need₂ = 6 , 0, 0
6, 0, 0 5,3, 2
Finish [2] is false and Need₂ > Work
So P₂ must wait
For i=3
Step 2
Need₃ = 0, 1, 1
0, 1, 1 5, 3, 2
Finish [3] = false and Need₃ < Work
So P₃ must be kept in safe sequence
5, 3, 2 2, 1, 1
Step 3
Work = Work + Allocation₃
A B C
Work = 7 4 3
0 1 2 3 4
Finish = false true false true false
For i = 4
Step 2
Need₄ = 4, 3, 1
4, 3, 1 7, 4, 3
Finish [4] = false and Need₄ < Work
So P₄ must be kept in safe sequence
7, 4, 3 0, 0, 2
Step 3
Work = Work + Allocation₄
A B C
Work = 7 4 5
0 1 2 3 4
Finish = false true false true true
For i = 0
Step 2
Need₀ = 7, 4, 3
7, 4, 3 7, 4, 5
Finish [0] is false and Need < Work
So P₀ must be kept in safe sequence
7, 4, 5 0, 1 , 0
Step 3
Work = Work + Allocation₀
A B C
Work = 7 5 5
0 1 2 3 4
Finish = true true false true true
For i = 2
Step 2
Need₂ = 6 , 0, 0
6, 0, 0 7, 5, 5
Finish [2] is false and Need₂ < Work
So P₂ must be kept in safe sequence
7, 5, 5 3, 0, 2
Step 3
Work = Work + Allocation₂
A B C
Work = 10 5 7
0 1 2 3 4
Finish = true true true true true
Finish [i] = true for 0 ≤ i ≤ n
Step 4
Hence the system is in Safe state
The safe sequence is P₁,P₃ , P₄ ,P₀ ,P₂

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

Gate question:

http://quiz.geeksforgeeks.org/gate-gate-cs-2014-set-1-question-41/

Reference:

Operating System Concepts 8th Edition by Abraham Silberschatz, Peter B. Galvin, Greg Gagne

This article has been contributed by ~~Vikash Kumar~~. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GATE CS Corner

See Placement Course for placement preparation, GATE Corner for GATE CS Preparation and Quiz Corner for all Quizzes on GeeksQuiz.

Category: Operating Systems

(Login to Rate and Mark)

2.6 Average Difficulty : **2.6/5.0**

Based on **5** vote(s)

Average Rating : **5/5.0**

Based on **1** vote(s)

★★★★★

□ Add to TODO List

□ Mark as DONE

Load Comments