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?

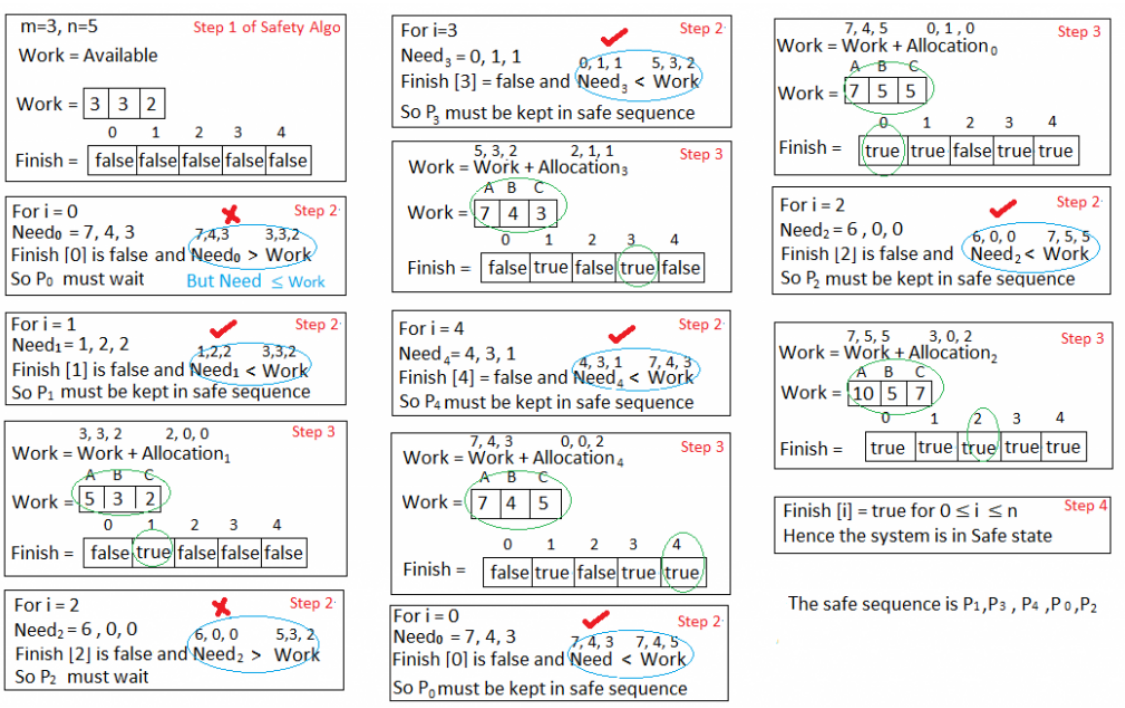
Need [i, j] = Max [i, j] – Allocation [i, j]

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?

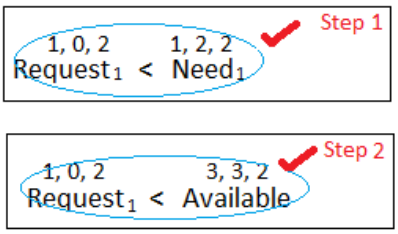
Applying the Safety algorithm on the given system,



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

A B C
$Request_1$ = 1, 0, 2

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

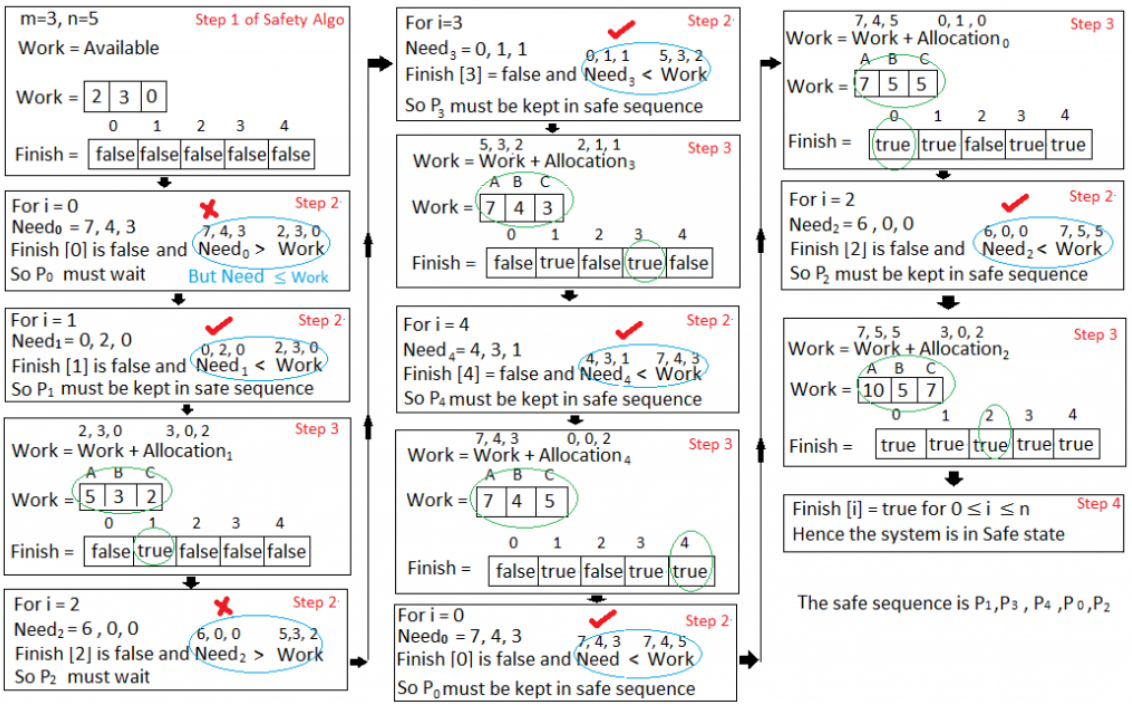


Step 3

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.

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 **Company Wise Coding Practice**

Operating Systems Process Synchronization Login to Improve this Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Recommended Posts:

Program for Banker's Algorithm | Set 1 (Safety Algorithm)
Priority Inversion : What the heck !
Operating System | Process Management | Deadlock Introduction
What's difference between Priority Inversion and Priority Inheritance ?
Readers-Writers Problem | Set 1 (Introduction and Readers Preference Solution)
Operating System | Introduction of Operating System – Set 1
Operating System | Process-based and Thread-based Multitasking
Operating System | Difference between multitasking, multithreading and multiprocessing
Operating System | Reader-Writers solution using Monitors
IPC through shared memory

(Login to Rate)

3.3 Average Difficulty : **3.3/5.0**
Based on **21** vote(s)

Basic Easy Medium Hard Expert

Add to TODO List
Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments Share this post!