Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

Banker’s Algorithm for Deadlock Avoidance

References

1. Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, Operating System Concepts, WILEY,ISBN 978-1-118-06333-0, 9th Edition

Banker’s Algorithm

Aim:

To implement Bankers Algorithm for deadlock avoidance using

(a) Safety algorithm
(b) resource request algorithm

Objectives:

To understand the concept of deadlock.
How to avoid deadlock by implementing safety algorithm.

Data Structures for the Banker's Algorithm

Let *n* = number of processes, and *m* = number of resources types.

Available: Vector of length *m*. If available [*j*] = *k*, there are *k* instances of resource type R_j available.

Max: *n x m* matrix. If *Max* [*i*,*j*] = *k*, then process P_i may request at most *k* instances of resource type R_j.

Allocation: *n x m* matrix. If Allocation[*i*,*j*] = *k* then P_i is currently allocated *k* instances of $R_{j.}$

Need: *n x m* matrix. If *Need*[*i*,*j*] = *k*, then P_i may need *k* more instances of R_j to complete its task.

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

Safety Algorithm

To determine whether system is in safe state

1. Let *Work* and *Finish* be vectors of length m and n, respectively. Initialize:

    Work := *Available*

    Finish [i] = *false* for $i = 1,2, \ldots, n$.

2. Find an i such that both:

    (a) *Finish* [i] = *false*

    (b) $Need_i \leq Work$

    If no such i exists, go to step 4.

3. $Work := Work + Allocation_i$
 Finish[i] := *true*
 go to step 2.

4. If *Finish* [i] = true for all i, then the system is in a safe state.

Resource-Request Algorithm

$Request_i$ = request vector for process P_i.

If $Request_i[j] = k$ then process P_i wants k instances of resource type R_j.

1. If $Request_i \leq Need_i$ go to step 2. Otherwise, raise error condition, since process has exceeded its maximum claim.

2. If $Request_i \leq Available$, go to step 3. Otherwise P_i must wait, since resources are not available.

3. Pretend to allocate requested resources to P_i by modifying the state as follows:

   $Available := Available - Request_i;$

   $Allocation_i := Allocation_i + Request_i;$

   $Need_i := Need_i - Request_{i;;}$

- *Call safety algorithm*
- *If safe $\Rightarrow$ the resources are allocated to P_i.*
- *If unsafe $\Rightarrow$ P_i must wait, and the old resource-allocation state is restored*

Example of Banker's Algorithm

5 processes P_0 through P_4; 3 resource types A (10 instances), B (5 instances), and C (7 instances).

Snapshot at time T_0:

	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	

Example (Cont.)

The content of the matrix. Need is defined to be Max – Allocation.

	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

The system is in a safe state since the sequence $< P_1, P_3, P_4, P_0, P_2 >$ or $< P_1, P_3, P_4, P_2, P_0 >$ satisfies safety criteria.

m=3, n=5 — Step 1 of Safety Algo

Work = Available

Work =

3	3	2

	0	1	2	3	4
Finish =	false	false	false	false	false

For i = 0 — Step 2

$Need_0$ = 7, 4, 3

Finish [0] is false and $Need_0 > Work$ (7,4,3 > 3,3,2)

So P_0 must wait — But Need ≤ Work

For i = 1 — Step 2

$Need_1$ = 1, 2, 2

Finish [1] is false and $Need_1 < Work$ (1,2,2 < 3,3,2)

So P_1 must be kept in safe sequence

Step 3

Work = Work + $Allocation_1$ (3, 3, 2 + 2, 0, 0)

A	B	C
5	3	2

	0	1	2	3	4
Finish =	false	true	false	false	false

For i = 2 — Step 2

$Need_2$ = 6 , 0, 0

Finish [2] is false and $Need_2 > Work$ (6, 0, 0 > 5,3, 2)

So P_2 must wait

For i=3 — Step 2

$Need_3$ = 0, 1, 1

Finish [3] = false and $Need_3 < Work$ (0, 1, 1 < 5, 3, 2)

So P_3 must be kept in safe sequence

Step 3

Work = Work + $Allocation_3$ (5, 3, 2 + 2, 1, 1)

A	B	C
7	4	3

	0	1	2	3	4
Finish =	false	true	false	true	false

For i = 4 — Step 2

$Need_4$ = 4, 3, 1

Finish [4] = false and $Need_4 < Work$ (4, 3, 1 < 7, 4, 3)

So P_4 must be kept in safe sequence

Step 3

Work = Work + $Allocation_4$ (7, 4, 3 + 0, 0, 2)

A	B	C
7	4	5

	0	1	2	3	4
Finish =	false	true	false	true	true

For i = 0 — Step 2

$Need_0$ = 7, 4, 3

Finish [0] is false and Need < Work (7, 4, 3 < 7, 4, 5)

So P_0 must be kept in safe sequence

Step 3

Work = Work + $Allocation_0$ (7, 4, 5 + 0, 1 , 0)

A	B	C
7	5	5

	0	1	2	3	4
Finish =	true	true	false	true	true

For i = 2 — Step 2

$Need_2$ = 6 , 0, 0

Finish [2] is false and $Need_2 < Work$ (6, 0, 0 < 7, 5, 5)

So P_2 must be kept in safe sequence

Step 3

Work = Work + $Allocation_2$ (7, 5, 5 + 3, 0, 2)

A	B	C
10	5	7

	0	1	2	3	4
Finish =	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

Example (Cont.)……

At time T_1:

P_1 request (1,0,2)

Resource-Request algorithm

$$\text{Request}_1 = \overset{A}{1}, \overset{B}{0}, \overset{C}{2}$$

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

Step 1

$$\overset{1,\,0,\,2}{\text{Request}_1} < \overset{1,\,2,\,2}{\text{Need}_1}$$

Step 2

$$\overset{1,\,0,\,2}{\text{Request}_1} < \overset{3,\,3,\,2}{\text{Available}}$$

Step 3

$\text{Available} = \text{Available} - \text{Request}_1$

$\text{Allocation}_1 = \text{Allocation}_1 + \text{Request}_1$

$\text{Need}_1 = \text{Need}_1 - \text{Request}_1$

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	

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

Work = Available

Work =

2	3	0

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 $Need_0 > Work$ (7, 4, 3 > 2, 3, 0)

So P_0 must wait

But Need ≤ Work

For i = 1 Step 2

$Need_1$ = 0, 2, 0

Finish [1] is false and $Need_1 < Work$ (0, 2, 0 < 2, 3, 0)

So P_1 must be kept in safe sequence

Work = Work + $Allocation_1$ (2, 3, 0 + 3, 0, 2) Step 3

Work =

A	B	C
5	3	2

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 $Need_2 > Work$ (6, 0, 0 > 5,3, 2)

So P_2 must wait

For i=3 Step 2

$Need_3$ = 0, 1, 1

Finish [3] = false and $Need_3 < Work$ (0, 1, 1 < 5, 3, 2)

So P_3 must be kept in safe sequence

Work = Work + $Allocation_3$ (5, 3, 2 + 2, 1, 1) Step 3

Work =

A	B	C
7	4	3

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 $Need_4 < Work$ (4, 3, 1 < 7, 4, 3)

So P_4 must be kept in safe sequence

Work = Work + $Allocation_4$ (7, 4, 3 + 0, 0, 2) Step 3

Work =

A	B	C
7	4	5

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 Need < Work (7, 4, 3 < 7, 4, 5)

So P_0 must be kept in safe sequence

Work = Work + $Allocation_0$ (7, 4, 5 + 0, 1 , 0) Step 3

Work =

A	B	C
7	5	5

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 $Need_2 < Work$ (6, 0, 0 < 7, 5, 5)

So P_2 must be kept in safe sequence

Work = Work + $Allocation_2$ (7, 5, 5 + 3, 0, 2) Step 3

Work =

A	B	C
10	5	7

Finish =

0	1	2	3	4
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 P_1, P_3, P_4, P_0, P_2

Example (Cont.): P_1 request (1,0,2)

Check that Request ≤ Available (that is, $(1,0,2) \le (3,3,2) \Rightarrow$ *true*.

	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	

Executing safety algorithm shows that sequence $<P_1, P_3, P_4, P_0, P_2>$ satisfies safety requirement.

Example (Cont.)

At time T_2: Can request for (3,3,0) by P_4 be granted?

At time T_3: Can request for (0,2,0) by P_0 be granted?

Steps for implementation

1. Accept no of processes
2. Accept no of resource types
3. Accept available, max, allocation matrices
4. Determine need matrix
5. Invoke safety algorithm and display the safety sequence
6. If system is in safe state, accept resource request from user
7. Invoke resource-request algorithm
8. Display whether request can be granted and display safe sequence if any

Steps for implementation

1. Accept no of processes
2. Accept no of resource types
3. Accept available, max, allocation matrices
4. Determine need matrix
5. Invoke safety algorithm
6. If safe, accept resource request from user
7. Invoke resource-request algorithm
8. Display whether request can be granted and display safe sequence if any

output

Following is the SAFE Sequence

P1 -> P3 -> P4 -> P0 -> P2