Banker’s Algorithm

Name : Najway Dhu Abraheem Alhajji
Student No :163103036
Department : Electrical and Computer Engineering

Banker’s Algorithm

The **Banker's algorithm**, sometimes referred to as the **avoidance algorithm**, is a resource allocation and deadlock avoidance algorithm developed by” Edsger Dijkstra” that tests for safety by simulating the allocation of predetermined maximum possible amounts of all resources, and then makes an "s-state" check to test for possible deadlock conditions for all other pending activities, before deciding whether allocation should be allowed to continue.

Banker's Algorithm

For the Banker's algorithm to work, it needs to know three things:

- How much of each resource each process could possibly request[MAX]
- How much of each resource each process is currently holding[ALLOCATED]
- How much of each resource the system currently has available[AVAILABLE]

Resources may be allocated to a process only if it satisfies the following conditions:

request $\leq$ available, else process waits until resources are available.

Data Structures for the Banker's Algorithm

Let n = number of processes, and m = number of resources types.] = k then Pi is currently allocated k instances of Rj.

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

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

■ **Allocation:** n x m matrix. If Allocation[i,j

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

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

Safety Algorithm

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 and *i such that both:*
   (a) *Finish [i] = false*
   (b) *Needi <= Work*
   If no such *i exists, go to step 4.*
3. *Work = Work + Allocationi*
   Finish[i] = true go to step 2.
4. If *Finish [i] == true for all i, then the system is in a safe state.*

Example of Banker's Algorithm

- 5 processes : P0 –P4;and 3 resource types(A,B,C)
- A(10 instances), B (5 instances), C (7 instances)
- Snapshot at time T0

* av "A"=A – sum of allocation

*av "A" =10-7 =3

*av "B" =5-2=3

*av "C"=7-5=2

Available=332

	Max			Allocation			Available		
	A	B	C	A	B	C	A	B	C
P0	7	5	3	0	1	0			
P1	3	2	2	2	0	0			
P2	9	0	2	3	0	2			
P3	2	2	2	2	1	1			
P4	4	3	3	0	0	2			

Example

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

	Need		
	A	B	C
P0	7	4	3
P1	1	2	2
P2	6	0	0
P3	0	1	1
P4	4	3	1

Applying the Safety algorithm on the given system

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
Work =	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
Work =	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
Work =	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
Work =	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
Work =	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 Safe State

	Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	4	3	7	5	5
P1	2	0	0	1	2	2	5	3	2
P2	3	0	2	6	0	0	10	5	7
P3	2	1	1	0	1	1	7	4	3
P4	0	0	2	4	3	1	7	4	5

Example

The system is in a safe state since the sequence *<P1,P3,P4,P0,P2> satisfies safety criteria.*

Example unsafe

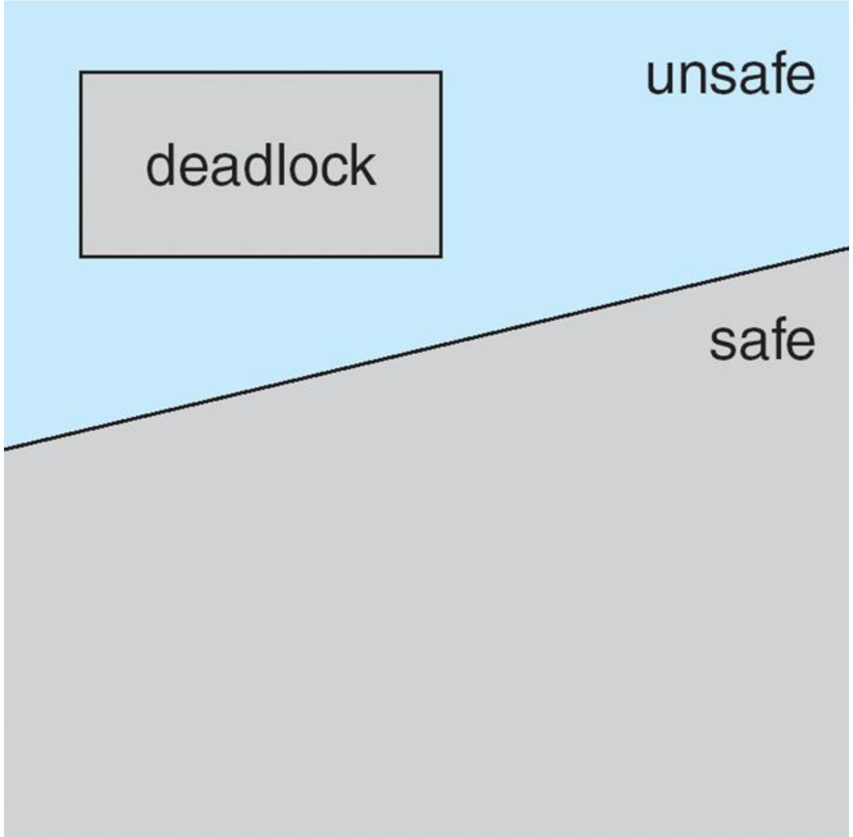
Can request for (3,3,0) by P4 be granted

	Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	4	3			
P1	2	0	0	1	2	2			
P2	3	0	2	6	0	0			
P3	2	1	1	0	1	1			
P4	~~0~~ 3	~~0~~ 3	~~2~~ 2	1	0	2			

* av "A"=A - sum of allocation
*av "A" =10-10 =0
*av "B" =5-5=0
*av "C"=7-5=2
Available=002

The system is in unsafe state

Safe, Unsafe, Deadlock State



THANK YOU