Bankers Algorithms

Ref: https://www.geeksforgeeks.org/bankers-algorithm-in-operating-system-2/

Problem Statement

Write a program to implement Banker's Algorithm for deadlock handling.

Input Required

- Number of processes in the system (**n**)
- The number of resources types (**m**)
- **Maximum Resources**: the number of available resources of each type.
  - It is a **1-d array** of size **m**
- **Max**: maximum demand of each process in a system.
  - It is a **2-d array** of size **n*m**
- **Allocation**: the number of resources of each type currently allocated to each process.
  - It is a **2-d array** of size **n*m**

Calculations

- **Need**: indicates the remaining resource need of each process.
  - It is a **2-d array** of size **n*m**
  - **Need [i, j] = Max [i, j] – Allocation [i, j]**
- **Allocated:** Total number of allocated resources of each type. Addition of resource type column wise from Allocation array.
  - It is a 1**-d array** of size **m**
- **Work/Available:**
  - It is a 1**-d array** of size **m**
  - **Work[i]=Available[i] - Allocated[i]**

Expected Output

- One Safe Sequence

- Variations:
  - Mutliple Safe Sequence
  - Resource Request for Process

Example: Sample Input

- Number of processes, n = 5 (P0, P1, P2, P3, P4)
- Number of resources types, m = 3 (A, B, C)
- **Maximum Resources**: A has 10 instances, B has 5 instances and C has 7 instances.
- **Maximum Resources = [10, 5, 7]**

Process	Allocation			Max		
	A	B	C	A	B	C
P_0	0	1	0	7	5	3
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: Calculations

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

- Allocated = [7, 2, 5]
- Work/Available =

  [3, 3, 2] =

  [10, 5, 7] – [7, 2, 5]

Example: Sample Output

	Max			Allocation			Need		
	A	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**
P0	7	5	3	0	1	0	7	4	3
P1	3	2	2	2	0	0	1	2	2
P2	9	0	2	3	0	2	6	0	0
P3	2	2	2	2	1	1	0	1	1
P4	4	3	3	0	0	2	4	3	1

Maximun Resources = [10, 5, 7]   Available Resources = [3, 3, 2]

	Available		
	A	**B**	**C**
Initially Work/Available	3	3	2
After exexution of P1	5	3	2
After exexution of P3	7	4	3
After exexution of P4	7	4	5
After exexution of P0	7	5	5
After exexution of P2	10	5	7

The Safe Sequence is P1, P3, P4, P0, P2

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, 3, 4….

2. Find an i such that both

   a) Finish[i] = false

   b) $Need_i$ <= 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 a safe state

Working: Safety Algorithm

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

Step 2

For i = 0

$Need_0$ = 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

Step 2

For i = 1

$Need_1$ = 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)

Work =

A	B	C
5	3	2

Finish =

0	1	2	3	4
false	true	false	false	false

Step 2

For i = 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

Step 2

For i=3

$Need_3$ = 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)

Work =

A	B	C
7	4	3

Finish =

0	1	2	3	4
false	true	false	true	false

Step 2

For i = 4

$Need_4$ = 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)

Work =

A	B	C
7	4	5

Finish =

0	1	2	3	4
false	true	false	true	true

Step 2

For i = 0

$Need_0$ = 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)

Work =

A	B	C
7	5	5

Finish =

0	1	2	3	4
true	true	false	true	true

Step 2

For i = 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)

Work =

A	B	C
10	5	7

Finish =

0	1	2	3	4
true	true	true	true	true

Step 4

Finish [i] = true for $0 \leq i \leq n$

Hence the system is in Safe state

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

Resource-Request Algorithm

1. If $Request_i <= Need_i$

   Goto step (2) ; otherwise, raise an error condition, since the process has exceeded its maximum claim.

2. If $Request_i <= Available$

   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:

   a) $Available = Available – Request_i$

   b) $Allocation_i = Allocation_i + Request_i$

   c) $Need_i = Need_i – Request_i$

What will happen if process P1 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 1

1, 0, 2   1, 2, 2

$Request_1 < Need_1$ ✔

Step 2

1, 0, 2   3, 3, 2

$Request_1 < Available$ ✔

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	

Determine whether this new system state is safe? To do so, we again execute Safety algorithm on the above data structures.

Step 1 of Safety Algo

m=3, n=5

Work = Available

Work =

2	3	0

Finish =

0	1	2	3	4
false	false	false	false	false

Step 2

For i = 0

$Need_0$ = 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

Step 2

For i = 1

$Need_1$ = 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

Step 3

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

Work =

A	B	C
5	3	2

Finish =

0	1	2	3	4
false	true	false	false	false

Step 2

For i = 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

Step 2

For i=3

$Need_3$ = 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)

Work =

A	B	C
7	4	3

Finish =

0	1	2	3	4
false	true	false	true	false

Step 2

For i = 4

$Need_4$ = 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)

Work =

A	B	C
7	4	5

Finish =

0	1	2	3	4
false	true	false	true	true

Step 2

For i = 0

$Need_0$ = 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)

Work =

A	B	C
7	5	5

Finish =

0	1	2	3	4
true	true	false	true	true

Step 2

For i = 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)

Work =

A	B	C
10	5	7

Finish =

0	1	2	3	4
true	true	true	true	true

Step 4

Finish [i] = true for $0 \le i \le n$

Hence the system is in Safe state

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

Hence the new system state is safe, so we can immediately grant the request for process P1 .

Thank You!!