Example of Banker's Algorithm

To illustrate the use of Banker's Algorithm, consider a system with five processes P0 through P5 and three resource types A, B and C. Resource type A has ten instances, resource type B has five instances and resource type C has seven instances.

Suppose that, at time T0, the following snapshot of the system has been taken : A = 10, B = 5 & C = 7

Process	Allocation A B C	Max Need A B C	Available A B C	Remaining Need A B C
P1	0 1 0	7 5 3	3 3 2 +2 0 0	7 4 3
P2	2 0 0	3 2 2	5 3 2	1 2 2
P3	3 0 2	9 0 2	+2 1 1	6 0 0
P4	2 1 1	4 2 2	7 4 3	2 1 1
P5	0 0 2	5 3 3	+0 +0 +2	5 3 1
	7 2 5		7 4 5	

Calculate available resource i.e. Total resources − No. of resources allocated to all processes.

Available resources A = 10 − 7 = 3
Available resources B = 5 − 2 = 3
Available resources C = 7 − 5 = 2

Calculate Remaining need of resources = ~~Maxium~~ Maximum Need of resources − allocated resources

for eg. for process P1, Remaining need of resource A = 7 − 0 = 7

Date
Page

Remaining need of resource $B = 5-1 = 4$

Remaining need of resource $C = 3-0 = 3$

(1) Now we will check that whether we can fullfill the remaining need of any process with the help of available number of resources A, B and C.

(2) For process P_1 we require 7 instance of resource A but we have only 3 instance available for resource A. Similarly we require 4 instances of resource B but we have only 3. We also require 3 instances of resource C but have 2 instances of resource C available with us. So I can full fill remaining need of only resource C means process P_1 will not be executed.

(3) Now check for process P_2. Here process P_2 remaining need for resources A, B and C is less than respective available instances of resources A, B and C. Means need of process P_2 is fullfilled so process P_2 will get executed and resources allocated to process P_2 will be released and added to current available resources.

Now available resources $A = 5, B = 3$ & $C = 2$

(4) Now check for process P_3, remaining need for resource A can not be fullfilled with current available instances of resources A, so we have to check for process P_4.

sequence of processes	Safe
$P_2 \rightarrow P_4 \rightarrow P_5 \rightarrow P_1 \rightarrow P_3$	sequence

(5) For process P_4, remaining need for resources A, B and C can be fullfilled by current available instances of resources A, B and C. Now process P_4 can be executed and ~~relase~~ released the resources allocated to process P_4. So now available resources A = 7, B = 4 & C = 3

(6) Now check for process P_5, remaining need for resource A, B & C can be fullfilled by current available instances of resource A, B and C. So Process P_5 will be executed and release the resources allocated to process P_5. Now available resources A = 7, B = 4 and C = 5.

(7) Now we can check for process P_1 or P_3. Only sequence will change. If we consider process P_1 then we can fullfill the remaining need of resources A, B & C. So process P_1 will be executed and release allocated resources. Now available resources A = 7+0 = 7, B = 4+1 = 5 and C = 5+0 = 5.

(8) For process P3, we can fullfill remaining need so process P_3 will be executed and it will release allocated resources. Now current available resources A = 7+3 = 10, B = 5+0 = 5 and C = 5+2 = 7

(9) When all processes executed total no. of resource A = 10, B = 5 and C = 7 are available. As all processes executed there is no deadlock condition.

Banker's Algorithm Example Solutions

1) Assume that there are 5 processes, P_0 through P_4 and 4 types of resources. At T_0 we have the following system state:

Max Instances of Resource Type A = 3 [2 allocated + 1 Available]

Max Instances of Resource Type B = 17 [12 allocated + 5 Available]

Max Instances of Resource Type C = 16 [14 allocated + 2 available]

Max Instances of Resource Type D = 12 [12 allocated + 0 Available]

	Given Matrices												Need Matrix = Max Matrix-Allocation			
	Allocation				Max Matrix				Available matrix							
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	1	1	0	0	2	1	0	1	5	2	0	0	1	0	0
P_1	1	2	3	1	1	6	5	2	1	6	3	0	0	4	2	1
P_2	1	3	6	5	2	3	6	6	1	12	6	2	1	0	0	1
P_3	0	6	3	2	0	6	5	2	1	12	7	6	0	0	2	0
P_4	0	0	1	4	0	6	5	6	2	14	10	7	0	6	4	2
Total	2	12	14	12					3	17	16	12				

1. Create the need Matrix (Max allocation)

Need (i) = Max(i) - Allocated (i)

(i=0) ⇒ (0,2,1,0) − (0,1,1,0) = (0,1,0,0)

(i=1) ⇒ (1,6,5,2) − (1,2,3,1) = (0,4,2,1)

(i=2) ⇒ (2,3,6,6) − (1,3,6,5) = (1,0,0,1)

(i=3) ⇒ (0,6,5,2) − (0,6,3,2) = (0,0,2,0)

(i=4) ⇒ (0,6,5,6) − (0,0,1,4) = (0,6,4,2)

2) Use the safety algorithm to test if the system is in a safe state or not

ⓐ We will first define work & finish:

Initially work = available = (1,5,2,0)
Finish = False for all processes

Finish	Matrix
P_0	False
P_1	False
P_2	False
P_3	False
P_4	False

Work vector			
1	5	2	0

ⓑ Check the needs of each process [needs $(P_i) <= \max(P_i)$], if this condition is true: Ⓘ Execute the process, change Finish [i] = True.
Ⓘ Release the allocated Resources by this process
Ⓘ Change the work variable = Allocated (P_i) + work

need0 (0,1,0,0) ≤ work (1,5,2,0)
p0 will be executed because need (p0) < work ⇒ P0 will be True

Finish Matrix	
P_0 - I	True
P_1	False
P_2	False
P_3	False
P_4	False

P_0 will release the allocated resources (0,1,1,0)
∴ Work = work (1,5,2,0) + Allocated (P_0)
Vector = (1,5,2,0) + (0,1,1,0)

∴ Work Vector

1	6	3	0

Need1 (0,4,2,1) <= work (1,6,3,0) condition is false so P1 will not be executed.

Need2 (1,0,0,1) <= work (1,6,3,0) condition is false so P2 will not be executed.

Need3 (0,0,2,0) <= work (1,6,3,0) condition is True so P3 will be executed

Finish Matrix	
P0-1	True
P_1	False
P_2	False
P3-2	~~P3~~ True
P_4	False

P3 will release the allocated resources (0,6,3,2)

∴ work vector = work (1,6,3,0) + Allocated P3

= (1,6,3,0) + (0,6,3,2)

= (1,12,6,2)

work vector

1	12	6	2

Need 4 (0,6,4,2) <= work ~~(1,6,3,0)~~ (1,12,6,2)

condition is true so P4 ... executed

P4 will release the allocated resources (0,0,1,4)

work vector = Work (1,12,6,2) + Allocated P4

= (1,12,6,2) + (0,0,1,4)

= 1,12,7,6

work vector

1	12	7	6

Finish Matrix	
P0 - 1	True
P1	False
P2	False
P3 - 2	True
P4 - 3	True

Need 1 (0,4,2,1) <= work (1,12,7,6)

condition is True so P_1 will be executed.

P_1 will release the allocated resources (1,2,3,1)

∴ Work vector = work (1,12,7,6) + Allocated P_1

= (1,12,7,6) + (1,2,3,1)

= (2,14,10,7)

Work vector

2	14	10	7

Finish Matrix	
P0 - 1	True
P1 - 4	True
P2	False
P3 - 2	True
P4 - 3	True

Need2 (1,0,0,1) <= work (2,14,10,7)
condition is true, so P2 will be executed.
P2 will release the allocated resources
(1,3,6,5)

∴ work vector = work (2,14,10,7) + allocated P2

= (2,14,10,7) + (1,3,6,5)

= (3,17,16,12)

Work vector

3	17	16	12

Finish Matrix	
P_0 - 1	True
P_1 - 4	True
P_2 - 5	True
P_3 - 2	True
P_4 - 3	True

The system is in a safe state and the processes will be executed in the following order

P0, P3, P4, P1, P2 called as Safe sequence.