Deadlock Avoidance (Banker's Algorithm)

	Allocation			Max			Need			Available		
	A	B	C	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	7	4	3	3	3	2
P_1	2	0	0	3	2	2	1	2	2			
P_2	3	0	2	9	0	2	6	0	0			
P_3	2	1	1	2	2	2	0	1	1			
P_4	0	0	2	4	3	3	4	3	1			

$n=5$, $m=3$ (A, B, C) ; Need = (Max − Allocation)

$\boxed{\text{Total} \rightarrow 10 \quad 5 \quad 7}$

Is the system safe? →

work = available = (3 3 2)

Finish = (false, false, false, false, false)

1. Finish[1] = false, $Need_1 = (1\ 2\ 2) \leq (3\ 3\ 2)$

work = work + $allocation_1$ = (3 3 2) + (2 0 0)
= (5 3 2)

Finish[1] = true

2. Finish[3] = false, $Need_3 = (0\ 1\ 1) \leq (5\ 3\ 2)$

Finish[3] = true;

work = work + $allocation_3$ = (5 3 2) + (2 1 1)
= (7 4 3)

3. Finish[4] = false, $Need_4 = (4\ 3\ 1) \leq (7\ 4\ 3)$

Finish[4] = true;

$work = work + allocation_4 = (7\ 4\ 3) + (0\ 0\ 2)$

$= (7\ 4\ 5)$

4. Finish[2] = false; $Need_2 = (6\ 0\ 0) \leq (7\ 4\ 5)$

Finish[2] = true;

$work = work + allocation_2$

$= (7\ 4\ 5) + (3\ 0\ 2)$

$= (10\ 4\ 7)$

5. Finish[0] = false; $Need_0 = (7\ 4\ 3) \leq (10\ 4\ 7)$

Finish[0] = true;

$work = work + allocation_0$

$= (10\ 4\ 7) + (0\ 1\ 0)$

$= (10\ 5\ 7)$

$\therefore$ Safe with sequence

$\langle P_1, P_3, P_4, P_2, P_0 \rangle$

Banker's Algorithm

	Alloc			Max			Need			Available		
	A	B	C	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	7	4	3	3	3	2
P_1	2	0	0	3	2	2	1	2	2			
P_2	3	0	2	9	0	2	6	0	0			
P_3	2	1	1	2	2	2	0	1	1			
P_4	0	0	2	4	3	3	4	3	1			

Let P_4 has request $\rightarrow (3\ 3\ 0)$

$Request_4 = (3\ 3\ 0)$

$Need_4 = (4\ 3\ 1)$

① $Request_4 \leq Need_4$ ✓

② $Request_4 \leq available$

$(3\ 3\ 0) \leq (3\ 3\ 2)$ ✓

③ $Available = Available - Request_4$

$= (3\ 3\ 2) - (3\ 3\ 0)$

$= (0\ 0\ 2)$

④ $Allocation_4 = Allocation_4 + Request_4$

$= (0\ 0\ 2) + (3\ 3\ 0)$

$= (3\ 3\ 2)$

⑤ $Need_4 = Need_4 - Request_4$

$= (4\ 3\ 1) - (3\ 3\ 0)$

$= (1\ 0\ 1)$

Now, the current state of the system $\longrightarrow$

	Allocation			Max			Need			Available		
	A	B	C	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	7	4	3	0	0	2
P_1	2	0	0	3	2	2	1	2	2			
P_2	3	0	2	9	0	2	6	0	0			
P_3	2	1	1	2	2	2	0	1	1			
P_4	3	3	2	4	3	3	1	0	1			

Is the above resource-allocation state safe?

① work = available = (0 0 2)

② Find 'i' such that Finish[i] = false and $Need_i \le work$

could not find such 'i'.

→ System is unsafe.

→ P_4 has to wait.

→ Restore old state.